Research Article

Network Intrusion Detection with Threat Agent Profiling

Tomáš Bajtoš, Andrej Gajdoš, Lenka Kleinová, Katarína Lučivjanská , and Pavol Sokol

Faculty of Science, Pavol Jozef Šafárik University in Košice, Košice, Slovakia

Correspondence should be addressed to Pavol Sokol; pavol.sokol@upjs.sk

Received 13 November 2017; Accepted 8 February 2018; Published 25 March 2018

Academic Editor: Jesús Díaz-Verdejo

Copyright © 2018 Tomáš Bajtoš et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the increase in usage of computer systems and computer networks, the problem of intrusion detection in network security has become an important issue. In this paper, we discuss approaches that simplify network administrator's work. We applied clustering methods for security incident profiling. We consider *K*-means, PAM, and CLARA clustering algorithms. For this purpose, we used data collected in Warden system from various security tools. We do not aim to differentiate between normal and abnormal network traffic, but we focus on grouping similar threat agents based on attributes of security events. We suggest a case of a fine classification and a case of a coarse classification and discuss advantages of both cases.

1. Introduction

In the information age, network services and users face cyber threats due to rapid development; networks, network services, and network users are facing cyber threats like malware, data breach, phishing, social engineering, and so forth. These threats must be identified before organizations or users lose any data or reputation. Nowadays, attackers use advance methods, tools, and approaches to avoid detection, like IP address spoofing, encrypted payload, human failure, and so forth.

The aim of any administrator of network services is to monitor, collect, and analyse network traffic, users' activities, and system logs. They have become fundamental to guard against cyber threats to ensure cybersecurity. They are part of measures to ensure integrity, availability, or confidentiality of networks, network services, and network users.

Conventional approaches are *cyber defence systems*, which can be defined as security mechanisms that monitor, track, and block malicious network activities or cyberattacks [1]. Examples of these defence systems are firewalls, authentication tools, and detection systems.

Detection tools of cyber defence systems capture *security events* from the logs of information sources. Security events can be defined as "a low level entity (e.g., TCP packet, system call, and syslog entry) from which an analysis is performed by a security tool" [2]. Depending on their origin, there are

host-based security events (e.g., user's computer) or network-based security events (e.g., network devices and NetFlow probes).

One of the most widespread used cyber defence systems is *intrusion detection system* (IDS). IDS can be defined as "a defense system, which detects hostile activities or exploits in a network" [3]. There are three main types of IDS based on the used method of detection [4, 5]:

- (i) Signature-based (misuse-based)
- (ii) Anomaly-based
- (iii) Hybrid

Signature-based (misuse-based) IDS uses signatures of known attacks (a priori knowledge on attacks). They are effective for detecting known types of attacks without generating an overwhelming number of false alarms [3]. The second type of IDS is *anomaly-based IDS*. This type of IDS monitors network's and system's normal behaviour and identifies any differences from it [6]. The last type of IDS is *hybrid IDS*. It combines misuse and anomaly detection. The standard architecture of hybrid IDS consists of "an anomaly detection module, a misuse detection module, and a decision module combining the results of the two detection modules" [3].

Intrusion Detection Working Group defined a *general IDS architecture* based on the consideration of four types of

FIGURE 1: General IDS architecture.

functional modules. These modules are shown in Figure 1 and are as follows [6]:

- (i) *Event modules* are made of sensor elements monitoring the target system and acquiring information events.
- (ii) *Database modules* store information from event modules.
- (iii) *Analysis modules* analyse events and detect potential hostile behaviour, generating an alarm if necessary.
- (iv) *Response modules* execute response to prevent any detected intrusion if it occurs.

Event boxes receive overwhelming size of data from the monitored environments. The aim of analysis boxes is to process the data in a way that simplifies work of network administrators. It can be achieved by automating activities in the response boxes or allowing administrators to focus only on relevant events.

One solution is to profile network traffic and incidents recorded in event modules. *Profiling module* as a part of analysis modules can be defined as module that groups similar network connections and searches for dominant behaviour using various types of algorithms [1]. Profiling is usually used to distinguish between normal and abnormal network traffic [7]. Profiling modules perform various types of algorithms or methods to group similar network connections, events, or activities and search for dominant behaviour. Workflow of the profiling box is shown in Figure 2. It consists of four steps [1]:

- (i) Data collection
- (ii) Data preprocessing
- (iii) Profiling
- (iv) Reporting

FIGURE 2: Workflow of profiling in profiling module.

Researchers outlined two of the largest problems in security profiling [1]:

- (i) The huge amount of data and the difficulty in detecting patterns in the data and in the learned patterns
- (ii) Visualization ability which can strengthen the role of security profiling by security administration

In this paper, we focus on the behaviour of threat agents. *Threat agent* can be defined as “a system entity that performs a threat action or an event that results in a threat action” [8]. The main aim of this paper is analysing the profiling of security events based on data collected by security sensors. This profiling is closely associated with prediction of threat agent behaviour and the attacks themselves. The prediction also helps with protection of organizations, since the administrators are better informed and they can be better prepared for security incidents in their organization. We only focus on the clustering methods. To formalize the scope of our work, we state the following research questions:

- (i) Analysis of security events’ attributes for threat agent profiling
- (ii) Analysis of profiling of threat agents based on clustering of security events’ attributes

This paper is organized into five sections. Section 2 focuses on the review of published research related to clustering methods in cybersecurity and profiling in cybersecurity. Section 3 outlines the methodology of data collection, preprocessing data, and clustering methods. Section 4 presents results of analysis and discusses the important points. The last section contains conclusions and our suggestions for the future research.

2. Related Works

This section presents the related works carried out by various researchers or research groups. As the paper addresses profiling in cybersecurity area and implements clustering methods to profile, we divide related works into 2 categories:

- (i) Clustering methods in cybersecurity
- (ii) Profiling in cybersecurity

Clustering is often used in intrusion detection systems to decide if the traffic is normal or anomalous. One of the most used algorithms is *K*-means. Münz et al. [9] applied *K*-means clustering algorithm to feature datasets extracted from flow records. Training data is divided into clusters of time intervals of normal and anomalous traffic. Li and Wang [10] improved clustering algorithm through studying the traditional means clustering algorithm. The experiments proved that the new algorithm could improve accuracy of data classification and detection efficiency significantly.

Ranjan and Sahoo [11] described a new way of intrusion detection using *K*-medoids clustering algorithm and certain modifications of it. The algorithm specified a new way of selection of initial medoids and proved to be better than *K*-means for anomaly intrusion detection. The proposed approach has many advantages over the existing algorithm, which mainly overcomes the disadvantages of dependency on initial centroids, dependency on the number of clusters, and irrelevant clusters. Eslamnezhad and Varjani [12] proposed a new detection method based on a MinMax *K*-means clustering algorithm which overcomes the shortage of sensitivity to initial centers in *K*-means algorithm and increases the quality of clustering.

