The ISC Int'l Journal of Information Security

ISeCure

July 2023, Volume 15, Number 2 (pp. 178–215)

http://www.isecure-journal.org

Towards Event Aggregation for Reducing the Volume of Logged Events During IKC Stages of APT Attacks

Ali Ahmadian Ramaki [1,2], Abbas Ghaemi-Bafghi [1,*], and Abbas Rasoolzadegan [2]

[1] *Data and Communication Security Laboratory, Ferdowsi University of Mashhad, Mashhad, Iran*

[2] *Software Quality Laboratory, Ferdowsi University of Mashhad, Mashhad, Iran*

ARTICLE INFO.

Article history:

Received: December 14, 2021

Revised: December 20, 2021

Accepted: June 19, 2023

Published Online: July 1, 2023

Keywords:

Advanced Persistent Threat, Event Aggregation, Heterogeneous Event Logs, Intrusion Kill Chain, Security Event Management

Type: Research Article

doi: 10.22042/isecure.2023.319798.730

dor: 20.1001.1.20082045.2023.15.2.5.4

ABSTRACT

Targeted attacks like Advanced Persistent Threats (APTs) have become a primary concern of many enterprise networks. As a common approach to counter these attacks, security staff deploys various security and non-security sensors at different lines of defense (Network, Host, and Application) to track the attacker's behaviors during their kill chain. However, one of the drawbacks of this approach is the massive amount of events raised by heterogeneous security and non-security sensors. The bulk logged events makes it challenging to analyze them for later processing, i.e., event correlation for timely detection of APT attacks. Some research papers have been published on event aggregation to reduce the volume of logged low-level events. However, most research works have provided a method to aggregate the events of a single-type and homogeneous event source, i.e., NIDS. In addition, their main focus is only on the degree to which the event volume is reduced. At the same time, the amount of security information lost during the event aggregation process is also significant. In this paper, we propose a three-phase event aggregation method to reduce the volume of logged heterogeneous events during APT attacks, considering the lowest rate of loss of security information. To this aim, the sensors' low-level events are first clustered into similar event groups. Then, after filtering noisy event clusters, the remaining clusters are summarized based on an Attribute-Oriented Induction (AOI) method in a controllable manner to reduce the unimportant or duplicated events. The method has been evaluated on the three publicly available datasets: SotM34, Bryant, and LANL. The experimental results show that the method is efficient enough in event aggregation and can reduce event volume up to 99.7% with an acceptable information loss ratio (ILR) level.

© 2023 ISC. All rights reserved.

1 Introduction

Today, the variety of services organizations can provide in computer systems, and networks have become necessary and inevitable. Providing many and varied services will make the networks bigger and more sophisticated the technologies used, i.e., cloud com-

* Corresponding author.

**This article is an extended/revised version of an ISCISC'18 paper.

Email addresses: ali.ahmadianramaki@mail.um.ac.ir, ghaemib@um.ac.ir, rasoolzadegan@um.ac.ir

ISSN: 2008-2045 © 2023 ISC. All rights reserved.

puting, the internet of things (IoT), and blockchain. One of the main concerns of network growth is the occurrence of complex security-related incidents due to increasing network size, poor network management, and less hardening of networks against cybersecurity attacks [1, 2]. One of the main challenges in large networks is combating a modern type of attack, which is getting more and more complex and targeted, i.e., advanced persistent threats (APTs) [3, 4]. In this type of attack, the attacker, based on a variety of attack tools and techniques, goes through a next-generation threat life cycle to achieve its target, which is called intrusion kill chain (IKC) or cyber kill chain (CKC)[3, 5–8]. In other words, based on a specified IKC model, the attackers use sequencing of attack stages involving various attack steps to infiltrate the target network and obtain their final goals by moving laterally across multiple network hosts for exfiltration of sensitive data or sabotage.

Generally, in an IKC model, the malicious intruders use combined attacks that exploit different vulnerabilities of the targeted network during the attack scenario. In other words, one of the main characteristics of an IKC-based attack is multi-level intrusion which refers to the combined nature of these types of attacks that uses all the Network, Host, and Application levels security breaches to perform the attack [7, 9, 10]. Hence, finding the root cause of security incidents will be challenging for network security administrators. Although the attackers use advanced attack vectors at the side of the attack surface, some promising approaches are proposed by the security research communities to combat and minimize the attacker's behaviors.

Due to both the communal nature of the complex attacks and the lack of any well-known attack signatures for the modern IKC-based cybersecurity attacks, traditional network-level security solutions like Firewalls and Intrusion detection systems (IDSs) are not enough to prevent APT attack strategies and block them, solely [7, 9–11]. One of the main promising approaches for tracking the attacker's behaviors and detecting the IKC is the use of multiple and various heterogeneous security and non-security sensors in different lines of defense of a monitored network (Network, Host, and Application) to enhance the security level and mitigate the potential risks caused by security breaches [3, 9–11]. Security sensors generate security events with high risk, and non-security sensors generate ordinary events of the system with low risk.

In an organization with heterogeneous sensors or even total facilities like a security operation center (SOC) and security information and event management (SIEM), each sensor in a defense line tends to operate differently on each class of attack regarding intrusion activities of a unique stage of IKC [10]. Therefore, they record different intrusion reports and log messages in various formats. Although heterogeneous sensors play a vital role in enhancing the security of computer networks, their use of them in organizations will bring some main challenges which have been reported in the previous works [11–13]. One of the main drawbacks of using heterogeneous sensors is the massive amount of security and non-security events/alerts raised by heterogeneous sensors, which most of them, about 99%, are duplicated and have the exact root cause or irrelevant and false-positive [12, 13]. So, this makes it challenging to analyze logged events for later processing, i.e., correlation analysis for complex attack scenario detection like APT attacks, which is performed using an alert/event correlation system of total security solutions like SIEM.

Till now, various frameworks for the alert/event correlation system have been proposed by researchers [12], and [14] which include various functional components. Figure 1 depicts the architecture of the proposed alert/event correlation process model by Salah *et al.* [12], which consists of four modules and eight components. According to this figure, the components of the alert preprocessing module have the task of normalizing events to convert the raw events logged by sensors from different vendors into a unified format and tokenizing event features for further processing. The components of the alert reduction module are responsible for validating events to filter false-positive ones by incorporating some contextual information about the target network. The components of the alert correlation module try to merge multiple alerts/events which share the exact root cause for event reduction purposes and then correlate the aggregated events for constructing multi-stage attack scenarios. The output of this module may be referred to the previous module for further investigation. Finally, the components of the alert prioritization module intend to analyze the risk score of constructed attack scenarios for reporting them to the security staff based on their severity.

Generally, in an event correlation system, the event flooding problem and false-positive events detection is solved by an event aggregation/fusion component [12, 14, 15]. This component is desirable by fusing similar events to an aggregated event (meta-event) and also eliminating the redundant events [12, 13]. The main idea of the event aggregation component is to reduce the volume of logged events by various sensors during IKC of APT attacks to better manage them for later usage, like event correlation analysis that aims to correlate event logs for identifying multi-stage attack scenarios. To the best of our knowledge, there is no suitable IKC-based aggregation method in the

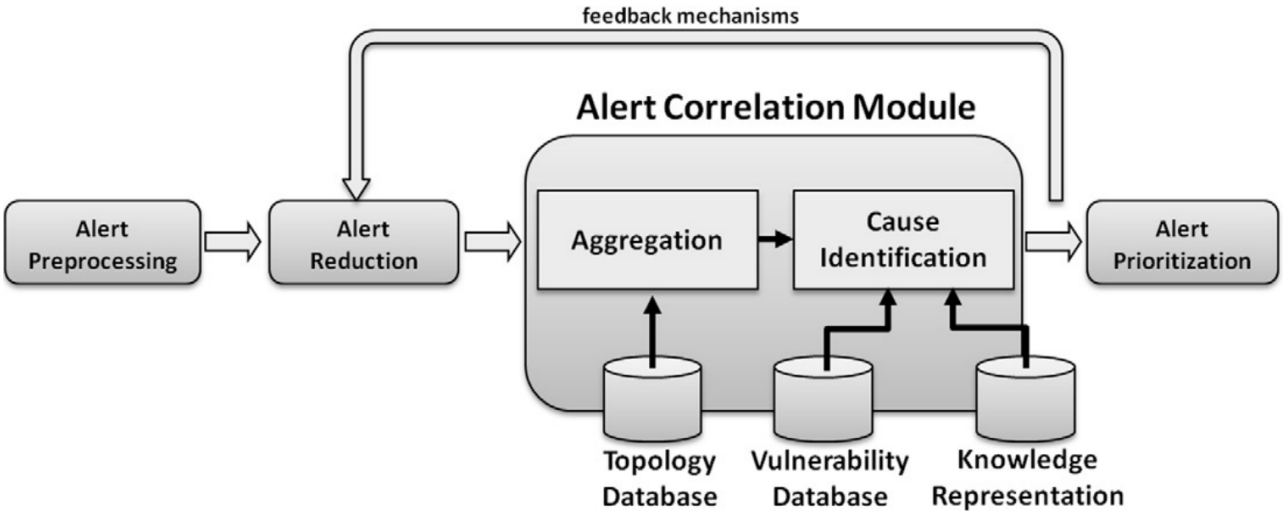


Figure 1. An architecture of the proposed alert/event correlation process model by Salah *et al.* [12]

presence of heterogeneous sensors in the literature; meanwhile, heterogeneous event aggregation is necessary for timely detection and response to the APT attack. Also, most methods in the literature rely only on aggregating the events of an individual detection level, i.e., the NIDS sensor from the network level. In addition, most try to provide the maximum amount of reduction in the volume of logged events, while they do not care about the lack of security information in the event logs.

In this paper, to overcome the current limitations of the existing event aggregation methods, a heterogeneous event aggregation method is proposed to reduce the volume of logged events during the IKC stages of APT attacks. To this primary goal, after collecting logged events by the various heterogeneous security and non-security sensors, they are first normalized into a standard log format to perform other operations related to event aggregation. Then, in the first phase, according to a defined time window for receiving events by the aggregation component, they are classified regarding the type of sensor. Afterward, the related events to each sensor are clustered based on attribute-based similarity matching. Next, in the second phase, the generated event clusters are analyzed using a clustering-based local outlier factor method to remove and discard noisy events and false-positive. Finally, in the third phase, the events in event clusters are summarized based on an attribute-oriented induction method to summarize events flexibly and form final aggregated events. Based on the experiments on some publicly available datasets, SotM34, Bryant, and LANL, the proposed method can reduce the volume of logged events by the heterogeneous sensors to a maximum level with an acceptable level of security information loss. The main contributions of this paper are as follows:

- First, we provide a mapping between different IKC stages of the APT attacks and heterogeneous sensors of the three detection levels (Network, Host, and Application) with their related event features.
- Second, we propose a three-phase aggregation method using clustering-based and attribute-oriented induction methods for aggregating and summarizing heterogeneous events.
- Third, we conduct an extensive performance study based on the three publicly available datasets to show the effectiveness of our approach and compare the method with existing approaches regarding the main standard evaluation metrics.

The rest of this paper is organized as follows. In Section 2, state-of-the-art is reviewed to present some related works to the alert/event aggregation methods. In Section 3, the relevant literature is reviewed to give background information about the IKC models of APT attacks and the most critical assumptions and necessities to provide an event aggregation method in the context of APT attacks. Section 4 presents the proposed event aggregation method with the related algorithms. Also, a concise and valuable running example is provided to demonstrate the functions of the proposed method components throughout the section. In Section 5, we conduct several experiments based on our proposed method using different well-known and standard datasets in the intrusion detection field to evaluate the correctness and performance of our approach by comparing it with the other related works. In addition, in this section, several discussions are presented to analyze the effectiveness of the parameters of the proposed method on each other, along with the benefits and advantages of the methods for researchers and practitioners. Finally, Section 6 concludes the

paper and presents future work.

2 Literature Review

In the literature, many research works have been found with different event aggregation approaches for reducing the volume of logged events/alerts of various sensors to improve their quality. Table 1 provides a comparative analysis of the existing research on events/alerts aggregation methods. This section briefly describes the leading existing research works in this area.

Bryant and Saiedian in [7] proposed a framework to model the APT attack life cycle for forensics investigation in the form of a newly introduced 7-phase IKC model explained in the previous subsection. This model focuses on phases that leave trails inside the victim's network and can be observed within sensor data at different detection levels. Their proposed system reconstructs APT attack scenarios by aggregating and correlating logged low-level events with heterogeneous event log sources. During the IKC detection process, their proposed correlation component aggregates low-level events of each security sensor in a detection level regarding a perfect similarity matching process based on one or more event features. To this aim, source and destination IP addresses are used for aggregation analysis of network-level sensors like NIDS, Firewalls, and Edge Router. Also, the Host/Computer Name and User/Account Name are used to aggregate host-level sensors like Host OS, HIDS, and Antivirus. In addition, for aggregating Application-level services like DNS, Email, and Web Servers, the IP address is used for similarity analysis.

Ramaki *et al.* in [16] and [29], and also Soleimani and Ghorbani in [22] have proposed two alert correlation frameworks that are similar to each other to detect multi-step attacks based on multi-layer episode mining algorithms. The input of the two developed systems is IDS alerts produced by the Snort IDS. During the correlation process by the proposed framework, one of the main components is the aggregation component for constructing hyper-alerts by fusing low-level alerts and removing duplicated and unrelated alerts. The proposed aggregation component in these frameworks combines all individual alerts with the same attack types. It creates a synthetic hyper-alert for each attack type based on an aggregation time interval as the time windows. The resulting hyper-alert is an inclusive alert, which includes all similar alerts in a specified time window. By applying attribute-based similarity analysis, they have obtained an acceptable level of event aggregation ratio (EAR), about 99.94%, for the Snort alerts of multi-step attack scenarios in the DARPA dataset. However, their approach is limited to the NIDS sensor and does not apply to a network environment with multi-sensor, disparate and heterogeneous sensors.

Another research work is done by Husak *et al.* [17]. The authors of this paper proposed an alert aggregation framework for NIDS alerts. They presented four use cases for the aggregation of alerts from NIDS sensors: duplicated alerts, continuing alerts, aggregating alerts from overlapping sensors (overlap in their detection scope), and aggregating alerts from non-overlapping sensors. They used an attribute-based similarity analysis for the alert aggregation process based on the main IDS alert features: source IP and port number, destination IP and port, event type, sensor ID, and timestamp. By doing some helpful case studies on a private network, they found that the system can aggregate volumes of low-level alerts up to 85% in the EAR rate. Like this work, Nadeem *et al.* [27] proposed an attack graph generation process with an alert aggregation component. This component is responsible for aggregating the IDS alerts, resulting in which hyper-alerts or attack episodes are generated. By using attribute-based similarity analysis on the alerts, their method yields a reduction ratio of 99.98% of the original alert set.

Spathoulas and Katsikas [18] proposed an alert post-processing framework to improve the quality of generated alerts by the NIDS sensor using alert aggregation. The input of their developed system is alert sets of multiple NIDS sensors. Their proposed system operates in three main phases, 1) the preparation phase for merging the different received alert sets by various NIDS sensors into one alert set, including aggregated alerts with minimum information redundancy, 2) the clustering phase for creating clusters of similar events by using a set of defined similarity functions for each of event features, and 3) visualizing phase for graphically representing generated clusters to depict overall security status of the monitored network. The alert aggregation based on each attack ID in the first phase for creating aggregated alert sets and then clustering analysis on the output alert sets based on some similarity functions provides insights into the main explicit and implicit events that may be happening on the monitored network.

Fredj in [28] has proposed a global alert correlation system that receives heterogeneous security alerts from network-level sensors, i.e., NIDS and Firewall, unifies them by a customized event normalization format, discards false and unrelated alerts, aggregates remaining important once, correlates normalized generated alerts for attack scenario reconstruction. To minimize the number of generated alerts by the security sensors and also decrease the overall overhead of the alert correlation process, the use of an IP-based

Table 1. Overview of the related works to the event aggregation

Class \ Metric	Sensor Name(s)	Aggregation Field(s)	Aggregation Method	Evaluated Performance Metric(s)	Ref.
Single-type Sources	NIDS	Source IP Address, Source Port Number, Destination IP Address, Destination Port Number, Timestamp, Event Type, Attack Severity	Attribute-based Similarity Analysis	EAR	[16]
	NIDS	Sensor ID, Source IP Address, Source Port Number, Destination IP Address, Destination Port Number, Timestamp, Event Type	Attribute-based Similarity Analysis	EAR	[17]
	NIDS	Source IP Address, Destination IP Address, Attack ID, Attack Classification	Clustering Analysis	EPR	[18]
	NIDS	Source IP Address, Source Port Number, Destination IP Address, Destination Port Number, Timestamp, Attack ID, Event Name, Attack Classification, Priority, Protocol	Clustering Analysis, Text Similarity Check	EAR	[19]
	NIDS	Source IP Address, Source Port Number, Destination IP Address, Destination Port Number, Timestamp, Attack Classification, Vulnerability ID (CVE)	Temporal Relationship Analysis	EAR	[20]
	NIDS	Source IP Address, Source Port Number, Destination IP Address, Destination Port Number, Timestamp, Attack Classification, Priority, Protocol Type, TTL, TOS, Packet Length	Attribute-based Similarity Analysis	EAR	[21]
	NIDS	Source IP Address, Source Port Number, Destination IP Address, Destination Port Number, Timestamp, Event Type, Attack Severity	Attribute-based Similarity Analysis	EAR	[22]
	NIDS	Source IP Address and/or Destination IP Address, Source Port and/or Destination Port, Timestamp, Attack Severity, Autonomous System Number (ASN), Protocol Name	Clustering Analysis	EAR	[23]
	NIDS	Source IP Address, Source Port Number, Destination IP Address, Destination Port Number, Timestamp, Attack Classification	Rough Set Theory	EAR	[24]
	NIDS	Source IP Address, Destination IP Address, Timestamp, Attack Classification, Vulnerability ID (CVE)	Clustering Analysis	EAR, EPR	[25]
	NIDS	Source IP Address, Destination IP Address, Timestamp, Attack Classification	Ontology-based Similarity Analysis	EAR, ILR	[26]
	NIDS	Source IP Address, Destination IP Address, Destination Port, Timestamp, Attack Classification	Attribute-based Similarity Analysis	EAR	[27]
Multi-type Sources	Firewall, NIDS, Antivirus, HIPS, Domain Controller, Host OS, Edge Router, NetFlow	Source IP Address, Destination IP Address, Computer Name, Account Name, Login Name	Attribute-based Similarity Analysis	–	[7]
	Firewall, NIDS	Source IP Address and/or Destination IP Address, Source Port and/or Destination Port, Attack Classification, Attack Severity	Attribute-based Similarity Analysis	EAR	[28]
	Firewall, NIDS	Source IP Address, Source Port Number, Destination IP Address, Destination Port Number, Timestamp, Event Type, Attack Severity	Attribute-based Similarity Analysis	EAR	[29]

alert aggregation component that works based on a set of aggregation rules, i.e., similar source IP, similar destination IP, and similar C class network address to create alert clusters. However, it claims that the aggregation component can deal with heterogeneous log sources but is limited only to the network-level sensors.

Sun and Chen in [19] have developed an IDS alerts aggregation system based on the frequent pattern growth-based approach. Their proposed system consists of three main components: removal of noisy data, mining association rules, and text similarity check. They used the density-based spatial clustering of applications with noise (DBSCAN) algorithm for removing noise alerts. After filtering the noise data, they applied the FP-Growth algorithm to mine association rules based on a predefined time interval. By using these rules, the alerts are grouped into one meta-alert. Then, a text-similarity analysis of the resulting meta-alerts outputs the final aggregated alert set.

Alserhani in [20] has provided an alert correlation framework for detecting multi-step attack scenarios called MARS. In the MARS engine, one of the main components is event aggregation. This component is responsible for fusing a series of NIDS alerts related to a single step of a multi-stage attack based on a temporal relationship. To this aim, they used a predefined window time for determining whether two generated alerts by the sensors are close enough to be aggregated into a single alert. In the MARS framework, low-level alerts are modeled into a directed acyclic graph (alerts as vertices and casual relationships between alerts as edges). Further aggregation based on a graph reduction technique removes duplication in graph vertices. Their experimental results on the DARPA dataset show that their method can reduce Snort alerts by a maximum EAR rate of about %98.6 under certain conditions.

