BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INFORMATION SYSTEMS
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

RATING LOG EVENTS USING REPUTATION AND ANOMALY SCORES

VYUŽITÍ REPUTAČNÍHO SKÓRE A SKÓRE ANOMÁLIÍ PRO HODNOCENÍ UDÁLOSTÍ V LOGOVACÍCH SOUBORECH

MASTER'S THESIS
DIPLOMOVÁ PRÁCE

AUTHOR **Bc. JAN ZBOŘIL**
AUTOR PRÁCE

SUPERVISOR **doc. Ing. PETR MATOUŠEK, Ph.D., M.A.**
VEDOUCÍ PRÁCE

BRNO 2024

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Master's Thesis Assignment

154624

Institut:	Department of Information Systems (DIFS)
Student:	**Zbořil Jan, Bc.**
Programme:	Information Technology and Artificial Intelligence
Specialization:	Computer Networks
Title:	**Rating Log Events using Reputation and Anomaly Scores**
Category:	Networking
Academic year:	2023/24

Assignment:

1. Get familiar with methods for computing reputation and anomaly scores of network nodes. Based on your literature research and log files observation, choose log event attributes which are suitable for computing these scores.
2. Based on your supervisor's recommendation, analyze available log datasets. Pre-process log events into the form suitable for score computation.
3. Propose and implement functions for calculating an anomaly score and a reputation score of network nodes. Select a way how to set attribute weights in the scoring function.
4. Implement a reputation system with long-term computation of a reputation score and an anomaly score based on retrieved log events.
5. Validate your results with respect to other reputation systems, e.g., CESNET Nerd, Cisco Talos, etc.
6. Demonstrate on real data how calculated scores can be employed in log events rating in order to reduce false alarms.

Literature:

- Dulaunoy, A., Wagener, G., Iklody, A., Mokaddem, S., & Wagner, C. An indicator scoring method for misp platforms. In *The Networking Conference TNC* (Vol. 18), 2018.
- Bartoš, V., Žádník, M, Habib, E.S., Vasilomanolakis, E: Network entity characterization and attack prediction, Future Generation Computer Systems, Volume 97, 2019, ISSN 0167-739X.
- Bartoš, V.: Reputation score, dokumentace k systému NERD dostupná na URL https://github.com/CESNET/NERD/wiki/Reputation-score [srpen 2023].
- Henriques, J.; Caldeira, F.; Cruz, T.; Simões, P. Combining K-Means and XGBoost Models for Anomaly Detection Using Log Datasets. *Electronics,* 2020, *9*, 1164.
- Mehta, S., Kothuri, P., Garcia, D.L.: Anomaly Detection for Network Connection Logs, 2018, 1812.01941, arXiv.
- Catillo, M., Pecchia, A., Villano, U.: AutoLog: Anomaly detection by deep autoencoding of system logs, Expert Systems with Applications, Volume 191, 2022, 116263, ISSN 0957-4174.

Requirements for the semestral defence:
Point 1 to 3.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Matoušek Petr, doc. Ing., Ph.D., M.A.**
Head of Department:	Kolář Dušan, doc. Dr. Ing.
Beginning of work:	1.11.2023
Submission deadline:	17.5.2024
Approval date:	30.10.2023

Abstract

The current amount of data flowing through computer networks cannot be monitored by individuals. This data is also being saved by IDS or IPS systems to logs, which grow ever faster. The goal is thus to automatically reduce the amount of such logs, for them to contain only the most valuable information. Rating scores, such as anomaly score or a reputation scores, are valid metrics for determining whether the information (i.e., log event) is valuable or not.

The goal of this thesis is to explore the current state of methods used for anomaly detection and reputation scoring. And to propose a solution on how to use data captured in the logs of network analysers like Suricata to detect anomalies in the traffic and score network nodes. A complete solution from data processing, scoring using methods for computation of reputation score and anomaly detection, and result interpretation, is developed and demonstrated on real-world data. A way of reducing the amount of log events by using the calculated scores is demonstrated. A resulting method of combining both scores to automatically rate the log events is demonstrated and explained on examples of the real scored data. Possible future uses of the results are discussed.

Abstrakt

Pro administrátory, bezpečnostní inženýry a síťové experty je nemožné sledovat současné množství dat proudící v počítačových sítích. Komplexní systémy jako IDS nebo IPS jsou navrženy tak, aby kromě své primární funkce také ukládaly síťový provoz. Cílem této práce je automaticky redukovat počet záznamů v lozích generovaných těmito systémy tak, aby obsahovaly pouze nejdůležitější informace. Anomální a reputační skóre představují metriky pro rozhodování tohoto problému - zda je záznam v logu důležitý či nikoliv.

Cílem práce je prozkoumat současný stav metod běžně používaných pro tyto účely a navrhnout řešení, jak využít data síťových analyzátorů, jako je Suricata, k detekci anomálií v provozu a ohodnocení reputace síťových uzlů. Je vyvinuto kompletní řešení od zpracování dat, výpočtu skóre, redukce velikosti logů výběrem důležitých záznamů, a interpretace výsledků. Řešení je demonstrováno na reálných datech. Jsou diskutovány možnosti využití výsledků a použitých metod, jejich možné vylepšení a možné rozšíření v budoucích pracech.

Keywords

anomaly detection, PCA, Principal Component Analysis, reputation score, network traffic logs

Klíčová slova

detekce anomálií, PCA, Analýza hlavních komponent, reputační skóre, logy síťového provozu

Reference

ZBOŘIL, Jan. *Rating Log Events using Reputation and Anomaly Scores.* Brno, 2024. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor doc. Ing. Petr Matoušek, Ph.D., M.A.

Rozšířený abstrakt

V dnešních počítačových sítích proudí větší množství dat než kdykoliv předtím. Zatímco při dřívějších objemech dat bylo možné síťový provoz zkoumat buď manuálně, nebo s použitím jednoduchých skriptů nebo analyzátorů, dnes je nutné data agregovat, filtrovat a administrátorům zpřístupnit jen ucelené shrnující informace nebo jen ty nejdůležitější informace v plné podobě.

Síťové monitorovací systémy typu IDS nebo IPS nemohou detekovaný provoz agregovat a vytvářet statistiky bez ukládání – logování – celého provozu. Tyto logy jsou pak cenným zdrojem kompletních informací i pro jiné účely. V této práci je využité právě záznamů v lozích generovaných nástrojem Suricata.

Práce navrhuje metodu redukce logů síťového provozu pomocí kombinace metody pro detekci anomálií v síťovém provozu a metody pro výpočet reputace síťových uzlů. V teoretických kapitolách představuje problematiku anomálií v síťovém provozu, jejich dělení dle současné literatury a různé metody používané pro jejich detekci. Podrobněji je popsána Analýza hlavních komponent (Principal Component Analysis, PCA). Tato metoda se řadí mezi statistické metody pro detekci anomálií. V práci je vysvětlen její matematický základ a způsoby použití. Práce poté představuje proces převodu detekovaných anomálií na číselné skóre.

Následuje pojednání o reputačních systémech. Před samotným vysvětlením jejich funkcionality se diplomová práce věnuje pojmům důvěra a důvěryhodnost, jelikož představují základní premisu na níž se reputační systémy zakládají. Rozvedena je myšlenka použití zmíněných pojmů v on-line prostředí. Kapitola o reputačních systémech následně popisuje základní prerekvizity pro vytvoření funkčního systému, běžně používané architektury a komunikaci v rámci reputačního sytému, jež má svá opodstatněná specifika oproti jiných systémům.

Mnoho konkrétních případů využití různých metod pro detekci anomálií a výpočtu reputačního skóre je představeno v kapitole o souvisejících pracích. U každého z představených článků je metoda zhodnocena a deklarováno případné využití znalostí či inspirace daným článkem v rámci této diplomové práce.

Pro zjištění, kterou metodu použít pro výpočet skóre, a které hodnoty obsažené ve vstupních logovacích souborech jsou vhodné pro zpracování, byla vyhotovena analýza dostupného datové sady. Jedná se o data skládající se z rozšířených NetFlow záznamů ve formátu EVE JSON[1]. Data byla generována ze síťového provozu na Fakultě informačních technologií Vysokého učení technického v Brně po dobu tří měsíců na přelomu roků 2022 a 2023. V analýze bylo poukázáno na nejčastěji se vyskytující vzory v chování a zajímavé aspekty chování některých stanic (IP adres). Byly popsány a dále demonstrovány datové záznamy vhodné pro další zpracování.

Před samotným výpočtem obou skóre a ohodnocením logů proběhlo předzpracování logů ze systému Suricata tak, aby byla data vhodná pro později použité metody. Data podstoupila proces výběru vhodných dimenzí, převod z kategorických dat na numerická a normalizaci do intervalu <0,1>.

Popisu konkrétní implementace detekce anomálií pomocí metody učení bez učitele PCA a automatickému zvolení prahové hodnoty se věnuje kapitola o implementaci řešení. Byly navrženy a popsány dvě metody využívající PCA pro detekci anomálií. Jedna z nich, založená na rozdělení vstupní datové sady podle časové značky jednotlivých záznamů, byla

[1]Suricata EVE JSON format - https://docs.suricata.io/en/latest/output/eve/eve-json-output.html, accessed [2024-03-02]

dále použitá pro hodnocení logovacích záznamů podle vypočítaného skóre. V rámci této metody je popsána kompletní metodika od výběru odlehlých bodů pro pozdější ověření spolehlivosti metody, přes rozdělení datové sady na trénovací a testovací část, proces učení modelu, iterativní hledání nejlepšího prahu, až po demonstraci výsledků pro zvolené experimentální logy. Následuje popis procesu ohodnocení logovacích událostí pomocí vypočítaného skóre a demonstrace procesu redukce velikosti původních logů.

V práci je popsána implementace metody výpočtu reputačního skóre včetně všech matematických vzorců. Je popsán proces volby vhodných dat z originálních logů systému Suricata. Každý výpočet je odůvodněn - proč je využitý, a jak se jeho zahrnutí do celkového výpočtu reputačního skóre promítá do koncového skóre.

Práce je zakončená pojednáním o provedených experimentech s výslednými logovacími soubory zmenšenými o nezajímavé záznamy. K rozhodnutí, zda je záznam důležitý nebo ne, bylo rozhodnuto právě pomocí anomálního a reputačního skóre pro každou IP adresu. Do práce je zahrnuto vysvětlení průběhu obou skóre v čase a jejich vzájemné závislosti. Jsou popsány záznamy ohodnocené jako anomální spolu s odůvodněním, proč je model právě takto označil. Následuje porovnání vytvořeného řešení s již existujícími programy a návrh na budoucí možná rošíření práce či její uplatnění.

Výsledkem práce je vytvořený proces ohodnocení záznamů a redukce záznamů originálních logů systému Suricata pomocí dvou skóre pro každou zdrojovou IP adresu – anomálního a reputačního. Obě skóre se mění v čase v závislosti na síťovém provozu vykazovaném každou konkrétní IP adresou.

Rating Log Events using Reputation and Anomaly Scores

Declaration

I hereby declare that this Master's thesis was prepared as an original work by me, the author, under the supervision of doc. Ing. Petr Matoušek, Ph.D., M.A. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Jan Zbořil
May 15, 2024

Acknowledgements

For those, who were with me and no longer are.

Contents

Chapter 1

Introduction

In these days, the amount of data sent across computer networks is larger than ever. Network engineers, system administrators and security personnel can no longer manually process quantity of data, and thus rely more on various software solutions including *intrusion detection systems (IDS)*, *intrusion prevention systems (IPS)*, *next generation firewalls*, and other kinds of network analysers and threat detectors.

The use of such solutions and the computation capabilities of modern hardware enable us to develop more intriguing solutions which help the aforementioned personnel to better comprehend data presented to them, better visualize it, or show previously hidden patterns and connections in the data. This understanding in turn leads to better management, design, and security of the managed networks. The modern software solutions not only produces alerts, but also generate very informative logs in semi-standardized format. These logs can be simple records of a system operation, or detailed records showing all information the system used to perform its tasks.

This thesis utilizes the detailed traffic logs of a Suricata system[1] to perform anomaly detection and to calculate reputation rating score for IP endpoints detected in traffic flowing through the Suricata instance. The goal of this thesis is to rate events generated by Suricata based on an anomaly score and a reputation score computed from the data present in Suricata logs. The anomaly score can be used to choose only events diverging from the usual network traffic. Such selection can greatly reduce the number of events from an amount, which could not be manually inspected by a human administrator, to a feasible number of events to monitor. The analysis of this anomalous events can be made easier by matching these events with a reputation score. If an anomalous event presents itself with a good reputation score, it represents a low severity event for an administrator. However, if an anomaly event has a negative reputation, then an administrator should be concerned with such an event.

Means to reach the specified goal include an analysis of available data, its preprocessing, anomaly detection using a Principal Component Analysis method. Results of anomaly scoring are then matched against the original logs to select only the anomalous events. Reputation score is calculated by a custom method and matched to anomalous events.

This work is organized as follows. Chapter 2 explains concepts of anomalies in network traffic, why are computer scientist interested in them and what useful information they provide, together with currently known means to detect them. Methods for anomaly detection and related work in this field are presented in the chapter. Chapter 3 talks about trust

[1] Suricata IDS: https://suricata.io, accessed [2024-01-20]

and reputation in online environments, as well as in the real world. It explains differences and similarities between them and shows the principles of modern network reputation systems. Examples of real world reputation systems are shown for demonstration. Chapter 4 explores the dataset of the Suricata IDS logs available for purposes of this thesis. It points to interesting data patterns and vindicates the choices made for feature extraction. Chapter 5 talks about proposed system architecture, its capabilities and methods used to process data, detect anomalies and rate network nodes. It renders real world results obtained by applying methods to an example dataset. Experiments showing the application of achieved result are presented in Chapter 6. It discusses the achieved results, ways of utilising both the anomaly and reputation scores to score the log events, comparison of calculated reputation score to state-of-the-art systems, contribution of this thesis to the current research space, and proposes possibilities of future research in the area of anomaly detection and reputation scoring. The final Chapter 7 summarizes the work done in scope of the master's thesis.

Chapter 2

Introduction to Anomaly Detection and Scoring

The objective of this chapter is to provide the reader with an introduction to the problematic of rating nodes in a network through analysing their behaviour. It defines terms *anomaly*, and *anomaly score*. It explains methods and procedures used in the real world to detect anomalies, and which are used for anomaly scoring in this thesis. In Section 2.1, the term *anomaly* and its various understandings are explained. Section 2.2 talks about various methods used for anomaly detection.

2.1 Anomalies in network traffic

In order to detect anomalies, one must first understand the term anomaly. The term itself is not concretely defined. In the work of Fernandez et al. [20], multiple definitions are listed: Anomaly is *"an observation (or a subset of observations) which appears to be inconsistent with the remainder of that set of data*, *"patterns in data not conforming to a well-defined notion of normal behaviour*, *anomalies are unusual and significant changes in a network's traffic levels, which can often span multiple links*, *non-conforming interesting patterns compared to the well-defined notion of normal behaviour*. In [35], an anomaly is *a deviation from the expected (normal, regular) state or behaviour*. In some studies [48], researchers considers attacks to be anomalous and a non-attack behaviour not conforming to normal state as non-anomalous.

In the context of thesis, an anomaly is *a deviation from the expected (normal, regular) state or behaviour*. It is not differentiated between a malicious or non-malicious traffic, but between a normal state and deviations from this normal state. A downside of this approach is that genuine traffic with low periodicity may be tagged as anomalous. This can happen, for example, if we have traffic records for university information system with baseline created during summer and students then register to courses at the beginning of September. The anomaly detection system can flag this traffic as anomalous when compared to the baseline computed during the summer, although the students' registration represents non-malicious traffic.

2.1.1 Traffic Anomaly Types

In his survey, Fernandez et al. [20] say that there are many types of network anomalies. Networks anomalies can be separated into two main categories: a) by their nature, and b) by their causal aspect.

Anomaly Nature Based Categorization

Categorization based on nature of anomalies relies on the way how they are characterized. It uses the context of a point in its dataset to determine the anomalousness of such a point. Detection methods which search for nature based anomalies are not concerned with outer sources, causes of the anomalies, nor intents of the original sender of the data.

These anomalies can be separated in three subcategories: a) point anomalies, b) collective anomalies, and c) contextual anomalies:

a) Point anomalies are the fundamental ones. A data point becomes a point anomaly, if its values significantly differs from the rest of points included in a dataset.

b) a collective anomaly consists of a group of data deviating from a normal state. If only one of the data point in the group is taken in account, it is not considered anomalous. However, repeated occurrences of the same point anomaly can be considered a contextual anomaly.

c) a contextual anomaly is considered anomalous based on the state of a system. For detection of contextual anomalies, two sets of data are needed. One of the sets defines a context in which anomalies are to be detected (e.g., time series of normal behaviour during a day). The other set consists of data points to be examined for presence of anomalies (e.g., data collected in one day, to be matched against the normal behaviour).

Anomaly Causal Aspect Based Categorization

As the category name implies, this categorization relies on the cause of the anomalies. It also distinguishes between malicious and non-malicious intents. Because not all anomalous points are representants of an attack, they can be grouped into categories [4]: a) operational/misconfiguration/failure events, b) flash crowd/legitimate but abnormal use, c) measurement anomalies, d) network abuse anomalies/malicious attacks:

a) These are non-malicious events, typically caused by hardware or software failure, bugs, human errors, bad configurations, or non-sufficient system resources.

b) Flash crowds are *large floods in traffic, which occur when rapid growth of users attempts to access a specific network resource, causing a dramatic surge in server load* [20]. They occur when more users access a network resource than the resource can handle. The course registration example is a legitimate but abnormal use.

c) Measurement are caused by problem during data collection, resulting in a skewed or missing baseline data.

d) Network abuse anomalies are anomalies caused by malicious intents of original data senders. They seek to disrupt, destroy or deny network devices, servers, and other resources. They are threats trying to perform a security incident or breach.

2.2 Detection methods

Currently, many methods for anomaly detection are used. It cannot be said which method is the best, as all of them have an optimal use case and are targeted to be used in different scenarios.

Popular methods and algorithms for anomaly detection includes [20, 34]:

- *Statistical methods* commonly apply probabilistic models. Examples are: Wavelet Analysis, Principal Component Analysis, Hidden Markov Models, etc.
- *Clustering methods* group objects into distinct clusters of object based on their similar behaviours or attributes. Examples are: K-means, DBSCAN, BIRCH, Gaussian Mixture Model, etc.
- *Finite state machine methods* use a finite automata model of computation to model a network behaviour.
- *Classification-based methods* use machine learning to classify traffic as anomalous or not anomalous. Examples are: naive Bayesian, Support Vector Machines, Neural Networks, K-nearest Neighbours, Decision Trees, etc.
- *Evolutionary computation methods* are inspired by biology. Examples are: Particle Swarm Optimization, Artificial Immune Systems, Genetic Programming, etc.
- *Hybrid methods* combine previously mentioned methods or their parts to create new anomaly detection systems.

Although all listed methods can be used to detect anomalies from system logs, next sections talk primarily about statistical methods. Other methods are then described briefly. This is done because of the author's familiarity with these methods, and due to statistical methods being tested and proved to be working by many researches (see Section 2.4). This is also a reason for choosing a statistical method for anomaly detection in this thesis.

2.2.1 Statistical Methods

Statistical methods rely on probabilistic methods to define normal network behaviour. When using these methods, an anomaly generally does not mean a detected attack. These methods detect changes in traffic, whether it is an attack or not. Most methods define a hard threshold to separate anomalous data point and the normal ones. The main concern when using statistical methods is how to determine the optimal values of the threshold for a model in order not to produce unsatisfying amount of false positives [20].

Statistical (and others) methods can process diverse kinds of input data formats and sources such as IP flow [24], NetFlow [31], Firewall and IDS logs [14], TCP dump [57], SNMP [41], system logs, etc. Validation metric is also not standardized, and lots of researchers use different ones. Metrics range from packet count, through false positive rate, recall scores, correlation coefficients, to linear regression [20].

Advantages of statistical methods include intrinsic capability to detect anomalies better than other methods, ability to adapt and learn the expected behaviour of a network. They do not need any priori information about the environment [20].

Disadvantages include a possibility for an attack to be included in the training data and, as such, the model learns its pattern as the normal state of a network. Statistical models

often require a significant portion of time for their training, as well as plenty of training data coming from the tested system before they reach operationality. A static quality of the threshold may not yield satisfactory results in the real world, where conditions often change.

The fact that these methods cannot distinguish between reasons for changes in traffic data, and thus, cannot differentiate attack from an increased amount of non-malicious traffic can be considered both advantage or disadvantage, depending on the desired outcome.

2.2.2 Principal Component Analysis

With its invention in 1906 by Karl Pearson [32], the Principal Component Analysis (PCA) is one of the oldest and widely used method to drastically reduce datasets dimensionality while most original information is preserved [26]. The PCA method achieves this by deriving new uncorrelated variables which maximize variance from the original dataset. These new variables are called Principal Components. The lossy nature of PCA can be used to detect anomalies in network traffic. The reduced variance of a PCA model fitted to normal data points cannot explain some outlier values, and thus they can be marked as anomalous.

Mathematical explanation of PCA

Assume dataset D with p numerical dimensions and n total observation. This data can be represented by $n \times p$ matrix $\mathbf{X}$ or n observation vectors $x_1, \ldots, x_p$. The PCA methods aims to find linear combinations of $\mathbf{X}$ having the maximal possible variance. These combinations are calculated by $\sum_{j=1}^{p} a_j \mathbf{x}_j = \mathbf{Xa}$, where vector $\mathbf{a}$ is a vector of constants $a_1, \ldots, a_p$. The variance of a combination is then given by $var(\mathbf{Xa}) = \mathbf{a'Sa}$, where $\mathbf{a'}$ is a transposed vector $\mathbf{a}$ and $\mathbf{S}$ is a covariance matrix of the dataset. The solution thus lays in maximizing the $\mathbf{a'Sa}$ by finding optimal values of vector $\mathbf{a}$.

Requiring the vectors to be unit-norm is necessary for finding a well-defined solution. Thus, the maximizing of $\mathbf{a'Sa}$ can be rewritten as $\mathbf{a'Sa} - \lambda(\mathbf{a'a} - 1)$, where λ is a Lagrange multiplier. After differentiation of $\mathbf{a'Sa} - \lambda(\mathbf{a'a} - 1) = 0$ with respect to $\mathbf{a}$, we get $\mathbf{Sa} - \lambda\mathbf{a} = 0 \Leftrightarrow \mathbf{Sa} = \lambda\mathbf{a}$. It can be deducted, that $\mathbf{a}$ must be an eigenvector and λ the eigenvalue of covariance matrix $\mathbf{S}$.

With matrix $\mathbf{S}$ shape being $p \times p$, it has p eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$. Their eigenvectors can be defined to form an orthonormal set of vectors. It can be shown with Lagrange multiplier method, that a set of eigenvectors of a matrix $\mathbf{S}$ is the solution to getting p new combinations $\mathbf{Xa}_k = \sum_{j=1}^{p} a_{jk} \mathbf{x}_j$, which are uncorrelated with previous linear combinations. These linear combinations $\mathbf{Xa}_k$ are named *principal components* of the original dataset D. Values of vector $\mathbf{Xa}_k$ are called *PC score* and values of eigenvectors $\mathbf{a}_k$ are called *PC loadings* [26].

When the resulting linear combinations are sorted in descending order by their corresponding eigenvalues, then the first component explains the most variance of the original data, while each next principal component explains less variance. Ratio of explained variance $V_{\mathbf{XA_k}}$ of a principal component $\mathbf{XA_k}$ can be obtained by calculating (2.1).

$$V_{\mathbf{XA_k}} = \frac{\lambda_k}{\sum_{j=1}^{p} \lambda_j}. \tag{2.1}$$

Dimensionality reduction of the original dataset is performed by keeping only a predefined number of components with greatest explained variance.

PCA for anomaly detection

The first few principal components contain most information and explain the biggest percentage of a data sample variance. They are strongly related to features having large variances [48]. This means that most data samples in the original dataset have their values explainable by the first components. The outliers in these components normally match to outliers in some dimension of the original data.

The last principal components, e.g., those with lowest amount of explained variance, usually have vastly different values for data points, which are anomalous in regard to the correlation in the original dataset. These anomalies represent more subtle anomalies, which cannot be detected using the major principal components. If most of the variance of a sample is explained by the latter components, then that sample is suspected to be an outlier [48].

Shyu et al. [48] proposed anomaly detection based on the aforementioned criteria. First, they calculated the PCA components, scores, eigenvectors, and eigenvalues. They then classified a data point as anomalous when it matched Equation (2.2). A data point was considered normal if it fulfilled conditions in (2.3):

$$\sum_{i=1}^{q} \frac{y_i^2}{\lambda_i} > c_1 \ \vee \ \sum_{i=p-r+1}^{p} \frac{y_i^2}{\lambda_i} > c_2 \tag{2.2}$$

$$\sum_{i=1}^{q} \frac{y_i^2}{\lambda_i} \leq c_1 \ \wedge \ \sum_{i=p-r+1}^{p} \frac{y_i^2}{\lambda_i} \leq c_2 \tag{2.3}$$

where c_1, c_2 are predefined thresholds, y_i is the PC score of a component i and $lambda_i$ is the eigenvalue of a component i.

Jeff Prosise proposed a method based on the inherent data loss when transforming a dataset to a PCA reduced form with less dimensionality [42]. The PCA transformation to lower number of dimensions is reversible, although a reverse-transformed dataset is only an approximation of the original one. One can thus first apply PCA on the dataset, and subsequently apply reverse transform on the reverse dataset. The loss of information between the original dataset and reverse transformed dataset (accuracy) is tight to a number of PCA components used and ratio of explained variance. The outlier data points should exhibit a larger values of loss, since the PCA model cannot keep as much information about such data point. This is caused by the low amount of explained variance of the major components, and on the other hand, a high mount of explained variance of the minor ones.

Prosise calculates the loss of information between normal and transformed data point using mean square error method. The hard decision whether a point is anomalous is made by comparing the loss value and a predefined threshold. Threshold can be set manually based on an educated guess, or it can be based on other researches. A threshold finding algorithm can be used if the data is labelled or pseudo-labelled.

Mahalanobis distance

Because of the vulnerability of PCA to outliers in training data, it is necessary to remove such data points from the training dataset. A metric is needed to separate between normal and outlier data. While many distances like Euclidean distance or Canberra metric are often used for measuring similarity between points, the Mahalanobis distance represents

a distance between two points in multivariate space. This feature is exploited for outlier detection, because datasets normally contain multiple dimensions which can be, and often are, correlated [50].

The Mahalanobis distance measures the distance between a point and a statistical distribution while taking into account correlations between variables in the dataset. It represents a distance between a (generally multidimensional) point x, and a distribution with the mean vector of a distribution $\boldsymbol{\mu}$ and covariance matrix $\mathbf{S}$. The general formula for calculating Mahalanobis distance is:

$$D^2 = (x - \boldsymbol{\mu})^T \mathbf{S}^{-1} (x - \boldsymbol{\mu}) \tag{2.4}$$

where D^2 is squared Mahalanobis distance, $\mathbf{S}^{-1}$ is an inverse covariance matrix of a data distribution and ^{T} represents a transposition.

Mahalanobis distance can be applied on dataset extracted from log files which is not labelled. A subset of such dataset, consisting of records with the highest Mahalanobis distance relative to the rest of it, can be proclaimed to contain outlier records. The equation and facts provided in this section are taken from [19].

2.2.3 Other Methods for Anomaly Detection from Logs

Methods for anomaly detection from system logs vary primarily by the means of extraction of features from the logs and the nature of such selected features. If individual features, which can be represented by a number or a character string, are extracted from the logs, then clustering methods, or statistical methods are often used. Clustering methods can be used in conjunction with decision trees algorithms to improve the overall efficiency of an algorithm.

Clustering

Clustering methods can be used to group log events based on the proximity of data point to each other. The proximity is measured by a distance (Euclidean, Mahalanobis). Normal events should all be grouped into a large cluster, while the anomalous points too different to be a part of a normal cluster are either grouped together into a smaller cluster, or scattered around the normal cluster. Events in the smaller cluster can be declared anomalous or be subject to further examination [25].