To overcome disadvantages of misuse detection and anomaly detection, hybrid methods are used. There are several papers applying hybrid methods, combining *K*-means and some other techniques. Hybrid classifiers can provide improved accuracy but have a complex structure and high computational cost. Varuna and Natesan [13] introduced a new hybrid learning method, which integrates *K*-means clustering and Naive Bayes classification. Muda et al. [14] proposed a hybrid learning approach by combining *K*-means clustering and Naive Bayes classifiers. Their approach was evaluated using the commonly used KDD Cup'99 benchmark dataset. The fundamental solution is to separate instances between the potential attacks and the normal instances during a preliminary stage into different clusters. Subsequently, the clusters are further classified into more specific categories, namely, Probe, R2L, U2R, DoS, and Normal. Elbasiony et al. [15] introduced the data-mining-based network intrusion detection systems. Two data-mining techniques are used in misuse, anomaly, and hybrid detection. First, the random forests algorithm is used as a data mining classification algorithm into a misuse detection. Second, the *K*-means algorithm is used as a data-mining clustering algorithm into a proposed unsupervised anomaly detection method. Third, the random forests algorithm is used with the weighted *K*-means algorithm to build a hybrid framework to overcome the drawbacks of both misuse detection and anomaly detection.

Important research in the clustering methods applications is the outlier problem. Several authors [16–18] tried to answer the question of which outlier is an anomaly. Liao and Vemuri [17] use the Euclidean distance to define the membership of data points to a given cluster. Breunig et al. [18] state that some detection proposals associate a certain degree of being an outlier for each point.

Using clustering methods is important also for profiling in cybersecurity based on behaviour of IP hosts and anomaly detection. Jakalan et al. [19] focused on the behaviour of IP hosts from the prospective of their communication behaviour patterns. They created hosts' behaviour profiles of the observed IP nodes by clustering hosts into groups of similar communication behaviour. DBSCAN clustering algorithm is used and it found 14 most important features important to represent host behaviour communication patterns (e.g., number of peers, duration of flow, and number of sent SYN-ACK packets). Erman et al. [20] evaluated two different clustering algorithms, *K*-means and DBSCAN, for the network traffic classification problem. Their analysis was based on each algorithm's ability to produce clusters that have a high predictive power of a single traffic class and each algorithm's ability to generate a minimal number of clusters that contain most of the connections. They compared these clustering algorithms to the AutoClass algorithm. The results showed that the DBSCAN algorithm produces the best overall accuracy. Marchette [7] focused on clustering of computers into groups that consist of computers, which tend to have similar activity profiles. In the paper, the authors used two clustering methods: *K*-means and method of Cowen and Priebe. Xu et al. [21, 22] focused on clustering of hosts in the same IP prefixes. They used bipartite graphs to represent hosts' communications in network traffic and described a spectral clustering algorithm for automatic discovery of behaviour clusters in network prefixes based on hosts' communications.

3. Methodology

This part of the paper describes the input data and the way of their analysis. We took into account the workflow in profiling module, according to which we also divided this chapter.

3.1. Data Collection. For the purposes of this research, data were collected during 2 weeks (from 2017-03-16 to 2017-03-31) by *Warden system* [26]. *Warden* is a part of CESNET Large Infrastructure project and it enables security teams to efficiently exchange information on detected events (threats) from honeypots, intrusion detection systems, network threat probes, and even external sources, designed as multiclient queue. Scheme of *Warden system* is shown in Figure 3.

Collected data contain approximately 72 million records from various data sources. Table 1 shows significant sources of collected data and amount of data collected by the source.

Warden in version 3 uses a flexible and descriptive event format, based on JSON-Intrusion Detection Extensible Alert (IDEA) format [27]. IDEA is a descriptive data model using key:value format and JSON structure. The IDEA format

FIGURE 3: Scheme of Warden system.

TABLE 1: Sources of data.

Name of sensor	Count	Description
Dionaea	74731	Honeypot
Kippo	19132	Honeypot
Nemea	2847552	Set of detectors
LaBrea	66561368	Honeypot
Fail2Ban	4606	Detector
HostStats	1252748	NetFlow Analyzer
Flowmon ADS	446	Monitoring tool
IntelMQ	1687132	Security feeds collector
Sentinel	1650	Endpoint security
FTAS	1836	NetFlow Analyzer
Other	15484	For example, Warden filter sender

is defined as maximum 2-level tree of key:value pairs. It allows for just one basic level of indirection when represented in relational models (save for arrays) and avoids lack of predictability and discoverability in multiple-level or recursive schemes. The keys “Format,” “ID,” “DetectTime,” and “Category” are mandatory. The rest of the keys are optional [28]. The keys, which are significant for our research, are stated in Table 2.

3.2. Preprocessing. An analysis of data collected from Warden system is difficult without their transformation. For this reason, they had to be preprocessed. Each record from Warden stands for a security event. Since we consider the IP address as a threat agent, in the context of this paper, *threat agent* is a specific system entity with a public IP address or several system entities of the same private network subnet

using that public IP address to communicate with other devices on the Internet (e.g., using NAT) and perform a threat action.

For easier processing, data was stored in *PostgreSQL* database [29]. The reason for selecting this database storage is the fact that PostgreSQL can very effectively work with JSON format. It directly gets individual values without having to additionally parse strings. Data were stored in the table, which contains 2 columns: ID and IDEA data, where the IDEA data column values are in the IDEA format.

From those data, a table with 12 columns was made by transforming data. Each column has its own data type. Therefore, it is easier to perform specific operations, for example, numerical operations which were not possible to do directly from the JSON format. Columns in this table represent properties: ID, source IP address, target IP address, category, category count, protocol, protocol count, port, duration, start timestamp, end timestamp, and ISP. However, this table contains attacks, not threat agents; therefore another transformation was needed. This transformation consists in merging the same source IP addresses, thus creating one entry per one threat agent.

In the final input for clustering, every threat agent is represented by a 41-element vector. This vector consists of 22 elements related to a type of attacks this threat agent performed. For every type, there is a number stating how many times this threat agent performed a certain type of attack. Out of the next values, the first 12 values are related to a protocol used by the threat agent in the same manner as described for the type of the attack. 13th value expresses how many times the threat agent attacked from a port in range of 0–1023. 14th value expresses the number of times they attacked from port in range of 1024–65535 to attack. The rest of attributes are the following: overall duration of the threat agent activity, maximal idleness between two subsequent attacks of the threat agent, minimal idleness

TABLE 2: Significant key in IDEA format.

Name of key	Type	Example
Category	Array of Event	Recon.Scanning
Source:IP4	Array of Net4	10.10.0.1
Source:Port	Array of Integer	6550
Source:Proto	Array of ProtocolName	TCP
Target:IP4	Array of Net4	10.10.10.2
Target:Port	Array of Integer	80
Target:Proto	Array of ProtocolName	HTTP
EventTime	Timestamp	2017-03-16 18:06:44
CeaseTime	Timestamp	2017-03-31 21:51:30

between two subsequent attacks of the threat agent, and number of different networks aimed at by threat agent (this was determined from the ISP of target IP address), and the last element of the vector representing the threat agent is a number of different targets.

For a statistical analysis, we can exploit information in attributes that attain more than only zero values (attribution reduction). In our case, types of categories except Recon.Scanning and Availability.DDoS have zero values. The same is for all protocols except TCP and UDP. Also both groups of ports have exclusively zero values.

