SN Computer Science (2020) 1:238
https://doi.org/10.1007/s42979-020-00226-8

ORIGINAL RESEARCH

A Methodology for Runtime Detection and Extraction of Threat Patterns

Christos Bellas[1] · Athanasios Naskos[1] · Georgia Kougka[1] · George Vlahavas[1] · Anastasios Gounaris[1] · Athena Vakali[1] · Apostolos Papadopoulos[1] · Evmorfia Biliri[2] · Nefeli Bountouni[2] · Gustavo Gonzalez Granadillo[3]

Received: 9 March 2020 / Accepted: 10 June 2020 / Published online: 21 July 2020

Abstract
As the confidentiality and integrity of modern health infrastructures is threatened by intrusions and real-time attacks related to privacy and cyber-security, there is a need for proposing novel methodologies to predict future incidents and identify new threat patterns. The main scope of this article is to propose an advanced extension to current Intrusion Detection System (IDS) solutions, which (i) harvests the knowledge out of health data sources or network monitoring to construct models for new threat patterns and (ii) encompasses methods for detecting threat patterns utilizing also advanced unsupervised machine learning data analytic methodologies. Although the work is motivated by the health sector, it is developed in a manner that is directly applicable to other domains.

Keywords IDS · Complex event processing · SIEM · Machine learning · Outlier detection

Introduction

The landscape of cyber-attacks is wide and extremely diverse, since attacks differ in multiple dimensions, such as their source [18, 26], the technique details and actor [19, 32, 33], the affected Open System Interconnection (OSI) layer [18] and the system infrastructure targeted [33]. The protection of any computing system encompasses the integrity, confidentiality and availability of its resources [17]; when these three security conditions are met, the system is considered safeguarded against intrusions.

To this end, organisations typically set up preventative infrastructures, with Intrusion Detection Systems (IDS) and Intrusion Prevention Systems (IPS) lying at their core. An IDS comes as a hardware or/and software-based solution responsible for the prompt detection of real-time attacks and notification of the system or administrators about the attempted intrusion [24]. IPSs are server- or appliance equipment-based products that initiate the appropriate prevention responses to block the detected attack [24]. The real activity of an IPS starts when the role of an IDS finishes, that is, upon the detection of an intrusion, although some IPSs may

✉ Georgia Kougka
georkoug@csd.auth.gr

Christos Bellas
chribell@csd.auth.gr

Athanasios Naskos
anaskos@csd.auth.gr

George Vlahavas
gvlahavas@csd.auth.gr

Anastasios Gounaris
gounaria@csd.auth.gr

Athena Vakali
avakali@csd.auth.gr

Apostolos Papadopoulos
papadopo@csd.auth.gr

Evmorfia Biliri
evmorfia@suite5.eu

Nefeli Bountouni
nefeli@suite5.eu

Gustavo Gonzalez Granadillo
gustavo.gonzalez@atos.net

[1] Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece

[2] Suite5, Limassol, Cyprus

[3] Atos Research and Innovation, Barcelona, Spain

incorporate IDS functionalities like monitoring and attack detection [12]. However, such systems, and broader modern SIEM (Security Information and Event Management) solutions, fall short in meeting modern requirements in modern health sector's infrastructures, as will be illustrated shortly.

Background

Knowledge extraction from available data for identification of attacks is closely related to the attack detection approaches and techniques used in IDSs. The field of intrusion detection has been widely studied in literature and multiple techniques have been developed to address this challenging cyberworld problem. IDSs, seen from a research and industry perspective, are generally complex systems that vary across many dimensions, such as the source of audit data, applied detection techniques, provided decision-making functionalities and so on. According to [26], IDSs can be

- Host based, which are locally installed IDSs on host machines and act as agents constantly monitoring the individual host for malevolent activities, such as changes to critical system files, an unwanted sequence of system calls, unusual CPU activity and more [9]. They are mainly concerned with events like process identifiers and system calls related to OS information [13].
- Network based, which are usually placed at strategic points of the network, such as gateways and routers, and monitor inbound and outbound traffic to detect possible attacks. Their focus is on network-related information, as for example traffic volume, IP addresses, service ports, protocols.
- Hybrid, which, combine features of the two categories above.

In terms of the methods and algorithms employed, a high-level categorization of the detection approaches used in IDSs includes two types: misuse detection and anomaly-based detection [12, 26, 27]. Misuse detection deals with known attacks, while anomaly detection aims to recognize novel attacks.

A misuse detection IDS generates attack signatures (profiles) from previously known attacks, which serve as a reference to detect future attacks. The term "signature-based detection" is often used interchangeably with misuse detection [9, 18]. These systems provide accurate predictions of known attacks and demonstrate low false-positive rate. However, they are not capable of catching novel attacks. Another drawback of misuse detection is its reliance on the system for regular knowledge maintenance and constant updates. Techniques for misuse detection can be further categorized as knowledge- and machine learning-based. Knowledge-based systems depend on predefined patterns and rules in order to audit network traffic and host logs [26]. In machine learning-based systems, models are trained based on established methodologies, e.g., Artificial Neural Networks (ANNs), Decision Trees (DTs), Support Vector Machines (SVMs), and so on [4]. These systems rely on training data; as such, their performance deteriorates in the absence of a labelled training set, rendering single classifiers inadequate to cope with zero-day and other unknown attacks [23] and [7].

Anomaly-based detection IDSs depend on learned normal patterns in order to flag any deviation from normality in scanned data as anomaly. IDSs designed for anomaly detection are based on the assumption that an attacker behaves differently from a normal user [6]. They perform behavioural modelling and are thus accurate and consistent in networks that follow static behavioural patterns [9]. In other cases, they show a high false-positive rate. Anomaly detection is implemented using statistical, knowledge-based or machine learning techniques [9, 13] and [18].

Motivation and Contributions

Our research is motivated by the current vulnerability of health sector's infrastructures to threats related to privacy and cybersecurity. It is conducted in the context of the EU H2020 CUREX project, the main goal of which is to produce a novel, flexible and scalable situational awareness-oriented platform that can address comprehensively the protection of the confidentiality and integrity of health data focusing on health data exchange cases [11]. The IDS solutions mentioned above are inadequate to address privacy-related threats combined with more commonly encountered cybersecurity risks.

Targeted example Imagine a scenario where an external audit detects that in a handful of cases, during transfer of personal health records, more personal data were revealed than necessary. What is the pattern that need to be monitored to avoid future occurrences, while not putting unnecessary burden on the system in terms of security measures? Modern IDS and SIEM systems do not include mechanisms for such cases that typically involve inappropriate system usage at multiple layers of a health IT system and call for techniques for pattern construction for threats not known a priori.

In this work, we propose a hybrid IDS that consumes both network monitoring logs and manually extracted timestamped annotated events that may refer to host machines as well. Our solution, termed as *Knowledge Extraction Analytics (KEA)* component, adopts methodologies to predict future incidents and identify new threat patterns. More specifically, it applies both misuse and anomaly detection techniques. In the misuse detection case, the techniques that have adopted are both knowledge and ML-based, while in the anomaly detection case, it uses ML-based techniques. The algorithms that we support are inspired by predictive

maintenance in Industry 4.0 [28, 29] and emphasize on (i) continuously monitoring for known patterns; (ii) continuously monitoring for unusual behavior following an unsupervised learning approach; and (iii) quick and easy model building for new threats at arbitrary combinations of system layers, provided that logs exist.

