electronics

MDPI

Article

Exploiting the Outcome of Outlier Detection for Novel Attack Pattern Recognition on Streaming Data

Michael Heigl [1,2,*], Enrico Weigelt [2], Andreas Urmann [2], Dalibor Fiala [1] and Martin Schramm [2]

[1] Department of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia, Technická 8, 301 00 Plzeň, Czech Republic; dalfia@kiv.zcu.cz

[2] Institute ProtectIT, Faculty of Computer Science, Deggendorf Institute of Technology, Dieter-Görlitz-Platz 1, 94469 Deggendorf, Germany; enrico.weigelt@th-deg.de (E.W.); andreas.urmann@th-deg.de (A.U.); martin.schramm@th-deg.de (M.S.)

* Correspondence: heigl@kiv.zcu.cz or michael.heigl@th-deg.de; Tel.: +49-991-3615-537

Abstract: Future-oriented networking infrastructures are characterized by highly dynamic Streaming Data (SD) whose volume, speed and number of dimensions increased significantly over the past couple of years, energized by trends such as Software-Defined Networking or Artificial Intelligence. As an essential core component of network security, Intrusion Detection Systems (IDS) help to uncover malicious activity. In particular, consecutively applied alert correlation methods can aid in mining attack patterns based on the alerts generated by IDS. However, most of the existing methods lack the functionality to deal with SD data affected by the phenomenon called concept drift and are mainly designed to operate on the output from signature-based IDS. Although unsupervised Outlier Detection (OD) methods have the ability to detect yet unknown attacks, most of the alert correlation methods cannot handle the outcome of such anomaly-based IDS. In this paper, we introduce a novel framework called Streaming Outlier Analysis and Attack Pattern Recognition, denoted as SOAAPR, which is able to process the output of various online unsupervised OD methods in a streaming fashion to extract information about novel attack patterns. Three different privacy-preserving, fingerprint-like signatures are computed from the clustered set of correlated alerts by SOAAPR, which characterizes and represents the potential attack scenarios with respect to their communication relations, their manifestation in the data's features and their temporal behavior. Beyond the recognition of known attacks, comparing derived signatures, they can be leveraged to find similarities between yet unknown and novel attack patterns. The evaluation, which is split into two parts, takes advantage of attack scenarios from the widely-used and popular CICIDS2017 and CSE-CIC-IDS2018 datasets. Firstly, the streaming alert correlation capability is evaluated on CICIDS2017 and compared to a state-of-the-art offline algorithm, called Graph-based Alert Correlation (GAC), which has the potential to deal with the outcome of anomaly-based IDS. Secondly, the three types of signatures are computed from attack scenarios in the datasets and compared to each other. The discussion of results, on the one hand, shows that SOAAPR can compete with GAC in terms of alert correlation capability leveraging four different metrics and outperforms it significantly in terms of processing time by an average factor of 70 in 11 attack scenarios. On the other hand, in most cases, all three types of signatures seem to reliably characterize attack scenarios such that similar ones are grouped together, with up to 99.05% similarity between the FTP and SSH Patator attack.

Keywords: intrusion detection; alert analysis; alert correlation; outlier detection; attack scenario; streaming data; network security

Citation: Heigl, M.; Weigelt, E.; Urmann, A.; Fiala, D.; Schramm, M. Exploiting the Outcome of Outlier Detection for Novel Attack Pattern Recognition on Streaming Data. *Electronics* **2021**, *10*, 2160. https://doi.org/10.3390/electronics10172160

Academic Editor: Younho Lee

Received: 31 July 2021
Accepted: 3 September 2021
Published: 4 September 2021

1. Introduction

In recent times, trends and technologies, such as Internet of Things, Software-Defined Everything and Artificial Intelligence, have accelerated the increasing interconnection of networking devices. These circumstances have led to steadily blurred boundaries between diverse application domains and are characterized by high-volume, high-speed and high-dimensional Streaming Data (SD) posing enormous challenges for applied security mechanisms. Intrusion Detection Systems (IDS) have crystallized as an essential core component in a holistic security solution by identifying malicious activity in the case of compromising. Apart from the distinction of the architecture for IDS, *Host-based IDS* or *Network-based IDS*, in general, two types of detection methods exist. *Misuse-based* systems, also denoted as signature-based or knowledge-based, detect attacks based on already known patterns (signatures). Being quite fast and reliable in terms of detecting known attacks, they are not designed to uncover novel ones whose signatures are not available; thus, misuse-based detection is no longer enough [1]. The growth of Machine Learning (ML) has led to a boost in *anomaly-based* detection methods that create a model of trusted activity from a set of collected data samples and identify malicious activity by analyzing behavior deviations. Although this type of method is predestined to detect novel, yet unknown, attack patterns without requiring a priori knowledge, they are accompanied by a high ratio of False Positives (FPs) and False Negatives (FNs), which limits their utilization in real-world scenarios. In addition, major challenges and open issues exist for IDS [2–4], such as producing an unmanageable amount of alarms, which are overwhelming for human experts, especially when they are erroneously classified as attacks. Coping with the increasing data volume challenge and enhancing the detection capability from a more global perspective, the concept of collaboration for IDS emerged, which deploys a multitude of collaboratively communicating IDS with a central alert correlation unit or with each other. Alert correlation is a common practice with aims such as false alert reduction, attack pattern identification, root cause detection of attacks or prediction of future attack steps by processing alerts from the heterogeneously applied IDS. Furthermore, they aid to reduce the sheer unmanageable number of events (alarm flooding) generated, especially with the continuously growing amount of high-dimensional data, which can no longer be handled by human analysts. However, limited work exists for alert correlation that is able to efficiently process alerts in a streaming fashion. In addition, most of the existing solutions assume nearly 100% confidence of alerts mapping an attack, which mainly applies for misuse-based IDS. This means that generated alerts based on a signature-match contain very precise information with high confidence about the attack or single attack step (of a multi-stage attack) due to the incorporation of existing knowledge. The reason for this is that alerts are typically composed of intrinsic attributes, such as timestamp, source and destination IP or port information, and an alert type field, often referred to as attack class or intrusion type, which specifies or encodes the attack. In this work, we denote an attack scenario as a single-stage attack, while attack campaigns are multi-stage attacks. Similar attack scenarios are called an attack category. Attacks detected by misuse-based IDS are denoted as the intrusion type or class.

Unsupervised Outlier Detection (OD), as part of anomaly-based IDS methods, detect abnormal behavior by monitoring significant deviations from what has previously been seen under the viewpoint that outliers, indicators rooted from attacks, are statistically different from normal data with a significantly smaller ratio and do not happen frequently. According to [5], the most dangerous attacks happen only rarely such that OD will seemingly play a crucial role in future network security. However, by only detecting such outliers as deviations from data attributes, e.g., a newly occurring IP-address in a computer network, the attack recognition ability is significantly limited since processed data, in the worst case, are only assigned a simple label of normal or abnormal. Since alerts consist of various attributes, further information must be gained from OD methods to be applied in alert correlation. However, due to the missing knowledge of known attacks, the alert type information cannot be provided by OD methods.

With the ubiquity of SD across multiple domains in different real-world applications, many online unsupervised OD algorithms have been developed in recent years. Those are able to efficiently process the time-varying data streams one-pass at a time while dealing with phenomena, such as concept drifts, which require timely model updates to counteract accuracy loss if data change as time passes. With online OD algorithms, alerts can be generated in a streaming manner and lead to a dynamic, huge, infinite and fast changing alert stream for which conventional offline alert correlation methods are not designed [6]. Thus, similar to the statement in [7], when it comes to some critical streaming applications, whereby a fast but less accurate OD model is preferred, we strongly support the claim by [6] that it is more significant to detect an on-going attack in a timely manner than analyzing it afterwards in an offline fashion. Detecting attacks at an early stage significantly reduces damage since, even when applying advanced detection systems, sophisticated attackers can nest undetected for up to 100 days [8].

Thus, as the main contribution of this article, we propose the so-called **S**treaming **O**utlier **A**nalysis and **A**ttack **P**attern **R**ecognition framework, denoted as **SOAAPR**. It is able to process the output of locally and autonomously or distributively operating online OD methods by equipping them with an alert generation functionality using the Intrusion Detection Extensible Alert (IDEA) (https://idea.cesnet.cz/ (accessed on 7 June 2021) representation enriched with information that aid alert correlation. Aggregated alerts (of the same outlier event) are fused into meta-alerts for the sake of alert reduction, which are subsequently correlated and clustered in a streaming fashion. Clusters that are considered complete are immediately forwarded to ensure a very low response time for security analysts. A recent and innovative approach [9] is incorporated that, when triggered, transforms the reduced alert clusters, potentially representing an attack scenario, into a graph representation to derive motif signatures that capture the attack's communication relation, denoted as sig_{com} in this work. Two more signatures proposed by us, sig_{attr} and sig_{temp}, capture an attack's expression in the data's features and the time sequence pattern of the attack-related alerts. Those fingerprint-like characteristics allow a privacy-preserving sharing of a novel attack pattern, e.g., utilizing the Structured Threat Information eXpression (STIX) (https://oasis-open.github.io/cti-documentation/ (accessed on 7 June 2021), similar to shared signatures of misuse-based systems. A common problem with this type of IDS is that if similar attacks slightly change, the knowledge base may not be able to detect it any more [10]. However, the benefit of our signatures is that they represent the attack behavior instead of specific intrusion types/classes and can thus be used to identify completely novel attacks with similar behavior from the same attack category.

Our experiments with the popular, security-related CICIDS2017 and CSE-CIC-IDS2018 datasets indicate that SOAAPR is able to reliably correlate alerts of a variety of attack scenarios and, in contrast to the offline competitor Graph-based Alert Correlation (GAC) that is also able to deal with the outcome of OD algorithms, takes notably less processing time. Furthermore, we compare the three signatures, sig_{com}, sig_{attr} and sig_{temp}, in terms of their computational requirements and capability to characterize attack scenarios.

This work offers the following contributions and mainly differs from others presented in Section 2 because, to the best of our knowledge:

- SOAAPR is the first framework that exploits the output of online OD algorithms to mine attack pattern information in a streaming fashion.
- Online OD algorithms are equipped with an alert generation functionality that can be used for alert processing, including timestamp, feature scoring causing the outlierness (root cause) and equally normalized outlier score results utilizing a common format—IDEA.
- Two more novel types of fingerprint-like signatures, apart from sig_{com}, are introduced, sig_{attr} and sig_{temp}, which can be used to characterize and compare attack patterns.
- Attack scenario information in the form of a signature-tuple can be shared in a privacy-preserved way generated from a novel attack pattern utilizing the STIX language.

The remainder of this work is organized as follows. Section 2 provides relevant background information regarding unsupervised OD on SD, as well as aspects of alert correlation, and provides related work with the most popular state-of-the-art solutions for alert correlation with respect to (i) outlier detection and (ii) streaming alerts. In Section 3, details on the conceptualization and operation principle of SOAAPR for streaming outlier analysis to identify attack pattern can be found. It contains a detailed description of the major modules for streaming alert correlation and generation as well as a comparison functionality for all three types of signatures. In Section 4, the evaluation methodology is described along with details on the data sources and the evaluation criteria. The discussion of results (Section 5) is split into two major parts. Firstly, the streaming alert correlation from SOAAPR is compared to the competitor GAC, and, secondly, the results, evaluating the signature generation and comparison, are discussed. The conclusions are drawn in Section 6, along with a glance at future work.

2. Related Work

2.1. Aspects on Unsupervised Online Outlier Detection

OD, which is also referred to as anomaly or novelty detection, identifies atypical patterns that significantly deviate from the norm utilizing a given measure. Their prerequisite is that (i) the data is *imbalanced* meaning that outliers represent the minority compared to normal data, (ii) that outliers are *distinct*, i.e., they are statistically different from normal data and (iii) that outliers are *rare*, which means they do not happen regularly. A broad spectrum of techniques for OD exists, which are most commonly statistic, distance, cluster or density-based. Other methods, including their properties, are discussed in [11,12].

As already pointed out, in particular, the missing ground truth values in evolving (theoretically infinite) data that demand real or almost near real-time processing, taking the evolution and speed of data into account, require unsupervised OD methods capable of dealing with SD. We consider OD on SD, as is the context in [13]. Widely accepted and popular solutions, such as Hoeffding Trees [14] or Online Random Forests [15], achieve good accuracy and robustness in data streams [16] but are not designed to operate on unlabeled data. Over the past couple of years, methods have been proposed that satisfy the unsupervised and online requirement, such as [17–19], but just a few, Isolation Forest (iForest) [20], HS-Trees [21], RS-Hash [22] and Loda [23], have been shown to outperform numerous competitors and are therefore regarded as state of the art [24,25]. Even if iForest was originally intended as an offline algorithm, a handful of variants, such as [16,26–29], have been proposed that are adapting it or are taking advantage of its concept to operate on SD.

With respect to our proposed SOAAPR framework, referring to Section 3, we are mainly interested in two functionalities of online OD algorithms. Firstly, for instance, to deal with FP, OD should provide outlier score values instead of simple binary values. Secondly, finding the actual root cause of incidents is still an open challenge for IDS. The importance of the features of the input data can play a major role when it comes to analyzing detected outliers. Thus, OD algorithms are required that are able to score or rank features according to their contribution to a data instance's anomalousness. The former criterion is fulfilled by all of the aforementioned algorithms, although their scoring range differs. For instance, while Isolation Forest's scoring takes values from $[0, 1]$, Loda yields values from $[0, \infty)$. Referring to the second criterion, to the best of our knowledge, only Loda and the adaption of iForest for SD, PCB-iForest$_{\text{IBFS}}$, proposed in [13], provides by design the functionality to measure the statistic significance of each feature to its contribution to a data instance's scoring result in an unsupervised manner. From a supervised perspective, the Random Forests (RF) [30] algorithm, for which an online variant also exists [15], can provide feature importance scoring functionality using the SHapley Additive exPlanations (SHAP) method [31]. This method is founded on the so-called Shapley values, which provide an explanation of a prediction by computing the contribution of each feature to the prediction—a method from the coalitional game theory.

2.2. Aspects on Alert Correlation

Salah et al. [32] and Hubballi et al. [33] provide a comprehensive overview in the field of alert correlation, which is defined as a measure of the relation between multiple alarms such that new meanings can be assigned to them. Thus, not only the verification of the alerts' validity can be performed but complex attack scenarios can also be identified. Alert correlation techniques try to reconstruct the attack scenarios from alerts that may exhibit an attack that involves multiple stages in compromising a network [34]. Further, they aid the time-consuming and error-prone security investigation by significantly reducing alerts by dropping irrelevant ones or grouping them based on logical relations.

A taxonomy of alert correlation techniques is provided in [32], including the number of data sources, correlation methods and types of architectures used in the correlation process. Alert correlation architectures can be categorized into *centralized* (data collection performed locally and reported as alerts to a central server executing correlation analysis), *distributed* (alerts or high-level meta-alerts are exchanged, aggregated and correlated in a completely cooperative and distributed fashion between equally weighted agents; communication is performed using a peer-to-peer protocol) and *hierarchical* architectures (separates correlations into hierarchical layers of local analysis, regional analysis and global analysis). The number of data sources—*single* or *multiple*—clarifies whether the alert correlation method is sourced by either a single data source, e.g., a database or a single security measure, or by a collaborative set, which allows for a more precise and coherent view about the observed system. In general, correlation methods can be subdivided into *similarity-based*, *sequential-based* and *case-based* ones, whereas the authors of [34,35] introduce, apart from similarity-based and case-based (referred to as *knowledge-based*), *statistical-based* methods and *hybrid* approaches.

This work focuses on a streaming alert correlation for the identification of novel attack patterns suitable for application on online OD algorithms. Thus, we further split related work into two parts since—to the best of our knowledge—no work exists that has yet been tailored for this task. Therefore, we first present related work that deals with the alert correlation for OD (but offline) and, second, work that deals with the streaming alert correlation (but designed for misuse-based IDS).

2.3. Alert Correlation for Outlier Detection

Bolzoni et al., in [36], formulated the problem: when an anomaly-based IDS raises an alert, it cannot associate the alert with an intrusion type/class, mostly mandatory for alert correlation. Anomaly-based IDS can only provide little information, such as the IP-addresses and port information, and in addition, a security analyst might add the intrusion type or class label but only in a laborious manual analysis process. Thus, the authors proposed Panacea [36] in order to automatically classify attacks utilizing a supervised Support Vector Machine learning model. It inspects the payload of data instances and searches for unusual novel patterns, e.g., byte sequences of certain intrusion types by leveraging previously learned information. The intrusion type can be assigned by finding the most similar alert payload. However, Panacea is payload-centered, which hampers the application for certain attack categories, such as Portscan or DDoS, not involving malicious payload content. Further, it requires training data which, particularly for attack-payload, is typically not available and faces problems when dealing with payload-encrypted traffic.

With a different focus on filtering false alarms, the work in [37] addresses the issue that reported alarms from anomaly-based IDS also lack rich information. They may only identify the anomalous connection stream but cannot provide intrusion type or class information. Their proposed framework is composed of the feature constructor, the cluster constructor and the so-called simple best fit cluster (SBFS) in order to monitor the generated alerts from an anomaly-based IDS. The feature constructor extracts network traffic flow information and derives certain metrics used to construct clusters of normal alarm patterns in a training phase utilizing the cluster constructor. Incoming alerts from an anomaly-based IDS are then evaluated in the SBFS module incorporating information from their respective

network traffic flow features if deviations from the trained model occurred. However, the main intention of this work is not the identification of a novel attack pattern but rather the reduction of anomaly-based IDS outputs.

An approach that detects multi-stage attacks in an unsupervised way without details on single-stage attacks is proposed in [38]. Since the authors state that conventional multi-stage attack detection is designed for misuse-based IDS, which are leveraged for single-stage attack detection, their proposal is designed to operate on both signature and anomaly-based IDS alerts. The main idea of the approach is that suspicious flows are generated, clustered and labeled in the rule generation phase. Labeling in this phase means that the intrusion type or class labels are assigned to each cluster. According to [39], the assignment of clusters to corresponding attack stages still needs to be investigated.

A recent work [39] proposes Adept, a distributed framework, to detect individual attack stages in order to uncover a coordinated attack in the IoT security domain. Anomaly detection is performed on the network traffic of IoT devices, and potential anomalies are sent to a security manager. It will aggregate and mine alerts using a method called frequent itemset mining (FIM). The resulting alert- and pattern-level information will be supplied to a ML approach to identify the individual attack stages. However, the attribution of incoming alerts to different attack stages is performed using a supervised approach, in particular, leveraging *k*-Nearest-Neighbor, RF and Support Vector Machine. Furthermore, their extraction and identification of attack patterns is founded on a simple anomaly detection method designed for the needs in the IoT domain. Thus, common state-of-the-art off-the-shelf anomaly detection methods are not compatible with Adept.

A promising generic graph-based alert correlation solution, denoted as GAC, is proposed by Haas et al. in [40]. Since it only relies on the alert attributes IP and port of source and destination it can be exploited to correlate alerts generated by anomaly-based IDS. GAC is composed of three building blocks: alert clustering, context supplementation and attack interconnection. For alert clustering, similarities between each of the alerts are computed by attribute-specific comparison functions. Then, a so-called attribute graph is derived with alerts as nodes and their similarity values as weighted edges. In leveraging community clustering, in particular, the Clique Percolation Method, loosely coupled clusters can be extracted from the attribute graph that potentially contain alerts of a single-stage attack scenario. Context supplementation then transforms the resulting clusters into a graph—flow graph—that characterizes the communication patterns between the alerts. From the resulting flow graph, four different attack categories can be identified depending on the communication relation between attacker(s) and victim(s). The last block, attack interconnection, aids in identifying multi-stage attacks by revealing relations between individual attack scenarios by comparing the set of attackers and victims of each attack cluster.

In a subsequent work [9], the authors proposed a more flexible solution than context supplementation by assigning clusters to one of four attack categories: one-to-one (oto), one-to-many (otm), many-to-one (mto) and many-to-many (mtm). The so-called motif-based approach builds upon the alert clustering stage from GAC or any other method that groups alerts into clusters, potentially reflecting attack scenarios. Then, if clusters are obtained, a communication structure graph is derived by the IP-address and port information extracted from the alerts in the clusters. Thereby, the communication relation of who attacks whom and which ports are relevant for an attack are reflected in a directed graph structure. A fingerprint-like characteristic can be extracted from the graph by leveraging a concept called motif signatures. These are different characteristic sub-graphs, e.g., three nodes with 16 possible edge patterns among them, whose occurrence in the graph represents a motif signature. Network motifs were initially proposed by [41] and have been transferred to the security-domain, as discussed in [9]. However, the application of network motifs in [9] is transferred for the characterization of attacks and allows a fine-grained, privacy-preserving and dynamic generation of signatures for a multitude

of known and unknown attack scenarios, as well as the differentiation and comparison among them.

2.4. Streaming Alert Correlation

A real-time correlation of intrusion alerts is proposed by Wang et al. in [42]. It requires alerts that contain the intrusion type and relies on a vulnerability-centric correlation that maps exploit information and its vulnerability relation with alerts. Hence, the focus of the solution is on multi-stage attack detection, and by its so-called Queue Graph (QG) approach, it is able to counteract the limitations of sliding windows prone to be tricked by adversaries. Using an extension of the QG, the attack graph based approach is able to hypothesize missing alerts and predict future ones.

