Open Access

Open Journal of Big Data (OJBD)
Volume 6, Issue 1, 2022
http://www.ronpub.com/ojbd
ISSN 2365-029X

A SIEM Architecture for Advanced Anomaly Detection

Tim Laue[A], Timo Klecker[B], Carsten Kleiner[A], Kai-Oliver Detken[B]

[A]Hochschule Hannover - University of Applied Sciences and Arts, Ricklinger Stadtweg 120, 30459 Hannover, Germany, {tim.laue | carsten.kleiner }@hs-hannover.de
[B]DECOIT GmbH, Fahrenheitstraße 9, 28359 Bremen, Germany {klecker | detken }@decoit.de

Abstract

Dramatic increases in the number of cyber security attacks and breaches toward businesses and organizations have been experienced in recent years. The negative impacts of these breaches not only cause the stealing and compromising of sensitive information, malfunctioning of network devices, disruption of everyday operations, financial damage to the attacked business or organization itself, but also may navigate to peer businesses/organizations in the same industry. Therefore, prevention and early detection of these attacks play a significant role in the continuity of operations in IT-dependent organizations. At the same time detection of various types of attacks has become extremely difficult as attacks get more sophisticated, distributed and enabled by Artificial Intelligence (AI). Detection and handling of these attacks require sophisticated intrusion detection systems which run on powerful hardware and are administered by highly experienced security staff. Yet, these resources are costly to employ, especially for small and medium-sized enterprises (SMEs). To address these issues, we developed an architecture -within the GLACIER project- that can be realized as an in-house operated Security Information Event Management (SIEM) system for SMEs. It is affordable for SMEs as it is solely based on free and open-source components and thus does not require any licensing fees. Moreover, it is a Self-Contained System (SCS) and does not require too much management effort. It requires short configuration and learning phases after which it can be self-contained as long as the monitored infrastructure is stable (apart from a reaction to the generated alerts which may be outsourced to a service provider in SMEs, if necessary). Another main benefit of this system is to supply data to advanced detection algorithms, such as multidimensional analysis algorithms, in addition to traditional SIEM-specific tasks like data collection, normalization, enrichment, and storage. It supports the application of novel methods to detect security-related anomalies. The most distinct feature of this system that differentiates it from similar solutions in the market is its user feedback capability. Detected anomalies are displayed in a Graphical User Interface (GUI) to the security staff who are allowed to give feedback for anomalies. Subsequently, this feedback is utilized to fine-tune the anomaly detection algorithm. In addition, this GUI also provides access to network actors for quick incident responses. The system in general is suitable for both Information Technology (IT) and Operational Technology (OT) environments, while the detection algorithm must be specifically trained for each of these environments individually.

Type of Paper and Keywords

Regular research paper: *SIEM, IDS, architecture, multi-dimensional data, open source, security*

1 INTRODUCTION

The increasing integration of traditional IT components and production/control systems (operational technology, OT) creates new risks for the companies. So, in addition to state-of-the-art security systems such as firewalls and malware protection, log collection, monitoring, and analysis systems have become recent and frequently followed trends in the field of attack detection. Today's attackers can easily surpass the traditional signature-based and rule-based protection and install malicious code on the devices within companies' networks inconspicuously. According to the BITKOM study [9], the number and complexity of cyber-attacks are constantly increasing. This study reveals that 70% of the German economy in 2019 is affected by digital attacks, at the same time a 27% increase is observed compared to two years ago. Additionally, sensitive data has been stolen from one in five surveyed companies. It is expected that these numbers are higher in reality, because companies do not always report all breaches. Additionally, most companies have not established sufficient intrusion detection/prevention systems (IDS/IPS) or SIEM systems to detect attacks and breaches.

The intrusion of an attacker can be detected through unusual system or application behavior and abnormal network communication. However, detection usually requires the aggregation and correlation of data from different systems for further analysis. The huge volume of data also poses particular challenges for intrusion detection systems. Limited availability of benchmark datasets for various applications and network logs prevent a comprehensive comparison of intrusion detection algorithms and products. Moreover, most available datasets used for evaluation are artificial or specifically tailored to the detection method. The lack of sufficient data sets also leads to the non-transferability of learned anomaly detection models among different environments. It is almost impossible to model all attack classes. Some attacks can take months to unfold. On the other hand, attack scenarios continue to evolve which makes the definition of the attack classes obsolete. The problems for IDS outlined in [29] can be generalized to the detection of security incidents. To overcome these problems, a SIEM system should correlate information from heterogeneous sources to provide the Security Operation Center (SOC) staff with a holistic network overview. SIEM systems should be regarded as a central platform to collect and correlate logs and events from conventional security tools like IDS/IPS. SIEM systems also should surpass traditional detection systems by focusing on user-friendliness and the detection of relevant anomalies by precisely reducing the number of alarms in IDS/IPS systems. Regardless of the known shortcomings of signature-based attack detection, they are still solely used in many organizations. Because anomaly detection algorithms are usually very costly and difficult to configure.

The GLACIER project [11] tries to address all of the aforementioned aspects in an integrated system. In particular, it will provide the following features:

a. Unification and consolidation of log information

b. Horizontal scalability

c. Anomaly detection for automated intrusion detection (as improvement over signature-based detection)

d. Development of novel multidimensional anomaly detection algorithms

e. Visualization of the anomaly results

GLACIER provides improvements over typical commercial SIEM systems in the market by means of the following novel and unique features:

- Extremely flexible, configurable and highly scalable components in the data collection chain: thus, the system can be applied to diverse network environments (office, industry production, etc.) and adapted to arbitrary sized environments by parallelizing the collection components. The scalability is also valid for the data storage component by using the highly scalable Elastic Search engine (`https://www.elastic.co/`).
- Pluggable data analysis component: it enables the integration of external and independent attack detection algorithms, if desired. However, the results with our specific cellwise estimator detection engine are very promising.
- User feedback: it enables automated training and improvement of the detection algorithm by a feedback mechanism that empowers domain experts to provide feedback on the alerts in order to improve the detection capabilities of the analysis engine in the future.
- Flexible and configurable replay capabilities: it enables reuse of the previous event streams in order to train new analysis models or test whether previous attacks would later be detectable by an improved analysis engine model.
- Normal-behavior-based analysis: it enables detection of previously unknown and zero-day attacks.

- All components of the system can be controlled by a central component, the Component Controller.
- The system is based on open-source components.

A review of prototypes in the related literature and off-the-shelf commercial SIEM systems in Section 2 reveals that GLACIER is far beyond the reviewed prototypes and the combination of its novel features surpasses other SIEM systems in the market. Most importantly, it supports an easy integration of any anomaly detection algorithm (might be developed in the future) into GLACIER, only by replacing the Analysis Engine component, as long as all source data needed for the algorithm is collected in the data provisioning chain and the algorithm satisfies near real-time detection time requirements.

In this paper after reviewing some related work in Section 2, we will describe the architecture of the proposed system and explain some of the components in more detail in Section 3. We will also discuss some of the design choices there. To illustrate how the architecture will operate when detecting attacks, we describe some use cases in Section 4. An extensive experimental evaluation of the architecture and its integration into a real-world IT and OT environments of a large city freshwater provider in both its office and production networks is described, analyzed and discussed in Section 5. We finally conclude in Section 6.1 and give some ideas for future improvements in Section 6.2.

2 Related Work

In the field of IT security, research has long been conducted on intrusion detection systems (IDS), which examine network data and recognize attack patterns [17]. A distinction can mainly be made between signature-based and anomaly-based methods. Signature-based methods are limited to previously known and recorded attack scenarios, while anomaly-based methods analyse normal behaviour and detect deviations and can thus also detect previously unknown attacks [28]. However, anomaly-based methods usually have the disadvantage of producing a high number of false positives.

In order to detect complex attacks, a comprehensive view of all security-relevant data is necessary. SIEM systems are precisely used for this purpose by performing data integration and evaluation. Static rules or anomaly detection can also be used at this level. In contrast to IDS, SIEM must be able to handle much more heterogeneous data and higher data volumes [29]. There are already several publications (e.g., [22], [16], [14]) that use standard data mining methods such as cluster analysis to improve attack detection. However, all these methods apply to homogeneous data (trained and tested using the homogeneous database of IDS systems) and cannot be applied to heterogeneous data, as is the case in SIEM-like systems.

Anomaly detection methods have been used in different domains than IT security ([1]). In addition to the basic techniques, we are particularly interested in methods that can detect contextual or collective anomalies (see [5]). An example is a star-cubing method presented in [27], which efficiently calculates all cube cells that exceed a certain threshold value. However, these methods must also be able to be used in data streams and must be efficient enough to enable online detection of incidents. Furthermore, the cells of a cube can also be interpreted as time series. Therefore, some types of time-series algorithms can be used for anomaly detection (see [12]) and groupings of data.