In summary, the novelty of our proposal lies in the novel application of items (ii) and (iii) above in a IDS system and on the provision of a unified solution for monitoring for known patterns, training for new attacks and detecting anomalous behavior without previous training. An additional strong point is that it can be easily combined with existing IDS solutions and transferred to other settings than the health sector. We explain the architecture and the main techniques, while we provide a reference prototype implementation.

The remainder of this article is structured as follows. In Sect. 2, we present the main functionality and the architecture of our solution. The technical details are in Sects. 3 and 4. In Sect. 5, we present the prototype implementation along with evaluation results. Additional related work is discussed in Sect. 6, while we conclude in Sect. 7.

Overview of Our Proposal

Functionality

The Knowledge Extraction Analytics (KEA) component harvests knowledge out of systems, sub-components and network communication interfaces that deal with sensitive data sources (assets) to construct models and design methods capable of detecting threat patterns. Machine learning and broader data analytic methodologies are utilized to develop classification models for revealing vulnerabilities and profiling threats; these models, along with known rules, are then used for runtime threat detection and prediction. In addition, runtime anomaly detection is continuously performed without the need of any prior model training.

Semantically, KEA processes two types of input data:

rm 1. infrastructure status logs and raw measurements of relevance; and
rm 2. timestamps of intrusion incidents and suspicious actions.

The former is received either through direct monitoring of assets, e.g., pcap files containing information about the network traffic, or through post-processing and relying on other logging mechanisms, e.g., intrusion incidents logs provided by an existing SIEM suite. In terms of data formats, typically the status logs belong to a vocabulary of a length in the orders of thousands, while raw measurements are most commonly time-series of (multi-dimensional) points. This input data type is used to perform both complex event processing (CEP) [10] and anomaly detection, and to train models capable of detecting new threats. The second type of input data concerns timestamps of suspicious actions and incidents not detected by the system. Along with the relevant logs, it is used to train additional models so that patterns for these threats are extracted as well.

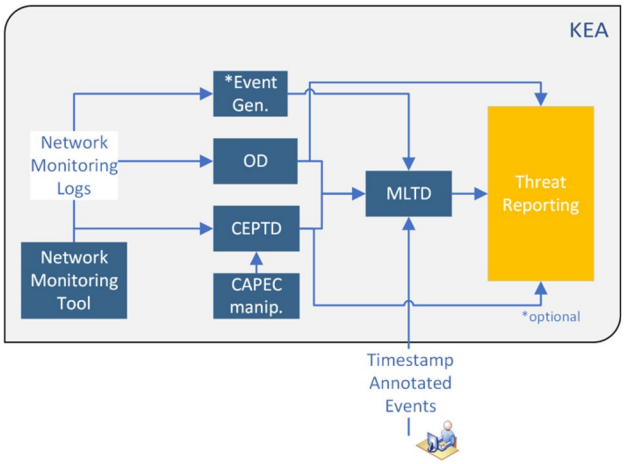


Fig. 1 The KEA architecture

Architecture

In this section, we introduce the architecture of the KEA component, analyze its individual modules and describe how they structure and communicate. Figure 1 shows the KEA architecture and its key modules.

The KEA analytics functionality introduced above is mainly implemented in three components:

rm 1. *Complex event processing-based threat detection (CEPTD)*,
rm 2. *Machine learning-based threat detection (MLTD)* and
rm 3. *Outlier detection (OD)*.

More specifically, the CEPTD uses the Common Attack Patterns Enumeration and Classification (CAPEC) [1] database as a source of predefined threat profiles for the construction of threat detection rules, used to process network traffic metrics and audit logs from existing Intrusion Detection Systems (IDSs) to detect threats. The MLTD utilizes ML algorithms and more specifically, event-based prediction algorithms, which are trained with timestamped annotated events and intrusion incidents obtained by the systems that deal with health data, to identify new threat patterns, predict prominent threats and produce reports about the patterns and the threats. The mapping of the identified threat

patterns from the MLTD component to the CAPEC database is examined to provide a fully CAPEC-compatible analytics solution. Finally, the latter main KEA component, namely OD, encapsulates the outlier detection technique based on unsupervised learning. This mechanism acts as a complementary tool to the deployed CEPTD and MLTD subcomponents to detect threats outside the CAPEC sphere. The final KEA solution is an ensemble one, as multiple algorithms are employed.

The output of the KEA component is in the form of patterns describing threats and vulnerabilities and early detected or predicted intrusion incidents or preceding patterns, using the CEPTD, MLTD and outlier detection approach.

Apart from the main components implementing the analytics functionality, KEA, as a complete working system, comprises additional auxiliary modules. In the following, we present all the components of KEA as depicted in Fig. 1 and their functionalities.

Machine learning-based threat detection (MLTD) component The MLTD component combines multiple sources of information and uses event-based prediction algorithms to predict future incidents and identify new threat patterns to assist the IDS (intrusion detection system) rule generation process. More specifically, the MLTD aims to detect and predict current or prominent security threats of special importance and to identify new threat patterns combining timestamped annotated events and network traffic information data. It conforms to the supervised machine learning paradigm, which implies that it requires (periodic) retraining. Multiple ML algorithms can be encapsulated in this component. Its main technique is transferred from Industry 4.0 [28, 29].

CAPEC rules manipulation component This component is responsible for the mapping of the CAPEC rules to Common Vulnerabilities and Exposures (CVEs) and the generation of Suricata [3] compatible rules (see below). The CVE list[1] provides a reference method for publicly known information-security vulnerabilities and exposures. In general, this component assists the rule generation process for the IDS running in the CEPTD. Leveraging CAPEC rules is essential for both the systematic treatment of threats according to the de facto standard approaches, and the exploitation of existing threat and vulnerability knowledge to the largest possible extent.

Complex event processing-based threat detection (CEPTD) component The CEPTD sub-component is based on a SELKS framework [3] installation, which is essentially a CEP system tailored to security issues, to identify online and report possible threats/intrusions. Specifically, the Suricata part of the SELKS framework processes the input pcap files to identify threats based on a set of rules. Complex event processing techniques build upon trained models and are responsible for the online operation to detect/predict incidents and threats on the fly.

Outlier detection (OD) component. The OD component monitors the network traffic and uses stream processing methodologies for detecting anomalous behavior in a scalable manner. Complementary, OD is applied offline to historical data for threat detection purposes and online, to detect suspicious behavior at runtime. In this setting, we employ unsupervised learning techniques, such as the MCOD technique [5, 21], to act in parallel with the supervised learning ones to support cases where training is not possible.

Event generation component This component extends the applicability of the afore-mentioned solutions not to work solely on annotated events but also on stream sensor data in the form of time series. The latter format is very common in several network monitoring applications [15].

Threat reporting component The threat reporting subcomponent is a Kibana-based [3] user interface, which visualizes the identified threat patterns from the MLTD, and the detected events from the CEPTD and OD components.