After data transformation and attributes reduction, for each threat agent (IP address), we consider four categories of attributes:

- (i) *Type of security event* is based on a value of key “Category” in the IDEA format. In the collected data, we consider only two categories: Recon.Scanning and Availability.DDoS.
- (ii) *Communication-related data* is based on values of keys “Source:Port,” “Source:Proto,” “Target:Port,” and “Target:Proto” in the IDEA format. In the collected data, these data are identical to previous category. For this reason, they are omitted in the analysis.
- (iii) *Temporal-related data* is based on values of keys “EventTime” and “CeaseTime” in the IDEA format.
- (iv) *Spatial-related data* is based on values of key “Target:IP4” in the IDEA format. In the collected data, we consider a number of different targets and a number of Internet service providers.

Vectors representing threat agent consist of the following attributes:

- (i) IP address of threat agent
- (ii) Category Recon.Scanning
- (iii) Category Availability.DDoS
- (iv) Duration
- (v) Max. idleness
- (vi) Min. idleness
- (vii) ISP
- (viii) Unique targets

Regarding *IP address* of threat agent, it corresponds to key “Source:IP4” in IDEA format. From the perspective of privacy issues, we omitted IP address from vector of threat agents.

Recon.Scanning category of security event corresponds to key “Recon.Scann-ing” in IDEA format. *Availability.DDoS* is category of security event that corresponds to key “Availability.DDoS” in IDEA format.

Timeline of all events for threat agent can be seen in Figure 4. On one hand, T_{kl}^E (*EventTime*) is start of security event l associated with threat agent k . On the other hand, T_{kl}^C (*CeaseTime*) is end of security event l associated with threat agent k .

$$\text{Time of event}_{kl} = \begin{cases} T_{kl}^C - T_{kl}^E & \text{iff } T_{kl}^C \neq T_{kl}^E \\ T_{kl}^E & \text{iff solitary.} \end{cases} \quad (1)$$

Duration is sum of all time of events for the threat agent.

$$\text{Duration}_k = \sum_l \text{Time of event}_{kl}. \quad (2)$$

MaxIdleness_k is maximum of all time periods between security events (time of inactivity) for threat agent k .

$$\begin{aligned} \text{MaxIdleness}_k &= \max \{ T_{kn}^E - T_{kn-1}^C, T_{kn-1}^E - T_{kn-2}^C, \dots, T_{k2}^E - T_{k1}^C \}. \end{aligned} \quad (3)$$

MinIdleness_k is minimum of all time periods between security events (time of inactivity) for threat agent k .

$$\begin{aligned} \text{MinIdleness}_k &= \min \{ T_{kn}^E - T_{kn-1}^C, T_{kn-1}^E - T_{kn-2}^C, \dots, T_{k2}^E - T_{k1}^C \}. \end{aligned} \quad (4)$$

ISP_count is a number of unique networks recorded for the threat agent (IP address) according to Internet service providers (ISP). This was collected using IP-API service [30]. This service provides spatial data about an IP address and its ISP. *Unique_targets* is a number of unique targets (hosts with IPv4 address) according to threat agent. Relationship between *ISP_count* and unique targets can be expressed as $\text{ISP_count} \leq \text{Unique_targets}$.

FIGURE 4: Timeline of events for threat agent.

3.3. Clustering Methods. Nowadays, various kinds of clustering algorithms are employed in different fields to separate individual objects of interest into groups. The resulting clustering has to be supported by statistical performance measures. Clustering methods differ in the choice of the objective function as well as the distance matrix used and the approach to construct the dissimilarity matrix. They can be broadly categorized [31] into two groups: hierarchical and partitioning. Inspired by new, more comprehensive and specific datasets, other categories have also emerged. Let us mention several of the most popular among them: density-based clustering methods [32], grid-based clustering methods [33], model-based clustering methods [34], categorical or mixed data clustering methods [35, 36], fuzzy clustering methods [37], and others. Some clustering approaches can be sensitive to outliers so their robust modifications [38] have been developed.

For a partitioning method, it is typical that the general process of partition-based clustering [39] is iterative. The first step defines or chooses a predefined number of representatives of the cluster and the second step updates the representatives after each iteration if the measure for the clustering quality (objective function) has improved. In our research, we decided to partition methods because of many advantages [40] they have.

First, most of the partitioning methods (moving centres, K-means, K-modes, and K-prototypes) have low computational complexity [40]. Therefore, they can be implemented for large volumes of data. Furthermore, the number of iterations needed to minimize the within-cluster sum of squares is generally small, making these methods even more suitable for such applications.

The second advantage [40] is that, unlike hierarchical methods, in which the clusters are not altered once they have been constructed, the reassignment algorithms constantly improve quality of clusters. Thus, the quality of clusters can quickly reach a high level when the form of the (spherical) data is suitable.

Third, there is a benefit of an easy and intuitive interpretation, in particular in our application. Partitioning methods we employ have uniquely defined representatives. And this property is desirable when we want to characterize specific groups of threat agents.

Partitioning methods are not ideal in all aspects and it is good to be aware of some drawbacks at the implementation. First, the final partition depends greatly on the more or less arbitrary initial choice of the centres. Consequently, we do not have a global optimum but simply the best possible partition

based on the starting partition. The solution could be to run the clustering algorithm several times with different initial cluster centers. The run with the best value of clustering quality measure (objective function) is selected as the final clustering solution and guarantees that we are not stuck within a local optimum only.

Another challenge [41] is to specify the optimal number of clusters. The solution could be to run clustering algorithm for a range of k values. Then, choose the best k by comparing the clustering results obtained for the different k values. We employ some popular criteria to help us choose the optimal number of clusters. They are mentioned in the text below.

We choose three widespread partitioning clustering methods [31, 39, 42] for our purpose: K-means, PAM (Partitioning Around Medoids), and CLARA (Clustering LARGE Application). In the following paragraphs, we introduce the main ideas behind these well-known methods.

The K-means algorithm [39, 41, 43], one of the mostly used clustering algorithms, searches for a partition of a given set of numeric objects $\mathbf{X}$ into k (given) clusters, which minimizes the within-groups sum of squared errors. This process is often formulated [44] as the following mathematical program problem P :

$$\begin{aligned} \text{Minimize } P(\mathbf{W}, \mathbf{Q}) &= \sum_{i=1}^k \sum_{j=1}^n w_{j,i} d(X_j, Q_i) \\ \text{Subject to } \sum_{i=1}^k w_{j,i} &= 1, \quad 1 \leq j \leq n \\ w_{j,i} &\in \{0, 1\}, \quad 1 \leq j \leq n, \quad 1 \leq i \leq k, \end{aligned} \quad (5)$$

where $\mathbf{W}$ is an $n \times k$ partition matrix, $\mathbf{Q} = \{Q_1, Q_2, \dots, Q_k\}$ is a set of objects in the same object domain, and $d(\cdot, \cdot)$ is the squared Euclidean distance between two objects.