Shittu *et al.* in [21] have proposed a comprehensive system for analyzing intrusion alerts, called ACSAnIA, which operates some post-correlation analysis on generated alerts set by a set of NIDS sensors. These analyses include clustering and prioritizing alerts. One

of the key benefits of the ACSAnIA framework is that it aggregates low-level alerts using similarity analysis to construct meta-alerts. They are a group of alerts with identical values for the specified event features, i.e., intrusion type, signature ID, source and destination IP addresses, and source and destination port numbers. In this paper, for evaluation purposes, a cluster quality metric, called silhouette coefficient, is used for identifying well-clustered alerts where there is a high intra-similarity between the alerts of each cluster but a low inter-similarity between the various alert clusters.

Carlisto *et al.* in [23] proposed multi-step attack extraction models using process mining-based approaches. Based on their proposed method, the two main concepts in process mining are mapped to the area of intrusion alert analysis, activity, and case (process instance). To this aim, each of the logged event by the various NIDS sensors corresponds to an activity, and a case is multiple events that are grouped and share the same information. In their proposed approach, the aggregation component generates possible cases regarding typical features of the alerts. Each case is related to a single step of a complex attack. This work has two main aggregation strategies: one-to-many (alerts with one single source IP address as attackers and many destination IP addresses as targets) and many-to-one (alerts with many source IP addresses as attackers and one single destination IP address as targets).

Zhang *et al.* in [24] proposed an alert aggregation framework for network-level sensors (NIDS and Firewall) based on the Rough Set Theory. The proposed framework consists of the four main steps as follows: 1) Alert normalization for converting logged event formats to the standard IDMEF format to have a unified set of events, 2) Feature weight assignment for computing the importance of each event feature according to information provided by it, 3) Similarity analysis for computing similarity between two received alerts based on a predefined threshold, and 4) comparing time interval of alerts where if two detected similar alerts (similarity $\geq$ threshold) by the previous component occurred at a very close time interval, they are aggregated. Their experimental results show that their method can reduce low-level alerts by a maximum EAR rate of about %98.

Kim *et al.* in [25] proposed a scalable security event aggregation system over MapReduce called SEAM-MR. Their proposed system uses big data technologies to deal with large-scale security data generated during modern attacks, i.e., APTs. The SEAM-MR contains three core functions, namely, 1) periodic aggregation for collecting and aggregating events within the last time window, 2) on-demand aggregation for accessing users to the system for the aggregation analysis, and 3) query support for practical analysis to retrieve aggregated events for situation analysis. They presented seven usecases of aggregation of events from NIDS sensors regarding their typical features, i.e., attack source, attack target, attack source group, attack target group, and attack class. The aggregation engine provides a set of MR functions for the seven situations. They evaluated their method on a Hadoop cluster using a variety of synthetic datasets with different properties. They reported that the SEAM-MR system decreases the volume of some datasets up to 85%.

Saad and Traore in [26] have developed an IDS alert aggregation component to tackle the alert flooding problem. To this aim, they proposed a new alert aggregation and reduction technique based on semantic similarity between IDS alerts. The key idea in the proposed technique is that alerts that correspond to the same attack instance are semantically similar, even though they have different formats. Regarding the developed approach, semantic similarity measures for every two different alerts based on the attributes of an IDS alert by proposing an ontology for the intrusion detection domain. The key benefit of this approach is information preservation during alert aggregation purposes. Also, the authors proposed a new metric, information loss ratio (ILR), and showed that their system has a lower ILR metric level than traditional aggregation techniques. Although ILR is a good metric for aggregation components, their ontology is limited to IDS alerts. In this paper, we intend to expand the ontology for all event types of different sensors in all three mentioned detection levels.

3 Preliminary Research Questions and Findings

According to the primary goal of this paper mentioned in Section 1, after reviewing the event aggregation methods presented in Section 2, it is required to review state of art for answering the five main research questions (RQs) to obtain this goal and guide us to propose a novel event aggregation method. These RQs are depicted in Table 2 along with their rationality to justify their relation to the primary goal of this study. In the rest of this section, firstly, the literature is reviewed in Section 3.1 to Section 3.5, and finally, the results are discussed in Section 3.6.

3.1 Typical Stages of IKC Models Used in APT Attacks. (Regarding the RQ1)

To the best of our knowledge, based on a systematic literature review (SLR) method in the area of APT attacks, there are 18 different IKC models proposed

Table 2. The main research questions (RQs) for the literature review

RQ#	RQ	Title Rational
1	What are the common stages of IKC models used in APT attacks?	By answering this RQ, we understand the existing IKC models in state of the art and the common stages of them used in APT attacks along-with the malicious activities of the attackers during the stages. In addition, we choose a proper IKC model to present our proposed event aggregation method based on it.
2	How is the layered security model used to log malicious activities of attackers behind APT attacks?	By answering this RQ, we find out how the security staff tracks the signs of malicious activities of the attackers during different stages of the IKC model. Also, we present a mapping between chosen IKC stages and detection levels of the layered security model.
3	What are the most suitable sensors used to log events during IKC stages?	By answering this RQ, we identify the most suitable security and non-security sensors available at different detection levels as the main low-level event generators. In addition, we recognize the main event features of each sensor for later usage in our proposed event aggregation method.
4	What is a proper tool stack for event collection and processing in the presence of heterogeneous sensors and how does it work?	By answering this RQ, we understand the capabilities of a good event log collection tool that can gather, analyze, and store generated raw event logs by various sensors at different detection levels.
5	What is a suitable intrusion alert/event normalization format for tokenizing information fields within the events logged by heterogeneous sensors?	By answering this RQ, we find out how logged events by various sensors and different vendors can convert to a unified data format to be understood by the components of the proposed event aggregation method.

in this field both in academic research works including [7, 9, 30–35], and reports of industrial companies containing [36–43]. Each of all the existing IKC models has a specified number of stages (phases) used to perform the targeted attacks using a set of intrusion activities [5]. Based on the analysis done in [7], most of the introduced models have a typical set of attack stages used in APT attacks, which has led to the introduction of the Bryant kill chain model. The developed IKC model by Bryant and Saiedian has been made using some modifications to previous models, [7] and [30], which makes it a suitable choice for data-driven analysis, i.e., event aggregation in the security solutions like SIEM, SOC, and threat hunting. One of the most important considerations that make this model suitable for event logs-based processes in applications like APT attack detection, threat intelligence, and cyber situational awareness is the elimination of the weaponization stage available in former IKC models due to the failure to track the attacker's activities on the victim network. The main macro-phases and stages of the Bryant IKC model are depicted in Figure 2.

As shown in Figure 2, the Bryant IKC consists of seven stages which are described as follows (abbreviations are used throughout the following sections):

- **Reconnaissance (REC)**: In this stage, the attackers gather information about the target. The main objective of this stage is to gain knowledge about the victim and find practical methods and technologies to intrude into the network.
- **Delivery (DEL)**: In this stage, the attackers try to deliver a prepared attack payload to the network by using the established connections between attacker-side nodes to a victim or a collection of victims on the target side.
- **Installation (INS)**: In this stage, the attackers' primary objective is to install attack payload/malware on the infected systems. This stage is necessary for persistent access to further intrusion activities.
- **Privilege Escalation (ESC)**: In this stage, attackers aim to escalate their privileges on the victims, which helps them move the target to find valuable information.
- **Lateral Movement (LAT)**: The main objective of this stage, sometimes called internal reconnaissance, is to differentiate between the reconnaissance activity from the external and internal networks. This stage is optional based on the attack goals for moving laterally within the compromised network.
- **Actions on Objects (ACT)**: By using this stage, the adversaries achieve the attack goals by performing several destructive activities inside the target network, which can take months.
- **Exfiltration (EXF)**: In this stage, after the successful execution of attack stages on the final objects, the attackers attempt to exfiltrate sensitive data from the target network by using suitable communication and control (C&C) servers. In addition, to complete the mission, he/she tries to delete intrusion evidence from compromised machines.

Regarding Figure 2, it should be noted that some attack stages of the Bryant IKC model are optional and may not be performed by the attackers, or someone may be repeated several times under different conditions by them. In addition, sometimes, the stages of the IKC may not be by the order shown in the figure exactly because the attacker may not be successful at one stage and may have to repeat the previous stages of the IKC model. Generally, each stage of the IKC model has some everyday intrusion activities as at-

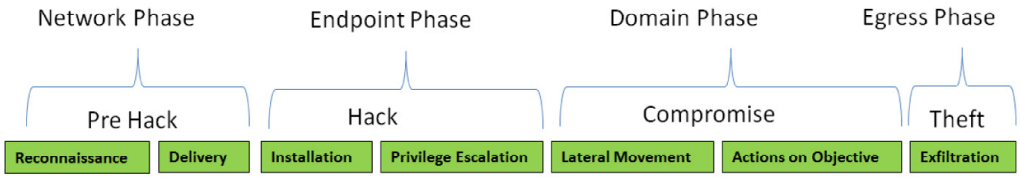


Figure 2. Bryant kill chain model macro-phases and attack stages [7]

tack steps used by the attackers for attack progression during the IKC. Table 3 presents a mapping between the Bryant IKC stages and the commonly used intrusion activities by the adversaries behind APT attacks. Based on the results of this table, the malicious activities used by the attackers range from passive attacks, i.e., various types of network scanning and social engineering attacks, to active ones, i.e., vulnerability exploitation, malicious code execution, software modification, and eventually system destruction or data theft.

3.2 The Role of the Layered Security Model in Network Security (Regarding the RQ2)

In the network security area, the layered security model is defined as the concept of securing a computer network through a sequence of defensive mechanisms so that if one fails, another is already in place to prevent an attack [6, 7, 44]. The basic assumption of this model is that no single solution can successfully safeguard the network from attacks due to the great variety of attacks. According to Figure 2, when the attacker initiates an APT attack by using different intrusion activities at the network level, he/she accesses his/her goal in the compromised machines of the internal domain, followed by some intrusion activities in the host and application levels [7, 9, 10]. Hence, regarding Figure 3, three primary detection levels are used for deploying various security and non-security sensors to log events during the intrusion activities of each IKC stage. In other words, an APT attack scenario based on the IKC model consists of a succession of events logged by security and non-security sensors. These logged events by heterogeneous sensors with different functionalities can be categorized into three main categories, 1) benign or normal events (uncolored points), 2) suspicious events (colored points), and 3) attack-related events (colored and patterned points). According to Figure 3, 1) a significant part of the logged events during attack scenarios are benign, 2) the variety of logged suspicious events depends on the number of deployed sensors in the monitored network, 3) suspicious events are related to failed attack attempts, and 4) the attack-related events may construct different attack scenarios. A mapping between the IKC stages and the three primary detection levels is depicted in Figure 4. The colors in this figure, from green to red, indicate the risk level of the attacker's activities advancing the IKC model from low to high.

In this paper, the selected Bryant IKC model is used for two primary purposes, 1) designing a mapping between IKC stages and their related sensors and 2) presenting the behavior of APT attackers based on the IKC stages during a sample attack scenario. The first is explained in Section 4.1, and the second is provided in the following. To this aim, Figure 5 shows an example IKC-based attack scenario and its stages as a sequence diagram to model the attack life cycle. As illustrated in the figure, during the attack scenario, the attacker aims to intrude into the internal network to steal a credential file of the organization. The main steps of the APT attack scenario with the related IKC stages are provided in the following. It should be noted that this example is used in the following sections to understand how the proposed aggregation method works.

(1) **Active Scanning (REC)**: The attacker initiates (with IP address 172.25.110.11) the attack by scanning the target network (IP address range 192.168.1.1/24) to find some organization's hosts, which are up for delivering various services.
(2) **Application Vulnerability Scanning (REC)**: After scanning activities and identifying open ports on a specified host (with IP address 192.168.1.12), the attacker attempts to gather information about the system by enumeration. In this step, the attacker finds a vulnerable Adobe Reader application on the enumerated host.
(3) **File Attachment (DEL)**: According to the founded vulnerability, the attacker develops malware, attaches it to a PDF file, and sends the malicious PDF to the target host via an email.
(4) **Drive by Download (INS)**: After opening Gmail in the browser by the user on behalf of the targeted host, he/she is tempted to download the attached malicious PDF with an embedded malicious payload.
(5) **Malicious Code Execution (INS)**: After downloading, the user starts the trojan PDF file installation process which causes the embedded malicious payload to open a reverse connection to a remote C&C server.

Table 3. Common intrusion activities during each stage of the Bryant IKC model

No.	Bryant IKC Stage	Intrusion Activities
1	Reconnaissance	Network Probing, Active Scanning, Social Engineering, Network Topology Obtaining, Web Application Vulnerability Scanning, Database Vulnerability Scanning, Zero-day Vulnerability Scanning, Spear Phishing, Email Spam, Phishing Websites, Acquiring Removal Devices
2	Delivery	Abusing Benign Applications' Vulnerabilities, File Attachment, Spear Phishing Link, Compromised Legitimate Websites, Rogue DNS, Rogue Software, SQL Injection
3	Installation	Drive-by Download, Vulnerability Exploitation, Malicious Code Execution, Software Modification, RAT and Installing Backdoor and Rootkit, Modifying Registry Key, DLL Hijacking, DLL Side Loading, Scheduled Task
4	Privilege Escalation	Create or Modify Valid Accounts, Create or Modify System Process, Domain Policy Modification, Vulnerability Exploitation, Process Injection, Defense Evasion, Credential Access
5	Lateral Movement	Internal Reconnaissance, Internal Spear Phishing, Session Hijacking, Lateral Tools Transfer, Remote Service Exploitation
6	Actions on Objects	Command Execution, Reverse Connection, Sensitive Data Collection/ Manipulation/Deletion, Email Collection, Data Obfuscation, Capture Keystroke, Violation Against Data, DDoS Attack, Maintaining Remote Access, Clear Logs
7	Exfiltration	Data Exfiltration/Theft, Automated Exfiltration, Scheduled Transfer

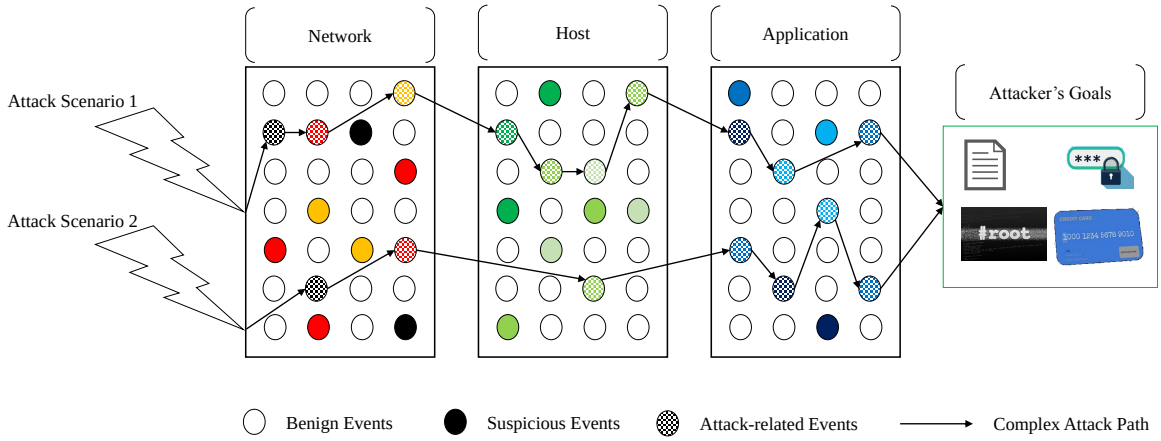


Figure 3. Logging events by heterogeneous sensors at different detection levels, Network, Host, and Application

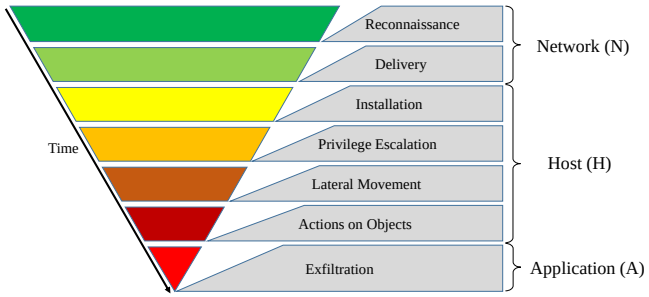


Figure 4. Bryant IKC stages mapping to the detection levels, Network, Host, and Application

(6) **Command Execution (ACT)**: After establishing the connection between a compromised host and the C&C server, the C&C server sends a command to search for a credential file containing an employee's private information.

(7) **Maintaining Remote Access (ACT)**: The C&C server forwards some commands to download a vulnerability scanner tool from the server through the file transfer protocol (FTP).

(8) **Sensitive Data Collection (ACT)**: After downloading the scanner, the compromised host as a C&C client receives commands to run the downloaded scanner and send the credential file to a specified remote web server.

(9) **Data Exfiltration (EXF)**: After executing commands by the infected system, it sends the credential file to the web server through hypertext transfer protocol (HTTP) protocol.

(10) **Internal Reconnaissance (LAT)**: To collect more information about the targeted network, the attacker starts to move laterally in the internal network by using the conquered machine as a pivot for the later intrusion activities.

3.3 Suitable Sensors for Logging Events during IKC Stages (Regarding the RQ3)

According to the related studies [5, 7, 9, 13, 16, 45], to track APT attacks, at first, it is required to collect events logged by the heterogeneous sensors which are deployed in different detection levels. Sample monitored network with heterogeneous sensors from different detection levels is shown in Figure 6. As shown in

this figure, the collector agents transfer logged events by the Network level sensors (i.e., IDS, Router, and Switch), Host level sensors (i.e., HIDS, Antivirus, and OS logs), and Application level sensors (Email, Web and Database servers) to a security incident management platform of the organization like a SIEM solution.

Each sensor has a specified set of attributes/features according to the sensor functionality within a predefined format. Based on our analysis, many studies have mentioned the suitable main sensors for logging security and non-security events alongside their features. However, these are not discussed in detail [7, 8, 10, 11, 17, 18, 28, 44–48]. Table 5 presents the most suitable heterogeneous sensors in each detection level with their event features used as the input for the proposed event aggregation method.

As mentioned before, during each attack stage of an IKC model, some low-level events are logged by the various heterogeneous sensors of the different detection levels. For example, a list of logged low-level events during the attack steps of the sample APT attack scenario (Figure 5) is shown in Table 6. According to the table, each event has a unique sensor ID (SID) and a set of features logged by the sensor which is provided in Table 5 for each SID. The empty cells in the table show that the related sensor has no value for the feature. It should be noted that for simplicity, only some essential features of the sensors are mentioned in the table.

3.4 The Proper Tool Stack for Event Processing (Regarding the RQ4)

Generally, before later processing on the logged events, i.e., event aggregation, at first, low-level events logged by the heterogeneous sensors must be collected and stored. So far, in the security event collection and monitoring field, various tools have been developed with different functionalities and capabilities [49]. One of the most widely used of them is the Elastic or ELK stack. This platform consists of three main parts (nodes), Elasticsearch for event gathering and indexing, Logstash for event normalization and transforming, and Kibana for event visualization and dashboarding [50]. In the ELK architecture, some agent-based event collectors are called Beats [51]. Beats are data shippers (log forwarders) of the ELK stack, which have different types for various platforms and applications, i.e., Filebeat and Winlogbeat. The main reasons for choosing Beats of ELK stack to collect logged events in our method are as follows: 1) the ability to deploy agents in different detection levels (Network, Host, and Application), 2) their high-speed capability and good performance delivering in event collection and processing, and 3) provide scalability due to the distributed architecture of the ELK stack nodes.

3.5 The Suitable Event Normalization Format (Regarding the RQ5)

After collecting heterogeneous events, the gathered events may have various formats according to the variety of output log formats of a specific sensor [12, 13, 16]. Hence, before any later processing like aggregation analysis, it is first required to convert the event format to a unified and standard log format to understand more of the meaning of the events by the other components [12, 13]. The event normalization component is a fundamental component of the schema that converts all received low-level events into a joint and standard log format. Till now, security researchers in the intrusion detection field have developed several event normalization formats [13, 52]. To our knowledge, one of the recently developed normalization formats appropriate for applying in an environment with heterogeneous sensors is object log format (OLF) [52].