In the OnLine Analytical Processing (OLAP) environment, there are also various studies on the multidimensional visualization of data cubes [18] [26] [19], which mostly use established methods such as scatter plots, radar charts or parallel coordinates for the visualization of multivariate data (for example, data which is logged by a firewall and represents characteristics of a VPN connection). The challenge in the GLACIER project is to present multidimensional visualization of data from time-dependent data streams in a comprehensible way, as well (for example, data that shows Geo-IP information of the user who makes VPN connections in one week). To the best of our knowledge, there is not any research study that considers the combination of both advanced representations (multivariate and multidimensional) of network security.

In [7], a cloud computing adoption framework was presented for securing cloud data and the framework was evaluated later in [6]. In contrast to [7], our framework is specifically designed for SMEs and can operate under the on-premise model in most cases. Detailed knowledge about the environment to be monitored in these cases proofs beneficial for the detection behavior.

Analyzing user behavior (sometimes also known as User Entity Behavior Analysis, UEBA) is also another interesting field of research for detecting cybersecurity-related anomalies and many publications address UEBA. [21] developed an OLAP Cube data model for event log data. Based on this, multidimensional statistical tests were applied to data to detect anomalies. Our proposed architecture utilizes a novel detection engine which is also based on an OLAP Cube data model. It is described in [13] and follows a general concept roughly proposed in [8]. However, our model is at a much higher level of detail and provides a concrete implementation. Similarly, [3] also introduced a multidimensional detection approach based on tensors which showed good results in specific situations. Also, [4] introduced an analysis framework with good detection quality which is suitable for high-velocity data streams, but requires a fixed and predefined set of features for detection. However, this approach is also primarily appropriate for smaller numbers of dimensions.

In general, our proposed architecture is highly agnostic of the specific detection method. This is because detection algorithms inside the **Analysis Engine** component can be easily exchanged with other detection frameworks or external detection models can be added to the Engine. The evaluation presented in Section 5 is based on our own cellwise estimator approach ([13]).

From a more commercial point of view, there are different varieties of SIEMs in the market. The manufacturers of commercial SIEMs have also recognized that systems with fixed rules and regulations (first generation SIEM) are too rigid and personnel-intensive in maintenance and development from the customer's point of view. So, they have started to add new features to traditional SIEMs. The following part of this section presents systems and services related to modern SIEMs (aka second-generation SIEM) which are grouped based on their properties/methods.

Static sets of correlation rules: a list of logical expressions is defined by providers in advance and updated in specific intervals, e.g., monthly; dynamic lists much more frequently. Each rule is responsible to trigger a specific action, if a particular event occurs. For example, if a specific user logs into the same computer with several IP addresses which have different Geographic IP Informatio (GEO info) within 1 hour, the SIEM must raise an alarm and block the user. A rule also can be very simple like comparing traffic features against dynamic lists of suspicious objects (e.g., IP addresses, URLs, hashes of binary code). Exabeam SIEM, IBM QRadar SIEM, ArcSight SIEM, Elastic SIEM, Tenable LCE and McAfee Enterprise Security SIEM, all have such rule-based anomaly detection features. The lists of suspicious objects or the behavior patterns depicted in the rules ("threat intelligence") are obtained by the manufacturers through a wide variety of methods (e.g., manual searches, honeypots, statistical analyses across several customers, analysis of unstructured texts such as postings in darknet). Often a permanent connection to the providers' systems (SIEM providers) is necessary. In this case, the company's data needs to be transmitted to the service provider for analysis or for correlation with other customers. This solution usually is not favorable to companies/businesses, because it does not always meet the requirements of GDPR-compliance.

Statistical time series analysis: baselines of individual metrics (e.g., user numbers, network bandwidth) are captured dynamically over time. A baseline of each metric is regarded as its "normal state" and updated dynamically. Consequently, significant deviations from the normal states are called anomalies. The monitored metrics (numerical values) and threshold values for deviation have to be defined by the IT administrators in each organization based on their expert knowledge of the typical system behavior. This is also a non-favorable solution, because IT administrators usually do not have enough time and sometimes even expertise to adjust all required metrics and thresholds which are needed by these methods. In some cases thresholds can also be defined and updated dynamically [10] (e.g., assigning a moving average to a threshold). Such capabilities are found in many commercial products, including Exabeam SIEM, IBM QRadar SIEM, and Elastic SIEM.

User and Entity Behavior Analysis (UEBA): creates models of normal behavior for individual users or components such as IP addresses, servers, and applications using statistical analysis or learning methods to detect deviations from the normal state. According to the Gartner analysis [23], machine learning methods (supervised/unsupervised ML) are increasingly used in addition to rule-based and statistical approaches for UEBA. For example, machine learning approaches select metrics for individual event data fields (e.g., authentication processes, activities in applications), whose temporal development and correlation are considered [20]. Such techniques are used in several products including IBM QRadar UBA App, Exabeam SIEM, LogRhythm UEBA, ArcSight UBA, DarkTrace Enterprise. According to [2] UEBA methods are crucial for Security Operation Centers (SOC). They produce fewer alarms when compare to event-based analysis. This is particularly important for SMEs where the number of SOC personnel is extremely small or even outsourced (therefore, fewer alarm rate results in reducing SOC costs for SMEs). Detailed information about UEBA and their importance can be found in [15] or [24]. Unfortunately, there is only very limited number of UEBA methods in the open-source community (e. g. [25]), as stated in [15]. Our proposed architecture including the detection algorithm which also leverages UEBA concepts is based on open-source components and is thus a step to fill the gap.

The next section explains other innovations of GLACIER's architecture in more detail.

3 Description of the Proposed Architecture

This section starts with a general description of the architecture of GLACIER and is followed by detailed descriptions of its individual parts. We also describe how GLACIER achieves the goals promised in Section 1.

The proposed architecture ensures horizontal scalability by designing each component (excluding GUIs) to be suitable for containerization, which we realize using Docker. In this paper, rounded rectangles inside figures depict components that run inside Docker. Layered rounded rectangles indicate that there are multiple parallel implementations of the component. Dotted rectangles denote data flow between components. Control flow is mostly omitted. However, if the demonstration of control flows is necessary, they are represented by dotted ellipses.

3.1 Overview

Figure 1 displays an overview of component groups of the GLACIER architecture, surrounding systems, and their interaction with each other.

The **Data Collection** is responsible for gathering data from heterogeneous **Dynamic Sources** and consolidating it, if necessary for security analysis. These sources can be any network component that produces events suitable for monitoring, like hosts, firewalls or OT components. Afterwards, events are normalized to a common format, enriched and archived. During enrichment, the system utilizes context information from **Static Sources**, like lightweight-directory access protocol servers (LDAP), configuration management databases (CMDB) or IP geolocation services. Archived data can be viewed using the **Audit Graphical User Interface GUI**.

Enriched data is forwarded to the **Data Analysis** component for anomaly detection analyses. These can be visualized in the **SIEM GUI** alongside training data and learned models. The GUI also supports immediate feedback for analyses and also user reaction to incidents by means of (e.g., Network Access Control (NAC) interfaces or Common Vulnerabilities and Exposures (CVE) scanners) in the network. In addition, **Automatic Actors** can be triggered (e.g., to send notifications to security staff) without user involvement. All GLACIER components are configured and supervised by **Management**. The **Management GUI** is used for administrative tasks and supports using the **Replay** functionality to recreate previously encountered situations in the network by replaying events. This part of GLACIER is not part of the system offered to customers, since it mostly serves to conduct experiments for evaluating the performance of the data processing chain and the analysis algorithms. Thus, it is primarily used by the GLACIER team internally for research and improvements.

3.2 Data Collection

This section will describe the components of the data collection group in Figure 2. The components in this part of the system collect event data from different points in the network, pre-process and archive it, and finally pass it on to data analysis.

At many points within the architecture, **Brokers** are used for buffering event messages to decouple different stages of data processing, thus enabling horizontal scalability. These brokers do not necessarily run in separate containers, instead, they are realized as different topics in the same RabbitMQ (an open-source message-broker software used internally). The message queues within the brokers hold their messages in memory to keep throughput as high as possible, unless they are flagged as important, in which case they will be stored on disc as well.

