The 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications
22-25 September, 2021, Cracow, Poland

A SIEM Architecture for Multidimensional Anomaly Detection

Tim Laue[1], Carsten Kleiner[1], Kai-Oliver Detken[2], Timo Klecker[2]
[1] University of Applied Sciences and Arts of Hanover, Ricklinger Stadtweg 120, D-30459 Hanover
{tim.laue/carsten.kleiner}@hs-hannover.de, https://www.hs-hannover.de
[2] DECOIT® GmbH, Fahrenheitstraße 9, D-28359 Bremen,
{detken/klecker}@decoit.de, https://www.decoit.de

In recent years businesses and organizations have experienced an increase in the occurrence of IT-security related threats, causing the compromise of sensitive information, disruption of everyday operations, and ultimately financial damage. Meanwhile, these attacks have become more varied and sophisticated, making them increasingly hard to detect. In order to address these issues we initiated the GLACIER[1]-project [1]. As a part of the project we created an architecture, which can be realized as an in-house operated SIEM system for SMEs. In addition to SIEM-specific tasks like network data collection, normalization, enrichment and storage, the systems main purpose is to supply data to advanced multidimensional analysis algorithms. These provide a novel way to reliably detect security-related anomalies. Found anomalies are displayed in a GUI, which allows giving feedback for tuning the anomaly detection algorithm, while also providing access to network actors for quick incidence responses. The architecture can be implemented using exclusively free, open-source components and is suitable for both information technology (IT) and operational technology (OT) environments.

Keywords: SIEM, intrusion detection, security architecture, multi-dimensional data, anaomaly detection, open source, security

I. Introduction

The increasing integration of traditional IT components and production/control systems (operational technology, OT) creates new risks that companies have to face. So, in addition to state-of-the-art security systems such as firewalls and malware protection, log and monitoring systems are becoming increasingly important. This is because every company must assume that professional attackers can overcome the existing perimeter protection with appropriate effort and that the malicious code used is not always reliably detected. The number and complexity of cyber-attacks is constantly increasing, as a BITKOM study [2] has shown. According to this study, 70% of the German economy is affected by digital attacks, compared to 43% two years ago. Sensitive data has been stolen from one in five companies, although leaks like these can probably not always be detected, so the number of unreported cases will likely be higher. This is because most companies have not established sufficient IDS/IPS or SIEM systems yet.

The intrusion of an attacker can only be detected through unusual system or application behavior and abnormal network communication. However, detection typically also requires that the data from different systems be aggregated and correlated in an analysis system. The large data volumes involved pose particular challenges. Due to the poor availability of current test data sets, new solutions cannot be evaluated comprehensively, which makes it massively more difficult to compare algorithms and products. The lack of sufficient data sets also leads to the fact that learned models for anomaly detection are not transferable. In addition, attack scenarios continue to evolve, making the definition of the attack class obsolete. Some attacks can take months to unfold. They are also difficult to model. The problems for intrusion detection systems (IDS) outlined in [3] can therefore be generalized to the detection of security incidents.

The goal of a SIEM system, on the other hand, should be to be able to correlate protocols from heterogeneous sources in order to provide the Security Operation Center (SOC) staff with a holistic network overview. They should therefore be regarded as a further development of conventional IDS/IPS systems. In order to prevent them from suffering the same niche existence as their predecessors, the focus should be on user-friendliness and the detection of relevant anomalies. It is precisely the number of ”false positives” in IDS/IPS systems that has led to them being used relatively little in today’s companies. However, since anomaly detection is currently still signature-based in most cases, new types of attacks are often noticed too late or not at all. Signature-based approaches determine anomalies by observing well-known attack scenarios (signatures) and are thus not capable of detecting previously unknown attack types.

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

a. Unification and consolidation of log information
b. Horizontal scalability

[1] GLACIER = Attack detection through multidimensional analysis of security-relevant data streams

c. Anomaly detection for automated intrusion detection
d. Development of novel multidimensional anomaly detection algorithms
e. Visualization of the anomaly results

II. Releated 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 [4]. A distinction can mainly be made between signature-based and anomaly-based methods. Signatures 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 [5]. However, anomaly-based methods usually have the disadvantage of producing a high number of false positives.