This optimization problem is solved iteratively [41]. The algorithm starts by randomly selecting k objects from the dataset to serve as the initial centers for the clusters. The selected objects are also known as cluster means or centroids. Next, each of the remaining objects is assigned to its closest centroid, where closeness is based on the Euclidean distance between the object and the cluster mean. After that, the algorithm computes the new mean value of each cluster. When the centers have been recalculated, each observation is checked again to see if it might be closer to a different cluster. All objects are reassigned again using the updated cluster means. These steps repeat until the clusters formed

in the current iteration are the same as those obtained in the previous iteration.

The second algorithm we consider is PAM [39–41, 45]. The goal of this clustering method [40] is to find k representative objects (medoids among the observations of the dataset) of clusters which minimize the sum of the dissimilarities of the observations to their closest representative object. A medoid is a representative of a cluster, chosen as its most central object. The centrality is tested by a systematic permutation of one representative and another object of the population chosen at random to see if the quality of the clustering increases. In other words, if the sum of the distances of all the objects from their representatives decreases, the algorithm stops when no further permutation improves the quality.

The PAM algorithm is known to be more robust to outliers than K -means algorithm. It is due to the principle of the given algorithm. The complexity could be considered as its main disadvantage.

To reduce the computing time and RAM storage problem, one can use the modification of the PAM algorithm, namely, the CLARA algorithm [39–41, 45]. The main idea behind this method [39] is that, instead of taking the whole set of data into consideration, the CLARA algorithm randomly chooses a small portion of the actual data as a representative of the data. Medoids are then chosen from this sample using PAM. If the sample is selected in a fairly random manner, it should closely represent the original dataset. CLARA draws multiple samples of the dataset, applies PAM to each sample, finds the medoids, and then returns its best clustering as the output.

Choosing the best clustering method for given data can be a challenging task for an analyst [41, 46]. Therefore, one has to employ measures to compare simultaneously multiple clustering algorithms. In combination with external facts, they help to choose the best performing clustering method with the optimal number of clusters. We follow this approach in our analysis.

More precisely, we compute internal measures [41, 47, 48] and stability measures [41, 47]. Internal measures use intrinsic information in the data to assess the quality of the clustering. As the goal of clustering is to aggregate similar objects within the same cluster and distinct objects in different clusters, internal measures are mostly based not only on compactness and separation of the groups but also on connectivity (see [41, 47, 48] for more details). To internally validate our choice of the clustering algorithm, we calculate the connectivity, the silhouette coefficient, and the Dunn index in the analysis. Higher values of mentioned measures are desirable with exception of connectivity; a value of this measure should be minimized.

Stability measures, a special version of internal measures, evaluate consistency of a clustering result by comparing it with the clusters obtained in cases if each variable is removed, one at a time. In our analysis, we included the following stability measures: the average proportion of nonoverlap (APN), the average distance (AD), and the average distance between means (ADM) (see [41, 47] for more details). The values of APN and ADM lie in $[0, 1]$, whereby smaller values

TABLE 3: Sensitivity of outliers' identification: different methods.

Method	R package	Percentage (%)
DBSCAN	fpc [23]	80.92
ICS	ICSOutlier [24]	84.39
LOF	DMwR2 [25]	85.55

Notes. Percentage is a concordance rate of particular method presented with respect to the percentile method we use.

represent highly consistent clustering results. The value of AD lies in $[0, \infty)$ and smaller values are also preferred.

These introduced measures for comparing clustering algorithms are cleverly implemented in *clValid* package [47] that was very helpful in our clustering analysis.

We also used popular approaches such as elbow method and silhouette method [45, 49] to help us determine the optimal number of clusters.

Moreover, in the final stage of our analysis, we implement clustering for a dataset without outliers and check the influence of such objects on our clustering approaches. Although there are various sophisticated techniques to cope with outliers [50] (e.g., clustering algorithms themselves can identify outliers in data (K -means [51], trimmed K -means [52], and DBSCAN [41])), we use a simple and intuitive approach based on percentiles. We identify an observation to be an outlier if at least one of the characteristics has the value above 99th percentile. We do not consider a lower cutoff point as there is a natural zero bound for each variable.

For a comparison with the percentile method described above, we investigate other common methods to identify outliers:

- (1) Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [53]
- (2) The Invariant Coordinate Selection (ICS) [54]
- (3) Local Outlier Factor (LOF) [18]

In Table 3, we report the concordance rate for outliers identified by other above-mentioned methods with respect to the percentile method. The rate can be interpreted as the fraction of outliers identical to those classified by the percentile method.

There is a quite good agreement in identifying outliers by introduced methods. Consequently, the clustering procedures deliver very similar results after removing outliers from data. According to this finding, we implement percentile method in following computations because it is easy to use and not very time-consuming in contrast to other methods.

4. Results and Discussion

First, variables in the dataset must be scaled to obtain comparable weights of individual variables in the clustering algorithm. We employed one of the most widespread scaling approaches, scaling by the range. Let X^i be the i -th variable (column) and let X_j^i be its j -th element in our dataset. Let n be the number of objects (rows) and let l be the number of variables (columns) in our dataset. Finally, let us denote

FIGURE 5: Scaled data visualization using first two principal components.

by X_j^i the transformed (scaled) data point. Then, for all $i \in \{1, 2, \dots, l\}$ and $j \in \{1, 2, \dots, n\}$, we proceed with the following scaling:

$$X_j^i = \frac{X_j^i - \min_j X_j^i}{\max_j X_j^i - \min_j X_j^i}. \quad (6)$$

Before applying a clustering method on any dataset, it is important to assess its clustering tendency. In other words, one needs to detect if the dataset contains meaningful clusters (i.e., it is a nonrandom structure) or not. If a nonrandom structure is explored, the next task is to determine a number of clusters.

Going to a specific dataset, the best way is to start with data visualization. In our case, we have multidimensional data and they cannot be displayed exactly in their full range. We need to reduce their dimension, for example, by using principal components. Then we can obtain approximated data visualization. For such visualization, we used *factoextra* package [55].

In Figure 5, we can observe that data are by large explained by the first two components. A two-dimensional projection explained more than 90% of the entire variation in data. In what follows, we aim to better understand the data structure.

For the purpose of assessing clustering tendency of our data, we calculated Hopkins statistic [56], which is very well implemented in *clustertend* package [57]. It is used to assess the clustering tendency of a dataset by measuring the probability that a given dataset is generated by a uniform data distribution. Simply said, it tests the spatial randomness of the data. In our case, the value of Hopkins statistic is equal to 0.0031 and it means [41] that our dataset is highly clusterable.

As the initial results indicate existence of clusters in our data, we proceed with searching for the best method and the optimal number of clusters. We consider the three clustering methods, *K*-means [58], PAM [59], and CLARA [59], discussed in the previous section and employ the internal and stability measures to assess how appropriate their use is.

Figure 6 shows values of internal measures for different clustering methods and different number of clusters. A range for a number of clusters is considered to be from 2 to 7, as 7 is taken as the maximum reasonable number of clusters we aim to have in our classification based on seven variables in the dataset. Figure 7 reports corresponding results for stability measures. Furthermore, we consider the elbow method and we plot the total within sum of squares in Figure 8. Based on the three figures, we can make several observations. First, all internal measures prefer *K*-means with two clusters (searching for the minimal value in connectivity measures and maximal value in two others). Second, the elbow method suggests using two clusters indicated by a strong decline at this value. Third, the stability measures do not provide a uniform answer to the questions of what is the optimal method and what is the optimal number of clusters. However, there is a strong pattern across all of them; that is, the stability measures prefer more clusters. Moreover, PAM seems to be least sensitive to different stability measures. Therefore, in addition to *K*-means with two clusters for a coarse classification, we also implement PAM with 7 clusters for a finer classification.