Decision Trees

Decision trees, an example of a supervised learning method, create a tree of decision nodes and solution leaves. A data point is classified by traversing the decision nodes from the root of a tree until it reaches one of the leaves with the final class. Tree boosting algorithms use an additive model to build multiple trees in order to build a final decision forest iteratively.

Autoencoders

Another method for feature extraction is to parse individual log lines into chunks of uniform length and tokenizing these chunks into terms. In [15], authors counted the number of individual terms in chunks and developed a method to score a chunk based on the number of terms in it. These scores form a normative baseline of the autoencoder, a form of *feedforward neural network where the output layer has the same dimension as the input*

layer. After the learning phase, an autoencoder is able to threshold reconstructed scores on its output, and thus differentiate between anomalous and normal log chunks. Chunks not included in the baseline cannot be restored by the autoencoder, so they achieve a large reconstruction error rate.

DBSCAN

Both the tokenization of log records and extraction of features can be used simultaneously. Alghamdi et al. [1] used the tokenized values as another dimensions together with extracted features to train DBSCAN clustering algorithm. They do not use the resulting clusters for anomaly detection, but for further processing, pattern extraction and behavioural analysis.

2.2.4 Choosing a Method

An important factor for method choosing is the availability of annotated datasets. When having access to an annotated dataset with correctly assigned labels, researchers usually select from supervised learning methods, like in [52]. On the other hand, if researches study real life data without labels, they must either use semi-supervises methods, like Catillo et al. [15] did, or unsupervised methods, which were used by Mehta et al. in [36].

2.3 Anomaly Scoring

The act of anomaly scoring tries to express the anomalies detected by aforementioned methods via a numeric value. These values can be derived from various numbers available after the detection. a relative number, e.g. ratio between count of anomalous point in a dataset and a total dataset size, can server as a score.

When using machine learning methods for detection, metrics and scoring described in [44] can be used to both rate the success of anomaly classifiers, and to compute anomaly scores and perceived confidence in the detection. These methods include *Confusion matrices*, *F1 scores*, *Recall*, *Precision*, etc.

If a result of the scoring method can be represented as a number, such metric can serve as the final score. In case the value does not fall into the interval <0,1>, then a normalization might prove useful. A normalized score can also be represented as a percentage. Whether a score of 1 or 100 % means a totally anomalous event or an event with no anomalies depends on the intention of an author.

2.4 Anomaly Detection Related Work

Many works were created to solve the problem of anomaly detection. The vast amount of existing papers show that there is not a single solution for this task. This is given by a diverse nature of both input data and desired results presented in the research. This section provides a brief look into possible solutions used today.

2.4.1 K-Means and XGBoost Models

João Henriques [25] with his colleagues used the K-Means method and the gradient tree boosting algorithm XGBoost for anomaly detection in a dataset of HTTP traffic, which is

not annotated. The dataset used is the NASA-HTTP dataset[1]. For the use of unlabelled dataset, authors decided to work with unsupervised methods.

They first used K-Means clustering method to separate HTTP records into binary categories (normal/anomalous). Then they utilised the gradient tree boosting algorithm XGBoost to generate a decision tree which allows to quickly categorize new incoming data.

Given the large size of the data, they decided to first extract features of the HTTP logs. These are *day, month, year, hour, minute, second, operation, page, method, day of week, IP address, length*, and response. These selected features were later used for training of a Scalable K-Means++ model proposed by Bahmani in [2]. The final decision between anomalous and normal cluster was based on the size of respective clusters, where the cluster with smaller amount of data points were declared anomalous.

They then applied the XGBoost algorithm to create a binary decision on data pseudo-labelled by the K-Means algorithm. This step is crucial to quickly determine the nature of data points coming in the future without re-calculating the entire clusters again. Instead of implementing the models by hand, the Dask[2] Python library, which already includes the parallelized version of algorithms, was used.

They did not explicitly state achieved model recall or rate of false-positives, only stating that the used XGBoost model labelled 100 % of events clustered into the anomaly cluster using K-Means correctly. The advantage of their method lies in creation of a decision tree, which reduces resources necessary for classification of new data points.

Features extracted from the raw HTTP logs by the authors (especially the *day of the week, hour, minute* features) inspired the set of features extracted for the anomaly scoring in this thesis. As the presented methods belong to the category of unsupervised learning, and they do not require labelled dataset, they were a valid option for use in this thesis. The final decision was however in favour of the PCA method, mainly for its simplicity, and no need for combining more methods (clustering and decision trees) for the classification.

2.4.2 K-Prototype and K-Nearest Neighbours

Zhaoli Liui et al. [33] proposed an integrated method for anomaly detection from massive system logs using K-prototype clustering and K-Nearest Neighbours classification algorithms. They argue, that using these methods is appropriate, because it allows to detect anomalies without any priori information. Thus, even new, previously undetected, anomalies can be found.

They divided the process of detection into five distinct steps: 1) log collection, 2) feature extraction, 3) clustering, 4) filtering, 5) refinement. Note, that steps one and two are usually necessary in any anomaly detection, no matter of the used method.

They processed system logs from more than 50 servers, from which 10 features split into two categories (logging activity, session statistics) were selected. Their dataset was mainly unlabelled, except for logs from one target server. Logs of this target sever were labelled in coordination with security engineers, who performed scheduled attacks targeting the server. These labelled logs were used for evaluation. These features range from usernames used for legged sessions, through session lifetime to a frequency of file operations during a user's session. Since these features include both numerical and categorical attributes, they chose the K-prototype algorithm to calculate clusters. Upon the inspection of the returned clusters, they proclaimed large, dense clusters as normal and filtered them out

[1]NASA-HTTP dataset: https://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html, accessed [2024-01-20]
[2]Dask library: https://www.dask.org, accessed [2024-01-20]

before taking the next steps. The chosen threshold T for this decision is $T = n/K$, where n is the event count and K is the number of clusters. Clusters with more events than T are considered normal and discarded. They managed to reduce the dataset size by more than 70 % by the application of this simple filter.

From information about spatial properties of every data point obtained by the clustering, they calculated two distances per data point. The first distance value represents the distance between a given data point and a centre of the cluster to which the data point belongs. Second value is the distance between a data point and a data point nearest to it. They used only these two distances as features for training a k-NN classifier on a reduced dataset. They argue that using the K-prototype algorithm only yields non-desirable results in terms of precision and false positives, while calculation k-NN with entire dataset as input is time-consuming.

They demonstrate, with graphs, that their proposed solution results are more precise than using only one of the methods, while also being fairly time efficient. They evaluated their combined method on several datasets using the *precision*, *recall*, and *false alarm* metrics. In all their experiments, the precision stayed above 90 %, false alarm rates were below 2.5 %, and they maintained the recall values above 0.85.

Although they used different source of data and tried to achieve different goal than what the goal of this thesis is (user activity vs. network traffic), the anomaly detection system proposed by Zhaoli Liui et al. could be also used for other purposes. Especially , the filtration of potential anomaly candidate is transferable into the settings of network anomaly detection, given the enormous amount of traffic needed to be processed. However, this approach is viable only when using a combination of methods like the authors did (K-Means + K-NN), which limits its potential.

As their logs are mainly focused on device access and performed operations, their features are not transferable to the work of this thesis. The use of k-NN method is also not possible in this thesis, as labelled dataset is necessary for its training.

2.4.3 k-NN, Isolation Forests, Local Outlier Factor

Mehta, Kothuri and Garcia published a paper concerning detection of anomalies from network connection logs [36]. They utilised multiple distributed architectures and systems to collect connection logs and to analyse them. From network connection logs collected in the network of CERN[3], primarily kept stored for audit, performance monitoring and alerting, they extracted features for later modelling using one of the techniques, that were demonstrated in the paper. The feature selection was aided by first reducing the dimensionality of the dataset using Principal Component Analysis (PCA, see Section 2.2.2) and Singular Vector Decomposition (SVD).

Arguing, that only an unlabelled dataset was available to them, they chose to explore possibilities of unsupervised learning. Four distinct models were fitted with preprocessed data: k-NN, isolation forests, local outlier factor, and one-class support vector machine. Demonstration of results for every approach was shown. Their results are meant as a proof-of-concept before implementing a system for real world use at CERN.

They admitted, that their approach was not able to correctly classify all connections-e.g., connections from legitimate users detected outside working hours were marked as anomalous. It can be polemized, if such events are desired to be marked normal or anomalous.

[3]Conseil Européen pour la Recherche Nucléaire (European Organization for Nuclear Research)

The general division of anomaly detection methods and types of anomalies, as well as a use of the PCA and SVD methods for dimensionality reduction decribed by the authors, represent valuable knowledge in the scope of this thesis in terms of determining what methods are best utilised for unlabelled/labelled data. However, not much information is given about feature selection, model training, detailed interpretation of results, or even metrics for result evaluation. No direct comparisons can thus be made.

2.4.4 Deep Autoencoder

AutoLog is a name of the system developed by Catillo et al. [15]. They took a different approach than extracting features manually. Their solution for data extraction was to split log lines into tokens, and then using a predefined mathematical formula to score chunks: batches of data of equal length consisting of several tokens. Chunks score is computed by the sum of entropies of all terms found in the chunk. Entropy of a term is computed against a database of normative chunks.

They used these scores to train a deep autoencoder, a type of neural network consisting of an encoder part (creating a compressed representation of an input) and a decoder (reconstructing original input from the compressed represenation of data). An autoencoder is able to reconstruct input scores on its output. The reconstruction is, however, never perfect, resulting in a loss of information. The loss of information is given by a nature of an autoencoder. It is able to best reconstruct data similar to data, on which it was trained. Data unseen during the training phase cannot be represented by the encoder part of the neural network, and it cannot be thus reconstructed by the decoder part. For scores not seen in the training dataset, the autoencoder is not able to reconstruct them with minimal loss. This fact enables researchers to separate anomalies from normal traffic.

They tested their solution with several configurations of neural network and validated the model on four datasets: a proprietary industrial system logs, a microservices system logs, public supercomputing system logs, and logs of Apache Hadoop cluster. They also evaluated results from five other methods together with their Autolog in a comparative study.

The authors leveraged the fact that their dataset is labeled, to compute evaluation metrics for unsupervised learning methods, which they tested against the same dataset as AutoLog was trained on. Methods used in the comparison were: Isolation Forest, One-Class SVM, a decision tree, and a vanilla autoencoder (autoencoder consisting of only one hidden layer between the input and the output layer). All methods was evaluated using the *F1 score*, *Precision*, and *Recall* metrics. AutoLog outperformed all other methods, with decisions trees being almost always second in performance, not more than six percent behind.

This paper demonstrated that other methods for anomaly detection in log files exist and that they are successful if used correctly, and paired with a suitable dataset. The main observable advantage of this method is its functionality with many heterogeneous types of log files. The structured labelled logs available and the very different approach taken by the authors of this paper and the one presented in this thesis make it difficult to draw significant connections between the AutoLog paper and this thesis. The main parallel found is the way the PCA method is utilised in this thesis and the way in which an autoencoder works. The PCA is used in two runs on the same data - for a transform from a high dimensional dataset to a dataset made of few features only. Reversed PCA is the applied

on his reduced dataset, and the original dataset is reconstructed with inherent error. This process resembles an autoencoder.

2.4.5 DBSCAN

Alghamdi and Reger [1] designed a framework for the pattern extraction from heterogenous log files provided in the SoTM34 log dataset[4], labeled by the authors with help of existing analyses of the SoTM34 dataset. Their aim was to use these pattern for multi-stage threat detection.

They developed a strategy for behaviour analysis based on logs. For data preprocessing, they use a mix of both methods showed in the aforementioned papers — AutoLog [15] and Henriques' paper [25]. They utilize both tokenized strings from log lines, and numeric features extracted from the logs. These features are unique for different protocol-specific log files. The number of extracted features varies between four and twenty-five, depending on the specific log file chosen, as the dataset contains multiple logs (HTTP Access, HTTP Error, HTTP SSL Error, SYSLOG Messages, SYSLOG Secure, SYSLOG Mail, and SNORT). After normalizing the numerical values in the dataset with min-max scaler to place all values in the range between zero and one, clustering is performed using the DBSCAN algorithm on the behavioural features—they differentiated between static features like IP addresses and behavioural features obtained by log tokenization. DBSCAN was run on the behavioural features to capture the specific behaviour of threats. The static features were later combined with DBSCAN results to create a well specified threat records for alerting in a custom developed Action Center.

They evaluated achieved results using the following tests: *Homogeneity Completeness*, *V-measure*, *Adjusted Rand Index (ARI)* and *Adjusted Mutual Info (AMI)*. These tests were chosen by the authors, because *„evaluating the performance of clustering algorithms is not as straightforward as classification algorithms where precision and recall are calculated by counting the number of errors and correct classified data points based on a priori knowledge of actual labels“*. Alghamdi's method achieved more than 90% rates in most tests of the well-known SotM34 dataset. Only results under the 90 % mark were of the Syslog Mail and Syslog Messages log files. Here, the methods achieved 82-87 % in ARI and AMI. They did not explain this lower rates for Syslog.

Features selected by the authors from the HTTP logs were inspirational for the selection of features for both anomaly detection and reputation rating in the theses. Fields extractable from the HTTP events, like *Referrer* or *User Agent* provide useful information which can decide whether the event is anomalous or whether a malicious intent is present in the data.

2.4.6 Survey on Network Anomaly Detection

Fernandez and his colleagues [20] published a comprehensive survey on network anomaly detection. They defined a set of goals to describe in their study, as they felt other similar studies focused too much on a specific area. They first defined the term anomaly and described existing categories of anomalies based either on their nature or their cause. Network data types were listed and explained, as each research in the field prefers to use different sources for their datasets. As the anomaly detection is often used in IDSs, they also talked about these systems. In the chapter about different detection techniques and methods, they

[4]Scan of the Month, Scan 34: https://honeynet.onofri.org/scans/scan34/, accessed [2024-02-20]

explained in detail groups of detection methods based on the main principle of the method. These are the groups referred to in Section 2.2. For each category, they talk about several studies and research, which used such methods. They also created comprehensive tables comparing those studies for each category. Their knowledge was cited in the theoretical chapters of this thesis. Information provides in this paper influenced the final selection of the PCA method for use in this thesis, as the PCA is suitable to be paired with large dataset of unlabelled data, and it is able to detect point anomalies in data.

2.5 Summary

This chapter described anomalies in network traffic using various definitions, as no one concrete definition exists. It explained the term *anomaly* in the scope of this thesis (meaning *not normal, while intent of a deviation does not matter).* Anomaly types and methods for anomaly detection were discussed. More text was dedicated to statistical methods and its representant, the Principal Component Analysis, for its use in the practical part of the thesis. Other methods were spoken of shortly. Mahalanobis distance for outlier marking was introduced, along with other methods one might use to find anomalies. A suggestion to transforming results of a method into the final anomaly score was brought up. Related work in anomaly detection provided information about existing solutions. An insight into what methods are suitable for a desired goal and the best matched dataset properties for the method was brought forward. Commentary about usefulness of the related papers for this thesis was provided.

Chapter 3

Introduction to Reputation Systems

Because of the general availability of Internet connectivity, and its global interconnected nature, any node connected to it can establish connections to any other connected node (if mechanisms such as *network address translation (NAT)* are not considered). This leads to plenty of communications, where one side does not necessarily know the identity of the entity at the other end of a connection. Without the familiarity, the node or its user cannot easily determine intentions of the other side and the integrity of its services. Publicly available data usable for rating endpoints accessible via the Internet can be a useful resource helping with a defined problem.

This chapter explains the problematic of calculating reputation scores and shows the new and current methods used to deal with this technique. It provides a context to fully comprehend the reasons why the scoring of network IP addresses is important and why the reputation of network nodes represents an ongoing academic and business effort.

The chapter starts with a description of trust in the real world, as the core principles of it can be, and often are, brought over to the digital world.

3.1 Trust in a Real Life Scenarios

In the real life, where a person has physical relations with other people or businesses (entities), one can rely on (*trust*) another person or community for their opinion on the entity, which signifies the *reputation* of an entity.

Thus, the term *trust* can be defined in many ways, but with the similar notion. Jøsang et al. [28] defines two types of trust — *Reliability trust* 3.1.1 and *Decision trust* 3.1.2.

Definition 3.1.1 (Reliability trust)**.** Trust is the subjective probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends.

Definition 3.1.2 (Decision trust)**.** Trust is the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible.

It can be concluded from these definitions, that *trust* is a relation between two entities, a truster (the one who trust), and a trustee (the one who is trusted). If a trustee proves to the truster, that they can be trusted, they become *trustworthy*. Trust as a relation is

asymmetric and transitive. If a node a trusts a node B, which trusts a node C, then the node a trusts node C, if node B refers C to A. This is called a derived trust by [28]. The behaviour is shown in Figure 3.1. In this figure, party B trusts party C. If B refers C to A, then a can trust C.

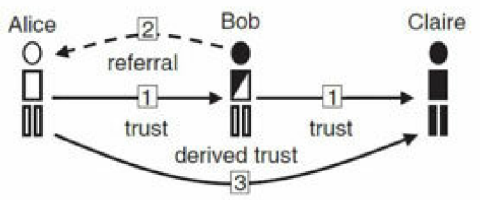


Figure 3.1: Transitive property of the trust relation. Image taken from [35].

According to Slee [49], there exists a third party to the system of trusters and trustees, and its name is an *opportunists*. An opportunist is an entity seeking to deceive potential trusters by mimicking *signs or signals* of trustworthiness, although it is not trustworthy. Gambetta [22] says, that the existence of opportunists creates a *problem of secondary trust*. The question of trust then becomes *Can I trust the signs of trustworthiness of the trustee*, rather than *Can I trust the trustee*. He argues, that this problem almost always accompanies the primary one.

a *signal*, mentioned above, is an action or a sign, which is easy to display by a trustworthy person, but difficult or impossible for an opportunist/untrustworthy entity to display. If a signal, which would discriminate between these two groups of entities, does not exist, then no effective mode of differentiating trustworthy persons and opportunist can be found.

3.2 Trust in Online World

The entity in the online world with which is being communicated cannot be assessed using previous experiences, especially when the connection between two nodes happens for the first time. Also, unlike the real life, it is not so easy to ask many other users or nodes for referrals of other nodes.

3.2.1 Systems for User Ratings

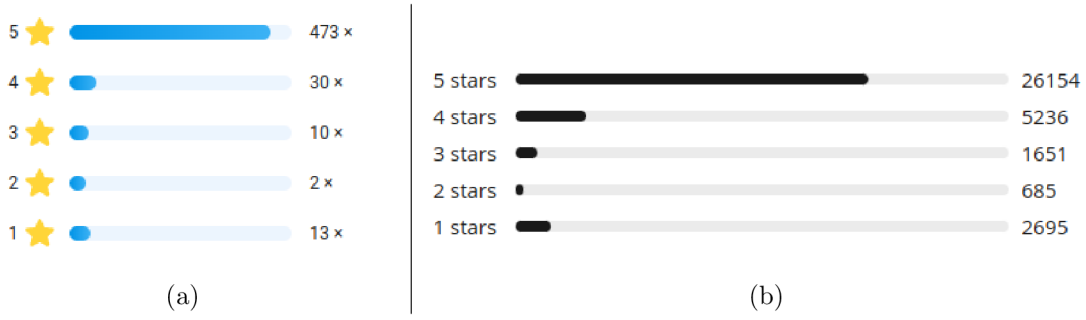
For interactions spanning both the online and the real world, there exist *rating* or *scoring* systems enabling users to rank various services and give their feedback in forms of written comments, numeric scores, stars, points, and other techniques. This feedback is called *rating*. Examples of such systems can be found on e-commerce sites, restaurant review webs, film databases, home sharing platforms, etc.

If a system is used primarily for a one-sided rating, meaning that the rating can be comprehended as an asymmetric relation, then the system is not much susceptible to opportunists or speculators. There is no reaction to feedback by the reviewed side, either because it is a non-living entity like a film or a product. In these systems, there is a low chance for a user to get something in return, thus the user does not feel pressured to give dishonest reviews.

The issue of fabricated or biased reviews become an adamant issue for so-called *peer-to-peer internet reputation systems* or businesses operating within a *sharing economy* [49]. When a receiving side do not have the ability or will to reply (restaurant review, online

marketplace review), there is a possibility of user receiving an offer from the counterpart to change their original rating. This can mean editing an original review or even its deletion in exchange for a free meal (if we are speaking about restaurant review) or a discount for the next order. In this case, the seeming trustworthiness of a rated entity is increased by deleting a negative rating, while in reality, its trustworthiness should be decreased. The entity becomes an opportunist by mimicking the signals of trustworthiness, i.e., good reviews.

This problem is even more increased when referring to services, where both parties are affected by potential negative review. These are best represented by home or car sharing platforms like Airbnb or OuiCar. While using such services, users often encounter reviews biased to one of the edges of possible rating. The review providing rating in the middle of the scale are often missing. This is caused by a phenomenon called a *response bias* [56] seen in Figure 3.2. Review 3.2a is a review for a smartphone at one of the largest online retailers in Czechia, while 3.2b is a review of wireless headphones from a known Chinese marketplace. These reviews can be non-reflecting of reality, because users leaving a review might be influenced by the other side—either by a common agreement of both sides to leave positive feedback, or as a fear of later retaliation. Described conditions can lead to a rating system which is not useful to an end user. It does not provide enough signals of trust, on which user can decide, whether he will become a truster or not.



(a) (b)

Figure 3.2: Response bias in a customer reviews.

Trust management

At the end of the 80s and beginning of the 90s of the last century, there emerged first mechanisms aimed at mimicking the trust dynamics of the physical world. Under the term *trust management*, we can imagine systems like Pretty Good Privacy (PGP) or X.509 standard [23]. The problem with these solution lies in their static nature. While the reputation of an entity changes through time in real life, in these systems, it becomes unchanging during a predetermined period of time. Trust management systems can be considered as the first predecessors of reputation systems of today.

The PGP system creates a so-called *web of trust*, and it is mainly used for authentication of humans for email services. In cryptography, this means that there is no single point of trust. Instead, trust is a transitive relation, resembling the relationship between physical world entities. An entity using PGP gradually builds its network of other trusted entities (their public keys), either by directly trusting someone's public key or by trusting a new key, which is signed by an already trusted key (indirect trust). The idea is that with the number of users growing, the chance of having a middle man user (B) who the original user

(A) trusts and who, at the same time, trusts another user (C) grows. This premise is based on a well known phenomenon called the *small world problem* [55]. The user a can then trust the public key of user C. Entities in this model can become opportunists, when they acquire a private key of another entity, thus being able to act in their name.

The X.509 is a centralised, public key infrastructure system, where no web of trust exists. A hierarchy of certification authorities (CAs) signs certificates for all involved parties. The entities cannot, like in PGP, sign each other keys. When a user wants to verify the certificate of another entity, it sees the certificate of the entity, which is signed by a private key of some certification authority. The user has a list of public keys of authorities, which he implicitly trusts. Keys of authorities can be provided by an operating system or a web browser. Signatures of the original certificates are validated using these keys. The certificate might also be signed by an intermediate CA, whose signature is signed by CA on a higher level of hierarchy. The user validates the certificates, starting at the bottom, until he finds a certificate which is signed by a CA they trust, or the certificate chain ends without successful validation. It is not trusted in this case.

3.3 Reputation systems

The aforementioned techniques and mechanisms are useful for authentication of user and network nodes. They do not, however, measure the reliability of a node by its behaviour. Academics and industry need a solution to quickly obtain and possibly share information about behaviour of nodes in a network. Fortunately, the enormous amount of data, or more often - metadata and packet headers, flowing through today's network can be automatically captured, stored, analysed and shared via multiple specialised platforms using specific protocols; thus a network reputation system is created.

For a reputation system to work correctly, it must have these properties [43]:

- long-lived entities that inspire an expectation of future interaction,
- capturing and distribution of feedback about current interactions (such information must be visible in the future), and
- use of feedback to guide trust decisions.

It also needs to have an information appropriate for a reputation measurement in a given application, a metric for a calculation, and a risk rating. For this, a reputation system uses various algorithms which, in turn, use both present and historical data. A reputation system should be resistant against attempted manipulations [35].

With the information above, a reputation system can be defined as follows:

Definition 3.3.1 (Network reputation system)**.** a network reputation system is a computational system, which applies different methods and metrics on its inputs to produce a reputation score (rating) of a network node.

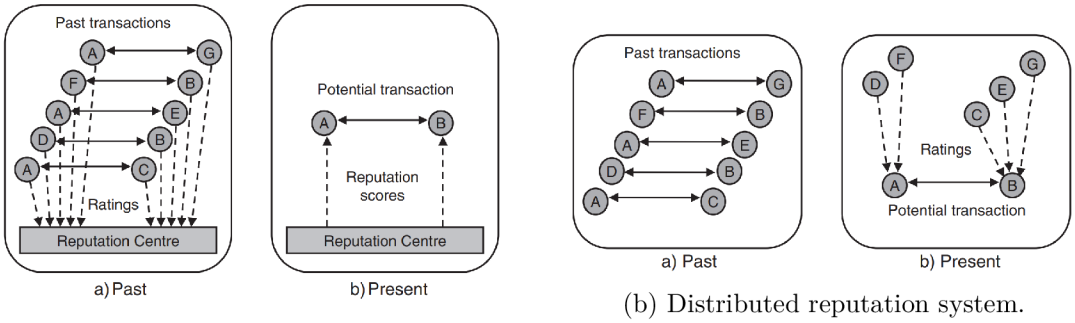
Reputation systems are considered a prevention security measure, as they cannot stop a security incident which already happened. IP addresses of a potential attacker can, however, be put onto a black list based on the reputation system recommendation and thus stop the attack before even starting.

3.3.1 Architecture of Reputation Systems

Each reputation system is different, with its creators preferring different methods to achieve results fitted to their exact needs. However, similarities can be found between all of them. Jøsang [28] says that there are two main types of reputation systems architecture: a) a centralised system, and b) a distributed system:

a) These systems consist of a centralized computation node and a network of agents. The central node utilises non-stop streams of ratings coming from each agent. The system computes reputation based on the input streams and other information available to it, like past transaction. A centralized system is shown in Figure 3.3a.

b) Distributed systems have no central node. Each distributed entity calculates its own score based on its own obtained ratings. It then shares the scores with other nodes in the distributed system. The system consists of two fundamental parts: a distributed communication protocol, and a scoring method. This dynamic form of reputation system is commonly used in peer-to-peer networks for rating reliability of a the node [35]. a distributed system is shown in Figure 3.3b.



(a) Centralized reputation system.

(b) Distributed reputation system.

Figure 3.3: Two architectures of reputation systems [35].

Regardless of the chosen architecture, a reputation system as a whole consists of agents, a reputation system core with rating functions and/or heuristics, and a database of rated nodes/entities. The relations between these components are shown in Figure 3.4. Note that in real world examples, some components can be merged into one entity with the same functionality.

Information gathered from agents alone is not sufficeient to calculate a reputation score. Thus, supplementary data are to be added into the calculation process. This includes:

- logs or events from IPSs, IDSs, honeypots, probes, or firewalls,
- lists of malign IP addresses, domains, or autonomous systems,
- spam detectors,
- white lists,
- traffic information consisting of IP addresses, domain names, user agents, URLs, etc.,
- online databases like DNS, Whois, geolocation, etc.

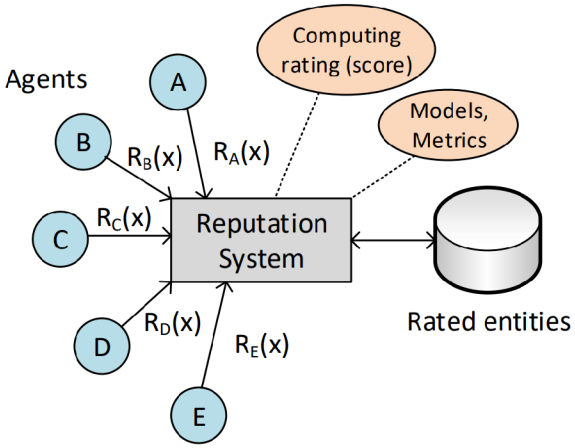


Figure 3.4: An architecture of a general reputation system. [35].