Controller component The orchestration of all the previously presented components along their communication is handled by the controller (CTRLER) component (not shown in Fig. 1). This components offers three main services: (i) an API providing the needed endpoints for the data exchange and the execution handling of the KEA components, (ii) a time series database for the persistent storage of the processed information needed for the training of the ML models of the MLTD sub-component, and (iii) a message queueing system for the exchange of the processed information serving the online analysis needs.

Based on the above, existing IDS functionality is covered by CEPTD. The main novelty of our methodology is that CEPTD is accompanied by MLTD and OD, which are examined in more detail in the next two sections, respectively. Also, the CAPEC rule manipulation and event generation components provide extra functionality.

Identifying and Detecting Threats

This section describes our novel approach to (i) identifying new threat patterns inspired by techniques used in predictive maintenance for Industry 4.0 [22, 28, 29] and (ii) applying CAPEC-compliant CEP and exploiting publicly available threat profile repositories.

Extracting New Threat Patterns from Logs

As discussed in Sect. 2, the MLTD sub-component applies event-based prediction algorithms to predict future incidents

[1] https://cve.mitre.org/

and identify new threat patterns that will contribute to the rule generation process. Event-based prediction is a well-studied field of machine learning, where series of events are fed into prediction models to estimate the occurrence of an unwanted incident. Event-based prediction is based on the assumption that there exist preceding patterns of "warning" events before the occurrence of a security incident with special severity. Hence, identifying and training on these patterns may assist the avoidance of such incidents.

One of the key roles of KEA is to obtain event logs, such as the (timestamped) annotated events, whether these are extracted semi-automatically, e.g., from security experts accessing system logs, or automatically, as the output from IDSs. Having events obtained from various sources, we are able to apply event-based prediction approaches in order, at a first stage, to identify unknown threat patterns, and, at second stage, to predict security incidents at runtime and as we are going to explain in the following.

Pre-processing Phase

There are two input requirements to apply our technique: (i) timestamped events, which essentially form a (partially ordered) sequence $< E_1, E_2, E_3, \cdots >$, where E_i is a set of events with timestamp i and (ii) timestamps of occurrences of severe security incidents. Each event $ev^x \in E_i$ belongs to a type, denoted by its annotation x, and the event dictionary, i.e., the domain of x, is large but finite. Initially, the collected events are partitioned into ranges defined by the occurrences of known security incidents of special severity, which will be called for the rest of this paper as *target events* [22]. These ranges are further partitioned into time segments, the size of which (i.e., minutes, hours, days) correspond to the time granularity of the analysis. The rationale behind the time segmentation is that the segments that are closer to the end of the range may contain suspicious events that are potentially indicative of the target event (main threat). The goal is to learn a function that quantifies the risk of the targeted incident occurring in the near future, given the events that precede it. Once such a function is learned, then events are monitored on the fly to raise early alarms before the target event occurs.

The technique involves several pre-processing steps. First, one or more consecutive segments are grouped and the events they contain form a bag. Then each such group of coalesced segments ending at timestamp t is assigned a risk factor according to a sigmoid function $F(t)$, which maps higher values to the segments that are closer to a security incident as it is presented in Fig. 2. The steepness and shift of the sigmoid function are configured to better map the expectation of the time before the incident at which correlated events will start occurring. This technique has been successfully applied in the aviation industry for predictive maintenance purposes [22] and is tested in other predictive maintenance cases in [28, 29].

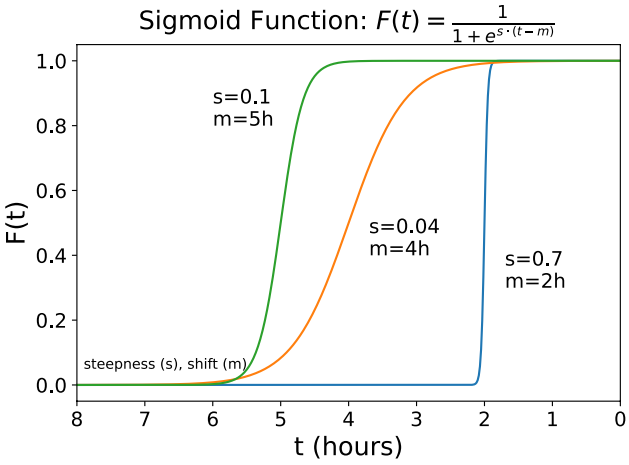


Fig. 2 Examples of sigmoid functions for the risk assessment of the events

A second step deals with dimensionality reduction. In practice, the event types are hundreds if not thousands. Essentially, each (coalesced) segment forms a vector $(b_1, b_2, \ldots, b_{|x|})$, where each event type corresponds to a different dimension. The binary variables $b_1, b_2, \ldots, b_{|x|}$ denote whether event types $ev^1, ev^2, \ldots, ev^{|x|}$ appear in the segment, respectively. Therefore, to increase the effectiveness of the approach, standard preprocessing techniques can be applied: (i) very rare or very common events compared to the frequency of the target events are pruned. (ii) Multiple occurrences of the same event in the same segment can either be noise or may not provide useful information; hence, multiple occurrences can be collapsed into a single one; this is done through the usage of binary variables. (iii) Standard feature selection (e.g., techniques like Relief) can also be used to further reduce the dimensionality of the data.

Finally, to deal with the imbalance of the labels (given that the actual security incidents are rare) and as several events appear shortly before the occurrence of the incident, but only a small subset of them is related to them, a Multiple Instance Learning (MIL) can be used for bagging the events and to facilitate the detection of the event patterns that can act as predictors. This rationale is implemented through considering more segments groups close to the target events; i.e., the data closer to the incidents (according to a specified threshold) are over-sampled, so that training is improved.

Model Training

The segmented data in combination with the risk quantification values are fed into a Random Forests (RF) algorithm [8] as a training set to form a regression problem. More specifically, we have a set of segment vectors, each associated with

a risk score, as given by the sigmoid function based on the timestamp the segment ends. The aim of the training is to learn the weights assigned to each vector dimension, i.e., to each event type. Having this dataset prepared, there are two options:

- the first one is to train a RF model with all the different types of target events;
- the second one is to train multiple RF models one for each one of the target event types.

The former provides a general solution capable of handling all the known incidents, while the latter provides more targeted solution for each type of target event. While the former is more practical and easier to handle, the latter is the most appropriate one for new threat identification. RF has an inherited capability of extracting the importance of the features based on two methods, namely *mean decrease impurity* and *mean decrease accuracy* [25]. Using one of these two methods, we are able to select the top-k important features identifying indirectly new threat patterns to create new rules for monitoring for specific threats. Overall, this allows, except from security incident detection and prediction, the KEA component to offer a functionality of identifying new threat patterns, which are not covered by known CVEs. This also explains our choice of RF, because we are also interested in combining the usage of a powerful technique with constructing a human-understandable pattern of the threat corresponding to the target events.

Transforming Time Series to Event Sequences

An optional functionality of the KEA component is the artificial event generation, so that the afore-mentioned technique, which requires event sequences as input, is applicable to time series data as well. To this end, an option could be to utilize a time-series discretization methodology, which generates events out of time series extracted from network monitoring tools, following established techniques as described in [4].