The internal and stability measures provide guidance on which method (from a set of *K*-means, PAM, and CLARA) and which number of clusters (from one to seven clusters) deliver the best properties. For example, the results from the initial diagnostics indicate that if we construct 7 clusters using *K*-means instead of PAM, the clustering will be unstable and uneven. In other words, the decomposition would be not representative. Moreover, CLARA seems to be less appropriate for both coarse and fine classifications. Therefore, we do not implement it at any further stage of the analysis.

Overall, the initial diagnostics of the clustering methods and the optimal numbers of parameters support our view of different refinement of our classification strategy. Based on this, we decided to focus on three different approaches to profiling module:

- (1) One-stage profiling without analysis of outliers
- (2) One-stage profiling with analysis of outliers
- (3) Two-stage profiling analysis

4.1. One-Stage Profiling without Analysis of Outliers. In the first approach, we use one-stage profiling with 2 clustering algorithms (*K*-means and PAM), which are used independently of each other. The first approach can be seen in Figure 9. Here we do not separate any threat agents as outliers. We discuss the outliers in the second approach in the next section.

First, we construct two clusters based on *K*-means to classify our threat agents in a coarse classification. Second, we implement PAM with seven clusters to provide a finer classification and capture a higher variety of non-automatized threat agents.

Table 4 gives an overview of the structure with two clusters. The first cluster is big and contains almost 89% of all threat agents. A representative of this cluster (last 7 columns) is characterized by attacking several targets of one ISP. At

TABLE 4: Representatives of individual clusters, K -means with 2 clusters.

Cl.	Nr.	Perc.	Scan.	DDoS	Durat.	MaxI	MinI	ISP	Targ.
1	4028	88,96	22	0	18137	8341	950	1	7
2	500	11,04	40	2	37156	540852	22649	7	22

Notes. The second and third columns report the number and percentage of threat agents in a specific cluster, respectively. The last seven columns correspond to the following characteristics: Recon.Scanning, Availability.DDoS, duration, max. idleness, min. idleness, a number of ISP, and a number of unique targets.

FIGURE 6: Internal measures for all three clustering methods.

the same time, their behaviour is characterized by rather short breaks between single security events lasting about 5 hours. Interesting factor is the maximum idle time between the security incidents (about 140 minutes), what suggests that threat agent is not coming back to a particular network after a longer time period.

The second cluster is smaller (about 11%) but seems to group more interesting types of threat agents. Threat agents in this cluster are characterized by bigger number of targeted devices in various ISPs. Availability.DDoS attribution is discussed in the next subsection in more detail. Duration time of security events is prolonged and there is a significant rise of other values as well, which might suggest a longer period of activity for the threat agents. This suggests that we are not able to create an appropriate security rules. For this reason, further analysis is needed (clustering with PAM algorithm).

For a better grasp of the clustering output, we also provide visualization of the two clusters in two dimensions in Figure 10.

Now we proceed with an analysis of 7 clusters. Size of individual clusters and characteristics of the representatives are reported in Table 5. Based on them, we can give an interpretation to the members of each cluster.

The first cluster of threat agents (about 82%) is characterized by attacking one device at one ISP. These are short automated actions, suggested by short values of MaxIdleness and MinIdleness. The average time of security events is 733 seconds (12 minutes). In our opinion, this cluster could represent threat agents, hosts infected with malware, which are controlled by command and control servers.

The second cluster of threat agents shows very short attack duration time. Minimum difference between values MaxIdleness and MinIdleness suggests that it is a short,

TABLE 5: Representatives of individual clusters, PAM with 7 clusters.

Cl.	Nr.	Perc.	Scan.	DDoS	Durat.	MaxI	MinI	ISP	Targ.
1	3707	81,87	2	0	1466	10	0	1	1
2	214	4,73	5	0	519	57666	1345	3	4
3	39	0,86	21	0	5116	97521	96082	8	21
4	101	2,23	10	0	7473	253549	0	2	2
5	327	7,22	24	0	6047	490830	8159	8	24
6	71	1,57	29	0	7708	494034	89426	10	29
7	69	1,52	11	0	1647	845105	0	4	9

Notes. The second and third columns report the number and percentage of threat agents in a specific cluster, respectively. The last seven columns correspond to the following characteristics: Recon.Scanning, Availability.DDoS, duration, max. idleness, min. idleness, a number of ISP, and a number of unique targets.

FIGURE 7: Stability measures for all three clustering methods.

automated attack. Unlike the previous cluster, these are security events at multiple devices in multiple ISPs. In this case, we suggest paying further attention to such security events as they do not play any role in aiding the defence of the network.

The third cluster of threat agents is characterized by security events targeted at multiple devices at multiple ISPs. It is interesting that this threat agent attacked each device only once (same values of Recon.Scanning and Targets) and at the same time has the highest value of MinIdleness. Given other values (duration and MaxIdleness), it can be concluded that this was a manual attack. These threat agents need to be further dealt with (not only by adding a firewall rule).

The fourth and the seventh clusters of threat agents are automated attacks due to value of MinIdleness, which target multiple devices at multiple ISPs. The difference between these groups is the values of Duration and MaxIdleness. Threat agents in the fourth cluster repeated network scan due to the value of Recon.Scanning but with short attack duration time. The high value of MaxIdleness might suggest the existence of a bot and its participation in several campaigns.

The threat agents in the fifth cluster scanned the target device only once (values of Recon.Scanning and Target). Time values (Duration, MaxIdleness, and MinIdleness) suggest that it was a scan during one campaign or it could be

FIGURE 8: Elbow method for all three clustering methods.

FIGURE 9: Scheme of profiling module with one-stage profiling without analysis of outliers.

scanning of IPv4 address space of countries (in our case Czech Republic). We suggest treating these threat agents by adding a firewall rule.

The threat of the sixth cluster is similar in its behaviour to threat agents of the fifth cluster. There is only difference in value MinIdleness. Threat agents in this cluster are characterized by the largest number of targeted networks at the largest number of ISPs. In our opinion, it could be scanning of whole IPv4 address space (e.g., by shadowserver and censys.io). These are periodical automated scans to monitor the available devices on the Internet for discovering new threats and assessing their impact. It is beneficial to share security events of these threat agents with other organizations; figure out if it is a scanning service targeting the whole address space; if not, add a firewall rule.

For a better grasp of the clustering output, we also provide visualization of the seven clusters in two dimensions in Figure 11.

4.2. One-Stage Profiling with Analysis of Outliers. In the second approach, we extend our analysis from previous approach by one more layer. This approach can be seen in Figure 12. We treat very specific threat agents separately and suggest that an expert devotes additional time to analyse such threat agents. We identify those threat agents as *outliers*. In statistics, outliers are specific objects that differ from the core of the dataset in some way. For our purpose, we consider an observation (a threat agent) to be an outlier if at least one of the characteristics has the value above 99th percentile. Altogether, we found 173 outliers.