Archivers are components that insert enriched data into databases. For each new insertion, they test whether the data is already present and overwrite it, if it is. This behavior is different for the **Raw Archiver**, which has to avoid overwriting enriched events with their raw counterparts in the archive. Each dynamic source has an **Agent** (a parser) for collecting and forwarding its data. **Agent** converts any non-textual data to a text-based, structured JSON format in the process. However, at this stage, the JSON objects will be mostly flat. More effort will be spent to structure data at the normalization stage. Agents can be actively polling for data or passively receiving it, depending on the source type. They will also summarize and discard certain events according to configuration to minimize the load on the system at the source. To guide the normalization process, agents append their own type to events.

The **Raw Filter** gathers the raw events produced by the agents, giving each event an ID that uniquely identifies it across the remainder of the system. Additionally, it provides a second opportunity for filtering the event stream. Most notably it holds a whitelist of agents known to the system, discarding events stemming from unregistered sources.

Each agent (or agent type) has a **Normalizer**

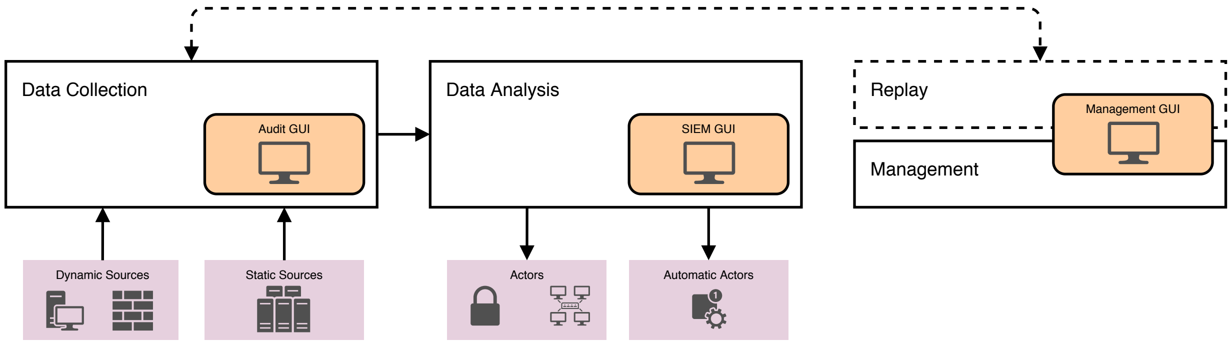


Figure 1: Overview of component groups of the GLACIER architecture

task to transform its JSON output into a common format, thereby integrating data from all sources, while enforcing data quality constraints. **Normalizer** also assigns common names for features that convey the same meaning across all data sources (e.g., assigning SRC_IPaddr for fields that include the IP address of the source). We will use a well-documented and portable format that is available in the Elastic Common Schema (ECS). Each normalizer appends its own type and timestamp to the normalized event to make the normalization process reversible and repeatable, which is useful when normalizers or the data format are changed or when errors occur in the process.

The **Enricher** fetches context information (i.e. user-related data from directory services such as LDAP or Active Directory, domain information and IP/MAC-related information from services) and attaches it to events. This results in both completing the attribute list of events and creating dimensional hierarchies on top of some of these attributes. Similar to the normalizers, it appends information about the enrichment process to the event to make it repeatable.

To reduce the load on static sources, enrichers store the most recent history of context data they retrieve in the **Cache**. This cache has been implemented as a Redis (an open-source, in-memory key-value data store) instance.

Enriched Filter can be configured to filter out events that require information for the analysis which will only become available after enrichment.

The **Asset Listener** gathers enriched event data and infers a list of active assets in the network, which are then forwarded to the analysis database for display in the SIEM GUI.

Long-term storage of events is handled by the **Archive**. It stores all events passing through the system, both in raw format and after enrichment and is realized as an ElasticSearch instance.

The data in the archive can be viewed using the **Audit GUI**. This can serve for compliance, forensics or simply for double-checking analysis results. Since data is stored in ElasticSearch, we have integrated Kibana (a visualization tool by Elastic Stack to present data from ElasticSearch) into our framework for visualization purposes. It only requires minimal additional effort.

3.3 Data Analysis

The components in this part of the system are responsible for analyzing the gathered events, detecting anomalies and presenting them to users as shown in Figure 3.

Events enter the data analysis chain through the Event Store. This database contains a relatively short history of fully enriched events and offers high-bandwidth access for near real-time analysis. This database is implemented using ElasticSearch as well.

The **Event Store Cleaner** is an active component for deleting any entries in the event store that lie outside the desired time window for analysis.

Any data surrounding data analysis results is stored in the **Analysis Database**. For this, a traditional relational database system is required which is available on the open-source market in PostgreSQL which has thus been chosen for the implementation.

The **Analysis Engine** is responsible for finding anomalies in the event stream, as well as hosting experiments with anomaly detection algorithms. Detected anomalies are assigned an anomaly score and an ID, and are treated as incidents in the following parts of the data analysis. A notification is sent for each incident to trigger automatic actors. Additionally, a flag in each event is analyzed to determine whether it represents an incident on its own already. This may e. g., be the case for antivirus software notifications which have been provided by agents.

The analysis engine currently utilizes a novel machine-learning algorithm, which uses OLAP cubes as an underlying data structure (for more detail, please refer to [13]). Remarkably, the architecture facilitates easy exchange of this method by other detection algorithms,

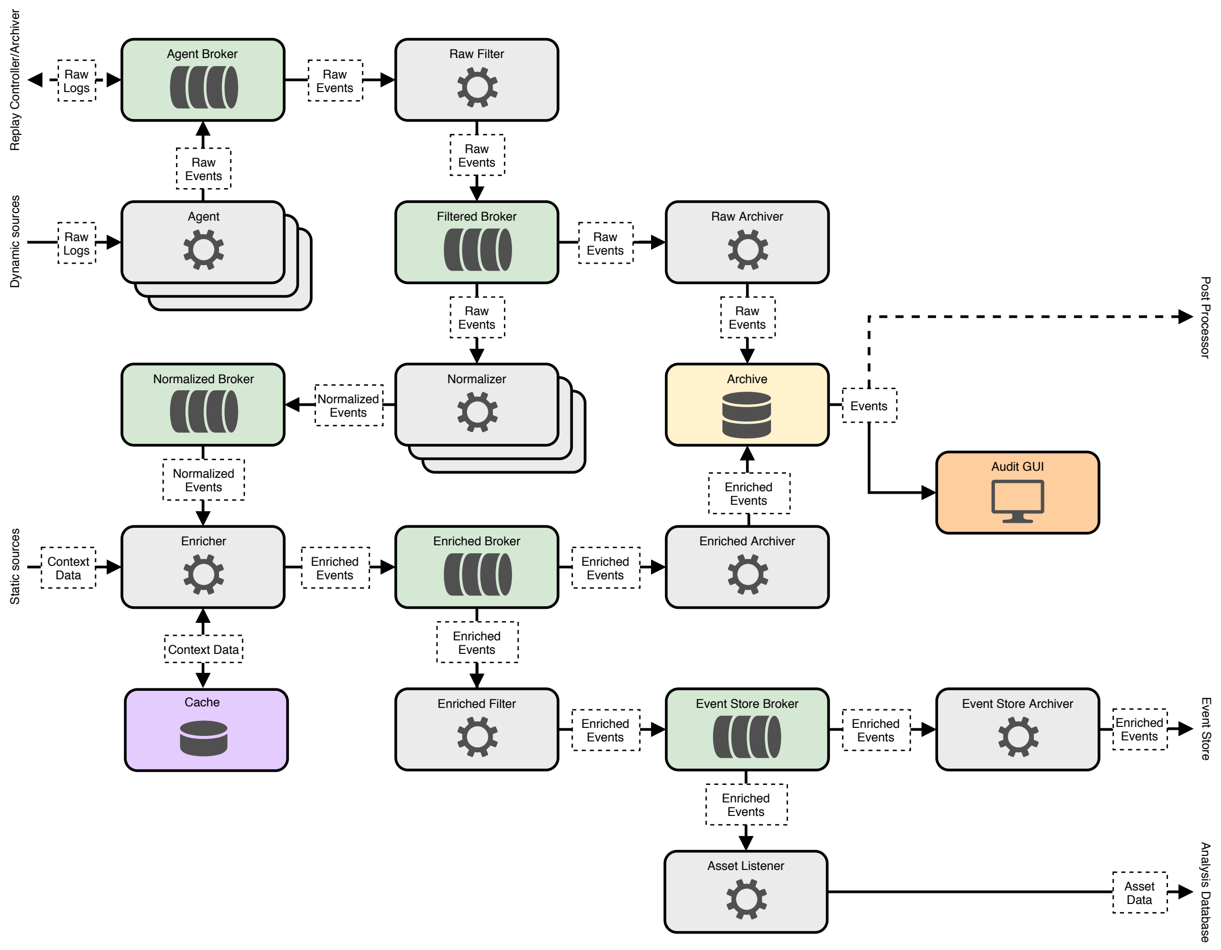


Figure 2: Components of the data collection process

if desired, as the interface of this component is well documented and slim. In order to analyze events, they are first separated into time slices and aggregated into cubes. Machine learning models, cubes obtained from training, inference data and user feedback are stored in the analysis database and retrieved, if needed. All stored pieces are required, especially in the case that the models need to be retrained on updated training data or new feedback.