However, we propose the application a novel time-series discretization methodology tailored to extracting patterns capable of predicting rare events [29]. Our key novelty is that, instead of classifying time-series as a discrete set of events, we map time-series to a sequence of artificial events thus placing no burden to security experts to annotate the sensor measurements or to select and deploy the most efficient IDS. We only require information about the time of several critical incidents to train our methods, as previously.

To this end, we employ the Matrix Profile (MP) [2], which is a data structure that annotates time series. The application of the MP can provide efficient solutions to demanding data mining problems on time series. More specifically, problems with high complexity, such as anomaly detection and motif discovery can be reduced to trivial problems after applying MP. This methodology is based on the estimation of the (normalized) Euclidean distances between the subsequences of a time series to define similarities between them; these similarities in turn form the basis of several analytics tasks, as explained below.

For a time series T of length n, we estimate MP, which is a vector of length $(n - PL + 1)$, where PL is the predefined pattern length taken as a user-defined input parameter. $MP(i)$ denotes the distance of the sub-sequence of length PL starting at the i^{th} position in T to its nearest neighbor. Any distance metric can be used, but as explained in [34], the default option is the z-normalized Euclidean distance. The MP vector is accompanied by the Matrix Profile Index (MPI), which is of same size as MP. $MPI(i)$ keeps the pointer to the position of the closest neighbor of the subsequence of length PL starting at $T(i)$. The lower the values in the MP, the higher the similarity of the PL-size pattern beginning at the corresponding point to its closest neighbor.

With the help of the MP and MPI vectors, we transform T to a fully ordered sequence of events of length $n - PL + 1$, i.e., each timestamp between PL and n is associated with a single event. To do so, we consider that the MPI is essentially a directed graph, where each edge points to the most similar subsequence. We consider the MPI as a graph $G = (V, E)$, where the $V = v_1, ..., v_n$ denotes a set of nodes and $E = e_1, ..., e_z$ defines the edges of the graph G weighted by the values in MP. In this graph, each vertex has exactly one outgoing edge. As a following step, we estimate the weakly connected components (sub-graphs) of the MPI graph and map each such component to a distinct artificial event, i.e., in this step, we disregard edge directions. We can either stop at this step or apply some pruning reasoning in order to shrink the number of generated events. The pruning refers to both edges with high weights and components with very few elements. Finally, every point of the time series that is part of a connected component is labeled by the id of that component; this id becomes the x annotation in ev^x. The generated set of artificial events is fed into the MLTD component as all the other inputs, to be processed accordingly.

Leveraging Established Databases

The classification of detected threats using the knowledge extracted from the CAPEC database is another aspect of threat analysis. This is the result of a connection path, where a Suricata rule links to a CVE, then the CVE to a CWE and finally the CWE to a CAPEC.

More specifically, as already mentioned, the CEPTD component is based on the SELKS framework, which is a free and open source IDS/IPS platform and includes as a sub-component the Suricata system (see Fig. 3). Suricata

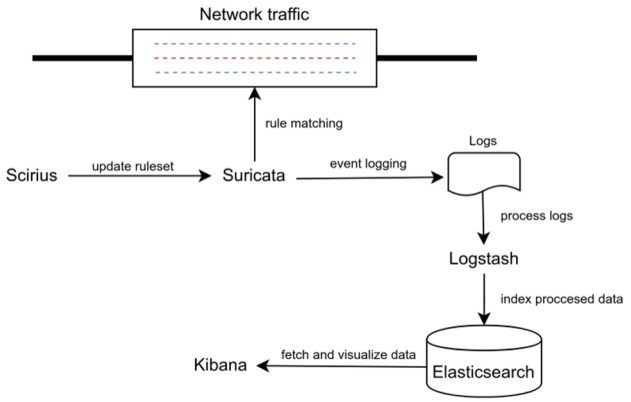


Fig. 3 The SELKS framework

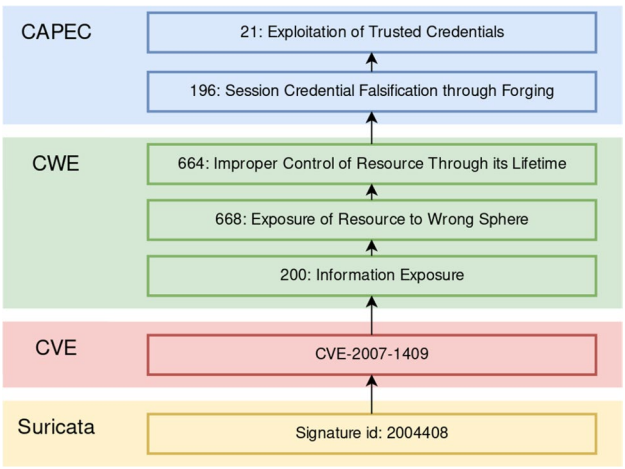


Fig. 4 Example of Suricata rule and CAPEC linkage

IDS supports two running modes, an online and an offline. In the online mode, the input to the threat detection mechanism is the real-time network traffic. On the other hand, in the offline mode, the input is a pcap file, i.e. a sequence of captured network packets for a specific time period. The latter is useful in cases such as profiling and evaluating new rules. For Suricata to log any suspicious activity, a mechanism called EVE is used. The EVE output facility outputs alerts, which may include metadata, file info and protocol specific records in JSON format. Alerts are essentially event records for rule matches. EVE can output to multiple methods, such as syslog and redis. However, the simplest approach is to store all alerts into a single file. Subsequently, third party tools such as Logstash can process this file.

Due to the constantly emerging CVEs and the complex relations among CAPECs and CWE (Common Weakness Enumeration)[2] , there is no automatic way of linking a Suricata logged alert to a CAPEC category. Furthermore, a CAPEC could consist of other CAPECs (the same also applies for CWEs). This results into a non-intuitive end-to-end connection. In most cases, further input by a domain expert is required. To this end, we have developed a semi-automated mapping alternative.

An example is shown in Fig. 4. In this example, Suricata detects a threat based on the rule with id 2004408 which is linked with CVE-2007-1409 and concerns the WordPress platform. More specifically, it describes that WordPress allows remote attackers to obtain sensitive information via a direct request, which reveals the path in an error message. This CVE in turn is linked directly with CWE-200 and with CWE-668, CWE-664 indirectly. Finally, using the most generic CWE, we can classify the threat to CAPEC-21. This is not the only possible mapping. For example, one could avoid CWE-664 and link CWE-668 with CAPEC-21. Each mapping mainly depends on the interpretation of the domain expert. However, it allows IT operators to use a globally accepted terminology.

Finally, in the context of KEA, we additionally employ the SELKS framework as follows:

Step 1: We receive the asset information from ADT.
Step 2: We then search CVE lists for relevant rules.
Step 3: We feed these rules to the SELKS framework for runtime detection.