TABLE 6: Representatives of individual clusters, K -means with 2 clusters without outliers.

Cl.	Nr.	Perc.	Scan.	Durat.	MaxI	MinI	ISP	Targ.
1	3945	90,59	5	3482	6398	618	1	2
2	410	9,41	19	7479	500102	15768	7	16

Notes. The second and third columns report the number and percentage of threat agents in a specific cluster, respectively. The last six columns correspond to the following characteristics: Recon.Scanning, duration, max. idleness, min. idleness, a number of ISP, and a number of unique targets.

FIGURE 10: Decomposition of threat agents into two clusters. Visualization using the first two principal components.

Table 6 gives an overview of the structure with two clusters. When compared to Table 4, it can be seen that expelling the outliers had a bigger impact on the number of individual Recon.Scanning and a number of different targets, whose value went down in both clusters. The number of different ISP did not change. Next change is in the value of Duration, which is significantly lower in clusters in Table 6. Interestingly, the ratio of the value between the two clusters stays the same.

The first cluster contains almost 90,6% of all threat agents. A representative of this cluster (last 6 columns) is characterized by low values of MaxIdleness and MinIdleness. In this cluster of threat agents, security events were recorded in one ISP to two different targets. Because value of Recon.Scanning is higher than the value of unique targets, the threat agents attacked each device multiple times. The average time of these events is 700 seconds (11 minutes).

Like in previous approach, the second cluster is smaller (about 9,5%) but seems to group more interesting types of threat agents. Threat agents in this cluster are characterized by bigger number of targeted devices in various ISP. Duration time of security events is prolonged and there is a significant rise of other values as well, which might suggest a longer period of activity for the threat agents. In this case, too, we must conclude that we are not able to create appropriate

FIGURE 11: Decomposition of threat agents into seven clusters. Visualization using the first two principal components.

security rules. For this reason, further analysis is needed (clustering with PAM algorithm).

For a better grasp of the clustering output, we also provide visualization of the two clusters without outliers in two dimensions in Figure 13.

Further, we proceed with an analysis of 7 clusters. Size of individual clusters and characteristics of the representatives are reported in Table 7. Based on them, we can give an interpretation about the members of each cluster.

Compared to the first approach, the attributes of following clusters did not change: clusters 1, 2, 4, and 5. All clusters, with the exception of clusters 1 and 7, have a lower number of threat agents in them. Small change can be seen in clusters 3, 6, and 7. In cluster 7, the value of MinIdleness is negative, meaning that before one security event generated by these threat agents finished, another was recorded. This might suggest that the threat agent's IP address is public and behind it there are several different hosts participating in these security events.

For a better grasp of the clustering output, we also provide visualization of the seven clusters without outliers in two dimensions in Figure 14.

Overall, we conclude that analysis with outliers not only changed individual clusters but also showed group of threat

TABLE 7: Representatives of individual clusters without outliers, PAM with 7 clusters.

Cl.	Nr.	Perc.	Scan.	Durat.	MaxI	MinI	ISP	Targ.
1	3692	84,78	2	1466	10	0	1	1
2	184	4,23	5	519	57666	1345	3	4
3	17	0,39	18	4524	97466	78312	7	18
4	87	2,00	10	7473	253549	0	2	2
5	254	5,83	24	6047	490830	8159	8	24
6	43	0,99	22	5672	547353	78251	7	22
7	78	1,79	7	574	608392	-10	3	5

Notes. The second and third columns report the number and percentage of threat agents in a specific cluster, respectively. The last six columns correspond to the following characteristics: Recon.Scanning, duration, max. idleness, min. idleness, a number of ISP, and a number of unique targets.

FIGURE 12: Scheme of profiling module with one-stage profiling with analysis of outliers.

FIGURE 13: Decomposition of threat agents into two clusters without outliers. Visualization using the first two principal components.

FIGURE 14: Decomposition of threat agents into seven clusters without outliers. Visualization using the first two principal components.

agents that need to be analysed individually. Such division does not impact rules for individual clusters. With *K*-means algorithm, the percentage of same-clustered threat agents is the same whether clustering is done with or without outliers: 99.68%. With PAM algorithm, the matching score is slightly lower but still delivers a value sufficiently close to 100%.

Because of this, we advise to use profiling according to analysis with outliers.

4.3. Two-Stage Profiling Approaches. We use two-stage profiling with 2 clustering algorithms (*K*-means and PAM).

TABLE 8: Representatives of individual clusters, K -means and PAM with 7 clusters.

Cl.	Nr.	Perc.	Scan.	DDoS	Durat.	MaxI	MinI	ISP	Targ.
1	4028	88.96	22	0	18137	8341	950	1	7
2	41	0.91	5	0	4625	341471	0	2	2
3	175	3.86	28	0	5829	472121	15893	8	27
4	87	1.92	25	0	5953	542251	6467	10	25
5	74	1.63	2	0	1306	601559	0	2	2
6	73	1.61	29	0	7708	494034	89426	10	29
7	50	1.10	13	0	1803	908578	12326	5	9

Notes. The second and third columns report the number and percentage of threat agents in a specific cluster, respectively. The last seven columns correspond to the following characteristics: Recon.Scanning, Availability.DDoS, duration, max. idleness, min. idleness, a number of ISP, and a number of unique targets.

TABLE 9: Representatives of individual clusters without outliers, K -means and PAM with 7 clusters.

Cl.	Nr.	Perc.	Scan.	Durat.	MaxI	MinI	ISP	Targ.
1	3945	90,59	5	3482	6398	618	1	2
2	41	0,94	4	2590	325932	0	2	2
3	104	2,39	22	4944	454451	12671	9	22
4	88	2,02	19	4854	490817	8186	6	18
5	76	1,75	25	5953	542251	6467	10	25
6	43	0,99	22	5672	547353	78251	7	22
7	58	1,33	9	1008	609776	-7	2	6

Notes. The second and third columns report the number and percentage of threat agents in a specific cluster, respectively. The last six columns correspond to the following characteristics: Recon.Scanning, duration, max. idleness, min. idleness, a number of ISP, and a number of unique targets.

FIGURE 15: Scheme of profiling module with two-stage profiling without analysis of outliers.

K -means algorithm is used to split threat agents into two clusters. Then first cluster remains unchanged and the second cluster is divided into 6 clusters using PAM algorithm. Like one-stage approaches, we focus on 2 approaches. The first approach is two-stage profiling without outliers' analysis (Figure 15). In this approach, we do not separate any threat agents as outliers. The second approach is two-stage profiling with outliers' analysis (Figure 16). We treat very specific threat agents separately and suggest that an expert devotes additional time to analyse such threat agents.

Unlike one-stage approaches, we do not analyse threat agents per two different outcomes; analysis of one summary division is sufficient. Table 8 gives an overview of the structure with seven clusters of threat agents in the analysis of two-stage profiling without outliers. The attributes of clusters of threat agents in the analysis of two-stage profiling with outliers are listed in Table 9.