The **Post Processor** takes incident data produced by the analysis engine and enriches it for display in the SIEM GUI. Enriched incident information partly achieved by querying the archive database and by incorporating user feedback. Fully processed incidents are stored in the analysis database.

Incidents are visualized in the SIEM GUI. In addition to showing the information describing an incident, it is possible to access related incidents, for example, incidents that share at least one attribute value (e.g., incident events which have a common destination port) with one another. This can be cross-referenced with a visualization of network assets, which is built from the asset data provided by the asset listener. The GUI gives users recommendations for reacting to incidents and access to actors who carry out the reactions (e.g., moving a host to quarantine). Users can also give feedback to the analysis engine on each incident. This includes adjusting the anomaly score to the desired value or tagging incidents with labels that will be shown as descriptive text on similar or duplicated incidents in the future. The history of actions that are taken by users in conjunction with an incident is stored as a ticket in the analysis database.

3.4 Management & Replay

In this section, the last two parts of the architecture will be described. The first is Management which allows for starting, stopping, and configuring other GLACIER components. The second is the Replay chain which stores raw events and reintroduces them into the data

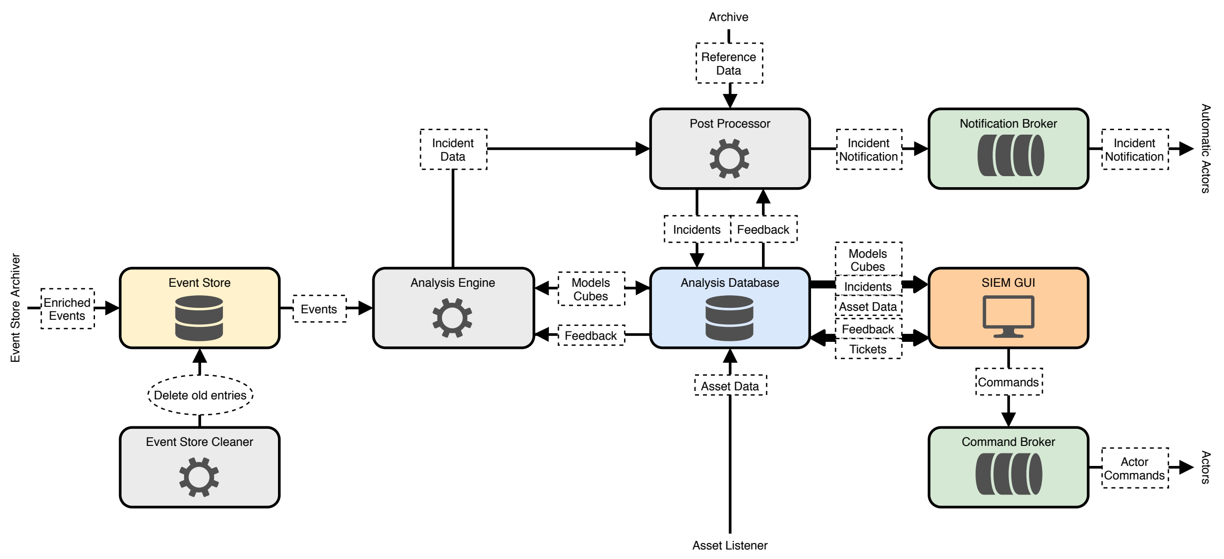


Figure 3: Components of the data analysis chain

collection stream, if requested. Both component groups are shown in Figure 4.

The **Management GUI** lies at the core of the GLACIER management. Here, users are provided with an overview of the GLACIER system, showing all active components and their current status. Status information is passed down from the components themselves in the form of regular heartbeats. Additionally, it enables users to send commands to these components to manage their lifecycle or configuration. Static component configurations and heartbeat data are persisted in the **Configuration Store**.

Commands issued via the GUI are distributed by the **Component Manager**, which retrieves configurations as needed and sends instructions to the **Command Listener**. It also triggers updates of the docker repository.

The first component deployed on each individual GLACIER host is the **Component Controller**. This component is in charge of starting and stopping other components in their own containers on the host in question while managing their settings by relaying instructions from the **Component Manager**. The images used for building containers are stored in the **Docker Repository**.

Raw events from the data collection chain are stored in the **Replay Archive**, which is implemented using ElasticSearch. The **Replay Controller** can then send events back to the agent queue, should a replay be requested. Replayed events should appear to the system as naturally occurring events.

4 Use Cases

For use cases, there can be different actors with different requirements for a system. For example, systems with different authorization levels can distinguish between administrators and users who can perform disjoint actions and would therefore have completely different use cases. A distinction between different actors within a company does not seem to make sense for our system, since the users can always perform the same actions. Only the goals of the system can differ, but it can be assumed that the overlaps are so extensive that no further distinction is necessary. An example would be using of the system by an experienced administrator and a relatively inexperienced administrator. Due to the self-learning GUI, the actual use of the interface might differ, but the requirements for the system do not differ significantly. The defined use cases encompass the use of the system, advantages of using the system, and economic aspects and they result from the following requirements:

a. Fulfill security requirements to be legally secure

b. Take IT security measures quickly and easily with low costs and little know-how

c. Detect threats to remove vulnerabilities

d. Perform active scans to assess the current threat situation

e. Carry out passive monitoring in order to be alerted to dangers (anomalies) to be able to react

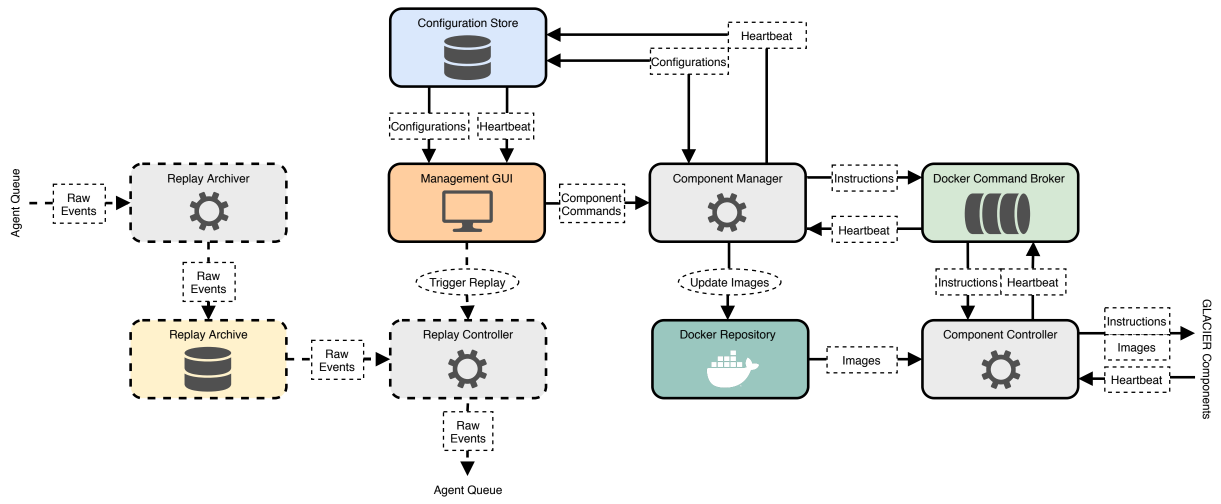


Figure 4: Components of management and replay

f. Access historical events to create and evaluate statistics

g. Resource-saving monitoring preventing adverse effects on the stability and speed of the network

h. Perform quick scans to investigate current threats on the network or to initiate investigations in case of a specific incident

i. Continuous monitoring to detect long-term threats

j. Easy to understand and use GUI to quickly identify risks and changes

k. Central control and coordination function to minimize the support effort

l. Aggregate information to understand risks, incidents and vulnerabilities

m. Generate understandable recommendations for action so IT administrators can immediately decide how to react to an incident

n. Usability in OT and/or IT networks

o. Receive regular reports to assess the current status and compare it with older reports

p. Respond promptly to threats to meet security requirements

The above requirements describe the behavior of the system in an abstract way without dealing with technical aspects. From them, the following technical use cases could be inferred:

a. Use Case 1: Definition of communication rules (hosts, networks, time restrictions) for detecting violations

b. Use Case 2: Analysis of logs (Windows event logs and Syslog)

c. Use Case 3: File integrity monitoring (access via SSH or agents on the corresponding system)

d. Use Case 4: Detection of failed SSH logins

e. Use Case 5: Vulnerability Scan

f. Use Case 6: Malware detection in network communication

g. Use Case 7: Login attempts on Windows server systems

h. Use Case 8: Detecting new network connections

i. Use Case 9: Detecting new protocols within the network

These user scenarios originated from associated partners of the GLACIER project and were compiled in the course of the analysis of the current state and requirements definition. The GLACIER architecture intensely depends on the diversity of its use case scenarios. The more scenarios are implemented, the more incidents can be detected and displayed in the **SIEM-GUI**. We select Use Case 6 to illustrate the feasibility of its implementation in a detailed manner and apply the GLACIER system architecture for it as presented earlier in this paper.