However, it has long been clear that a more comprehensive view of all security-relevant data is necessary in order to be able to identify more complex threat scenarios. SIEM systems are used for this purpose, which perform precisely this data integration and evaluation. Static rules or anomaly detection can also be used at this level. In contrast to IDS, SIEM must generally be able to handle much more heterogeneous data and larger data volumes [6]. There are already several publications (e.g. [7], [8], [9]) that use standard data mining methods such as cluster analysis to improve attack detection. However, they all have in common that they all start from the homogeneous database of an IDS system and cannot be applied to heterogeneous data, as is the case in SIEM-like systems.

Independently of the application in IT security, research has long been conducted on concepts for anomaly detection, see e.g. [10]. In addition to the basic techniques, we are particularly interested in methods that can detect contextual or collective anomalies (see [11]). An example is the star-cubing method presented in [12], 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, which means that suitable methods for time series anomaly detection (see [13]) can be applied to different groupings of data. The multidimensional anaomaly detection methods used in this project are based on the algorithm in [20].

In the OLAP environment, there are also various studies on the multidimensional visualization of data cubes [14] [15] [16], mostly using established methods for the visualization of multivariate data, such as scatter plots, radar charts or parallel coordinates. The challenge in the GLACIER project is therefore on the one hand to present multidimensional data from time-dependent data streams in a comprehensible way, and on the other hand to find visualization approaches that take into account the mental models of security analysts. Publications that apply the above advanced representations to network security data are hard to find so far. Especially the combination with the previously mentioned views has not been researched yet.

The market for commercial products offers a variety of options in the SIEM area. As the manufacturers of commercial systems have also recognized that systems with fixed rules and regulations (first generation SIEM) are too inflexible and too personnel-intensive in maintenance and development from the customer’s point of view, the following section looks at systems and services related to modern SIEM systems (second generation SIEM) grouped by properties / methods used:

a. *Static sets of rules maintained by the provider* that compare against dynamic lists of suspicious objects (e.g. IP addresses, URLs, hashes of binary code), e.g. IBM QRadar SIEM, Tenable LCE and McAfee Enterprise Security SIEM. The sets of rules are renewed in the course of updates, e.g. monthly, the dynamic lists much more frequently. 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).
b. *Statistical time series analysis of individual metrics* (e.g. user numbers, network bandwidth) to determine the development over time as a ”normal state”, dynamically update it and then detect significant deviations. The monitored metrics (numerical values) and threshold values for deviations must be defined by the IT administrator. This capability is found in many commercial products, including IBM QRadar SIEM.
c. *User and Entity Behavior Analysis (UEBA)* creates models of normal behavior for individual users or components such as IP addresses, servers, applications by means of statistical analysis or learning methods in order to detect deviations. According to the Gartner analysis [17], machine learning methods (supervised / unsupervised ML) are increasingly used in addition to rule-based and statistical approaches. These techniques are used in several products (e.g. IBM QRadar UBA App, LogRhythm UEBA, ArcSight UBA, DarkTrace Enterprise.

GLACIER is looking in detail at the third option for developing an open source based SIEM solution.

III. Approach

This section will present a description of the architecture and how it achieves the goals outlined in section 1, leading with an overview of the architecture, followed by detailed descriptions of its individual parts. Whenever there is a planned implementation for a concept or component it will be mentioned accordingly.

The architecture ensures horizontal scalability by designing each component (excluding GUIs) to be suitable for containerization, which we intend to realize using Docker. In the graphics rounded rectangles 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, while control flow is mostly omitted. If control flow is shown it is represented by dotted ellipses.

A. *Overview*

This section presents an overview of GLACIER major component groups, as well as the surrounding systems, and their interaction with each other. They are visualized in figure 1.

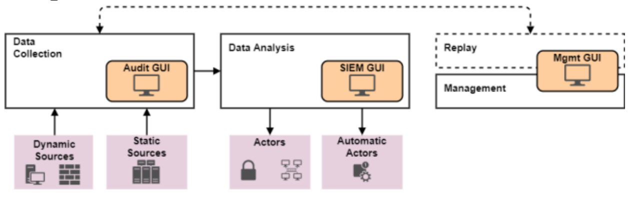

Figure 1. Overview of component groups of the GLACIER architecture

In *Data Collection* heterogeneous data is gathered from *Dynamic Sources* and consolidated as necessary for security analysis. These sources can be any network component that produces events suitable for monitoring, like hosts, firewalls or OT components. Events are normalized to a common format, enriched and archived. During enrichment the system utilizes context information from *Static Sources*, like LDAP servers, CMDBs or IP geolocation services. Archived data can be viewed using the *Audit GUI*.