For a better grasp of the clustering output, we also provide visualization of the seven clusters with outliers (Figure 17) and without outliers (Figure 18) in two dimensions.

We compare results of one-stage and two-stage profiling. The outcome of the comparison is that the percentage of the same-clustered threat agents in the one-stage analysis and in the two-stage analysis is 71,64%. The second comparison is a percentual ratio of the same-clustered threat agents (without outliers) with the one-staged profiling and with the two-staged profiling, a much better 75.91%. A higher impact of outliers can be seen in the two-stage profiling. The evidence is the percentual ratio of same-clustered threat agents in the two-staged profiling with and without outliers. The outcome is 90.38%. Compared to the one-stage profiling (99.68% and, resp., 98,6%), it is a relatively low number.

4.4. Attribute Availability.DDoS. DDoS attribute in security events was recorded only in 1019 cases, which is a very small number compared to the number of all the recorded events. At the same time, these values appeared only for three threat agents. These threat agents were matched to the same cluster or were outliers, which can be seen in Table 10. In all approaches with an analysis of outliers, these threat agents belong to the outlier group. This shows that an analysis with outliers should be favoured.

While analysing threat agents with DDoS attribute, elementary properties of K -means and PAM algorithms can be observed. In particular, K -means might choose an imaginary element for the centroid. For this reason, DDoS attribution is listed in Table 3. On the other hand, the PAM algorithm chooses a real element as a medoid. It is now clear that threat

TABLE 10: Clusters and attributes of threat agents with DDoS attribute.

Attributes	WOA (K)	WOA (P)	WA (K)	WA (P)	2SWOA	2SWA
Cluster	2	4	Out	Out	2	Out
Count	500	101	—	—	41	—
Perc.	11,04	2,23	—	—	0,91	—
Scanning	40	10	—	—	5	—
DDoS	2	0	—	—	0	—
Duration	37156	7473	—	—	4625	—
MaxId.	540852	253549	—	—	341471	—
MinId.	22649	0	—	—	0	—
ISP	7	2	—	—	2	—
UTargets	22	2	—	—	2	—

Notes. The first column represents attributes. The other six columns correspond to the following profiling approaches: one-stage profiling without analysis of outliers (*K*-means algorithm), one-stage profiling without analysis of outliers (PAM algorithm), one-stage profiling with analysis of outliers (*K*-means algorithm), one-stage profiling with analysis of outliers (PAM algorithm), two-stage profiling without analysis of outliers (*K*-means and PAM algorithms), and two-stage profiling with analysis of outliers (*K*-means and PAM algorithms). The rows correspond to the following attributes: number of clusters, count of threat agents, percentage of threat agents in cluster to all threat agents, Recon.Scanning, availability, duration, max. liveness, min. idleness, a number of ISP, and a number of unique targets. "Out" means outliers.

FIGURE 16: Scheme of profiling module with two-stage profiling with analysis of outliers.

FIGURE 17: Decomposition of threat agents into seven clusters by two-step clustering. Visualization using the first two principal components.

agents with DDoS attributes are not such elements (see Tables 5, 7, and 8).

5. Conclusion

In this paper, we discussed an application of clustering algorithms for security event profiling. We used data collected during two weeks in Warden system, which include security data from various sensors, tools and honeypots deployed to CESNET, and their partner networks. We applied *K*-means and PAM clustering methods to group threat agents based on attributes of security events. In this paper, we discuss the various approaches (one-staged and two-staged profiling with and without analysis of outliers) of using clustering algorithms (*K*-means and PAM) in profiling modules. One-stage profiling with analysis of outliers comes out as the best approach for profiling module. Future research can point to determining size of private network subnet using that public IP address to perform a threat action according to the parameters shown in this paper. The privacy in preprocessing appears as a very interesting research issue.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

FIGURE 18: Decomposition of threat agents into seven clusters without outliers by two-step clustering. Visualization using the first two principal components.

Acknowledgments

The authors would like to acknowledge CESNET and Warden sharing system as part of SABU project for data and valued input. The research was supported by the Slovak APVV project under Contract no. APVV-14-0598.

References

- [1] S. Dua and X. Du, *Data Mining and Machine Learning in Cybersecurity*, CRC Press, 2011.
- [2] B. Morin, L. Mé, H. Debar, and M. Ducassé, "M2D2: A formal data model for IDS alert correlation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Preface*, vol. 2516, pp. 115–137, 2002.
- [3] O. Depren, M. Topallar, E. Anarim, and M. K. Ciliz, "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks," *Expert Systems with Applications*, vol. 29, no. 4, pp. 713–722, 2005.
- [4] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Computer Networks*, vol. 31, no. 8, pp. 805–822, 1999.
- [5] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1153–1176, 2016.
- [6] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1-2, pp. 18–28, 2009.
- [7] D. J. Marchette, "A statistical method for profiling network traffic," in *Proceedings of the Workshop on Intrusion Detection and Network Monitoring*, pp. 119–128, 1999.
- [8] R. Shirey, "Internet Security Glossary, Version 2," RFC Editor RFC4949, 2007.
- [9] G. Münz, S. Li, and G. Carle, "Traffic anomaly detection using k-means clustering," in *Proceedings of the GI/ITG Workshop MMBnet*, 2007.
- [10] T. Li and J. Wang, "Research on network intrusion detection system based on improved k-means clustering algorithm," in *Proceedings of the International Forum on Computer Science-Technology and Applications (IFCSTA '09)*, pp. 76–79, December 2009.
- [11] R. Ranjan and G. Sahoo, "A new clustering approach for anomaly intrusion detection," *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, 2014.
- [12] M. Eslamnezhad and A. Y. Varjani, "Intrusion detection based on MinMax K-means clustering," in *Proceedings of the 7th International Symposium on Telecommunications (IST '14)*, pp. 804–808, IEEE, Tehran, Iran, September 2014.
- [13] S. Varuna and P. Natesan, "An integration of k-means clustering and naïve bayes classifier for Intrusion Detection," in *Proceedings of the 3rd International Conference on Signal Processing, Communication and Networking, ICSCN 2015*, pp. 1–5, March 2015.
- [14] Z. Muda, W. Yassin, M. Sulaiman, and N. Udzir, "K-means clustering and naïve bayes classification for intrusion detection," *Journal of IT in Asia*, vol. 4, no. 1, pp. 13–25, 2016.
- [15] R. M. Elbasiony, E. A. Sallam, T. E. Eltobely, and M. M. Fahmy, "A hybrid network intrusion detection framework based on random forests and weighted k-means," *Ain Shams Engineering Journal*, vol. 4, no. 4, pp. 753–762, 2013.
- [16] K. Sequeira and M. Zaki, "ADMIT: Anomaly-based data mining for intrusions," in *Proceedings of the KDD - 2002 Proceedings of the Eight ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 386–395, July 2002.
- [17] Y. Liao and V. R. Vemuri, "Use of k-nearest neighbor classifier for intrusion detection," *Computers & Security*, vol. 21, no. 5, pp. 439–448, 2002.
- [18] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," *ACM SIGMOD Record*, vol. 29, no. 2, pp. 93–104, 2000.
- [19] A. Jakalan, J. Gong, and S. Liu, "Profiling IP hosts based on traffic behavior," in *Proceedings of the IEEE International Conference on Communication Software and Networks, ICCSN 2015*, pp. 105–111, June 2015.
- [20] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proceedings of the SIGCOMM Workshop on Mining Network Data (MineNet '06)*, pp. 281–286, ACM, Pisa, Italy, September 2006.
- [21] K. Xu, F. Wang, and L. Gu, "Network-aware behavior clustering of Internet end hosts," in *Proceedings of the IEEE INFOCOM 2011*, pp. 2078–2086, April 2011.
- [22] K. Xu, F. Wang, and L. Gu, "Behavior analysis of internet traffic via bipartite graphs and one-mode projections," *IEEE/ACM Transactions on Networking*, vol. 22, no. 3, pp. 931–942, 2014.
- [23] C. Hennig, *Fpc: Flexible Procedures for Clustering. R package version 2.1-10*, 2015.
- [24] A. Archimbaud, K. Nordhausen, and A. Ruiz-Gazen, *Outlier Detection Using Invariant Coordinate Selection. R package version 0.2-0*, 2016.
- [25] L. Torgo, *Data Mining with R, Learning with Case Studies*, Chapman and Hall/CRC, 2nd edition, 2006.