3.3.2 Communication in Reputation Systems

RFC 7070 [9] proposes an architecture/solution for *allowing one to request reputation-related data over the Internet*. The aim of this document is to simplify the interchange of information of reputation scores across the Internet.

It defines a *Reputation client*, an entity seeking to obtain a reputation score of a given content (e.g., email) based on an identifier (e.g., sender's domain). A *Reputation service* then responds to the query with data. It also gathers data from agents and uses these scores to compute scores.

RFC 7072 [11] specifies a format for interchange of information between entities defined in RFC 7070. This format defines a template URI scheme, which serves as a query by a reputation client to a reputation service. Proposed schema in this RFC is `http://service/application/subject/assertion`. Example of filled schema is: `http://reputation.com/email/baddomain.com/spam`. This query asks the `reputation.com` server to send information about the `baddomain.com` regarding spam emails originating from that domain.

a *Reputon* is a format of answer returned in response to a query to a reputation service. It is defined in RFC 7071 [10]. It serializes the reputation scores computed by the service. The returned document is of the `application/reputon+json`. An example of a rated object as a reputon is located below:

```
Content-Type: application/reputon+json

  {
    "application": "baseball",
    "reputons": [
      {
        "rater": "baseball-reference.example.com",
        "assertion": "strong-hitter",
        "rated": "Alex Rodriguez",
        "rating": 0.4,
        "confidence": 0.2,
        "sample-size": 50000
      }
    ]
  }
```

This data example is extracted from the original RFC 7071. It shows, that 50 000 people rated Alex Rodriguez, a baseball player, as mediocre strong hitter. Confidence value of 0.2 tells that there was not a consensus between the reputation agents.

3.3.3 Examples of Reputation Systems

This section deals with the design of reputation systems. Reputation systems used in the real world differs in the goal they want to active. This also affects the calculation methods of scores in different systems. Systems like PageRank are designed to rank web pages, while others (e.g., CESNET Network Entity Reputation Database (NERD)) aims to build *a constantly-updated database of the known malicious network entities* [6].

PageRank

PageRank system [12] appeared in 1998 at Stanford University. The goal of this system is not to rank network nodes, but web pages. Created by Larry Page and Sergey Brin, it later became the base for Google search engine. The authors represented web pages, interconnected by hypertext, by an oriented graph. The PageRank algorithm calculates a score for a web page A using the formula shown in Equation (3.1):

$$PR(A) = (1-d) + d(\sum_{i=1}^{n} \frac{PR(T_i)}{C(T_i)}) \tag{3.1}$$

In this equation, the result is score $PR(A)$ of page A. d represents a damping-factor. T_i is a page with a link to page A. $C(T_i)$ is the number of links leading out of the page T_i. Index i ranges from one to n, where n is the total number of pages with links to the page A.

Rating web pages using this formula ensures, that *a page can have a high PageRank if there are many pages that point to it, or if there are some pages that point to it and have a high PageRank.*

Beta Reputation System

Audun Jøsang and Roslan Ismail [27] published an article about the Beta Reputation System. This reputation system was developed to be used either separately or to be integrated into e-commerce applications. The main idea of their work is to utilise the beta probability density function to represent probabilities of binary events. Beta distribution $f(p|\alpha,\beta)$ represents a probability p of a positive event based on the values of parameters α, β. Function $f(p|\alpha,\beta)$ can be then represented via the gamma function:

$$f(p|\alpha,\beta) = \frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha,\beta)} \tag{3.2}$$

$$B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}, E(p) = \frac{\alpha}{\alpha+\beta} \tag{3.3}$$

Each target entity (the one which needs to be rated) collects ratings from reputation agents. Positive ratings can be denoted by $r = \alpha - 1$, while negative ratings are denoted as $s = \beta - 1$. A reputation function of entity T based on experiences r_T and s_T is:

$$\varphi(p|\alpha,\beta) = \frac{\Gamma(r_T + s_T + 2)}{\Gamma(r_T+1) + \Gamma(s_T+1)} \cdot p^{r_T} \cdot (1-p)^{s_T} \tag{3.4}$$

Reputation score Rep of T is represented as normalized mean E of the distribution in interval <0,1>:

$$Rep_T(r_T, s_T) = (E(\varphi(p|\alpha, \beta))) - 0.5) \cdot 2 = \frac{r_T - s_T}{r_T + s_T + 2} \tag{3.5}$$

They then define a method of discounting (a feedback from highly rated agents should be considered more significant than feedback from badly rated agents) and forgetting (newer feedback should carry more weight than an older feedback). They show how these two methods can be integrated into the Beta Reputation System. Authors also demonstrated the system performance in relation to varying settings of weight of feedback, feedback value, discount values, and forgetting factor.

As the Beta Reputation System scoring is based on a positive and negative feedback from agents, it could be used to rate network nodes with agent modifications. An agent would have to be able to provide the feedback about each transaction with network node. The feedback could be derived from conclusions made about the transaction. For example, if the transaction was a download of a file from web server, and an antivirus running on the agent would find the hash of the downloaded file in a database of malicious software, the feedback would be negative.

Due to the need of increased capabilities of agents participating in the rating process, and the intended use of the system (e-commerce), the Beta Reputation System was not selected for use in this thesis.

Cisco Talos

Cisco Talos is a threat intelligence group inside the Cisco Security ecosystem. The Reputation Center of this group *provides access to expansive threat data and related information* [54]. It provides an IP and Domain Reputation Center—real-time threat detection network, which collects data from web, email, firewall, and IPS services. A simple web client[1] allows users to query the database of the Reputation Center for an IP address, URL, or a file hash. Talos maintains not only a network reputation centre, but also a database of files. Outputs from the file reputation system are used in Cisco firewalls or Snort products. Cisco also provides a list of naming convention patterns of malicious files, a list of content, and threat categories used by Talos to classify websites and attacks. Data presented by Talos is updated every three hours.

CESNET Network Entity Reputation Database (NERD)

NERD (Network Entity Reputation Database) is a reputation system developed by the CESNET - a Czech NREN. *NERD is a service which aims to gather, aggregate and provide all the available information about known malicious network entities (mostly IP addresses)* [7].

The NERD software rates entity records represented by a set of attributes. Currently, an entity can be an IP address, BGP prefixes, autonomous systems, IP blocks, and organizations. Attributes are then dividend into two main classes: 1. primary data, 2. secondary data.

1. Primary data describing malicious activities of a given entity. Currently sourced from Warden (*a system for efficient sharing information about detected events* [17]).

[1] Talos Reputation Center: https://www.talosintelligence.com/reputation_center, accessed [2024-02-23]

2. Secondary data, represented by all other data somehow related to the ranked entity. They can be computed by NERD, like a reputation score, or sourced from external sources like domain names, geolocation, whether an IP is in a blocklist.

NERD stores records of IP addresses for a limited time. If a record comes from Warden, then it is deleted after 14 days without alert. The time period increases to 180 day when MISP database[2] is a source of an alert.

Two scores are computed by NERD to rate network entities. The first one is the Future Misbehavior Probability (FMP) score developed by Bartoš et al. [5]. It uses deep learning techniques based on various gradient boosting decision trees to determine a score representing an estimated probability that the rated entity will behave viciously in the near future. Authors selected features from data of the Warden system, that were collected across three months (September-November 2017). Their entire dataset contains 155 million alerts for more than five million IP addresses. The base features are as follows:

1. number of alerts in the last day
2. total number of connection attempts (attack volume) in the last day
3. number of detectors reporting the address in the last day
4. number of alerts in the last week
5. total number of connection attempts (attack volume) in the last week
6. number of detectors reporting the address in the last week
7. EWMA (Exponential Weighted Moving Average) of number of alerts per day over the last week
8. EWMA of total number of connection attempts per day over the last week
9. EWMA of a binary signal expressing presence of an alert (0 or 1) in each day over the last week
10. time from the last alert (in days)
11. average interval between alerts within the last week (in days, infinity if less than two alerts were reported)
12. median of intervals between alerts within the last week (in days, infinity if less than two alerts were reported)

Other features from secondary sources are added to this list, to complete a dataset with a total of 42 features. Because of the significant imbalance of the dataset, they sampled the normal class of data, which resulted in smaller training datasets. A non-linear transformation of data was then applied. Preprocessed data was used for model fitting of four models. Chosen models consisted of two neural networks (with two and three hidden layers, each with 58 nodes, a rectified linear unit used as activation function) and two gradient boosting decision trees (100 or 200 trees with max depth of three and seven). A mean square error was used as a metric to model performance evaluation. The score always falls into

[2]MISP project homepage: https://www.misp-project.org, accessed [2024-02-24]

the range <0,1>, where zero is the best possible score, and one represents the worst case prediction. A gradient boosting decision tree model consisting of 200 trees proved as the best, with more than 80 % of detected threats, while keeping the false positive rate under 10 % on the testing dataset. The dataset was labelled and composed of alerts received in the Warden system. Authors note, that the false positive data point *does not necessarily mean blacklisting a legitimate IP address, the address may still be malicious, just not attacking any of the monitored networks within the prediction window.* Taking a resulting FMP score from the predictions and combining it with a predefined threshold or a specified number of worst IP addresses, can be used to generate a blacklist which can be used in a near future of a score calculation to block traffic proven to be malicious by the model. Authors show the hit rates of blacklist generated from the FMP score. Hit rates range from 100 % when using a blacklist of 100 top scored IP addresses, to 43 % when using a 2 000 address long blacklist.

Before the FMP score was implemented, a simpler method for reputation score was used by NERD. This reputation score was calculated as a weighted average of daily scores covering the last 14 days. Daily score is represented by the following formula:

$$R_d^{ip} = (1 - \frac{1}{2^{E_d}}) \cdot (1 - \frac{1}{2^{D_d}}) \tag{3.6}$$

where R_d^{ip} is the daily reputation score of an IP address ip, E_d is the number of alerts reported in Warden for a given day d with the IP address ip as a source. D_d is a number of unique detectors that reported alerts E_d.

The method used for reputation scoring in this thesis is inspired by this NERD formula.

3.4 Summary

This chapter introduced a reader into the problematics of trust and reputation. It showed how trust can be defined in both real and virtual world. Examples of trust management systems were demonstrated. An introduction to reputation systems was presented, and architectures of such systems discussed. The chapter then talked about a working model of a communication of a reputation system proposed in several RFCs. Real world examples of used reputation systems were provided to show the vast differences between offerings from various subjects in this field.

Chapter 4

Dataset Analysis

This chapter introduces a reader to the provided dataset used for the calculation of anomaly and reputation scores. The origin of the dataset and its form are shown. It explores the available data and describes it in various ways, including graphs, histograms, timelines, or text descriptions. Interesting data features are pointed out.

4.1 Introduction to the Dataset

Our dataset contains logs from the Suricata IDS. Suricata is an open sourced project under the Open Information Security Foundation (OISF). It serves as an Intrusion Detection System (IDS), Intrusion Prevention System (IPS), and Network Security Monitoring engine.

The original, non-processed raw dataset includes:

- Extensible Event Format (EVE) JSON logs with statistical, flow and protocol data,
- statistical logs,
- Suricata system logs.

These logs are generated in real time from traffic directed into the device with a running Suricata instance. The instance used to capture study data operates on the premise of the Faculty of Information Technology, Brno University of Technology. Data for the analysis were captured from the 25th October 2022 to 31st January 2023. A subset of data consisting of a week from 1st November to 7th November 2023 was chosen for initial data exploration.

4.1.1 EVE JSON logs

EVE is an abbreviation of *Extensible Event Format* [53]. It represents a way to generate a single format for various events, which are then saved into one JSON log. This log includes alerts, anomalies, metadata, file info and protocol specific records. Network traffic captured and processed by Suricata is saved into EVE logs in the form of NetFlow data extended with information provided by Suricata itself (e.g. signature based alerts).

The EVE output is configurable by YAML configuration file. Suricata administrator can define which events are to be logged and the extent of supplementary data for each event. All records in EVE logs contain a timestamp, flow identifier, input interface, event type, source and destination information, and a dictionary with event type specific data.

Logs and configuration file example can be found in the Suricata official documentation[1]. a general record in the EVE logs is a JSON record. An anonymized version of a record (MAC and IP addresses) looks like follows:


```
{"timestamp":"2023-01-21T07:53:42.119299+0100","flow_id":1265033423671534,
"in_iface":"eth","event_type":"flow","src_ip":"X.X.X.X",
"src_port":53844,"dest_ip":"Y.Y.Y.Y","dest_port":21027,"proto":"UDP",
"app_proto":"failed","flow":{"pkts_toserver":1,"pkts_toclient":0,
"bytes_toserver":487,"bytes_toclient":0,
"start":"2023-01-21T07:53:12.119022+0100","end":"2023-01-21T07:53:12.119022+0100",
"age":0,"state":"new","reason":"timeout","alerted":false},"ether":
{"dest_macs":["aa:aa:aa:aa:aa:aa"],"src_macs":["bb:bb:bb:bb:bb:bb"]},
"host":"suricata-XXX"}
```


The record above keeps information about a flow which was reported by Suricata on 2023-01-21 at 07:53:42. Its ID can be used to match this record with another record with the same ID but different event type, which provides protocol specific information. There can be zero or more of such records. The flow originates from X.X.X.X:53844, while its destination is Y.Y.Y.Y:21027. Because of the *failed* value for the *app_proto* field, there is not another record with the same ID holding detailed information in the same log file. Failed means that Suricata was not able to infer the correct L4 protocol from the traffic. Valid protocols for the *app_proto* are: Alert, Anomaly, HTTP, DNS, FTP, FT_DATA, TLS, TFTP, SMB, BITTORRENT-DHT, SSH, Flow, RDP, RFB, MQTT, HTTP2, PGSQL, IKE, Modbus, QUIC, and DHCP. Data under the *flow* dictionary contains statistics about the given flow, together with the time of detection of the first and last packets of the given flow. The *host* field signifies the hostname of a device running the Suricata IDS.

4.1.2 Statistical Logs

Statistical log files are distinct log files generated by Suricata and comprising tables containing statistical data computed from the EVE logs. Each log record is a table, that starts with the timestamp and uptime the creation. Each table row contains a counter name (e.g. *decoder.bytes*), a value identifier (e.g., *Total, Detect, RxPcapem21, FlowManagerThread, etc.*) and the corresponding value. By default, these statistics are saved into *stats.log* file and a new table with statistics is appended to the log every eight seconds. Despite the fact that the data contained within these logs can demonstrate long-term trends, the statistical logs are not used for anomaly nor reputation scoring. If a need for such statistics arises during the scoring, they are easy to calculate from the dataset provided for scoring.

An example of statistical log records is shown in Table 4.1. Only few table rows are shown. The table shows its creation timestamp. Statistics are grouped by a broader category. For example, both number of detected bytes and packets belong to the *decoder* category. Note that the counters shown in the table depend on the Suricata settings and are unique for every Suricata instance.

4.1.3 Suricata Syslog

Systems logs of Suricata on the capturing probes utilize the widely available systemd logging capabilities. They consist of records showing the correct operational state of the running

[1]Suricata EVE logs format: https://docs.suricata.io/en/latest/output/eve/eve-json-output.html, accessed [2024-01-20]

Date: 10/25/2022 – 19:21:21 (uptime: 0d, 00h 00m 08s)		
Counter	TM Name	Value
capture.kernel_packets	Total	80
decoder.pkts	Total	82
decoder.bytes	Total	8064
decoder.max_pkt_size	Total	216
flow.tcp	Total	1
flow.icmpv4	Total	1
flow.wrk.spare_sync	Total	2
flow.mgr.full_hash_pass	Total	1
flow.spare	Total	9800
tcp.memuse	Total	2424832
flow.memuse	Total	7474304

Table 4.1: Statistical log record example.

program, with the expected severities ranging from informational records to critical failure reports. These logs include records ranging from the information about used memory, operations of various system components, to information about parsing downloaded attack signatures. These files were not used for further processing and scoring, as they provide information about the Suricata itself instead of the network traffic needed for scoring. Examples of these log records are listed below:

```
29/11/2022 -- 17:46:16 - <Error> - [ERRCODE: SC_ERR_AFP_READ(191)] - Interface
   'enp2s0' is down
29/11/2022 -- 17:46:16 - <Warning> - [ERRCODE: SC_ERR_AFP_CREATE(190)] - Couldn't
   init AF_PACKET socket, retrying soon
29/11/2022 -- 17:46:16 - <Info> - All AFP capture threads are running.
29/11/2022 -- 17:46:17 - <Info> - Interface 'eth1' is back
29/11/2022 -- 17:46:17 - <Info> - Interface 'eth2' is back
29/11/2022 -- 17:48:09 - <Notice> - Signal Received.  Stopping engine.
29/11/2022 -- 17:48:09 - <Info> - time elapsed 113.431s
29/11/2022 -- 17:48:09 - <Info> - Alerts: 0
29/11/2022 -- 17:48:10 - <Info> - cleaning up signature grouping structure...
   complete
29/11/2022 -- 17:48:10 - <Notice> - Stats for 'eth1':  pkts: 676, drop: 0 (0.00%),
   invalid chksum: 0
```

4.2 Data Exploration

Most addresses in following sections, including this, are anonymized because of the data originate from a private network. If any address is kept non-anonymized, it is a public IP address outside the BUT range.

Because of the great amount of data captured by Suricata during the three months of operation, only a subset of data was selected for the exploration. A chosen part of logs was captured between 1st November 2022 and 7th November 2022. This amount of data is considered sufficient, as it spans enough days to contain and explain longer trends, e.g., traffic difference between work days and weekends. A longer duration was not selected

Day	Number of Events
2022-11-01, Tuesday	199 644
2022-11-02, Wednesday	343 995
2022-11-03, Thursday	319 702
2022-11-04, Friday	170 997
2022-11-05, Saturday	33 567
2022-11-06, Sunday	40 446
2022-11-07, Monday	170 284

Table 4.2: Number of events per day.

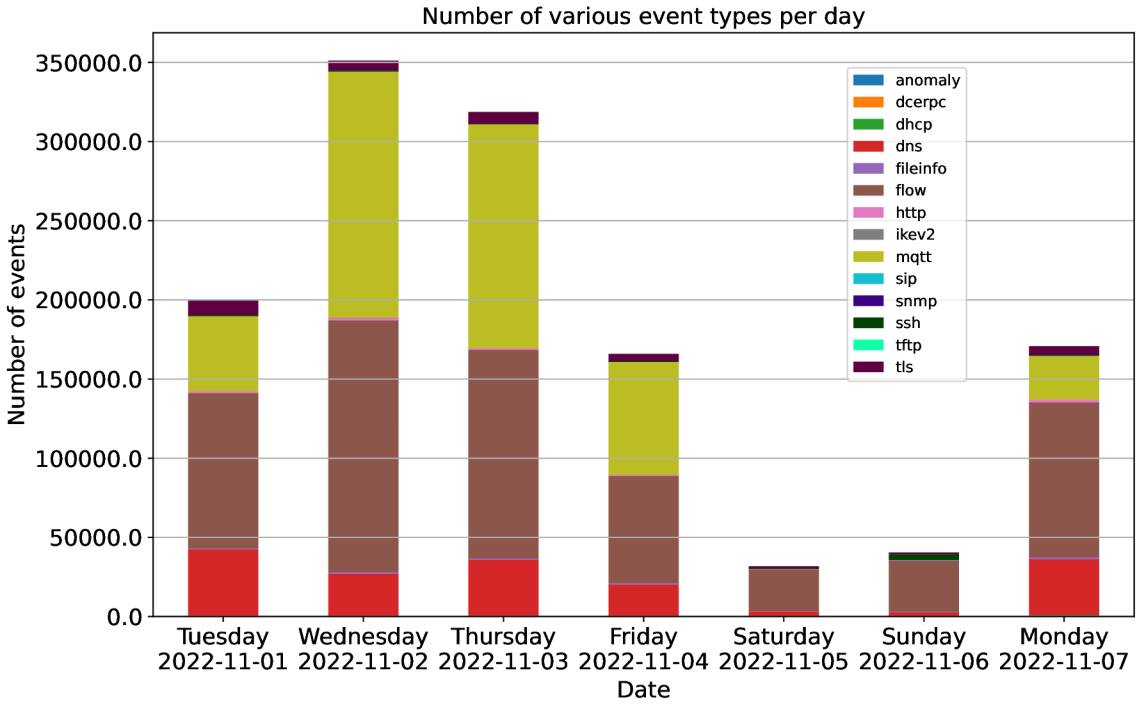
due to the increased difficulty in processing a greater volume of data. During these seven days, 931.4 MiB of logs with 1 354 665 EVE records were captured. The size of the entire dataset spanning 92 days is 17.4 GB accounting for 11 237 669 Suricata flow records. These flow records contain information about 547,229 GB transmitted through the network and analysed by Suricata. These and further statistics for the selected week are shown in Table 4.2. The 5th and 6th November 2022 are Saturday and Sunday respectively, which can explain the lower amount of captured events.

For many common protocols, the generated EVE logs do not only include the flow information (source/destination IP addresses and ports, bytes to/from, packets to/from, etc.), but also specific detailed records containing protocol-specific information. These records are linked to their corresponding flow with the use of the same ID for both records. A value of mandatory `event_type` field signifies whether the record is a generic flow or protocol specific record.

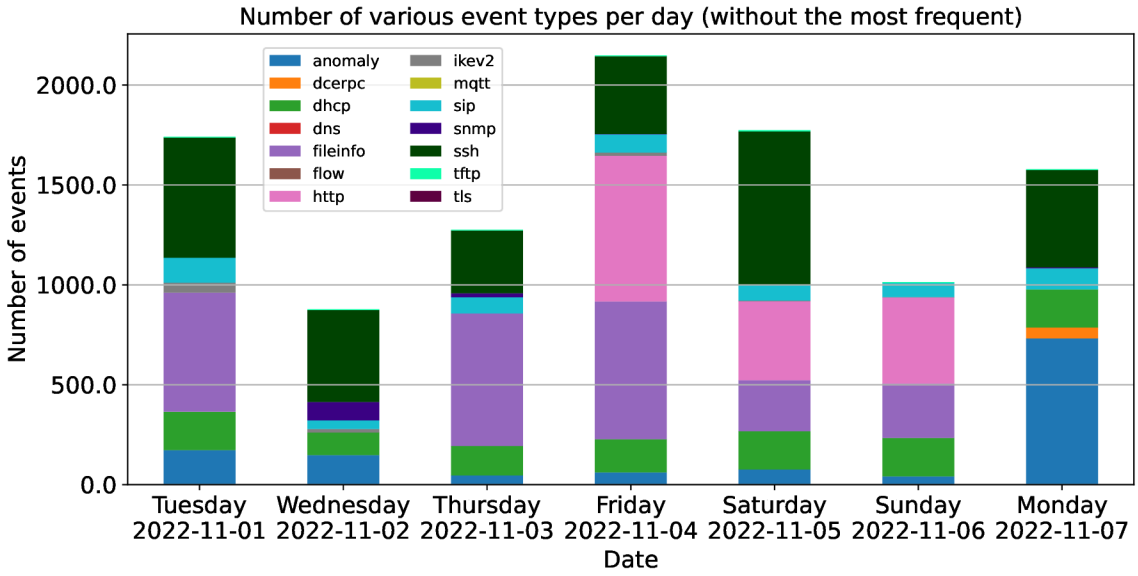
Events captured in each day and separated by the `event_type` are demonstrated in Figures 4.1a and 4.1b. Legend colours from Figure 4.1a apply to both Figures 4.1a and 4.1b. Most data transmitted is made of *flow* event type. This behaviour is expected, as the *flow* records are created for all records, even those without any specific protocol detected. The most utilized specific protocol is MQTT. The high presence of this protocol is due to the specific environment in which the dataset was collected. DNS makes a significant portion of all records. Rise of the SSH events during the weekend between 5th and 6th November together with reduction in other traffic is not surprising. As personnel working at the faculty during weekdays are not present at the faculty, user generated content like DNS is falling. On the other hand, remote SSH connections are more prevalent.

Figure 4.2 shows the cumulative amount of records of each event type across the exploration period. It is clear, that the number of flow records outnumber all other event types. This is mainly caused by the fact, that every flow ID has at least one flow records, and it can have zero or more records of another type. MQQT follows the lead with 443 000 events. DNS, TLS, HTTP and SSH come next as examples of user generated traffic. Fileinfo event type is closely tied to HTTP. Fileinfo describes files downloaded by HTTP in more detail.

Features for anomaly detection were selected from the most prominent events. These are flow, mqtt, dns, tls, and http event types. Description of data belonging to this list of types together with features derived from them is provided in the following sections.



(a) Events and their types captured in the exploration period.



(b) Less represented events and types (badly seen in Figure 4.1a) captured during the period.

Figure 4.1: Number of events and their types throughout days.

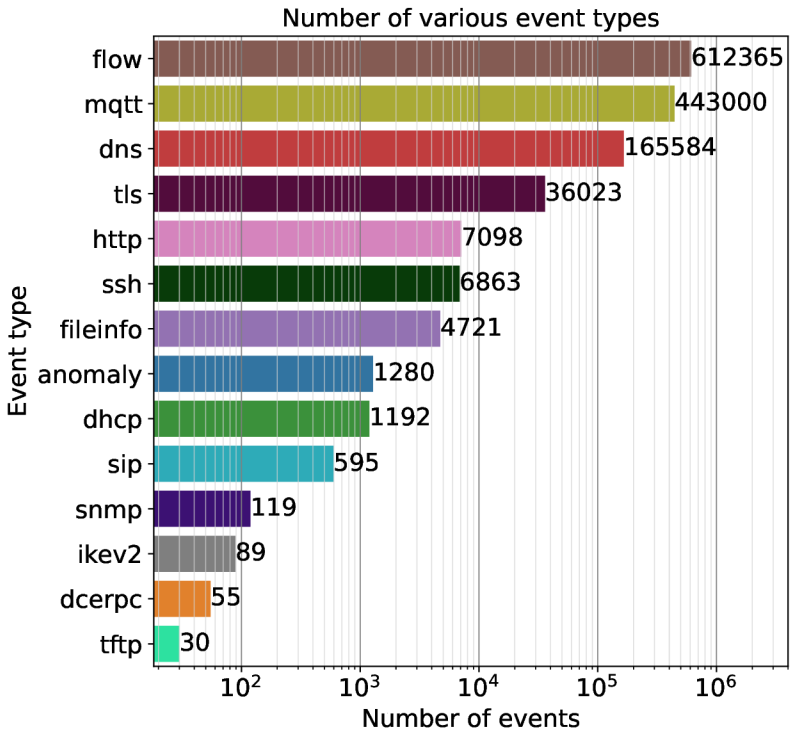


Figure 4.2: Observed number of events per type

4.2.1 Flow Event Type

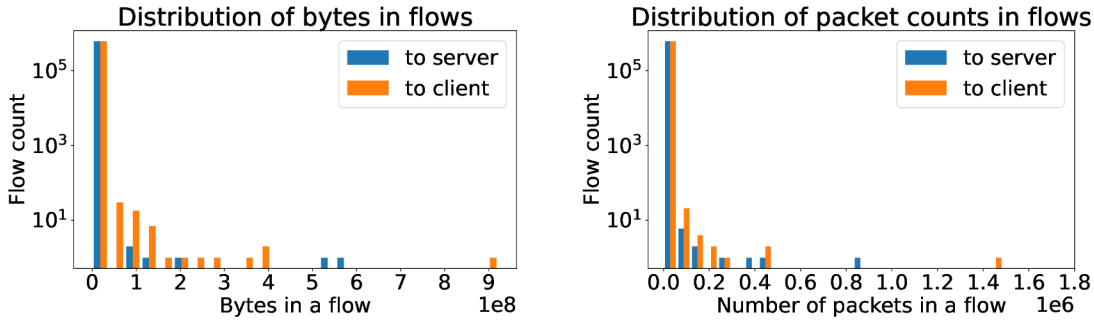
In Figure 4.2, the *flow* event type is the most frequent (brown colour in the figure). This observation is supported by two important factors:

1. For each record with an event type value other than *flow*, there exists a record with *flow* event type and the same flow ID. General information about the flow is stored in the *flow* record, whereas the protocol specific values are located in the log row with that particular event type.

2. There is only one record with *flow* value of the event type, when Suricata is not able to infer application protocol of the flow.