Enriched data is forwarded to *Data Analysis*, where it is analyzed for anomalies. These can be visualized in the *SIEM GUI*, alongside training data and learned models. The GUI also allows giving feedback to the analysis algorithms and gives users ways to immediately react to incidents by engaging *Actors* in the network, like NAC interfaces or CVE scanners. 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 the management. In addition to administration, the *Management GUI* allows using the Replay functionality to recreate previously encountered situations in the network by replaying events. This part of GLACIER will not be included in the final version of the system, since it mostly serves conducting experiments to evaluate the performance of the data processing chain and the analysis algorithms.

B. *Data Collection*

This section will describe the components of the data collection group, as shown 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 to buffer events messages in order to decouple different stages of data processing, thus enabling horizontal scalability. These brokers do not necessarily run in separate containers, instead they will probably be realized as different topics in the same RabbitMQ instance. 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 to disc as well.

Archivers are components that insert data into databases. For each new insertion they test whether the data is already present and overwrite it if it is. This behaviour is different for the *Raw Archiver*, which has to avoid overwriting enriched events with their raw counterparts in the archive.

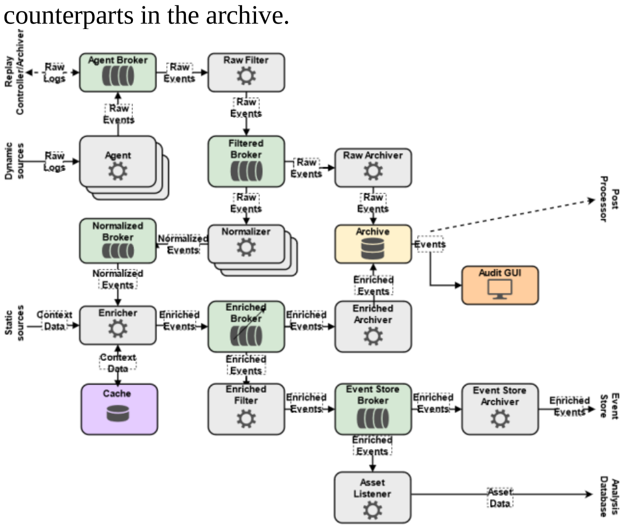

Figure 2. Components of the data collection process

Each dynamic source has an *Agent* collecting and forwarding its data, converting any non-textual data to a text based, structured format in the process. This format will be JSON, however at this stage the JSON objects will be mostly flat, with most of the effort to structure them will be concentrated at the normalization stage. Agents can be actively polling for data or passively receiving it, depending on the source type. They will also summarize 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 which 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* tasked with transforming its JSON output into a common format, thereby integrating data from all sources, while enforcing data quality constraints. We intend to use the Elastic Common Schema (ECS) for this purpose. Each normalizer appends its on type and a 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 ldap or ad, dns information or ip/mac relation) and attaches it to events, which serves 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 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 will be implemented as a Redis instance. In the case that events need to be filtered out before analysis on the basis of information that is available only after enrichment, the *Enriched Filter* can be configured to do so.

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 will be 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. Kibana, being the part of the ELK stack, is a natural choice for an implementation.

C. Data Analysis

The components in this part of the system are tasked with analyzing the gathered events for anomalies and presenting them to users. This component group is displayed 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 will be realized 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*. It will be implemented using PostgreSQL.

The component responsible for finding anomalies in the event stream, as well as hosting experiments with anomaly detection algorithms, is the *Analysis Engine*. Found anomalies are assigned an anomaly score and an ID, and are treated as incidents in the later parts of data analysis. A notification is sent for each incident to trigger automatic actors. Additionally, a flag in each event is read on input to determine whether it represents an incident independently, like for example antivirus software notifications. The analysis engine utilizes novel machine learning algorithms, which use OLAP cubes as underlying data structure. So in order to analyze events, they are first separated into time slices and aggregated to cubes. Machine learning models, cubed training and inference data and user feedback are stored in the analysis database and retrieved if needed. This is required especially in the case that the models need to be retrained on updated training data or new feedback.