Ma et al., in [6], propose a real-time system that automatically discovers attack strategies from evolving alert streams. They leverage a well-known streaming clustering method, called CluStream [43], that is designed with an online and offline module and replaces the latter with an alert correlation component. The online module is used to generate high-level alerts, hyper-alerts, that maintain statistics from the streaming alerts, summarizing their characteristics over different time periods. Alerts must feature, among other attributes, the intrusion type denoted as SigID, which is obtained by misuse-based IDS. Signature-like characteristics can then be derived by the assumption that a multi-stage attack of the same type typically happens in a certain time span and the sequence of their hyper-alerts is similar.

A framework for incremental frequent structure mining is proposed in [44] that aggregates alerts into structured communication patterns depending on the connectivity-relation of involved hosts: mto, otm and mtm. The frequency of those patterns is mined from the streaming alerts and is considered finished if it is not changed for a user-definable amount of time. From those patterns, a so-called Frequent Structured Pattern Tree, FSP_Tree, is created that encodes the most significant patterns along with their time-sensitive information in a Pattern Tree.

Ren et al. in [45] propose an online alert correlation using two components. In an offline module, a Bayesian correlation approach is utilized to extract causal relations among alert features. Based on those patterns, the relevance of alerts for attack steps can be analyzed, which will be stored in a Correlation and Relevance Table. Those reference tables can be consulted if new alerts stream into the online module of the system to uncover multi-step attacks. Again, the intrusion type/class field of an alert plays a crucial role and additional alert features must be derived in order to form hyper-alerts of the same type. However, as mentioned by Sundaramurthy et al. in [10], the approach by Ren et al. can only learn and detect the type of attacks that previously occurred. Therefore, Sundaramurthy et al. proposed a slightly different approach, which is knowledge-based but works on the knowledge of an attacker's intention and constraints rather than attack specifics. Thus, they use a semantic model that maps potential meanings to alerts without incorporating types of attack scenarios.

A real-time method, denoted RTECA, is proposed in [46] that extracts so-called critical episodes, which are sequences of alerts that could be part of multi-step attack scenarios. Thus, their framework aggregates alerts, including their intrinsic attributes with the intrusion type and attack severity information in order to generate hyper-alerts and merge similar alerts together. The timely sorted alerts are categorized in larger parts, batches, each divided into smaller parts, called episode windows. The framework is composed of an online and offline module. In the former, an online attack tree is generated based on the alerts, and the steps of multi-step attacks are determined. In the offline phase, alert similarities are computed, and an offline attack tree is generated by the alerts of critical episodes in order to learn multi-step attack scenarios.

Utilizing an offline and online module too, Daneshgar et al. in [47] proposed a method that clusters alerts as fuzzy events according to their similarities and historical events, which are obtained from the offline module, in an online manner. A fuzzy frequent pattern

mining module in the offline phase mines for relations based on statistical characteristics between alerts to extract fuzzy patterns. The resulting correlation strength can, in turn, be taken into account for the similarity measure utilized in the fuzzy clustering.

Zhang et al. in [48] proposed a framework named IACF, which stands for Intrusion Action-Based Correlation Framework. Its components, split into an extraction and modeling phase, cover alert normalization, action extraction, session rebuilding and building, as well as updating a correlation graph. Actions in this work refer to a set of alerts potentially indicating a single-stage attack. Subsequently, sessions are a sequence of actions that represent the association relation between actions based on temporal metrics. IACF is able to prune sessions in order to remove redundant actions, fuse them and construct a correlation graph, which can be utilized to predict future attack steps. In a consecutive work [5], the authors leverage the Hierarchical Temporal Memory (HTM) algorithm for online intrusion scenario discovery and prediction. Again, normalizing alerts is the first step before hyper-alert processing is performed by online clustering, session reconstruction and session encoding. Alerts are clustered by computing the similarity of the types field in an online manner. Similarly to IACF, sessions are reconstructed and encoded in order to feed the HTM online learning as part of the intrusion scenario discovery module. HTM learns the patterns, predicts potential next steps and scores anomalies. Outcomes of the HTM are used to update a correlation matrix representing the correlation strengths between actions from which potential future attack paths can be extracted.

2.5. Delimitation from SOAAPR

Inspired by the above literature, this article transfers and improves methods for aggregation, (streaming) clustering and attack categorization in the field of attack pattern mining by only utilizing the outcome of OD algorithms. To better guide the reader, a delimitation from SOAAPR to related work is given in this section.

Regarding the fusion of alerts of multiple IDS sensors in order to derive high-level alerts with strong confidence about the mitigation of FP and FN, our alert preparation module in SOAAPR works similarly to the aggregation component in RTECA [46]. Both approaches fuse similar alerts based on the concept of attribute similarity. However, RTECA strongly relies on alert type and attack severity information, which is not present when operating on anomaly-based IDS. Thus, we take advantage of additional OD functionality incorporating the outlier score and feature contribution of a data instance leading to an alert generation. Furthermore, as used by [6], similar alerts can be fused if they have only minor temporal differences. SOAAPR differentiates between two cases in which either alerts are obtained from multiple algorithms running in parallel on one system or from multiple distributively operating ones.

In terms of clustering for anomaly-based IDS, GAC's clustering solution [40] is most related to SOAAPR. However, our approach differs in various ways. Since working on a finite set of alerts, it might be difficult for a security operator to choose an appropriate point in time when to start and end recording alerts, which will be fed to GAC. If the recording time is too short, an attack might not yet be finished, and further attack-related alerts are not included. If the recording time is too large, the GAC approach consumes considerable resources. This has been discussed in their work and was mitigated by chunk-based processing in which the alert data set is divided into smaller chunks that can be efficiently computed by GAC. Doing this might split the alerts of an attack scenario into two chunks. Furthermore, if no chunks are considered and no notion of time through alert timestamps is incorporated, it is very likely that alerts with similar attributes that have no temporal dependency and are from different attack scenarios might be clustered together. SOAAPR only mines for temporally "unique" attack stages by taking the timestamp information into account. Furthermore, it automatically checks whether a cluster is completed such that no old alerts get correlated into a recent cluster. Considering parametrization, GAC suffers from choosing the parameter k, searching for fully connected sub-graphs of size k within the community clustering and a suitable threshold τ providing a minimum similarity value

between alerts. The former allows the assignment of alerts to multiple clusters such that resulting clusters may potentially contain alerts of unrelated attacks. For SOAAPR, we assume that each alert can only be assigned to one cluster, and we handle the confidence of alerts in a designated alert preparation stage. Furthermore, instead of having a minimum similarity threshold on the alert level, SOAAPR assigns alerts into clusters with the highest overall similarity. Having a minimum similarity on a cluster level makes it more robust to fluctuations in the alert attributes.

With respect to streaming alert clustering, competitors mainly focused on multi-stage attack detection by correlating alerts enriched with the intrusion type and assuming that raised alerts correspond to a single-step attack. Therefore, streaming solutions are mainly designed for the application of misuse-based IDS. The incremental frequent structure mining approach in [44] is similar to our SOAAPR by creating frequency patterns of alerts as potential attack scenarios, i.e., single step attacks, and only incorporating IP-address information. However, our approach differs in various ways. Largely comparable to those patterns, the clusters in SOAAPR are obtained by more flexible comparison functions, allowing for the consideration of various alert attributes. Problems with the temporal handling of patterns, stated in the outlook of [44], are slowly developing patterns with large delays in between the steps and the fixed *Keep_Active* parameter used to decide when a pattern is considered "stable", i.e., finished. By assigning individual and cluster-characteristic time values, SOAAPR is more flexible when determining a cluster as "saturated", i.e., finished. Additionally, requiring a mining interval time in which every x minutes the pattern tree structure is updated makes it possible for an adversary to perform an attack unnoticed within this time span.

Attempts for discovering novel single-stage scenarios as part of multi-stage attacks are given by RTECA in [46] while mining for critical episodes. However, although stating real-time operation, processing is only started when a window is completed. In contrast, for SOAAPR, each alert is processed immediately, and no window needs to be filled. Aggregated alerts in RTECA are merged by utilizing the intrusion type and attack severity attributes. Our approach dispenses this information and leverages attributes that can be provided by OD algorithms. Although similarity functions in RTECA used to update the correlation matrix are similar to ours, the online clustering process in SOAAPR is more efficient compared to the online attack tree generation when considering the processing of each newly arriving alert.

The online fuzzy clustering module used in [47] operates most similarly to the clustering in SOAAPR. It exploits a common approach that similar alerts belong to an existing pattern (in our approach a cluster) whose similarity degree is high or triggers the generation of a new pattern in the case that existing ones are not sufficiently similar (minimum threshold of similarity). The fuzzy clustering approach also considers the notion of time, which strengthens our approach introducing a "time to live" for each cluster. However, the threshold used for the lifetime of patterns is one timing constraint, which might be exploited by a strong adversary performing its attack until the lifetime is not exceeded. SOAAPR provides two timing constraints that prevents such attacks and also takes care of ensuring new alerts are not added to an existing cluster if the cluster's temporal existence is outdated. However, contrary to SOAAPR, this time difference is part of the similarity functions, which makes its threshold less intuitive for the operator. The parameter tuning is discussed by the authors in their conclusion, and some parameter improvement is part of further work. A membership degree in the fuzzy patterns can be compared to the outlier score values of each alert representing an alert's confidence as part of a cluster. Although not relying on the intrusion type attribute, and thus not adaptable for OD, it is designed and evaluated using alerts from misuse-based IDS.

With their recent work, Zhang et al., in [5,48], proposed real-time intrusion scenario detection methods. With IACF, alerts streaming in real-time are grouped into actions based on the similarity of the intrusion type and destination port. The former strongly requires existing knowledge, such as using misuse-based IDS, which is a major difference

to SOAAPR and our clustering, where the comparison functions is less strict but does not allow duplicate actions because of duplicated alerts or FP. Timely formed sessions from a sequence of actions are then split by a similar method used in SOAAPR to determine saturated clusters. However, we compute the discreteness of time intervals between alerts instead of actions on a session level in IACF. Furthermore, for the sake of reducing the impact of FP, the pruning algorithm might also filter out single-stage attacks. We deem this step as critical since we support and transfer the statement in [7] for OD-based alert correlation that, especially for critical streaming applications, it is more important not to miss critical TP anomalies forming a single-stage attack while accepting a certain rate of FP.

In terms of attack characterization, we see the huge potential of generic signatures derived from attack characteristics, as proposed by the motif-based approach in [9]. Thus, we evaluate the applicability of those communication-related fingerprints and extend them by considering feature attributes and a temporal behavior also potentially characterizing attacks since we deem the communication relation of attacks as not the only and best option to characterize attacks.

Contrary to the mined patterns in [44], SOAAPR provides a more unpredictable, cluster-specific generation of signatures. The mentioned high number of FP patterns is not surprising when dealing with alerts that do not state high confidence and map to a single attack step. SOAAPR will also likely be suffering from FP clusters due to the impossible prevention of FP and FN alerts in real-world applications and its time-specific online clustering approach. However, SOAAPR provides signatures from clusters that can be compared among each other or with ones previously stored as knowledge base, subsequently allowing for a deeper analysis by a human operator.

The derived candidate attack sequence patterns from the approach in [6] are composed of hyper-alert sequences within a certain time span, which characterize a multi-stage attack. Due to the different lengths of those patterns, a comparison with others is impeded and hyper-alerts still contain privacy-relevant information, such as IP-addresses. The intrusion scenario construction phase in IACF [48] derives correlation graphs extracted from pruned sessions. However, as stated by the authors in the conclusion, the generated graphs are not very intuitive for human analysts. This is because sessions are decomposed into binary correlations of sessions. SOAAPR generates comparable, fixed-sized signatures in which similarity scores can be computed. Furthermore, those signatures are free from privacy-relevant information and can be shared with others. Signatures in our approach represent single-stage attack scenarios and not multi-stage attacks. However, by chaining our signatures, multi-stage attack comparison is also possible. Furthermore, depending on the attribute intrusion type, the authors' more recent work [5] performs online clustering in a similar manner as SOAAPR but clusters actions into clusters with high similarity of the type field. This makes the framework highly dependent on existing knowledge, for instance, obtained from alerts generated by misuse-based IDS.

We want to note that much of the alert correlation work, e.g., [49], which relies on misuse-based IDS, assumes that raised alerts can be treated as an attack, i.e., single-stage attack or attack step. Based on those assumptions, a lot of work exists that identifies multi-staged attacks by analyzing those alerts. However, SOAAPR significantly differs from those by assuming that raised alerts from anomaly-based IDS, specifically by OD algorithms in this work, only represent indicators of potential known or unknown attack scenarios, i.e., single-stage attacks. By equipping OD algorithms to produce enriched alerts, SOAAPR mines alerts to identify those attack scenarios. Additionally, OD algorithms have their limitations when identifying a broad spectrum of attack scenarios based on the assumption that outliers are rare, distinct and do not happen frequently compared to normal data. Thus, it is difficult to produce alerts for the whole duration of long-term attacks with a massive amount of data, such as DoS-like or Brute Force, since online OD algorithms might adapt to them by supposing a concept drift. Furthermore, for attack detection using OD, it is assumed that detected outliers are indicators for maliciously triggered events although potentially being only an anomalous event rooted in a non-malicious fault, for instance.

Intentional masking and swamping that might blend OD, as discussed in [50], are also not considered. Rather, with this work, we would like to point out that identifying the attack pattern from OD can work to a certain extent, but in real-world applications, we strongly suggest to leverage a hybrid system of anomaly- and misuse-based IDS. Although the intention of SOAAPR is highly ambitious, the approach is confirmed since real attacks are more likely small probability events, and the most dangerous attacks only happen rarely [5,51]. This so-called "rare data problem" is the case for which OD algorithms are especially predestined.

3. Streaming Outlier Analysis and Attack Pattern Recognition

3.1. Operation Principle

The workflow and operation principle for **S**treaming **O**utlier **A**nalysis and **A**ttack **P**attern **R**ecognition, denoted as **SOAAPR**, is shown in Figure 1. One may utilize SOAAPR in conjunction with online OD algorithms in two different interaction modes depending on the network infrastructure and the available resources. On the one hand, online OD can be performed self-sufficiently on a single system applying multiple OD algorithms in parallel, denoted as *single system—multiple algorithms*. The amount of algorithms in parallel depends on the available resources of the single system. All algorithms perform OD on the same data instances (data points) $\boldsymbol{x}_t$ with dimension d of the data stream $\{\boldsymbol{X}_t \in \mathbb{R}^{n_t \times d}, t = 1, 2, \ldots\}$. The continuous transmission of data records arrives sequentially at each time step t in which the count of features is denoted as d (dimension) and $\boldsymbol{x}_t$ the n_t-th d-dimensional most recent incoming data instance at time t. The feature set of $\boldsymbol{X}_t$ is denoted as $\mathcal{F} = \{f_1, f_2, \ldots, f_d\}$. *Alert Generation* and *Alert Preparation* can be performed locally on the single system after each classifier yields its result, either a TP or a FP. For a greater extent of visibility across larger network infrastructures, collaboratively operating IDS sensors have found their way into alert detection or alert correlation. Therefore, on the other hand, SOAAPR is able to deal with multiple self-sufficiently working online OD algorithms, denoted as *multiple systems—single algorithms*. Each distributively or decentrally applied algorithm in the network infrastructure operates on data instances $\boldsymbol{x}_t$, $\boldsymbol{y}_t$ or $\boldsymbol{z}_t$ from different data streams $\boldsymbol{X}_t$, $\boldsymbol{Y}_t$ and $\boldsymbol{Z}_t$. *Alert Preparation* in this interaction mode is performed by SOAAPR. A combination of both interaction modes, denoted as *multiple systems—multiple algorithms*, is also possible. Thus, *Alert Preparation* must be performed on both the multiple single systems utilizing multiple online OD algorithms and within SOAAPR processing the streaming alerts.

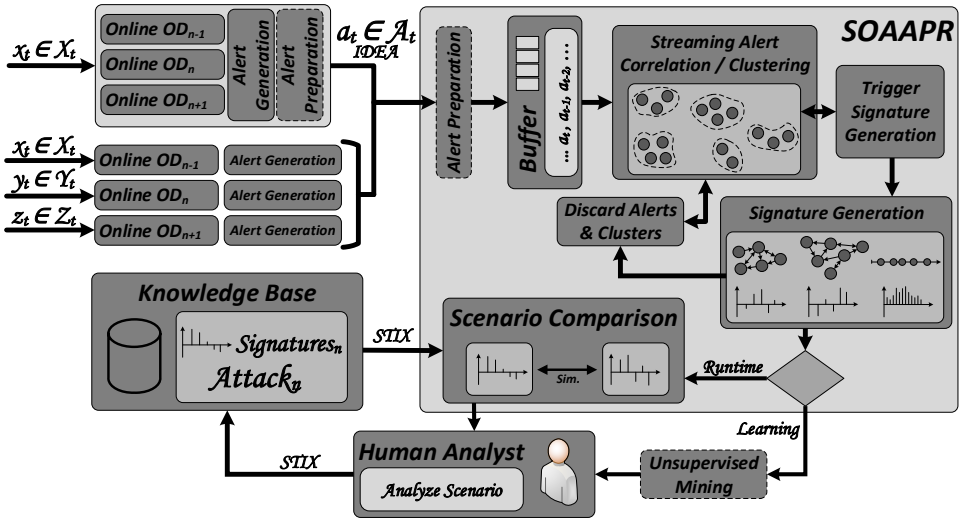


Figure 1. Flowchart and Operation Principle of Streaming Outlier Analysis for Attack Pattern Recognition.

Alert instances $\boldsymbol{a}_t$ from the stream $\{\boldsymbol{A}_t \in \mathbb{R}^{m_t \times l}, t = 1, 2, \ldots\}$, generated by either a TP or a FP, are streaming into SOAAPR at each time step t, in which the count of attributes of

$\boldsymbol{a}_t$ is denoted as l and $\boldsymbol{a}_t = \{a_1, a_2, \ldots, a_l\}$ the m_t-th l-dimensional most recent incoming alert at time t. Within SOAAPR, a *Buffer* is used to temporarily store alerts for the *Streaming Alert Correlation / Clustering*. No longer required alerts, e.g., from unusable clusters, will be flushed by the *Discard Alerts and Clusters* module. The *Trigger Signature Generation* component monitors the evolving clusters $C_i^{(t)}$ from the set $\boldsymbol{C}^{(t)}$ and triggers the *Signature Generation* from suitable clusters, ideally representing an attack or step of an attack, denoted as the attack scenario $S_i^{(t)}$ from the set $\boldsymbol{S}^{(t)}$, in order to create three types of signatures denoted as $sig_{<type>}^{(t)}$.

Further processing of the generated signatures depends on two operation modes—*Runtime* and *Learning*—based on whether an existing knowledge base is available or not. In the *Learning* phase, generated signatures can—if desired—be clustered according to their similarity and are presented to a human analyst. Instead of a massive amount of alerts generated without applying SOAAPR, the expert must only analyze a reduced amount of alerts already preprocessed in a set clustered as $C_i^{(t)}$ and ideally corresponding to the attack scenario $S_i^{(t)}$. In the case of a true attack scenario, the expert condenses information about the attack scenario using STIX and attaches the respective signature $sig_{<type>}^{(t)}$ to it. Since the resulting knowledge of the attack scenario only consists of fingerprint-like attack-characteristic information, free of privacy-relevant or confidential data, such as in-house IP-addresses. Thus, it can be shared in a cross-company manner. Having a knowledge base established, in the *Runtime* phase, generated signatures can be compared within the *Scenario Comparison* module by calculating similarity values. This way, one can take advantage of the strengths of misuse-based signatures with their fast, efficient and reliable attack identification capability but surpasses it via its ability to identify completely new, yet unknown, attack scenarios by being similar to known scenarios (from the same attack category) with a similar pattern. In the following, the core components of SOAAPR are discussed in more detail.

3.2. Alert Generation and Preparation

An OD algorithm $OD(\cdot) : \boldsymbol{x_t} \rightarrow f(\boldsymbol{x_t})$ assigns either a class label as given by $f(\boldsymbol{x_t}) \in \{normal, abnormal\}$ or a score value $f(\boldsymbol{x_t}) \in \mathbb{R}$, describing the strengths of anomalousness, for each data object in $\boldsymbol{X_t}$. Score values carry more information and aid the *Alert Preparation*, for instance, by dealing with FN. Thus, *Alert Generation* creates alarms utilizing the IDEA format for instance by enriching the intrinsic alert properties such as timestamp, IP-address and port source and destination information with (i) a normalized value of the outlier score and (ii) the respective top-γ-features mainly causing the outlierness as the subset $\mathcal{F}_S = \{f_{i1}, f_{i2}, ..., f_{i\gamma}\} \subseteq \mathcal{F}$.

We deem (i) crucial when combining multiple OD algorithms with different scoring functions since, for instance, algorithms such as Kitsune [18] or Loda [23] score instances higher the more abnormal they are within their model, $f(\boldsymbol{x_t}) \in [0, \infty)$. This impedes setting a unified threshold value across all applied OD algorithms, in particular, when they operate on different hyperparameter sets. As an example, Loda utilizing two alternating sets of histograms with different window sizes will yield different averaged score values due to the different state of knowledge about normal data. With increasing window sizes, the models become more accurate while incorporating larger amounts of normal data instances, which, in turn, are then scored less compared to models obtained from smaller window sizes. We suggest normalizing the outlier scores using an improved version of the *Gaussian Scaling* proposed by [52], in which the mean μ and the standard deviation σ are used to encounter the aforementioned problems. Since μ and σ work well for normally distributed values, i.e., assuming a normal distribution of the outlier scores, we replace them by the median *med* and the median absolute deviation *mad* because they are a better option for distributions with skewness.