Malware detection in network communication has already been implemented in several products. However, most of these systems attempt to identify malware by matching the actual network traffic with well-known malware attack patterns. This approach works well for known attacks, however, it will never be able to identify previously unknown (zero-day) attacks. From a risk perspective, such novel attacks constitute the most dangerous type of attacks and thus high priority should be given to detecting them. In the GLACIER system, such novel attacks shall be identified by their deviations from the regular system behavior. For example, in this use case 6, via unusual network communication. For this purpose, the system has to learn the regular system behavior over a period of time, and then a near real-time check for deviations can be performed.

In terms of the high-level architecture of Figure 1 the Data Analysis component will be responsible for learning the regular system behavior, as well as for the real-time detection of deviations. In order to force the Analytical Component to learn the regular system behavior fast, the Replay component can be used to feed the system with historical records of observed system usage patterns faster than receiving new data about a network communication in real-time. Technically, this is implemented by feeding the historical usage data into the **Data Collection** component and propagating them to the **Analysis** component, as if they were actual records. To be able to use the learned regular behavior to detect deviations, the actual network usage data is observed and consolidated in the **Data Collection** component and then forwarded to the **Analysis** component for detection.

For implementing use case 6 by means of learning the regular behavior and detecting deviations from a normal background, the **Data Collection** (cf. Fig. 2) component requires more information. **Data Collection** should receive static information about the network (e.g., user-related information from LDAP) as well as dynamic information (e.g., actual network connections and flows). Dynamic data sources are the most important information providers to the **Analysis** component. They provide basic information about the current usage of the network (e.g., a login event to a specific machine). However, when the system is in detection mode, these events can be filtered for relevance to reduce the load on the **Analysis** component (e.g., information for unimportant machines might be dropped). In addition, based on log information provided by the sources, events can be forwarded to the **Replay** component to record them for future training phases. After normalizing the remaining events into a unified format suitable for the **Analysis** component, the events might be enriched. Static data is fed into the system via the **Enricher** which is able to correlate this information with dynamic events. This might be required to match actual login events with the (static) priority of the user on that machine.

Enriched events with such static information can thereafter be forwarded to the **Analysis** component through the **Event Store Broker**. **Event Store Broker** provides events to the **Asset Listener** and **Event Store** which are then forwarded to the **Analysis Database**, correspondingly. Both are the data providers of the **Analysis Engine** as shown in Fig. 2.

The **Analysis Engine** can now check individual events or groups of events against the learned regular system behavior and store them in the **Analysis Database** in order to detect potential abnormalities. If suspicious issues are detected (e. g., login and external data transfer from a server that is usually only accessed internally), the engine will create an **Incident**. This **Incident** can be a potential malware attack that has been detected. It needs further examination by a knowledgeable security operator via the **SIEM GUI** or automated response (e.g., disconnection of the server from the public network) by **Automated Actors**. In addition, the **SIEM GUI** can be used to collect feedback about this incident and the message created with this incident in order to continuously improve the analysis component. In the initialization phase, raw events should be propagated to the management and replay component for a certain amount of time and stored in the **Replay Archive**. After collecting sufficient data of raw events, the dataset that is stored in the **Replay Archive** (cf. Fig. 4) is used to start training a new multidimensional behavior model from the **Management GUI** by means of the **Replay Controller**. In this case, the analysis component is not used for actual detection but to learn the regular baseline system behavior. The data processing pipeline, however, is similar to the detection case described above.

Based on the explained aspects for Use Case 6, the system architecture of the GLACIER system has proven feasible and appropriate. The other use cases can be similarly handled by the system. To cover a wide spectrum of use cases, models that keep track of regular system behavior should be as diverse and specific at the same time, as possible. This requires integration of many different data sources for events into the collection component indicating the need for a highly scalable architecture at this collection end.

5 EXPERIMENTAL EVALUATION

5.1 Field Test

A prototype of GLACIER was tested in a field test at an associated project partner to verify the developments, eliminate errors, and prove the effectiveness. The following test objectives were defined:

a. Data is collected by sensors (both active and passive)

b. Data is examined according to incidents

c. The multidimensional analysis engine is analyzed

d. The performance of the proposed architecture is analyzed in live operation

e. All components are inspected for errors

Two time-frames were defined for the field tests. In the first time-frame, training was carried out to model the normal behavior of the network. This was performed in both the IT and OT environments for one week in each case. Results of the training were then integrated into the prototype and one month later an analysis phase was performed based on the training data. The used hardware platform was a hardware-appliance with 64 GB RAM, 1 TB SSD and 1 Gbit/s network interfaces. Two and three mirror ports were used in the OT and IT environment correspondingly. Various analyses were done in the overall evaluation (see Fig. 5). First, the incidents (tickets) were viewed in the **SIEM-GUI** (see Fig. 6). Then, an evaluation was performed using Kibana. Finally, Check_MK was used to analyze the performance and the active scans were evaluated again directly with OpenVAS.

The recorded data was examined after completion of the analysis phase. A total of 40,000 incidents (tickets) were accumulated in the OT environment within two weeks, while only 3,300 incidents were detected in the IT environment. The incidents were detected by the GLACIER specific detection engine based on a multidimensional analysis of network traffic. Detected anomalies might be simply related to unusual high/low volumes of network traffic from/to a certain host. They can also incorporate additional analysis dimensions such as an unusual high amount of traffic to a certain set of hosts over a certain network protocol that does not belong to the usual application protocols. Details about the analysis engine can be found in [13]. A sample of the detected anomalies is shown in Fig. 7. The upper section of the figure is part of the anomaly cube detected by the analysis engine. The textual representation includes the anomaly itself with the computed anomaly score and an explanation as to why it is considered an anomaly. The cube hierarchy can be navigated by opening/closing certain levels. In the bottom of Fig. 7 a graphical representation of a configurable part of the anomaly cube is displayed (For example, a user can filter out unfavorable incidents or false positives). Both the data to be displayed and visual options can be configured by means of User Interface (UI) elements.

5.2 Discussion

In relation to the defined technical use cases in section 4, the use cases number 6, 8 and 9 can be considered as solved using the employed version of the analysis engine. Use cases number 1 and 4 can be considered partially solved. In general, use cases 2, 3 and 7 could be detected by the system, but they require additional detection components (or a different configuration and training of the current engine). Finally, use case number 5 was solved by the system in general, but the OpenVAS scanner was also employed as an additional detection component.

The system recorded a relatively large amount of IPv6 traffic and Domain Network System (DNS) queries in the OT environment. Both were unusual behavior because IPv6 is not actually used internally and DNS queries should remain internal. However, there were too many DNS queries that were often used for querying public DNS servers from Google and others. In addition, there were IPv6 addresses that could be sighted in both IT and OT networks, although there was no direct connection between them. Finally, ports 137, 138 and 139 were observed in the tickets. This might be a sign of activity of NetBIOS in the network. However, NetBIOS is currently considered obsolete and should no longer be used. The detected usage was probably related to the old operating systems in the OT environment.

In the OT environment, 284 million logs were recorded within the test period of two weeks. That was 20 GB of data per day. There were only minor fluctuations in daily usage. On the other hand, there were 120 million logs for the IT environment corresponding to 16 GB. As expected, there were noticeable fluctuations in daily usage in the IT environment due to users' working hours and breaks. These fluctuations were interpreted as normal. Simple Network Management Protocol (SNMP) was the most used protocol as system monitoring checked the hosts every 60-120 seconds. The frequency of the SNMP protocol was followed by Hypertext Transfer Protocol (HTTP) and Discovery and Basic Configuration-Remote Procedure Call (DCP-RPC) protocols.