Distribution of flows based on the amount of transferred bytes is demonstrated in Figure 4.3a. The same distribution in terms of transferred packets is shown in Figure 4.3b. Note the logarithmic scale of the Y axis. It is clear, that the dataset contains many short flows. This increases a possibility, that the proposed anomaly detector can flag longer flows as anomalous, if the data is not pre-processed or normalized accordingly. Both flow counts in bytes and packets are selected features for the developed anomaly detector, as they provide an insight into the general shape of the traffic.

The longest flows in both bytes and packets in Figure 4.3 are caused mainly by MQTT flows, which often last for multiple days. Suricata considered the continuing MQTT conversations as a one flow, instead of splitting it into more flows. Other large flows are mainly HTPP and TLS flows, which downloaded large files.



(a) Histogram of flows by number of bytes in a flow

(b) Histogram of flows by number of packets in a flow

Figure 4.3: Flow distribution. Note that X scale (10^8, 10^6).

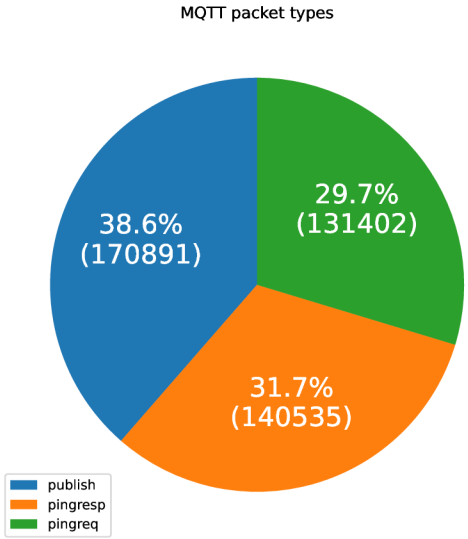


Figure 4.4: MQTT message types in explored data.

4.2.2 MQTT Event Type

The MQTT protocol takes the second place when measuring by number of flows (see Figure 4.2, despite the fact that the traffic is generated by only six distinct devices. The MQTT is a message oriented transport protocol based on the publish/subscribe model. It is designed to be used as a common communication protocol between various Internet of Things (IoT) devices or other Machine to Machine (M2M) contexts [3]. Captured traffic consisted of messages exchanged between two control PCs and four smart appliances, such as an air quality measuring unit, coffee machine or a refrigerator. Percentages for various MQTT events are shown in 4.4. Three most occurring message types are distributed uniformly. Note, that only message types with more than 1 000 occurrences are shown in the graph. Messages with lower number of occurrences are not shown in the figure. These are: connect, connack, subscribe and suback. Only 38.6 % of all MQTT flows are used to carry useful data—these have the *publish* type. All MQTT communication originated and terminated at two nodes in the internal BUT network.

For the records with MQTT event type are prevalent in the dataset, features extracted from MQTT are used for anomaly detection.

4.2.3 DNS Event Type

The top queried DNS servers and their top queried addresses for resource records of all types are shown in Table 4.3. Two devices in the original dataset (A.A.A.A, B.B.B.B) served as a prominent DNS servers. For this reason, DNS type and flags were extracted as features for anomaly detection dataset. Deviation from the commonly seen DNS related values could be thus predicted by the model.

Dest. IP	Query	Query Count	Dest. IP	Query	Query Count
A.A.A.A	aaa.com	27	D.D.D.D	bbb.cz	669
	iii.com	13		aaa.com	468
	bbb.cz	12		iii.com	252
	ccc.com	9		ddd.com	219
	ddd.com	9		lll.cz	202
B.B.B.B	eee.com	1	E.E.E.E	aaa.com	6
	fff.cz	1		mmm.com	4
	ggg.com	1		nnn.com	4
	hhh.com	1		ooo.cz	4
C.C.C.C	bbb.cz	193		ppp.com	4
	aaa.com	117			
	iii.com	75			
	ddd.com	65			
	kkk.com	56			

Table 4.3: Top DNS queries in examination dataset.

Source IP	No. Flows
A.A.A.A	94567
B.B.B.B	66164
fe80::1	4167
E.E.E.E	254
fe80::2	217
fe80::3	63

Table 4.4: Top DNS talker source addresses based on number of flows.

The top talkers by source IP address are shown in Table 4.4. Information in DNS logs can be used for both anomaly detection and reputation scoring. Counts of various DNS events, queries, and answers all can be used as features for anomaly detection. For example, if an endpoint normally queries the name A.A.A.A, and suddenly switches to B.B.B.B, the anomaly detection can potentially flag this as an anomaly, if the change is relatively drastic to the rest of the dataset, depending on the features used for model training.

For reputation score, information about queried domains can be utilized to penalize the resulting scores, if given domains are listed in a well-known source of blocked, malicious,

or unethical domains. By using domain geolocation, the score can be dropped if a node is located in a country with known security risks, or if it predominantly communicates with other nodes from these countries. This also applies to domains known for spreading misinformation, hoaxes, or which takes part in informational cyberwarfare. Lists of blocked domains, like the Current List of Administratively Disabled Domains[2] by cz.nic, rankings like National Cyber Security Index[3] or studies including the one by Comparitech [8], can be used as source of such information. Scanning attacks can be detected from DNS logs by tracing the patterns in a node behaviour. If a node suddenly queries a DNS server for all resource records for a particular domain, this could suggest that a form of network scanning is in progress.

For the large amount of DNS data in the exploration dataset, DNS related fields were chosen as features for the anomaly detection.

4.2.4 TLS and HTTP Event Types

Due to the encryption used by TLS, internal content of messages cannot be learned from this traffic. Indirect methods, like geolocation or information obtained from certificates, can be used for primarily for purposes of reputation rating.

For unencrypted HTTP, features like the HTTP content type can be used in anomaly detection. User agent field is valuable for both anomaly and reputation scoring. A node, using predominantly one agent and then switching to another one, can possibly trigger an anomaly alert. User agent strings can be utilized for reputation scoring. Some tools for network reconnaissance like Nikto include string `Nikto` in the requests generated by Nikto by default [21]. It can be argued for lowering the reputation score for nodes performing network reconnaissance. Detecting an old, outdated user agent, vulnerable to modern threats, creates a reason for lowering the node reputation score. However, user agent strings can be easily changed and manipulated with by an eventual attacker, and thus relying solely on them is not advised. Today, modern web browsers hide or send false user agent strings to deflect possible blockings of webpages or their incorrect rendering [58].

Features extractable from HTTP logs including *hostname*, *user agent*, *content type*, *method*, or*status* can be used as an input for anomaly detection model.

Similarly to queried domains in DNS, target hostnames or URLs can be used to detect connection to questionable or malicious websites. Common attacks against web servers, including SQL injection often includes malicious payload in the `application/x-www-form-urlencoded` format. Searching for common patterns in URLs can thus help to better evaluate the final reputation score.

4.2.5 Anomaly and Alert Event Type

There were not many occurrences of the anomaly or alert event type records present in the span of one week, for which the data analysis was concluded. Anomaly and alert records are thus evaluated for the entirety of the original dataset (25th October 2022 to 31th January 2023). During this time, a total of 26 896 anomaly records were detected together with 1 827 alert records. It is important to mention, that anomalies as reported by Suricata are not the same anomalies as understood in the context of this thesis. Anomalies in this

[2]Czech: aktuálně administrativně vyřazené domény: https://www.nic.cz/page/4310/aktualne-administrativne-vyrazene-domeny/ accessed [2024-04-02]

[3]National Cyber Security Index: https://ncsi.ega.ee/ncsi-index/ accessed [2024-04-02]

Prot.	Anomaly Event	Count
HTTP	REQUEST_AUTH_UNRECOGNIZED	732
	REQUEST_BODY_UNEXPECTED	76
	UNABLE_TO_MATCH_RESPONSE_TO_REQUEST	75
	MISSING_HOST_HEADER	30
	REQUEST_HEADER_HOST_INVALID	8
	REQUEST_LINE_INCOMPLETE	5
IKEV2	WEAK_CRYPTO_AUTH	816
	WEAK_CRYPTO_DH	775
	WEAK_CRYPTO_PRF	410
	WEAK_CRYPTO_ENC	52
	WEAK_CRYPTO_NODH	11
SMB	NEGOTIATE_MALFORMED_DIALECTS	23
	MALFORMED_DATA	7
SMTP	APPLAYER_DETECT_PROTOCOL_ONLY_ONE_DIRECT.	10
SSH	INVALID_BANNER	19956
	LONG_KEX_RECORD	447
	APPLAYER_DETECT_PROTOCOL_ONLY_ONE_DIRECT.	21
	APPLAYER_MISMATCH_PROTOCOL_BOTH_DIRECT.	3
TLS	CERTIFICATE_INVALID_DER	3376
	INVALID_RECORD_TYPE	23
	INVALID_SSL_RECORD	23
	ERROR_MESSAGE_ENCOUNTERED	12
	APPLAYER_MISMATCH_PROTOCOL_BOTH_DIRECTIONS	2

Table 4.5: Anomaly records reported by Suricata.

thesis are data points not conforming to a normal state of traffic. Suricata anomalies are flows, for which Suricata could not find a matching signature, or which somehow violate the expected communication patterns.

The anomalies are grouped into six groups by the application protocol. All reported anomaly events are shown in Table 4.5. The table shows all records which were marked anomalous by Suricata. They are grouped by the application protocol used. HTTP anomalies generally relate to malformed packet. IKEV2 alerts are caused by the use of insufficient cryptography. Most SSH anomalies are caused by the invalid SSH banner. This event is caused by the SSH banner having characters deemed invalid by Suricata.

Alert records are generated by Suricata if the traffic is matched against any signatures loaded into the Suricata program. The signature records can also contain data about an attack source and destination, as well as metadata such as affected operating systems. Detected alerts, their categories, signatures, and counts are shown in Table 4.6. For the alerts, six unique alert signatures separated into three categories were reported by Suricata. The categories are: Attempted Denial of Service, Detection of a Network Scan and Generic Protocol Command Decode. For example, the most common signature in the *Attempted Denial of Service* category implies, that there could be potential denial-of-service attack using SSDP protocol in the network.

Two features for anomaly detection are based on the anomaly and alert event types. These are *category of an alert* and *L4 protocol pro anomaly*.

Category	Signature Name	Count
Attempted Denial of Service	ET DOS Possible SSDP Amplification Scan in Progress	110
	ET DOS Possible NTP DDoS Inbound Frequent Un-Authed MON_LIST Requests IMPL 0x03	9
	ET DOS Possible NTP DDoS Inbound Frequent Un-Authed MON_LIST Requests IMPL 0x02	2
Detection of a Network Scan	ET SCAN Zmap User-Agent (Inbound)	6
Generic Protocol Command Decode	ET INFO WinHttp AutoProxy Request wpad.dat Possible BadTunnel	1694
	ET ATTACK_RESPONSE Possible IPMI 2.0 RAKP Remote SHA1 Password Hash Retrieval RAKP message 2 status code Unauthorized Name	6

Table 4.6: Alert records reported by Suricata.

4.2.6 Grouping by IP Adresses

The events cannot only be differentiated by event type, but also by other log fields such as source or destination addresses for analysis based on endpoints; or ports, if analysis per protocol is desired. For the purpose of computing the reputation and anomaly scores, the most important feature is the source IP address, since it is the main differentiator for

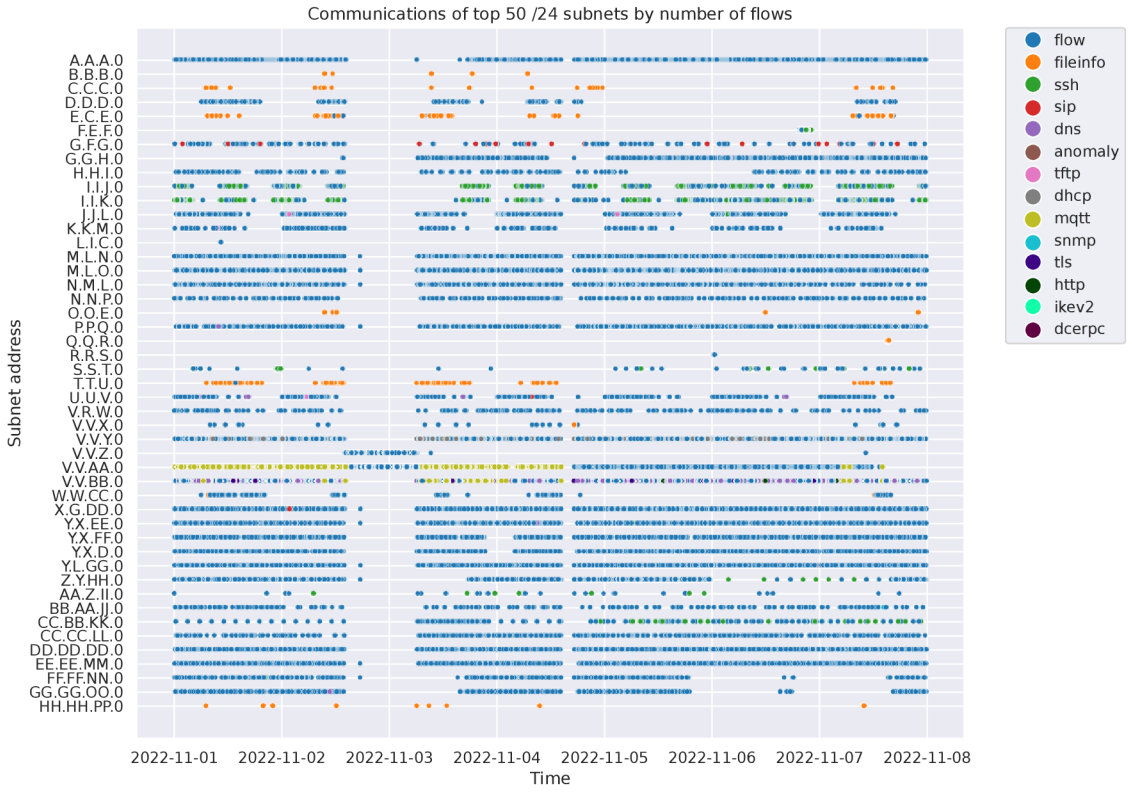


Figure 4.5: Top 50 source IP addresses ordered by the number of flows.

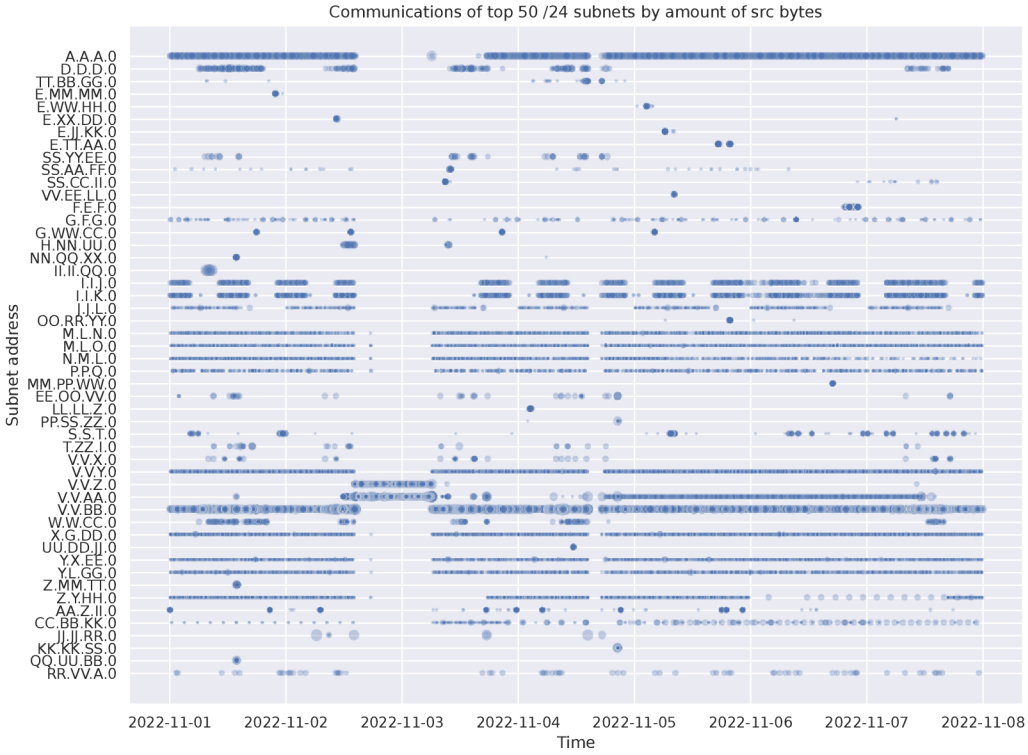


Figure 4.6: Top 50 source IP addresses ordered by the total number of bytes.

which an anomaly score can be computed - e.g. the score is computed for each IP address separately.

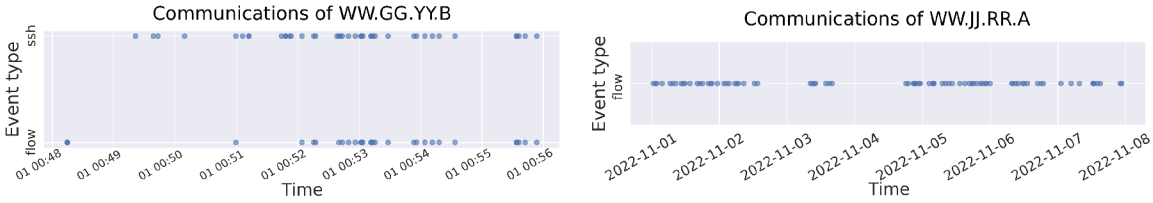
Figure 4.5 shows the top 50 IPv4 addresses by number of detected flows. The shade levels in the figure show the density of flows (darker colour represents more flows during a time period), while the colour of a data point corresponds to an EVE log event type. Data in this figure show great contrast between traffic from various IP addresses. Grouping records by IP addresses enables to calculate scores for every address separately without contaminating each other. This is valuable specifically for a pair of addresses defined by different behaviour. For example: Traffic from T.T.U.0 in the figure consists mainly of the *fileinfo* event type. If an SSH flow, previously unseen for this address, suddenly appears, anomaly detection should flag this event as anomalous for a given IP address.

However, problems could rise if the anomaly score was computed for all IP addresses as a whole. For example, the presence of many SSH flows in traffic of address A in the training data could result in marking SSH flow of address B as normal, although address B normally does not send any SSH traffic.

For B.B.B.0 or T.T.U.0, there were no flows detected at 2022-11-06 and 22-11-07. These days were Saturday and Sunday. The model should consider weekend traffic of these addresses as anomalies. If the anomaly score was computed from all addresses, then there could be addresses which communicated during the weekend. The model could then fail to detect the weekend traffic B.B.B.0 or T.T.U.0 as anomalies.

Figure 4.6 demonstrates top 50 IPv4 addresses by number of sent bytes. Shade levels show density of flows. The size of a data point represents the amount of bytes sent in one flow. Note, that the presence of an IP address in the figure do not necessarily mean, that the

address communicated throughout the entire examination period. For example, flows from the WW.GG.YY.B were detected only on the 1st November, in a span of eight minutes. However, the number of flows was greater than the number of flows of WW.JJ.RR.A, which communicated for an entire week. See Figure 4.7 as a reference. Figure 4.7a shows quick and isolated burst of flows, while 4.7b demonstrates continuous communication. For nodes, which do not communicate uniformly, the longer time window might be needed to evaluate their behaviour. The importance of average byte count in a flow can also serve as a feature, as they provide yet another metric trackable in time.



(a) WW.GG.YY.B communicated for eight minutes only.

(b) WW.JJ.RR.A communicated each day during a week.

Figure 4.7: Two different behaviours of an endpoint communication pattern.

The values illustrated for IP addresses in Figures 4.5 and 4.6, such as time distribution of flows, number of bytes, or detected events can be used as input for score calculation. The calculation can be carried out either for a) single IP address or b) range of IP addresses:

a) An IP address of an end node yields more precise results, if there is enough data about a node. When data is sparse, it might be impossible to calculate the score at all, or the calculated value might not be entirely representative of the real behaviour of the node. This can be mitigated by using an aggregation of addresses by subnets, described in b.

b) The network address together with a prefix or netmask clearly describes a range of IP addresses. The anomaly or reputation score is then calculated for a block of IP addresses. Data from all included nodes is used for a score calculation, reducing the need for having extensive data for each endpoint. The technique works well for smaller subnets, which usually belong to a single organization. This aggregation cannot be done indefinitely. Large aggregation would group many autonomous systems. Calculating the score from large networks can result in skewed results. The optimal network range can vary depending on the desired outcome. The size of the optimal range is to be decided by testing, and is specific to an environment.

Possible use of aggregation is shown in Figure 4.8. The figure shows communication of addresses which participated in a port scanning attack. The attack cannot be inferred from the traffic data or the figure alone. All IP addresses in the figure demonstrate similar traffic patterns. These addresses can thus be aggregated and treated as a single subnet. Traffic originates from subnet 89.248.165.0/24. This subnet belongs, according to WHOIS database, to Recyber project. The Recyber project web[4] claims that its intentions are not malicious. Whether is this statement true or not, the nodes in this subnet are caught doing network scans. Thus, the calculated reputation score should, in theory, be reduced. In this

[4]Recyber project: https://www.recyber.net accessed [2024-04-02]

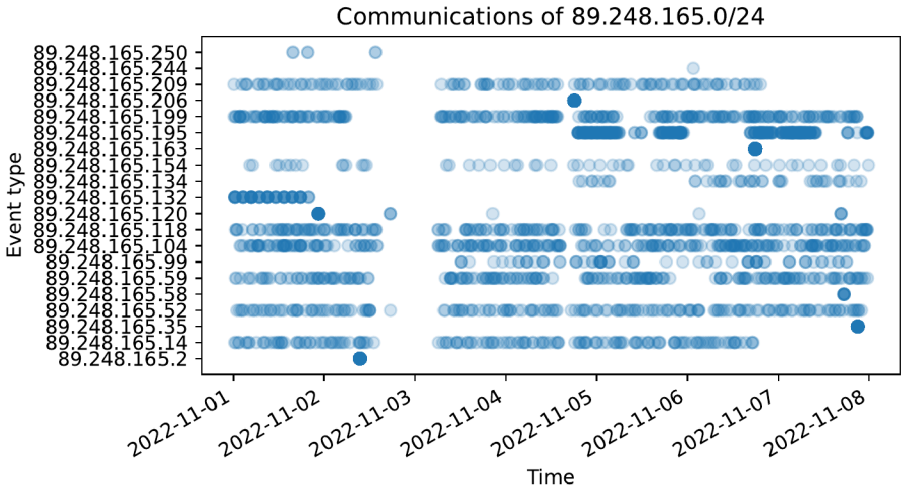


Figure 4.8: Traffic originating from 89.248.165.0/24 subnet.

case, aggregating nodes to a subnet with /24 prefix results in no data loss since all nodes belonging to that /24 subnet are demonstrating essentially the same traffic. The calculated score should have higher weight, since more data points were used for the calculations.

4.3 Summary

This chapter demonstrates the nature of Suricata dataset used in this thesis for model training and testing. It described data of which the dataset consists. Suricata EVE logs containing enhanced flow data were described. Most present event types in the data were presented to a reader with selected examples, graphs, tables, and commentary. Features extracted from EVE logs for model training were discussed for event types described in the chapter. Sensitive data shown in this chapter were first anonymized. The presented analysis explains nuances and thoughts behind selecting precisely these features. It was important to carry out this analysis, for the selection of the features, understanding of the dataset and evaluation of results according to the training data would be impossible without it. The number of flows per protocol, distribution of flow lengths and security related data is necessary for the result evaluation.

Chapter 5

Design of a System for Anomaly and Reputation Scoring

Chapter describes the inner workings of the proposed system in detail, including its general architecture, data preprocessing and manipulation, calculating the desired anomaly and reputation scores, to ranking the log file events using the score. The comparison of selected methods for anomaly processing is demonstrated, even though the final system uses only the best performing solution.

5.1 Anomaly Detection

This section talks about processes which had to be done to compute anomaly scores and to pick out anomalous events from Suricata EVE logs. It starts with a description of preprocessing steps, goes through transformation and normalization, the scoring method itself, and it finally demonstrates the way of reducing the amount of log events using the calculated scores (Section 5.1.4). All steps concerning anomaly detection described in this section are seen in Figure 5.1. References to this figure are made throughout the following paragraphs. The calculation is split into two main parts. Data preprocessing has the green background in the figure, while the scoring itself is blue.

5.1.1 Data Preprocessing for Anomaly Detection

This section aims to introduce a reader to the process of extracting useful information from raw Suricata logs. This data is then normalized and processed by the score calculating procedures.

The features to extract were chosen in order to contain valuable information while reducing the dataset size for further processing by scoring functions. Chosen features were of two main categories — categorical and quantitative. The categorical attributes are later converted into quantitative feature, since many algorithms and processes for anomaly detection (like k-means) require quantitative data only. During log extraction, there is no appropriate way to normalize data, because normalization typically works over an entire dataset. Only a subset of a dataset is available at a given time during the feature extraction. The entire dataset is available later, after the whole extraction of data from logs is completed.

The feature selection was based on features, which were selected by authors referred about in Section 2.4, in works by Henriques et al. (Sec. 2.4.1), Zhaoli et al. (Sec. 2.4.2),

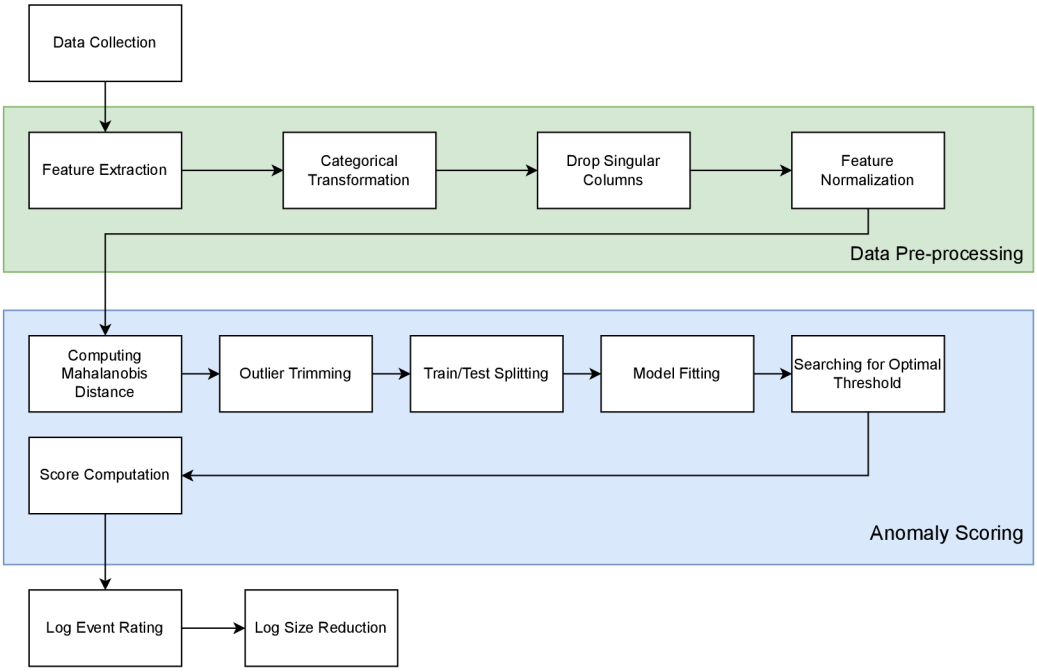


Figure 5.1: Processing pipeline for anomaly detection

Mehta et al. (Sec. 2.4.3), and Catillo et al. (Sec. 2.4.4). Importance of features as perceived by the author also played a role.

Extraction Algorithm

The selected algorithm for data extraction is simple. It iteratively reads data from logs stored in a predefined location. For each of these log, an internal dictionary called `flows` is created. The log file is then scanned line by line. Data is extracted from each line and stored in the aforementioned `flows` dictionary with the `flow_id` as a key, and it is filled with information depending on the `event_type` value of the record. The fact that all records adhere to standardized format across the dataset enables to select features deterministically.