Figure 2 depicts an exemplary distribution showing non-negative (positive) skewness of the Loda online algorithm utilized on the fourth CSV file of the security-related UNSW-

NB15 [53] dataset. It clearly points out the difference between the mean and median caused by the unequal ratio of outliers. Thus, the normalization formula leads to Equation (1), where $erf()$ is the monotone and ranking stable Gaussian Error Function, and med_t and mad_t are moving or rolling variants of the median and median absolute deviation to be applied in a streaming fashion.

$$\tilde{f}(\boldsymbol{x_t}) = max\{0, erf(\frac{f(\boldsymbol{x_t}) - med_t}{mad_t\sqrt{2}})\} \tag{1}$$

Applying this formula will translate the arbitrary outlier score values in the range $[0, 1]$ into interpretable values describing the probability of a data instance being an outlier. Using domain expertise, a reasonable threshold can be determined over runtime, yielding a decent classification performance that assigns a binary value from the set $\{normal, abnormal\}$ for each $\tilde{f}$.

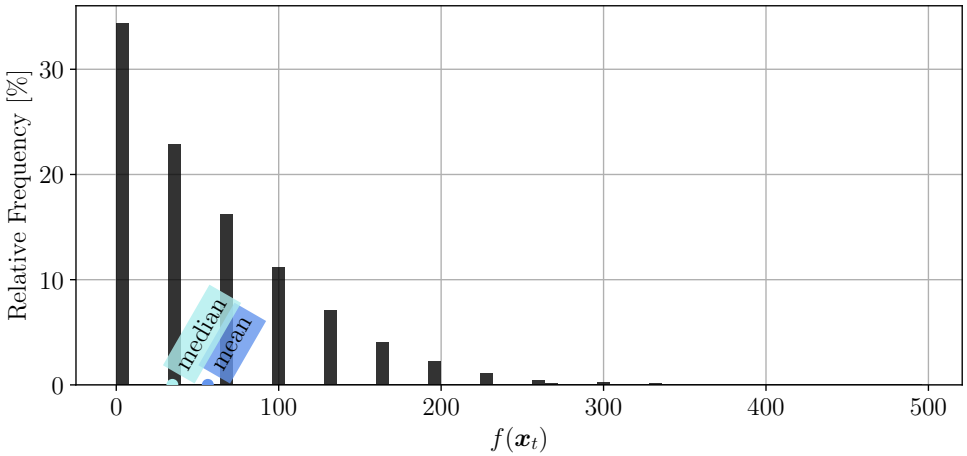


Figure 2. Exemplary Loda outlier score values including the mean and median.

In terms of *Alert Preparation*, it must be distinguished between the modes *single system—multiple algorithms* and *multiple systems—single algorithms*. In general, *Alert Preparation* helps to reduce FP as well as FN by exploiting the strengths of multiple classifiers sending their alerts. For *single system—multiple algorithms*, the classifier processes the same data instance from the stream $\boldsymbol{X}_t$ in parallel; thus, feature interpretability (additional alarm attribute (ii)) is not mandatory since it can strongly be assumed that resulting outliers from the same data instance are caused by the same feature deviating too much from normal behavior.

However, having *multiple systems—single algorithms*, with the feature information causing the outlierness, alerts can be assigned to the same event but from different stream perspectives, e.g., $\boldsymbol{Y}_t$ or $\boldsymbol{Z}_t$, considering the timestamp information as well. Even in the distributed case, an attack might cause outliers close in time to be detected by the classifiers, which subsequently generate alerts. If the classifiers are able to determine the features causing the outlierness, those alerts can be mapped to the same event with high confidentiality whose γ most contributing features have high similarity. Independent of the interaction mode, alerts generated from detected outliers of the same event (data instance) by multiple classifiers can be fused together to reduce the amount of alerts that SOAAPR needs to process (alert filtering). We suggest deriving meta-alerts that summarize the multiple alerts generated by the classifiers for the same event. Having the same alert attributes in terms of IP-address and port source and destination information with timestamps close in time (and highly similar respective features causing the outlierness for the *multiple systems—single algorithms* option) but with different normalized outlier score values, those alerts can be mapped to a single meta-alert. The normalized outlier score values (additional alarm attribute (i)) are combined by $\tilde{f}_{meta} = \sum_i g_i \tilde{f}_i$ into a meta-outlier score depending

on potentially additional weights g_i for each classifier, where $\sum_i g_i = 1$ and $\tilde{f}_{meta} \in [0,1]$ applies. For the sake of simplicity, we assume equal weights in the following.

Table 1 shows an example of five online OD algorithms classifying five data instances with ground truth $\{normal, normal, abnormal, normal, abnormal\}$. Further, we assume an anomaly threshold of >0.5, which triggers the generation of alerts by each classifier. A normal data instance can either be a True Negative (TN) or an FP and an abnormal data point can be either a True Positive (TP) or an FN. Having ground truth information available, even for misclassification of individual classifiers, meta-alerts would be generated with high confidence referring to the meta-outlier score value, denoted as $\tilde{f}^{(GT)}_{i(meta)}$. Since alerts are only generated from TPs and FPs, $\tilde{f}_{i(meta)}$ can lead to falsely generated meta-alerts without incorporating information on how many classifiers generated alerts. Thus, the meta-alert outlier score can be penalized by some measure, e.g., utilizing majority voting or the ratio of alerts generated to the number of classifiers. Taking advantage of the latter leads to a penalized meta-alert outlier score, denoted as $\tilde{f}^{(penalized)}_{i(meta)}$. Setting a dedicated threshold for meta-alert generation based on the penalized outlier scores, for instance, 0.3 in Table 1, significantly reduces FPs and potentially alleviates the number of FNs that limit the alert generation by obscuring the actual TPs.

It is noted that this measure only works for *single system—multiple algorithms* since those algorithms definitely operate on the same input data and the total number of classifiers processing the same event can be determined. In contrast, the number of classifiers that operate distributively in the *multiple systems—single algorithms* case, processing the same event, cannot be reliably determined.

Table 1. Exemplary meta-alert generation based on five online OD algorithms classifying five data instances (3× normal, 2× abnormal) (yellow-colored background for alerts generated by each classifier for True Positives (TP) and False Positives (FP); red-colored background for critical False Negatives (FN); (GT) denotes if Ground Truth about the binary classification was available).

Classifier	Normal	Normal	Abnormal	Normal	Abnormal
#1	TN (0.2)	TN (0.3)	TP (0.9)	FP (0.6)	FN (0.5)
#2	TN (0.1)	FP (0.7)	FN (0.5)	TN (0.3)	TP (0.9)
#3	FP (0.6)	FP (0.6)	TP (0.9)	TN (0.3)	FN (0.4)
#4	TN (0.1)	TN (0.2)	FN (0.5)	TN (0.2)	TP (1.0)
#5	TN (0.2)	TN (0.2)	TP (0.8)	TN (0.1)	FN (0.5)
$g_i^{(GT)}$	0.20	0.20	0.20	0.20	0.20
$\tilde{f}^{(GT)}_{i(meta)}$	0.24	0.40	0.72	0.30	0.66
g_i	1.00	0.50	0.33	1.00	0.50
$\tilde{f}_{i(meta)}$	0.60	0.65	0.87	0.60	0.95
$\frac{\#alerts}{\#classifiers}$	1/5	2/5	3/5	1/5	2/5
$\tilde{f}^{(penalized)}_{i(meta)}$	0.12	0.26	0.52	0.12	0.38
Meta-Alert	✗	✗	✓	✗	✓

3.3. Streaming Alert Correlation and Clustering

The core component of SOAAPR in order to group incoming alerts into clusters, potentially representing attack scenarios, is the *Streaming Alert Correlation/Clustering* component. It is again noted that we are not yet interested in uncovering attack campaigns or multi-stage attacks but attack scenarios or attack steps that are rooted in timely correlated outliers. Although so-called Advanced Persistent Threats are characterized by intelligent adversaries that might exploit the notion of time by stealthily prolonging their attack campaign, we are focusing on the preliminary steps adversaries have to undertake that are of a reasonable finite length of time. For multi-stage attack uncovering, we refer to other work such as [38,54].

Table 2 shows the 14 different attack scenarios of the CICIDS2017 datasets together with their respective characteristics in terms of number of (anomalous) records associated for each attack and their duration. Furthermore, the outlier percentage of each attack within the dataset is given. Since OD is tailored for highly imbalanced data, it will not work well for the *DoS Hulk*, *Portscan* and *DDoS* attack. However, especially for DoS-like attacks, plenty of work exists for detection and prevention [55–57]. It can clearly be seen from the table that, on average, if we exclude the DoS type, attacks are typically not longer than 4 h. Furthermore, we assume that attacks are rooted in a reasonable amount of outliers such that—if clustered—meaningful signatures of those potential attack scenarios can be derived. The number of instances in Table 2 confirms this assumption by showing the average amount of (anomalous) instances per attack scenario and attack category (excluding the three above mentioned) is 2830 with a minimum of 11 for *Heartbleed*. However, we want to note that each attack category has its unique characteristics in terms of the average number of instances per attack scenario and their duration. Thus, it might be reasonable to apply multiple instances of SOAAPR, each adjusted for a dedicated attack category. This would also allow applying SOAAPR with other detection mechanisms, except OD ones, which might also be tailored for DoS-like attacks.

Table 2. The 14 different attacks with respective characteristics of the CICIDS2017 datasets.

Dataset	Attack Type	# Instances	Outliers (%)	Duration (min)
Tuesday-WorkingHours	SSH-Patator	5897	1.32	62
	FTP-Patator	7938	1.78	73
Wednesday-WorkingHours	DoS Hulk	231,073	33.35	24
	DoS GoldenEye	10,293	1.49	9
	DoS Slowloris	5796	0.84	467
	DoS Slowhttptest	5499	0.79	22
	Heartbleed	11	0.002	20
Thursday-WorkingHours-Morning	Web Attack—Brute Force	1507	0.88	45
	Web Attack—XSS	652	0.38	20
	Web Attack—Sql Injection	21	0.01	2
Thursday-WorkingHours-Afternoon	Infiltration *	36	0.01	86
Friday-WorkingHours-Morning	Bot	1966	1.03	205
Friday-WorkingHours-Afternoon	Portscan	158,930	55.48	138
Friday-WorkingHours-Afternoon (2)	DDoS	128,027	56.71	20

* Although labeled as one attack scenario in the dataset, it actually consists of three short-term ones: Meta exploit Win Vista, Infiltration—Cool disk—MAC and Infiltration—Dropbox download—Win Vista.

Furthermore, we deem it more reasonable that each incoming alert is only assigned to one cluster. This stands in contrast to approaches, such as [40], whose community clustering might assign alerts to several clusters under the assumption of dealing with uncertainty in clustering, and, although clusters might contain alerts of unrelated attacks, they might include all TP alerts. This assumption can be seen as critical since alerts of unrelated attack scenarios, denoted in this work as "noisy", might lead to blurred signatures, and similar attack scenarios in the future may lead to completely different signatures without those additional noisy alerts. SOAAPR operates on alerts with high confidentiality of TPs due to the *Alert Preparation* mechanism.

The *Buffer* component serves as a FIFO (First In, First Out) data queue that helps to cope with potential bursts of alert floods while relaxing the processing time of the *Streaming Alert Correlation/Clustering* module. The streaming alert processing is performed as provided with Algorithm 1 for each oldest (first) alert $\boldsymbol{a}_t$ in the *Buffer*. The timely order of alerts is assumed and to be given by the *Alert Preparation* component. As of now, we limit ourselves to five alert attributes for streaming clustering: IP-address (source, destination—$\{ip_{src}, ip_{dst}\}$), port information (source, destination—$\{port_{src}, port_{dst}\}$) and the timestamp indicating the time when the outlier was detected, denoted as t_a. Moreover, one is able to

weight alert attributes, for instance, to give less weight to the source IP-address or source port as an adversary might spoof it during its attack scenario [48].

Algorithm 1: The Operation Principle of *Streaming Alert Correlation / Clustering*.

Input: Alert $\boldsymbol{a}_t$ composed of attributes $\{ip_{src}, ip_{dst}, port_{src}, port_{dst}, t_a\}$, Minimum Similarity min_sim, A cluster's total time to live t_{ttl}

Output: Updated set $C^{(t)}$ of clusters each $C_i^{(t)}$ composed of an alert subset $\boldsymbol{A}_i^{(t)} \subset \boldsymbol{A}_t$, a create timestamp t_c, a last alert added timestamp t_{laa}, an alert counter cnt, an expiry date t_{expiry}, an alert adding frequency mean t_freq_{mean} with its standard deviation t_freq_{std}, a flag $state$ indicating a cluster's state "evolving", "saturated" or "discard"

▷Initialization—add alert to new cluster

if $C^{(t)} == \varnothing$ **then**
  $C_0^{(t)} \leftarrow \boldsymbol{a}_t$ $t_c^{(C_0)} \leftarrow t_a$ $t_{laa}^{(C_0)} \leftarrow t_a$ $t_{expiry}^{(C_0)} = t_c^{(C_0)} + t_{ttl}$
  $cnt^{(C_0)} \leftarrow 1$ $state^{(C_0)} \leftarrow evolving$
$best\_cluster \leftarrow 0$
$highest\_sim \leftarrow 0$

▷Iterate over all existing clusters

for i *in* $C^{(t)}$ **do**
  ▷Clusters are regularly updated—check if cluster is still *evolving*
  if $state^{(C_i)} == evolving$ **then**
    ▷Compute the similarity of the new alert with the cluster (Equations (2) and (3))
    $sim \leftarrow$ **similarity**$(C_i^{(t)}, \boldsymbol{a}_t)$
    if $sim > highest\_sim$ **then**
      $best\_cluster \leftarrow i$
      $highest\_sim \leftarrow sim$
$no\_clusters \leftarrow i$
$i \leftarrow best\_cluster$

▷Add alert to existing cluster or create new cluster

if $highest\_sim \geq min\_sim$ ***AND*** $state^{(C_i)} == evolving$ **then**
  ▷Add alert to existing cluster
  $C_i^{(t)} \leftarrow \boldsymbol{a}_t$ $cnt^{(C_i)} \leftarrow cnt^{(C_i)} + 1$
  $t\_freq_{mean}^{(C_i)} \leftarrow \mathbf{t\_freq_{mean}^{(C_i)}}$**.moving_mean**$(t_a - t_{laa}^{(C_i)})$
  $t\_freq_{std}^{(C_i)} \leftarrow \mathbf{t\_freq_{std}^{(C_i)}}$**.moving_std**$(t_a - t_{laa}^{(C_i)})$
  $t_{laa}^{(C_i)} \leftarrow t_a$
else
  ▷Add alert to new cluster
  $i \leftarrow no\_clusters$ $C_i^{(t)} \leftarrow \boldsymbol{a}_t$ $t_c^{(C_i)} \leftarrow t_a$ $t_{laa}^{(C_i)} \leftarrow t_a$
  $t_{expiry}^{(C_i)} = t_c^{(C_i)} + t_{ttl}$ $cnt^{(C_i)} \leftarrow 1$ $state^{(C_i)} \leftarrow evolving$
return $C^{(t)}$

In order to achieve streaming clustering for our purposes, we extend each cluster $C_i^{(t)}$ with additional properties beyond the simple subset of alerts, which we deem mandatory to answer the following fundamental questions: Firstly, when is a cluster saturated, i.e., when is it ready for the process of signature generation? Secondly, when is a cluster with its alerts considered outdated and should be discarded? To answer those two questions, we refer to Table 2 and define two significant user-definable parameters: a maximum total time to live for a cluster, denoted as t_{ttl}, and a minimum number of alerts that should

be clustered in order to reasonably represent an attack scenario for signature derivation, denoted as min_alerts. Based on those parameters, clusters are assigned the additional information of its create timestamp t_c, a last alert added timestamp t_{laa}, an alert counter cnt, an expiry date t_{expiry}, a flag $state$ indicating a cluster's state *evolving*, *saturated* or *discard* and an alert adding frequency mean t_freq_{mean} and its standard deviation t_freq_{std}. The latter two characteristics are used to determine whether a cluster could be seen as completed in a timely fashion without stressing the very latest expiry date. Thus, as long as the minimum number of alerts per cluster is not reached, the time difference between each alert is computed, and their moving mean and standard deviation are computed by, e.g., the well-known Welford's algorithm [58]. If the minimum number of alerts per cluster is reached, alerts can still continuously be added to a correlated cluster as long as this process happens frequently. However, if no more alerts have been added to a specific cluster for some time, it is considered *saturated* and ready for signature generation. A trivial but effective method is a one-dimensional (time) OD method based on the assumption that, if a value is a certain number of standard deviations away from the mean, that data point is identified as an outlier. The specified number of standard deviations is called the threshold whose value is user-definable, denoted as the frequency exceeding threshold k.

With this set of information, we are able to answer the above questions. Therefore, a cluster is considered *saturated* and ready to *Trigger Signature Generation* when it has reached a minimum number of alerts and (from a time perspective) either the expiry date ($t_{expiry} = t_c + t_{ttl}$) has been exceeded or, for a certain amount of time, no more alerts have been added to it. Both time constraints are given by $t_{now} > t_{expiry}\ OR\ t_{now} > t_{laa} + t\_freq_{mean} + k \cdot t\_freq_{std}$, in which t_{now} is the current time. Both options are mandatory since the expiry date prevents, apart from the circumstance that falsely triggered alarms continuously keep clusters alive, an adversary from unceasingly triggering alerts on purpose such that a cluster is not considered saturated for the duration of the attack concealing its actual one. The second part of the time constraint prevents an adversary from leveraging its attack until the expiry date is met, allowing a maximum attack time period of t_{ttl}. SOAAPR is also robust to attack scenarios that can be performed on sliding window-based approaches, as discussed in [42], in which an attacker can prevent two attack steps from falling into one window. In order to further lower the determinism for an attacker to not exploit either time boundary, a certain amount of jitter might be introduced. Thus, similar to the mechanism in [59], the attacker might not exactly guess both timing constraints.

Algorithm 2 is proposed to monitor the clusters in a regular time-triggered manner (each time step Δt) to check their conditions. This is necessary since, if no alert is streaming in SOAAPR for a longer time span, clusters for the event-triggered case might already be considered *saturated* or to be *discarded*.

For better comprehensibility, Figure 3 shows an exemplary scenario for two observation times (a) and (b) of (timely) evolving clusters in three dimensions—two hypothetical alert attributes and the dimension of time. Although C_{i+1} in (a) might have attribute correlation with C_i, C_i is already considered *saturated* since the alert within C_{i+1} is outdated due to the frequency constraint. A third cluster C_{i+2} is currently in the *evolving* state. Since C_i was ready for signature generation and C_{i+1} could not reach a minimum number of alerts till the expiry date, which was assigned the state *discard*, both have been removed, as shown in state (b). Cluster C_{i+2} is assigned the state *saturated* in (b) since it reached the minimum number of alerts, and no more alerts have been added for a certain amount of time. A fourth cluster C_{i+3} recently added two alerts and thus is in the *evolving* state. Furthermore, a cluster with its corresponding set of alerts can be discarded by the *Discard Alerts and Clusters* module when a signature of a cluster has been computed (assuming that the above conditions are met) or if the minimum number of alerts per cluster could not be reached before the cluster is considered to be expired.

Algorithm 2: Monitoring of *Trigger Signature Generation* and *Discard Alerts and Clusters*.

Input: Time step Δt, Minimum number of alerts *min_alerts*, Frequency exceeding threshold k, Set of clusters $C^{(t)}$

Output: Set of clusters with state "saturated" $C^{(t)}_{saturated}$ and "discard" $C^{(t)}_{discard}$

▷Regularly update clusters

foreach Δt **do**

  $t_{now} \leftarrow$ **current_time()**

  ▷Iterate over all existing clusters

  for i *in* $C^{(t)}$ **do**

    ▷Check if cluster is "saturated" or needs to be "discarded"

    if $(t_{now} > t^{(C_i)}_{expiry})$ ***OR*** $(t_{now} > t^{(C_i)}_{laa} + t\_freq^{(C_i)}_{mean} + k \cdot t\_freq^{(C_i)}_{std})$ **then**

      if $cnt^{(C_i)} \geq min\_alerts$ **then**

        $state^{(C_i)} \leftarrow saturated$

      else

        $state^{(C_i)} \leftarrow discard$

    if $state^{(C_i)} == saturated$ **then**

      $C^{(t)}_{saturated} \leftarrow C^{(t)}_i$

    if $state^{(C_i)} == discard$ **then**

      $C^{(t)}_{discard} \leftarrow C^{(t)}_i$

return $C^{(t)}_{saturated}, C^{(t)}_{discard}$

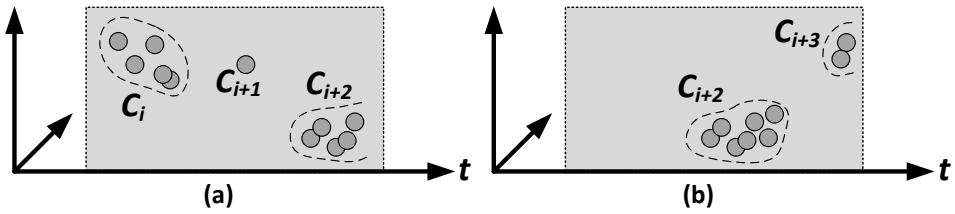


Figure 3. An exemplary scenario of evolving alert clusters for two different observation times (**a**,**b**)—indicated by the frames with gray background—with two hypothetical alert attributes and the timestamp attribute.