In the **Check_MK** evaluation, the OT environment reached an average load of 60%, compared to 40% in the IT environment. The **Enrichers** had little to do in both environments. This was also true for the other SIEM components. The analysis engine in the IT environment had much more to do with a load of 22% in contrast to the OT environment with a load of 5%. The fluctuations in CPU utilization in Fig. 8 and Fig. 9 also illustrate once again the differences between IT and OT networks. This could also be seen from the Random Access Memory (RAM) consumption that was -on average- 80% (51 GB)

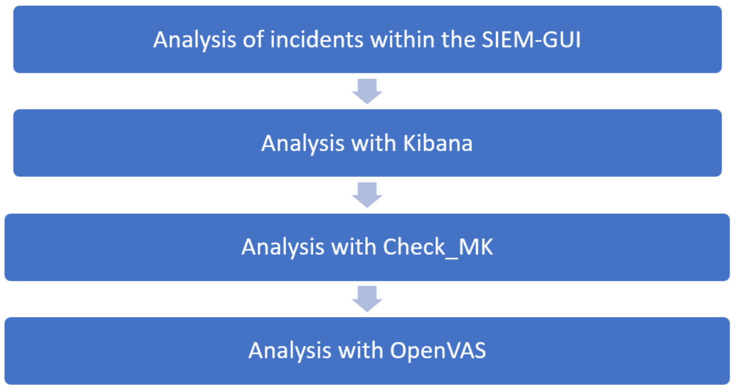
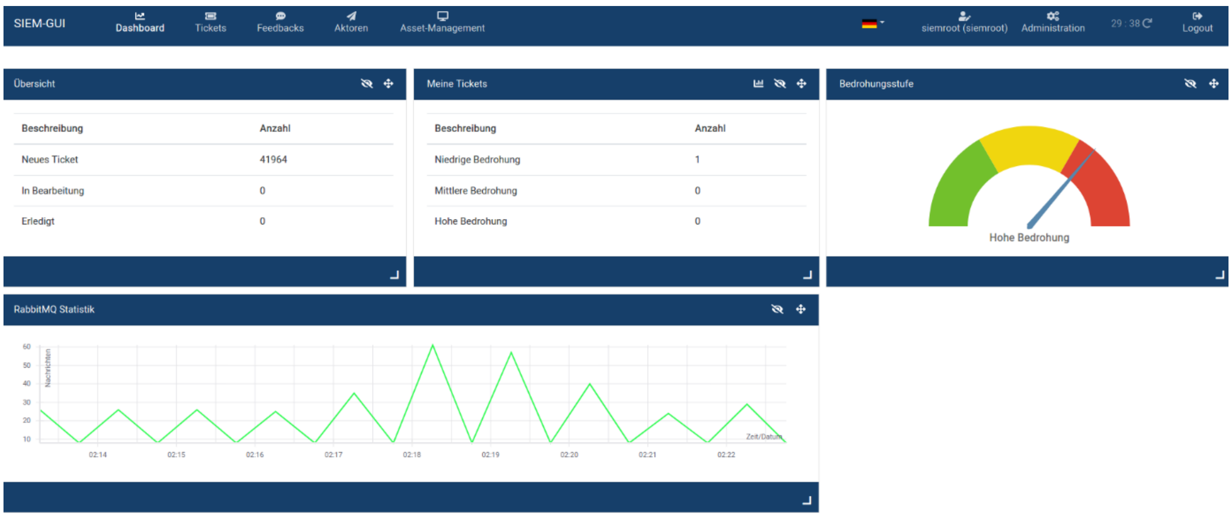


Figure 5: Schedule of the evaluation



Übersicht

Beschreibung	Anzahl
Neues Ticket	41964
In Bearbeitung	0
Erledigt	0

Meine Tickets

Beschreibung	Anzahl
Niedrige Bedrohung	1
Mittlere Bedrohung	0
Hohe Bedrohung	0

Figure 6: General ticket overview in the SIEM-GUI

in the OT environment and 36% in the IT environment.

Finally, the active scans with OpenVAS could still find some interesting incidents. For example, certificates using outdated or too weak signature algorithms were detected. In some cases, Telnet was still active and too weak and unused keys were found. It was also possible to observe outdated software versions or operating systems that were no longer supported. Overall, all test targets defined by the project partners have been achieved, while the performance of the used hardware appliance was sufficiently powerful. There was sufficient scalability available and the prototype ran stably.

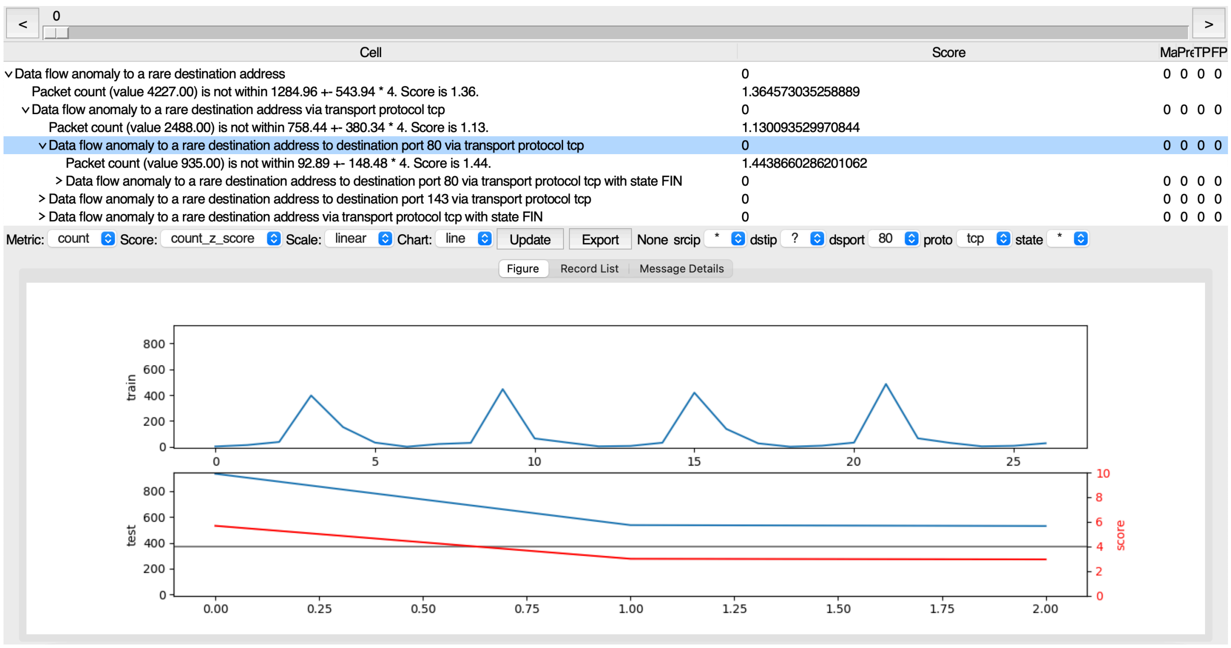
The developed analysis engine found too many incidents (40,000) within two weeks. However, it was often not possible to derive any recommendation for action, because the descriptions of the incidents were not precise enough. Nonetheless, the applicability and usefulness of the analysis engine have been demonstrated. Its efficiency (detecting less but more accurate incidents) can be improved using more training data, manual feedback and with more experiments how to model the cube, esp. in the more regular OT environment.

Therefore, we need to integrate more sensors and log data from other security systems. Indeed, an increase in the data volume will increase the need for memory, which should be taken into account in the future.

6 Conclusion and Future Work

6.1 Conclusion

In this paper, we have introduced a system architecture to implement an advanced security incident and event management system. The predominant advantages of our approach are the great flexibility and utilization of license cost-free software components. The system is highly flexible. Different types of components can be incorporated into the system to collect generated



Cell	Score	Ma	Pre	TP	FP
˅ Data flow anomaly to a rare destination address	0	0	0	0	0
Packet count (value 4227.00) is not within 1284.96 +- 543.94 * 4. Score is 1.36.	1.364573035258889				
˅ Data flow anomaly to a rare destination address via transport protocol tcp	0	0	0	0	0
Packet count (value 2488.00) is not within 758.44 +- 380.34 * 4. Score is 1.13.	1.130093529970844				
˅ Data flow anomaly to a rare destination address to destination port 80 via transport protocol tcp	0	0	0	0	0
Packet count (value 935.00) is not within 92.89 +- 148.48 * 4. Score is 1.44.	1.4438660286201062				
> Data flow anomaly to a rare destination address to destination port 80 via transport protocol tcp with state FIN	0	0	0	0	0
> Data flow anomaly to a rare destination address to destination port 143 via transport protocol tcp	0	0	0	0	0
> Data flow anomaly to a rare destination address via transport protocol tcp with state FIN	0	0	0	0	0

Metric: count Score: count_z_score Scale: linear Chart: line Update Export None srcip * dstip ? dsport 80 proto tcp state *

Figure 7: Sample of UI showing detected anomalies

data from various sensors. This can be realized through the data collection pipeline including **Brokers**, **Normalizers** and **Enrichers**. Moreover, the incident detection component can also be expanded as required. We propose an analysis engine based on an innovative multidimensional data analysis method that is capable of detecting incidents. It also provides valuable explanations regarding the reason for labeling an event as an incident. Other traditional detection engines could also be added to our proposed architecture, for example, OpenVAS which is shown in the case study. Finally, flexibility concerning the application domain and detection use cases are among other important features of this architecture.