In this paper, we use the Logstash module of the ELK stack to convert received events to the OLF format. For this purpose, based on the output of each sensor in the system, first, the set of required information fields (features) is determined based on the OLF format for normalization. Next, essential fields of that sensor event are extracted and tokenized by defining a regular expression in the form of a well-defined Regex pattern for each type of sensor. For example, this operation for the output of the Snort IDS sensor from the network detection level is depicted in Table 4. The used Regex pattern in this figure is the Grok pattern, a facility of Logstash module [1]. It should be noted that each sensor has its own Grok pattern for event normalization purposes. After extracting the main information fields from the logged events by a unique sensor, they can be stored in a CSV file format with column names of OLF format related to each security sensor for further usage.

3.6 Discussion

In this section, the most important results obtained from the literature review are discussed. The discussion is provided to understand the main challenges and issues of the existing works and the requirements and capabilities of a recommended event aggregation method capable of reducing the volume of logged events by heterogeneous sensors. The main remarkable results are as follows:

- After studying the details of the existing IKC models in Section 3.1, it can be found that the Bryant kill chain presents the typical attack stages of them used in APT attacks. Hence, this

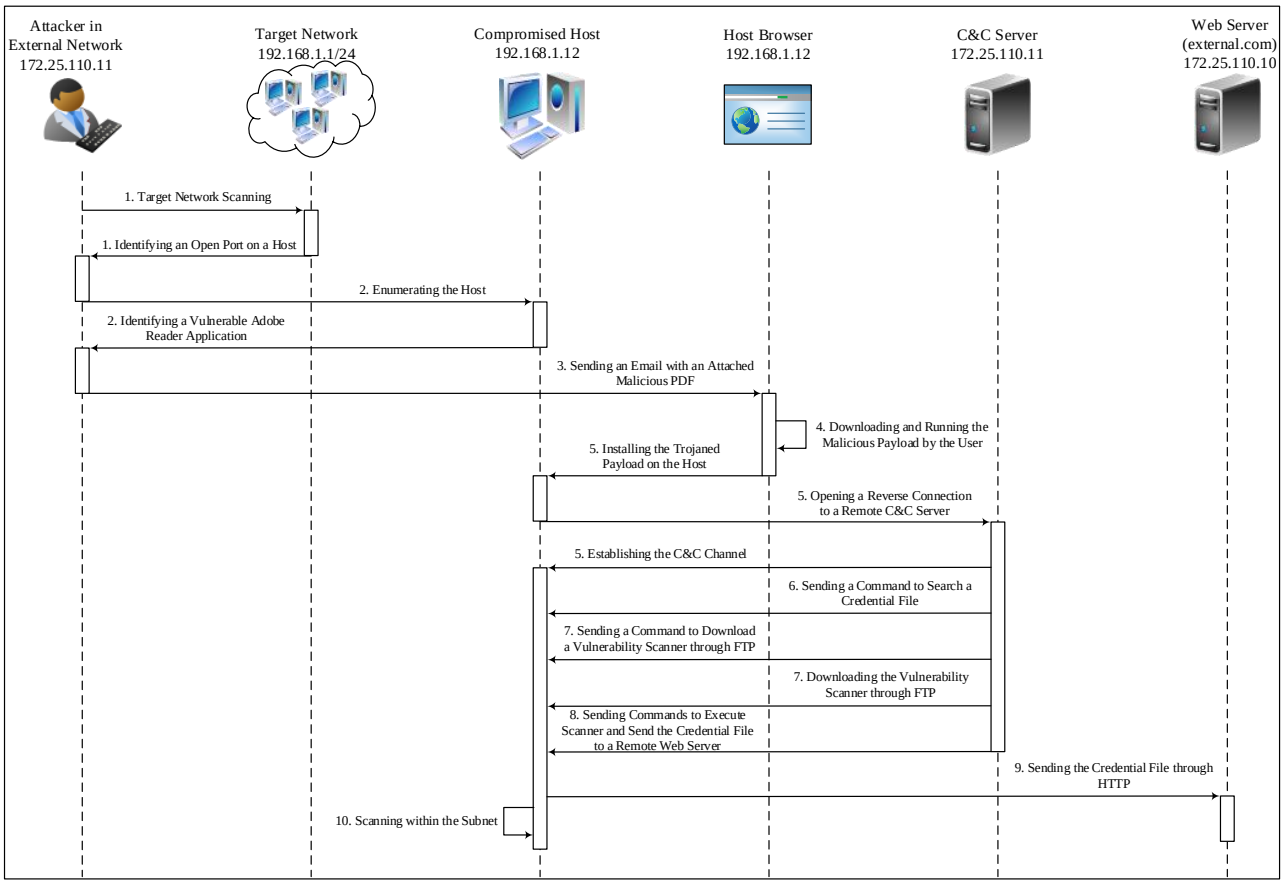


Figure 5. A sample APT attack scenario based on the Bryant IKC stages

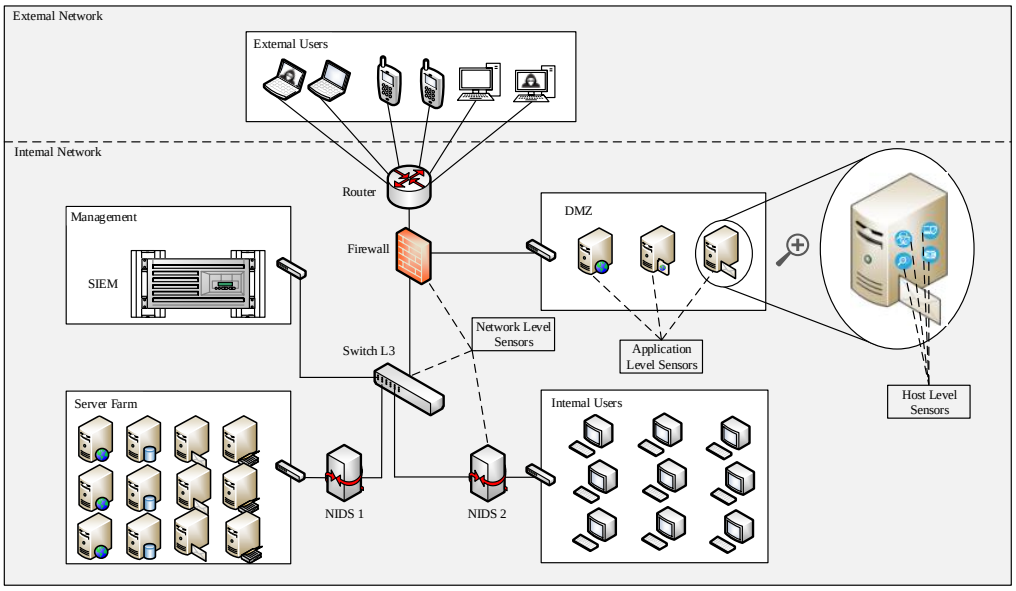


Figure 6. Sample monitored network with deployed heterogeneous sensors

IKC model is chosen for further usage related to the aggregation analysis. In addition, a publicly available dataset by the presenters of this model will be a good case for the evaluation purposes of the proposed method.

- By investigating the layered security model and finding its relationship to the typical stages of the IKC models used in APT attacks (Section 3.2), it can be found that the primary detection levels as layers of security for tracking

Table 4. Sample converted Snort IDS event type of alert to the OLF format

Raw Event	Mar 31 20:20:21 bastion snort[2546]: [1:2003:8] MS-SQL Worm propagation attempt[Classification: Misc Attack] [Priority: 2]: UDP 60.40.70.25:3354 -¿ 11.11.79.90:1434
Regex Pattern	{%{MONTH}+%{MONTHDAY+%{TIME}:time}%{HOSTNAME:host}%{PROG:appname}{\[%{INT:pid}\]\:\[%{INT:generator˙id}:%{INT:signature˙id}:%{INT:signature˙revision˙id}\]:event˙id\}(?< module > \b[\w\−] + \b)%{DATA:msg}\[Classification\:%{DATA:original˙event}\]\[Priority\:%{INT:priority}\]\{%{WORD:protocol}\}%{IP:src˙addr}:%{INT:src˙port}\−\>%{IP:dst˙addr}:%{INT:dst˙port}}
Normalized Event in OLF	Mar 31 20:20:21(time) bastion(producer -¿ host) snort(producer -¿ appname)[2546(producer -¿ pid)]:[1:2003:8](event_id) MS-SQL(producer -¿ module) Worm propagation attempt(msg) [Classification: Misc Attack](original_event) [Priority: 2]:(priority) UDP(network-¿udp) 60.40.70.25(network-¿src_ipv4) 3354(network-¿src_port) 11.11.79.90(network-¿dst_ipv4) 1434(network-¿dst_port)

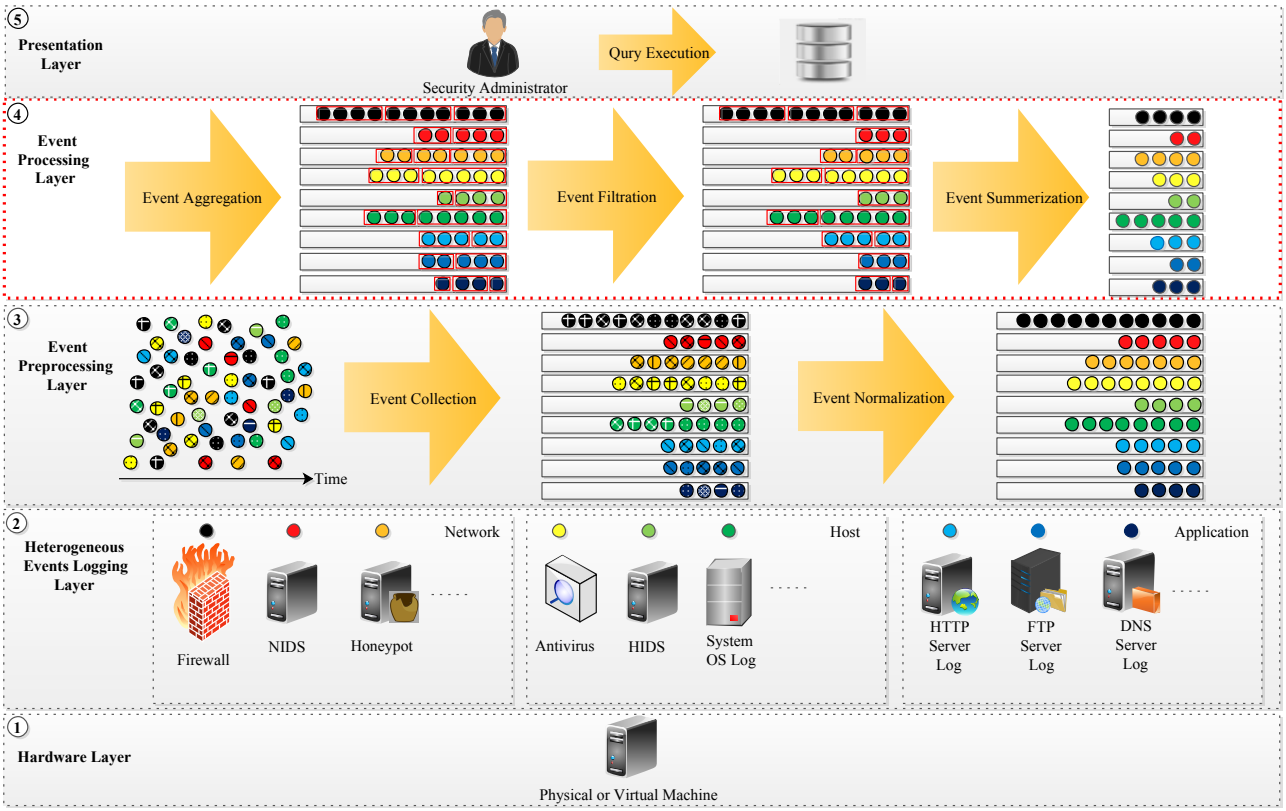


Figure 7. An abstract schema of a heterogeneous event aggregation framework

and mitigating the intrusion activities during the stages of IKC model are Network, Host, and Application. Hence, based on these detection levels, we present a set of heterogeneous security and non-security sensors useful for logging APT attack symptoms during the IKC stages and their related attack steps (Section 3.3). Besides, for proposing our event aggregation method, a good event feature analysis is done based on the suitable sensors related to each detection level.

- According to the presented analysis in Table 1 (Section 2), we found that many different techniques have been used in the aggregation component of an alert/event correlation system, including statistical, clustering, and attribute-based similarity analysis. Generally, the main shortcoming of the existing machine learning-based methods is their low flexibility in adapting to the needs of users and experts for the degree of aggregation. In addition, these methods are impractical for conducting aggregation analysis with massive data generated by heterogeneous sensors due to high computational costs and poor performance. Moreover, it could be inferred that some research works suffer from producing irrelevant aggregated events due to failure to remove false-positive events that cause incorrect attack scenarios in the later stages of an event

Table 5. Heterogeneous security and non-security sensors of different detection levels and their event features

		Detection Level																			
		Network							Host							Application					
		Sensor-ID (SID)																			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
No.	**Feature**	**Network-based IDS (NIDS)***	**Firewall***	**Router****	**Honeypot***	**NetFlow****	**Network Behavior Analysis (NBA)***	**Deep Packet Inspection (DPI)****	**Host-based IDS (HIDS)***	**Antivirus***	**Anti-malware***	**Host OS Log****	**Audit Log***	**Domain Controller****	**Vulnerability Scanner***	**DNS Server****	**Mail Server****	**Web Server****	**Data Loss Prevention (DLP)***	**FTP Server****	**Proxy Server****
1	Timestamp	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2	Source IP Address	✓	✓	✓	✓	✓	✓	✓	–	–	–	✓	✓	✓	–	✓	✓	✓	✓	✓	✓
3	Source MAC Address	–	–	–	–	✓	–	–	–	–	–	–	–	–	–	✓	–	–	✓	–	✓
4	Source Port	✓	✓	✓	✓	✓	✓	✓	–	–	–	✓	✓	✓	–	✓	–	–	–	–	✓
5	Destination IP Address	✓	✓	✓	✓	✓	✓	✓	–	–	–	✓	✓	✓	–	✓	✓	✓	✓	✓	✓
6	Destination MAC Address	–	–	–	–	✓	–	–	–	–	–	–	–	–	–	✓	–	–	–	–	✓
7	Destination Port	✓	✓	✓	✓	✓	✓	✓	–	–	–	✓	✓	✓	–	✓	✓	–	–	✓	✓
8	Event/Attack Type/Class/Name	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
9	Event/Attack/Signature ID	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
10	CVE ID	✓	–	–	✓	–	–	✓	✓	–	–	–	–	–	✓	–	–	–	–	–	–
11	Priority/Severity Level	✓	✓	–	✓	–	–	✓	✓	–	–	–	–	–	✓	–	–	–	✓	–	–
12	Process Name	–	–	–	–	–	–	–	–	–	–	✓	✓	–	–	–	–	–	✓	✓	–
13	Process ID	–	–	–	–	–	–	–	–	–	–	✓	✓	–	–	–	–	–	✓	✓	–
14	Parent Process ID	–	–	–	–	–	–	–	–	–	–	–	✓	–	–	–	–	–	–	–	–
15	File Name	–	–	–	–	–	–	–	✓	✓	✓	✓	–	–	–	–	–	–	✓	✓	–
16	File Path	–	–	–	–	–	–	–	✓	✓	✓	✓	–	–	–	–	–	–	✓	✓	–
17	File Hash	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	✓	–	–
18	Folder Name	–	–	–	–	–	–	–	–	–	–	✓	–	–	–	–	–	–	–	–	–
19	Folder Path	–	–	–	–	–	–	–	–	–	–	✓	–	–	–	–	–	–	–	–	–
20	Registry Key	–	–	–	–	–	–	–	–	–	–	✓	–	–	–	–	–	–	–	–	–
21	Privilege Information	–	–	–	–	–	–	–	–	–	–	✓	–	–	–	–	–	–	–	–	–
22	Payload	✓	–	–	✓	–	–	–	✓	✓	✓	–	–	–	–	–	–	–	–	–	–
23	Queried Name	–	–	–	–	–	–	–	–	–	–	–	–	–	–	✓	–	–	–	–	–
24	Queried Type	–	–	–	–	–	–	–	–	–	–	–	–	–	–	✓	–	–	–	–	–
25	Resolved IP Address	–	–	–	–	–	–	–	–	–	–	–	–	–	–	✓	–	–	–	–	–
26	Message Type (Request/Response)	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	✓	✓	–	–	–
27	Message Length	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	✓	✓	–	–	–
28	URL Path	✓	–	–	–	–	–	–	–	–	–	–	–	–	–	–	✓	✓	✓	–	✓
29	Client Request Method (Get/Post)	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	✓	✓	–	✓
30	Response Code	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	✓	–	–	✓
31	Host/Computer Name	–	–	–	–	–	–	–	✓	✓	✓	✓	–	✓	–	–	–	–	✓	✓	–
32	User/Account Name	–	–	–	–	–	–	–	✓	✓	✓	✓	–	✓	–	–	–	–	✓	✓	✓
33	User/Account Group Name	–	–	–	–	–	–	–	✓	✓	✓	✓	–	–	–	–	–	–	–	–	✓
34	Referrer of Requested URI	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	✓	–	–	✓
35	Resolved	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	✓	–	–	–
36	Object Name	–	–	–	–	–	–	–	–	–	–	✓	✓	–	–	–	–	–	–	–	–
37	Time to Live (TTL)	✓	✓	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
38	Domain Name	–	–	–	–	–	–	–	–	–	–	–	–	–	–	✓	✓	✓	✓	✓	✓
39	Email Address	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	✓	–	✓	–	–
40	Session ID	–	✓	✓	–	✓	✓	–	–	–	–	–	–	–	–	–	–	–	–	✓	–
41	Protocol Type	✓	✓	–	–	✓	✓	✓	–	–	–	–	–	–	–	–	–	–	–	–	✓
42	Transmitted Bytes	–	–	✓	–	✓	✓	✓	–	–	–	–	–	–	–	–	✓	–	–	–	✓

*Security Sensor, **Non-security Sensor

correlation system.

- Regarding the provided comparison in Table 1, it can be found that the related works can be classified into two main categories, namely, single-type and multi-type event source. Single-type event source class introduces those research

Table 6. Sample logged events by heterogeneous sensors during the sample attack scenario in Figure 5

Step#	SID	ID	Data Time	Source IP Address	Source Port	Destination IP Address	Destination Port	Event ID	TTL	Protocc	Computer Name	User Account	File Name	File Path	Process ID	Other Features
1	1	e1	Feb 25 12:11:05	172.25.110.11	80	192.168.1.12	20	r-c1-1	63	UDP	–	–	–	–	–	–
	2	e2	Feb 25 12:11:10	172.25.110.11	2256	192.168.1.12	22	d-c2-1	39	ICMP	–	–	–	–	–	–
	1	e3	Feb 25 12:11:12	172.25.110.11	80	192.168.1.12	20	r-c1-2	55	UDP	–	–	–	–	–	–
	2	e4	Feb 25 12:11:28	172.25.110.11	2265	192.168.1.12	22	d-c1-1	48	ICMP	–	–	–	–	–	–
2	1	e5	Feb 25 12:11:30	172.25.110.11	2265	192.168.1.12	190	r-c2-1	159	TCP	–	–	–	–	–	–
	1	e6	Feb 25 12:11:01	172.25.110.11	2256	192.168.1.12	190	r-c2-2	252	TCP	–	–	–	–	–	–
	2	e7	Feb 25 12:12:32	172.25.110.11	443	192.168.1.12	22	d-c1-2	48	ICMP	–	–	–	–	–	–
	2	e8	Feb 25 12:12:36	172.25.110.11	2265	192.168.1.13	20	f-c3-1	252	TCP	–	–	–	–	–	–
3	1	e9	Feb 25 12:12:41	172.25.110.11	2265	192.168.1.12	190	r-c2-2	199	TCP	–	–	–	–	–	–
4	11	e10	Feb 25 12:12:41	172.25.110.11	–	192.168.1.12	–	h-c1-2	–	–	Client1	John	mal.pdf	c: documents	p1	Parent Process ID = p0
5	11	e11	Feb 25 12:12:46	172.25.110.11	–	192.168.1.12	–	h-c1-2	–	–	Client1	John	mal.pdf	c: documents	p2	Parent Process ID = p1
	11	e12	Feb 25 12:12:58	172.25.110.11	–	192.168.1.12	–	e-c2-2	–	–	Client1	John	mal.pdf	c: documents	p3	Parent Process ID = p1
	10	e13	Feb 25 12:13:08	172.25.110.12	–	192.168.1.12	–	h-c1-1	–	–	Client1	John	mal.pdf	c: documents	p1	Parent Process ID = p2
	5	e14	Feb 25 12:17:43	192.168.1.12	80	172.25.110.11	80	r-c1-1	–	–	–	–	–	–	–	–
6	13	e15	Feb 25 12:18:24	192.168.1.12	–	192.168.1.12	–	l-c1-1	–	–	Client1	John	–	–	–	–
	13	e16	Feb 25 12:18:38	192.168.1.12	–	192.168.1.12	–	g-c1-1	–	–	Client1	John	–	–	–	–
	12	e17	Feb 25 12:19:02	192.168.1.12	–	192.168.1.12	–	p-c1-2	–	–	Client1	John	plan.txt	d: mng/plan	p4	–
7	17	e18	Feb 25 12:19:12	192.168.1.12	1265	172.25.110.11	53	e-c1-1	–	–	–	–	–	–	–	Query Type = request, Queried Domain = external.com
	11	e19	Feb 25 12:19:35	172.25.110.11	53	192.168.1.12	1265	e-c1-2	–	–	–	–	–	–	–	Query Type = response, Resolved IP Address =172.25.110.10
8	1	e20	Feb 25 12:19:49	192.168.1.12	–	172.25.110.10	–	f-c1-2	–	–	Client1	John	plan.txt	d: mng/plan	–	–
9	12	e21	Feb 25 12:20:13	192.168.1.12	–	172.25.110.10	–	w-c1-2	–	–	–	–	–	–	p4	Host Domain = external.com, Referrer = external.com
10	15	e22	Feb 25 12:20:23	192.168.1.12	–	192.168.1.12	–	p-c1-1	–	–	Client1	John	nsu.exe	c: downloads	p5	–
	15	e23	Feb 25 12:20:55	192.168.1.12	1234	192.168.2.1	2145	s-c1-1	165	–	–	–	–	–	–	–

works which have proposed aggregation techniques for logged events by a unique security device [16–26]. The main concentration of this class is the network-based IDS (NIDS) sensors. In contrast, the related works to the multi-type event source class aim to aggregate heterogeneous events which are produced by different devices of a monitored network [7, 28, 29]. These works combine a series of events with similar event features, which refer to attack stages related to the same activity. However, most research works have provided a method to aggregate the events of a single-type event source, i.e., NIDS. Also, there is no suitable IKC-based aggregation method in the presence of heterogeneous sensors in the literature.