To this end, we access publicly available repositories of pre-identified Suricata rules.[3]

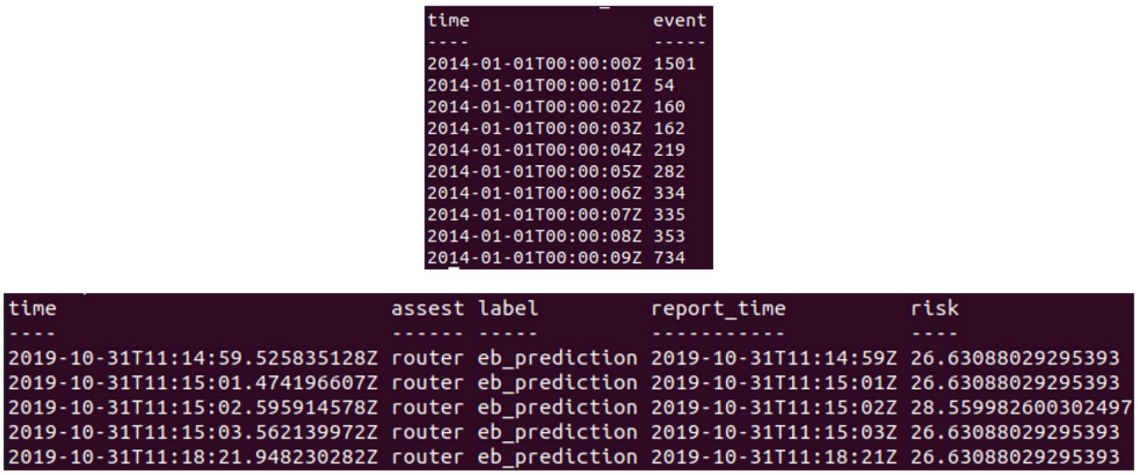
Outlier Detection

The KEA component incorporates an outlier detection (OD) approach to enhance its detection capabilities. The output of the detection tool is consumed internally by the MLTD component.

Outlier detection techniques are used to identify noise and anomalies in a dataset. In a streaming environment, because of the infinite nature of the data, detecting outliers needs to be done in combination with windowing techniques. A window splits the data stream into either overlapping or non-overlapping finite sets of data-points based either on the arrival of each point or the number of points. It is important to apply the OD process online in order to spot suspicious behaviour on time. Of course, offline deployment on historic data is also useful to produce events for model training in the MLTD component. To satisfy both requirements, we have

[2] https://cwe.mitre.org/

[3] e.g., http://emergingthreats.net



```
time                  event
----                  -----
2014-01-01T00:00:00Z 1501
2014-01-01T00:00:01Z 54
2014-01-01T00:00:02Z 160
2014-01-01T00:00:03Z 162
2014-01-01T00:00:04Z 219
2014-01-01T00:00:05Z 282
2014-01-01T00:00:06Z 334
2014-01-01T00:00:07Z 335
2014-01-01T00:00:08Z 353
2014-01-01T00:00:09Z 734
```

```
time                          assest label         report_time          risk
----                          ------ -----         -----------          ----
2019-10-31T11:14:59.525835128Z router eb_prediction 2019-10-31T11:14:59Z 26.63088029295393
2019-10-31T11:15:01.474196607Z router eb_prediction 2019-10-31T11:15:01Z 26.63088029295393
2019-10-31T11:15:02.595914578Z router eb_prediction 2019-10-31T11:15:02Z 28.559982600302497
2019-10-31T11:15:03.562139972Z router eb_prediction 2019-10-31T11:15:03Z 26.63088029295393
2019-10-31T11:18:21.948230282Z router eb_prediction 2019-10-31T11:18:21Z 26.63088029295393
```

Fig. 5 TimescaleDB examples: input timestamped annotated events (top) and predicted threats with associated quantified risk (bottom)

selected the application of an unsupervised outlier detection algorithm applied on streaming data, capable of handling big loads of data streams.

We have selected a distance-based outlier detection algorithm called MCOD, in which the number of a data point's neighbors represent its status as an anomaly or a normal point based on the following definition [20]:

"Given a set of objects O and the threshold parameters R and k, report all the objects o_i for which the number of neighbors $o_i.nn < k$, i.e., the number of objects o_j, $j \neq i$ for which $dist(o_i, o_j) \leq R$ is less than k. The report should be updated after each window slide."

In this definition, there is a sliding window containing the latest measurements, e.g., the measurements of the last hour or the last 100K measurements. Each object that deviates significantly from the rest of the window contents is reported. As such, abnormal behavior is contextualized with respect to the majority of the other behavioral logs in the same time period. A full description of MCOD is described in [21], while in [30], there exist efficient parallel implementations to handle intensive streams. Additionally, [31] provides impartial information about MCOD efficiency and superiority over competitors.

Implementation Aspects and Evaluation

Here we provide further implementation details about the system aspects of KEA along with some evaluation results. The source code and the required files for the reproduction of the results presented in this section are openly available in our git repository[4]

The MLTD Sub-component

The MLTD component is responsible for maintaining historical data for the training of the utilized machine learning models. The training process is repeated periodically as more incidents are collected in order to enhance the model efficiency. The processed information is in the form of timestamp annotated events like some of the input; hence a time series database is required. In the KEA prototype, we have employed TimescaleDB[5]. Figure 5 shows an example of input and output of the MLTD component. The input from external sources, such as network monitoring tools is collected and pipelined to the MLTD component using a message queuing mechanism; we have employed MQTT[6].

The key novelty of MLTD is that it offers a threat pattern identification functionality, providing access to the internal information of the trained model and more specifically to the importance of each event (i.e., feature). An IT admin obtains the *k* top important features, obtained possibly from different sources and utilizing the mapping between the event ids and the actual event details is able to define new threat patterns for known security incidents, which are not currently taken into consideration by the deployed IDSs.

As a proof of concept, a dataset with artificial/hypothetical event ids (male names in our case) is created and stored in a TimescaleDB deployment. We have infused