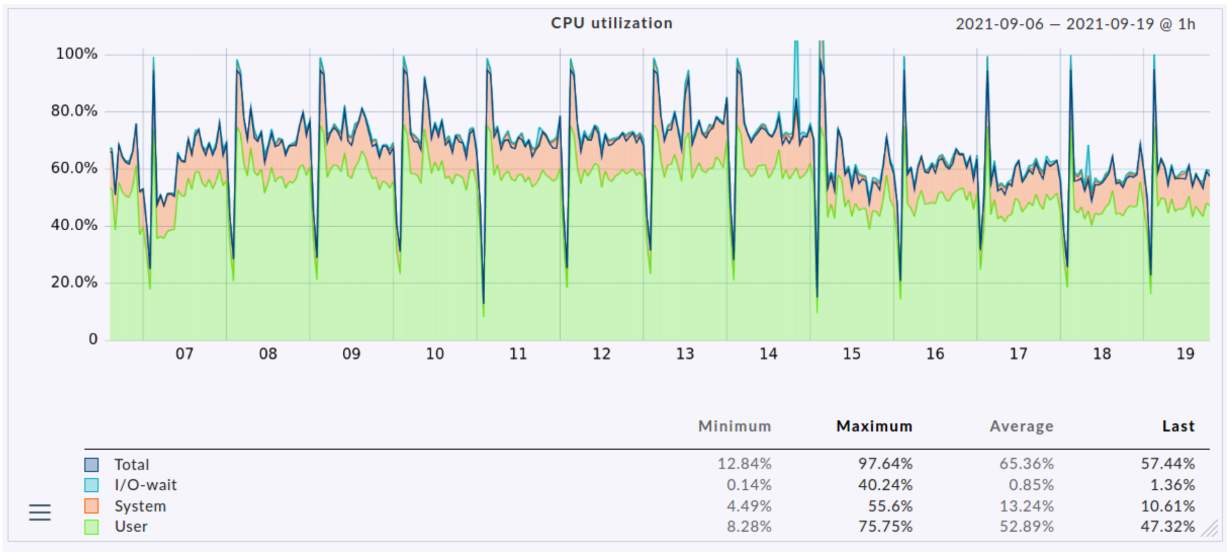
After elaborating on the components of the proposed architecture, we also defined typical use cases that can be solved based on this architecture. We put a special emphasis on SME use cases because they tend to demand solutions with low operational costs. Our architecture provides this due to its open-source approach coupled with easy-to-use and easy-to-understand explanations for the detected incidents after a comprehensive training phase.

Finally, we illustrated the feasibility of the architecture and approach by presenting results from a successful field study that was performed in the both OT and IT networks of a German large city freshwater provider. The field study confirms both the flexibility and adaptability of the architecture as well as its domain agnostic aspect.

6.2 Future Work

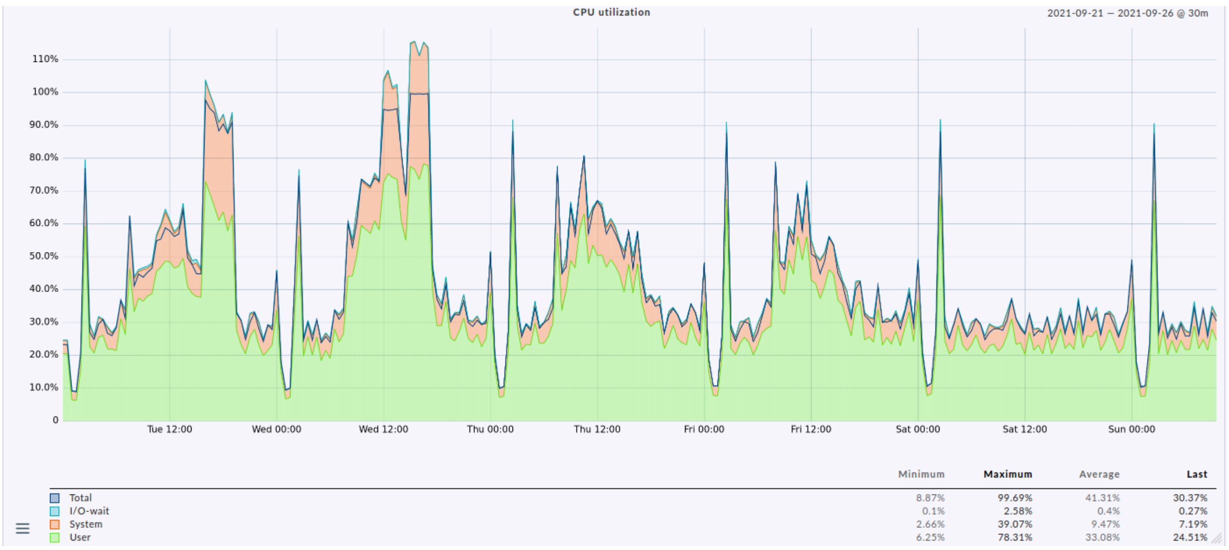
Our proposed architecture incorporates the required features for security-based anomaly detection in IT and OT environments which is presented in section 1. However, some components of the architecture still need to be improved, which we intend to do in currently active projects based on the GLACIER project. For instance, in a subsequent project, the architecture will be applied to the domain of virtual power plants to detect potential security incidents in this domain.

For the data collection chain, a vertical subset of all planned components has already been implemented that were required for the field study. This delivered valuable insights for realizing the remaining variants of some components continuously. In particular, to improve anomaly detection results, more sensors have to be added to the system to provide further information for modeling the normal system state and consequently for analyzing potential deviations. Additionally, the current version of the proposed framework still reports a high number of incidents which puts a high burden on the SOC personnel even though the detailed explanations of the reasons causing the anomaly speed up the processing of the incidents by SOC personnel significantly. Therefore, we intend to fine-tune the number of notified incidents (especially, reducing false-positive alarms) by an improved training phase. Lab tests (cf. [13]) already revealed the possibility to reduce false alarms. We plan to remedy this problem using an



	Minimum	Maximum	Average	Last
Total	12.84%	97.64%	65.36%	57.44%
I/O-wait	0.14%	40.24%	0.85%	1.36%
System	4.49%	55.6%	13.24%	10.61%
User	8.28%	75.75%	52.89%	47.32%

Figure 8: CPU utilization within the OT environment



	Minimum	Maximum	Average	Last
Total	8.87%	99.69%	41.31%	30.37%
I/O-wait	0.1%	2.58%	0.4%	0.27%
System	2.66%	39.07%	9.47%	7.19%
User	6.25%	78.31%	33.08%	24.51%

Figure 9: CPU utilization within the IT environment

improved configuration and a longer training phase in the real network (both IT and OT networks).

For the data analysis component group, the focus will be on improving the analysis engine at its core in the future. The initial version of the analysis engine which is based on a multidimensional cube-based analysis of data to detect anomalies has to be extended and adapted to different event types, particularly for industrial scenarios. Moreover, the analysis algorithms need to be improved. It requires a systematic evaluation of the algorithms on a rich set of event types in traditional IT environments. More specifically, Use Case 2 from section 4 (in conjunction with 4 and 7) will be implemented and evaluated in the near future. It also requires a correlation of these results (from Use Case 2 and 4) with the results from the case study illustrated above (Use Case 6).

In addition, the rather complex results from the multidimensional analysis have to be presented to the security operator in a usable manner. To realize this, we plan to extend the SIEM GUI to demonstrate an improved visualization of the incidents (as the main focus) and the corresponding generated explanations for the incidents (as a second focus) in the future.

The main components of the management and replay part of the architecture are already operational. In the near future, the focus will be on generating typical replay scenarios for both traditional IT environments and industrial environments. While initial replay scenarios have already been generated, more realistic and comprehensive cause-effect analysis will be essential for improving the management and replay component.