Every flow inside the log file has at least one record. If the record type of the flow is `flow`, then general properties of this flow are stored in the `flows` dictionary and the type is set as flow. General features are listed in Table 5.1. Number of packets and bytes sent towards the server and towards the client are flow specific features. The Suricata system determines the correct direction by port numbers.

a *source IP address* is later used as an identifier, for which a score is computed. *Day, hour*, and *duration* features cannot be directly extracted from log records, but are computed from items included in the log record. The extraction of an hour in a day and a weekday name/number is considered important as the traffic size generally differs in various days of a week.

a record belongs to a specific protocol or special type, if its type is not `flow`. For each protocol with considerable amount of traffic (see Figures 4.1a and 4.1b) or a protocol with interesting data inside. protocol specific features are added to the correct item in the `flows` dictionary. It is important to know, that Suricata can produce more than one record with the same flow ID. If it detects no particular protocol, a record of type `flow` is generated. If a protocol is found, it creates the general `flow` record and a protocol specific record with

Feature	Feature type
Type	Categorical
Day (in a week)	Quantitative
Hour	Quantitative
Source port	Categorical
Destination port	Categorical
Application protocol	Categorical
Number of packets sent to the server (Source IP)	Quantitative
Number of packets sent to the client (Destination IP)	Quantitative
Number of bytes sent to the server (Source IP)	Quantitative
Number of bytes sent to the client (Destination IP)	Quantitative
Source IP address	Categorical
Flow duration	Quantitative

Table 5.1: Flow specific features

the same flow ID. This was utilised to map protocol specific values to general values already existing in the `flows` dictionary and vice versa. The list of chosen protocol specific values with explanation is shown in Table 5.2.

Protocol/special type	Feature	Feature type
HTTP	Hostname	Categorical
	User agent	Categorical
	Content type	Categorical
	Method	Categorical
	Status	Categorical
	Length	Quantitative
DNS	Flags	Categorical
	Type	Categorical
MQTT	Host	Categorical
	Message type	Categorical
Anomaly	Protocol	Categorical
Alert	Category	Categorical

Table 5.2: Protocol specific and special features

After all records in a log file were processed and assigned to the corresponding key in the `flows` dictionary, this dictionary is sent to a procedure which converts it to CSV records, and saves it as CSV files in a predetermined location. CSV files are created separately for each source IP address per date. Thus, one IP address have more CSV files when it communicates within more days. The naming convention for the resulting CSV is *YYYY_MM_DD_IP_anomaly.csv*. If a CSV file for a given IP address and day already exists, new records are only converted to CSV format and appended to an existing file.

The formalised algorithm for log data extraction is demonstrated in Algorithm 1. Note that in the resulting program, the algorithm is further divided into multiple functions and includes more conditions for filtering unwanted lines, etc. Statistical records described in Section 4.1.2 are an example of unwanted lines.

In the picture of data processing pipeline in Figure 5.1, feature extraction is at the beginning.

Algorithm 1 Data extraction from logs

```
L ← list(log_files)                                   ▷ Load all log files
S ← list(protocol_specific_features)                  ▷ Init list of proto. specific features
T ← list(flow_specific_features)                      ▷ Init list of flow specific features
for l ∈ L do                                          ▷ For each log file l in all logs L
    F ← {}                                            ▷ Init flow dictionary F
    R ← lines(l)                                      ▷ Load all lines in log file l
    for r ∈ R do                                      ▷ For all log lines r
        if r.flow_id not in F  then                   ▷ Init new item in F dict. Flow ID = key
            F.flow_id ← r.flow_id
        else if r.event_type = flow then              ▷ Add flow specific features to the records
            F.flow_id.attr ← r.T
        elser.event_type ∈ S                          ▷ Add protocol specific features to the records
            F.flow_id.attr ← r.S
        end if
    end for
    C ← to_csv(F)                                     ▷ Transfrom flow dictionary F to CSV
    save_to_disk(C)                                   ▷ Save CSV to disk
end for
```

Working Example

Examples of original records in a Suricata generated log and the corresponding line in the output in the CSV file are listed below.

In the following flow records, only flow specific features were populated, as the event type is flow. Other features are populated with the default value of -1. Suricata was not able to determine the specific application protocol of this flow. This record was saved into the EVE log due to a timeout in communication. Flow timeouts are described in the official documentation[1]


```
{
  "timestamp": "2022-10-26T17:03:16.577697+0200",
  "flow_id": 1885783963288316,
  "in_iface": "eth",
  "event_type": "flow",
  "src_ip": "X.X.X.X",
  "src_port": 58921,
  "dest_ip": "Y.Y.Y.Y",
  "dest_port": 5355,
  "proto": "UDP",
  "app_proto": "failed",
  "flow": {
    "pkts_toserver": 2,
    "pkts_toclient": 0,
```


[1]Flow Timeouts in official Suriacata documentation: https://docs.suricata.io/en/latest/configuration/-suricata-yaml.html#flow-time-outs, accessed [2024-04-02]

```
      "bytes_toserver": 150,
      "bytes_toclient": 0,
      "start": "2022-10-26T16:59:13.344828+0200",
      "end": "2022-10-26T16:59:13.766202+0200",
      "age": 0,
      "state": "new",
      "reason": "timeout",
      "alerted": false
    }
  }
```


a resulting CSV line and its header is:


```
type,day,hour,src_port,dest_port,app_proto,pkts_toserver,pkts_toclient,
  bytes_toserver,bytes_toclient,duration,http_hostname,http_user_agent,
  http_content_type,http_method,http_status,http_length,dns_flags,dns_type,
  mqtt_host,mqtt_type,anomaly_proto,alert_category,ip

flow,1,2,58921,5355,failed,2,0,150,0,0.421374,-1,-1,-1,-1,-1,-1,-1,-1,
  -1,-1,-1,-1,X.X.X.X
```


The following record of TCP flow has the application protocol was determined. The protocol of the flow is HTTP, which corresponds to the destination port 80. Protocol specific features are under the `http` key. Flow specific features are not populated when the extraction algorithm encounters an HTTP record. They are populated when a flow record with the same flow ID is processed.


```
{
  "timestamp": "2022-10-26T17:10:22.821458+0200",
  "flow_id": 229369739830515,
  "in_iface": "eth",
  "event_type": "http",
  "src_ip": "X.X.X.X",
  "src_port": 49528,
  "dest_ip": "Y.Y.Y.Y",
  "dest_port": 80,
  "proto": "TCP",
  "tx_id": 8,
  "http": {
    "hostname": "registry.npmjs.org",
    "url": "/@web-types%2Fvue-router",
    "http_user_agent": "JetBrains IDE",
    "http_content_type": "text/plain",
    "http_method": "GET",
    "protocol": "HTTP/1.1",
    "status": 301,
    "redirect": "https://registry.npmjs.org/@web-types%2Fvue-router",
    "length": 0
  }
}
```


a resulting CSV line (header is same as in the previous example):

```
http,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,registry.npmjs.org,JetBrains IDE,
   text/plain,GET,301,0,-1,-1,-1,-1,-1,-1,X.X.X.X
```

5.1.2 Categorical Data Transformation and Normalization for Anomaly Detection

Data prepared by the data processing module are loaded into a Pandas[2] `DataFrame` structure for each IP address, since scoring is done per IP address basis. Each data point is assigned its date value obtained from its CSV file name.

When data is loaded from the CSV files into the scoring module, few operations have to be done before any score calculation or application of machine learning algorithms can be applied to the data. Categorical data has to be transformed to numerical (quantitative). Python Scikit Learn (henceforth Scikit) library, especially the `OrdinalEncoder` class[3], is used to convert all categorical dimensions to numeric values (list of integers). This method maps all occurrences of one category value in a dimension to a unique integer value ranging from 0 to n, where n is the number of unique values in the original dimension. Example data transformation by ordinal encoding is shown in Table 5.3a and Table 5.3b. Ordinal encoding algorithm is written in Algorithm 2.

(a) Data before ordinal encoding

Source Port	Dest. Port	L4 Prot.
54484	443	tls
53510	53	dns
53510	53	dns
54485	443	tls
4322	53	dns

(b) Data after ordinal encoding

Source Port	Dest. Port	L4 Prot.
45974	234	14
44903	317	3
44903	317	3
45975	234	14
33584	317	3

Table 5.3: Ordinal encoding. Data are a subset of a real dataset - explaining the high values of transformed data.

If there is any column in the input dataset with a singular value, it is dropped before further calculation. Such data would not have by its nature any impact upon the anomaly detection and it would interfere with the next steps (calculation of Mahalanobis distance).

After the categorical to numeric conversion, data normalization has to be carried out. Normalization is done with Scikit included method `sklearn.preprocessing.normalize`, with the results being in the *L2* norm. L2 normalization is performed along the individual features. An L2 norm (Euclidean norm) is calculated by Formula (5.1) for each dimension in dataset D. (dimension is a synonym to feature). n is the number of records in the dimension. A normalized value x_{dn} for a data point $x \in D$ for dimension d is given as a fraction of its value and L2 norm of the corresponding dimension (5.2). Norm of vector $\mathbf{x}$ is often written as $||\mathbf{x}||$. Equations and notation taken from [13].

All these operations are shown in the first row (green) in the pipeline diagram 5.1.

[2]Pandas library: https://pandas.pydata.org, accessed [2024-01-20]

[3]sklearn.preprocessing.OrdinalEncoder documentation: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html, accessed [2024-01-20]

Algorithm 2 Ordinal Encoding

Require: **x** ▷ List of values to be converted
  $I \leftarrow []$ ▷ Init empty list
  for $(i = 0; i < len(\mathbf{x}); i++)$ **do**
    if $x_i \notin I$ **then**
      $I.append(x_i)$ ▷ Pupulate list I with unique values from **x**
    end if
  end for
  $C \leftarrow []$ ▷ Init empty list for output
  for $(i = 0; i < len(\mathbf{x}); i++)$ **do**
    $C_i \leftarrow I.index(x_i))$ ▷ Position of key x_i in list I is the converted value.
  end for
  return C

$$||d||_2 = \sqrt{\sum_{i=1}^{n} x_i^2} \tag{5.1}$$

$$x_{dn} = \frac{x_d}{||d||_2} \tag{5.2}$$

5.1.3 Scoring Method for Anomaly Detection

The Principal Component Analysis (PCA) unsupervised learning method was chosen to determine anomalous points in the dataset. Two anomaly scoring approaches utilising PCA were implemented and tested. Each approach comes with both its strengths and weaknesses. These approaches are the *PCA bulk method* and *PCA time based method*. After evaluating both methods, the *PCA time based method* was chosen as for the final detection system.

Theoretical principles of PCA method are discussed in section 2.2.2. The entirety of operations in steps performed on transformed data leading to final score are shown in the second row (blue) of the pipeline Figure 5.1.

PCA Time Based Method

The main idea of this method is to split the sorted dataset into two parts by date. One part being for the training, or observing data; and the other part for testing, and validating data.

Split ratio of the dataset was set to 2/3. For one month (30 days) of data, there would be 20 days from which the model learns the normal state of traffic. Outliers are first stripped out of the training data. PCA model provided by `sklearn.decomposition.PCA` is fitted with the training data. Fitted model is then tested against testing data to detect anomalies. Testing data is composed of the other third of data (last 10 days in a month).

Outlier Trimming Using Mahalanobis Distance

The PCA method results are highly vulnerable to any presence of outliers in the training data, because such outliers are main contributing factors in variances, covariances, and

correlations [48]. To include these data points in the calculation have an impact on the solution — especially for components explaining the most variance in the data. For these reasons, outliers were trimmed out from the training data before PCA model fitting.

Data points were deemed outliers based on the Mahalanobis distance, see Section 2.2.2 for detailed explanation of the metric. Mahalanobis distance was chosen for its suitability for multivariate data. Euclidean distance could also be used, it is, however, not suitable for data with correlated features [40]. Mahalanobis distance was also successfully used together with PCA by Shyu et al. [48]. The metric value for the specific use case of outlier trimming is calculated for each data point x_i in the dataset D by the formula (5.3),

$$d_i^2(\boldsymbol{x_i}, \overline{\boldsymbol{x}}) = (\boldsymbol{x_i} - \overline{\boldsymbol{x}})^T \mathbf{S}_D^{-1} (\boldsymbol{x_i} - \overline{\boldsymbol{x}}) \tag{5.3}$$

where $\boldsymbol{x_i}$ is a data point selected from all data points $\boldsymbol{X}$, $\boldsymbol{X} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n})$, and $\boldsymbol{x_i} = (x_{i1}, x_{i2}, \ldots, x_{ip})$ where p is the number of dimensions. $\overline{\boldsymbol{x}}$ is the mean vector of dimensions of dataset D, $\overline{\boldsymbol{x}} = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_p)$. The matrix $\mathbf{S}_D^{-1}$ is the inverse covariation matrix of D. The equation is taken from [19].

The distance is calculated for all points, no matter if a given point belongs into the training or testing dataset. The entire dataset is then sorted in descending order by calculated Mahalanobis distance. The Y percentile of top values is marked as outliers. $Y = 0.5$ was chosen as it proved to yield best results in the testing. The model was experiencing the least amount of false positives and false negatives when using this value. Other values were tested, but they did not provide better results than $Y = 0.5$. This value was also used by Shyu et al. [48] for outlier trimming before using PCA to detect anomalies in their paper.

Outliers with high Mahalanobis distance were trimmed out from the training dataset and left in the testing dataset. These point were labelled as outliers in the testing dataset, which enables evaluation of results. For one moth of data, the first 20 days would be trimmed off the outliers. They would stay present in the latter 10 days.

PCA Model Fitting and Anomaly Scoring

a PCA model provided by Scikit `sklearn.decomposition.PCA` class was fitted by the training data. The best value for the number of principal components kept in the resulting dataset was decided to be two. The number of principal components to keep was determined by testing. Hyperparameter testing report is located in later sections.

Steps demonstrated in Algorithm 3 needs to be performed to calculate anomaly score. PCA is *linear dimensionality reduction using Singular Value Decomposition of the data to project it to a lower dimensional space.*[47]. As PCA is a linear transformation, an inverse operation to a dimensionality reduction exists. This inverse transformation creates a matrix with the original number of dimensions.

The PCA model is fitted using the training dataset. This step computes the predefined amount of principal components. The selected number of components is two, as a model with two components yielded the best results. Model fitting is the only step required in the training phase.

Several steps need to be performed to obtain anomaly score. The testing dataset can be represented as a matrix $\boldsymbol{X}$ composed of rows of the dataset (matrix rows) and dimensions/features of the dataset (matrix columns). The testing dataset in the form of matrix $\boldsymbol{X}$ is projected on the principal components learned during the training phase. New matrix $\boldsymbol{Y}$ is the result of this transformation. $\boldsymbol{Y}$ keeps the same number of rows as $\boldsymbol{X}$, but it has

lower number of columns than $\boldsymbol{X}$. The number of columns of $\boldsymbol{Y}$ is equal to a number of principal components specified during the training phase.

The matrix $\boldsymbol{Y}$ then serves as input for inverse transformation. The inverse transform can be written as a dot product of $\boldsymbol{Y}$ and matrix of principal components. Application of the inverse transformation creates a matrix $\boldsymbol{X'}$, which has the same shape as $\boldsymbol{X}$. If the number of principal components during training was equal to the number of dimensions of $\boldsymbol{X}$, then the inverse transformation would be lossless. Because a smaller number of principal components was chosen for training, there was inherent loss of information during transformation of $\boldsymbol{X}$ to $\boldsymbol{Y}$, and $\boldsymbol{Y}$ to $\boldsymbol{X'}$.

The loss of information can be represented by the mean square error between the corresponding data point values in the original ($\boldsymbol{X}$) and inverse transformed ($\boldsymbol{X'}$) dataset. A mean square error (5.4) is computed for each corresponding pair of data points $\boldsymbol{x} = (x_1, x_2, \ldots, x_n), \boldsymbol{x} \in \boldsymbol{X}$ and $\boldsymbol{x'} = (x'_1, x'_2, \ldots, x'_n), \boldsymbol{x'} \in \boldsymbol{X'}$. These values form a loss vector $\boldsymbol{l} = (l_i, l_2, \ldots, l_n)$, where n is the dataset size. In equation (5.4), n is a number of dimensions of dataset $\boldsymbol{X'}$, x_{ij} is the value of dimension j of original data point x_i, and x'_{ij} is the value of dimension j of inverse transformed data point x'_i. Equation (5.4) taken from [51]

$$l_i = \sum_{j=1}^{n} (x_{ij} - x'_{ij})^2 \tag{5.4}$$

Algorithm 3 PCA Anomaly Detection - Time based method

Require: Preprocessed numeric normalized dataset D, PCA number of components nc, Set of outliers O in D

$split \leftarrow \frac{2}{3}$ of all days in dataset
$train \leftarrow \{x : x \in D, x.date < split, x \notin O\}$
$test \leftarrow \{x : x \in dataset, x.date \geq split\}$
$pca \leftarrow PCA(nc).fit(train)$ ▷ Fits PCA model to training data
for each $x \in D$ **do** ▷ $x = (x_1, x_2, \ldots, x_p)$; p = number of dimensions in D
  $y \leftarrow pca.transform(x)$ ▷ Apply dimesion reduction for sample x
  $x' \leftarrow pca.reverseTransform(y)$ ▷ Approximate original x
  $l_x = \sum_{i=1}^{p} (x_i - x''_i)^2$ ▷ Sum accross all dimensions
end for
$\boldsymbol{l_n} \leftarrow \frac{\boldsymbol{l} - min(\boldsymbol{l})}{max(\boldsymbol{l}) - min(\boldsymbol{l})}$ ▷ Loss min-max normalization
$t \leftarrow max(\boldsymbol{l_n}) \cdot \frac{1}{2}$ ▷ Threshold is one half of max value in $\boldsymbol{l_n}$
$bt, bm \leftarrow IterThreshold(\boldsymbol{l_n}, t, O, D)$ ▷ Get best threshold, confusion matrix, see Alg 4
return bt, bm

a threshold value needs to be calculated, for deciding, whether a data point is anomalous. If a loss value of the point exceeds the threshold, it is considered an anomaly. Because PCA method cannot determine the best threshold value itself unlike other methods for anomaly detection [46, 47], an iterative method for optimal threshold setting was developed. First, all loss values $l_i \in \boldsymbol{l}$ are normalized to <0,1> range using Min-Max normalization with the following formula:

$$l_{i_{norm}} = \frac{l_i - l_{min}}{l_{max} - l_{min}} \tag{5.5}$$

The iterative threshold searching algorithm is demonstrated in Algorithm 4. The algorithm expects the initial value of the threshold, a vector of losses, testing dataset and records labelled as outliers in the testing dataset. The initial threshold value is set to one half of the maximal loss value.

Iteratively, for many threshold values, data points are marked anomalous or normal based on comparison of their loss value and the value of the threshold. Next, a confusion matrix[4] is computed from the thresholded data and the outlier labelled values. Outlier marking was done in the data preprocessing step. If the sum of false negatives and false positives is smaller than the recorded best value, the best value is overwritten; and the current threshold value is considered the best threshold value.

If a number of false positives is smaller than the number of false negative, then the threshold is too large and is divided by two for the next iteration. If a number of false positives is larger than the number of false negative, then the threshold is too small, and it is increased by its half in the next iteration. Algorithm iterates over various threshold values, unless the difference between the best threshold and the threshold in the current iteration is less than 10^{-10}, or the number of iteration reaches 100.

The condition for loop completion was tested for more than 500 unique IP addresses. None of these testing runs experienced result improvements after iteration 46.

Algorithm 4 Iterative threshold

Require: initial threshold t_i, anomaly labelled data O, normalized loss vector $\boldsymbol{l_n}$, testing dataset D

$bt \leftarrow t_i$
$bm \leftarrow \emptyset$
$t \leftarrow t_i$
while $|bt - t| < 10^{-10}$ or $iterations \leq 100$ **do**
  $\boldsymbol{a} \leftarrow \forall x \in D \begin{cases} a_x \leftarrow true & for \quad l_x \in \boldsymbol{l_n} > t \\ a_x \leftarrow false & for \quad l_x \in_{\boldsymbol{n}} \leq t \end{cases}$
  $cm \leftarrow confusionMatrix(\boldsymbol{a}, \boldsymbol{al})$
  if $cm[0][1] + cm[1][0] < bm[0][1] + bm[1][0]$ **then** ▷ false negatives + false positives
    $bt \leftarrow t$
    $bm \leftarrow cm$
  end if
  if $cm[0][1] < cm[1][0]$ **then** ▷ false positives ≤ false negatives → t too large
    $t = \frac{t}{2}$
  else
    $t = t + \frac{t}{2}$
  end if
end while
return bt, bm

The unknown ratio between outlier points in the training and testing dataset is one of the main advantages of this method. All outliers are labelled during the outlier separation. A pseudo-labelled dataset is acquired, which can be used to compute detection ratings such

[4] sklearn.metrics.confusion_matrix documentation: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html, accessed [2024-01-20]

as a confusion matrix. Therefore, the model cannot be tuned in such a way that it is overfitted and has no false negatives or false positives.

PCA Time Method Example

This method was tested on a dataset created from Suricata logs for data originating from one station located in the BUT network gathered during every day in January 2023. The dataset contains a total of 1 942 372 preprocessed log events and 24 dimensions.

After disregarding of dimensions with only one unique value, dimension number dropped to 19. Top 0.5 % of data points were selected by the highest Mahalanobis distance and marked as outliers. Total amount of 9 712 values were marked.

Whether a data point fell into the training or testing dataset depended on its date. If the date was 2023-01-21 or lower, and the data point was not marked as anomalous, the data point was added into the training dataset. The data point was included in the training dataset otherwise. The size of the training dataset was 1 525 604, and the testing dataset contained 408 221 samples.

Two principal components were computed from the training dataset. PCA transformations described above were performed on the testing dataset. Loss values were calculated for every testing data point. The maximal observed loss value was 0,663 while the lowest one was $3.99 \cdot 10^{-10}$.

After the loss values were normalized, the iterative algorithm selected the best threshold. The iterative algorithm selected the best confusion matrix (Figure 5.2) and corresponding threshold of $7.62939453125 \cdot 10^{-6}$. The F1 Score of the best model for this dataset is 0,82, while precision 0,95 and recall is 0,72. Out of 1 065 outliers in the training dataset, the model detected 842 of them as anomalous. The entire calculation without dataset loading took approximately 45 seconds on a system equipped with AMD Ryzen 5 1600 CPU and 16 GB of DDR4 2666 MHz RAM.

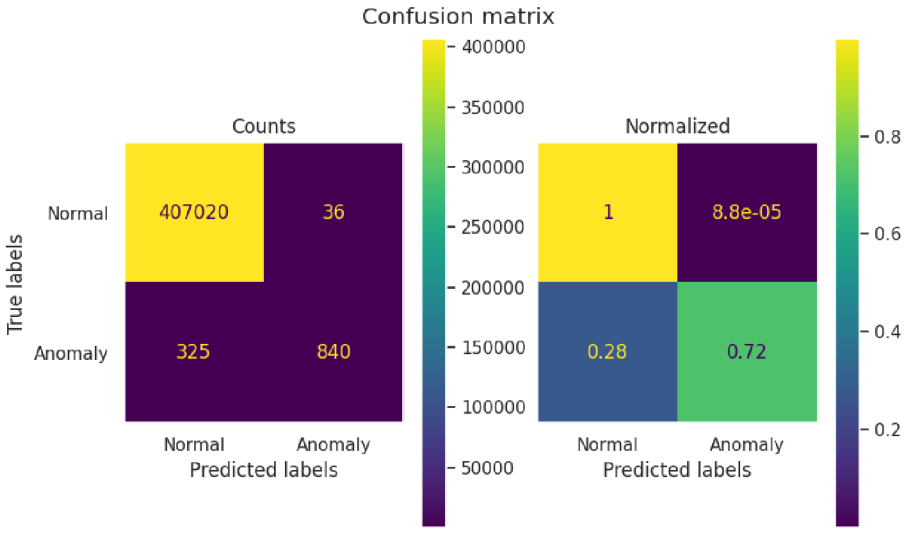


Figure 5.2: Confusion matrix for the example dataset.

PCA Hyperparameter tuning

To yield the best possible results, hyperparameters had to be selected. Hyperparameters are parameters whose values control the learning process and determine the values of model parameters that a learning algorithm ends up learning. Hyperparameters are selected by a programmer. [38]. Hyperparameters have to be chosen before model application, they cannot be manipulated afterwards. It is crucial to pick the best fitting values at the start.

The exhaustive combinations of hyperparameters and their effects on resulting model accuracy are demonstrated in Table 5.4. *mle* option in the *No. Components* column denotes the use of automatic choice of dimensionality for PCA by Thomas P. Minka [37]. *Singular Value Decomposition (SVD) Solver* hyperparameter selects the method used for decomposition of the dataset (matrix) into the principal components. *Whiten* specifies, whether the computed principal components are whitened or not. Whitening transformation transforms vectors of values with a known covariance matrix into new vectors, whose covariances are eliminated and variance normalized. [45]. Number of selected PCA components is the most dominant hyperparameter. Other parameters, shown in the table, have no influence on the computed scores. These parameters were thus left at their default settings.

No. Components	Whiten	SVD Solver	F1	FN+FP
2	True, False	arpack, randomized, full, auto	0.82	361
1	True, False	arpack, randomized, full, auto	0.82	371
3	True, False	arpack, randomized, full, auto	0.81	382
4	True, False	arpack, randomized, full, auto	0.81	382
5	True, False	arpack, randomized, full, auto	0.81	382
8	True, False	arpack, randomized, full, auto	0.79	412
7	True, False	arpack, randomized, full, auto	0.77	442
9	True, False	arpack, randomized, full, auto	0.75	492
6	True, False	arpack, randomized, full, auto	0.72	516
10	True	arpack	0.62	639
mle	True, False	auto, full	0.62	640
10	True	randomized, full, auto	0.62	640
10	False	arpack, randomized, full, auto	0.62	640

Table 5.4: F1 Scores and false negatives + positives with respect to PCA hyperparameters. Hyperparameters form the table header. Multiple options in one cell signifies, that any of the listed values can be used to obtain the same results.

PCA Bulk Method

PCA bulk method differs by the means of dividing the dataset into training and testing parts. PCA bulk method uses the entire dataset for both the training and testing. Two copies of the dataset thus have to be created.

Outliers are computed using Mahalanobis distance in the same way as was done in the Time method. The outliers are then removed from the training dataset, and they are kept in the testing dataset with the outlier label. The process of model training remains the same as for the Time method. Anomaly scoring is then done for the entire dataset, which contains all data including outliers. The scoring process is equal to the Time based method.

Using this method yields worse results, as even recent data points are included in the training dataset. Model is not able to detect anomalies caused by the difference in the recent and old traffic.

5.1.4 Log Size Reduction Using Anomaly Score

One of the goals of this thesis is to rate log events using the anomaly scores. This goal is achieved by linking the results of anomaly detection back to the dataset source—the logs. Each record rated by the anomaly scorer can be traced back to its original log file, thanks to having a full path to the original log and flow ID available for it. Since more lines of the log file can report different information about the same flow (see Chapter 4), usage of the flow ID as a key is preferable than using the line number of a record as a key. Inclusion of statistical logs in the Suricata EVE logs further complicates the line number approach, as these lines would be need to filtered out.

First, a dictionary (associative array) of flow IDs and original log paths as keys was created for each rated IP address. A calculated score, a threshold and a hard decision, whether the record with given ID is anomalous or not, were stored as values in this data structure. The original logs are then read into a dictionary and indexed by flow ID. Rated flows are matched against records in the original log files. If a match is found, information about anomaly is added to that record.

All lines of the log, no matter if they contain anomalies or not, can be saved into a file. There is also an option for saving only the anomalous records. Enabling the option greatly reduces the size of the output logs compared to their original counterparts. Often, the reduced logs contain only a handful of the original amount of records, decreasing the amount of records from hundreds of thousands to tenths of records. The largest difference in a number of records in original and reduced log file was more than 424 000, when the original file contained 424 133 of records, and the final file only 82 of lines. The largest log contained 379 records after reduction, while its original consisted of 15 809 records. Complete report of the testing is in Section 6.5.