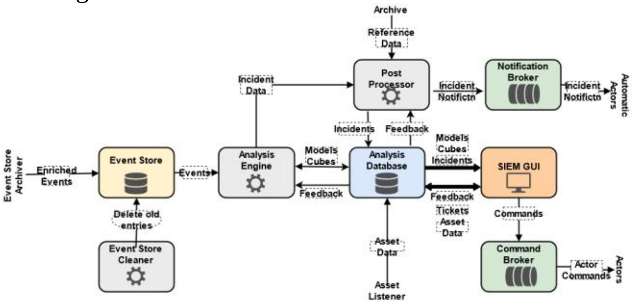

Figure 3. Components of the data analysis chain

The *Post Processor* takes incident data produced by the analysis engine and enriches it for display in the SIEM GUI, which is 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, i.e. incidents that share at least one attribute value 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 incidents and access to actors which carry out the reactions, for example by moving a host to quarantine. Users can also give feedback to the analysis engine on each incident, which can include adjusting the anomaly score to a desired value or tagging incidents with labels which will be shown as descriptive text on similar incidents in the future. The history of actions users take in conjunction with an incident are stored as a ticket in the analysis database.

IV. 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. In the system used here, a distinction into different actors within a company does not seem to make sense, 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 strong that no further distinction is necessary. An example would be the use of the system by an experienced administrator and a rather 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 use cases cover the use of the system, advantages of using the system and economic aspects. They result from the following requirements:

a. Fulfill security requirements in order 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 and be able to react
f. Access historical events to create and evaluate statistics
g. Resource-saving monitoring so as not to influence 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 that you can immediately decide how to react to an incident
n. Useability 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 requirements thus describe the behavior of the system in an abstract way without dealing with technical aspects. From these, however, the following technical use cases could be defined:

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 originate 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 lives from the multitude of its use case scenarios. The more scenarios are implemented, the more incidents can be detected and displayed in the SIEM-GUI.

At this point as a practical example use case 6 (malware detection) shall be detailed in order to illustrate the feasibility of implementation of this use case and at the same time motivate the GLACIER system architecture as presented earlier in this chapter.

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 and thus new kinds of attacks. From a risk perspective such novel attacks constitute the most dangerous type of attack and thus high priority should be given to detect those. In the GLACIER system such novel attack types shall be identified by realizing them as deviations from the regular system behavior, in this case network communication. For this purpose, the system needs to be able to learn the regular system behavior over a period of time and then a near real-time check for deviations can be realized.

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 online detection of deviations. In order to learn the regular system behavior as fast as possible, the Replay component can be used to feed the system with historical records of observed system usage patterns faster than in real time so that the Analysis component can learn the patterns fast. Technically, this is implemented by feeding the historical usage data into the *Data Collection* component and propagating those to the Analysis component as if they were actual records. In order 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.

Looking into more detail into the Data Collection component for this use case (cf. fig. 2) there will be a need to use static information about the network (e. g. user related information from LDAP) to be monitored as well as dynamic information (e. g. actual network connections and flows) in order to both learn the regular behavior and detect deviations. Most important are dynamic data sources that provide basic information about the current usage of the network, e. g. a login event to a specific machine. Based on log information provided by the sources events can be forwarded to the *Replay* component to record them for future training phases. If the system is in detection mode, these events can be filtered for relevance in order to reduce the load on the analysis component, e. g. information for unimportant machines might be dropped here. 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 by means of the *Event Store Broker* which provides them to the *Asset Listener* to be supplied to the *Event Store* (cf. fig. 3) which is the foundation for the *Analysis Engine*. The Analysis Engine can now check individual events or groups of events against the regular system behavior learned earlier and stored in the analysis database in order to detect potential abnormalities. In case potential issues have been detected, e. g. login and external data transfer from a server that is usually only accessed internally, the engine will create an Incident since a potential malware attack has been detected which needs to be either examined further by a knowledgeable security operator via the SIEM GUI or requires automated processing (e. g. disconnection the server from the public network) by Automated Actors. In addition, the SIEM GUI can be used to collect feedback regarding the quality of the message created with this incident in order to continuously improve the analysis component.