[4] http://interlab.csd.auth.gr/anaskos/curex-knowledge-extraction-analytics-kea

[5] https://www.timescale.com/

[6] http://mqtt.org/

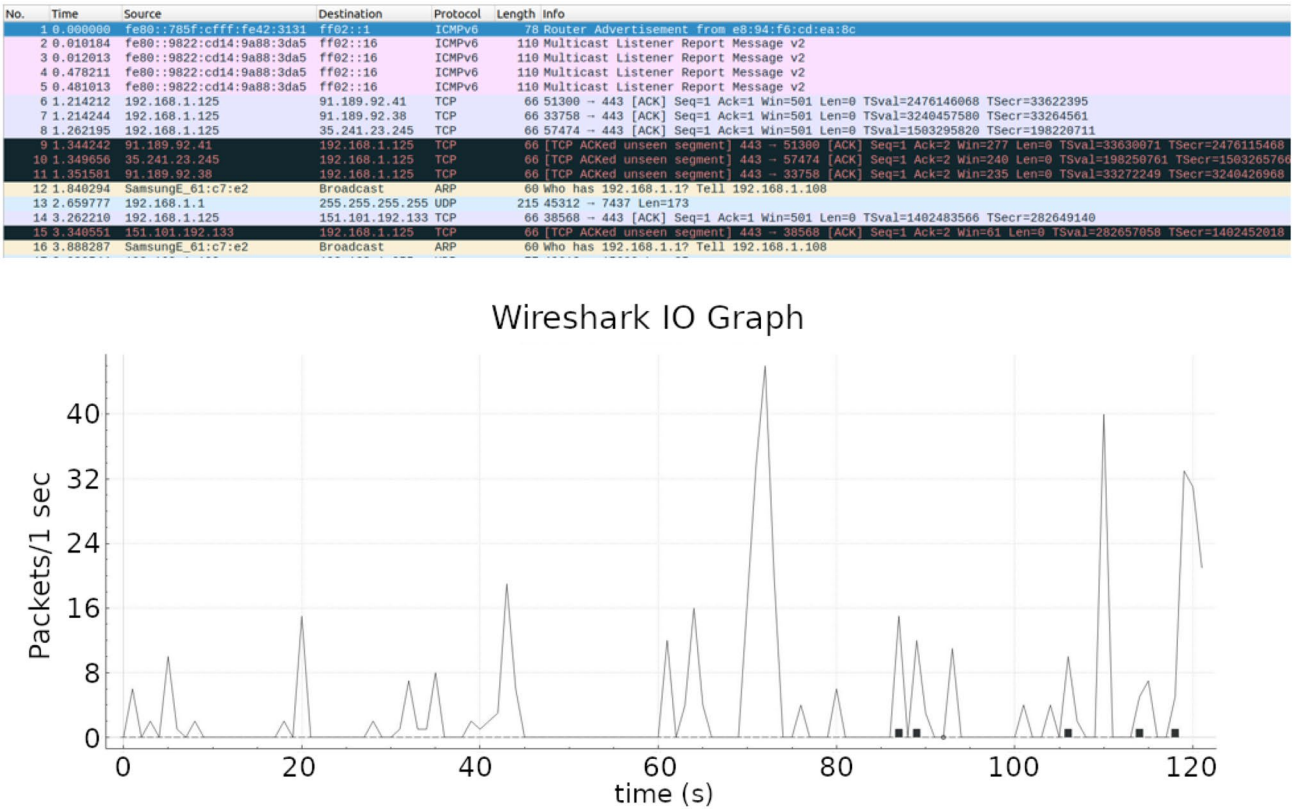


Fig. 6 Outlier detection example scenario: input pcap file (top) and TCP packet traffic (bottom)

hidden pre-specified threat event patterns, to resemble a scenario in which, before a major security incident, minor warning incidents might occur. We consider that in a real-world scenario, there exists a mapping between each event id and the actual event details (i.e., origin, description, etc.). The dataset spans for 1.5 months with event rate 2 events per minute. Every 2 days, (hence in 22 days in total), 6 h before a specific date time, 4 occurrences of the same pattern of events are infused with 20 min distance with each other. The goal is to train a model with this dataset and using the feature importance feature of the Random Forest algorithm, to check if the events of the pattern are included in most important ones.

The events of the repeated pattern are '***carrillo***', '***mendoza***', '*hawkins*', '***frank-lopez***', '***vasquez-ford***', '*young*', '*contreras*', '***dawson***', '*chandler*', '*everett*', '*arias-griffin*', '*coleman*', '*olson-scott*', '***powell***', '*branch-frank*', '*melendez*', '***flynn-davidson***', '***morgan***', '*turner-white*', '*robertson*'.

We have marked with bold the events that are included in the top-10 of the most important features of the Random Forest model. The first 8 out of the top-10 most important events are included in the pattern, i.e., despite having many irrelevant events between the warning events, our approach is capable of constructing a valid pattern.

The OD Sub-component

Regarding the OD component, MCOD is publicly available by authors of this work, as explained in [21, 30]. Here we provide an example execution of the OD sub-component for a given pcap file. For this example, a pcap file was extracted through the monitoring of the wireless network interface of a PC. The file captured 689 packet transmissions. A part of this pcap file is presented in Fig. 6(top). As it is shown in the protocol column, there are multiple protocols monitored; however, for the specific use case, only the TCP protocol is taken into consideration. Figure 6(bottom) plots the TCP packet traffic (black line) and the packet retransmissions (black bars). As it is presented, there is an increase in the retransmission rate after the 80th second. For the specific

Table 1 OD sub-component load test results for different threads

Threads	Iterations	MCOD mean time	Request mean time
1	100	1.3 ms	1.15 s
10	10	0.82 ms	4.39 s
20	10	0.45 ms	8.55 s

Table 2 OD sub-component load test results for different pcap file sizes

# of packets	Threads	Iterations	MCOD mean time	Request mean time
689	1	100	1.3ms	1.15s
28	1	100	0.04ms	0.62s

example, we have configured the MCOD algorithm with a small radius R and a large k parameter (i.e. a point is considered inlier only if it has a high number of neighbours in a "small" area) to become more sensitive and catch the small number of retransmissions included in this pcap file. The tool was able to identify an outlier value providing the following output: "Arrival time (seconds): 89 -Actual arrival time of the packet 2019-10-28T14:08:23Z"

Given that the OD runs continuously, to evaluate its performance, we have used the Apache JMeter[7] to produce multiple sequential and parallel requests to the OD, using the presented pcap file. For the experiments, an Intel 6 core (12 threads) server is used with 32GB of RAM and 256GB SSD.

Table 1 presents the results of the load test. We have used the default configuration in the repository, where the MCOD employs a sliding window containing the last 60 measurements and slides every 10 new items. The first column shows the number of parallel threads applying requests (i.e., uploading the pcap file for analysis) to the OD subcomponent. The second column presents the number of iterations of the experiment to compute the mean response time presented in the 3rd and 4th columns. The 3rd column presents the mean response time (in milliseconds) of the MCOD algorithm, while the 4th column presents the mean value of the total response time (in seconds) of the OD to process the pcap file (i.e., includes the pcap file parsing time, the data preprocessing steps, and the reporting of the results to the TimescaleDB).

We have repeated the single thread experiment using a smaller pcap file (28 captured packets) to present the change in the performance regarding the MCOD and the total OD response time; Table 2 presents the results. As we observe, the MCOD response time is reduced by 97% (from 1.3ms to 0.04ms) while the total response time is reduced by 46% (from 1.15s to 0.62s). The above results support the claim that MCOD is a lightweight, efficient solution in terms of performance.

The CEPTD Sub-component

Finally, as explained above, KEA encapsulates SELKS, which also comes with visualization components, e.g., Fig. 7 shows a visualised report about alerts signatures and destination ports based on incoming traffic. Since SELKS is not a solution developed by our team, and we simply re-use it as a basic component in the KEA overall architecture, we do not include any further evaluation.

Additional Related Work

In Sect. 1, we have already explained the scope of modern IDS solutions and how we go beyond them. In this section, we present more concrete state of the art regarding SIEM systems [14, 16].