Many algorithms for data stream clustering exist that operate on some similarity measure such as the partition-based approaches incremental k-means, HPStream or CluStream, which has already been used for alert correlation in [6], or DenStream, I-DBSCAN or LDBSCAN as density-based cluster methods [60]. For instance, established clusters feature a cluster centroid each in l dimensions, and a new alert, characterized as a l-dimensional data point, could be added to the cluster whose distance to the data point is the lowest. Adding the time t as an additional dimension, one could obtain moving (evolving) clusters in t, which gets completed as time passes, as illustrated in Figure 3. When a new alert is very far away in time from the ones within the cluster, even if the other alert attributes are highly similar, it might not be added to it if the time dimension's weight lets the similarity fall below the minimum similarity. However, we have deliberately decided against this approach for two reasons. Firstly, alert attributes are coordinates in the l-dimensional space, and similarity is only a measure for each coordinate. If one is additionally interested in the similarity between two different coordinates, such as the similarity of the source IP from one alert with the destination IP of another alert, this procedure cannot be utilized. Secondly, the setting of a minimum similarity value *min_sim* as a criterion, whether to add an alert to an existing cluster or not, depends on t which significantly decreases its interpretability. The graph-based approach in GAC allows arbitrary similarity functions,

although the authors limited the attributes to only compare $\{ip_{src}, ip_{dst}, port_{src}, port_{dst}\}$ between any two alerts. It is not able to incorporate timing information, and an edge in the graph between two nodes (alerts) is only added if a minimum similarity value is reached. This fine-granular setting significantly influences the resulting alert similarity graph and the final alert clusters obtained from it. Thus, we have chosen a more general approach by adding an alert to a cluster in a streaming fashion by measuring the similarity of a new alert $\boldsymbol{a}_t$ with the whole i-th cluster $C_i = \{\boldsymbol{a}_j | j \in \mathbb{Z}, 1 \leq j \leq |C_i|\}$ by utilizing Equation (2), where $C_i[j]$ is the j-th alert $\boldsymbol{a}_j$ of C_i. The notion of time is, contrary to [40], included by the time constraints of the expiry date and frequency transgression while not blurring *min_sim* in a timely manner.

$$sim(\boldsymbol{a}_t, C_i) = \frac{\sum_{j=1}^{|C_i|} alert\_sim(\boldsymbol{a}_t, C_i[j])}{|C_i|} \in [0,1] \tag{2}$$

The similarity in between the new alert $\boldsymbol{a}_t$ and the j-th alert $\boldsymbol{a}_j$ of C_i can be computed by Equation (3), utilizing attribute-specific comparison functions, denoted as $f_k(\boldsymbol{a}_t^{(x)}, \boldsymbol{a}_j^{(y)})$, which might be individually weighted by w_k (typically $\sum_{k=1}^{K} w_k = 1$) for a total amount of K comparison functions and x, y not necessarily the same attributes of $\boldsymbol{a}_t$ and $\boldsymbol{a}_j$.

$$alert\_sim(\boldsymbol{a}_t, \boldsymbol{a}_j) = \frac{\sum_{k=1}^{K} w_k \cdot f_k(\boldsymbol{a}_t^{(x)}, \boldsymbol{a}_j^{(y)})}{\sum_{k=1}^{K} w_k} \in [0,1] \tag{3}$$

For network-related alerts, the attributes $\{ip_{src}, ip_{dst}, port_{src}, port_{dst}\}$ are the most important and common ones [40]. However, contrary to GAC with $f_k \in \{0,1\}$, whether the compared attributes are unequal (0) or equal (1), and $x == y$ for each k, we utilize comparison functions shown in Table 3. In addition, with regard to $x \neq y$, we check whether the source IP-address of the new alarm compares to the destination IP-address of an existing alert, which might be an indicator that a host was already compromised, taking into account that the victim communicates with the attacker. The same applies for the identity check of the source port of $\boldsymbol{a}_t$ with the destination port of $\boldsymbol{a}_j$.

Table 3. The proposed comparison functions f_k to compute the similarity between alert $\boldsymbol{a}_t$ and $\boldsymbol{a}_j$, utilizing the alert attributes $\{ip_{src}, ip_{dst}, port_{src}, port_{dst}\}$.

f_k	**Computation**
$f_1 \in \{0,1\}$	$ip_{src}^{(\boldsymbol{a}_t)} \stackrel{?}{=} ip_{src}^{(\boldsymbol{a}_j)}$
$f_2 \in \{0,1\}$	$ip_{dst}^{(\boldsymbol{a}_t)} \stackrel{?}{=} ip_{dst}^{(\boldsymbol{a}_j)}$
$f_3 \in \{0,1\}$	$ip_{src}^{(\boldsymbol{a}_t)} \stackrel{?}{=} ip_{dst}^{(\boldsymbol{a}_j)}$
$f_4 \in \{0,1\}$	$port_{dst}^{(\boldsymbol{a}_t)} \stackrel{?}{=} port_{dst}^{(\boldsymbol{a}_j)}$
$f_5 \in \{0,1\}$	$port_{src}^{(\boldsymbol{a}_t)} \stackrel{?}{=} port_{src}^{(\boldsymbol{a}_j)}$
$f_6 \in \{0,1\}$	$port_{src}^{(\boldsymbol{a}_t)} \stackrel{?}{=} port_{dst}^{(\boldsymbol{a}_j)}$

For iterating over all existing clusters of the current set $C^{(t)}$ with each new alert $\boldsymbol{a}_t$ and calculating the similarity of it with the subset of alerts in each $C_i^{(t)}$, one might assume high complexity in terms of time and space. However, the number of comparison functions K is fixed, and the number of clusters in the current set $|C(t)|$ can be seen as fixed as well since it only fluctuates when new clusters occur, but expired or saturated clusters disappear over time. Thus, with respect to streaming alerts $\boldsymbol{a}_t$, the *Streaming Alert Correlation/Clustering* module has only linear time and space complexity since only the number of alerts in an existing cluster can increase until it is expired, which demands space and time by computing the overall similarity of the new alert with all alerts of each existing cluster. As already mentioned, an attacker might take advantage of producing as many "decoy"

alerts as possible to keep alive and fill a cluster until t_{expiry} is satisfied to stress out the time and space complexity. However, the attacker might only be able to trigger a limited amount of decoy alerts since the alerts are generated by the online OD algorithms, and too many "outliers" would represent a concept drift in $\boldsymbol{X}_t$. Thus, malicious data might be seen as normal after a certain amount of time, and no more alerts that stress SOAAPR are generated. SOAAPR achieves a decent tradeoff between a "real-time" detection by ideally analyzing each individual alert generated by online OD in order to immediately react to an attack (which is impossible in real-world scenarios by the massive amount of streaming alerts afflicted with FP and FN) and a "near real-time" detection with a certain delay by the clustering process to obtain a decent amount and human-manageable set of alerts representing potential attack scenarios.

3.4. Signature Generation and Sharing

Inspired by the idea in [9] to derive privacy-preserving signatures and fingerprint-like characteristics of novel attack patterns by only utilizing the alert information commonly available with IP and port information, we extend it in SOAAPR. Clusters containing a huge number of alerts representing an attack scenario can be significantly reduced to a fixed-sized characteristic by transforming the communication relation of hosts that were involved in an attack into a directed graph-based structure to derive so-called motif signatures initially proposed in [41]. This enables a more fine-grained characterization of attacks compared to other work discussed in Section 2.5, such as [40], only differing between four types of communication patterns. However, with the ever-increasing complexity of novel attacks, we deem that (i) not only the communication relation is a mandatory attack characteristic but also (ii) the data's attributes or features of the data $\boldsymbol{X}_t$, which are mainly responsible for shaping an attack and, thus, causing outliers, as well as (iii) the temporal pattern between the alerts. Considering these three metrics for fingerprinting clusters by deriving three signatures denoted as sig_{com} (i), sig_{attr} (ii) and sig_{temp} (iii), we can achieve a more comprehensive and even more fine-grained characterization and comparison of attack scenarios while still satisfying the privacy-preservation benefit.

We see ourselves encouraged in our assumption of introducing the additional signature sig_{attr} since certain types of attacks and their affected outliers are predominantly caused by the same features. As could be shown in [61], some features have been more significant over multiple attack scenarios, e.g., *B.Packet Len Std*, *Flow Duration* or *Flow IAT Std*, and certain types of attacks are more reflected by dedicated features referring to Table 3 in [61]. For instance, *Subflow Fwd Bytes* and *Total Length Fwd Package* are most influential for Infiltration and Bot attack types, or the *Bwd Packet Length Std* is a typical feature whose outlierness indicates DoS-like attacks [61,62].

Likewise, our assumption proposing that sig_{temp} is strengthened by the statement in [48] that a series of intrusion actions performed by an attacker is more concentrated in the temporal domain than random FPs. As similarly stated in [48], sig_{temp} will likely not characterize a real attack with precision since an attacker might try to manipulate the timing of its attack steps forging the time interval between triggered alerts, or sig_{temp} might be blurred by FP alerts. It is nevertheless a reasonable approach for capturing the temporal behavior of attacks. In particular, it is easier for an attacker to manipulate the timing of a multi-stage attack than for a single step (focus of SOAAPR) since some tools, e.g., Metasploit, are often used, whose execution, resulting in potential alerts, might not be tampered in a timely way.

Although each signature might not fully characterize an attack on its own, such as sig_{com} proposed in [9], the combination of the signature triplet, potentially weighted as well, better allows characterizing attacks and finding novel patterns that somehow share similarities with other signatures. Thus, deviations of one of the signatures from similar attack scenarios can be better compensated by the others. It must be noted that meaningful signatures can be derived when a cluster contains all relevant instances of an attack, ideally free from FPs and FNs, which is an especially ambitious intention of utilizing OD. Having

the knowledge about the communication relation of hosts, whose features are the most important for a certain attack scenario and the timing behavior of a potential attack scenario, significantly provides a more intuitive root cause analysis process for a human expert than analyzing correlated alerts on its own.

3.4.1. Generation and Comparison of sig_{com}

Since it is described in detail in [9], we only provide the most important steps to perform signature generation for sig_{com} and comparison but strongly recommend the original work. In order to derive sig_{com}, we take advantage of the alert attributes $\{ip_{src}, ip_{dst}, port_{src}, port_{dst}\}$ representing the communication structure of two hosts communicating over a port ($ip_{src} : port_{src} \rightarrow ip_{dst} : port_{dst}$). Those attributes are contained in each alert $\boldsymbol{a}_j$ of the i-th cluster C_i, as shown in Figure 4. To *transform* all alerts into a network graph $G_{com}(C_i)$, nodes of the graph either represent hosts via the IP-address $\{ip_{src}, ip_{dst}\}$ or ports that are bound to the respective IP-address $\{ip_{src} : port_{src}, ip_{dst} : port_{dst}\}$. The edges of the graph are reflected by information on who attacked whom: $\{(ip_{src}, ip_{src} : port_{src}), (ip_{src} : port_{src}, ip_{dst} : port_{dst}), (ip_{dst}, ip_{dst} : port_{dst})\}$.

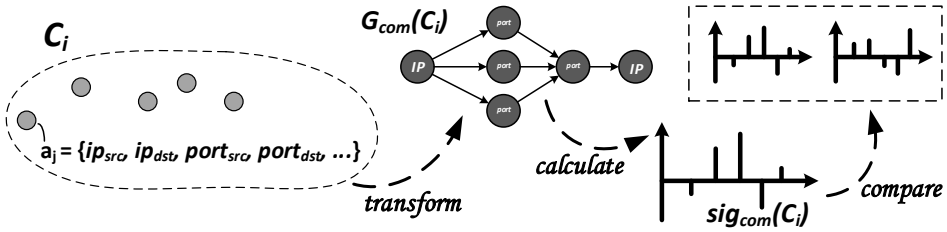


Figure 4. The generation and comparison process of the signature sig_{com} with respect to [9].

With this intuitive communication direction, a directed graph structure is obtained. In order to *calculate* the signature, all sub-graphs $G^*_{com} \subseteq G_{com}$ are enumerated, which are assigned to motif patterns m_i. The accumulated number of occurrences of every m_i will yield an absolute signature M^A. Since M^A depends on the graph size, a comparison can be achieved by utilizing the so-called Z-Score [41]. Briefly summarized, a random graph structure with the same size and the same number of edges as G_{com} is derived and utilized to generate a "relative" signature denoted as M^Z, which represents $sig_{com}(C_i)$ with respect to Figure 4. In order to *compare* two such signatures, M^Z_1 and M^Z_2, they are interpreted as two vectors of a fixed length in a multi-dimensional space. The similarity between M^Z_1 and M^Z_2, independent of their length, can then be derived by calculating the angle ϕ between the two vectors, as shown in Equation (4), in which $< \vec{M}^Z_1, \vec{M}^Z_2 >$ is the inner product and $||\vec{M}^Z_{1/2}||_2$ the Euclidean norms.

$$sim(M^Z_1, M^Z_2) = \frac{cos^{-1}(\phi)}{\pi}; \quad cos(\phi) = \frac{< \vec{M}^Z_1, \vec{M}^Z_2 >}{||\vec{M}^Z_1||_2 \cdot ||\vec{M}^Z_2||_2} \tag{4}$$

3.4.2. Generation and Comparison of sig_{attr}

Considering that for different attack scenarios, certain features are more distinctive, we take advantage of a different approach than the graph-based one for sig_{com} in order to *calculate* and *compare* signatures sig_{attr}. In general, we *transform* feature importance information into a histogram $H_{attr}(C_i)$ describing sig_{attr} that characterizes the potential attack scenario clustered in C_i.

In the example in Figure 5, each alert $\boldsymbol{a}_j$ of the cluster C_i provides the feature importance score values for the top-γ-features of the set $\mathcal{F}$ consisting of d features (score values of other features are set to zero) and the corresponding penalized meta-alert outlier score $\tilde{f}^{(penalized)}_{j(meta)}$. Each feature f_i of $\mathcal{F}$ represents a histogram bin (bucket), and the count (frequency) of this bin is computed by adding up each score per feature $s^{(i)}_f$ weighted

with the outlier score provided by each alert in the cluster. In the example, this leads to $b_{sum}^{(i)} = \sum_j s_{f_i}^{(j)} \tilde{f}_{j(meta)}^{(penalized)}$. Finally, we compute the relative frequency h of each bin by $h^{(f_i)} = \frac{b_{sum}^{(f_i)}}{\sum_i^d b_{sum}^{(f_i)}}$.

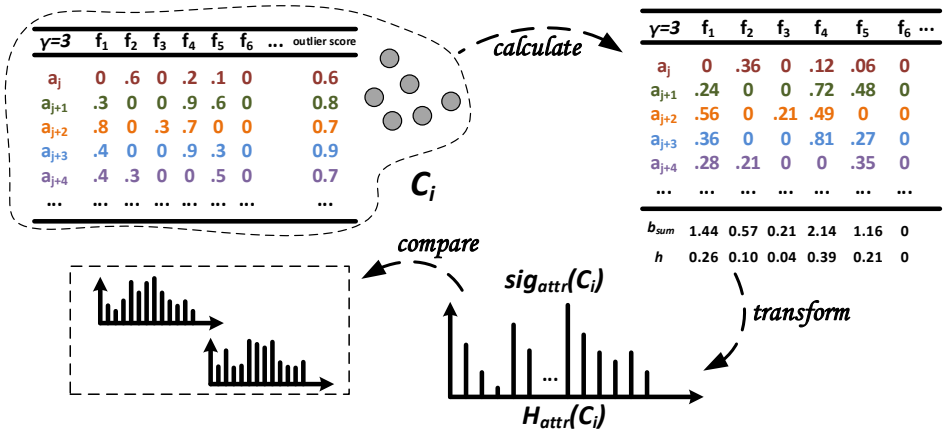


Figure 5. The generation and comparison process of the signature sig_{attr}.

In order to *compare* two signatures, one can take advantage of methods computing the similarity between two statistical distributions, which are represented by any two histograms $H_{attr}(C_i)$ and $H_{attr}(C_j)$ obtained from clusters C_i and C_j. Although the two-sampled Kolmogorov–Smirnov test can be modified to function on discrete data, as it applies for binned values in the histogram, it is normally used for continuous data [63]. Two of the standard choices in the discrete case are the well-known chi-squared test and the Bhattacharyya distance measure [64]. Here, we apply the Bhattacharyya distance $D_B(H_i, H_j)$ between the histograms $H_i = \{h_i^{(b)}\}_{b=1,\ldots,B}$ and $H_j = \{h_j^{(b)}\}_{b=1,\ldots,B}$ for B equi-width bins b with the (relative) frequency h, which is defined as given in Equation (5). It takes values in the range of $(0 \leq D_B \leq \infty)$. BC is the Bhattacharyya Coefficient (Equation (5)) for which $0 \leq BC \leq 1$ applies. We obtain identical histograms and thus identical signatures sig_{attr} between two attack scenarios if $D_B == 0$ applies. The higher the D_B, the more different the signatures are.

$$D_B(H_i, H_j) = -ln(BC(H_i, H_j)); \quad BC(H_i, H_j) = \sum_{b=1}^{B} \sqrt{h_i^{(b)} h_j^{(b)}} \tag{5}$$

It is noted that the applied OD algorithms must be capable of providing the top-γ-features. Although only a limited amount of work is able to satisfy this requirement, it is an important functionality for the design of future anomaly-based IDS [13]. The higher the value for γ, the more information must be transmitted with each alert increasing the communication overhead. Therefore, a decent value for γ could provide a tradeoff between a meaningful signature and a reasonable communication overhead not stressing resources. Furthermore, in order to be able to accumulate a knowledge base of signatures sig_{attr}, they all must have been created by the same feature set $\mathcal{F}$, such that each feature within $\mathcal{F}$ represents the exact same bin in sig_{attr}. However, in many applications, OD algorithms are utilized on the same pre-processed data by systems such as Argus or Bro-IDS, as utilized in [53], or CICFlowMeter-V3, as used in [61]. Nevertheless, a knowledge base should be built up providing signatures for a multitude of commonly applied feature sets such that the user can choose the set, which suits the application best. To what extent signatures obtained from a different amount of γ features but from the same feature set $\mathcal{F}$ can be compared must be evaluated in future work.

3.4.3. Generation and Comparison of sig_{temp}

For generating a signature that represents the temporal (timing) characteristic of an attack, called sig_{temp}, we take advantage of the same procedure as for sig_{attr}. The two metrics, the duration of an attack and its number of respective alerts (with respect to Table 2), extracted from a saturated cluster could be utilized to compare a potential novel attack with a knowledge base consisting of signatures for different attack scenarios with an averaged duration per attack and number of events. However, averaged values do not ideally represent the ground truth. Therefore, a more fine-grained way is to characterize an attack by computing the difference in time, Δt, in between the temporally ordered alerts (events).

With respect to Figure 6, we *transform* the Δt information of alerts from cluster C_i into a histogram with a fixed amount n_bins of equal-width bins b, denoted as $H_{temp}(C_i)$, which describes the signature $sig_{temp}(C_i)$. The histogram ranges from 0 to a subtle maximum value, mostly representing the attack scenarios' best, and the bin-width can be computed dividing the maximum value by the number of bins. If some outlying Δts occur, they are added to the last bin. In order to preserve information about the number of instances, a frequency histogram is proposed; otherwise, one might *calculate* a relative frequency histogram. One might further differentiate between histogram types, such as *short*, *mid* or *long*, to adjust the number and width of the bins, depending on the order of magnitude of Δts. To *compare* two signatures (histograms) of the same type, we again leverage the Bhattacharyya distance measure, as discussed with sig_{attr}.

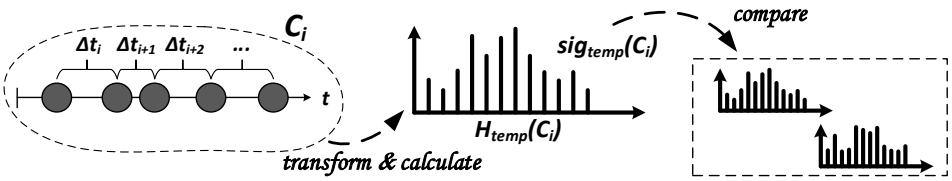


Figure 6. The generation and comparison process of the signature sig_{temp}.

It is noted that the correctness of sig_{temp} critically depends on the correct order of alerts and assumes no strong variation in delays of alerts or a high number of FPs or FNs. However, if the notion of time of IDS sensors is not correctly loosely synchronized or an adversary may tamper with the time, imprecise temporal characteristics of alerts may cause incorrect or confusing results for sig_{temp}. We leave the discussion of this phenomenon for future work.