- In most related works, only the EAR is used to evaluate the aggregation methods. However, this is a volume-centric metric and, therefore, could not measure output quality in the event aggregation methods. To the best of our knowledge, there is no suitable method in the literature to consider information loss problems during the aggregation process in the presence of heterogeneous sensors.

In this paper, to address the problems mentioned above and issues, we propose a three-phase event aggregation method in the next section that collects, normalizes, clusters, filters, and summarizes logged events by various heterogeneous sensors during the IKC stages of APT attacks to generate aggregated events in a controllable and flexible manner.

4 The Proposed Event Aggregation Method

In this section, we introduce our proposed event aggregation method. Before describing the proposed method, it is required to explain an abstract schema of a heterogeneous event aggregation framework to understand our proposed method's position in the whole structure. The abstract schema is shown in Figure 7, which consists of the five principal layers as follows:

- **Hardware (Layer 1):** in a monitored network, each of the heterogeneous sensors can be in the form of a dedicated physical machine or a virtual machine configured and managed by security staff for logging events.
- **Heterogeneous Events Logging (Layer 2):** in this layer, low-level events are logged by the various sensors of the different detection levels (Network, Host, and Application) during the time. The logged events by heterogeneous sensors are collected by the Beats of the ELK stack. Afterward, the collected events are transferred to the OLF format and stored in Elasticsearch. Then, the normalized events are ingested to the upper layer in a streaming fashion for the related processing to the event aggregation analysis.
- **Event Preprocessing (Layer 3):** in this layer, after the collection of logged events from heterogeneous sensors (with a variety of event log formats regarding different types and vendors of a specific sensor), they are received by an event normalization component which converts them into a common event format by using a standard normalization format which is an essential task for later processing, i.e., aggregation analysis.
- **Event Processing (Layer 4):** this layer con-

tains a three-phase event aggregation process that receives normalized events in a specified time interval as input and, after event clustering, filtering, and summarizing processes on the received events, produces the final aggregated events as output.

- **Presentation (Layer 5):** in this layer, a set of aggregated events (without any redundant information) are reported to the security administrators based on queried data to give them a complete picture of the organization's security status.

After a brief explanation of the event preprocessing layer components in the abstract schema, in the rest of this section, a detailed description of the three main components of the proposed event aggregation method, namely, event aggregation, event filtration, and event summarization (event processing layer of abstract schema in Figure 7) are provided. It should be noted that during the description of the components, as needed, the events logged in the sample attack scenario presented in Table 6 are used for understanding the input, process, and output of the three components mentioned above. The workflow of the main components in the event processing layer is shown in Figure 8, described in the following subsections. In addition, a list of abbreviations and symbols is provided in Table 7 to make it easier and better to understand the contents of the algorithms, figures, formulas, and tables of this section and the following sections.

4.1 Event Aggregation Component

The event aggregation component works based on the clustering analysis technique, shown in the Algorithm 1. As illustrated in the algorithm, this component receives a set of various sensor types (each sensor has a unique sensor-ID), a predefined aggregation time window (ATW) for event streaming, a set of received heterogeneous events in the ATW, and a configuration file as input. The configuration file contains two types of event feature sets for each sensor, namely, the non-summarizable feature set $NSFS$ (more critical features of a unique sensor used in the event aggregation component) and the summarizable feature set SFS (less critical features of a unique sensor used in the event summarization component, Section 4.2.1) and a time window length (TWL) of each sensor for aggregation analysis. After the aggregation process, the algorithm generates aggregated events as output and sends them to the following process. In the event aggregation component, after receiving an event set in an ATW and retrieving the parameters from the configuration file (Lines 1-2), the heterogeneous events are classified into distinct groups based on their sensor-ID (Line 3). Then, the related event groups to sensor-ID from heterogeneous sensor sets are sorted based on the event timestamp (Lines 4-7). Afterward, based on the $NSFS$ of the sensor-ID, a set of aggregation rules is created (Lines 8-9). The aggregation process uses the aggregation rules regarding the TWL of the sensor-ID. The basic philosophy of this process is that if two events of a unique sensor have the same value for the $NSFS$ while being close in time regarding the TWL, then they can be aggregated to create an event cluster. The candidate $NSFS$ and SFS of each sensor mapped to the seven Bryant IKC stages are shown in Figure 18 in Section 6.

For the aggregation of each sensor event using aggregation rules, at first, they are classified into different event classes according to their event type or event id (EID) (feature 8 and 9 in Table 5) (Line 12). For each event class, an array of events with a unique event type is created, containing a set of ordered events that are similar in event type (Lines 13-14). In other words, one array of events is created for each event type. Then, similar events of the array are grouped within a cluster that inherits the $NSFS$ of the corresponding aggregation rules. A forward scan is enough for each array to accomplish the needed event clustering. By using two indices (base and current), each event (base event) is compared to its following events (current events) in the array. If they have similar feature values according to all aggregation rules related to a specified sensor and their time difference to the base event is lower than the TWL of the sensor-ID, the events are grouped into a unique cluster (Lines 15-33). The value of TWL is specified according to the speed of the event generation for each of the heterogeneous sensors. Afterward, those events in the TWL that satisfy aggregation rules are joined to the base event (Lines 23-26). After detecting two similar events, the number of the base event is incremented by 1, the current event is added to the generated cluster, and the value of the current index is set to 0, which means the event has been analyzed. After carrying out the mentioned process for distinct event types of all heterogeneous sensors, the aggregated events in the form of different event clusters (Lines 34-35) with different members are passed to the next event cluster filtration component to detect and filter noisy event clusters.

The output of the aggregation component for some logged events in Table 6 is illustrated in Figure 9. For the sake of simplicity, assume that the NIDS and Firewall events from the Network level (e_1 to e_9) and the Host OS events from the Host level are considered for the aggregation analysis. Also, it is assumed that after the event collection process, the logged events are normalized and sent to a central correlation system, including the aggregation process. According to Table 6, the NIDS, Firewall, and Host

Algorithm 1 *Event_Aggregation*(SS, ES, ATW, CF)

Input:

SS, a heterogeneous sensor set with a unique sensor-ID for each sensor type
ATW, aggregation time window for event streaming
ES, an event set received in the ATW
CF, a configuration file containing various sensors features information i.e. *NSFS* and *SFS* for each sensor and the aggregation-related threshold values

Output:

ECS, event clusters set

```
Get ES based on ATW size
Read CF
Classify heterogeneous logged events based on each sensor-ID s
Foreach (sensor-ID s in SS) do
    Get TWL of the s
    Get events of the s from the ES as s-es
    Sort events in s-es based on timestamp
    Get NSFS (f_1 − f_{j−1}) from the CF for the s
    Get a set of aggregation rule (aggrule) based on NSFS (f_1 − f_{j−1}) as s-aggrules
    Get ϵ from the CF
    Forall (event e in s-es) do
        Create an EID set from the EIDs of events in s-es as eid-s
        Foreach (EID in eid-s) do
            Create an array of events[] containing EID
            base=0
            While (base < events.size) do
                If (events[base].number > 0) Then
                    current = base + 1
                    While ((current < events.size) ∧ (timediff (base, current) < ϵ)) do
                        Foreach (aggrule in s-aggrules) do
                            eventsim[] = checksim(base, current) //If base = current, checksim returns 1
                        EndForeach
                        If (eventsim = 1)
                            events[base].number = events[base].number + 1
                            events[base] = events[base] + events[current]
                            events[current].number = 0
                        Else
                            current = current +1
                        EndIf
                    EndWhile
                Else
                    base = base + 1
                EndIf
                ECS = events
                Return ECS
            EndWhile
        EndForeach
    EndForall
EndForeach
EndFunction
```

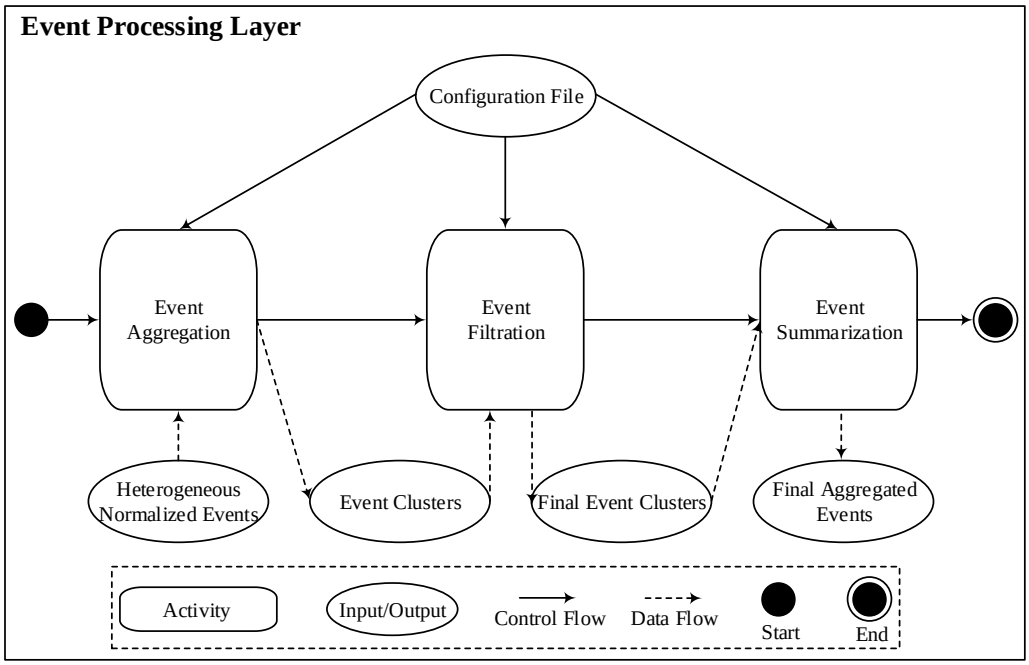


Figure 8. The workflow of the main components in the proposed event aggregation method

OS sensors have logged 5 (red color), 4 (yellow color), and 3 (green color) events, respectively.

Regarding Algorithm 1, logged low-level events by the mentioned sensors are received by the event aggregation component (part (a) in Figure 9). Based on the three SIDs in Table 6, the low-level events are categorized into three different event sets, NIDS events (IDs e_1, e_3, e_5, e_6, and e_9), Firewall events (IDs e_2, e_4, e_7, and e_8), and Host OS events (IDs e_{10}, e_{11}, and e_{12}) (part (b) in Figure 9). As mentioned before, some aggregation rules are used to cluster the low-level events according to the *NSF* set of a unique sensor type. Based on the provided information on the *NSF* of the three sensors in Table 8, the related aggregation rules are shown in Table 9 (for any two subsequent events e_i and e_j).

So, based on the execution of the Algorithm 1 on the three separated event sets, it could be inferred that the NIDS events can be clustered into two event clusters, C_0 (e_1, e_3, and e_5) and C_1 (e_6 and e_9), regarding aggregation rules with Rule ID 11 (with TWL=60 seconds), 12, and 13 of Table 9. For the Firewall events, the logged low-level events are also clustered into three event clusters, C_2 (e_2 and e_4), C_3 (e_7), and C_4 (e_8), according to the aggregation rules with Rule IDs 21 (with TWL=60 seconds), 22 to 25 of Table 9. In addition, regarding Host OS events, there are two distinct event clusters, C_5 (e_{10}) and C_6 (e_{11} and e_{12}), regarding aggregation rules with Rule IDs 31 (with TWL=300 seconds), 32 to 37 of Table 9. The event aggregation analysis and also the resulting event clusters are depicted in Figure 9 from part (c) to part (g).

4.2 Event Filtration Component

After generating event clusters from low-level events, it is time to identify significant events within a large set of logged events and filter noisy ones because they are false-positive and irrelevant. These events are generated due to many causes like misconfiguration, low accuracy of applied detection methods, and lack of attention to contextual information of target during the event generation [13]. In the literature, many techniques exist for event verification and filtration [12, 13]. One of the most important ones is using outlier detection-based approaches [53]. In this paper, we use local density cluster-based outlier factor (LDCOF)[54], an extension of clustering-based local outlier factor (CBLOF)[55]. By leveraging this metric, the system can detect noisy event clusters and filter them. A noisy event cluster is a cluster in which events are recognized as those belonging to none of the event clusters. Detecting and filtering noisy clusters is helpful to have a set of high-quality real event clusters for the next processing step. The procedure of the event cluster filtration algorithm is depicted in the Algorithm 2.

Regarding Algorithm 2, for detecting outlier event clusters, the component receives generated event clusters set (ECS) by the previous component and the configuration file as input. It generates a set of aggregated events for each sensor. After receiving ECS,

Table 7. The list of abbreviations and symbols used in this section and the following sections

(a) Abbreviations

Abbreviation	Definition
AES	Aggregated event set
AOI	Attribute-oriented induction
APC	Aggregation performance curve
ATW	Aggregation time window
CBLOF	Clustering-based local outlier factor
CF	Configuration file
CQ	Cluster quality
DBI	Davies-Bouldin index
DI	Dunn index
EAR	Event aggregation ratio
ECS	Event clusters set
EID	Event ID
EPR	Event processing ratio
EPS	Event per second
IC	Information content
ILR	Information loss ratio
LC	Large event cluster
LCA	Least common ancestor
LDCOF	Local density cluster-based outlier factor
NIDS	Network-based intrusion detection system
NSF	Non-summarizable feature
NSFS	Non-summarizable feature set
OS	Operating system
SC	Small event cluster
SES	Summarized event set
SF	Summarizable feature
SFS	Summarizable feature set
SS	A heterogeneous sensor set
TCP	Transmission control protocol
TTL	Time to live
TWL	Time window length
UDP	User datagram protocol

(b) Symbols

Symbol	Description
$Sensor-ID/SID$	A unique ID of a heterogeneous sensor
S/s	Sensor-ID
e	A unique logged event
$s-es$	An event set logged by sensor with sensor-ID s
fj	j^{th} feature of a logged event e
$aggrule$	Aggregation rule for a unique sensor-ID
$s-aggrules$	A set of aggregation rules for a sensor-ID s
ε	Create-time difference threshold
α	Parameter for calculation of SC and LC
β	Parameter for calculation of SC and LC
Ci	i^{th} event cluster
C	Event cluster set
ae	Aggregated event
$eid-s$	A set of event ID
$events[]$	A set of events with a unique EID logged by a unique SID
$events.size$	Size of events [] array containing events with same EID
$base$	Pointer of the base logged event in array
$current$	Pointer of the current logged event in array
D	Dataset
ei	i^{th} logged event
c	A unique concept with the related concept tree
nc	The number of events in cluster
$diam(ci)$	Diameter of event cluster ci
$EAR_{\max}$	The maximum rate of EAR
$EPR_{\max}$	The maximum rate of EPR
$srcIP$	Source IP address
$dstIP$	Destination IP address
$srcPort$	Source port number
$dstPort$	Destination port number
$compName$	Computer name
$usrAccount$	User account name
$parProcessID$	Parent process ID

Table 8. Sample sensors features classification into *NSF* and *SF*

Sensor-ID \ Feature Name		Date Time	Source IP Address	Source Port	Destination IP Address	Destination Port	Event ID	TTL	Protocol	Computer Name	User Account	File Name	File Path	Process ID	Parent Process ID
۱	Feature Type	NSF	NSF	SF	NSF	SF	SF	SF	SF	–	–	–	–	–	–
	Threshold Value	–	–	1	–	1	2	1	1	–	–	–	–	–	–
۲	Feature Type	NSF	NSF	NSF	NSF	NSF	SF	SF	SF	–	–	–	–	–	–
	Threshold Value	–	–	–	–	–	2	1	1	–	–	–	–	–	–
۱۱	Feature Type	NSF	NSF	–	NSF	–	SF	–	–	NSF	NSF	NSF	SF	SF	NSF
	Threshold Value	–	–	–	–	–	2	–	–	–	–	–	3	3	–

they are first sorted based on the number of events in each cluster, as shown in the figure (Lines 1-2). By considering the two numeric parameters α and β (Line 3), the generated clusters are divided into two sets, large clusters (LC) and small clusters (SC), regarding the size of clusters (Line 4). Suppose $|C_1|>|C_2|>|C_3|>\cdots>|C_k|$, then b is the boundary of a cluster, small or large, such that $|C_1|+|C_2|+\cdots+|C_b|\geq|D|*\alpha$ or $|C_b|/|C_{b+1}|\geq\beta$ where D is the whole dataset. It should be noted that α and β parameters are set so that the following conditions are met, 1) most of the events in the dataset are not outliers, and 2) LC and SC should be significantly different in size. Hence, $LC=\{C_i|i\leq b\}$ and $SC=\{C_j|j>b\}$. After identifying LC and SC clusters, each event of a cluster is assigned with the LDCOF factor to evaluate whether the event cluster is noise or not (Lines 6-13). The related equation to the factor is shown in Equation 1 where the average used distance of cluster in this equation is computed based on Equation 2. This factor is calculated according to the size of the cluster and the distance between the target event and its nearby

Table 9. Sample aggregation rules for the sensors (a-aggrules in Line 20 in Algorithm 1) NIDS, Firewall, and Host OS sensors

Sensor-ID (Name)	NSF Name	Aggregation Rule (aggrule) ID	Aggregation Rule (aggrule in Line 20 in Algorithm 1)
1 (NIDS)	Date Time	11	If ((e_j.dateTime – e_i.dateTime) ¡ ε)
	Source IP Address	12	If (e_i.srcIPAddress = e_j.srcIPAddress)
	Destination IP Address	13	If (e_i.dstIPAddress = e_j.dstIPAddress)
2 (Firewall)	Date Time	21	If ((e_j.dateTime – e_i.dateTime) ¡ ε)
	Source IP Address	22	If (e_i.srcIPAddress = e_j.srcIPAddress)
	Source Port	23	If (e_i.srcPort = e_j.srcPort)
	Destination IP Address	24	If (e_i.dstIPAddress = e_j.dstIPAddress)
	Destination Port	25	If (e_i.dstPort = e_j.dstPort)
3 (Host OS Log)	Date Time	31	If ((e_j.dateTime – e_i.dateTime) ¡ ε)
	Source IP Address	32	If (e_i.srcIPAddress = e_j.srcIPAddress)
	Destination IP Address	33	If (e_i.dstIPAddress = e_j.dstIPAddress)
	Computer Name	34	If (e_i.compName = e_j.compName)
	User Account	35	If (e_i.usrAccount = e_j.usrAccount)
	File Name	36	If (e_i.fileName = e_j.fileName)
	Parent Process ID	37	If (e_i.parProcessID = e_j.parProcessID)