5.2 Scoring Method for Reputation

Section describes all the necessary steps taken to compute a final reputation score from Suricata EVE logs. Suricata, the producer of logs, thus plays a role of an agent, which collects, preprocesses and provides data to the reputation system, which is the central node. The central node then computes the final score from preprocessed data available to it. This section talks about the preprocessing of data, computing daily score, which is symbolises a building step in computation of the final score, and the method for achieving the final reputation score for a network node from daily scores.

5.2.1 Data Pre-processing for Reputation Scoring

Suricata EVE logs provide data usable for calculation of a reputation score. This data can be divided into two categories – directly applicable information, and information which has to be combined with another external information in order to create a valid metric. this metric could be then used in a formula for scoring. Data for direct score computation are extracted from log events with the Anomaly or Alert event types.

Anomaly protocol, *type*, and *event* features are saved for the anomaly event type. *Anomaly protocol* is a string representing an L7 protocol of a given record. *Type* provides information about the cause of such anomaly record. Type can be of the following values: *decode*, *stream* or *app layer* [39]. *Event* is the most important of anomaly fields, as it represents the decoded name of the anomaly. Names decoded in the exploration dataset are described in Section 4.2.5. Again, it is important to mention, that the Anomaly event type of EVE logs represents different data than the anomalies scored by the method proposed in this thesis.

Alert-related events contain three useful features – *alert signature*, *alert category*, and *alert severity*. *Alert signature* is a text string containing a brief description of a given alert. Signatures used by Suricata are being periodically downloaded from an online database and loaded into Suricata processing engine. Examples of such signatures are described in Section 4.2.5.

Different signatures, belonging to the same topic/category have the same *Alert category* value. *Alert severity* is a number ranging from one to 255, although mainly values from one to four are used[5]. The lower the number, the greater severity of an alert is.

Information about DNS and HTTP events are indirect features used for reputation scoring. Indirect features are features, which values are not available in the dataset. They are obtained from external sources. All DNS queries coming from an IP address are examined during the pre-processing phase. Two features are populated by DNS data. These features are *type of query (A, AAA, TXT, etc.)*, and the *queried host name*. *HTTP host name*, *URL*, and *user agent* features are collected for HTTP events.

CSV file composed of the aforementioned features is created for an IP address and date. The reputation scoring module then reads these CSV files into a Pandas dataframe.

5.2.2 Daily Reputation Scoring

When CSV files are loaded, few settings have to be set before the scoring can be carried out. a date, for which a reputation score is to be computed, has to be defined. Interval of days needs to be selected. Data in this interval is used to calculate a reputation score for the previously chosen date. The last day of the interval must be the day, for which a score is computed. The scoring date can be set to the last date detected in the data, static date, or the date of calculation. The default value for the number of days from which to calculate the score has been set to fourteen. Fourteen was chosen as it captures the effects of ageing, and the 14 day interval is short enught, to be shown in figures. Supporting data which help to score the indirect events are downloaded from online sources and appropriate data structures are populated.

a daily reputation score is calculated for each source IP address detected in the data. The source IP address represents the original sender of the traffic detected by Suricata. It is therefore evident that the source IP address should be used as the identifier of an entity to be scored.

Daily score is a value representing the reputation score of a node computed from events recorded during one day. Daily score ranges from zero to one. The formula for calculation of the daily score is described by equation 5.6:

[5]Priority values used in Suricata rules: https://docs.suricata.io/en/suricata-6.0.0/rules/meta.html#priority, accessed [2024-02-19]

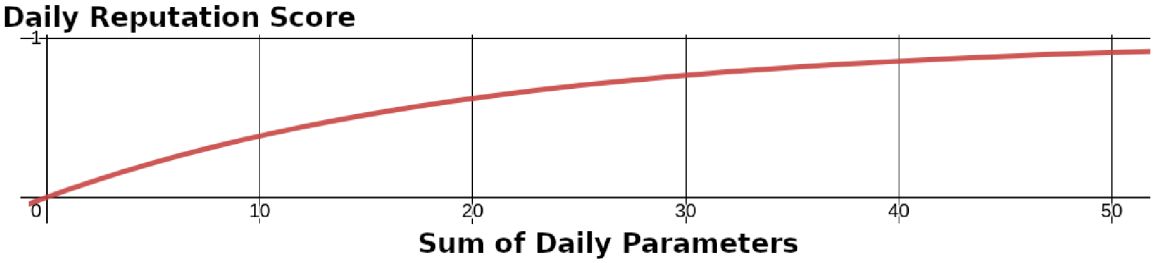


Figure 5.3: Relation between Sum of Daily Parameters and Daily Reputation Score. Both variables are dimensionless.

$$S_{daily} = 1 - \frac{1}{1.05^{(W_{AN}P_{AN} + W_{AL}P_{AL} + W_{DNS}P_{DNS} + W_{HTTP}P_{HTTP})}} \tag{5.6}$$

where P_{AN} is the parameter obtained from anomaly records, P_{AL} is the parameter computed from alerts records, P_{DNS} represents the DNS parameter and P_{HTTP} is the HTTP parameter. W_{AN} is the weight given to anomaly parameter, W_{AL} is the weight of alert parameter, W_{DNS} represents weight for DNS parameter and W_{HTTP} is the weight of HTTP parameter. Weights are dynamic parameters, which can be changed in settings. For the testing, weights were set to a default value of zero.

The constant value of 1.05 as the base was chosen for the parameters of the resulting curve shown in Figure 5.3. If the sum of daily parameters equals 50, the Daily reputation score is 0.913. The daily score reaches 0.99, when the sub score is 94. For the available dataset, this curve was able to differentiate between bad and good IP addresses. It was also suitable for keeping the reputation score stable, if a node behaved badly repeatedly, across multiple days. Results of the reputation scoring using the value of 1.05 are discussed in Section 6.2.

The value of daily reputation score ranges from zero to one. A score of zero represents a normally behaving IP address, without the presence of events which could lead to reputation score degradation. The more the score approaches a value of one, the more reputation it lost.

Methods for computing parameters are described below:

Alert Parameter

For computing the alert parameter P_{AL}, set U of unique signatures of *all alerts detected during a day* $\boldsymbol{A}$ is created. A sum of severities of all detected alerts is computed. Severities are inverted, since an event with severity value of one is the most dangerous. Suricata uses severity values between zero and four. The inverted severity value S_i^{-1} for alert $A_i \in \boldsymbol{A}$ is thus calculated as $S_i^{-1} = 5 - S_i$, where S_i is the severity of an alert A_i. The formula is shown in Equation (5.7).

$$P_{AL} = |U| \cdot \sum_{i=1}^{|\boldsymbol{A}|}(5 - Si) \tag{5.7}$$

Anomaly Parameter

Anomaly parameter is computed from the *list of anomalies detected in one day* $\boldsymbol{L}$. A set $\boldsymbol{S}$ is created from list $\boldsymbol{L}$, so every anomaly in the set is unique. Anomaly parameter is then obtained by multiplication of the magnitude of set $\boldsymbol{S}$ with length of list $\boldsymbol{L}$. The formula is shown in Equation (5.8).

$$P_{AN} = |\boldsymbol{L}| \cdot |\boldsymbol{S}| \tag{5.8}$$

DNS Parameter

DNS features extracted from the EVE logs represent together with HTTP dimension data which cannot be used on its own to compute reputation score. To be useful, they need to be matched with other information, not directly included in the logs. This additional information can be in form of various block listst. During the scorer initialization phase, lists of malign domains are downloaded from online sources. Chosen lists are:

- *Current List of Administratively Disabled Domains*[6] by cz.nic. This list is maintained by the .cz domain registry. It includes .cz domains, which are currently administratively disabled due to various problems, including *failed inspection of correctness of records of a registered domain, based on the verdict of law enforcement authorities, court decision, breaking registration rules of cz.nic*, or *decision of other public authorities (like customs administration)*. The list is refreshed hourly. As of February 2024, it contained 1464 domains. This list was chosen to be used for reputation scoring, as the original Suricata was deployed on premise of Brno University of Technology, located in Czechia.
- *Bad referrers*[7] list created for the *Nginx Bad Bot and User-Agent Blocker, Spam Referrer Blocker, Anti DDoS, Bad IP Blocker and WordPress Theme Detector Blocker project*. This list contains, for example:
  - Bad Referrers
  - Spam Bots and Bad Bots
  - Vulnerability scanners
  - E-mail harvesters
  - Content scrapers
  - Aggressive bots that scrape content
  - Government surveillance bots
  - Botnet Attack Networks (Mirai)

  The list is cooperatively maintained by several contributors via a GitHub repository [29], where a detailed description of the list, and its usage is located. Public can open a pull request to contribute changes to the list. The list contains more than 7 100 domains (February 2024).

[6] Current List of Administratively Disabled Domains: https://www.nic.cz/page/4310/aktualne-administrativne-vyrazene-domeny/, accessed [2024-02-04]

[7] Bad referrers list: https://github.com/mitchellkrogza/nginx-ultimate-bad-bot-blocker/blob/master/-_generator_lists/bad-referrers.list, accessed [2024-02-04]

Other lists might be used. For example, a bad domain list of Polish *CERT Polska* team [16], which belongs under NASK (Research and Academic Computer Network). This list contains more than 1 760 000 blocked domains as of February 2024. The list is available in multiple formats including CSV, JSON, or plain text. It is updated every five minutes. Submissions to the list are reported either by Polish telecommunication companies or by the public via a web form. Each submission is verified by at least two people from the CERT Polska.

This list was not used in reputation scoring due to its sheer size, since processing such a long list greatly impacted the run time of reputation scoring.

Hostnames in DNS queries included in the dataset are matched against the specified lists. If any match is found, the number of malicious events N is increased by one, and the host name is added into a set of unique matched host names $\boldsymbol{S}$. DNS parameter is computed as a product of number of malign events with a length of the host name set, as seen in Equation (5.9).

$$P_{DNS} = |\boldsymbol{S}| \cdot N \tag{5.9}$$

HTTP Parameter

HTTP parameter is computed similarly to the DNS parameter. Both lists referred to in the DNS section are also used for HTTP scoring. HTTP parameter is also affected by the *user agent* feature extracted from logs. *HTTP user agent* is a string symbolizing a program/script/utility which accessed the desired resource. There are lists of known malicious user agents, like the list [30] belonging to the same project as the one *Bad referrers* list from DNS scoring. The list contains more than 650 user agents (February 2024).

This list contains known web crawlers, bots, indexer software, web server scanners like *Nikto* or agents known to be used for other non-desired activities, like the *Mozlila* (sic) agent [18].

Set of malicious hostnames $\boldsymbol{S_h}$ is populated with hostnames detected in the logs, which were also found in any of the list of bad hostnames. A number of bad hostnames N_h is increased by one each time a nad hostname is found.

a set of bad referrers $\boldsymbol{S_r}$ is populated in the same way as list $\boldsymbol{S_h}$ is. Number of detected bad referrers N_r is also increased each time a bad referrer is found in a dataset.

Complete formula for HTTP parameter is shown in Equation (5.10)

$$P_{HTTP} = (N_r + N_h) \cdot (|\boldsymbol{S_r}| + |\boldsymbol{S_h}|) \tag{5.10}$$

5.2.3 Overall Reputation Scoring

When a daily score is computed for all days in the selected interval of days for traffic of a given IP address, one final reputation score has to be calculated for that IP address. The score is represented by the weighted average of daily scores. The final score S_f can be thus written as equation 5.11:

$$S_f = \frac{\sum_{i=1}^{n}(i)S_i}{\sum_{i=1}^{n} i} \tag{5.11}$$

where S_f is the resulting score. Day coefficient i ranges from one to n, where n is the size of the interval of days. The oldest day is $i = 1$, while the latest day has coefficient $i = n$. S_i then represents the score for a given day. The way of indexing days in the sum

operators ensures correct weighting of daily scores from the oldest, the least important, to the latest, the most important score. Assigning larger coefficient for more recent days ensures the correct ageing of the reputation score. Reputation score ageing is linear in this case.

On Interpretation of Reputation Score

There are two schools of though as to how a reputation score should be interpreted. Reputation score ranges from zero to one in both cases. However, the issue lies in determining which end of the value spectrum represents a positive rating and which represents a negative one. Currently, as the score computation is done, the value of zero represents a normally behaving network node, and a score near one means that the node has negative reputation. Since the method of scoring in this thesis is inspired by CESNET NERD system, where the same approach is used, it was decided to keep this interpretation of the score. Systems representing the other way of rating (e.i., higher score equals better reputation) are represented by the PageRank algorithm, or the SenderScore[8] (reputation of email senders). Cisco Talos uses words to describe the reputation (poor, neutral, good).

On Normalization of Reputation Score

When dealing with real world data, there is large chance of obtaining an unbalanced dataset. This raises a question whether to normalize data before scoring. Without any normalization, and using the number of alerts for scoring, the final reputation score of IP address X with 10 alerts and 10 000 total events would be the same as a score of IP address Y with 10 alert from 11 total events. If normalization takes place, then the address Y has much worse score than address X, although it participated in the same amount of malicious behaviour.

There were few considerations that had to be taken in account when designing a method for reputation scoring. First, only alerts, anomalies, or events that might lead to worsening of the resulting scores are extracted from the logs into the reputation dataset. Other traffic is not included in the dataset. Secondly, when an IP address produces just one event, reputation of such node should be decreased, no matter how large the amount of its positive historical data is. This scenario could emerge if, for example, a previously trustworthy node gets infected by a virus and starts to act maliciously. Moreover, a reliance on historical data is incorporated into the scoring function via the weighted average of scores from the past X days. Normalizing the dataset would, in this case, lead to stricter division between already rated IP addresses. It would be harder for an IP address to change its reputation score in time, as the great number of historical data would hide the recent changes in behaviour. This method of scoring also disallows the use of techniques to artificially keep the score low while performing malicious actions, like sending a large amount of artificially generated, but non-malicious traffic together with a much smaller amount of malicious flows.

For the reasons stated above, it was decided not to normalize the dataset.

5.3 Flow of Data After Scoring

For the entire process of scoring, matching scores to original logs, and log size reduction, the scoring system needs to load of preprocessed data, to score all entities (flows for anomaly

[8]SenderScore portal for email reputation: https://senderscore.org/assess/get-your-score/, accessed [2024-02-20]

detection, IP addresses for reputation), to save the scores to temporary files and then match computed scores to records in original logs. This process is shown in Figure 5.4.

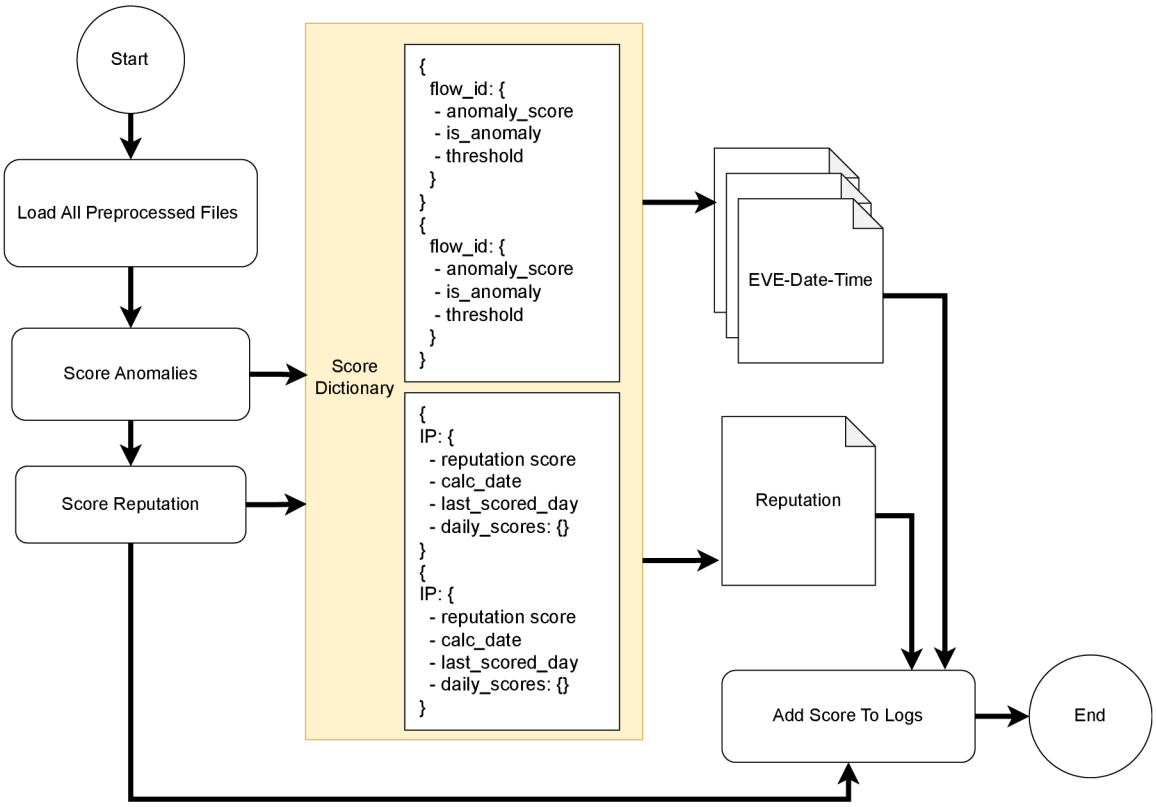


Figure 5.4: Flow of data from loading preprocessed data to saving reduced logs

The system is designed to perform every needed action only once, without cycles. First, preprocessed files are read from disk. The directory from which to read is specified in the settings file. The reading can be customised by specifying a date, from which to read data. Loading of datasets for both anomaly scoring and reputation scoring is done. Separate datasets are created for each IP address. Datasets are represented by Pandas Dataframe.

Anomaly scoring is performed after data is loaded. After an anomaly score is computed for flows of one IP address, these scores are saved into a dictionary of scores. The anomaly scores are saved with under the following compound key: *original log file name, flow ID*. An example of this dictionary key is *eve-2023-01-22-01_53.json, 188194883355008*.

Reputation score is computed after the anomaly score. Results of the reputation scoring are also saved into the same dictionary as the anomalies. Reputation score is saved under the *reputation_score.json, IP address* key.

Temporary files are created from the dictionary. One file is created per the primary key of the dictionary, e.g., *eve-2023-01-22-01_53.json, eve-2023-01-22-02_53.json, ..., reputation_score.json* files are created.

When matching calculated scores with original logs for reduction of records, first reputation scores are read from the *reputation_score.json*. The program then iterates over the pairs of original logs and files with scores. Anomaly scores are added to corresponding flow IDs in the original log. Reputation score is added to all flows with matching source IP

address. Finally, flow records with both scores above the threshold are saved into reduced log files.

Because of the *reputation_score.json* file, it is possible to read reputation scores without having to read all reduced logs. Location of all files can be specified in the settings file.

5.4 Summary

This chapter was split into two thematic sections, each talking about the specific implementation of an anomaly detection system, or a reputation system. In the anomaly part, it went through the process of feature extraction, with explanation of why given features were selected. Transformation of data from categorical to numerical values was described before explaining the normalization to L2 form. Since the available dataset was unlabelled, it was necessary to create pseudo-labels. This was achieved with outlier trimming. Outliers were detected using Mahalanobis distance. Two methods for anomaly detection based on the unsupervised PCA method were proposed. The method achieving better results (PCA time based method) was described with a practical evaluation on a testing dataset. Hyperparameter tuning for the PCA model was presented. In the end, the anomaly score was used to enhance the original EVE JSON logs.

a scoring method for reputation was proposed and implemented. The section about reputation scoring talked about the specifics of feature extraction for reputation, and the preprocessing of extracted data. It then describes the daily reputation scoring with mathematical formulas and presents parameters, of which the daily score is composed of. Reasons for using the various parameters were explained. The mathematical notion of computing the final reputation score from daily scores was shown. The discussion encompassed the interpretation of the resulting reputation score, as well as the contentions regarding whether a reputation score should be normalized.

Chapter 6

Experiments and Discussion

The aim of this chapter is to look critically at the resulting scores and reduce logs. Explanations of given behaviour of the scoring methods are provided by a commentary, figures, and tables. The chapter is divided into thematic groups, each dealing with one particular question or topic. The following topics are described: validation of reputation score against other working reputation systems, how the reputation score changes in time, the nature of anomalies detected by the developed system, whether there exists a correlation between anomalies and reputation, performance testing, and finally, how both scores are used to reduce amount of Suricata events in logs.

6.1 Validating Reputation Score Against Existing Solutions

Validation of reputation scores by comparison to other existing solutions is not easy, as the available data from online sources are often changing in time or they are calculated from recent historical records. The expectation prior to the writing of this section was that the reputation scores derived from the testing data would have been incomparable to those calculated by other tools. The reasoning is that the behaviour of Internet nodes has naturally changed from January 2023 to March 2024.

It was thus surprising, when a number of IP addresses scored by method developed for this thesis was also flagged with high reputation score by NERD[1]. in March 2024. Few of the scored addresses also have poor email sender reputation by Talos[2]. Selected IP addresses and their computed reputation scores are shown in Table 6.1. IP addresses from BUT IP address space are anonymized. Records above the dividing line are addresses belonging to a group of worst performing addresses. Their score in the table is the highest recorded score of respective address. Records below the line are addresses, which were rated by the system developed in this thesis, and which were rated as bad by NERD. These addresses reach high scores in NERD, because of their long lasting problematic behaviour. They are present in many black lists as reported by NERD. On the other hand, the presence of communication in the testing dataset was minimal, which explains their low reputation. The long list of *Other* column for IP address 190.171.189.85 stands for: The address is present in nine black lists, it participates in network scanning, it has ports 22, 25, 53, 80, 443 opened. There were also unauthorised automated login attempts incoming from that address. My reputation scoring did not assign worse score to 190.171.189.85, as there was not enough bad traffic

[1]NERD IP Search: https://nerd.cesnet.cz/nerd/ips/, accessed [2024-03-23]
[2]Talos Reputation Center: https://www.talosintelligence.com/reputation_center, accessed [2024-02-24]

Source IP	My experiments		NERD system - 12. 3. 2024				Talos system - 12. 3. 2024		
	Computed Score*	Date	Score	Added	Last Activity	Other	Web Rep.	Block List	Email Rep.
V.V.BB.QQ	0.908057	14.12.2022	-	-	-	-	-	-	-
76.223.92.165	0.618265	14.12.2022	-	2024-03-09	-	1 list, 443	-	-	-
13.248.212.111	0.594867	16.12.2022	-	2024-02-16	-	1 list, 443	-	-	-
V.V.BB.LL	0.302143	18.12.2022	-	-	-	-	-	-	-
V.V.BB.VV	0.218269	12.12.2022	-	-	-	-	-	-	-
139.59.152.202	0.210607	17.12.2022	0.000	2024-01-01	2024-02-26	3 lists, Scan, 22, 80	-	Expired	-
141.94.110.90	0.201445	08.12.2022	0.000	2024-01-01	2024-02-26	4 lists, Scan, 22, 8069	-	Expired	-
165.232.69.156	0.141505	21.12.2022	0.000	2024-01-01	2024-02-26	4 lists, Scan, 22, 8069	-	Expired	-
178.60.204.50	0.117610	13.12.2022	-	-	-	-	-	Expired	Poor
186.122.177.117	0.105949	06.12.2022	-	-	-	-	-	Expired	-
149.202.74.37	0.096894	01.01.2023	-	2024-03-11	-	3 lists	-	Expired	-
57.128.11.39	0.072859	22.12.2022	0.582	2023-11-19	2024-03-12	8 lists, Scan, 22, 80, 111, 8081	Untrusted	Yes	-
43.138.17.151	0.063309	05.12.2022	-	-	-	-	-	-	Poor
101.43.110.129	0.063309	18.12.2022	-	2024-03-11	-	1 blacklist	-	Expired	-
192.145.127.42	0.006803	19.12.2022	0.764	2023-08-15	2024-03-12	5 lists, Scan	Untrusted	Yes	-
139.28.218.34	0.006803	27.12.2022	0.500	2023-11-02	2024-03-12	7 lists, Scan	-	Expired	Poor
91.207.175.154	0.006803	03.12.2022	0.652	2021-11-23	2024-03-12	7 lists, Scan	-	Expired	-
185.245.86.226	0.006803	22.12.2022	0.750	2021-09-09	2024-03-12	10 lists, Scan, 123	-	Expired	Poor
190.171.189.85	0.006803	14.12.2022	0.0445	2023-10-27	2024-03-12	9 lists, Scan, Login attempts, 22, 25, 53, 80, 443, self-sign., iot, eol, database, starttls	-	Expired	-

Table 6.1: Comparative study of reputation over different reputation systems. *Note: the best score = 0 (trustworthy node), the worst score = 1 (totally untrusted node)

detected in Suricata. NERD has larger database of traffic where this address is probably more prominent. The *self-signed, iot, eol-product, database, starttls* options originate from Shodan's InternetDB[3].

The table shows information gathered from NERD and Talos for each address. Both NERD and Talos queries were made on 12th March 2024. If the address has a record in the *NERD Added* column, but is missing in *NERD Last Activity*, then it was never rated by NERD with a reputation score above zero, however the address is present in one of the block lists used by NERD. If the score is zero, the node took part in an offensive behaviour, however the model did not make a decision to increase the score. Information provided in the *NERD Other* column includes the number of blacklists in which the address is present, a list of ports opened on the machine, and other parameters, like *Scan*, which means that the IP address certainly participated in a kind of scanning attack. The three Talos columns describe the labels placed upon the IP address by Talos system. *Expired* in the *Block List* column signifies, that the address was once on a Talos black list, but was later removed from it. If that column does not have a value, the address never was on any Talos black list.

The reputation scoring of the thesis marked 68 addresses belonging to 87.236.176.0/24 network. Ten of these records ended with a non-zero reputation score. The score was however small, i.e., smaller than 10^{-3}. All of these 68 records are present in NERD. NERD evaluated them with scores between 0.207 and 0.436. All were added in NERD in September 2022 and scanning activities of all were recorded in March 2024. All these addresses are included in at least seven block lists.

This experiment proved the correctness of the chosen method for reputation scoring, as the results are comparable to other existing solutions. Both the developed method and existing solutions are able to mark the same IP addresses as malicious.

6.2 On Changes of the Reputation Score in Time

Looking at the reputation score in the span of multiple weeks, or even months, gives a better picture of what the score represents. Reputation score was computed for all IP addresses for all days in December 2022, and January 2023. Defaults settings of the scoring functions were used for evaluation in this section. The default settings are:

- The number of days (daily scores) used for calculation of the overall score: 14
- The weight of all sub scores: 1

Reputation scores of top 300 IP addresses by number of flows, of which only 130 were outside BUT, were evaluated. From these, only two addresses achieved score other than zero. This means, that 298 IP addresses had recorded events, which could potentially led to a reputation score increase. Resulting reputation score for such address was thus zero. Two addresses with most recorded flows and non-zero reputation score were V.V.BB.QQ and V.V.BB.VV. These also happen to figure in the list of ten addresses with the highest sum of reputation score across the examined time period, and are thus shown in Figure 6.1. V.V.BB.QQ and V.V.BB.VV addresses are in the V.V.BB.0/24 subnet which belongs to the top 50 subnets based on number of flows. V.V.BB.QQ and V.V.BB.VV are thus also represented in Figure 4.5 in Section 4.2.6.