3.4.4. Handling of the Signatures

Having a certain set of reference scenarios, denoted in this work as the *Knowledge Base*, one can identify novel attack scenarios in the *Runtime* phase by comparing the obtained signatures, sig_{com}, sig_{attr} and sig_{temp}, with existing ones utilizing the *Scenario Comparison* module. Signatures known from misuse-based IDS, anti-virus software or anti-malware systems are either one- or two-dimensional, as Blacklists or Whitelists are examples of the former and regular-expression functions is an example for the latter. A more recent method where multi-dimensional signatures can be seen as ML models that create a multi-dimensional model of input data applying mathematical techniques and score or classify observations. However, formats such as STIX are not compatible with those technologies as each class of threat classification may be founded on completely different trained models [65]. Thus, similar to the idea of conventional signatures, we transfer the idea of having such one-dimensional characteristics to anomaly-based ML systems. What is more, the signatures proposed, sig_{com}, sig_{attr} and sig_{temp}, must not necessarily match the exact signatures from the reference scenarios since the comparison measures can help to identify the similarity of novel patterns to existing ones from the *Knowledge Base*. Over time, the set of reference scenarios might grow very strongly. Thus, it might be time consuming to compare a novel attack pattern composed of the three signatures with

each one in the *Knowledge Base*. Thus, the *Unsupervised Mining* module makes it possible to apply hierarchical clustering in the *Learning* phase, which clusters similar unknown attack pattern before presenting them to the *Human Analyst*. Hierarchical clustering was also proposed by [9] but only for similar sig_{com}. Having a tree-like structure within the *Knowledge Base*, attack scenarios characterized by the three-tuple of signatures can be structured into clusters according to their similarities. For each of the hierarchical clusters, the signatures that represent the most of each sub-cluster can be determined to compare the novel attack pattern much faster by only comparing it with the representing signatures $sig_{com}^{(Ref)}$, $sig_{attr}^{(Ref)}$ and $sig_{temp}^{(Ref)}$. The overall best signature match can be obtained by $max(sig_{com}, sig_{com}^{(Ref)}) \wedge min(sig_{attr}, sig_{attr}^{(Ref)}) \wedge min(sig_{temp}, sig_{temp}^{(Ref)})$. Since none of the signatures contain privacy-relevant information, they can be enriched with additional attack information and, e.g., shared among companies using STIX in an automated manner [66].

4. Experimental Evaluation

4.1. Methodology and Settings

With respect to Figure 1, we split our evaluation of SOAAPR into two parts. Firstly (i), the *Streaming Alert Correlation/Clustering* module along with *Trigger Signature Generation* and *Discard Alerts and Clusters* is evaluated. Secondly (ii), the *Signature Generation* as well as the *Scenario Comparison* capability of SOAAPR are evaluated. For (i), only the traffic labeled CSV files of CICIDS2017 are parsed to derive the ideal number of instances per attack scenario serving as the ground truth clusters. Then, we iterate over the datasets and simulate an anomaly-based IDS by generating alerts that are fed into SOAAPR. Our proposed *Streaming Alert Correlation/Clustering* is used beside the competitor GAC to evaluate and compare their clustering capability. Since GAC does not rely on any intrusion type attribute for clustering, it could be leveraged for anomaly-based IDS, and its outcome can be used to generate motif-signatures sig_{com} by design. Furthermore, its chunk processing could be regarded as a trivial form of online processing. For those reasons, we see a comparison between SOAAPR with GAC as reasonable.

For (ii), we generate signatures sig_{com} as part of the *Signature Generation* module, leveraging ideally clustered attack scenarios from CICIDS2017 and comparing them (*Scenario Comparison* module) in order to show their applicability for attack characterization based on OD. In order to derive signatures sig_{attr}, the supervised RF algorithm is used on a selection of attack scenarios from CICIDS2017 and CSE-CIC-IDS2018, each extracted into a separate CSV file. The reason for choosing the supervised RF algorithm instead of an online unsupervised OD variant is the more reliable feature importance scoring functionality providing better interpretability for sig_{attr} evaluation, which can be obtained by the SHAP method. Another benefit of the supervised approach is that the outlier percentages do not influence the feature importance scoring functionality. Therefore, we could extract arbitrary attack scenarios, even those that are not characterized with a low anomaly percentage. Thus, evaluations of unsupervised online OD methods along with the *Alert Preparation* module are part of future work. We then produce clusters from alerts utilizing the top-γ-features and the outlier scores per instance obtained by multiple RF-instances (*single system—multiple algorithms*) in order to generate and compare signatures sig_{attr}. For the third signature sig_{temp}, we compute the Δts from each ideal attack cluster within CICIDS2017 and CSE-CIC-IDS2018 to generate and compare signatures.

Experiments were conducted on a virtualized Ubuntu 20.04.1 LTS equipped with 12 Intel(R) Xeon(R) CPU E5-2430 at 2.20 GHz and 32 GB memory running on a Proxmox server environment. Programs are coded in Python 3.9 using the latest PyCharm 2021.1.3 environment. GAC for graph representation and clustering as well as sig_{com} for graph representation and motif comparison utilizes the *igraph* (https://igraph.org/python/ (accessed on 25 June 2021) and *networkx* (https://github.com/networkx (accessed on 25 June 2021) libraries. RF is taken from *sklearn* (https://scikit-learn.org (accessed on 25 June 2021), SHAP from its respective library (https://pypi.org/project/shap/ (accessed on 25 June 2021) and SOAAPR's sig_{attr} as well as sig_{temp} generation rely on histograms generated with the pop-

ular *numpy* (https://numpy.org/(accessed on 25 June 2021) library. Across all clustering evaluations, GAC is configured with the default hyperparameters proposed by [40] with a clique size of 15, a similarity threshold between alerts of 0.25 and a chunk size of 5000 subsequent alerts since higher numbers of alerts increase the graph size and its computational complexity heavily. For an equal comparison with SOAAPR, we did not take advantage of parallel processing. SOAAPR's hyperparameters are min_sim, t_{ttl}, min_alerts and the frequency exceeding threshold k. Although SOAAPR allows six comparison functions with respect to Table 3, to ensure equal conditions in comparing with GAC, only f_1, f_2, f_4 and f_5 have been used. Unless otherwise stated, SOAAPR's streaming clustering parameters are set to $min\_sim = 0.25$, $t_{ttl} = 1day$, $min\_alerts = 8$ and $k = 50$. The hyperparameters provided decent results across all datasets and attack scenarios. The minimum similarity min_sim is set equally to GAC, and the total time to live t_{ttl} is set to one day since none of the attack scenarios present in CICIDS2017 and CSE-CIC-IDS2018 exceeds this boundary, and we are able to capture all alerts. The minimum number of alerts is set to 8 to capture attack scenarios with even a low number of associated alerts, e.g., Heartbleed in CICIDS2017. The frequency exceeding threshold is set very high in order to capture any fluctuations of Δts throughout all attack scenarios, which is also amplified by the poor time stamping quality present in CICIDS2017 and CSE-CIC-IDS2018.

4.2. Data Source

For the evaluation of the alert correlation, much of the work—even recent works [38,67,68]—applies the outdated DARPA 2000, having two scenarios LLDOS 1.0 and LLDOS 2.0.2 based on the output of the (misuse-based) ISS RealSecure IDS due to the lack of labeled alert data or attack data that can be used for the purpose of attack detection. Haas et al. use, apart from their synthetically generated alert data set, a real-world dataset from the Internet Storm Center (SANS Technology Institute, Internet Storm Center, https://isc.sans.edu/ (accessed on 25 June 2021) operating a platform called DShield for sharing data from security devices to evaluate their GAC clustering approach. However, this data source lacks ground truth information such that generated clusters by GAC or our SOAAPR approach cannot be evaluated for their accuracy. Since we are interested in the complete processing pipeline starting from the detection of outliers in $\boldsymbol{X}_t$ by the online OD algorithms, we set our focus on recent IDS datasets such as CICIDS2017 [61] and CSE-CIC-IDS2018 (License: https://registry.opendata.aws/cse-cic-ids2018/ (accessed on 25 June 2021) [61] provided by the University of New Brunswick on AWS or UNSW-NB15 since long-serving and still widely used datasets, such as KDD Cup 99 (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed on 25 June 2021) or NSL-KDD (https://www.unb.ca/cic/datasets/nsl.html (accessed on 25 June 2021), have been criticized by many researchers over the past couple of years [61,69]. Especially for the evaluation of anomaly-based IDS methods, the latest updated datasets, such as CSE-CIC-IDS2018, should be utilized [70].

Although CSE-CIC-IDS2018 is tailored for the evaluation of anomaly detection and consists of seven attack categories with a total of 14 different types of intrusions, e.g., SSH-BruteForce or DDoS-LOIC-UDP, it only provides statistical traffic features obtained by CICFlowMeter-V3 and saved as a CSV file. However, in order to generate meaningful alerts for SOAAPR, the typical TCP/IP level network traffic header features, IP-address and port number are mandatory. Furthermore, in some cases, timestamps from data record $n + 1$ is older than the timestamp from the data record n. Therefore, sorting the dataset according to its timestamp feature is necessary in order to preserve the chronological sequence of events. UNSW-NB15 provides IP and port feature information and has a huge variety of attacks and subtypes of attacks, but the corresponding ground truth CSV file cannot be properly used to map attack scenarios to each anomalous instance due to the unclear start and ending timestamps of events rather than assigning clear temporal-ordered timestamps to the dataset records. Although the ML CSV files from CICIDS2017 do not feature IP and port information, the additional available labeled traffic CSV files do, and these can be used to gather the mandatory information since both CSV files have the same timestamps

and number of records. However, there are two drawbacks with CICIDS2017. Firstly, each of the 14 attack scenarios, with respect to Table 2, only occur once, which hampers the signature comparison of similar attack scenarios. Secondly, the units of the timestamp are only provided in minutes. Since CSE-CIC-IDS2018 provides similar attack scenarios, e.g., two *Brute Force - Web* attacks, and timestamps are provided in a more fine-granular fashion, it is at least utilized for measurements where no IP and port information is required, such as for the sig_{attr} and sig_{temp} evaluation. Thus, although no accurate timestamping is provided, we ambitiously tried to compare attack scenarios using different datasets of CICIDS2017 and CSE-CIC-IDS2018. In order to encounter the timestamp problem in CICIDS2017, synthetic timestamps are inserted for data instances from an attack scenario assigned the same timestamp in minutes. However, in a real-world scenario, we might be able to assign more fine-grained units, such as milliseconds. In order to be able to generate meaningful temporal signatures, we introduce smaller units by assuming an equal Δt of alerts have timestamps from the same minute. We take into account blurred signatures for sig_{temp} on CICIDS2017 but see the potential of this fingerprinting using the more accurate results of CSE-CIC-IDS2018. Since our aim is to mine information from the outcome of OD algorithms, for the evaluation of SOAAPR's and GAC's clustering, we excluded the DoS Hulk attack scenario in the CICIDS2017 *Wednesday-WorkingHours* dataset and completely neglected the *Friday-WorkingHours-Afternoon* datasets due to the high percentage of outliers with respect to Table 2. The streaming clustering of SOAAPR depends on timely sorted alerts, which is why all alerts in CICIDS2017 have been sorted.

4.3. Evaluation Criteria

In terms of evaluation metrics for alert correlation, we rely on four different metrics. Two have been proposed by Ning et al. [71] called *completeness* and *soundness*. The former, also known as the true detection rate, is denoted as $COMP$ and calculated by the ratio of the number of correctly correlated alerts (CCA) divided by the number of related alerts RA, i.e., the real attack-scenario-related number of instances, which states the ground truth of each attack scenario (Equation (6)). The latter, denoted as $SOUND$, is the ratio of CCA and correlated alerts (CA), i.e., all the clustered alerts in C_i (Equation (7)). A further metric—the Jaccard index—will provide the similarity of the ideal cluster (ideal attack scenario) and C_i obtained by the alert correlation system. It compares two sets of elements, which, in this case, are the ideal cluster and the obtained cluster, and provides information about which alerts are shared between the two sets and which are distinct. It is denoted as JAC and computed by Equation (8). It takes values in the range of $[0, 1]$ and yields a higher value the more similar two sets are. Further metrics, including the compression rate, that might be interesting for systems afflicted with a high number of FP are available in [8].

$$COMP = \frac{\#CCA}{\#RA} \tag{6}$$

$$SOUND = \frac{\#CCA}{\#CA} \tag{7}$$

$$JAC = \frac{|RA \cap CA|}{|RA \cup CA|} \tag{8}$$

Furthermore, we measure the average runtime for GAC and SOAAPR's streaming clustering as a representative metric for computational performance. Thus, we accumulated the elapsed time for processing each dataset until the final clusters are obtained.

In terms of attack characterization, for each signature sig_{com}, sig_{attr} and sig_{temp} per attack scenario, we compute the similarity values in between each attack scenario with respect to the formulas given in Section 3.4. Furthermore, we measure the average runtime to generate each signature in order to compare the computational performance of sig_{com}, sig_{attr} and sig_{temp} of varying cluster sizes.

5. Discussion of Results

5.1. SOAAPR Clustering

Table 4 summarizes the clustering results of SOAAPR and GAC for the 11 selected attack scenarios of the CICIDS2017 datasets. It shows how many clusters were generated by each algorithm and how many clusters are assigned to each attack scenario. Furthermore, the number of associated alerts contained in the assigned clusters is provided together with the total number of ideal alerts that represents each attack scenario. With respect to Equations (6)–(8), the metrics are provided for the clustering performance and the overall elapsed time to cluster the attack scenarios for each dataset.

Table 4. Clustering performance results of SOAAPR and GAC on 11 attack scenarios of the CICIDS2017 datasets (No. Clusters—assigned cluster(s)/generated cluster(s); No. Alerts—assigned alerts/actual number of alerts per attack scenario; Metrics—$COMP/SOUND/JAC$; Time denotes the total amount of processing time for each dataset containing the respective attack scenarios).

Attack Scenario	SOAAPR				GAC			
	No. Clusters	No. Alerts	Metrics	Time (s)	No. Clusters	No. Alerts	Metrics	Time (s)
FTP-Patator	1/2	7938/7938	1.0/1.0/1.0	105.11	2/3 [1]	10,000/7938	1.0/1.0/0.92	1424.17
SSH-Patator	1/2	5897/5897	1.0/1.0/1.0		2/3 [1]	8835/5897	1.0/1.0/0.88	
WA—Brute Force	1/24	1507/1507	1.0/1.0/1.0		1/1	2180/2180	1.0/0.69/0.69	
WA—XSS	21/24	652/652	1.0/1.0/1.0	2.32	1/1	2180/2180	1.0/0.30/0.30	266.56
WA—Sql Injection	2/24	16/21	0.76/1.0/0.76		1/1	2180/2180	1.0/0.01/0.01	
DoS GoldenEye	2/4 [2]	4065/10,293	0.39/1.0/0.39		3/5 [1]	11,588/10,293	1.0/1.0/0.96	
DoS Slowloris	1/4 [2]	3588/5796	0.62/1.0/0.62	63.62	2/5 [1]	10,000/5796	1.0/1.0/0.94	12,107.17
DoS Slowhttptest	1/4 [2]	5501/5499	1.0/1.0/1.0 [3]		2/5 [1]	10,000/5499	1.0/1.0/0.81	
Heartbleed	0/4 [2]	0/11	0.0/0.0/0.0		0/5 [1]	0/11	0.0/0.0/0.0	
Infiltration	1/1	36/36	1.0/1.0/1.0	∼0.0	1/1	36/36	1.0/1.0/1.0	∼0.0
Bot	2/2	1926/1966	0.98/1.0/0.98	2.51	2/2	1962/1966	1.0/1.0/1.0 [3]	75.09

[1] Due to the "out of memory" error using 32 GB RAM, the alerts had to be processed in chunks of 5000, as proposed in [40]; [2] Changed hyperparameter set; [3] rounded values.

SOAAPR perfectly clusters both brute force scenarios, *FTP-Patator* and *SSH-Patator*, each in a separate cluster while only needing approximately 105 s. For GAC's graph-based clustering, in contrast, a high amount of alerts needed to be processed in chunks since 32 GB RAM on the evaluation machine were not enough. Nevertheless, the processing time was significantly higher by a factor of approximately 13. Three clusters have been generated by GAC, which split each attack scenario into two halves. Thus, both attacks have been assigned to two clusters each. Due to the process of splitting, GAC even misses some alerts for both scenarios, which yielded a JAC value lower than 1. For the sake of visualization, the sunburst diagram in Figure 7a is given showing GAC's clustering results.

GAC clustered the three web attack scenarios into one cluster completely since the alert attributes are highly similar and it cannot differentiate them in a timely way. In contrast, SOAAPR did generate 24 clusters and captured the *Brute Force* scenario into one cluster completely while splitting the *Sql Injection* into two clusters and *XSS* into 21, as depicted in the sunburst diagram of Figure 7b. The splitting of *XSS* is due to the fact that the first *min_alerts* of the attack scenario are timestamped in the same minute; thus, a meaningful frequency exceeding factor k cannot be built. Furthermore, all attack scenarios have similar alert attributes such that decreasing t_{ttl} would split the scenarios in a better manner. Setting $t_{ttl} = 45$ min (duration of *Brute Force* attack scenario) and increasing $min\_alerts = 500$ resulted in two clusters while capturing the *Brute Force* in one cluster with $COMP = SOUND = JAC = 1.0$ and *XSS* into the other with $COMP = 1.0$, $SOUND = 0.97$ and $JAC = 0.97$.

With the default hyperparameter setting, SOAAPR generates only one cluster for the DoS-like attacks and Heartbleed scenario. This is due to the fact that all of those attack scenarios have similar alert attributes. Slightly increasing the *min_sim* parameter to 0.5 instead of 0.25 generates three clusters in which all DoS-attacks are grouped together in one cluster, and the Heartbleed attack is perfectly assigned into another cluster with $COMP = SOUND = JAC = 1.0$. Both hyperparameter settings result in an approximate processing

time of 500 s. However, Table 4 shows the results of a hyperparameter set adapted to capture each DoS-attack. It was already mentioned that in real-world applications, it might be feasible to apply different GAC instances, each parameterized for mining a dedicated attack scenario. Thus, we decreased t_{ttl} to one hour and set the *min_alerts* to 2000 (DoS-attacks are typically characterized by a high number of events) and the frequency exceeding factor k to 300 since the Δts are typically quite low for DoS-attacks in order to be sure that each attack scenario is completed in a timely manner. Since t_{ttl} was decreased, SOAAPR's processing time was significantly reduced to approximately 60 s because it can discard clusters after t_{ttl} was exceeded, which is in contrast to GAC, with over a 3-hour processing time, an impressive speed-up. GAC again needed to process the attack scenarios in chunks, which resulted in five total clusters. For better illustration, the clustering results of the DoS-attacks and Heartbleed utilizing GAC (c) and SOAAPR (b) is depicted as sunburst diagrams in Figure 7.

In terms of the *Infiltration* and *Bot* attack scenario, both algorithms performed almost similarly. Especially with the low amount of alerts for *Infiltration*, GAC could compete with our approach. However, with the 1966 alerts of the Bot attack, GAC's processing time is a factor of 30 higher. Just decreasing the frequency exceeding factor k for the *Infiltration* dataset, SOAAPR generates two or three clusters instead of one, which better represents the ground truth of three individual infiltration attacks instead of the labeled single one referring to the footnote in Table 2.

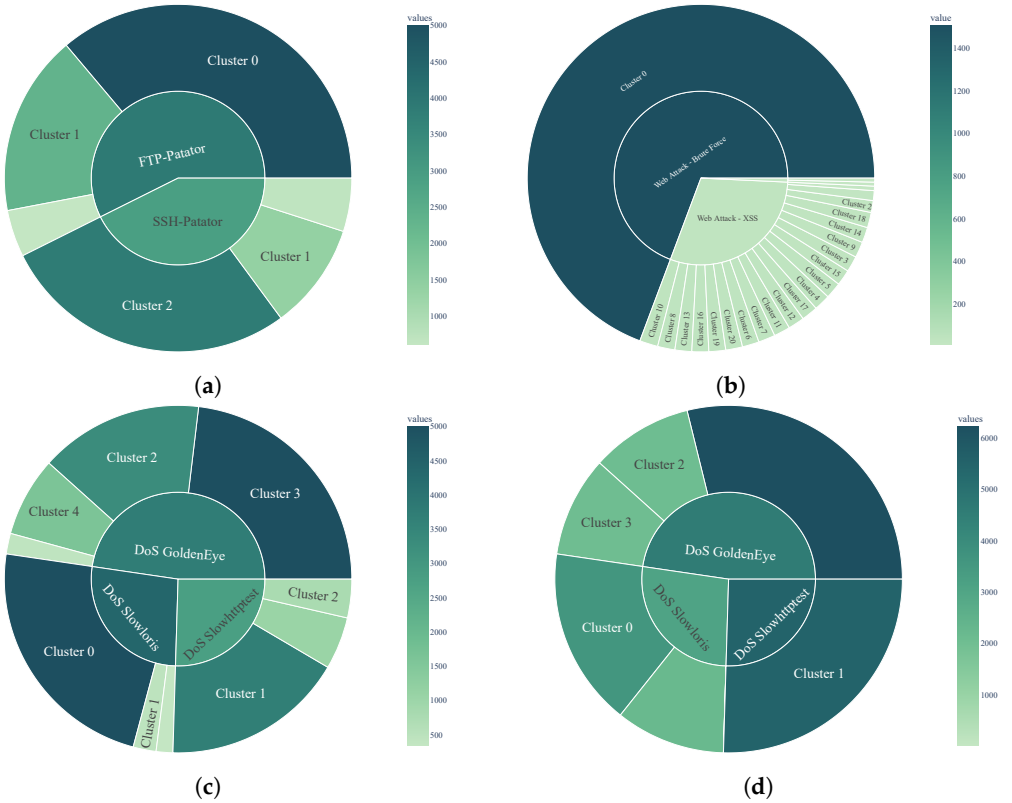


Figure 7. Sunburst diagrams for the clustering performance of (**a**) GAC on FTP and SSH Brute Force, (**b**) SOAAPR on the web attacks *Brute Force* (1 cluster), *XSS* (21 clusters) and *Sql Injection* (2 clusters), (**c**) GAC and (**d**) SOAAPR on the DoS-attacks.