IBM Qradar[8] is a tool that offers support for threat intelligence feeds and can optionally be extended with IBM Security X-Force Threat Intelligence, a module for identifying malicious IP address, URLs, and so on. The main weakness of this tool is that the endpoint monitoring for threat detection and response, or basic file integrity requires the use of third-party technologies. This SIEM provides basic reaction capabilities that include reporting and alerting functions. *Dell technologies (RSA)*[9] provides an evolved SIEM that analyzes data and behavior across a company's logs, packets and end-points, keeping at the same time an analysis on the behavior of people and processes within a network. The solution focuses on advanced threat detection and looks for integrating capabilities including network monitoring and analysis, end-point detection and response (EDR), and user and event behavior analytics (UEBA). In addition, the solution provides strong coverage of advanced threat defense, including real-time network and endpoint monitoring, forensic network and endpoint investigation. The main drawback of this solution is that although the number of technical components and the licensing models provide extensive flexibility in designing the deployment architecture, it requires understanding of the breadth of the options and the implications for cost, functionality and scalability. *LogRhythm*[10] is a tool that provides advanced SIEM features such as endpoint monitoring, network forensics, user and entity behavior analytics, and response capabilities. Although extensions and optional tools can be deployed to enhance the SIEM capabilities, e.g., network forensics, NetFlow monitoring, full-packet capture, the solution seems to be unsuitable for

[7] https://jmeter.apache.org/

[8] https://www.ibm.com/security/security-intelligence/qradar

[9] https://www.rsa.com/en-us/products/threat-detection-response/siem-security-information-event-management

[10] https://logrhythm.com/solutions/security/siem/

Suricata Alert Signature - Top 10

ID	Description	CNT
2200094	SURICATA zero length padN option	292
2200007	SURICATA IPv4 padding required	14
2009582	ET SCAN NMAP -sS window 1024	11
2027759	ET DNS Query for .co TLD	9
2002752	ET POLICY Reserved Internal IP Traffic	6
2100402	GPL ICMP_INFO Destination Unreachable Port Unreachable	6
2022218	ET POLICY Lets Encrypt Free SSL Cert Observed	5
2210037	SURICATA STREAM FIN recv but no session	4
2402000	ET DROP Dshield Block Listed Source group 1	3
2403368	ET CINS Active Threat Intelligence Poor Reputation IP group 69	2

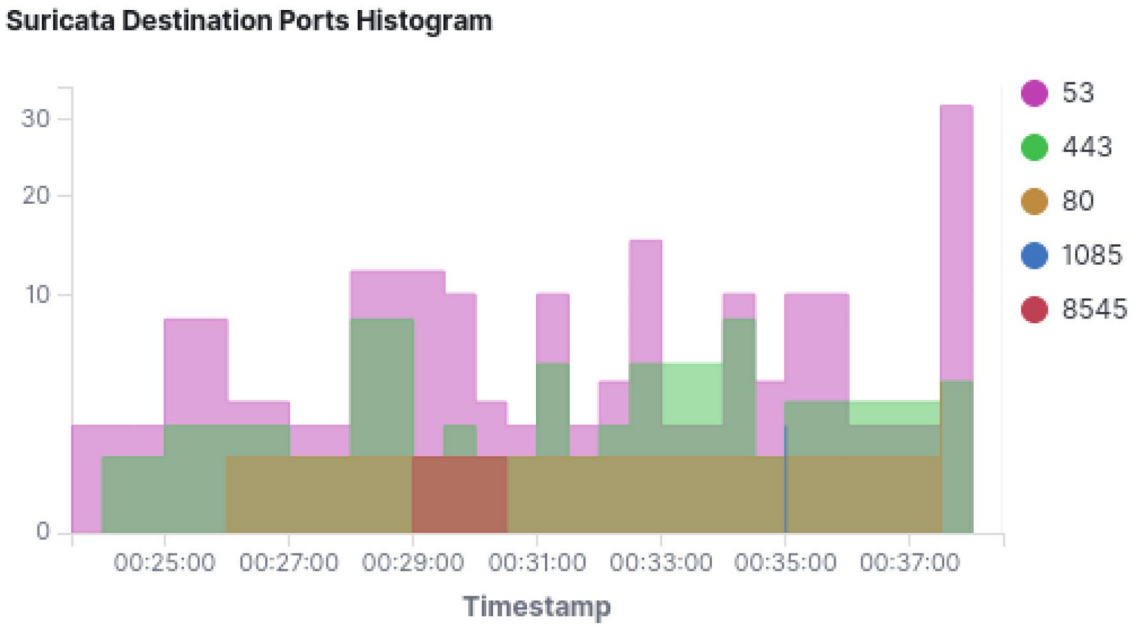


Fig. 7 Example alerts signatures (top) and destination ports histogram (bottom)

organizations with critical infrastructures. The reason is that LogRhythm is an ideal tool for organizations with resource-restricted security teams requiring a high degree of automation and out-of-the-box content, which is not the case for organizations with critical IT and network operations that require manual intervention in their daily activities.

AT& T Cybersecurity[11] offers both commercial and open source SIEM solutions. The main drawback of this solution is the limited user or entity behavior analytics as well as machine learning capabilities. In addition, basic reaction capabilities (e.g., send email, execute script, open ticket) are supported by both OSSIM and USM solutions but limited to the pre-defined set of conditions associated with a given security policy. Also, *C-S Prelude*[12] is able to recover several types of logs (e.g., system logs, syslog, flat files, etc.) and benefits from a native support with a number of systems dedicated to enriching information even further (e.g., snort9, ossec10). The tool provides powerful correlation engines that help operators to identify suspicious behaviors in huge volumes of data. Real-time data analysis, forensics, and research of APT (advanced persistent threats) are, therefore, intuitive and fast. Although Prelude is able to alert operators

[11] https://cybersecurity.att.com/

[12] http://www.prelude-siem.com

when an anomalous event is detected, reaction capabilities are limited to notifications and the execution of basic scripts based on correlation results. In addition, no user behavior or machine learning capabilities have been integrated to the current available versions.

Splunk[13] offers the capability to capture, monitor and report on data from systems, applications and security devices, and gives admins the possibility to quickly investigate raised issues and resolve security threats across different domains (e.g., network protection domain, access domain). One of the main limitations of Splunk is the use of basic predefined correlation rules for monitoring and reporting requirements. In addition, license models are based on data volume in gigabytes indexed per day, which makes this a costlier solution than other SIEM products where high volumes are expected. Sufficient planning and prioritization of data sources is required to avoid unnecessary consumption of data volumes. *Exabeam*[14] allows analysts to collect unlimited log data, use behavioral analytics to detect attacks, and automate incident response. Exabeam provides granular role-based data access and workflow to support privacy concerns which can be predefined and customized. Other tools that are less advanced as the ones mentioned above include *Micro Focus*[15], *Fortinet*[16], and *Solarwinds*[17].

We differ in the manner we employ (i) machine learning techniques to learn and extract potentially cross-layer hidden patterns and (ii) unsupervised outlier detection techniques to detect at runtime anomalous behavior. More importantly, our techniques can be seen as complementary to the ones already in practice, i.e., the proposed KEA solution can be combined and extend the tools above.

Conclusions

Nowadays, the IT infrastructure in health organisations not only have to enforce strict security mechanisms, but also to prevent any disclosure of personal data. To achieve this, the system processes activity logs at different layers and relies on continuous training for threats, since it is impossible to have defined all possible threat patterns a priori. This work presents a novel methodology to mitigate cybersecurity and privacy risks in IT infrastructures, motivated by the afore-mentioned challenges in the health sector. Apart from employing standard techniques in modern intrusion detection systems, we employ (i) advanced data analytics to extract hidden patterns corresponding to new threats that span different network and system layers and (ii) unsupervised outlier detection in parallel with complex event processing. Moreover, we apply complex event processing in a CAPEC-compliant manner, whereas our solution can leverage both logs and time series in training machine learning models for detecting unknown threats. Our methodology has been prototyped, made publicly available and can enhance capabilities of current solutions.