[3]190.171.189.85 record in Shodan's InternetDB: https://www.shodan.io/host/190.171.189.85, accessed [2024-03-12]

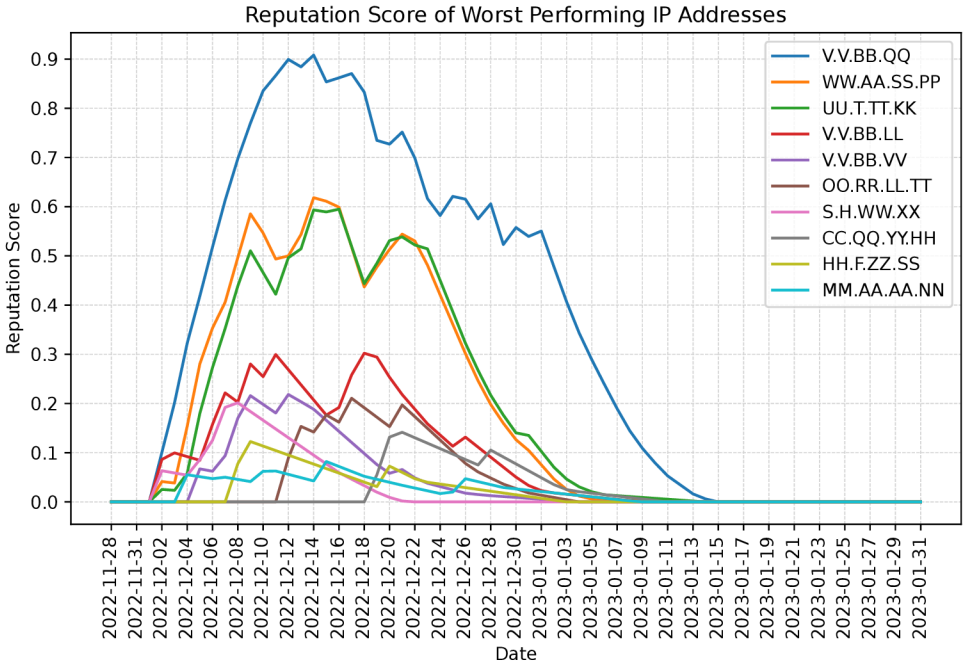


Figure 6.1: Time series of reputation score for the worst performing IP addresses.

V.V.BB.QQ has the worst reputation score of all tested IP addresses, with more than 350 000 events reported by Suricata in December 2022 and January 2023 (all records, no matter if malicious, anomalous, or normal). During the rising phase in Figure 6.1, from the start of December to 12th December, daily scores of V.V.BB.QQ were always above 0.7. Because of the formula taking scores of previous days into account, the score rose slower than that. The rise of the score and values of daily scores are demonstrated in Table 6.2. The table shows the cumulative property of the overall score, especially during the phase of score rising.

The overall reputation score raises slowly than the daily scores might suggest. This slow increase is explained by the use of weighted average of daily scores in the computation of the final score. Although the daily scores have values near one, the overall score raises slowly, approximately by 0.1 per day.

The fact is, that one day of bad behaviour cannot worsen the reputation score of a given IP address to its maximum. The bad behaviour should have to persist across multiple days for the score to reach its worst value, 12 days, by values presented in Figure 6.1. This figure also demonstrates the gradual fall of the overall score when no malicious events are recorded for several days.

V.V.BB.VV address is another address from the BUT address space. With the total of 1 943 000 EVE records, it belongs to IP addresses with the most records. Even though it has more records than V.V.BB.QQ, it keeps better reputation score.

Figure 6.2 shows score timeline for selected IP addresses. These addresses were chosen, because their reputation score is very small, but it is not zero. Reputation scores of these IP addresses demonstrate the change that happens, if only one alert is received for a given IP address during multiple days. The slower slope of reputation score after the peak values shows ageing of the score. Note the y scale, when making comparisons to Figure 6.1. The

Day	2022-12-01	2022-12-02	2022-12-03	2022-12-04	2022-12-05
Reputation Score	0.000	0.101	0.202	0.322	0.418
Daily Score	0.000	0.705	0.757	0.949	0.843

Day	2022-12-06	2022-12-07	2022-12-08	2022-12-09	2022-12-10
Reputation Score	0.518	0.613	0.697	0.770	0.835
Daily Score	0.928	0.965	0.958	0.944	0.964

Day	2022-12-11	2022-12-12
Reputation Score	0.867	0.899
Daily Score	0.790	0.858

Table 6.2: Reputation score and daily scores of V.V.BB.QQ IP address.

highest shown y value in Figure 6.2 is 0.009, while the maximal value in Figure 6.1 is above 0.9.

An objection to the scoring method can be brought upon analysing data in Figures 6.1 and 6.2. One might argue, that the reputation score is high for V.V.BB.QQ because of its large amount of traffic compared to other IP addresses (36 000 flow records in EVE logs). The counterargument to this statement is that there were also other IP addresses in the dataset with similar or higher volume of traffic but much lower occurrence of misbehaviour. They are 298 addresses out of 300, which are discussed at the beginning of this section. V.V.BB.VV with 1 943 000 events is the example. With more than 50 times the number

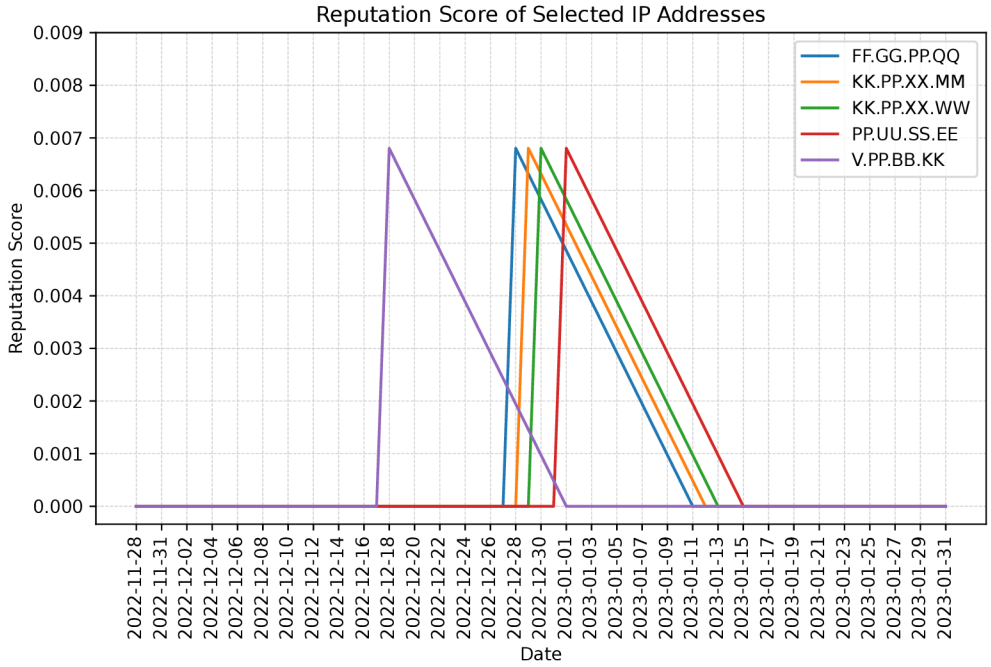


Figure 6.2: Time series of reputation score for selected IP addresses.

of events than V.V.BB.QQ, it has a better reputation score. Other arguments for using absolute numbers without normalization for scoring can be found in Section 5.2.3.

6.3 On Anomalies Found in Reduced Logs

Because of using unlabelled dataset for anomaly detection, a problem of validation arises. Since no predefined labels could be used to determine why a given point is marked as anomalous or normal by the model, a post-scoring analysis on the logs containing the computed score had to be done. Another reason for the necessity of performing this analysis is the nature of anomalies that the model tries to detect. Anomalies detected by the developed system are not malicious. Detected anomalies of the IP address represent data points, which are not in line with the past traffic of the given IP address. The analysis thus tries to evaluate the relation between the new and old traffic of the same IP address.

Because the anomaly score is computed for each IP address separately and the resulting score is normalized for every address, it does not make sense to compare scores or anomaly rates of multiple addresses together. The same applies to the threshold. Each IP address has different threshold for each day, which makes them incomparable. Threshold is different for each day, because it is calculated by an algorithm each day, and because the model is recomputed each day in order to include recent data. The algorithm tries to minimize the sum of false negatives and false positives. The algorithm is further described in Section 5.1.3.

Only log records of V.V.BB.VV address are taken into account for the analysis, as dataset made out of records with this address as source address was used for the model performance evaluation in Chapter 5. This IP address also has enough traffic, so its score should not be skewed by lack of data samples. Recorded events of V.V.BB.VV address for the last seven days of January 2023 are demonstrated in Figure 6.3. Most of the log events happened during the day, after 12:00. This behaviour is visible in the figure, when the events with such time have the lowest anomaly scores. Be mindful of the logarithmic Y axis, when examining the figure. For the large size of a dataset, the figure is composed of every tenth records (1:10) sampling.

The triangular shape created by events each day (in Figure 6.3) is caused by the amount of recorded events in various hours of the day. For example, between the 10th and 20th January (dates included in the training data), the mean of hour in a day is 11.86 with a deviation of 4.418 hours. Histogram of hour values across training data is shown in Figure 6.4. The values centred around noon with slight skew to the right in the histogram explain why the anomaly score in Figure 6.3 is the lowest each day at noon.

The following paragraphs explain the annotated anomalies in Figure 6.3:

A: Most of the events on 25th January with score lower than 10^{-4} are TLS flows with a long duration and large amount of transferred bytes and packets. The amount of packets sent or received ranges from several hundred to ten thousand. Duration of these flows are longer than 11 000 seconds. When compared to data shown in Chapter 4, Figure 4.3, it is clear why these flows reported higher anomaly scores. The score, however, did not exceed the threshold by much — by 10^{-2} at most.

Events with anomaly score in the range 10^{-4} - 10^{-3} consist of TLS or Flow records. Two of these anomalies were also caused by SSH connections to TCP port 22110. 125 records with this destination port were found in the original data. Total amount of ssh connection in the training data is 416, while the training data consisted of more than 1 400 000 records. SSH being such a small portion of the overall training data

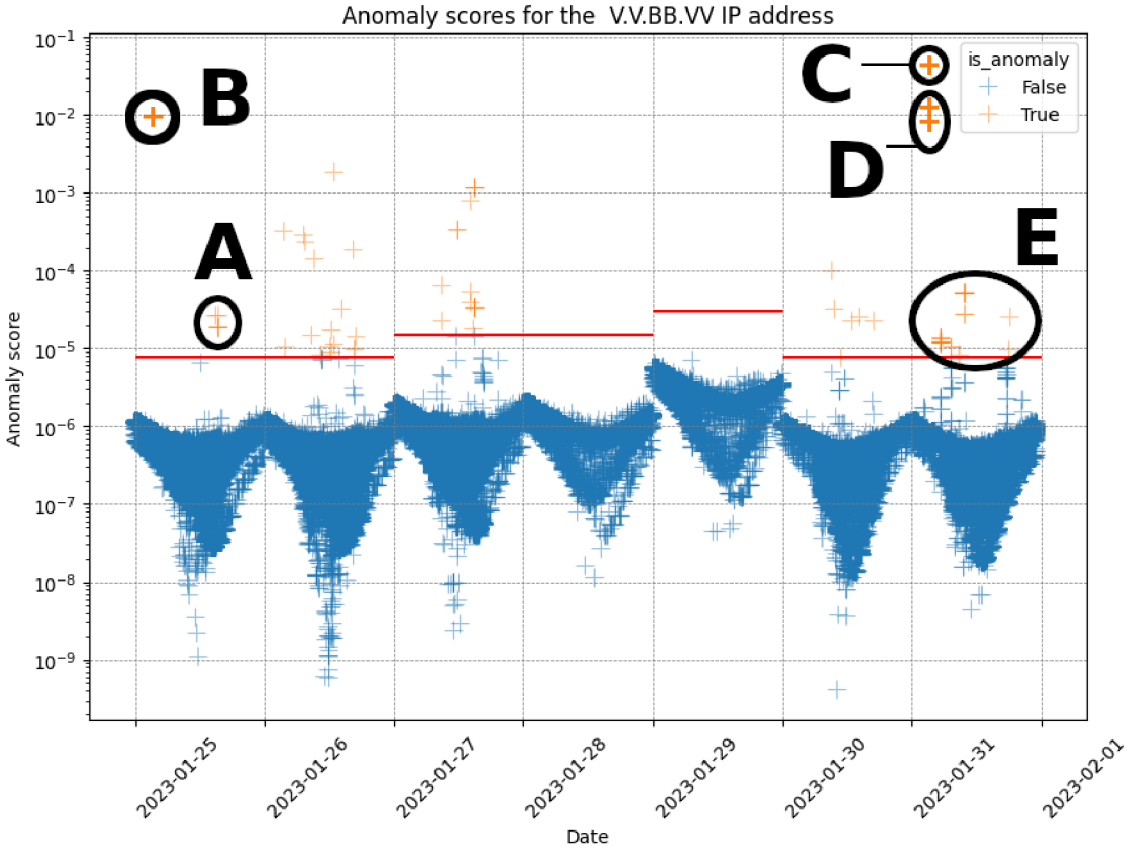


Figure 6.3: Log events of V.V.BB.VV for seven final days of January 2023. Threshold values are marked with red horizontal lines. Events with anomaly scores higher than the threshold are marked as anomalous by the model. Note the logarithmic scale on Y axis. Data in this figure are sampled in 1:10 ratio.

can explain, why such data points could be marked as anomalous. All of these flows also had longer duration than the most other flows in the training data.

B: The cluster with anomaly score value of 10^{-2} was in the entirety composed of 18 608 records with the same flow ID. These were MQTT records all captured during one second — at 04:16:05. 32 907 packets were sent from V.V.BB.VV, and it received 21 511 packets. 4.6 MB of traffic belonging to this stream was reported by Suricata. In 1 412 104 total records in the training data, only 416 records are MQTT. Most values for packets sent and received in the training data were below 1 000. This stream also had extremely long duration. Suricata reports, that this stream begun on 20th January at 04:54:50, and ended on 25th January at 04:15:36. After the end of this stream, one flow record and 18 608 MQTT records with conversation details were saved into the EVE JSON log, all with the same timestamp. This one long flow lasted for 42 964 seconds, which definitely surpasses the normal value measured for a flow duration. In the training data, the mean duration of a flow was 40,32, while its median was under one second.

C: Three most prominent anomaly clusters appeared on 31st January. All of these clusters are formed by MQTT flows with exceptionally large number of records in the

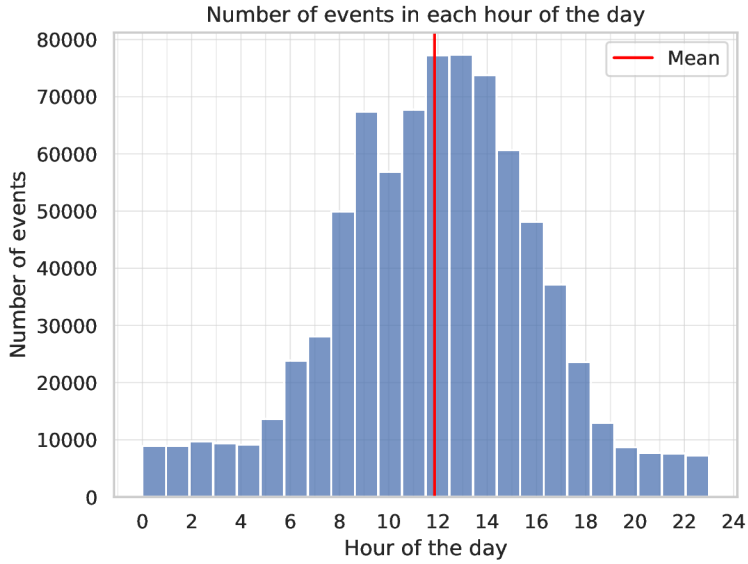


Figure 6.4: Histogram showing the distribution of records based on an hour of a day in the training data

.

EVE logs and long durations. The most prominent cluster (C) is composed of five different flows with the same anomaly score score of 0.043. Each of these flows was composed of more than 42 000 MQTT events, and each lasted for 947 188 seconds (10.96 days). All of them also sent and received similar number of packets and bytes. This explains their very similar anomaly scores. The only difference between the flows was a source port and the fact, that each flow sent information about a different device (two computers, a coffee machine, a refrigerator, and a network bridge). Nature of these streams is similar to the MQTT stream from 25th January, thus an explanation, why these streams are anomalous, is also the same.

D: Two streams near the 10^{-2} mark on the Y axis are also MQTT streams similar to the aforementioned ones. In comparison to the streams mentioned above, they demonstrate lower anomaly scores because of their shorter duration — 511 515 and 409 889 seconds. These two streams transferred information about a wireless access point and another coffee machine.

E: The anomalies detected on 31st January near the threshold line could be marked as false positives, as the difference between their anomaly score and the threshold is less than 10^{-5}. These flows are mainly of the HTTP or SSH type. All HTTP endpoints are websites belonging to Microsoft. Examples of endpoint hostnames are 1d.tlu.dl.delivery.mp.microsoft.com or b.c2r.ts.cdn.office.net. All connections were made using the Microsoft-Delivery-Optimization/10.1 user agent.

The most anomalous stream of 31st January is not shown in Figure 6.3 due to the sampling of the plot. This TLS flow lasted for 441 seconds and 5 397 920 bytes were downloaded from releases.nixos.org (151.101.2.217). No other TLS record with releases.nixos.org as destination was found in the training data. Median of duration of 606 982 events in training data with destination port 443 was 1,73 seconds. The

mean of duration is 128,11 seconds. Only 15 860 out of 606 982 were longer than this stream.

6.4 On Correlation Between Anomalies and Reputation

Since the very beginning of a work on this thesis, the question, whether the anomalous and reputation scores are related, has been present. a theoretical answer to this question lies in definitions of the two computed scores.

The anomaly score is defined as a metric of how much the traffic in differs from traffic originating from the same IP address in the past. The reputation score represents the trustworthiness of the host. The trustworthiness in the developed system is represented amount of malicious traffic sent by the IP address during the last two weeks.

Given these two interpretations of the term *anomaly score* and *reputation score*, the hypothesis is that these scores should not present a high degree of positive correlation. It can be argued, that the correlation of the two scores might, in fact, be negative. This hypothesis can be explained by the following scenario: If a reputation score is high, it means that the source IP address must sustain a long-lasting malicious or non-trustworthy behaviour. Long-lasting trend of behaviour implies that the anomaly score for such IP address should be low. Low value of the reputation score does not imply any specific value of the anomaly score. Traffic of an IP address can be changing its behaviour, thus increasing the anomaly score. A change in behaviour does not necessarily mean that the behaviour before or after this change was malicious.

If the anomalies were considered as malicious behaviour or threats, instead of a simple change of behaviour, the correlation of both scores should be positive. Both scores would then play a similar way, both scoring a level of bad patterns found in traffic. Each would just use a different method to do so.

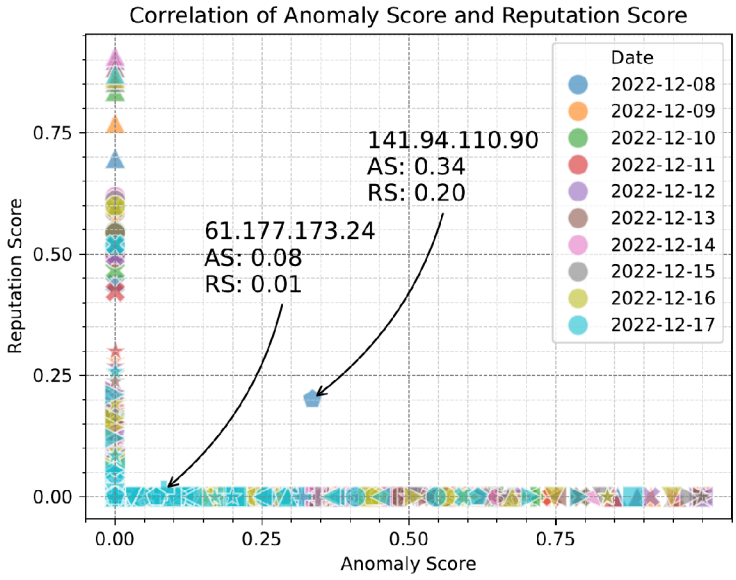


Figure 6.5: Reputation and anomaly scores for all scored IP addresses (markers) during 10 continuous days.

The predicted relation of the two scores is proven by experiments. Logs were rated for seven days with both an anomaly and a reputation score. Scores were then extracted from the logs for each IP address and plotted in Figures 6.5 and 6.6. Figure 6.5 shows all detected scores across ten days. Colours represent the day on which the log events were recorded, and the day for which both scores have been calculated. Symbols in the plot represent different IP addresses which were scored. Figure 6.6 illustrates the same experiment, except that it only shows data points, with values of both scores that were not zero. These figures clearly show that there is no significant correlation between the anomaly and the reputation scores.

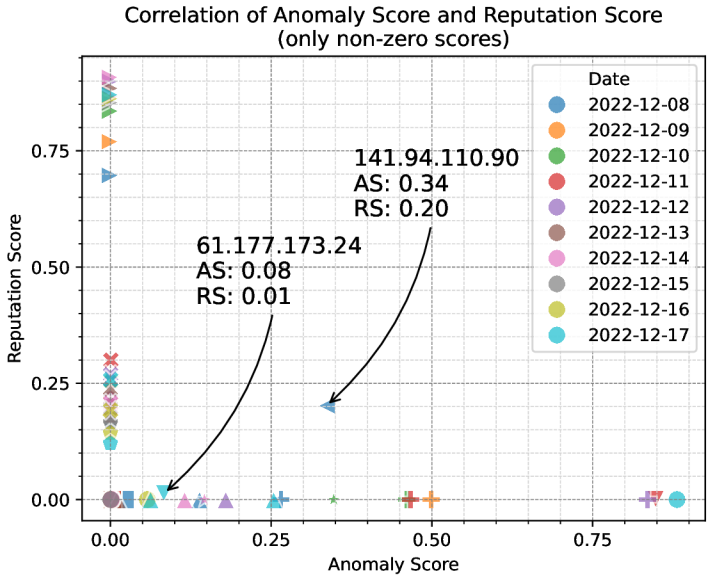


Figure 6.6: Reputation and anomaly scores for all scored IP addresses (markers) during 10 continuous days. Only records, for which neither of the scores was zero, are shown.

The two data points, which show certain amount of correlation, represent addresses 141.94.110.90 and 61.177.173.24. Reputation score of 0.2 of 141.94.110.90 address is caused by a series of *invalid banner* events generated on 2022-11-25, 2022-11-29, 2022-12-02, 2022-12-05, 2022-12-06, 2022-12-07, and 2022-12-08. The address is shown in Figure 6.6 on 2022-12-08, as no traffic originating from that address has been captured in the later days. Anomaly score value of 0.34 means a non-negligible deviation from absolutely normal behaviour of the given IP address, which is however below the threshold for that IP address on 2022-12-08. Threshold value for that day/IP is 0.5. It is the default threshold value, the model did not however find any better threshold.

IP address 141.94.110.90 is not shown in Figure 6.1, as it may have a significant reputation score, however, the total sum of scores within multiple days is lower than of which addresses shown in Figure 6.1.

6.5 Using Reputation Score to Enhance Reduced Logs

When the anomalous log events are found and extracted from the original logs, a person responsible for the log monitoring (usually a system administrator) still needs to observe these anomalous events to decide, whether a source of such events poses a risk to the

installed infrastructure and if an action needs to be taken or not. Even if the amount of time spent by going through logs is greatly reduced, new process is still not fully automated.

Partial or complete automation can be achieved by matching resulting reputation scores of source IP addresses with source IP addresses found in the log records. Now, when an anomalous record is paired with a reputation score, the job of an administrator is simplified. An administrator can now look just into one source of data (reduced logs with anomaly/reputation scores) to determine a follow-up action. In order to achieve trustworthy and reliable matches, it is important to match log events with a correct reputation score based on the timestamp. An example of a resulting log record with anomaly and reputation score is shown below:


```
{'anomaly_score': {'anomaly_score': 0.0013219913112075053,
                   'is_anomaly': True,
                   'threshold': 2.383783149425184e-05},
 'app_proto': 'failed',
 'dest_ip': 'V.V.BB.VV',
 'dest_port': 21027,
 'ether': {'dest_macs': ['ff:ff:ff:ff:ff:ff'],
           'src_macs': ['XX:XX:XX:XX:XX:XX']},
 'event_type': 'flow',
 'flow': {'age': 0,
          'alerted': False,
          'bytes_toclient': 0,
          'bytes_toserver': 485,
          'end': '2023-01-31T23:53:12.120033+0100',
          'pkts_toclient': 0,
          'pkts_toserver': 1,
          'reason': 'timeout',
          'start': '2023-01-31T23:53:12.120033+0100',
          'state': 'new'},
 'flow_id': 2072079663140589,
 'host': 'XXXX',
 'in_iface': 'XXXXX',
 'proto': 'UDP',
 'reputation': {'calc_date': '2024-02-22 18:13:21.074152',
                'daily_scores': {'2023-01-18': 0.0,
                                 '2023-01-19': 0.0,
                                 '2023-01-20': 0.0,
                                 '2023-01-21': 0.0,
                                 '2023-01-22': 0.0,
                                 '2023-01-23': 0.0,
                                 '2023-01-24': 0.0,
                                 '2023-01-25': 0.0,
                                 '2023-01-26': 0.0,
                                 '2023-01-27': 0.0,
                                 '2023-01-28': 0.323160637971313,
                                 '2023-01-29': 0.0,
                                 '2023-01-30': 0.0,
                                 '2023-01-31': 0.0},
                'last_scored_day': '2023-01-31',
                'score': 0.03627313283351473},
 'src_ip': 'V.V.BB.VV',
 'src_port': 53844,
 'timestamp': '2023-01-31T23:53:42.121035+0100'}
```

This log record shows an example of an event rated with both the anomaly and reputation scores. *anomaly_score* and *reputation* fields show the scores added by the system. Because the value of anomaly score is higher than the threshold, this record is considered anomalous. Reputation score for V.V.BB.VV was computed on 2024-02-22. Reputation score of 0.036 was computed for 2023-01-31 Records from 2023-01-18 to 2023-01-31 were used for the calculation of overall reputation score. Daily reputation score was zero for all days except 2023-01-28. 2023-01-28 was the only day when a malicious event was detected by Suricata.

Because all anomalous events have their reputation scores available in the log event and because both the anomaly and reputation scores are numeric, it is possible to develop an automatic process which determines the action to be taken. This automation can utilise static predefined thresholds to make decisions or a machine learning model can be developed to calculate more complex threshold rules.

During a testing period between 8th December 2022 and 17th December 2022, 727 602 Suricata log records were rated using anomaly score. Out of the 727 602 records, 4 542 were classified as anomalous, as seen in Table 6.3. When observed over the ten days of testing, there are 454 anomalous events per day by average. This amount of alerts is still higher than optimal amount for an administrator to check. An administrator would probably ignore these alerts if he had to investigate 454 events per day.

	Number of Events	% of All
All records	727 602	100.00
Non Anomaly	723 060	99.38
Anomaly	4 542	0.62

Table 6.3: Number of anomalous records detected by the developed system. Records from logs between 8th December 2022 and 17th December 2022.

30% of 727 602 records has a source IP address with the reputation score equal to zero. This means that no bad or malicious behaviour has been detected for these addresses. 505 274 records have been linked to source IP addresses, which did not behave well. This is demonstrated in Table 6.4. When the threshold of acceptable reputation score is increased, less records produce alerts. 13% of all 727 602 records is kept, if the reputation score of the sender IP address is above 0.3.

	Number of Events	% of All
All records	727 602	100.00
Reputation = 0	222 328	30.56
Reputation > 0	505 274	69.44
Reputation >= 0.1	503 925	69.26
Reputation >= 0.2	386 337	53.10
Reputation >= 0.3	100 714	13.84

Table 6.4: The number of events with various values of reputation score. Events with a reputation score equal to zero are non malicious. The records are from the logs between 8th December 2022 and 17th December 2022.

	Number of Events	% of All	% of Anomalous
All records	727 602	100.00	-
Anomaly & Reputation = 0	2 616	0.36	57.60
Anomaly & Reputation > 0	1 926	0.26	42.40
Anomaly & Reputation >= 0.1	1 904	0.26	41.92
Anomaly & Reputation >= 0.2	1 444	0.20	31.79
Anomaly & Reputation >= 0.3	743	0.10	16.36

Table 6.5: The number of anomalous records left after application of a reputation score. The records are from logs between 8th December 2022 and 17th December 2022.

Table 6.5 demonstrates the effect of combination of the reputation score and anomaly score during the testing period between 8th December 2022 and 17th December 2022. When combining anomalous events with reputation score calculated for a source IP address, the number of reported alerts is be significantly reduced. As the reputation score does not have any threshold computed by the reputation scoring algorithm, an administrator has to determine the value of threshold. Four threshold values were tested on the dataset. First, a reputation threshold was set to zero. This setting is not viable for the real world use, as it would produce alerts for records belonging to IP addresses with excellent behaviour. Then, reputation scores greater than zero, 0.1, 0.2, and 0.3 were tested. The number of reported events gradually declined to the point when it reached 743 reported events out of 4 542 anomalous events. Reputation score threshold of 0.3, reports 16 % of anomalous points only, which is 74 events per day, when divided over 10 testing days. The process of rating log events using anomaly and reputation scores has been able to reduce the number of events in the original log file to mere 0.1 % of the original number.