As of now, only ideal alerts (only TPs) have been considered and fed into the alert correlation methods. However, in real-world applications, alert correlation has to deal with FPs and FNs. In preliminary measurements, we introduced a certain confidence level that steers the interspersion of FPs and FNs with a certain probability. From the binary classification result providing the amount of TPs, TNs, FPs, and FNs, we can derive the so-called $F1$-score as the harmonic mean of precision and sensitivity by $F1 = \frac{TP}{TP+\frac{1}{2}\times(FP+FN)}$,

which is often used as a metric on imbalanced data [7]. The effects of FPs and FNs on the clustering result are exemplarily discussed for the *Bot* attack scenario for which both SOAAPR and GAC achieved good results, and the number of alerts is more meaningful compared to *Infiltration*. Introducing approximately 190 FPs and FNs yields a $F1 = 0.90$, and SOAAPR as well as GAC perfectly capture alerts into two clusters as with only TPs (Table 4), but both generate additional clusters. Those are denoted as ghost-clusters (SOAAPR—three, GAC—two) and are mainly caused by the FPs. The FNs only reduce the amount of alerts inside the clusters and can be seen less critical since the consecutive signature generation will also work if the cluster still contains a majority of representative alerts characterizing the attack. Injecting a higher number of FPs (approximately 1900) and FNs (approximately 400), yielding $F1 = 0.58$, will again cause the generation of the two *Bot*-related clusters (with the reduced amount of FN-alerts) but is associated with a significantly higher number of ghost-clusters (SOAAPR—40, GAC—28). However, SOAAPR can easily be adjusted to deal with those ghost-clusters, which, in total, have an average size of 40 alerts by increasing the *min_alerts* parameter to 100. Then, SOAAPR reduces the number of ghost-clusters to two while still capturing the two *Bot*-related clusters in approximately 4.6 s. GAC, in contrast, has no possibility of reducing ghost-clusters and takes approximately 79 s.

5.2. SOAAPR Signaturing

For each attack scenario in either CICIDS2017 (sig_{com}, sig_{attr} and sig_{temp}) or CI-CIDS2017 and CSE-CIC-IDS2018 (sig_{attr} and sig_{temp}), signatures are derived, and the similarity in between each attack scenario's signature is computed. As a result, we obtained an upper triangular matrix of similarity values from which we applied hierarchical/agglomerative clustering using the average-linkage method yielding a dendrogram for the sake of better visualization. As of now, we reserve evaluations showing the effects of FPs and FNs on the signature comparison for future work and only consider ideally clustered attack scenarios.

5.2.1. sig_{com}

Table 5 provides the sig_{com} similarity values between the 11 evaluated CICIDS2017 attack scenarios. It can clearly be seen from the table that attack scenarios that share the same typical communication relation between attacker and victim have high similarity. For instance, the brute force *FTP-/SSH-Patator* attacks have a strong correlation in terms of sig_{com} with a value of 0.9905 in the range of $(0, 1)$ since a single attacker IP with varying source ports attacks a single victim IP and a dedicated port (either FTP—21 or SSH—22). In contrast, *Heartbleed*, characterized by a single attacker IP from a single port attacking a single victim IP on a single port, differs significantly from all other attack scenarios and only shares a value of approximately 0.3 in similarity with the FTP and SSH brute force attacks. All three DoS-like attacks share a similar communication pattern having a similarity value of above 0.97 amongst each other.

To guide the reader better, Figure 8 visualizes the results of Table 5 in a dendrogram. Attack scenarios sharing high similarity are clustered together, such as in the DoS-attacks or the *FTP-/SSH-Patator* attack scenarios. The web attacks, *Brute Force*, *XSS* and *Sql Injection*, are extremely similar as well.

Most of the attack scenarios strongly share a similar communication relation with values approximately above 0.7 with respect to Table 5. In most of the attack scenarios, the attacker and victim share an oto-relation in terms of IP and an mto-relation (referring to [40]) in terms of port information. The main difference is the amount of varying source ports, which is higher the more alerts are present. Thus, DoS-like attacks having a larger amount of varying source ports lead to more nodes within the G_{com} graph and thus a slightly less similar communication relation, as, for instance, with web attack scenarios. In contrast, *Bot* and *Infiltration* differ from the rest of the attack scenarios, and *Heartbleed*,

especially, is the only attack scenario that is characterized by an oto communication relation in terms of IP and port information, thus being the least similar to all others.

Table 5. The upper triangular matrix of sig_{com} similarities between CICIDS2017 attack scenarios (WA—Web Attack).

	Bot	Infiltration	WA—Brute Force	WA—XSS	WA—Sql Injection	FTP-Patator	SSH-Patator	DoS slowloris	DoS Slow-httptest	DoS Golden-Eye	Heartbleed
Bot	-	0.7903	0.7477	0.7742	0.8068	0.6861	0.6955	0.6874	0.6868	0.6595	0.3085
Infiltration		-	0.8610	0.8731	0.9120	0.8210	0.8279	0.8220	0.8216	0.8006	0.3517
WA—Brute Force			-	0.9734	0.9251	0.9382	0.9477	0.9395	0.9389	0.9115	0.2908
WA—XSS				-	0.9448	0.9115	0.9210	0.9128	0.9123	0.8849	0.2870
WA—Sql Injection					-	0.8698	0.8787	0.8711	0.8705	0.8446	0.2807
FTP-Patator						-	0.9905	0.9987	0.9993	0.9734	0.3049
SSH-Patator							-	0.9918	0.9912	0.9639	0.3021
DoS slowloris								-	0.9994	0.9721	0.3045
DoS Slowhttptest									-	0.9727	0.3047
DoS GoldenEye										-	0.3134
Heartbleed											-

One of the intentions of introducing a motif-approach by Haas et al. in [9] was to provide the most fine-grained attack characterization as possible, with GAC only identifying four pre-defined cluster classes (oto, otm, mto and mtm) [40]. Although containing slightly more information within the graph-structure by indirectly incorporating the amount of varying IP or port information in the form of nodes in the graph, as assumed, sig_{com} alone does not characterize individual attack scenarios in a more fine-grained way. Excluding the privacy-preservation capability of the motif-approach, one might even leverage the pre-defined cluster classes and count the frequency of unique IP or port data values to obtain similar information gain as with sig_{com}, which—especially with a higher number of clustered alerts—might be far less computationally complex than the graph-based approach. The higher the number, the longer it takes to generate each attack's signature. *Infiltration* with 36 alerts takes only about 0.02 s, whereas *Bot* with approximately 2000 alerts takes around 10 s. *FTP-/SSH-Patator* brute force clusters with around 14,000 alerts take around 3 min, and the three DoS-attacks, including *Heartbleed*, with respect to Table 2, take approximately 10 min to be generated. The processing time for sig_{com} shows the exponential behavior over the number of alerts.

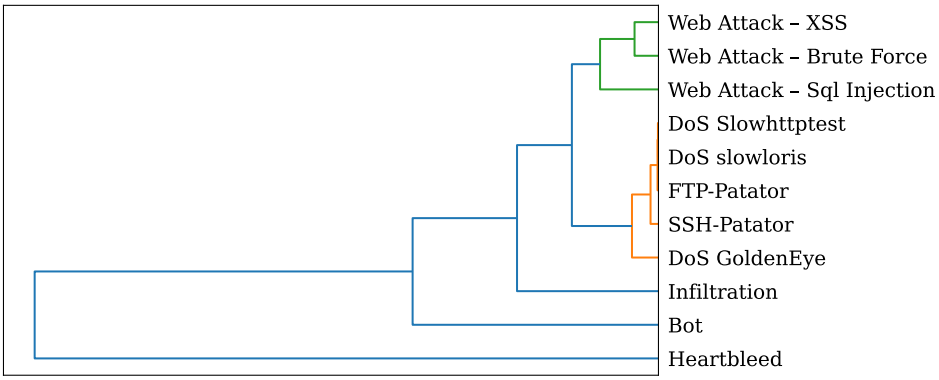


Figure 8. Hierarchical clustering of similarities between CICIDS2017 attack scenarios based on sig_{com}.

5.2.2. sig_{attr}

In order for the RF classifier to generate feature importance scores, each attack scenario has been extracted into a separate CSV file only containing benign and no malicious data. Since DoS-like, Portscan or Brute Force attacks are typically not the scope of OD algorithms,

we limit ourselves to the web attack (Brute Force, XSS, Sql Injection), Infiltration and Bot scenarios of CICIDS2017 (refer to Table 2) and CSE-CIC-IDS2018 (refer to Table 6).

Table 6. A selection of attack scenarios with respective characteristics of the CSE-CIC-IDS2018 dataset.

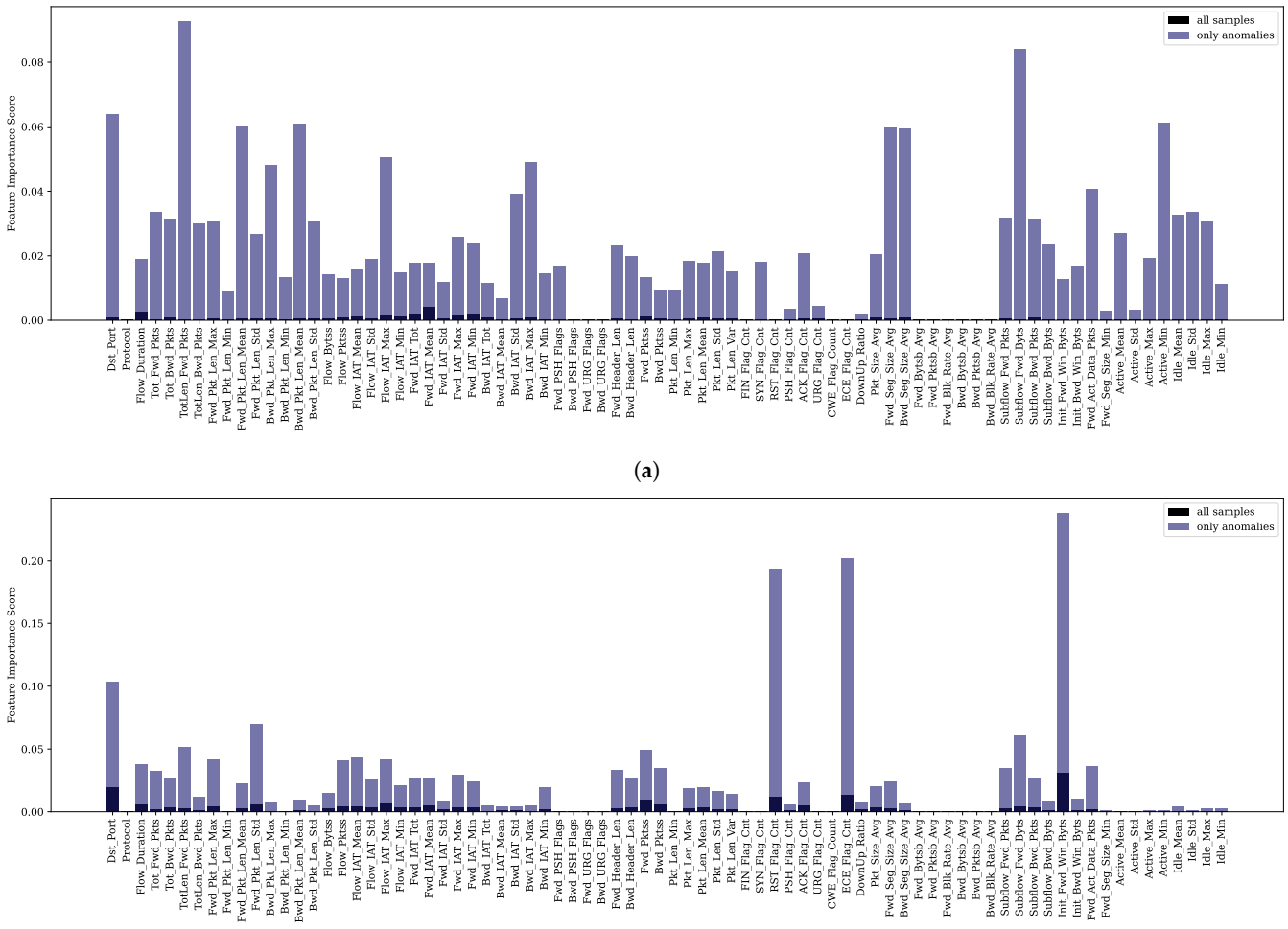
Dataset	Attack Type	# Instances	Outliers (%)	Duration (min)
Thursday-22-02	Brute-Force-Web-0	250	0.023	56.01
	Brute-Force-XSS-0	81	0.008	0.90
	SQL-Injection-0	31	0.003	13.35
Friday-23-02	Brute-Force-Web-1	363	0.035	48.85
	Brute-Force-XSS-1	151	0.014	69.02
	SQL-Injection-1	50	0.005	0.97
Wednesday-28-02	Infiltration-1	42,760	6.974	22.25
	Infiltration-0	26,111	4.259	6.00
Thursday-01-03	Infiltration-3	54,311	16.403	96.98
	Infiltration-2	38,752	11.704	57.98
Friday-02-03	Bot-0	190,240	18.143	473.27
	Bot-1	95,951	9.151	89.82

Founded on the assumption that the feature importance is a representative characteristic for an attack scenario, as discussed by the authors in [61], two exemplary signatures sig_{attr} generated by RF's feature importance scoring applied on both data sources CICIDS2017 and CSE-CIC-IDS2018 are provided in Figure 9. In order to compare attack scenarios from both datasets, the same set of 78 features had to be applied. With respect to Figure 9a, the *TotLen_Fwd_Pkts* and *Subflow_Fwd_Byts* are the most important features, which is also stated in [61] for the *Infiltration* attack scenario. For the web attack *Brute Force* scenario, the most important feature is the *Init_Fwd_Win_Byts*. Since [61] only provides feature importance for web attacks in general and not individually, as done in this work, the other two highly important features *RST_Flag_Cnt* and *ECE_Flag_Cnt*, with respect to Figure 9b, seem characteristic for this sub-attack category.

Choosing an appropriate γ is not only crucial for the amount of information an alert has to carry but also affects the similarity between the sig_{attr} of two attack scenarios. Therefore, we have computed similarity in the form of the Bhattacharyya distance over the top-γ-features between each two attack scenarios. Figure 10 shows the results of this measurement, respectively, for the two strongly similar attack scenarios of the CSE-CIC-IDS2018 dataset *SQL-Injection-0* and *SQL-Injection-1*, as well as the highly dissimilar scenarios of *SQL-Injection-0* from CSE-CIC-IDS2018 and *Bot* from CSECIC2017.

From those measurements, we obtain multiple insights on γ's effects. Firstly, with low γ values, attack scenarios are less similar, in general, but dissimilar attack scenarios have a higher magnitude for the Bhattacharyya distance. For instance, the Bhattacharyya distance for low γ in Figure 10a is approximately 0.5 and for the dissimilar attack scenarios in Figure 10b approximately 1.2. Secondly, the more similar two arbitrary attack scenarios are, the more sharply the curve will fall to lower similarity values. Thirdly, all attack scenarios reach a stationary point for the Bhattacharyya distance with a certain amount of features, whereby increasing γ does not affect the similarity. In turn, this also means that a certain amount of information to be transferred with alerts is enough, and more information does not improve the result. However, this also means that the higher the γ, the more decisive the magnitude of the stationary point of the Bhattacharyya distance is in distinguishing between any two attack scenarios, and a clear differentiation deteriorates. The reason behind this is that a mainly limited amount of features is representative for an attack scenario. Thus, if using more than these, the importance of the representative features is negatively influenced by the less important ones, meaning that feature importance of benign data takes over. Therefore, fourthly, a decent range for γ considering the feature set used in our measurements is approximately from 10 to 50, whereas if γ is set closer to 10,

it will benefit highly from similar attacks, such as a *XSS* attack with another *XSS* attack, while γ closer to 50 will benefit attack scenarios of the same category, such as web attacks in general.



(**a**)

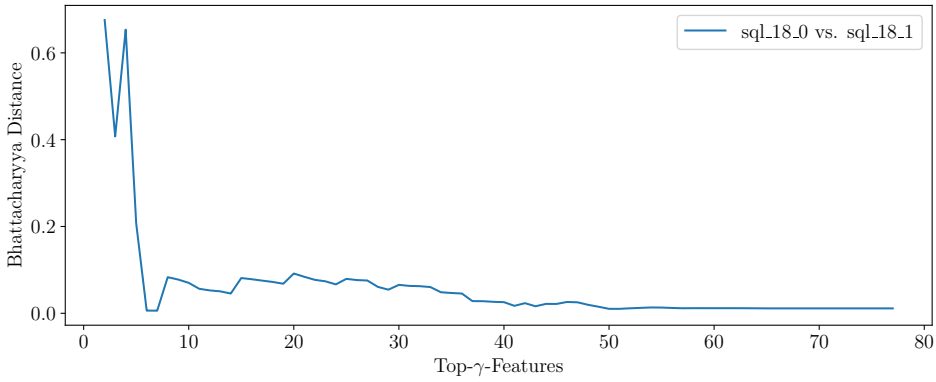
(**b**)

Figure 9. Two exemplary sig_{attr} using the same set of 78 features for the attack scenarios (**a**) *Infiltration* of CICIDS2017 and (**b**) *Brute-Force-Web-0* in CSE-CIC-IDS2018 (two colors for feature importance on all samples and only on anomalous samples—sig_{attr}).

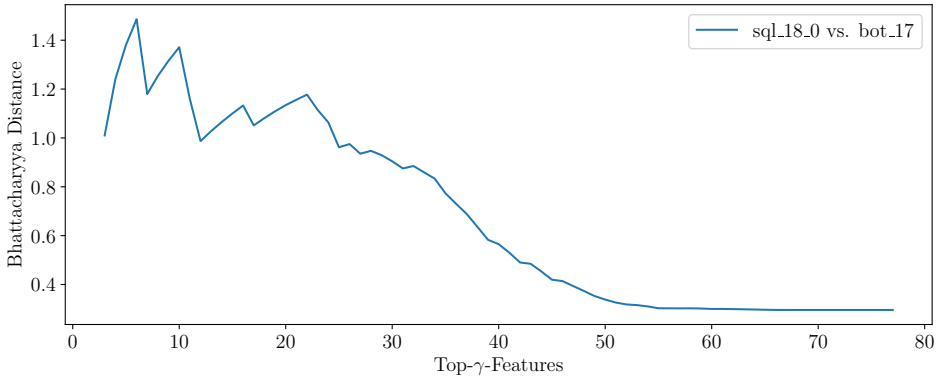
Figure 11 depicts the hierarchically clustered results of attack scenario similarity between a selection of CICIDS2017 and CSE-CIC-IDS2018 attacks based on sig_{attr} with γ set to 30. It can clearly be seen that related attack scenarios such as *xss_18_0* and *xss_18_1*, *sql_18_0* and *sql_18_1*, *brute_force_18_0* and *brute_force_18_1*, *bot_18_0* and *bot_18_1* or the Infiltration attacks from CSE-CIC-IDS2018 are highly similar.

In general, strong similarity between similar CICIDS2017 and CSE-CIC-IDS2018 attack scenarios, such as *infiltration_17* and *infiltration_18_X*, is not provided. Taking a look at the histogram comparison of two dissimilar examples from both datasets, *sql_18_0* and *sql_17* (Figure 12a) as well as *brute_force_18_1* and *brute_force_17* (Figure 12b), reveals that despite some feature similarity, both are afflicted with a high number of irrelevant features for the comparison. For instance, the two most important features of *sql_18_0*, *RST_Flag_Cnt* and *ECE_Flag_Cnt* are completely irrelevant to *sql_17* despite the rest of the features showing similarity. Equally, with respect to Figure 12b and the importance of *Init_Fwd_Win_Byts* and *Init_Bwd_Win_Byts* for web attacks [61], both attack scenarios show strong feature importance for only one of those two features. Furthermore, for each attack, some features

are more important for one attack, while they are completely irrelevant for others, such as *Active_Max/Min/Mean* for *brute_force_17*.



(**a**)



(**b**)

Figure 10. The dependency of γ on the Bhattacharyya distance (similarity) of two highly similar (**a**) and dissimilar (**b**) attack scenarios.