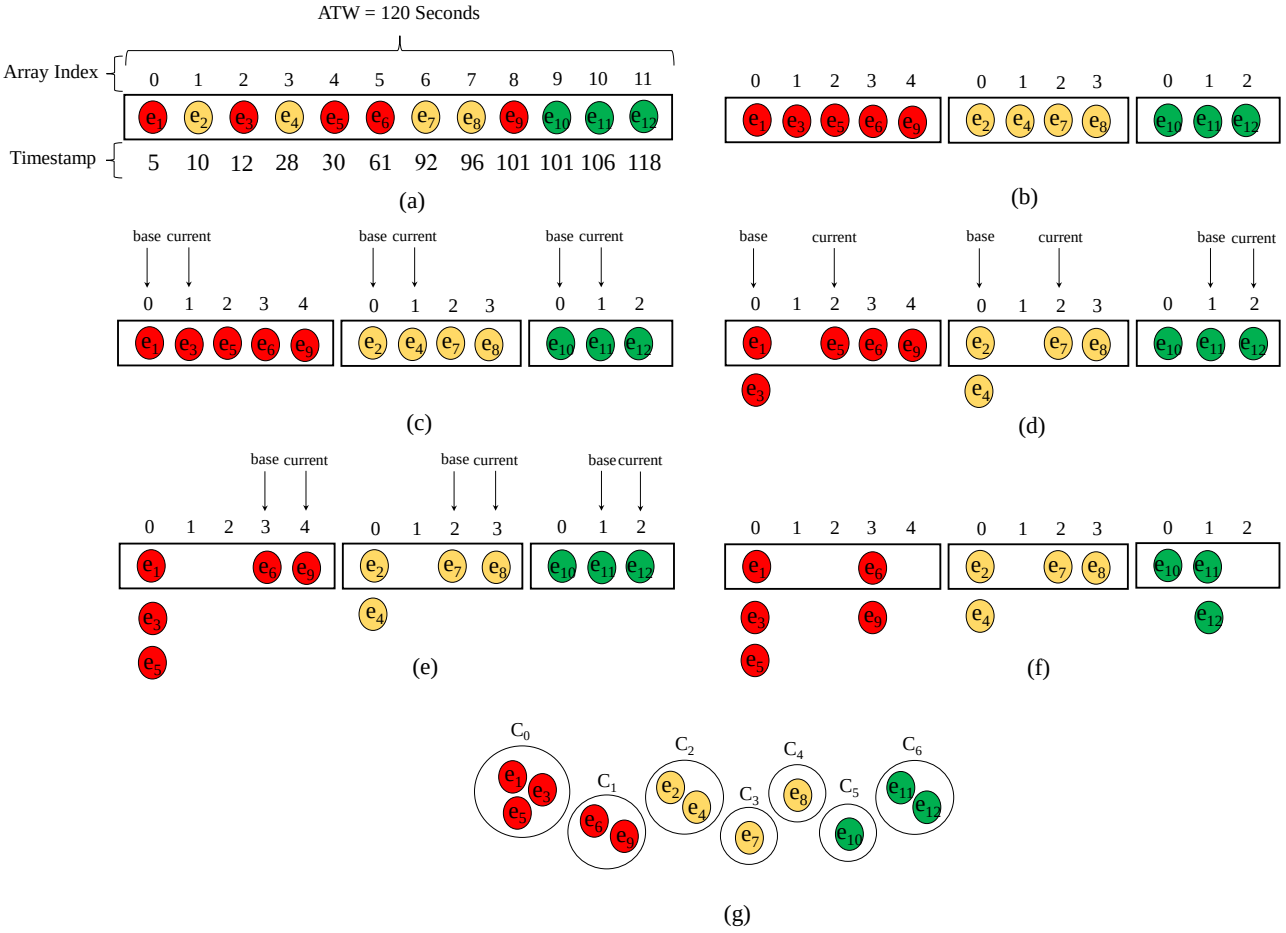


Figure 9. Sample output of event aggregation component, (a) low-level event stream, (b) grouped events, (c)-(f) event aggregation (ATW=120 Sec., TWL_{NIDS}=$TWL_{Firewall}$=60 Sec., and TWL_{HostOS}=120 Sec.), and (g) resulted event clusters

cluster. The rationale behind this is that when small clusters are considered outlying (noisy), the events within the *SC* are assigned to the nearest *LC*, which becomes its local neighborhood. Thus, the events in

SC are all outliers when compared with those in *LC* and discards from the *ECS*. Finally, the remaining event clusters are passed to the next component as the final aggregated event set (*AES*) (Lines 14-15). In our example, event clusters C_0, C_1, C_2, and C_6 identified as large clusters, while C_3, C_4, and C_5 are considered to be small clusters. Thus, by setting $\alpha = 0.75$ and $\beta = 1$, it can be inferred that the events in clusters C_3 and C_5 are outliers and can be discarded.

$$LDCOF(e) = \begin{cases} \frac{min(d(e,C_j))}{distance_{avg}(C_j)}, & if\ e \in C_i \in SC\ where\ C_j \in \text{LC}, \\ \frac{d(e,C_i)}{distance_{avg}(C_i)}, & if\ e \in C_i \in \text{LC} \end{cases} \tag{1}$$

$$distance_{avg}(C) = \frac{\sum_{i \in C} d(i, C)}{|C|} \tag{2}$$

4.2.1 Event Summarization Component

After completion of the aggregation component, the resulting event clusters are injected into the summarization component to find duplicated events and eliminate redundant ones. The procedure of the event summarization component is shown in the Algorithm 3. Based on the figure, this component receives the sensor-ID of sensor, a set of clustered events with the same sensor-ID, and the configuration file as input and returns a set of summarized events as output. Besides the information mentioned above about the configuration file, it also contains concept trees of the different sensors *SFS* used for the summarization component based on an AOI, which gives extra flexibility over traditional machine learning techniques for data fusion. According to the [56, 57], a concept tree describes the abstraction relationship (i.e., generalization/specialization) between similar concepts using a hierarchical structure. In the concept tree of an attribute, the root node contains the most abstract form of the concept, and the intermediate and leaves nodes represent refined concepts and instances, respectively.

In the summarization component, for each summarizable feature (*SF*), there is a unique concept tree in that the feature name is the root node. At the same time, the attribute values correspond to the tree's leaves. Some examples of the concept trees for different event features are in Figure 10 and Figure 11. As mentioned in the Algorithm 3, for a given clustered event from the aggregation component, at first, the *SF* of the corresponding sensor type is obtained with the related threshold vector. The threshold vector of the sensor indicates the level of summarization process for each *SF* of it according to the depth of *SF* concept tree. In other words, based on the depth level of the concept tree, the event summarization method operates by the defined threshold vectors in a completely flexible manner. After retrieving the *SFS* and their associated threshold vectors, for each *SF* (Lines 1-3), the related concept tree of the feature is extracted from the configuration file (Line 4). The extracted tree shows the relationship between children and parents as a directed edge from child to parent. Then, the following tasks are repeated for each *SF* until nothing else is left in the *SFS* vector.

For the examined *SF*, at first, the value of the *total-distinct-values* is calculated based on the different values for the *SF* in the generated event cluster (Line 5). Then, the value is compared with the corresponding threshold value in the threshold vector (Line 6). As long as the *total-distinct-values* for a given *SF* is more significant than a predefined threshold value, the *SF* value of the events is replaced with the value of its parent for all events in the event cluster; otherwise, it is kept (Lines 7-13). At the end of each repetition of the generalization loop, the *total-distinct-values* of the *SF* is calculated again for the last comparison (Line 14). In addition, if some duplicated events are in the generalized event set, similar events are fused into a unique summarized event, and duplicated ones are deleted (Lines 15-20). It should be noted that if the value of *total-distinct-values* is still more significant than the defined threshold value, the generalization process continues. This process will continue until all features in the *SFS* of the sensor have been checked and no other features are left. The resulting non-repetitive events of the final round are reported as the output of the summarization component (final aggregated events) (Line 23).

Regarding our example, after event aggregation and outlier event filtration processes, five event clusters, including ten different events, are injected into the event summarization component. As mentioned in the Algorithm 3, for the event summarization process, it is required to define the concept tree of the features in the *SF* set of a specified sensor. As seen in Table 8, the *SF* set of the NIDS and firewall sensors are {Source Port, Destination Port, Event ID, TTL, Protocol} with threshold vector {1, 1, 2, 1, 1} and {Event ID, TTL, Protocol} with threshold vector {2, 1, 1}, respectively. Also, the *SF* set of the Host OS is {Event ID, File Path, Process ID} with threshold vector {2, 3, 3}. Based on the defined *SF* sets for the sensors in Table 8, Figure 10 and Figure 11 indicate their related concept trees. A running example of Algorithm 3 is shown in Figure 12 for event e_1 of the NIDS sensor.

Table 10 shows the final results of the event summarization component on the event clusters C_0, C_1, C_2, C_4, and C_6. According to the results of this component, the pairs of events (e_1, e_3), (e_2, e_4), (e_6, e_9), and (e_{11}, e_{12}) are summarized and formed the aggre-

Algorithm 2 *Event_Cluster_Filtration*(ECS, CF)

Input:

ECS, event clusters set

CF, a configuration file containing various sensors features information i.e. *NSFS* and *SFS* for each sensor and the aggregation-related threshold values

Output:

AES, a set of aggregated events

1: Get ECS //$ECS = \{C_1, C_2, C_3, \cdots, C_k\}$
2: Sort event clusters in ECS based on their size //$\|C_1\| \geq \|C_2\| \geq \|C_3\| \geq \cdots \geq \|C_k\|$
3: Get α and β parameters
4: Compute LC and SC based on the α and β parameters
5: $AES = \phi$
6: Foreach (event e in ECS) do
7: If (e belongs to $C_i \in SC$) Then
8: Compute $LDCOF$ score of e based on the Equation 1
9: $ECS = ECS$ - e
10: Else
11: Compute $LDCOF$ score of e based on the Equation 1
12: $AES = AES \cup e$
13: EndIf
14: $AES = ECS \cup AES$
15: Return AES
16: EndForeach
17: EndFunction

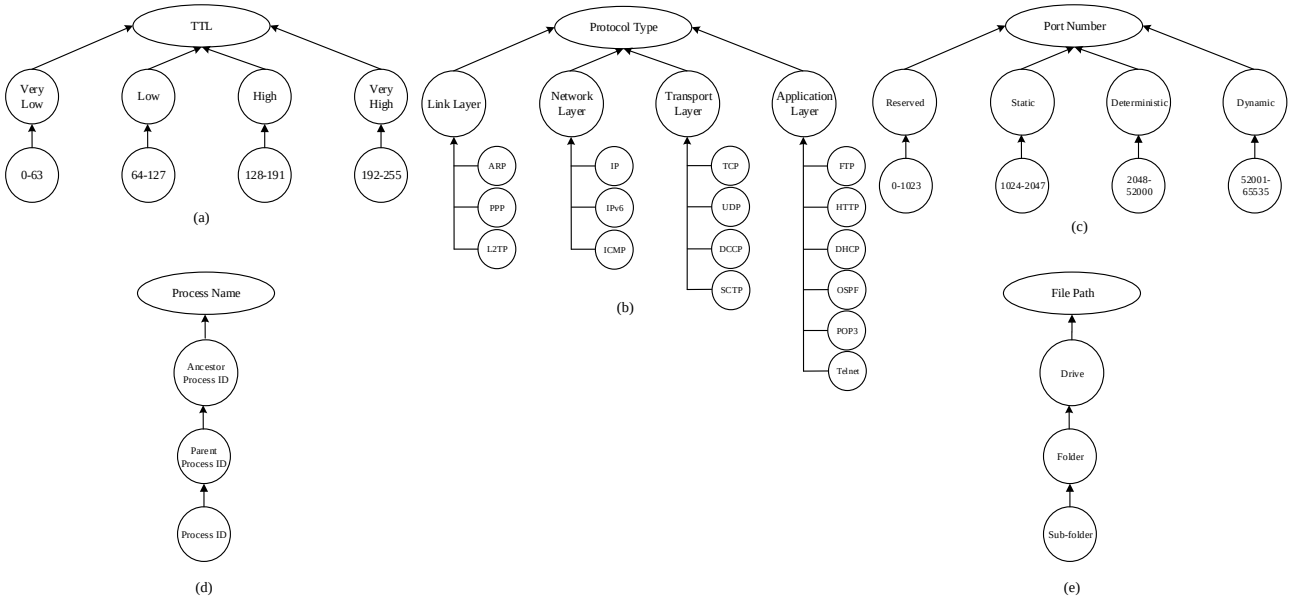


Figure 10. Sample concept trees for some *SF* features, (a) TTL, (b) Protocol Type, (c) Port Number, (d) Process Name, and (e) File Path

gated events ae_1, ae_2, ae_4, and ae_6. In addition, the two events e_5 and e_8 are not summarized with other events and remain in the output aggregated event set. For example, the analyzes performed by the event summarization component on the NIDS sensor event clusters, C_0 and C_1, are given below.

- The (e_1, e_3, and e_5) event cluster of the NIDS: regarding the four *SF* of the NIDS sensor and their related concept tree, the events of this cluster are summarized. For this purpose, at first, the Source Port and Destination Port values of the events are examined based on the Port Number concept tree (part (c) in Figure 11). In this step, regarding the Source Port values of the three events and their parent name in the related concept tree, the value of this feature for events e_1, e_3, and e_5 is changed to *Reserved*, *Reserved*, and

Algorithm 3 *Event_Summarization*(S, AES, CF)

Input:

S, a sensor with a unique sensor-ID

AES, aggregated event set with the same sensor-ID

CF, a configuration file containing various sensors features information i.e. $NSFS$ and SFS for each sensor and the aggregation-related threshold values

Output:

SES, summarized event set with the same sensor-ID

Get SFS $(f_i - f_n)$ of s from the CF
Foreach (feature f_j in SFS $(f_i - f_n)$ do
    Get $threshold\_numeric\_value$ for f_j
    Get Feature f_j concept tree hierarchy from CF
    $total\_distinct\_values$ = the number of distinct values of AES with f_j + $threshold\_numeric\_value$ for f_j
    While ($total\_distinct\_values > threshold\_numeric\_value\_f_j$) do
        Forall (ae in AES) do
            If (f_j value of ae has a parent in f_j concept tree) Then
                f_j value of ae = parent value
            Else
                f_j value of ae = f_j value of ae
            EndIf
        EndForall
        $total\_distinct\_values$ = the number of distinct values of AES with f_j
        If (duplicate ae exists in AES) Then
            Merge identical events into unique ae in AES
            SES = SES $\cup$ AES
        Else
            Break
        EndIf
    EndWhile
EndForeach
Return AES
EndFunction

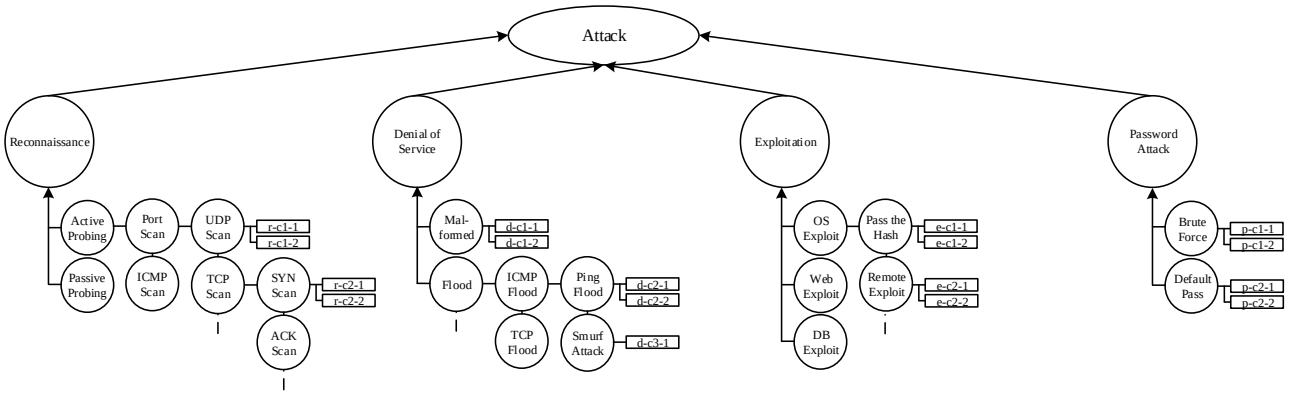


Figure 11. Sample concept tree for the Event ID feature

Deterministic, respectively. Since the threshold value of the Port Number is 1 (see Table 8), and there is no parent in the concept tree for further analysis, the summarization process is finished for the Port Number feature. Then, this analysis is done for the remained SF according to the concept trees of each of them. The results of the summarization process show that e_1 and e_3 are duplicated, and so on; one can be eliminated, but e_5 remains in the final event set.

- The (e_6 and e_9) event cluster of the NIDS: based on the analysis mentioned above for the previous event cluster and regarding the Source Port values of the two events and their parent name in the Port Number concept tree, the event values are changed to the *Deterministic* value. Since

f_j = Source Port ⟶ 1st iteration

Regarding part (c) in Fig. 9 (Threshold Value = 1)

Date Time	Source IP Address	Source Port	Destination IP Address	Destination Port	Event ID	TTL	Protocol
Feb 25 12:11:05	172.25.110.11	Reserved	192.168.1.12	20	r-c1-1	63	UDP

f_j = Destination Port ⟶ 1st iteration

Regarding part (c) in Fig. 9 (Threshold Value = 1)

Date Time	Source IP Address	Source Port	Destination IP Address	Destination Port	Event ID	TTL	Protocol
Feb 25 12:11:05	172.25.110.11	Reserved	192.168.1.12	Reserved	r-c1-1	63	UDP

f_j = Event ID ⟶ 1st iteration

Regarding Fig. 10 (Threshold Value = 2)

Date Time	Source IP Address	Source Port	Destination IP Address	Destination Port	Event ID	TTL	Protocol
Feb 25 12:11:05	172.25.110.11	Reserved	192.168.1.12	Reserved	UDP Scan	63	UDP

f_j = Event ID ⟶ 2nd iteration

Regarding Fig. 10 (Threshold Value = 2)

Date Time	Source IP Address	Source Port	Destination IP Address	Destination Port	Event ID	TTL	Protocol
Feb 25 12:11:05	172.25.110.11	Reserved	192.168.1.12	Reserved	Port Scan	63	UDP

f_j = TTL ⟶ 1st iteration

Regarding part (a) in Fig. 9 (Threshold Value = 1)

Date Time	Source IP Address	Source Port	Destination IP Address	Destination Port	Event ID	TTL	Protocol
Feb 25 12:11:05	172.25.110.11	Reserved	192.168.1.12	Reserved	Port Scan	Very Low	UDP

f_j = Protocol Type ⟶ 1st iteration

Regarding part (b) in Fig. 9 (Threshold Value = 1)

Date Time	Source IP Address	Source Port	Destination IP Address	Destination Port	Event ID	TTL	Protocol
Feb 25 12:11:05	172.25.110.11	Reserved	192.168.1.12	Reserved	Port Scan	Very Low	Transport Layer

Figure 12. Operation of the Algorithm 3 on the NIDS sensor events in our example

the threshold value of the Port Number is 1 (see Table 8) and the identical values, Deterministic, for the feature Source Port, the summarization process is finished for the Port Number feature. After the summarization process for the next *SF* features, the results show that the Destination Port, Event ID, TTL, and Protocol are equaled to *Reserved*, *TCP Scan*, *Very High*, and *Transport Layer*, respectively for the two events. Therefore, based on the summarized events, it could be inferred that e_6 and e_9 are redundant, and so on; one can be deleted.

5 Evaluation and Discussion

After explaining the proposed event aggregation method in the previous section, the evaluation method and the results of the experiments are described in this section. In the rest of this section, at first, the experimental results of the simulation and numerical analysis are reported in Section 5.1 by applying the proposed method on events of standard datasets. In addition, Section 5.2 provides some brief analytical discussion about the proposed method and its effects on the area of complex IKC-based APT attack scenarios detection and its limitations.