Acknowledgements The research work presented in this article has been supported by the European Commission under the H2020 Programme, through funding of the "CUREX: seCUre and pRivate hEalth data eXchange" project (G.A. id: 826404).

Compliance with ethical standards

Conflict of interest We declare a CoI against all partners working in Aristotle University of Thessaloniki and Suite5 and ATOS companies.

References

1. CAPEC, Common Attack Patterns Enumeration and Classification, A community resource for identifying and understanding attacks. https://capec.mitre.org/. Accessed 13 Sept 2019.
2. Matrix Profile. https://www.cs.ucr.edu/~eamonn/MatrixProfile.html. Accessed 16 Oct 2019.
3. SELKS. https://github.com/StamusNetworks/SELKS/wiki. Accessed 20 Sept 2019.
4. Aggarwal CC. Data mining: the textbook. Berlin: Springer; 2015.
5. Aggarwal CC. Outlier analysis. 2nd ed. Berlin: Springer; 2018.
6. Ahmed M, Mahmood AN, Hu J. A survey of network anomaly detection techniques. J Netw Comput Appl. 2016;60:19–31.
7. Amudha P, Karthik S, Sivakumari S. Classification techniques for intrusion detection—an overview 2013;
8. Breiman L. Random forests. Mach Learning. 2001;45(1):5–32.
9. Butun I, Morgera SD, Sankar R. A survey of intrusion detection systems in wireless sensor networks. IEEE Commun Surveys Tutorials. 2014;16(1):266–82.
10. Dayarathna M, Perera S. Recent advancements in event processing. ACM Comput Surv. 2018;51(2):33:1–36.
11. Díaz-Honrubia AJ, González AR, Zamorano JM, Jiménez JR, Gonzalez Granadillo G, Diaz R, Konidi M, Papachristou P, Nifakos S, Kougka G, Gounaris A. An overview of the CUREX platform. In: 32nd IEEE international symposium on computer-based

[13] https://www.splunk.com/en_us/siem-security-information-and-event-management.html

[14] https://www.exabeam.com/product/

[15] https://www.microfocus.com/en-us/home

[16] https://www.fortinet.com/products/siem/fortisiem.html

[17] https://www.solarwinds.com/security-event-manager

medical systems, CBMS 2019, Cordoba, Spain, June 5-7, 2019, 2019;162–167.
12. Faysel MA, Haque S. Towards cyber defense: research in intrusion detection and intrusion prevention systems 2010.
13. García-Teodoro P, Díaz-Verdejo J, Maciá-Fernández G, Vázquez E. Anomaly-based network intrusion detection: techniques, systems and challenges. Comput Secur. 2009;28:18–28.
14. Gartner: Magic Quadrant SIEM Report (2018). https://virtualizationandstorage.files.wordpress.com/2018/03/magic-quadrant-for-security-information-and-event-3-dec-2018.pdf 2018. Accessed 28 Aug 2019.
15. Gupta A, Birkner R, Canini M, Feamster N, Mac-Stoker C, Willinger W. Network monitoring as a streaming analytics problem. In: Proceedings of the 15th ACM workshop on hot topics in networks, 2016;106–112.
16. Harper A, VanDyke S, Blask C, Harris S, Miller D. Security Information and Event Management (SIEM) Implementation. : McGraw-Hill Osborne Media; 2010.
17. Heady R, Luger G, Maccabe A, Servilla M. The architecture of a network level intrusion detection system 1990.
18. Hindy H, Brosset D, Bayne E, Seeam A, Tachtatzis C, Atkinson RC, Bellekens XJA. A taxonomy and survey of intrusion detection system design techniques, network threats and datasets. ArXiv **abs/1806.03517** 2018.
19. Kendall KR. A database of computer attacks for the evaluation of intrusion detection systems 1999.
20. Kontaki M, Gounaris A, Papadopoulos AN, Tsichlas K, Manolopoulos Y. Continuous monitoring of distance-based outliers over data streams. In: 2011 IEEE 27th international conference on data engineering, Hannover, pp 135–146.
21. Kontaki M, Gounaris A, Papadopoulos AN, Tsichlas K, Manolopoulos Y. Efficient and flexible algorithms for monitoring distance-based outliers over data streams. Inform Syst. 2016;55(C):37–53.
22. Korvesis P, Besseau S, Vazirgiannis M. Predictive maintenance in aviation: Failure prediction from post-flight reports. In: 34th IEEE international conference on data engineering, ICDE 2018, Paris, France, April 16-19, 2018, 2018;1414–1422.
23. Laskov P, Düssel P, Schäfer C, Rieck K. Learning intrusion detection: Supervised or unsupervised? 2005;50–57.
24. Lewis K. Endpoint security. Computer and Information Security Handbook 2017;1049–1055.
25. Louppe G, Wehenkel L, Sutera A, Geurts P. Understanding variable importances in forests of randomized trees. In: Proceedings of the 26th international conference on neural information processing systems, Volume 1, NIPS'13, 2013;431–439.
26. Mishra P, Varadharajan V, Tupakula U, Pilli ES. A detailed investigation and analysis of using machine learning techniques for intrusion detection. IEEE Commun Surveys Tutorials. 2019;21:686–728.
27. Nadiammai G, Hemalatha M. Effective approach toward intrusion detection system using data mining techniques. Egypt Inform J. 2014;15(1):37–50.
28. Naskos A, Gounaris A. Efficiency assessment of event-based predictive maintenance in industry 4.0. In: Advances in data mining—applications and theoretical aspects, 19th industrial conference, ICDM 2019, New York, USA, July 17–July 21, 2019, 2019;103–117.
29. Naskos A, Kougka G, Toliopoulos T, Gounaris A, Vamvalis C, Caljouw D. Event-based predictive maintenance on top of sensor data in a real industry 4 . 0 case study. In: ECML/PKDD workshop on IoT Stream for Data Driven Predictive Maintenance 2019.
30. Toliopoulos T, Gounaris A, Tsichlas K, Papadopoulos AN, Sampaio S. Parallel continuous outlier mining in streaming data. In: 2018 IEEE 5th international conference on data science and advanced analytics (DSAA) 2018;227–236.
31. Tran L, Fan L, Shahabi C. Distance-based outlier detection in data streams. Proc VLDB Endow. 2016;9(12):1089–100.
32. Verizon: Data Breach Investigations Report (2019). https://www.cs.ucr.edu/~eamonn/MatrixProfile.html. Accessed 1 Nov 2019.
33. Welch DJ, Lathrop S. Wireless security threat taxonomy. IEEE Syst Man Cybernet Soci Inform Assurance Workshop. 2003;2003:76–83.
34. Yeh CM, Zhu Y, Ulanova L, Begum N, Ding Y, Dau HA, Silva DF, Mueen A, Keogh E. Matrix profile i: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In: 2016 IEEE 16th international conference on data mining (ICDM), 2016;1317–1322.