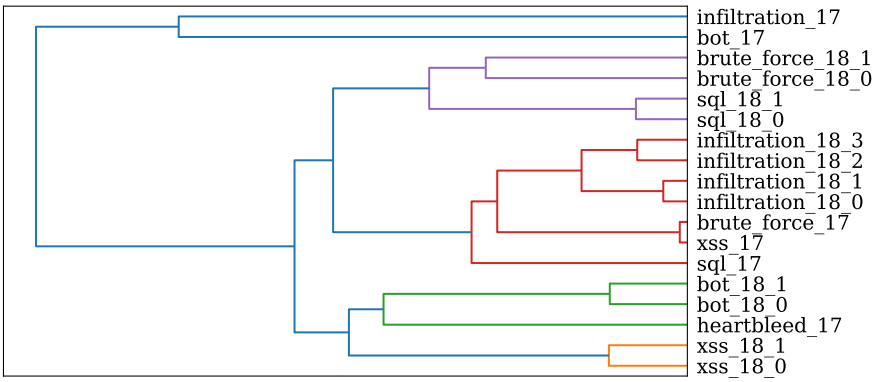
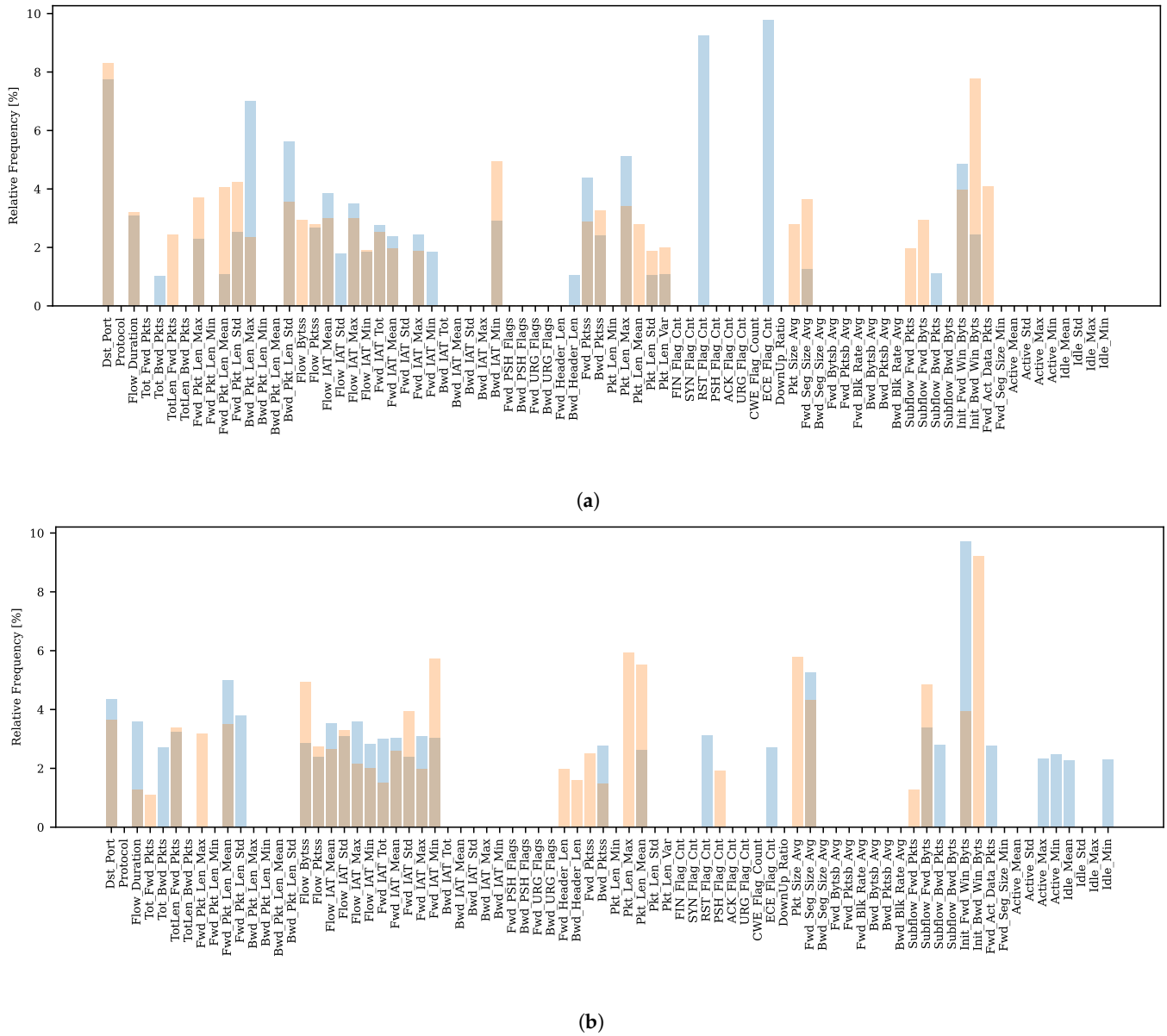


Figure 11. Hierarchical clustering of similarities between a selection of CICIDS2017 and CSE-CIC-IDS2018 attack scenarios based on sig_{attr} for $\gamma = 30$.

The infiltration-attacks between CICIDS2017 and CSE-CIC-IDS2018 can hardly be compared since the infiltration scenarios in CSE-CIC-IDS2018 include the portscanning attack steps, which are not considered within CICIDS2017. This is also confirmed by the significantly higher number of alerts associated with the 2018 infiltration scenarios reasoned by the portscan conducted after the attacker successfully gained access to the victim machine. Although similar CICIDS2017 and CSE-CIC-IDS2018 attack scenarios seem to have a weak similarity, the curvature characteristics from the γ-dependency-curves reveal better insights. Thus, with respect to Figure 10, we applied Gaussian Filtering to the

curves in order to smooth them and better visualize their curvature. In Figure 13, a sample selection of smoothed γ-dependency-curves for similar and dissimilar attack scenarios from CICIDS2017 and CSE-CIC-IDS2018 is given. Taking into account measurement deviations, curves from similar attack scenarios, *sql_18_0* and *sql_18_1*, and even *sql_18_0/1* and *sql_17*, show monotonic decreasing behavior, while the curves of dissimilar attack scenarios, such as all Sql-Injection-ones in Figure 13 with *bot_17*, contain concave sections. Although not clustered together in the dendrogram, similarity for similar 2017 and 2018 attack scenarios can be derived from the curvature characteristic. Nevertheless, there might be various reasons for the the poor result, such as the poor quality of the datasets, e.g., by a slight difference in feature generation using an older version of CICFlowMeter for CICIDS2017 or due to slightly different attacks used. Those could mimic changes in similar attack campaigns as time passes since, in real-world scenarios, adversaries change their strategy as well. However, further evaluation is necessary in future work.



(**a**)

(**b**)

Figure 12. A comparison of sig_{attr} for (**a**) SQL-Injection-0 of CSE-CIC-IDS2018 (blue) with Web Attack—Sql Injection of CICIDS2017 (orange) and (**b**) Brute-Force-Web-1 of CSE-CIC-IDS2018 (blue) with Web Attack—Brute Force of CICIDS2017 (orange) for $\gamma = 30$.

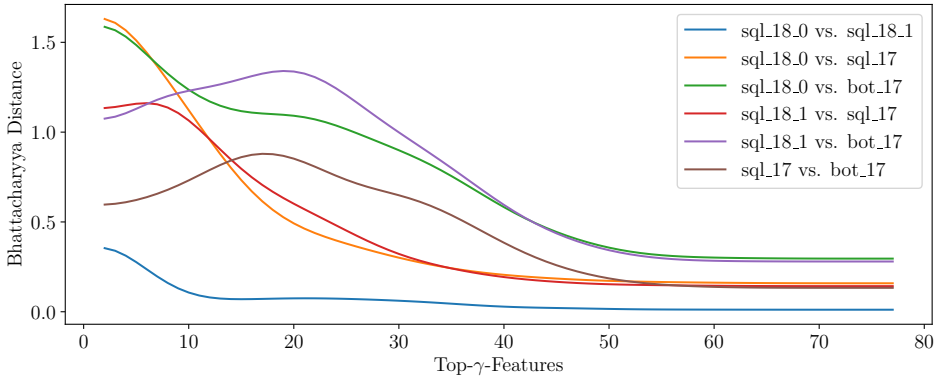


Figure 13. Smoothed γ-dependency curves using Gaussian Filtering for a selection of similar and dissimilar attack scenarios.

The results of the average processing times to generate sig_{attr} for the evaluated attack scenarios is provided in Table 7. In contrast to the exponentially increasing processing time of sig_{com}, the generation time for sig_{attr} shows linear behavior with the number of alerts inside a cluster. The mean time per alert in our evaluation is approximately 7 µs.

Table 7. The average processing time to generate sig_{attr} for a selection of attack scenarios of CICIDS2017 and CSE-CIC-IDS2018.

	brute_force_18_0	brute_force_18_1	brute_force_17	sql_18_0	sql_18_1	sql_17	xss_18_0	xss_18_1	xss_17	bot_18_0	bot_18_1	bot_17	infiltration_18_0	infiltration_18_1	infiltration_18_2	infiltration_18_3	infiltration_17	heartbleed_17
No. Alerts	250	363	1507	31	50	21	81	151	652	190,240	95,951	1966	42,760	26,111	38,752	54,311	36	308
Time (ms)	1.23	3.08	6.99	0.34	0.62	0.21	0.62	1.03	3.08	988.69	492.8	9.05	220.39	135.9	199.42	282.69	0.41	3.08

5.2.3. sig_{temp}

In order to find a histogram setting that satisfies all attack scenarios best, we computed the descriptive statistics maximum, minimum, mean and median values of Δts for each attack scenario in CICIDS2017 and CSE-CIC-IDS2018. Either because of the inaccurate timestamping in CICIDS2017, the misleading labeling, whereby multiple attack scenarios of the same category happening at different times are combined together, e.g., as for *Infiltration_17* and *PortScan_17*, we do not take into account the maximum Δt values to find a decent maximum bin parameter for the sig_{temp} histogram. Excluding the statistical strays in the maximum values for the CICIDS2017 attack scenarios, *Bot_17*, *DoS_slowloris_17*, *Infiltration_17* and *PortScan_17*, a decent maximum bin value for the histogram is 100 s (mean of residing values). Furthermore, *Heartbleed_17* is not a very representative attack scenario since it consists of 11 alarms likely reasoned in an attacker script that sends 11 Heartbeat messages at an interval of exactly 120 s to exploit the vulnerability. In a different scenario, an attacker might modify this frequency resulting in another Δt statistic. Referring to minimum, mean and median values, a decent fine-grained value for the bin width is 0.1 s (median of Δts' median).

It is again noted that in real-world applications, one might apply different histogram types, as proposed in Section 3.4.3, such as *short*, *mid* or *long*, to adjust the number and width of the bins depending on the order of magnitude of Δts. Thus, for instance, DoS-like attacks with lower mean and median Δt values than other attack types could be characterized as *short* attacks in advance with respect to the extremely low Δt values before generating signatures with significantly smaller values for the maximum bin value and bin width.

Figure 14 depicts the hierarchically clustered results of attack scenario similarity between CICIDS2017 and CSE-CIC-IDS2018 attacks based on sig_{temp}, with the mentioned

setting of maximum bin of 100 s and a bin width of 0.1 s. It can clearly be seen that similar attack scenarios, such as DoS-like attacks, infiltration, brute force or web attacks, are clustered together due to their high level of similarity. For instance, DoS-like attacks, such as *DoS_Hulk_18* and *DDOS_HOIC_18* as well as *DoS_Hulk_17* and *DDoS_17*, which are heavy DoS-attacks, are clustered together. Even the same DoS-attacks but from different datasets, *DoS GoldenEye_17* with *DoS-Goldeneye_18* and the less intensive *DoS slowloris_17* and *DoS-Slowloris_18*, show high similarity.

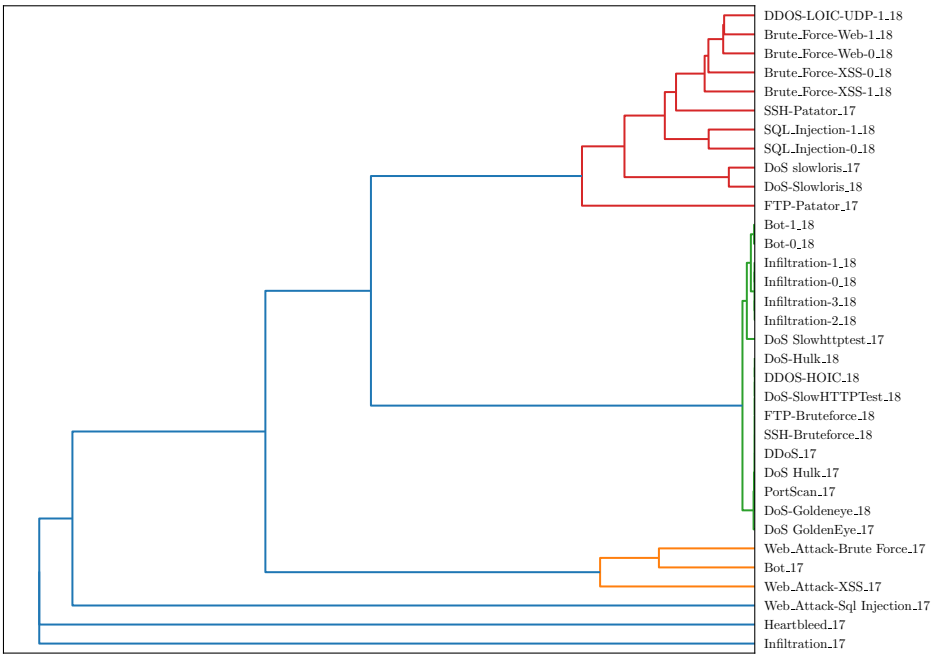


Figure 14. Hierarchical clustering of similarities between CICIDS2017 and CSE-CIC-IDS2018 attack scenarios based on sig_{temp}.

Notably, the same attack scenarios that were used multiple times in CSE-CIC-IDS2018 are, in most cases, clustered as significantly similar, such as all four Infiltration attacks, *Bot-0_18* and *Bot-1_18*, *SQL_Injection-0_18* and *SQL_Injection-1_18* or *Brute_Force-Web-0_18* and *Brute_Force-Web-1_18* as well as *Brute_Force-XSS-0_18* and *Brute_Force-XSS-1_18*.

Figure 15 demonstrates the high level of similarity leveraging sig_{temp} between the four timely different and unrelated infiltration attack scenarios taken from CSE-CIC-IDS2018. For the sake of visualization and with respect to the descriptive statistics maximum, minimum, mean and median for those attack scenarios, we set the histogram values to a maximum bin value of 1 s and the bin width to 0.01 s.

Although not clustered in Figure 14, similarity can be seen between comparable attack scenarios taken from different datasets of CICIDS2017 and CSE-CIC-IDS2018 when adapting the histogram settings. This is shown with the *XSS* and *Brute Force* web attacks in Figure 16. The histogram is set to a maximum bin value of 30 s and a bin width of 1 s. However, the less fine-grained binning of the CICIDS2017 attacks can clearly be seen, which is reasoned in the way timestamps were generated, as explained in Section 4.2.

In contrast to the average processing times for sig_{attr}, the processing of sig_{temp} is split into two parts—the generation of Δts based on timestamps of the timely sorted alerts of a cluster and the generation of the histogram, the actual sig_{temp}. The result of the average processing times for the attack scenarios of CICIDS2017 is provided with the same histogram setting of a maximum bin value of 100 s and bin width with 0.1 s in Table 8 and for CSE-CIC-IDS2018 in Table 9. As with sig_{attr}, the processing times of sig_{temp} shows the linear behavior with a mean time per alert for Δts generation of approximately 200 µs and for histogram generation of approximately 4.5 µs per alert. The latter depends on the

histogram setting and increases to a value of approximately 138 µs per alert for a maximum bin value of 550 s and bin width with 0.01 s, for instance.

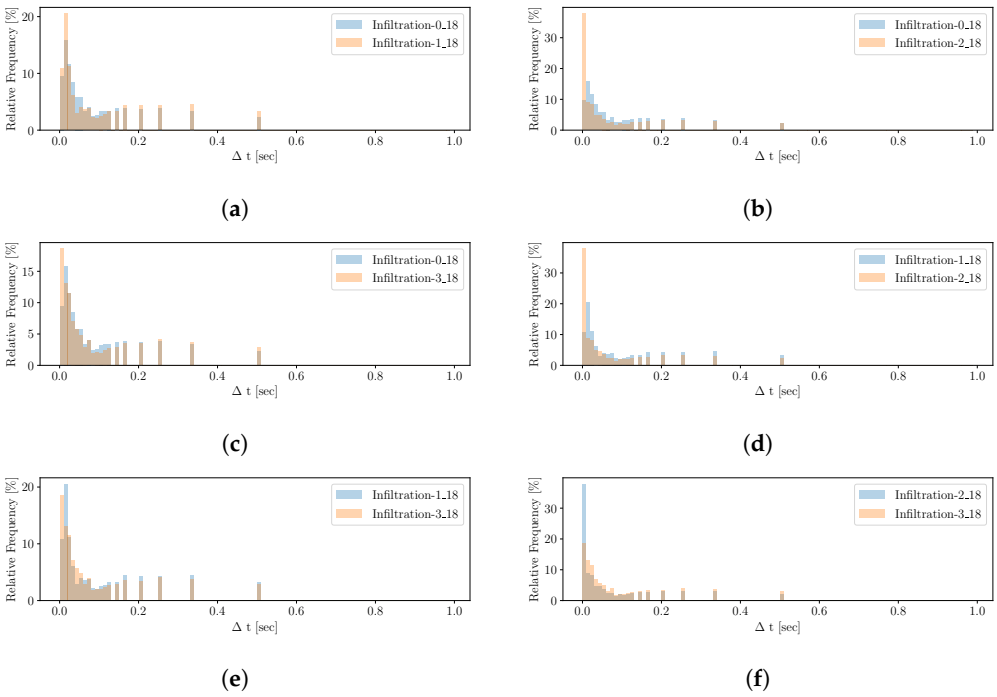


Figure 15. A comparison of Δ*t* histograms of four distinct *Infiltration* attack scenarios (**a**–**f**) present in CSE-CIC-IDS2018.

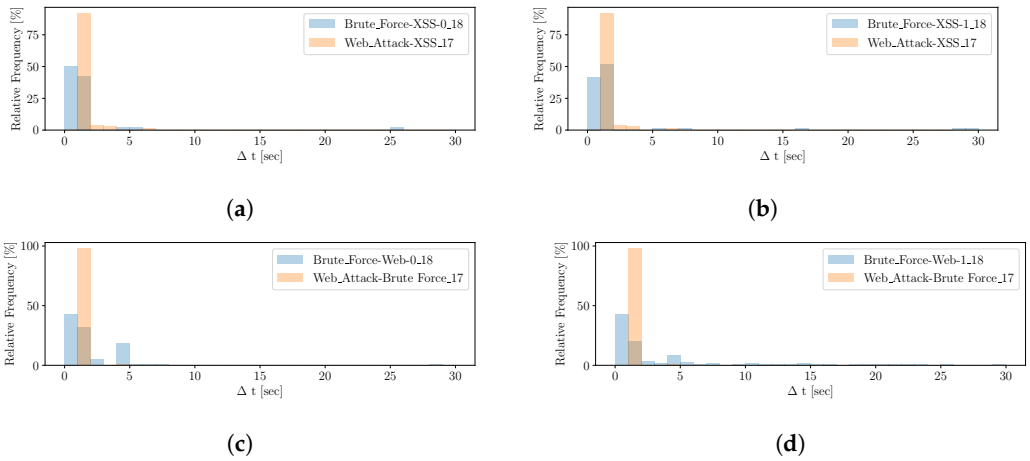


Figure 16. A comparison of Δ*t* histograms of four distinct web attack scenarios—*XSS* (**a**,**b**) and *Brute Force* (**c**,**d**)—present in CICIDS2017 and CSE-CIC-IDS2018.

Table 8. The average processing time to generate sig_{temp} for a selection of attack scenarios of CICIDS2017.

	brute_force_17	xss_17	sql_17	infiltration_17	ddos_17	dos_hulk_17	dos_slowloris_17	dos_slowhttptest_17	dos_goldeneye_17	port_scan_17	ftp-patator_17	ssh-patator_17	bot_17	heartbleed_17
No. Alerts	1507	652	21	36	128,027	231,073	5796	5499	10,293	158,930	7938	5897	1966	11
Generating Δ*ts* (s)	0.1873	0.0810	0.0030	0.0053	15.6347	28.8301	0.7102	0.6742	1.2875	19.6687	0.9818	0.7314	2.4444	0.0018
Generating Histograms (ms)	0.69	0.58	0.51	0.54	13.78	24.49	1.14	1.10	1.57	17.06	1.38	1.18	0.76	0.57

Table 9. Average processing time to generate sig_{temp} for a selection of attack scenarios of CSE-CIC-IDS2018.

	brute_force_18_0	brute_force_18_1	sql_18_0	sql_18_1	xss_18_0	xss_18_1	bot_18_0	bot_18_1	infiltration_18_0	infiltration_18_1	infiltration_18_2	infiltration_18_3	ddos-hoic_18	dos-hulk_18	dos-slowhttptest_18	dos-slowloris_18	ddos-loic-udp_18	dos-goldeneye_18	ftp-bruteforce_18	ssh-bruteforce_18
No. Alerts	363	250	50	31	81	151	190,240	95,951	42,760	26,111	38,752	54,311	686,012	461,912	139,890	10,990	1730	41,508	193,360	187,589
Generating Δts (s)	0.0484	0.0701	0.0088	0.0142	0.0214	0.0400	49.1716	24.7964	10.4209	6.3606	9.4362	13.2450	170.4973	104.9426	23.1637	1.8217	0.3151	6.5640	29.9785	38.7183
Generating Histograms (ms)	0.58	0.61	0.56	0.56	0.57	0.59	21.56	11.06	5.52	3.54	4.91	6.77	75.33	51.48	15.60	1.14	0.76	5.27	21.35	20.94

6. Conclusions and Future Work

With the advent of anomaly-based IDS to detect malicious activity in streaming network data, especially online-capable unsupervised Outlier Detection (OD) algorithms, novel techniques for alert correlation methods are increasingly needed. Those must be capable to mine information from the outcome of OD algorithms without knowledge information, such as the intrusion type, which is typically provided by misuse-based IDS. Alerts are continuously generated from the high-volume, high-speed and high-dimensional streaming data in the form of an alert stream, which might be afflicted to a high amount of False Positives (FPs) and False Negatives (FNs). Human experts can no longer be expected to handle this massive amount of alerts, and certain types of attacks are likely to be overlooked.