5.1 Simulation and Numerical Experiments

For the evaluation purposes of the proposed event aggregation and summarization method, a study has been conducted on the existing datasets in the area of intrusion detection. A comparative analysis of the datasets is provided in Table 16, in Section 6. Regarding Table 16, the three more relevant and publicly available datasets are chosen for the evaluation, namely, SotM34 [58], Bryant [7], and LANL [59]. It should be noted that all three mentioned datasets contain a set of heterogeneous events logged by different sensors of the three mentioned detection levels (Network, Host, and Application).

5.1.1 Datasets

The three chosen datasets for the simulation and numerical evaluation of the proposed event aggregation method are briefly explained from different aspects, i.e., including heterogeneous sensors and event size.

- **SotM34:** In the area of intrusion detection, one of the primary multi-source datasets is SotM34 [58], which creates by the netForensics Honeynet team during the Honeynet Project, Scan of the Month challenge. The dataset contains various heterogeneous events from three different security sensors, namely, NIDS (Snort), Firewall (IPTables), Host OS (Linux Syslog), and the only non-security sensor, Web Server (Apache). Different sensors in a honeypot system log the events. According to the SotM34 simulation environment, three central systems are used for recording the Honeynet activities, namely, bridge, bastion, and combo, for about four weeks (from 22 February 2005 to 17 March 2005). The first system filters malicious connections, and the second runs a NIDS service based on the well-known Snort tool. The third system is also used for the victim, which contains multiple virtual IP addresses on a specified network range. In our evaluation, the low-level events logged by the different sensors (260337 individual events) are used for the experiments.

  Part of the SotM34 dataset events forms an IKC-based multi-stage attack scenario [60]. According to the attack scenario, after identifying the target network machines by performing reconnaissance scans, the attacker sends a backdoor to a Combo node of the target network. Then, after downloading the backdoor by an authorized user, the backdoor is successfully in-

Table 10. The resulted aggregated events in our example

SID	ID	Date Time	Source IP Address	Source Port	Destination IP Address	Destination Port	Event ID	TTL	Protocol	Computer Name	User Account	File Name	File Path	Process ID	Parent Process ID
1	ae1	Feb 25 12:11:05	172.25.110.11	Reserved	192.168.1.12	Reserved	Port Scan	Low	Transport Layer	–	–	–	–	–	–
2	ae2	Feb 25 12:11:10	172.25.110.11	2256	192.168.1.12	22	ICMP	Very Low	Network Layer	–	–	–	–	–	–
1	ae3	Feb 25 12:11:30	172.25.110.11	Deterministic	192.168.1.12	Reserved	TCP Scan	High	Transport Layer	–	–	–	–	–	–
1	ae4	Feb 25 12:11:01	172.25.110.11	Deterministic	192.168.1.12	Reserved	TCP Scan	Very High	Transport Layer	–	–	–	–	–	–
2	ae5	Feb 25 12:12:36	172.25.110.11	2256	192.168.1.13	20	f-c3-1	252	TCP	–	–	–	–	–	–
11	ae6	Feb 25 12:12:46	172.25.110.11	–	192.168.1.12	–	OS Exploit	–	–	Client1	John	mal.pdf	C	p1	p1

stalled and runs on the victim's machine. In this step, the attacker exploits a vulnerability of an installed program on the Combo machine of the topology called AWStats, an Apache web server monitoring tool, to install the backdoor. Then, the adversary connects to the backdoor and makes multiple outbound HTTP connections to a server on the Internet. The logged events in the last step show that this was an exfiltration attack using some searching commands for files on the target machine, the Combo server.

- **Bryant:** As mentioned before, one of the main IKC models in the area of IKC-based multi-step attacks is the Bryant kill chain model [7], which is described in detail in Section 3.1. Regarding this model, the inventors of this IKC model have released a new heterogeneous events dataset based on their proposed IKC model, which is accessible upon request. The Bryant dataset includes 5962 different attack-related events which are related to a sample IKC-based APT attack scenario. The detailed explanation of the attack scenario and network topology for the attack simulation is described in [7]. According to their network topology, all the events are logged by some heterogeneous sensors, namely, NIDS, Firewall, and Edge Router from the Network level, HIPS, Antivirus, Vulnerability Scanner, and Host OS from the Host level, and Domain Controller from the Application level.

  A custom APT attack scenario was created to stimulate all seven IKC stages across multiple machines, generating data from multiple detection sensors and ultimately resulting in successful data theft. Regarding Figure 6 in [7], an attacker located on an external network attempting to access the internal corporate network. The attacker performs initial probing activities to identify victim machines. Then, he/she sends a phishing email through the corporate mail server to trick a user into installing a legitimate program with a known vulnerability and a custom backdoor disguised as a patch to the vulnerable program. After installing the fake patch by the user, a remote shell is established for the attacker machine. Then, the attacker conducts privilege escalation and initiates a remote desktop session on the compromised machine. After identifying the internal IP addresses of the company DMZ servers, he/she tries to find the SQL database of the web server and email server and exfiltrate them from the internal corporate network.
- **LANL:** One of the other primary datasets containing multi-source cybersecurity events is released by Los Alamos National Laboratory called LANL dataset to citer65. This high-volume dataset (about 10.7 GB) is publicly available and includes events of five different sensors, namely, NetFlow, Host OS, Audit Logs, and DNS Server, with 1,648,275,307 events. This dataset contains the normal behaviors of 12425 users, and 117684 computers of the LANL company that monitored the network for 58 consecutive days. Besides logging the users' normal activities, the dataset has a set of events for a simulated attack by a Red Team. The log files with the related event features are described in the [59]. In our experiments, except for the DNS events, only part of the other sensor events (a quarter of the total) is used for the evaluation due to the large volume of events and the computational resource constraints.

  The logged events in the LANL dataset are related to the malicious activities carried out by the Red Team during the IKC stages of the APT attack. Based on the activities carried out in this attack, the attackers initially identified the target organization's network to find exploitable vulnerabilities in the target network's assets. The attackers then tried to install malware on the victim network by sending phishing emails with an infected link to the target network workstations. After downloading and installing a malicious payload on a victim machine, the malware tries to monitor the connections established with the network servers and obtain information about the programs installed on them and the information of its users. In

addition, by communicating malware with remote C&C servers in the external network, the attacker tried to expand user privileges on the infected workstation for lateral movement activities. After migrating to the desired systems, which are generally the vital servers of the organization, the classified information is extracted and collected in the form of documents to be finally sent to the external network.

5.1.2 Evaluation Metrics

Besides the functional tests of the method, the other aspect of evaluation is related to the performance tests. According to the literature review, there are three main evaluation metrics for aggregation analysis which are as follows:

Event aggregation ratio (EAR): this metric shows the power of an event aggregation method in reducing the volume of logged events by the sensors. Based on Equation 3, this metric is calculated by dividing the number of aggregated events by the total events in a specified ATW. The greater the value of this parameter, the greater the event aggregation rate and more event reduction. For example, in our provided example, the EAR is 50% (total events = 12 (Table 6) and aggregated events = 6 (Table 10)).

$$EAR = \left(1 - \frac{\#\ of\ Summarized\ Events}{\#\ of\ Total\ Events}\right) * 100 \tag{3}$$

Event processing ratio (EPR): The EPR metric indicates the speed of the event aggregation process which is sometimes referred to as throughput. Based on Equation 4, this metric is obtained by dividing the number of processed events by the processing time (in seconds). The greater the value of this parameter, the greater the processing power of the aggregation component. For example, if the proposed method process 12 events of our example in 4 seconds, the EPR equals three events/second.

$$EPR = \left(1 - \frac{\#\ of\ Processed\ Events}{Processing\ Time\ (in\ Seconds)}\right) \tag{4}$$

Information loss ratio (ILR): Although in most related works, only the EPR is used to evaluate the aggregation process, this is a volume metric and is not sufficient for quality evaluation of the aggregation method. The basic idea of the ILR metric is to measure the amount of security-relevant data loss during the aggregation and summarization process. As mentioned, in the proposed aggregation and summarization method, two concepts that belong to the same concept tree are summarized by replacing them with their least common ancestor (LCA) from the corresponding concept tree. This causes to occur a loss of security information. In this paper, to measure information loss by replacing the parent concept with its child concept, Equation 5 is used, which is borrowed from the [26], the nearest work to our method. The ILR is a value in [0, 1] where 0 means no information loss and 1 indicates 100% information loss. The ILR metric is highly dependent on the configuration of the security administrator.

$$ILR(C) = \frac{\sum_{c \in C} \left(IC(c) - IC(LCA(C))\right)}{\sum_{c \in C} IC(c)} \tag{5}$$

Based on this equation, parameter C is a set of given concepts. Also, for each of the concepts in an event type (indicated by the type of sensor), the information content (IC) of concept c is calculated by using Equation 6, which is used to measure the amount of information served by the c concept.

$$IC(c) = -log\left(\frac{\frac{|leaves(c)|}{|subsumers(c)|} + 1}{maxleaves + 1}\right) \tag{6}$$

The value of this metric depends directly on the threshold vector values of the *SFS* features set. For example, assume that we want to compute the ILR rate by aggregating the ICMP Flood's two sub-classes, namely, Ping Flood and Smurf Attack (Figure 11). According to the Equation 6, we have the following IC values, IC(Ping Flood) = 1.08, IC(Smurf Attack) = 1.16, and IC(ICMP Flood) = 1.03. As illustrated in Figure 11, since the LCA of the Ping Flood and Smurf Attack is ICMP Flood, the ILR rate is equal to ILR(Ping Flood, Smurf Attack) = 0.08 based on Equation 5.

Since, in our proposed aggregation method, clustering analysis based on the *NSFS* of events plays a vital role in the method, it is required to measure the quality of event clusters to find how similar the events are within a cluster and how dissimilar with the events in the other event clusters. There are many internal cluster validity metrics presented in the area of cluster quality (CQ) to evaluate the event clustering task [61, 62]. In this paper, two related metrics are used for evaluating the clustering analysis of the event aggregation method, which are as follows:

Dunn index (DI): this metric is an internal cluster validity metric which defines as a ratio of the minimum distance between clusters to the maximum cluster diameter [61, 62]. This metric aims to determine the dense and well-separated cluster designed for non-overlapping event clusters. This metric for a specific number of clusters is calculated using Equation 7. The details of this equation are provided in [61]. According

to this metric, the larger the DI value (using Euclidian distance), the better the clustering.

$$DI_{n_c} = min_{i=1,\cdots,n_c}\left\{min_{j=i+1,\cdots,n_c}\left(\frac{d(c_i,c_j)}{max_{k=1,\cdots,n_c}diam(c_k)}\right)\right\} \quad (7)$$

Davies–Bouldin index (DBI): the other internal cluster validity metric is DBI that measures the output of a clustering algorithm in terms of minimal intra-cluster variances and maximal inter-cluster distances. This metric for a specific number of clusters is calculated using Equation 8. The details of this equation are provided in [62]. According to this metric, the smaller the DBI value (using Euclidian distance), the better the clustering. In other words, the clusters should have the minimum possible similarity.

$$DBI_{n_c} = \frac{1}{n_c}\sum_{i=1}^{n_c} max_{j\neq i}(DBI_{ij}) \quad (8)$$

5.1.3 Experimental Setup

Using the EPR evaluation metric, the performance of the proposed aggregation method has also been measured. In other words, the system's performance is in terms of execution times. For this purpose, we must to present the experimental setup of our evaluations. The proposed method has been developed in Java, and the experiments have been executed on a Windows 10-based machine with the Intel Core i7 CPU (4 MB Cache and 3.2 GHz), 16 GB of RAM, and 2 TB Hard Disk.

5.1.4 Experimental Results on the Datasets

In this section, we present the experimental evaluation of the proposed event aggregation method according to the three chosen datasets, namely, SotM34, Bryant, and LANL. Regarding our aggregation method, there is a need for *SFS* features set threshold vectors of the used sensors in each dataset for the event summarization component. Depending on the concept trees related to the *SFS* features of each sensor in the dataset, several vectors can be defined for the threshold values. In our experiments, different threshold vectors were manually tuned and used for various sensors, the most suitable of which was selected. Table 17 in Section 6 shows the threshold vectors of various sensors of each dataset for the experiments. The *SF* Numbers in the last column of Table 17) originates from Table 5. The used concept tree for the event type feature in the experiments is depicted in Figure 19 in Section 6.

According to the provided information in Table 17, the evaluation results for each of the three mentioned datasets are provided in Table 11. As can be seen in the table, for each sensor of the datasets, a specified value is determined for the time window length (*TWL*) regarding the suggested time intervals in [63] to focus on the sensors for detecting and preventing malicious activities of the attackers. After injection of the low-level events in the datasets to the proposed method with a length of 3600 seconds for the ATW parameter, some statistics related to each component of the method are reported in Table 11. In addition, the results of the performance mentioned above evaluation metrics are also provided in this table. In addition, for filtering noisy event clusters of SotM34, Bryant, and LANL datasets in the experiments, the α parameter is set to 90%, 90%, and 80%, respectively, an optimal value regarding each of the datasets.

Generally, regarding the various EAR and their corresponding ILR values, the experiments have proved that the proposed aggregation method is a general solution that reduces a significant event set with many redundant events into a set of summarized events with the least amount of security information loss. For example, the aggregation method is able to reduce significant amount of events for some Network level sensors, such as NIDS, Firewall, or Netflow, to nearly 99% in some circumstances (when most logged events are duplicated) with a reasonable level of ILR. Regarding the low value of the EPR for the Bryant dataset in Table 11, it should be noted that the proposed system requires an initial startup time, which is also included in the Execution Time. Therefore, the time required for initial setup and, on the other hand, the small number of events of each sensor in the dataset has caused the value of the EPR to be reduced. In other words, the system has not reached a steady state.

As mentioned earlier, the ILR metric is highly dependent on the configuration of the security administrator. In other words, by adjusting the threshold vector for SF features of any sensor, the ILR will be controllable. The security administrator can determine to what extent the lack of security information is allowed and does not affect subsequent operations. We know that the EAR and ILR metrics affect each other and, changing one causes the other to change the value. It is clear that when the EAR value increases for a specific dataset sensor, the value of the ILR becomes constant or decreases. In general, when a certain amount of changes in EAR is not affected as much as the ILR, the changes in the ILR are relatively acceptable. Hence, the results of the event aggregation will be valid and informative for the next event processing steps. According to Table 11, the results indicate that this is the case for most sensors of the used datasets; despite the changes of more than 50% in the EAR, the ILR has seen significantly fewer changes.

Moreover, according to the main goal of the paper, reducing the volume of logged events during the

Table 11. Detailed performance analysis of our proposed aggregation method based on the three standard datasets

	Sensor Type	Aggregation and Summarization						Evaluation Metrics				
		Total Events	TWL (Sec)	Event Clusters	Filtered Clusters	Summarized Events	Execution Time (Sec)	DI	DBI	EAR	EPR	ILR
SotM34	NIDS (Snort)	69,039	60	2,182	136	26,345	395	0.87	0.12	62%	174.7	0.29
	Firewall (IPTables)	179,752	60	9,865	1,120	46,747	1,125	0.79	0.19	73%	159.7	0.32
	Host OS (Linux Syslog)	3,925	300	420	95	1,836	22	0.72	0.26	54%	178.4	0.15
	Web Server (Apache)	7,621	300	1,121	49	4,421	49	0.65	0.28	42%	155.5	0.21
Bryant	NIDS (Snort)	1,589	60	86	16	17	18	0.91	0.11	99%	23.8	0.06
	Firewall (pFsense)	425	60	32	11	68	6	0.84	0.15	87%	21.8	0.28
	HIPS (McAfee)	267	300	27	13	51	7	0.81	0.24	81%	19.28	0.2
	Antivirus (McAfee)	25	300	5	1	5	5	0.78	0.3	80%	20.6	0.17
	Host OS	1,488	300	112	31	551	89	0.72	0.35	63%	19.8	0.26
	Web Server (IIS)	636	300	19	7	268	3	0.84	0.18	58%	18.6	0.11
	Mail Server (Microsoft Exchange)	344	300	11	3	124	164	0.68	0.38	64%	25.3	0.22
	Domain Controller	1,188	60	185	49	437	48	0.75	0.29	61%	21.8	0.25
LANL	NetFlow	32,494,353	600	213,607	61,213	7,031,615	259,955	0.99	0.13	79%	124.9	0.42
	Host OS (Authentication)	262,857,802	600	12,140,744	842,342	91,363,038	1,932,779	0.68	0.28	66%	135.9	0.33
	Audit Log	106,511,274	3600	614,647	152,214	37,137,051	700,733	0.72	0.34	66%	151.9	0.16
	DNS Server	40,821,591	3600	18,145,256	2,312,981	15,752,124	224,295	0.82	0.23	62%	181.9	0.29

IKC stages of APT attacks, another experiment is performed based on the logged events by the heterogeneous sensors during the IKC stage of the attack scenarios in all the tree used datasets (Section 5.1.1). The details of the experiment are provided in Table 12. Based on the results of the table, 1) to intersect each sensor with an IKC stage, a quadruple is presented, according to which the upper-left value is equal to the number of events associated with a specific IKC stage before performing the event aggregation and summarization processes, the upper-right value is equal to the number of events associated with the same IKC stage after the event aggregation and summarization processes, the lower-left value indicates the EAR metric, and the lower-right value represents the ILR metric and 2) some sensors do not produce an event for some of the IKC stages, which is indicated by a dash in the table.

Furthermore, a comparison analysis between the results of the proposed method and the related works (Section 2) is provided in Table 13. Regarding the results in the table, it can be found that:

- The obtained EAR for the events of various sensors is very close to the values reported in the literature that indicates combining aggregation and summarization operations has effectively reduced the volume of logged events. It should be noted that the metric EAR_{max} in the table is reported for the Snort NIDS sensor.
- Regarding the number of event features used in [18], our method has a relatively better rate for the EPR. However, in our method, in addition to the event aggregation and summarization time, the time for event normalization and filtration of noise events are also included in the reported execution time. The maximum EPR is achieved when all cores of the machine's processors perform only the event aggregation and summarization tasks (aggregator workers). It is worth noting that the high performance reported for the SEAS-MR method [25] is because this method is implemented and evaluated on a Hadoop big data cluster, which is beyond the scope of this paper.
- The proposed method can aggregate events with minimal loss of security information, resulting in a small number of high-quality cluster events as output that are of rich use to networks' administrators.

As mentioned, the EAR is one of the most important criteria for evaluating event aggregation approaches. Hence a comparative analysis based on this metric is presented in Table 14. In this analysis, the experiments are performed based on important sensor events from the SotM34 dataset. Regarding this table, the results of the proposed method are presented with similar methods, the implementation details of which are available. In addition, we believe that the EAR results alone do not correspond to the method's performance. Therefore other metrics, such as the EPR and the ILR, are considered for the analysis.