- [26] P. Kácha, M. Kostenec, and A. Kropáčová, "Warden 3: Security event exchange redesign," in *Proceedings of the 19th International Conference on Computers: Recent Advances in Computer Science*, 2015.
- [27] P. Kácha, "Idea, security event taxonomy mapping," in *Proceedings of the 18th International Conference on Circuits, Systems, Communications and Computers*, 2014.
- [28] P. Kácha, "Idea: designing the data model for security event exchange," in *Proceedings of the 17th International Conference on Computers: Recent Advances in Computer Science*, 2013.
- [29] Postgresql (2017). Postgresql project. Accessed: 10th November 2017.
- [30] IP-API (2017). Ip-api project. Accessed: 10th November 2017.
- [31] A. Nagpal, A. Jatain, and D. Gaur, "Review based on data clustering algorithms," in *Proceedings of the 2013 IEEE Conference on Information and Communication Technologies, ICT 2013*, pp. 298–303, India, April 2013.
- [32] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek, "Density-based clustering," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 231–240, 2011.
- [33] M. Ilango and V. Mohan, *A Survey of Grid Based Clustering Algorithms*, 2010.
- [34] C. Fraley and A. E. Raftery, "Model-based clustering, discriminant analysis, and density estimation," *Journal of the American Statistical Association*, vol. 97, no. 458, pp. 611–631, 2002.
- [35] P. Fränti, G. Brown, M. Loog, F. Escolano, and M. Pelillo, Eds., *A Comparison of Categorical Attribute Data Clustering Methods, Structural, Syntactic, and Statistical Pattern Recognition*, Springer, Berlin, Germany, 2014.
- [36] D. Lam, M. Wei, and D. Wunsch, "Clustering Data of Mixed Categorical and Numerical Type With Unsupervised Feature Learning," *IEEE Access*, vol. 3, pp. 1605–1616, 2015.
- [37] C. Döring, M.-J. Lesot, and R. Kruse, "Data analysis with fuzzy clustering methods," *Computational Statistics & Data Analysis*, vol. 51, no. 1, pp. 192–214, 2006.
- [38] L. A. García-Escudero, A. Gordaliza, C. Matrán, and A. n. Mayo-Isacar, "A review of robust clustering methods," *Advances in Data Analysis and Classification. ADAC*, vol. 4, no. 2-3, pp. 89–109, 2010.
- [39] B. Makhabel, "Learning Data Mining with R," in *Community experience distilled*, Packt Publishing, 2015.
- [40] S. Tufféry, *Data Mining and Statistics for Decision Making*, Wiley Series in Computational Statistics, Wiley, 2011.
- [41] A. Kassambara, *Practical Guide to Cluster Analysis in R: Unsupervised Machine Learning*, Multivariate Analysis, CreateSpace Independent Publishing Platform, 2017.
- [42] A. Saxena, M. Prasad, A. Gupta et al., "A review of clustering techniques and developments," *Neurocomputing*, vol. 267, pp. 664–681, 2017.
- [43] D. Lam and D. C. Wunsch, "Clustering," in *Academic Press Library in Signal Processing: Volume 1 - Signal Processing Theory and Machine Learning*, vol. 1 of *Academic Press Library in Signal Processing*, pp. 1115–1149, Elsevier, 2014.
- [44] Z. Huang, "Extensions to the k-means algorithm for clustering large data sets with categorical values," *Data Mining and Knowledge Discovery*, vol. 2, no. 3, pp. 283–304, 1998.
- [45] L. Kaufman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley & Sons, New York, NY, USA, 1990.
- [46] C. Hennig and M. Meila, "Cluster analysis: an overview," in *Handbook of cluster analysis*, Chapman & Hall/CRC Handbooks of Modern Statistical Methods, pp. 1–19, CRC Press, Boca Raton, FL, USA, 2016.
- [47] G. Brock, V. Pihur, S. Datta, and S. Datta, "CIVValid: An R package for cluster validation," *Journal of Statistical Software*, vol. 25, no. 4, pp. 1–22, 2008.
- [48] Y. Liu, Z. Li, H. Xiong, X. Gao, and J. Wu, "Understanding of internal clustering validation measures," in *Proceedings of the 10th IEEE International Conference on Data Mining, ICDM 2010*, pp. 911–916, December 2010.
- [49] M. Charrad, N. Ghazzali, V. Boiteau, and A. Niknafs, "Nbclust: An R package for determining the relevant number of clusters in a data set," *Journal of Statistical Software*, vol. 61, no. 6, pp. 1–36, 2014.
- [50] C. C. Aggarwal, *Data Mining: The Textbook*, Springer International Publishing, 2015.
- [51] G. Gan and M. K.-P. Ng, "k-means clustering with outlier removal," *Pattern Recognition Letters*, vol. 90, pp. 8–14, 2017.
- [52] D. Lei, Q. Zhu, J. Chen, H. Lin, and P. Yang, "Automatic k-means clustering algorithm for outlier detection," *Lecture Notes in Electrical Engineering*, vol. 154, pp. 363–372, 2012.
- [53] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD '96)*, pp. 226–231, 1996.
- [54] A. Archimbaud, K. Nordhausen, and A. Ruiz-Gazen, "Multivariate outlier detection with Ics," <https://arxiv.org/abs/1612.06118v3>.
- [55] A. Kassambara and F. Mundt, *factoextra: Extract and Visualize the Results of Multivariate Data Analyses*. R package version 1.0.5., 2017.
- [56] A. Banerjee and R. N. Davé, "Validating clusters using the Hopkins statistic," in *Proceedings of the 2004 IEEE International Conference on Fuzzy Systems - Proceedings*, pp. 149–153, July 2004.
- [57] L. YiLan and Z. RuTong, *clustertend: Check the Clustering Tendency*. R package version 1.4. 2015.
- [58] R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2017.
- [59] M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, and K. Hornik, *cluster: Cluster Analysis Basics and Extensions*, 2017.