Thus, this article introduces and discusses a novel framework called SOAAPR, which is able to deal with the outcomes from online OD algorithms in multiple configuration settings to improve the input quality for the streaming alert correlation/clustering module in terms of reducing FPs and mitigating FNs. For this, alerts are equipped, apart from the typical intrinsic attributes, such as IP, port or timestamp information, with feature importance scores and the respective outlier scores. The core component of SOAAPR clusters the streaming alerts according to their attributes' similarity. The resulting clusters can evolve over time and, if not discarded in the case of irrelevant clusters, can be saturated, meaning that these clusters are potentially capturing attack scenarios. In order to ensure a short response time for security analysts, the alarms of those clusters are promptly fed into a consecutive module that generates three types of signatures, denoted as sig_{com}, sig_{attr} and sig_{temp}. These fingerprint-like characteristics represent the attack scenarios in terms of the attack's communication behavior, their cause in the data's features and their temporal sequence of associated alerts. The signatures can then be used to find similarities between attack scenarios or seek for similar signatures that can be collected in a knowledge base or, e.g., shared with other companies or institutes.

The evaluation leveraging the widely-known CICIDS2017 and CSE-CIC-IDS2018 datasets is split into two parts. First, the streaming clustering module of SOAAPR is compared against a graph-based competitor, the alert clustering component of GAC [40]. We rely on four different metrics, the completeness, the soundness, the Jaccard index and the elapsed time for alert processing the datasets, as representatives for the computational performance. Since GAC's complexity increases with the graph size, chunk processing with 5000 alerts had to be applied. SOAAPR is configured with hyperparameters that are, for the sake of equal comparison, similar to GAC and are even set to capture attack scenarios with a low number of associated alerts. Second, the signaturing functionality of SOAAPR, generating and comparing sig_{com}, sig_{attr} and sig_{temp}, is the subject of our experimental investigations. Thus, we investigate the similarity between attack scenarios' signatures and their average computing time in our experiments.

The discussion of results for alert clustering reveals that SOAAPR reliably clusters attack scenarios as good as GAC while being significantly more efficient in terms of processing time. In the best case, SOAAPR clusters DoS-attack scenarios faster by a factor of 190 compared to GAC. Although it may be that attack scenarios are split into a couple of clusters, even in the worst case for the web attacks, SOAAPR reduces the associated 2180 alerts into only 24 clusters, which is a compression factor of magnitude 91. Adjusting

SOAAPR's hyperparameters even aids to notably reduce so-called ghost-clusters, which are mainly caused by FPs. Given the multitude of different attacks and their characteristics, we propose to leverage multiple SOAAPR instances each parameterized with hyperparameters specifically for each attack type for real-world application. A more holistic evaluation of the effects of FPs and FNs on the clustering result is topic for further work.

The discussion of results with respect to SOAAPR's signaturing reveals that all three signature types generally can be used to characterize attacks and find similarities between attack categories. For sig_{com}, an attack scenario similarity of up to 95.05% could be obtained. However, its processing time shows exponential behavior over the number of alerts. In order to compute and compare sig_{attr}, the supervised Random Forests classifier has been utilized to generate feature importance scores in the operation modes *single system—multiple algorithms*, in which multiple classifiers work in parallel to improve alert quality. While yielding strong similarity between comparable attack scenarios of the same dataset, similarity could even be shown with sig_{attr} between similar scenarios of different datasets by investigating the curves' curvature depending on the number of top-performing features. The results comparing the timing behavior of a total number of 34 attack scenarios, captured with sig_{temp}, yields a strong similarity especially between similar attack scenarios from the same datasets. In contrast to sig_{com}, the processing time of sig_{attr} and sig_{temp} is significantly faster and shows linear behavior. Overall, some congruent attack scenarios from different datasets showed weak similarity for sig_{temp}, which is mainly caused by the poor quality of the available datasets.

Thus, a topic for future work is to evaluate SOAAPR on other datasets that provide, for instance, better timestamping for sig_{temp} evaluation or contains IP and port information for sig_{com} evaluation. With respect to sig_{attr}, as of now, we relied on the supervised RF—SHAP—feature importance scoring functionality for better result interpretability. However, in future evaluation, we want to replace RF with an online unsupervised OD algorithm equipped with a feature importance scoring functionality, such as Loda [23] or PCB-iForest [13]. As we showed that the ambitious aim to exploit the outcome of OD algorithms in order to generate an attack pattern generally works, we would also want to investigate the impacts of introducing FPs and FNs on signature comparison.

Since SOAAPR's streaming clustering is sensitive to the timing behavior of an attack and its associated alerts, attack scenarios might be split into multiple tiny clusters if SOAAPR's hyperparameters are not properly set. Thus, further research will also focus on investigating to combine split clusters together if their overall similarity is high, e.g., by measuring the cluster centroids distances or leveraging the comparison of signatures sig_{com}, sig_{attr} and sig_{temp}. Split clusters with highly similar signatures might likely be assigned to the same attack scenario. Thus, not only the clustering result might be improved but also multi-stage attack detection might be enabled for which SOAAPR initially was not designed, similar to the Intrusion Session Rebuilding component of IACF [48] or the Attack Interconnection phase of GAC.

Author Contributions: Conceptualization, M.H.; Methodology, M.H. and E.W.; Software, M.H., A.U.; Validation, M.H. and D.F.; Writing—Original Draft, M.H.; Visualization, M.H., E.W. and A.U.; Writing—Review and Editing, M.H., E.W., D.F. and M.S.; Supervision, D.F. and M.S.; Funding acquisition, D.F. and M.S.; Project administration, M.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research work is partially supported by the research project 13FH645IB6 of the German Federal Ministry of Education and Research as well as by the Ministry of Education, Youth and Sports of the Czech Republic under grant No. LO1506.

Data Availability Statement: The 'CICIDS2017' and 'CSE-CIC-IDS2018' datasets utilized for the evaluation in this article are available in the Canadian Institute for Cybersecurity (CIC) repository http://www.unb.ca/cic/datasets/ids-2017.html and via AWS resources https://www.unb.ca/cic/datasets/ids-2018.html (accessed on: 25 May 2021).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mironeanu, C.; Archip, A.; Amarandei, C.-M.; Craus, M. Experimental Cyber Attack Detection Framework. *Electronics* **2021**, *10*, 1682. [CrossRef]
2. Liu, H.; Lang, B. Machine learning and deep learning methods for intrusion detection systems: A survey. *Appl. Sci.* **2019**, *9*, 4396. [CrossRef]
3. Ramaki, A.A.; Rasoolzadegan, A.; Bafghi, A.G. A systematic mapping study on intrusion alert analysis in intrusion detection systems. *ACM Comput. Surv.* **2018**, *51*, 1–41. [CrossRef]
4. Nespoli, P.; Papamartzivanos, D.; Gomez Marmol, F.; Kambourakis, G. Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks. *IEEE Commun. Surv. Tutor.* **2018**, *20*, 1361–1396. [CrossRef]
5. Zhang, K.; Zhao, F.; Luo, S.; Xin, Y.; Zhu, H.; Chen, Y. Online intrusion scenario discovery and prediction based on Hierarchical Temporal Memory (HTM). *Appl. Sci.* **2020**, *10*, 2596. [CrossRef]
6. Ma, J.; Li, Z.-T.; Li, W.-M. Real-time alert stream clustering and correlation for discovering attack strategies. In Proceedings of the 2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery, Jinan, China, 18–20 October 2008; Volume 4, pp. 379–384. [CrossRef]
7. Togbe, M.U.; Barry, M.; Boly, A.; Chabchoub, Y.; Chiky, R.; Montiel, J.; Tran, V.-T. Anomaly detection for data streams based on isolation forest using scikit-multiflow. In *Computational Science and Its Applications—ICCSA 2020*; Springer International Publishing: Cham, Switzerland, 2020; pp. 15–30. [CrossRef]
8. Kovačević, I.; Groš, S.; Slovenec, K. Systematic review and quantitative comparison of cyberattack scenario detection and projection. *Electronics* **2020**, *9*, 1722. [CrossRef]
9. Haas, S.; Wilkens, F.; Fischer, M. Efficient attack correlation and identification of attack scenarios based on network-motifs. In Proceedings of the 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC), London, UK, 29–31 October 2019; pp. 1–11. [CrossRef]
10. Sundaramurthy, S.C.; Zomlot, L.; Ou, X. Practical IDS Alert Correlation in the Face of Dynamic Threats. In Proceedings of the 11th International Conference on Security and Management (SAM), Las Vegas, NV, USA, 20 July 2011.
11. Pang, G.; Cao, L.; Chen, L.; Liu, H. Unsupervised feature selection for outlier detection by modelling hierarchical value-feature couplings. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM), Barcelona, Spain, 12–15 December 2016; pp. 410–419. [CrossRef]
12. Wang, H.; Bah, M.J.; Hammad, M. Progress in outlier detection techniques: A survey. *IEEE Access* **2019**, *7*, 107964–108000. [CrossRef]
13. Heigl, M.; Anand, K.A.; Urmann, A.; Fiala, D.; Schramm, M.; Hable, R. On the improvement of the isolation forest algorithm for outlier detection with streaming data. *Electronics* **2021**, *10*, 1534. [CrossRef]
14. Muallem, A.; Shetty, S.; Pan, J.W.; Zhao, J.; Biswal, B. Hoeffding tree algorithms for anomaly detection in streaming datasets: A survey. *J. Inf. Secur.* **2017**, *8*, 339–361. [CrossRef]
15. Saffari, A.; Leistner, C.; Santner, J.; Godec, M.; Bischof, H. On-line random forests. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision Workshops, Kyoto, Japan, 27 September–4 October 2009; pp. 1393–1400. [CrossRef]
16. Liu, L.; Hu, M.; Kang, C.; Li, X. Unsupervised anomaly detection for network data streams in industrial control systems. *Information* **2020**, *11*, 105. [CrossRef]
17. Yao, H.; Fu, X.; Yang, Y.; Postolache, O. An incremental local outlier detection method in the data stream. *Appl. Sci.* **2018**, *8*, 1248. [CrossRef]
18. Mirsky, Y.; Doitshman, T.; Elovici, Y.; Shabtai, A. Kitsune: An ensemble of autoencoders for online network intrusion detection. In Proceedings of the Network and Distributed System Security Symposium 2018 (NDSS'18), San Diego, CA, USA, 18–21 February 2018.
19. Yu, K.; Shi, W.; Santoro, N. Designing a streaming algorithm for outlier detection in data mining—An incremental approach. *Sensors* **2020**, *20*, 1261. [CrossRef]
20. Liu, F.T.; Ting, K.M.; Zhou, Z.-H. Isolation Forest. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 413–422. [CrossRef]
21. Tan, S.C.; Ting, K.M.; Liu, T.F. Fast anomaly detection for streaming data. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence—Volume Two*; AAAI Press: New York, NY, USA, 2011; pp. 1511–1516. [CrossRef]
22. Sathe, S.; Aggarwal, C.C. Subspace outlier detection in linear time with randomized hashing. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM), Barcelona, Spain, 12–15 December 2016; pp. 459–468. [CrossRef]
23. Pevný, T. Loda: Lightweight on-line detector of anomalies. *Mach. Learn.* **2016**, *102*, 275–304. [CrossRef]
24. Manzoor, E.; Lamba, H.; Akoglu, L. xStream: Outlier detection in feature-evolving data streams. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; ACM: New York, NY, USA, 2018. [CrossRef]
25. Aggarwal, C.C.; Sathe, S. Theoretical foundations and algorithms for outlier ensembles. *SIGKDD Explor.* **2015**, *17*, 24–47. [CrossRef]
26. Ding, Z.; Fei, M. An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window. *IFAC Proc.* **2013**, *46*,12–17. [CrossRef]

27. Togbe, M.U.; Chabchoub, Y.; Boly, A.; Barry, M.; Chiky, R.; Bahri, M. Anomalies detection using isolation in concept-drifting data streams. *Computers* **2021**, *10*, 13. [CrossRef]
28. Sun, H.; He, Q.; Liao, K.; Sellis, T.; Guo, L.; Zhang, X.; Shen, J.; Chen, F. Fast anomaly detection in multiple multi-dimensional data streams. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 1218–1223. [CrossRef]
29. Ma, H.; Ghojogh, B.; Samad, M.N.; Zheng, D.; Crowley, M. Isolation Mondrian forest for batch and online anomaly detection. In Proceedings of the 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 11–14 October 2020; pp. 3051–3058. [CrossRef]
30. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
31. Lundberg, S.M.; Lee, S.-I. A unified approach to interpreting model predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 4768–4777. [CrossRef]
32. Salah, S.; Maciá-Fernández, G.; Díaz-Verdejo, J.E. A model-based survey of alert correlation techniques. *Comput. Netw.* **2013**, *57*, 1289–1317. [CrossRef]
33. Hubballi, N.; Suryanarayanan, V. False alarm minimization techniques in signature-based intrusion detection systems: A survey. *Comput. Commun.* **2014**, *49*, 1–17. [CrossRef]
34. Bhuyan, M.H.; Bhattacharyya, D.K.; Kalita, J.K. *Network Traffic Anomaly Detection and Prevention: Concepts, Techniques, and Tools*, 1st ed.; Springer International Publishing: Cham, Switzerland, 2017; ISBN 9783319651866.
35. Mirheidari, S.A.; Arshad, S.; Jalili, R. Alert correlation algorithms: A survey and taxonomy. In *Cyberspace Safety and Security*; Springer International Publishing: Cham, Switzerland, 2013; pp. 183–197. [CrossRef]
36. Bolzoni, D.; Etalle, S.; Hartel, P.H. Panacea: Automating attack classification for anomaly-based network intrusion detection systems. In *Recent Advances in Intrusion Detection*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; pp. 1–20. [CrossRef]
37. Qassim, Q.S.; Zin, A.M.; Aziz, M.J.A. Anomaly-based network IDS false alarm filter using cluster-based alarm classification approach. *Int. J. Secur. Netw.* **2017**, *12*, 13. [CrossRef]
38. Shin, J.; Choi, S.-H.; Liu, P.; Choi, Y.-H. Unsupervised multi-stage attack detection framework without details on single-stage attacks. *Future Gener. Comput. Syst.* **2019**, *100*, 811–825. [CrossRef]
39. Sudheera, K.L.K.; Divakaran, D.M.; Singh, R.P.; Gurusamy, M. ADEPT: Detection and identification of correlated attack stages in IoT networks. *IEEE Internet Things J.* **2021**, *8*, 6591–6607. [CrossRef]
40. Haas, S.; Fischer, M. On the alert correlation process for the detection of multi-step attacks and a graph-based realization. *ACM SIGAPP Appl. Comput. Rev.* **2019**, *19*, 5–19. [CrossRef]
41. Milo, R.; Shen-Orr, S.; Itzkovitz, S.; Kashtan, N.; Chklovskii, D.; Alon, U. Network motifs: Simple building blocks of complex networks. *Science* **2002**, *298*, 824–827. [CrossRef]
42. Wang, L.; Liu, A.; Jajodia, S. Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. *Comput. Commun.* **2006**, *29*, 2917–2933. [CrossRef]
43. Aggarwal, C.C.; Yu, P.S.; Han, J.; Wang, J. A framework for clustering evolving data streams. In Proceedings of the VLDB '03: Proceedings of the 29th international conference on Very large data bases—Volume 29, Berlin, Germany, 9–12 September 2003; pp. 81–92. [CrossRef]
44. Sadoddin, R.; Ghorbani, A.A. Real-time alert correlation using stream data mining techniques. In Proceedings of the 20th National Conference on Innovative Applications of Artificial Intelligence, Chicago, IL, USA, 13–17 July 2008; Volume 3, pp. 1731–1737. [CrossRef]
45. Ren, H.; Stakhanova, N.; Ghorbani, A.A. An online adaptive approach to alert correlation. In *Detection of Intrusions and Malware, and Vulnerability Assessment*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 153–172. [CrossRef]
46. Ramaki, A.A.; Amini, M.; Ebrahimi Atani, R. RTECA: Real time episode correlation algorithm for multi-step attack scenarios detection. *Comput. Secur.* **2015**, *49*, 206–219. [CrossRef]
47. Faraji Daneshgar, F.; Abbaspour, M. Extracting fuzzy attack patterns using an online fuzzy adaptive alert correlation framework: Fuzzy adaptive alert correlation. *Secur. Commun. Netw.* **2016**, *9*, 2245–2260. [CrossRef]
48. Zhang, K.; Zhao, F.; Luo, S.; Xin, Y.; Zhu, H. An intrusion action-based IDS alert correlation analysis and prediction framework. *IEEE Access* **2019**, *7*, 150540–150551. [CrossRef]
49. Ossenbuhl, S.; Steinberger, J.; Baier, H. Towards automated incident handling: How to select an appropriate response against a network-based attack? In Proceedings of the 2015 Ninth International Conference on IT Security Incident Management & IT Forensics, Magdeburg, Germany, 18–20 May 2015; pp. 51–67. [CrossRef]
50. Zohrevand, Z.; Glässer, U. Should I Raise The Red Flag? A comprehensive survey of anomaly scoring methods toward mitigating false alarms. *arXiv* **2019**, arXiv:1904.06646.
51. Ourston, D.; Matzner, S.; Stump, W.; Hopkins, B. Applications of hidden Markov models to detecting multi-stage network attacks. In Proceedings of the 36th Annual Hawaii International Conference on System Sciences, Big Island, HI, USA, 6–9 January 2003; Volume 10. [CrossRef]
52. Kriegel, H.-P.; Kroger, P.; Schubert, E.; Zimek, A. Interpreting and Unifying Outlier Scores. In Proceedings of the 2011 SIAM International Conference on Data Mining, Mesa, AZ, USA, 28–30 April 2011; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2011. [CrossRef]

53. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; pp. 1–6. [CrossRef]
54. Zhang, H.; Jin, X.; Li, Y.; Jiang, Z.; Liang, Y.; Jin, Z.; Wen, Q. A multi-step attack detection model based on alerts of smart grid monitoring system. *IEEE Access* **2020**, *8*, 1031–1047. [CrossRef]
55. Xylogiannopoulos, K.; Karampelas, P.; Alhajj, R. Early DDoS detection based on data mining techniques. In *Information Security Theory and Practice—Securing the Internet of Things*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 190–199. [CrossRef]
56. Prakash, A.; Satish, M.; Bhargav, T.S.S.; Bhalaji, N. Detection and mitigation of denial of service attacks using stratified architecture. *Procedia Comput. Sci.* **2016**, *87*, 275–280. [CrossRef]
57. Galeano-Brajones, J.; Carmona-Murillo, J.; Valenzuela-Valdés, J.F.; Luna-Valero, F. Detection and mitigation of DoS and DDoS attacks in IoT-based stateful SDN: An experimental approach. *Sensors* **2020**, *20*, 816. [CrossRef]
58. Welford, B.P. Note on a method for calculating corrected sums of squares and products. *Technometrics* **1962**, *4*, 419–420. [CrossRef]
59. Heigl, M.; Doerr, L.; Tiefnig, N.; Fiala, D.; Schramm, M. A resource-preserving self-regulating Uncoupled MAC algorithm to be applied in incident detection. *Comput. Secur.* **2019**, *85*, 270–287. [CrossRef]
60. Zubaroğlu, A.; Atalay, V. Data stream clustering: A review. *Artif. Intell. Rev.* **2021**, *54*, 1201–1236. [CrossRef]
61. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy, Austin, TX, USA, 12–15 December 2021; pp. 108–116. [CrossRef]
62. Kurniabudi; Stiawan, D.; Darmawijoyo; Bin Idris, M.Y.; Bamhdi, A.M.; Budiarto, R. CICIDS-2017 dataset feature analysis with information gain for anomaly detection. *IEEE Access* **2020**, *8*, 132911–132921. [CrossRef]
63. Jeng, M. Error in statistical tests of error in statistical tests. *BMC Med. Res. Methodol.* **2006**, *6*, 45. [CrossRef] [PubMed]
64. Bhattacharyya, A. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.* **1943**, *35*, 99–109.
65. Gunter Ollmann. Why STIX/TAXII/CybOX Sharing Is Incompatible with AI Threat Detection Systems. Available online: https://www.linkedin.com/pulse/why-stixtaxiicybox-sharing-incompatible-ai-threat-systems-ollmann (accessed on 19 July 2021).
66. Sadique, F.; Cheung, S.; Vakilinia, I.; Badsha, S.; Sengupta, S. Automated structured threat information expression (STIX) document generation with privacy preservation. In Proceedings of the 2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA, 8–10 November 2018; pp. 847–853. [CrossRef]
67. Wang, Q.; Jiang, J.; Shi, Z.; Wang, W.; Lv, B.; Qi, B.; Yin, Q. A novel multi-source fusion model for known and unknown attack scenarios. In Proceedings of the 2018 17th IEEE International Conference On Trust, Security And Privacy in Computing and Communications/12th IEEE International Conference on Big Data Science and Engineering (TrustCom/BigDataSE), New York, NY, USA, 1–3 August 2018; pp. 727–736. [CrossRef]
68. Wang, X.; Yu, L.; He, H.; Gong, X. MAAC: Novel alert correlation method to detect multi-step attack. *arXiv* **2020**, arXiv:2011.07793.
69. Kenyon, A.; Deka, L.; Elizondo, D. Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets. *Comput. Secur.* **2020**, *99*, 102022. [CrossRef]
70. Ahmad, Z.; Shahid Khan, A.; Wai Shiang, C.; Abdullah, J.; Ahmad, F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*. [CrossRef]
71. Ning, P.; Cui, Y.; Reeves, D.S.; Xu, D. Techniques and tools for analyzing intrusion alerts. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **2004**, *7*, 274–318. [CrossRef]