It should be noted that each experiment was repeated ten times, and the average value of the metrics for each method has been reported. Based on the results in the table, it can be seen that the proposed method has a relative advantage over the other methods based on the considered metrics. Except for the host-level sensors like Host OS, where the rate of the

Table 12. The effect of aggregation method on the events related to the various IKC stages in the three standard datasets

	IKC Stage / Sensor Type	REC		DEL		INS		ESC		LAT		ACT		EXF	
SotM34	NIDS (Snort)	8536	2018	3635	1235	–		–		–		–		–	
		76.36%	0.15	66.03%	0.17	–		–		–		–		–	
	Firewall (IPTables)	11339	3694	9859	2145	–		–		–		–		–	
		67.43%	0.11	78.25%	0.2	–		–		–		–		–	
	Host OS (Linux Syslog)	–		–		251	132	1028	624	422	297	734	321	–	
		–		–		47.42%	0.15	39.3%	0.21	29.63%	0.27	56.27%	0.14	–	
	Web Server (Apache)	–		–		–		–		–		–		3216	1745
		–		–		–		–		–		–		45.75%	0.18
Bryant	NIDS (Snort)	245	6	188	3	–		–		–		–		–	
		97.56%	0.09	98.41%	0.11	–		–		–		–		–	
	Firewall (pFsense)	132	14	84	12	–		–		–		–		–	
		89.4%	0.14	85.72%	0.12	–		–		–		–		–	
	Mail Server	–		185	23	–		–		–		–		–	
	(Microsoft Exchange)	–		87.57%	0.23	–		–		–		–		–	
	Host OS	–		–		145	69	212	85	89	23	339	146	–	
		–		–		52.42%	0.19	59.91%	0.27	74.16%	0.16	56.94%	0.21	–	
	HIPS (McAfee)	–		–		145	20	84	13	–		–		–	
		–		–		86.21%	0.22	84.53%	0.18	–		–		–	
	Antivirus (McAfee)	–		–		7	2	11	2	–		–		–	
		–		–		71.43%	0.25	81.82%	0.2	–		–		–	
	Domain Controller	–		–		–		235	111	182	73	341	128	–	
		–		–		–		52.77%	0.27	59.9%	0.28	62.47%	0.31	–	
	Web Server (IIS)	–		–		–		–		–		–		269	102
		–		–		–		–		–		–		62.09%	0.25
LANL	NetFlow	12635842	3698521	8235691	5865210	–		–		–		–		–	
		70.73%	0.16	28.79%	0.09	–		–		–		–		–	
	Host OS	–		–		18365987	10256980	6985255	4586250	2156890	1365897	1458797	569876	–	
	(Authentication)	–		–		44.16%	0.12	34.35%	0.1	36.68%	0.12	60.94%	0.15	–	
	Audit Log	–		–		4236587	1968532	2659872	897562	1035682	236584	3608569	2896532	–	
		–		–		53.54%	0.15	66.26%	0.18	77.16%	0.26	19.74%	0.06	–	
	DNS Server	–		–		–		–		–		–		7985620	1968533
		–		–		–		–		–		–		75.35%	0.29

Table 13. Comparison performance of our method and the related works to the event aggregation

Metric / Ref.	# of Distinct Sensors	# of Distinct Features	EAR_{max}	Other Evaluated Metrics	Heterogenous Aggregation	Scalable Architecture	Stream Processing
[16]	1	1	0.999	–	×	×	✓
[17]	1	5	0.85	–	×	×	×
[18]	1	1	N/A*	$EPR_{max} = 11700$, $CQ_{max} = 0.949$	×	×	×
[19]	1	4	0.95	–	×	×	×
[20]	1	3	0.986	–	×	×	✓
[21]	1	6	0.997	$CQ_{max} = 0.989$	×	×	✓
[22]	1	8	0.98	–	×	×	✓
[23]	1	5	0.98	–	×	×	×
[24]	1	4	1.0	–	×	×	×
[25]	1	3	0.4	$EPR_{max} = 178571$	×	✓	×
[26]	1	2	0.993	$ILR_{max} = 0$	×	×	×
[27]	1	5	0.999	–	×	×	×
[7]	8	5	N/A	–	✓	×	×
[28]	2	6	0.841	–	✓	✓	×
[29]	2	8	0.999	–	✓	✓	✓
Our Method	20	48	0.997	$EPR_{max} = 15930$, $ILR_{max} = 0.06$, $CQ_{max} = 0.994$	✓	✓	✓

* N/A: Not Available.

EAR is slightly lower than the superior method due to the many features of this type of log, the proposed method performs better for other sensors. Regarding EPR, the proposed method also has a higher value due to the scalable nature of the architecture.

Based on the results of evaluations performed on the proposed aggregation method and comparing it with other similar methods, it can be inferred that the performance of our method can be influenced by several factors such as event generation rate by the sensor, the *TWL*, the number of event features, the number of *NSFS* and *SFS* of a sensor, and also the threshold values for each of the system parameters, i.e., *SF* threshold vectors. The effects of some mentioned factors are analyzed and discussed in the following sub-section.

5.2 Analysis and Discussion

After evaluating the proposed event aggregation method, in this section, we examine the results of

Table 14. The efficiency of our method against similar methods in terms of the most important evaluation metrics

Sensor Type	Ref.	# of Total Events	# of Summarized Events	EAR_{avg} (%)	ILR	EPR_{avg}
NIDS (Snort)	[7]	69,039	18,778	72.9	0.45	118.5
	[16]		7,801	88.8	0.38	65.6
	[21]		3,797	94.6	0.41	81.3
	[22]		5,937	91.5	0.31	75.6
	[29]		8,008	88.5	0.39	66.2
	Our Method		759	99	0.29	174.7
Host OS (Linux Syslog)	[7]	3,925	1,679	57.3	0.43	72.6
	[16]		2,362	39.9	0.36	65.9
	[21]		1,307	66.8	0.35	60.8
	[22]		1,978	49.7	0.19	62.3
	[29]		2,382	39.4	0.35	76.3
	Our Method		1,456	63	0.15	178.4
Web Server (Apache)	[7]	7,621	3,406	55.4	0.41	81.3
	[16]		4,709	38.3	0.25	75.9
	[21]		3,284	57	0.31	69.3
	[22]		3,680	51.8	0.23	72.5
	[29]		4,938	35.3	0.28	76.8
	Our Method		3,216	57.9	0.21	155.5

several analytical reviews on the datasets and the relationship of different parameters. Based on these analyses, we show how the proposed aggregation method helps to develop event preprocessing analyses such as event aggregation to design a SIEM or SOC solution.

5.2.1 The Relationship between EAR and TWL

During the execution of the proposed event aggregation method, one of the most influential parameters for the aggregation performance is the value of *TWL* parameter for the aggregation and summarization analysis. According to the results of the experiments in Figure 13, it can be inferred that the value of this parameter has a direct impact on the EAR metric. Regarding the results, the longer the *TWL* is, the higher the EAR will be. This analysis applies to all event types of different detection levels, as shown in the figure. It should be noted that Network-level sensors have a higher EAR value than the sensors of the other detection levels because of their higher event generation rates with less time for similar logged events.

5.2.2 The Relationship between EPR and TWL

The second analysis investigates the effect of the *TWL* on the execution time of the event aggregation algorithms and thus examines the rate of EPR. Figure 14 shows the results of algorithm execution on the LANL dataset's events with different values for the *TWL* parameter. The results of the experiments show that for all types of events, if the *TWL* is set to a smaller value, the event aggregation, filtration, and event summarization tasks (as core activities of the method) are faster due to fewer comparisons and computations. In addition, the number of features in *NSFS* and *SFS* for an event type affects overall performance. For example, based on Figure 14, DNS Server events are faster processed than NetFlow due to having fewer features.

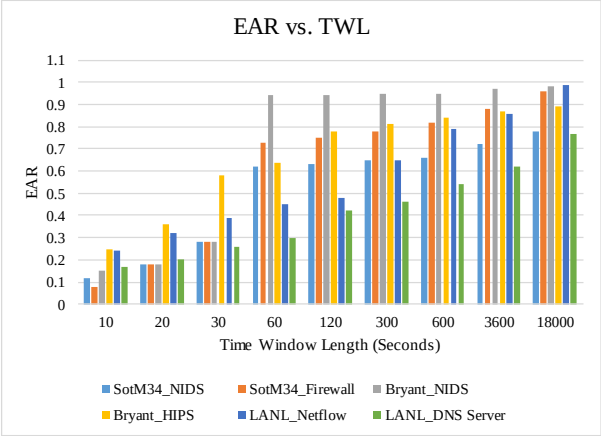


Figure 13. The effect of TWL on the EAR metric

Another test concerning the EPR metric is the system's scalability in the face of a different range of events received by the proposed aggregation system. Each specific sensor generates events at a certain rate, called the sensor's event per second (*EPS*), usually between 1 and 50. For the sensors deployed in a monitored network, an average EPS ratio is measurable regarding a specified time interval. According to the SANS report, [64], this ratio will reach more than 8000 EPS for an enterprise network, which in practice, leads to generating a massive amount of alerts/events in the network.

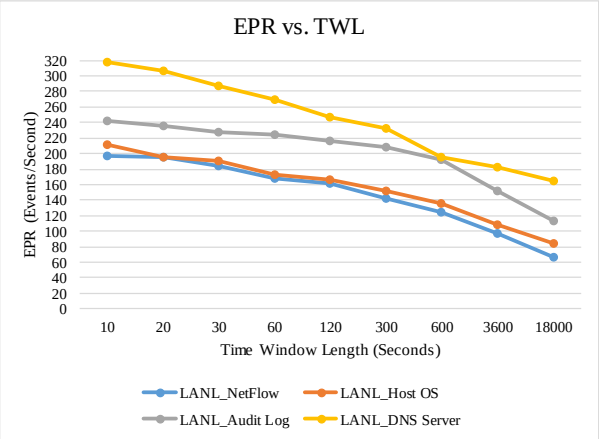


Figure 14. The effect of TWL on the EPR metric

In our proposed event aggregation method, the algorithms are designed to provide the highest level of parallelism in event aggregation and summarization processes. More precisely, according to the Algorithm 1, the logged events after being received are divided into two levels: sensor type (Line 3) and event type (Line 12), so that the relevant processes can be performed on them. In the proposed system, different event-processing tasks are done by different workers, each bound to a CPU core. According to the use of the ELK stack in the designing and implementation of the proposed system, we have four different types of workers, which are: 1) collector for receiving logged events and ingesting them into the system, 2) parser for event normalizing and transforming, 3) indexing for normalizing events and searching on them, and 4) aggregator for aggregating and summarizing indexed events. To evaluate the scalability of the proposed method and with the help of the different types of workers defined above, an experiment was performed using the LANL dataset with different EPS sizes, the results of which are shown in Figure 15. As can be seen in the figure, with increasing the number of workers based on the EPS rate, the EPR ratio has remained constant and has not changed significantly. Therefore, it can be concluded that by increasing the processing resources, the proposed method can handle different input rates without decreasing performance and not suffer from efficiency loss.

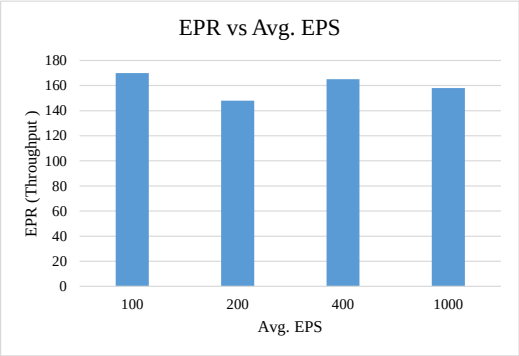
5.2.3 Aggregation Performance Curve (APC)

The other principal analysis in intrusion event aggregation is drawing the aggregation performance curve (APC) [26]. The APC presents the relationships between the EAR and ILR metrics when threshold vectors of the *SF* set vary. In this experiment, the NIDS events from the SotM34 dataset and the NIDS and Firewall events from the Bryant dataset are used for evaluated due to having similar event features. According to the experiment, the aggregation method is repeated eight times based on the eight different threshold vectors for the *SF* set of the events (Table 15). The EAR and ILR metrics are computed for each identical threshold vector in each algorithm iteration. The result of the experiment is shown in Figure 16 for all the three mentioned sensor events (for each curve, the leftmost point is for V1, and the rightmost point is for V8). Definable threshold vectors in the event summarization component allow the security administrator to adjust the threshold values of each feature of different sensors based on the amount of summary required. In other words, applying threshold vector allows events to be summarized flexibly.

According to the experiment results, it can be inferred that the EAR and ILR metrics directly impact each other. In other words, when the value of the EAR metric increases, the value of ILR increases too. In addition, changes in the values of the threshold vector are also influential on the EAR and ILR values. It means that changing from one threshold vector to another one may cause a notable change in EAR and ILR behavior or not. For example, changes in the values of the threshold vector, despite the increase in EAR metric for the Firewall events, it does not have much influence on the ILR metric due to the nature of event feature values. Therefore, the efficiency of the aggregation method cannot be judged from the EAR alone. However, its impact on the ILR, which is very important, must also be considered by security analysts.

5.2.4 The Effect of the TWL on Storage Space Reduction

The third analysis relates to the study of the effect of *TWL* parameter on required storage space in real applications, i.e., SIEM and SOC, which is essential for security practitioners. In the mentioned systems, the storage space is always one of the main challenges in storing the low-level events for the rest analysis, i.e., attack strategy mining and forensics investigations. Regarding the proposed aggregation method, the effect of different *TWL* values on the needed storage is illustrated in Figure 17 by using the SotM34 dataset events. Based on the results, the longer the *TWL* are adjusted, the less storage space is required to store output summarized events. For example, for the Firewall sensor, if the *TWL* is set to 18000 seconds (5 hours), the storage capacity is reduced by about six times when the *TWL* is set to 30 seconds. Generally, it is recommended to set a higher *TWL* where the rate of event generation per second is high by the sensor and the storage capacity is low.



Avg. EPS	# of Collector	# of Parser	# of Indexer	# of Aggregator
100	1	1	1	1
200	1	2	1	2
400	1	2	2	2
1000	1	2	2	3

Figure 15. The effect of sensor's average EPS on the EPR metric

Table 15. The threshold vectors for the *SFS* of the NIDS and Firewall sensors in our experiments

SFS Name / Vector#	Source IP Address	Source Port	Destination Port	Event Type (ID)	TTL	Protocol
V1	4	1	1	3	1	2
V2	4	1	1	3	1	2
V3	3	1	1	3	1	2
V4	3	1	1	2	1	2
V5	2	1	1	2	1	2
V6	2	1	1	2	1	1
V7	2	1	1	1	1	1
V8	1	1	1	1	1	1

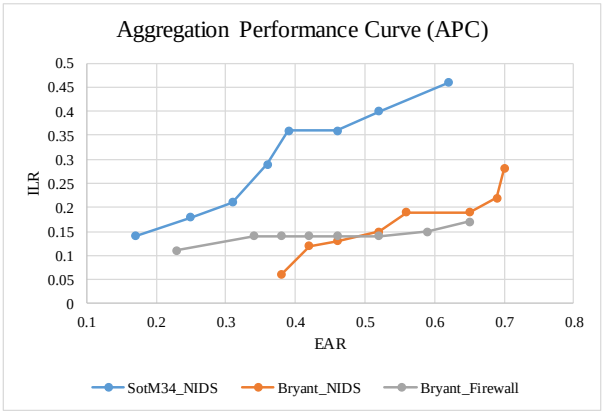


Figure 16. The APC for the NIDS and Firewall sensors in our experiments

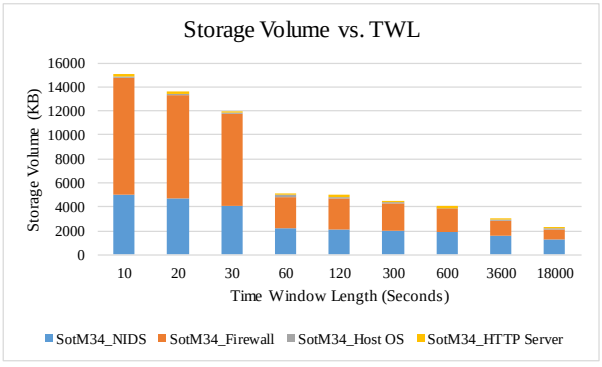


Figure 17. The effect of the *TWL* on storage space reduction

6 Conclusion and Future Work

One of the main promising approaches to track the attacker's behaviors and detect malicious activities during targeted multi-stage attacks like APTs is using various heterogeneous security and non-security sensors in different lines of defense in a monitored network (Network, Host, and Application). One of the main challenges of this approach is the massive amount of logged events raised by heterogeneous sensors for tracking the malicious attackers behind APT attacks. The main objective of this paper is to propose an event aggregation method to reduce the volume of logged events by heterogeneous sensors. These low-level events are generated during the different attack stages of an IKC model of complex targeted cybersecurity attacks, e.g., APTs. Such event aggregation analysis is a critical necessity to ease the management of the events for later usage, i.e., event correlation analysis for timely detection of APT attack scenarios. The input of the proposed method is a set of events logged by a set of heterogeneous sensors, which can be deployed

in the three primary detection levels of a target network (Network, Host, and Application). The proposed aggregation method has three main components as follows: 1) the event aggregation component for clustering similar logged events of each sensor based on an attribute-based similarity matching regarding some non-summarizable features of the sensor type (*NSF* set), 2) the event filtration component for eliminating the noisy events from the output event clusters by using a clustering-based local outlier factor, and 3) the event summarization component for fusing remaining events and improving their quality by leveraging an attribute-oriented induction method regarding some summarizable features (*SF* feature set) of each sensor type. Our implementation and experimental results have proved that the proposed method makes it possible to summarize heterogeneous events by eliminating redundant and false information with an acceptable level for the performance metrics.

In the future, some improvements can be made to the proposed aggregation method as follows: 1) The aggregation component of the proposed method works based on a fixed *TWL* for each of the sensors, which may not be adequately efficient for environments where the attacker's behavior changes dynamically during attacks. To solve this problem, setting a dynamic time window length for the sensors based on predicting how long an attacker will stay in a unique IKC stage of an APT attack will be a good solution. Another improvement that can be made to the event aggregation process is the ability to cluster the events hierarchically. It seems that by using this type of clustering analysis, the aggregation component can cluster events based on different levels of granularity, which makes it easier to analyze the clustering output for the network security administrator, and 3) Using new attack frameworks such as the *MITRE ATT&CK*, the impact of event aggregation analysis can be examined in addition to the level of IKC stages (Tactic in *MITRE ATT&CK*), at the level of attack steps (Technique in *MITRE ATT&CK*). The other main future work is proposing a general event correlation framework to detect APT attack scenarios based on the IKC models by analyzing aggregated events produced by the proposed aggregation method.

References

[1] S. Quintero-Bonilla and A. Martín del Rey. A new proposal on the advanced persistent threat: A survey. *Applied Sciences*, 10(11):3874, 2020.

[2] M. Husák, J. Komárková, E. Bou-Harb, and P. Čeleda. Survey of attack projection, prediction, and forecasting in cyber security. *IEEE Communications Surveys & Tutorials*, 21(1):640–660, 2018.

[3] S. Singh, P. K. Sharma, S. Y. Moon, D. Moon, and J. H. Park. A comprehensive study on apt attacks and countermeasures for future networks and communications: challenges and solutions. *The Journal of Supercomputing*, 75:4543–4574, 2019.

[4] G. Kim, C. Lee, J. Jo, and H. Lim. Automatic extraction of named entities of cyber threats using a deep bi-lstm-crf network. *International journal of machine learning and cybernetics*, 11:2341–2355, 2020.

[5] M. Khosravi-Farmad, A. A. Ramaki, and A. G. Bafghi. Moving target defense against advanced persistent threats for cybersecurity enhancement. In *2018 8th International Conference on Computer and Knowledge Engineering (ICCKE)*, pages 280–285. IEEE, 2018.

[6] P. K. Bahrami, A. Dehghantanha, T. Dargahi, R. M. Parizi, K. R. Choo, H. Javadi, Lizhe Wang, Fatos Xhafa, and Wei Ren. A layered security architecture based on cyber kill chain against advanced persistent threats, 2019.

[7] B. D. Bryant and H. Saiedian. A novel kill-chain framework for remote security log analysis with siem software. *Computers & Security*, 67:198–210, 2017.

[8] F. Wilkens, F. Ortmann, S. Haas, M. Vallentin, and M. Fischer. Multi-stage attack detection via kill chain state machines. In *Proceedings of the 3rd Workshop on Cyber-Security Arms Race*, pages 13–24, 2021.

[9] E. M. Hutchins, M. J. Cloppert, and R. M. Amin. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Leading Issues in Information Warfare & Security Research*, 1(1):80, 2011.

[10] R. Luh, S. Marschalek, M. Kaiser, H. Janicke, and S. Schrittwieser. Semantics-aware detection of targeted attacks: a survey. *Journal of Computer Virology and Hacking Techniques*, 13(1):47–85, 2017.

[11] A. Spadaro. *Event correlation for detecting advanced multi-stage cyber-attacks (Doctoral dissertation)*. Delft University of Technology, 2013.

[12] S. Salah, G. Maciá-Fernández, and J. E. DiAz-Verdejo. A model-based survey of alert correlation techniques. *Computer Networks*, 57(5):1289–1317, 2013.

[13] A. A. Ramaki, A. Rasoolzadegan, and A. G. Bafghi. A systematic mapping study on intrusion alert analysis in intrusion detection systems. *ACM Computing Surveys (CSUR)*, 51(3):55, 2018.

[14] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer. Comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on*

dependable and secure computing, 1(3):146–169, 2004.

[15] R. H. Syed, J. Pazardzievska, and J. Bourgeois. Fast attack detection using correlation and summarizing of security alerts in grid computing networks. *The Journal of Supercomputing*, 62(2):804–827, 2012.