Anomalies detected by the implemented system represent data points which are distant to data included in the training dataset. They are thus unexpected, strange, abnormal relatively to the rest of the data. It is not possible to detect these anomalies by simply observing the dataset of Suricata EVE logs. These anomalies do not signify an invalid behaviour. Invalid behaviour is reported in the the Suricata EVE logs in the built in *Anomaly event type*. Suricata documentation[4] describes the *Anomaly event type* as follows: *Events with type „anomaly“ report unexpected conditions such as truncated packets, packets with invalid values, events that render the packet invalid for further processing or unexpected behaviours.* Anomalies reported by Suricata are mainly syntactic, meaning that for example, that the syntactic parsing of an packet ended with an error (Suricata encountered invalid character, etc.).

6.6 Performance Testing

Performance testing of anomaly detection and evaluation of original log events was performed by repeated running of the entire process of data processing, model training, anomaly detection, and matching the resulting scores back to events in the original logs for V.V.BB.VV address. This IP address proved to be the best candidate as it contains more than 1 900 000 records, and it thus provides enough space for dataset shrinking. For each of

[4]Suricata documentation, anomaly event type:https://docs.suricata.io/en/latest/output/eve/eve-json-format.html#event-type-anomaly, accessed [2024-04-15]

the six tests, a portion of training and testing data was deleted from the dataset. The first test consisted of three repeated runs of the scoring program with dataset with 1 943 118 records. Next tests used subsets of the dataset used in the first test. A subset of the original dataset containing only 230 209 records was used in the last test. For each dataset reduction, records from the first five days were removed. The first test thus contained data from 2023-01-01 to 2023-01-31, the second test contained records from 2023-01-06 to 2023-01-31. The last test contained records from 2023-01-25 to 2023-01-31. The time needed for extraction of features from the original logs is not taken into account for the testing, as that process is carried out once per log file, and it includes all IP addresses detected in that file. Limiting the extraction to the V.V.BB.VV only is not possible without serious changes of the code.

Detailed duration for each step of the computing pipeline are shown in Table 6.6. The table shows the average duration of the subtask over the three runs performed for each test. Most time is spent by modifying EVE logs with the computed score, as this process has to load all log files and assign the score to each record with matching flow ID. Calculation of Mahalanobis distance, which takes from 22% (smallest dataset) to 44% (largest dataset) of the execution, is the longest part of the entire calculation. The process of model fitting and following anomaly scoring accounts to 5-12% of the entire duration without the final log size reduction. The reduction in time for the *Anomaly Score Calculation* and *Scoring Log Events* rows for the test no. 6 is explained by data points included in the training and testing datasets of test no. 6. The cluster of 18 608 flow records shown in Figure 6.3 under letter a belongs to the training dataset of test no. 6. This cluster belongs to testing dataset for the other tests. Because this large cluster of records is not in the testing dataset of test 6, the scoring of the testing dataset and matching these scores to the original log files takes less time.

Test Number	1	2	3	4	5	6
No. Records (Train+Test)	1943118	1590700	1238500	923721	425633	230209
Numeric Conversion	3.41	2.81	2.21	1.67	0.84	0.40
Ordinal Encoding	5.68	4.75	3.77	2.88	1.44	0.78
Drop Singular Cols	0.95	0.78	0.62	0.51	0.28	0.13
Normalization	0.16	0.13	0.11	0.08	0.03	0.02
Mahalanobis Distance	28.95	24.00	18.31	13.64	6.35	3.51
Outlier Trimming	1.26	0.96	0.71	0.50	0.19	0.10
Splitting Train/Test	1.04	0.81	0.63	0.47	0.21	0.12
Removing Temp Cols	0.10	0.08	0.07	0.06	0.03	0.02
PCA Model Fitting	2.70	2.18	1.56	1.15	0.04	0.02
Anomaly Score Calc.	3.42	3.41	3.44	3.42	3.45	1.94
Scoring Log Events	14.22	14.41	14.14	14.24	14.25	8.19
Reducing Logs	108.33	111.21	112.73	111.85	114.46	114.78
Total Time, No Log Reduce	65.65	57.28	47.88	40.36	28.02	15.67
Total Time	173.98	168.49	160.61	152.21	142.48	130.45

Table 6.6: Duration of a program execution for data preprocessing, anomaly detection, and enhancing the original logs. All measurements are in seconds.

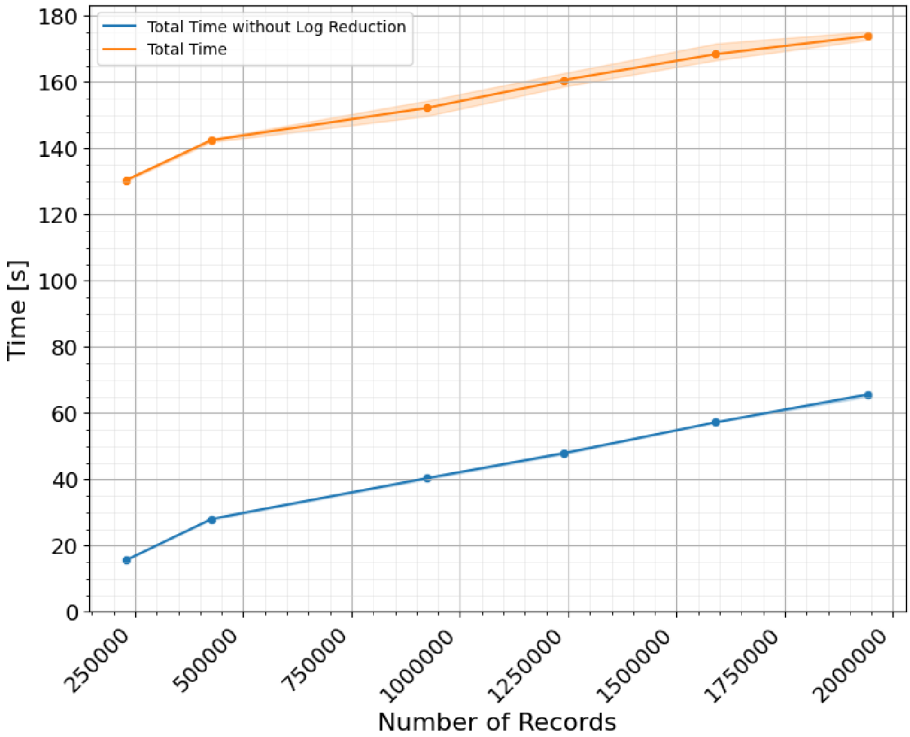


Figure 6.7: Time needed to score events in the original logs in relation to the amount of events in the dataset.

Figure 6.7 shows the relation between the application runtime and the number of events in the dataset. The plot consists of two lines. The blue one includes time needed to update events in the log files with the computed score. As this process always has to traverse the entire set of log files, its runtime is not impacted by the amount of events in the dataset. For this reason, the figure also shows a trend without the final log alteration, which is marked in orange. The plot shows that the scoring process scales linearly with the dataset size. Datasets with more than 2 000 000 data points were not tested due to hardware limitations.

Reputation scoring was not included in this testing, as the impact of it on the entire runtime is negligible.

Performance testing was done on a system equipped with Intel i5-8265U CPU and 16 GB of RAM. All log files, temporary files and program were stored on Intel SSDPEKNW512G8H SSD.

6.7 Environment Preparations

The Suricata EVE logs are highly configurable. As seen in the *EVE JSON output* section of the Suricata documentation[5], every event type defined can be selected to be included or excluded in those logs. Event types have configurable parameters, which further specify data fields included in a log record.

[5]EVE JSON Output documentation: https://docs.suricata.io/en/latest/output/eve/eve-json-output.html, accessed [2024-04-15]

In order for the detection system developed for this work, the following types and fields need to be included in records of the EVE JSON logs:

- Flow event type
  - src_port
  - dest_port
  - app_proto
  - pkts_toserver
  - pkts_toclient
  - bytes_toserver
  - bytes_tocli
  - event_type
  - timestamp
- DNS event type
  - flags
  - type
  - dns_rrname
  - dns_rrtype
- HTTP event type
  - hostname
  - http_method
  - http_user_agent
  - http_url
  - length
  - status
  - url
- MQTT event type
  - host
  - pingresp | publish | pingreq | connect | subscribe | suback | connack | disconnect
- Anomaly event typee
  - app_proto
  - type
  - event
- Alert event type
  - category

- severity
- signature

If any of the needed values are not present in the EVE JSON log given as the output for the system, the value will be filled with default value. Computation of anomaly and reputation score is performed, and the logs are reduced. However, the performance of the detection will be worse, as not all possible information is provided to the models. Examples of compatible record formats are shown in Section 5.1.1.

6.8 Summary

This chapter presented the results of the developed methods, discussed them, raised questions related to the methods or results, and tried to answer them. Validation of computed reputation score against other reputation systems (NERD, Talos) was included in Section 6.1. The developed system was able to detect malicious IP addresses, which were also reported by other tools, thus the developed system corresponds with existing solutions.

In 6.2, the practical implications of computing the reputation score from data gathered across multiple weeks was demonstrated. The relation between the daily sub scores and the overall reputation score was shown. The section explained, why reputation score cannot reach its worst possible values, if a network node behaved badly for only one day. Conditions leading to the rise or fall of the reputation score in time were brought up and demonstrated in figures.

Section 6.3 talked about the nature of detected anomalies and it provided explanations, why some points were selected as anomalies by the model. Examples of training data provided a background as to why the model acted as it did. The anomalies were shown in a figure together with non-anomalous points to show the differences. Selected anomalies were discussed. An analysis of why some data points are considered anomalous by the model was performed and described. The most anomalous flows were those with extremely log duration spanning multiple days. SSH flows with non-standard SSH ports were also detected as anomalies.

The relation between anomaly and reputation score was demonstrated and explained in 6.4. No significant correlation between both scores was found. This lack of correlation is in line with the hypothesis that if an IP address keeps a stable reputation score over time, then its behaviour is stable (not anomalous).

The result of combining anomaly and reputation scores into an automated alert reduction system was described 6.5. The positive real world results were presented. The developed system was able to reduce the initial 727 602 records to 743 records, which represents only 0.1% of the original number. The major goal of the thesis—the log reduction—is thus reached.

The complexity of the scoring method was assessed in Section 6.6. It was shown, that some actions like the computation of Mahalanobis distance, has worse time complexity than other parts of the calculation, and that it is the major contributor to the final runtime of the program. The final matching of computed scores to records in the original logs does not depend on the number of rated flows, as all logs have to be searched through for the needed flow IDs.

Chapter 7

Future Work and Conclusion

This chapter concludes the work with an overview of the achieved results, its contributions, and potential future extensions and improvements upon the developed scoring methods.

7.1 Future Work

Although this thesis studied the assigned problematics thoroughly, there remain possible considerations or extensions for this work.

An extension to this work can be developed. This extension will automatically infer the best possible future action, such as *create a firewall rule blocking an IP address*, based on the calculated scores or log events which were marked as anomalous, events with bad reputation score, or both. The method could use statically defined rules, or it could utilize machine learning to determine the best steps.

The current approach to the anomaly detection could be brought over to a future work, which will try different options for some steps in the computation pipeline. The main idea of the process should remain the same. The PCA method used for anomaly detection could be exchanged for other methods. Methods for unsupervised learning belonging to different categories could be tested and evaluated. Clustering methods like DBSCAN or K-means are some of the methods that can be used. Nonlinear variations of PCA or other nonlinear methods might prove better results, depending on the nature of the input dataset. Machine learning models could be a valid alternative to the mathematical functions used to calculate the reputation score.

Other possible experiments include keeping the methods, but changing the preprocessing — especially the way of transforming categorical to numeric attribute. A suitable conversion technique might be the One Hot Encoder. This encoder transfers one column (a feature) into many columns. The number of newly created columns is equal to by the number of unique values of the original feature. Each of the new columns is populated by values zero and one only. The value of a data point in the column x is one, if the value of the original feature was x. Values in other created columns of that data point remain at zero.

The current way of extracting data from the Suricata EVE logs creates a data points from the individual records. An alternative approach for the data extraction would be to utilise sliding time windows and aggregating multiple records into a single data point. Detection can be then performed on a dataset composed of these windows, instead of observing individual records. This approach would be better suitable to detect collective

anomalies in the data. Collective anomalies are defined in Section 2.1.1. The currently used approach is more suitable for a point anomaly detection.

7.2 Conclusion

In this master's thesis, a method for rating log events using a reputation and anomaly scores was proposed. The implemented method showed how log events from systems like Suricata can be used to train machine learning models for anomaly detection. It also shows how a reputation score can be computed from detected alerts or by combining information from logs with external data.

An anomaly detection system from system logs containing network traffic data was proposed, implemented and evaluated. The proposed model based on the statistical detection method, Principal Component Analysis, proved to be a good classifier for detection. Principles of modern anomaly detection in network traffic and commonly used methods for this case were described and brought forth to the reader. Available data from Suricata IDS instance running at the Faculty of Information Technology, Brno University of Technology, was analysed, and the set of features best describing them was extracted. The PCA model for anomaly detection was trained with selected features on a subset of available data for data originating from one preselected IP address (representing a network node), and tuned to yield the best possible results. The tweaked model was then used for evaluating the rest of IP addresses included in the available dataset.

Records of alerts and protocol anomalies found in the original logs were used together with other extracted data (HTTP and DNS records) to implement a scoring function for reputation. The reputation score starts at a baseline and is worsened every time an entity (i.e., an IP address) presents itself with a malicious or unwanted behaviour. Information that was not accessible in the original logs, such as diverse block lists of hosts or user agents, was utilized to enhance the scoring process. A proposed method of matching a reputation score to the original logs, together with an anomaly score, can serve as the base for an entirely automatic way of deciding, whether a given IP address is malicious, or if it is behaving extraordinarily. In that case, a system administrator can proceed with more in-depth exploration of traffic originating from such address and derive conclusions.

These two scores were matched to correct records in the original dataset, composed of enhanced NetFlow records generated by the Suricata tool. The matching was done by IP address and a date. A reduction of these log files by including only records with scores above a specified threshold was proposed, implemented and demonstrated. From more than 720 000 log events, only 743 anomalous events with bad reputation remained. Log records were reduced to 0.1% of their original number. Experiments and analysis of the results were conducted and observed behaviour of scores was explained. Nature of the anomalies was discussed and it was shown, why the PCA model marked some points as anomalies. Experiments about the stability of reputation score, relation between anomaly and reputation scores, and the performance of the scoring methods were carried out. A quick glance at the possible future work based on this thesis was presented.

Bibliography

[1] Alghamdi, A. A. and Reger, G. Pattern Extraction for Behaviours of Multi-Stage Threats via Unsupervised Learning. In: *2020 International Conference on Cyber Situational Awareness, Data Analytics and Assessment (CyberSA)*. 2020, p. 1–8.

[2] Bahmani, B., Moseley, B., Vattani, A., Kumar, R. and Vassilvitskii, S. Scalable K-Means++. *Proc. VLDB Endow.* VLDB Endowment. mar 2012, vol. 5, no. 7, p. 622–633. ISSN 2150-8097.

[3] Banks, A., Briggs, E., Borgendale, K. and Gupta, R. *MQTT Version 5.0* [online]. 5th ed. Burlington, US: OASIS, March 2019, 2019-03-07 [cit. 2023-09-28]. Available at: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

[4] Barford, P., Kline, J., Plonka, D. and Ron, A. A Signal Analysis of Network Traffic Anomalies. *Proceedings of the 2nd Internet Measurement Workshop (IMW 2002)*. september 2002.

[5] Bartos, V., Zadnik, M., Habib, S. M. and Vasilomanolakis, E. Network entity characterization and attack prediction. *Future Generation Computer Systems*. 2019, vol. 97, p. 674–686. ISSN 0167-739X. Available at: https://www.sciencedirect.com/science/article/pii/S0167739X18307799.

[6] Bartoš, V. *Network Entity Reputation Database (NERD): Database of malicious entities on the Internet and everything we know about them.* [online]. 2020 [cit. 2024-02-20]. Available at: https://nerd.cesnet.cz.

[7] Bartoš, V. *User Guide*. 2023 [cit. 2024-02-21]. Available at: https://github.com/CESNET/NERD/wiki/User-Guide.

[8] Bischoff, P. *Which countries have the worst (and best) cybersecurity? Global rankings* [online]. Comparitech Limited, Jan 2024 [cit. 2024-01-21]. Available at: https://www.comparitech.com/blog/vpn-privacy/cybersecurity-by-country/.

[9] Borenstein, D. N. S. and Kucherawy, M. *An Architecture for Reputation Reporting* [RFC 7070]. RFC Editor, november 2013. DOI: 10.17487/RFC7070. Available at: https://www.rfc-editor.org/info/rfc7070.

[10] Borenstein, D. N. S. and Kucherawy, M. *A Media Type for Reputation Interchange* [RFC 7071]. RFC Editor, november 2013. DOI: 10.17487/RFC7071. Available at: https://www.rfc-editor.org/info/rfc7071.

[11] Borenstein, D. N. S. and Kucherawy, M. *A Reputation Query Protocol* [RFC 7072]. RFC Editor, november 2013. DOI: 10.17487/RFC7072. Available at: https://www.rfc-editor.org/info/rfc7072.

[12] Brin, S. and Page, L. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks*. 1998, vol. 30, p. 107–117. Available at: http://www-db.stanford.edu/~backrub/google.html.

[13] Brownlee, J. *Gentle Introduction to Vector Norms in Machine Learning* [online]. Oct 2021 [cit. 2024-04-02]. Available at: https://machinelearningmastery.com/vector-norms-machine-learning/.

[14] Camacho, J., Pérez Villegas, A., García Teodoro, P. and Maciá Fernández, G. PCA-based multivariate statistical network monitoring for anomaly detection. *Computers & Security*. 2016, vol. 59, p. 118–137. ISSN 0167-4048.

[15] Catillo, M., Pecchia, A. and Villano, U. AutoLog: Anomaly detection by deep autoencoding of system logs. *Expert Systems with Applications*. 2022, vol. 191, p. 116263. ISSN 0957-4174.

[16] CERT Polska. *List of malicious domains* [online]. NASK, 2020 [cit. 2024-02-21]. Available at: https://cert.pl/en/posts/2020/03/malicious_domains/.

[17] CESNET, z. s. p. o.. *Warden* [online]. 2017 [cit. 2024-02-20]. Available at: https://warden.cesnet.cz/en/index.

[18] Cid, D. *The Mysterious Mozlila User Agent bot* [online]. Trunc, a NOC.org company, 2022 [cit. 2024-02-18]. Available at: https://trunc.org/learning/the-mozlila-user-agent-bot.

[19] Dodge, Y. Mahalanobis Distance. In: *The Concise Encyclopedia of Statistics*. New York, NY: Springer New York, 2008, p. 325–326. ISBN 978-0-387-32833-1.

[20] Fernandes, G., Rodrigues, J. J. P. C., Carvalho, L. F., Al Muhtadi, J. F. and Proença, M. L. A comprehensive survey on network anomaly detection. *Telecommunication Systems*. Mar 2019, vol. 70, no. 3, p. 447–489. ISSN 1572-9451.

[21] Fyffe, R. *Open Source Active Reconnaissance (Red Team)* [online]. CrowdStrike, mar 2016 [cit. 2023-12-26]. Available at: https://www.crowdstrike.com/blog/open-source-active-reconnaissance-red-team/.

[22] Gambetta, D. Can We Trust Trust? In: Gambetta, D., ed. *Trust: Making and Breaking Cooperative Relations*. Blackwell, 1988, p. 213–237.

[23] Grandison, T. and Sloman, M. A Survey of Trust in Internet Applications. *IEEE Communications Surveys and Tutorials*. Jan 2000, vol. 3, p. 2–16.

[24] Hamdi, M. and Boudriga, N. Detecting Denial-of-Service attacks using the wavelet transform. *Computer Communications*. 2007, vol. 30, no. 16, p. 3203–3213. ISSN 0140-3664. Special Issue: Advances in Communication Networking.

[25] Henriques, J., Caldeira, F., Cruz, T. and Simões, P. Combining K-Means and XGBoost Models for Anomaly Detection Using Log Datasets. *Electronics*. 2020, vol. 9, no. 7. ISSN 2079-9292. Available at: https://www.mdpi.com/2079-9292/9/7/1164.

[26] Jolliffe, I. T. and Cadima, J. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*. 2016, vol. 374, no. 2065, p. 20150202.

[27] Josang, A. and Ismail, R. The beta reputation system. In: Citeseer. *Proceedings of the 15th bled electronic commerce conference*. 2002, vol. 5, p. 2502–2511.

[28] Jøsang, A., Ismail, R. and Boyd, C. A survey of trust and reputation systems for online service provision. *Decision Support Systems*. 2007, vol. 43, no. 2, p. 618–644. ISSN 0167-9236. Emerging Issues in Collaborative Commerce.

[29] Krog, M. *Bad-referrers.list*. 2023 [cit. 2024-02-18]. Available at: https://github.com/mitchellkrogza/nginx-ultimate-bad-bot-blocker/blob/master/_generator_lists/bad-referrers.list.

[30] Krog, M. *Bad-user-agents.list*. 2023 [cit. 2024-02-18]. Available at: https://github.com/mitchellkrogza/nginx-ultimate-bad-bot-blocker/blob/master/_generator_lists/bad-user-agents.list.

[31] Lakhina, A., Crovella, M. and Diot, C. Diagnosing Network-Wide Traffic Anomalies. In: *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*. New York, NY, USA: Association for Computing Machinery, 2004, p. 219–230. SIGCOMM '04. ISBN 1581138628.

[32] Li, D. and Liu, S. Chapter 4 - Water Quality Evaluation. In: Li, D. and Liu, S., ed. *Water Quality Monitoring and Management*. Academic Press, 2019, p. 113–159. ISBN 978-0-12-811330-1.

[33] Liu, Z., Qin, T., Guan, X., Jiang, H. and Wang, C. An Integrated Method for Anomaly Detection From Massive System Logs. *IEEE Access*. 2018, vol. 6, p. 30602–30611.

[34] Madhulatha, T. An Overview on Clustering Methods. *IOSR Journal of Engineering*. May 2012, vol. 2.

[35] Matoušek, P. *Reputační systémy* [online]. 1st ed. Brno: Brno University of Technology, 2023 [cit. 2023-10-02]. Available at: https://moodle.vut.cz/pluginfile.php/579752/mod_resource/content/1/pds-reputace.pdf.

[36] Mehta, S., Kothuri, P. and Garcia, D. L. Anomaly Detection for Network Connection Logs. *ArXiv*. 2018, abs/1812.01941.

[37] Minka, T. P. *Automatic choice of dimensionality for PCA*. 1st ed. Cambridge, MA, USA: M.I.T. Media Laboratory Perceptual Computing Section, Dec 2000.

[38] Nyuytiymbiy, K. *Parameters and Hyperparameters in Machine Learning and Deep Learning* [online]. Towards Data Science, dec 2020 [cit. 2023-12-23]. Available at: https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac.

[39] OISF. *17.1.2. Eve JSON Format*. OISF, 2024. Available at: https://docs.suricata.io/en/latest/output/eve/eve-json-format.html#event-type-anomaly.

[40] Prabhakaran, S. *Mahalanobis Distance: Understanding the math with examples (python)* [online]. Statistics How To, 2024 [cit. 2023-12-07]. Available at: https://www.machinelearningplus.com/statistics/mahalanobis-distance/.

[41] Proença, M. L., Coppelmans, C., Bottoli, M., Alberti, A. and Mendes, L. S. The Hurst Parameter for Digital Signature of Network Segment. In: Souza, J. N. de, Dini, P. and Lorenz, P., ed. *Telecommunications and Networking - ICT 2004*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, p. 772–781. ISBN 978-3-540-27824-5.

[42] Prosise, J. *PCA-based anomaly detection*. 2021 [cit. 2023-12-04]. Available at: https://github.com/jeffprosise/Machine-Learning/blob/master/Anomaly%20Detection%20(PCA).ipynb.

[43] Resnick, P., Zeckhauser, R., Friedman, E. and Kuwabara, K. Reputation Systems. *Communications of the ACM*. 2000, vol. 43, no. 12, p. 45–48.

[44] Rácz, A., Bajusz, D. and Héberger, K. Multi-Level Comparison of Machine Learning Classifiers and Their Performance Metrics. *Molecules*. 2019, vol. 24, no. 15. ISSN 1420-3049. Available at: https://www.mdpi.com/1420-3049/24/15/2811.

[45] Sadeghi, S. *Whitening transformation* [online]. 1st ed. Windsor, ON, Canada: University of Windsor, 2024 [cit. 2024-04-02]. Available at: https://jlu.myweb.cs.uwindsor.ca/8380/Whitening_transformation.pdf.

[46] scikit-learn developers. *2.7. Novelty and Outlier Detection* [online]. scikit-learn, 2007-2023 [cit. 2023-11-26]. Available at: https://scikit-learn.org/stable/modules/outlier_detection.html.

[47] scikit-learn developers. *Sklearn.decomposition.PCA* [online]. scikit-learn, 2007-2023 [cit. 2023-11-26]. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

[48] Shyu, M.-L., Chen, S., Sarinnapakorn, K. and Chang, L. A Novel Anomaly Detection Scheme Based on Principal Component Classifier. *Proceedings of International Conference on Data Mining*. Jan 2003.

[49] Slee, T. *Some obvious things about internet reputation systems* [online]. Feb 2016 [cit. 2023-09-28]. Available at: http://tomslee.net/2013/09/some-obvious-things-about-internet-reputation-systems.html.

[50] StatisticsHowTo.com. *Mahalanobis Distance: Simple Definition, Examples* [online]. machinelearningplus, 2023 [cit. 2024-04-02]. Available at: https://www.statisticshowto.com/mahalanobis-distance/.

[51] Stewart, Ken. *Mean squared error (MSE)* [online]. Encyclopædia Britannica, Inc., 2024 [cit. 2024-02-04]. Available at: https://www.britannica.com/science/mean-squared-error.

[52] Subba, B., Biswas, S. and Karmakar, S. A Neural Network based system for Intrusion Detection and attack classification. In: *2016 Twenty Second National Conference on Communication (NCC)*. 2016, p. 1–6.

[53] Suricata project contributors. *Eve JSON Format* [online]. 2023 [cit. 2023-09-28]. Available at: https://docs.suricata.io/en/latest/output/eve/eve-json-output.html.

[54] Systems, C. *Reputation Center* [online]. 2024 [cit. 2024-02-20]. Available at: https://www.talosintelligence.com/reputation.

[55] Travers, J. and Milgram, S. An Experimental Study of the Small World Problem. *Sociometry*. [American Sociological Association, Sage Publications, Inc.]. 1969, vol. 32, no. 4, p. 425–443. ISSN 00380431.

[56] Tsekouras, D. The Effect of Rating Scale Design on Extreme Response Tendency in Consumer Product Ratings. *International Journal of Electronic Commerce*. Routledge. 2017, vol. 21, no. 2, p. 270–296.

[57] Yeung, D. S., Jin, S. and Wang, X. Covariance-Matrix Modeling and Detecting Various Flooding Attacks. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*. 2007, vol. 37, no. 2, p. 157–169.

[58] Ødegaard, R. *User Agent Changes* [online]. Vivaldi Technologies, dec 2019 [cit. 2023-12-26]. Available at: https://vivaldi.com/blog/user-agent-changes/.

Appendix A

Contents of the included storage media

```
/
├── lerras_agent/
│   ├── agent/
│   ├── agent_settings.yaml
│   ├── install_agent.sh
│   ├── README.md
│   ├── requirements.txt
│   └── service/
├── lerras_core/
│   ├── core/
│   ├── core_settings.yaml
│   ├── README.md
│   └── requirements.txt
├── lerras_experiments/
├── thesis_latex_source/
├── README.md
├── thesis.pdf
└── nextloud_link.txt
```