Acknowledgements

The authors would like to express their thanks to the anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper. The authors would also like to thank the German Federal Ministry of Education and Research (BMBF) for the financial support as well as all other partners involved in the research project GLACIER for their great collaboration. The project consists of the industrial partners rt-solutions.de GmbH, DECOIT® GmbH, and the research partner University of Applied Sciences and Arts of Hanover with the security group Trust@HSH. A special appreciation goes to the associated partner hanseWasser GmbH for its support and the possibility of prototype testing in their IT and OT infrastructure.

References

[1] C. C. Aggarwal, “Outlier Analysis,” in *Data Mining: The Textbook*. Cham: Springer International Publishing, 2015, pp. 237–263.

[2] O. Akinrolabu, I. Agrafiotis, and A. Erola, “The challenge of detecting sophisticated attacks: Insights from soc analysts,” in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, ser. ARES 2018. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3230833.3233280

[3] D. Bruns-Smith, M. M. Baskaran, J. Ezick, T. Henretty, and R. Lethin, “Cyber security through multidimensional data decompositions,” in *2016 Cybersecurity Symposium (CYBERSEC)*, 2016, pp. 59–67.

[4] P. Casas, F. Soro, J. Vanerio, G. Settanni, and A. D’Alconzo, “Network security and anomaly detection with big-dama, a big data analytics framework,” in *2017 IEEE 6th International Conference on Cloud Networking (CloudNet)*, 2017, pp. 1–7.

[5] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” *ACM Comput. Surv.*, vol. 41, 07 2009.

[6] V. Chang, Y.-H. Kuo, and M. Ramachandran, “Cloud computing adoption framework: A security framework for business clouds,” *Future Generation Computer Systems*, vol. 57, pp. 24–41, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X15003118

[7] V. Chang and M. Ramachandran, “Towards achieving data security with the cloud computing adoption framework,” *IEEE Transactions on Services Computing*, vol. 9, no. 1, pp. 138–151, 2016.

[8] A. A. Cárdenas, P. K. Manadhata, and S. P. Rajan, “Big data analytics for security,” *IEEE Security Privacy*, vol. 11, no. 6, pp. 74–76, 2013.

[9] M. Fiedler, “Cyberangriffe: dramatische zunahme und rekordschäden,” procontra-online.de, Alsterspree Verlag GmbH, Berlin, 11 2019, accessed: 2020-04-23.

[10] R. F. Fouladi, O. Ermiş, and E. Anarim, “A ddos attack detection and defense scheme using time-series analysis for sdn,” *Journal of Information Security and Applications*, vol. 54, p. 102587, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214212620307560

[11] “Glacier (angriffserkennung durch multidimensionale analyse sicherheitsrelevanter datenströme) project homepage,” https://www.glacier-project.de/index.php/home-eng.html, accessed: 2020-04-14.

[12] M. Gupta, J. Gao, C. C. Aggarwal, and J. Han, “Outlier detection for temporal data,” in *Synthesis Lectures on Data Mining and Knowledge Discovery, Vol. 5, No. 1*, 3 2014, pp. 1–129.

[13] F. Heine, C. Kleiner, P. Klostermeyer, V. Ahlers, T. Laue, and N. Wellermann, “Detecting attacks in network traffic using normality models: The cellwise estimator,” in *Proceedings of the 14th International Symposium on Foundations & Practice of Security*, Dec 2021, p. TBD. [Online]. Available: TBD

[14] M. A. Jabbar, R. Aluvalu, and S. S. S. Reddy, “Cluster based ensemble classification for intrusion detection system,” in *Proceedings of the 9th International Conference on Machine Learning and Computing*, ser. ICMLC 2017. New York, NY, USA: Association for Computing Machinery, 2017, p. 253–257.

[15] S. Khaliq, Z. U. Abideen Tariq, and A. Masood, “Role of user and entity behavior analytics in detecting insider attacks,” in *2020 International*

Conference on Cyber Warfare and Security (ICCWS), 2020, pp. 1–6.

[16] X. Li, X. Zheng, J. Li, and S. Wang, "Frequent itemsets mining in network traffic data," *Proceedings - 2012 5th International Conference on Intelligent Computation Technology and Automation, ICICTA 2012*, 01 2012.

[17] H.-J. Liao, C.-H. R. Lin], Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[18] A. S. Maniatis, P. Vassiliadis, S. Skiadopoulos, and Y. Vassiliou, "Advanced visualization for olap," in *Proceedings of the 6th ACM International Workshop on Data Warehousing and OLAP*, ser. DOLAP '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 9–16.

[19] C. Ordonez, Z. Chen, and J. García-García, "Interactive exploration and visualization of olap cubes," in *Proceedings of the ACM 14th International Workshop on Data Warehousing and OLAP*, ser. DOLAP '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 83–88.

[20] M. Patel and R. Ramesh, "Stepping up your uba game with the machine learning app," https://developer.ibm.com/qradar/2018/04/16/uba-machine-learning-app/, 04 2018, accessed: 2020-04-23.

[21] T. S. Prarthana and N. D. Gangadhar, "User behaviour anomaly detection in multidimensional data," in *2017 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, 2017, pp. 3–10.

[22] Z. Qu and X. Wang, "Study of rough set and clustering algorithm in network security management," in *2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*, vol. 1, 2009, pp. 326–329.

[23] G. Sadowski, A. Litan, T. Bussa, and T. Phillips, "Gartner market guide for user and entity behavior analytics," https://www.gartner.com/doc/3872885/market-guide-user-entity-behavior, 04 2018, accessed: 2020-04-23.

[24] M. A. Salitin and A. H. Zolait, "The role of user entity behavior analytics to detect network attacks in real time," in *2018 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, 2018, pp. 1–5.

[25] M. Shashanka, M.-Y. Shen, and J. Wang, "User and entity behavior analytics for enterprise security," in *2016 IEEE International Conference on Big Data (Big Data)*, 2016, pp. 1867–1874.

[26] C. Stolte, D. Tang, and P. Hanrahan, "Multiscale visualization using data cubes," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 9, pp. 176– 187, 05 2003.

[27] D. Xin, J. Han, X. Li, Z. Shao, and B. W. Wah, "Computing iceberg cubes by top-down and bottom-up integration: The starcubing approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 1, pp. 111–126, 2007.

[28] Y. Yu, "A survey of anomaly intrusion detection techniques," *J. Comput. Sci. Coll.*, vol. 28, no. 1, p. 9–17, 10 2012.

[29] R. Zuech, T. M. Khoshgoftaar, and R. Wald, "Intrusion detection and Big Heterogeneous Data: a Survey," *Journal of Big Data*, vol. 2, no. 1, p. 3, 2 2015.

AUTHOR BIOGRAPHIES

M. Sc. Tim Laue graduated from the Hannover University of Applied Sciences and Arts in 2019, obtaining a masters degree in computer science with a focus on information security and software engineering. He has since worked at the same university as a research associate for the GLACIER project.

Dipl. Inf (FH) Timo Klecker received his degree in computer science at the University of Applied Science in Bremen in 2008. Since 2006 he worked at DECOIT GmbH as student employee in programming, resulting in permanent employment at DECOIT GmbH as programmer after obtaining the degree. His tasks then began to develop to managing software projects and designing software architectures. Today his field of activity includes head of development, software architecture, system design and project management for different projecs, among other things in the area of IT security.

Prof. Dr. Carsten Kleiner graduated from Leibniz-University in Hannover, Germany, in 1997 with a diploma degree in mathematics. He also holds a M. Sc. degree in computer science from Purdue University (1996). He has been promoted to Dr. rer. nat. at Leibniz-University in Hannover in 2003 based on a thesis in the area of database system optimization. Since 1997 he has been working as software developer, IT consultant and software project manager for different companies in the area of database applications. He became full professor for secure information systems at University of Applied Sciences&Arts in Hannover, Germany, in 2004. He is co-leader of the Trust@HsH research group and co-founder of the research cluster smart data analytics. He has been and continuously is involved in several funded research projects in the areas of data quality, data analysis and data management, particularly in the context of IT security applications.

Prof. Dr.-Ing. Kai-Oliver Detken graduated from the University of Bremen as an Electronics Engineer in 1993. After study he worked from 1993 till 1997 at the institute BIBA in Bremen as research scientist in EU-funded R&D projects. In 1998 he changed to the company OptiNet GmbH to manage industrial projects in his professional areas. In 2001 he founded his own company DECOIT GmbH and work as docent for Computer Science at the University of Applied Science in Bremen simultaneously. In 2003 he obtained his PhD degree and got from the University of Applied Science in Bremen the title professor in 2008. His working and research areas includes networks, Internet technologies, Voice over IP, and IT security.