[16] A. A. Ramaki, M. Amini, and R. E. Atani. Rteca: Real time episode correlation algorithm for multi-step attack scenarios detection. *Computers & Security*, 49:206–219, 2015.

[17] M. Husak, M. Cermak, M. Lastovicka, and J. Vykopal. Exchanging security events: Which and how many alerts can we aggregate? In *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 604–607. IEEE, 2017.

[18] G. P. Spathoulas and S. K. Katsikas. Enhancing ids performance through comprehensive alert post-processing. *Computers & Security*, 37:176–196, 2013.

[19] Y. Sun and X. Chen. An improved frequent pattern growth based approach to intrusion detection system alert aggregation. *In Journal of Physics: Conference Series*, 1437:1, 2020.

[20] F. M. Alserhani. Alert correlation and aggregation techniques for reduction of security alerts and detection of multistage attack. *International Journal of Advanced Studies in Computers, Science and Engineering*, 5(2):1, 2016.

[21] R. Shittu, A. Healing, R. Ghanea-Hercock, R. Bloomfield, and M. Rajarajan. Intrusion alert prioritisation and attack detection using post-correlation analysis. *Computers & Security*, 50:1–15, 2015.

[22] M. Soleimani and A. A. Ghorbani. Multi-layer episode filtering for the multi-step attack detection. *Computer Communications*, 35(11):1368–1379, 2012.

[23] S. C. de Alvarenga, Barbon Jr, Miani S., R. S., M. Cukier, and B. B. Zarpelão. Process mining and hierarchical clustering to help intrusion alert visualization. *Computers & Security*, 73:474–491, 2018.

[24] R. Zhang, T. Guo, and J. Liu. An ids alerts aggregation algorithm based on rough set theory. *In IOP Conference Series: Materials Science and Engineering*, 322:6, 2018.

[25] J. Kim, I. Moon, K. Lee, S. C. Suh, and I. Kim. Scalable security event aggregation for situation analysis. In *First International Conference on Big Data Computing Service and Applications*, pages 14–23. IEEE, 2015.

[26] S. Saad and I. Traore. Heterogeneous multi-sensor ids alerts aggregation using semantic analysis. *Journal of Information Assurance & Security*, 7:2, 2012.

[27] A. Nadeem, S. Verwer, and S. J. Yang. Sage: Intrusion alert-driven attack graph extractor. In *2021 IEEE Symposium on Visualization for Cyber Security (VizSec)*, pages 36–41. IEEE, 2021.

[28] O. B. Fredj. A realistic graph-based alert correlation system. *Security and Communication Networks*, 8(15):2477–2493, 2015.

[29] A. A. Ramaki and A. Rasoolzadegan. Causal knowledge analysis for detecting and modeling multi-step attacks. *Security and Communication Networks*, 9(18):6042–6065, 2016.

[30] H. Kim, H. Kwon, and K. K. Kim. Modified cyber kill chain model for multimedia service environments. *Multimedia Tools and Applications*, 78(3):3153–3170, 2019.

[31] J. R. Rutherford and G. B. White. Using an improved cybersecurity kill chain to develop an improved honey community. In *49th Hawaii International Conference on System Sciences (HICSS)*, pages 2624–2632. IEEE, 2016.

[32] S. K. Pandey and B. M. Mehtre. A lifecycle based approach for malware analysis. In *Fourth International Conference on Communication Systems and Network Technologies (CSNT)*, pages 767–771. IEEE, 2014.

[33] M. I. Center. *APT1: Exposing one of China's cyber espionage units*. Mandiant.com, 2013.

[34] J. Flynn. Intrusion along the kill chain. In *Proceedings of BlackHat USA*. BlackHat, 2012.

[35] Command Five Pty Ltd. Advanced persistent threats: A decade in review. http://www.commandfive.com/papers/C5_APT_ADecadeInReview.pdf.

[36] Dell SecureWorks. Breaking the kill chain- knowing, detecting, disrupting and eradicating the advanced threat. https://webobjects.cdw.com/webobjects/media/pdf/paloalto/Breaking-the-Attack-Kill-Chain.pdf, 2015.

[37] P. Pols and J. van den Berg. The unified kill chain. *CSA Thesis, Hague*, pages 1–104, 2017.

[38] A. J. C. Lima. PhD thesis, Advanced persistent threats, 2015.

[39] B. Hudson. *Advanced Persistent Threats: Detection, Protection and Prevention*. Sophos Ltd, 2014.

[40] E. Tonelli. *WatchGuard APT Blocker*. WatchGuard Technologies, 2014.

[41] Cox A. Stalking the kill chain: The attacker's chain. 2012.

[42] P. Chen, L. Desmet, and C. Huygens. A study on advanced persistent threats. In *IFIP International Conference on Communications and Multimedia Security*, pages 63–72. Springer, 2014.

[43] Advanced Persistent Threats and other Advanced Attacks, editors. Websense, 2013.

[44] Paul Giura and Wei Wang. A context-based detection framework for advanced persistent threats. In *2012 International Conference on Cyber Security*, pages 69–74. IEEE, 2012.

[45] K. Pei, Z. Gu, B. Saltaformaggio, S. Ma, F. Wang, Z. Zhang, L. Si, X. Zhang, and D. Xu. Hercule: Attack story reconstruction via community discovery on correlated log graph. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 583–595, 2016.

[46] P. Bhatt, E. T. Yano, and P. Gustavsson. Towards a framework to detect multi-stage advanced persistent threats attacks. In *8th International Symposium on Oriented System Engineering (SOSE)*, pages 390–395. IEEE, 2014.

[47] P. Giura and W. Wang. Using large scale distributed computing to unveil advanced persistent threats. *Science*, 1(3):93–102, 2013.

[48] S. Bhatt, P. K. Manadhata, and L. Zomlot. The operational role of security information and event management systems. *IEEE Security & Privacy*, 12(5):35–41, 2014.

[49] C. Vega, P. Roquero, R. Leira, I. Gonzalez, and J. Aracil. Loginson: a transform and load system for very large-scale log analysis in large it infrastructures. *The Journal of Supercomputing*, 73(9):3879–3900, 2017.

[50] S. Chhajed. *Learning ELK stack*. Packt Publishing Ltd, 2015.

[51] O. Negoita and M. Carabas. Enhanced security using elasticsearch and machine learning. In *Science and Information Conference*, pages 244–254, 2020.

[52] A. Sapegin, D. Jaeger, A. Azodi, M. Gawron, F. Cheng, and C. Meinel. Hierarchical object log format for normalization of security events. In *9th International Conference on Information Assurance and Security (IAS)*, pages 25–30. IEEE, 2013.

[53] X. Zhao, J. Liang, and F. Cao. A simple and effective outlier detection algorithm for categorical data. *International Journal of Machine Learning and Cybernetics*, 5(3):469–477, 2014.

[54] M. Amer and M. Goldstein. Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer. In *In Proc of 3rd RapidMiner Community Meeting and Conference (RCOMM)*, pages 1–12, 2012.

[55] Z. He, X. Xu, and S. Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003.

[56] M. Ahmed. Data summarization: a survey. *Knowledge and Information Systems*, pages 1–25, 2018.

[57] J. Jiang, J. Chen, K. K. R. Choo, C. Liu, K. Liu, and M. Yu. A visualization scheme for network forensics based on attribute oriented induction based frequent item mining and hyper graph. In *International Conference on Digital Forensics and Cyber Crime*, pages 130–143, 2017.

[58] A. Chuvakin. *Scan 34 - the honeypot project*.

[59] A. D. Kent. Comprehensive, multi-source cybersecurity events data set. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2015.

[60] S. Panichprecha. *Abstracting and correlating heterogeneous events to detect complex scenarios (Doctoral dissertation)*. Queensland University of Technology, 2009.

[61] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. Clustering validity checking methods: part ii. *ACM Sigmod Record*, 31(3):19–27, 2002.

[62] T. Dziopa. Clustering validity indices evaluation with regard to semantic homogeneity. *In FedCSIS Position Papers*, pages 3–9, January 2016.

[63] A. Hassanzadeh and R. Burkett. Samiit: Spiral attack model in iiot mapping security alerts to attack life cycle phases. In *5th International Symposium for ICS & SCADA Cyber Security Research 2018*, pages 11–20, 2018.

[64] J. M. Butler. *Benchmarking security information event management (SIEM)*. A SANS Whitepaper, 2009.

Ali Ahmadian Ramaki received his B.Sc. and M.Sc. degrees in computer software engineering from The University of Guilan in 2011 and 2014, respectively. He is now Ph.D. Candidate of Software Engineering at the Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran. His research interests are network and system security, network and host intrusion detection and prevention system, security information and event management system, big data processing, and high performance networking.

Abbas Ghaemi-Bafghi was born on April 1973 in Bojnord, Iran. He received his B.Sc. degree in Applied Mathematics in Computer from Ferdowsi University of Mashhad, Iran, in 1995. He received his M.Sc. and Ph.D. degrees in Computer engineering from Amirkabir (Tehran Polytechnique) University of Technology, Iran, in 1997 and 2004, respectively. He is a member of the Computer Society of Iran (CSI) and the Iranian Society of Cryptology (ISC). He is an associate professor of the Computer Engineering Department at Ferdowsi University of Mashhad, Mashhad, Iran. His research interests are in cryptology, network security, and risk management. He also

has published more than 100 conference and journal papers.

Abbas Rasoolzadegan received his B.Sc. degree in Software Engineering from Aeronautical University, Tehran, Iran, in 2004. He also received M.Sc. and Ph.D. degrees in Software Engineering from the Amirkabir University of Technology, Tehran, Iran, in 2007 and 2013, respectively. He is currently an Associate Professor with the Computer Engineering Department of Ferdowsi University of Mashhad, Mashhad, Iran. His main research interest is software quality engineering, especially in terms of design patterns and refactoring. For more details, visit the SQLab homepage at http://sqlab.um.ac.ir.

Appendix

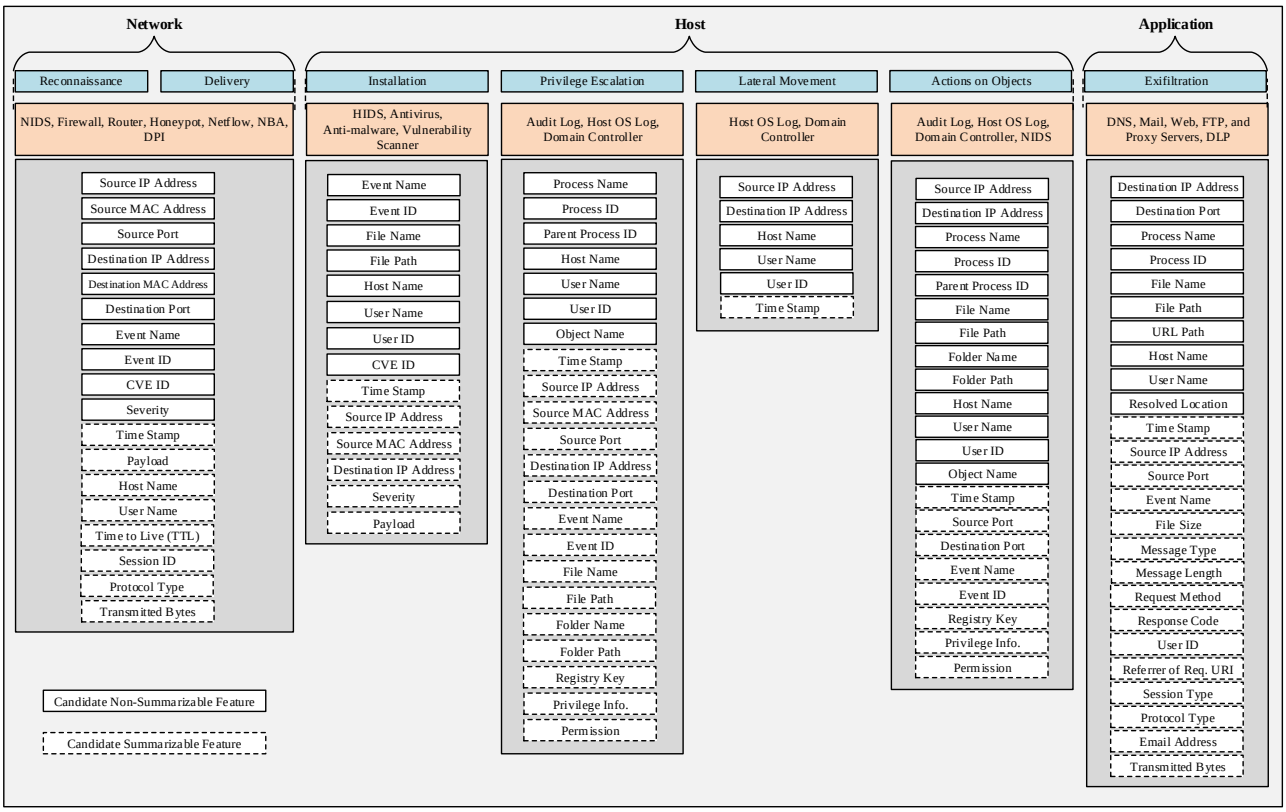


Figure 18. Candidate *NSFS* and *SFS* of each heterogeneous security and non-security sensor in different detection levels (Network, Host, and Application)

Table 16. Details of the existing publicly available datasets in the intrusion detection field

		Year	Duration	Size of Data	Attack Type	Host OS	Number of Instances (Network Address)	Labeled Data	Logging Full	Content for Packets	Logging All Sessions	of the Network Heterogeneous	Security Sensors	Complex and	Multi-step Attacks
Dataset	**Bryant**	2017	2 Days	38 MB	Exfiltration	Windows	48	✓	✓	✓	✓		✓	✓	
	LANL	2015	58 Days	~ 12 GB	N/A*	Windows	256	✓	✓	✓	✓		✓	✓	
	Defcon	2015	4 Days	24.5 GB	N/A	N/A	20	×	×	✓	✓		✓	×	
	ICTF	2015	1 Day	10 GB	N/A	Linux	36	×	✓	✓	✓		✓	×	
	ISCX	2012	7 Days	86 GB	Island Hopping	Windows and Linux	24	✓	✓	✓	×		✓	✓	
	CDX	2011	4 Days	12 GB	N/A	Linux	42	✓	✓	✓	×		✓	✓	
	The Internet Traffic Archive	2010	NA	5 GB	N/A	Linux	N/A	×	✓	✓	✓		✓	×	
	SotM34	2009	~ 4 Weeks	52.7 MB	Data Theft	Linux	N/A	✓	✓	✓	✓		✓	✓	
	LBNL	2008	~ 5 Days	11 GB	N/A	N/A	A few Thousand	×	✓	×	×		×	×	
	DARPA	2000	7 Weeks	4 GB	DoS, R2L, U2R, Probe	Unix	51	✓	×	✓	×		✓	×	
	KDD	1999	7 Weeks	743 GB	DoS, R2L, U2R	Unix	51	✓	×	✓	×		✓	×	

* N/A: Not Available.

Table 17. The threshold vectors for *SF* features of the sensors in our experiments

	Sensor Type	{SF Number (Threshold Value)}
SotM34	**NIDS (Snort)**	{1(2), 2(3), 3(1), 5(2), 7(1)}
	Firewall (IPTables)	{1(2), 2(3), 3(1), 4(2), 5(2), 7(1)}
	Host OS (Linux Syslog)	{1(2), 4(4), 8(3), 9(1), 11(2), 12(1)}
	Web Server (Apache)	{17(2), 19(2)}
Bryant	**NIDS (Snort)**	{1(2), 2(3), 3(1), 5(2), 7(1)}
	Firewall (pFsense)	{1(2), 2(3), 3(1), 4(2), 5(2), 7(1)}
	HIPS (McAfee)	{8(3), 17(2), 24(2), 25(1), 34(3), 35(2)}
	Antivirus (McAfee)	{17(2), 24(2), 25(1), 34(3), 35(2)}
	Host OS	{1(2), 4(4), 8(3), 9(1), 11(2), 12(1)}
	Web Server (IIS)	{2(3), 3(1), 5(2), 7(1), 17(2), 19(2)}
	Mail Server (Microsoft Exchange)	{1(2), 2(3), 27(1), 39(2)}
LANL	**NetFlow**	{1(2), 2(3), 3(1), 4(2), 5(2), 7(1)}
	Host OS (Authentication)	{8(3), 17(2), 24(2), 25(1), 34(3), 35(2)}
	Audit Log	{1(2), 4(4), 9(1), 11(2), 12(1)}
	DNS Server	{32(2), 34(2)}

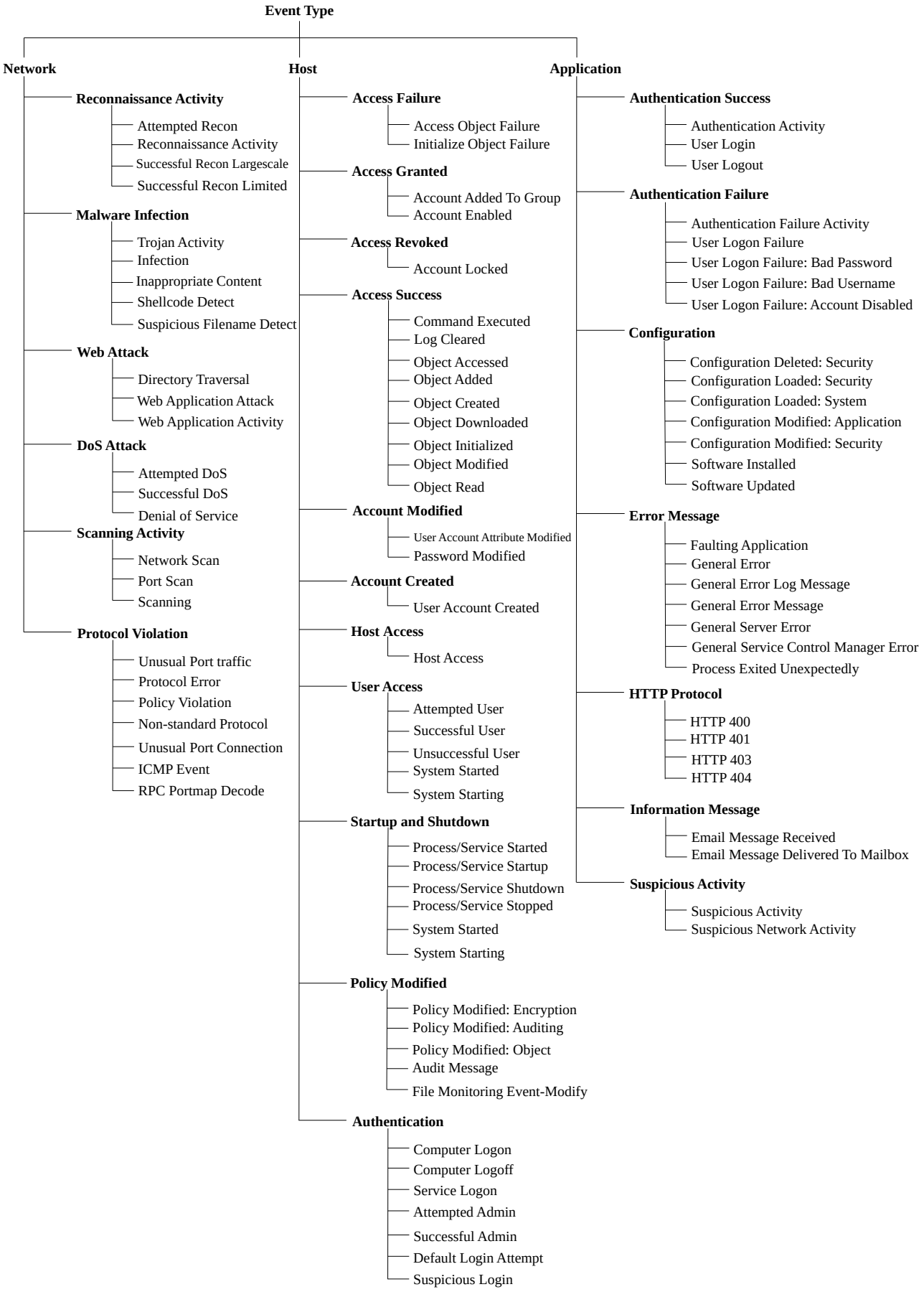


Figure 19. The used concept tree for the event type feature in our experiments