In order to learn the regular system behavior initially, raw events can be propagated to the management and replay component for a certain amount of time to be stored in the *Replay Archive* (cf. fig. 4). After sufficient data collection of raw events to be able to describe system behavior properly, the data set stored in the Replay Archive can be used to start training a new multidimensional system behavior model from the Management GUI with help of the *Replay Controller*. In this case, the analysis component is not used for actual detection but to learn the regular system behavior. The data processing pipeline, however, is similar to the detection case detailed above.

As a conclusion, other use cases can be handled by the system in a similar manner. To cover a wide spectrum of use cases the model for regular system behavior should be as diverse and specific as possible. This requires many different data sources for events to be integrated in the collection component explaining the need for a highly scalable architecture at this end.

V. Results of Experiments

To validate the developed architecture, tests were carried out in a real company environment. On the one hand, these tests served to perform functionality tests and, on the other hand, to minimize possible programming errors. In addition, the anomaly detection could be tested for its efficiency. Figure 4 shows the setup in the company network.

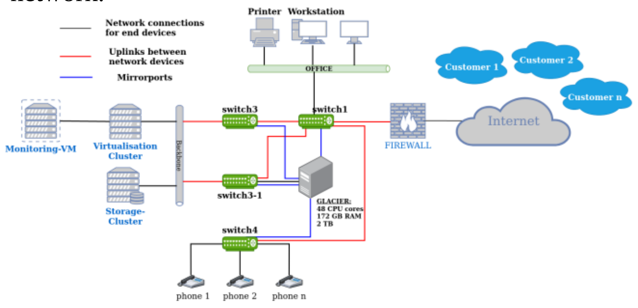

Figure 4. Test network structure

The tests were performed according to a specific scheme in order to obtain unambiguous results:

a. Running the installation script
b. Creating networks for asset management
c. Starting an vulnerability scan
d. Monitoring the components of availability
e. Collecting data using intrusion detection component
f. Starting the analysis engine (one week later)
g. Adjusting settings for the analysis algorithm and restarting with new training data from an entire week
h. Improving the analysis and restarting with training data from four weeks
i. Test of static analysis (define rules and create rule violations)

The results can be sorted into different categories: analysis engine, static analysis, vulnerability scan, and asset management.

In the *Analysis Engine*, the statements of most tickets refered to unusually high or low data traffic at certain times. Currently, there are still relatively many anomalies. This can be corrected with more training data and other settings. Furthermore, the *Static Analysis* still generates too many duplicate incidents (tickets). For example, a connection to an IP address was not allowed, which was specified via a rule. Instead of generating only one incident, 70 tickets were written during the tests. Additionally, the vulnerability analysis generated tickets that point to CVE vulnerabilities. All CVE vulnerabilities that were known were found. All of them were non-critical. *Asset Management* successfully captured the assets in the network. The ports and protocols used were shown per asset. Tickets referenced the assets found and could be used to track anomalies.

Overall, the tests showed that the SIEM architecture worked with the self-developed analysis engine well enough. However, too many tickets were still generated, which is an area for improvement. Also, duplicate tickets should be avoided in the future by means of adding a duplicate detection mechanism early on. In addition, it is also necessary to keep in mind the amount of data generated, which is necessary for the analysis. This is particularly important as data traffic will grow faster with usage of more advanced hardware components in the future.

VI. Conclusions

The architecture presented in the previous section will provide the required features for security-based anomaly detection in IT and OT environments as presented in section 1. For the data collection chain a vertical subset of the planned components has already been completed, yielding valuable insights for realizing the remaining ones. In particular, to improve anomaly detection results, more sensors have to be added to the system to provide further options for describing the normal system state and in consequence analyze potential deviations. This is true for both office as well as industrial settings.

In the data analysis component group the focus in future developments will be on improving the analysis engine at its core. An initial version of the analysis engine has already been implemented that uses multidimensional cube-based analysis of data to detect anomalies similar to the algorithm in [20]. This component has to be extended and adapted to different event types, particularly for industrial scenarios, and also the analysis algorithms need to be improved. A systematic evaluation of the algorithms on a rich set of event types in traditional IT environments is required for this. More realistic and comprehensive training data sets will also be essential for improving the analysis component.

In summary, we can conclude that the suggested system architecture is a good step forward towards achieving a security incident analysis system which can flexibly adjust to changing system behaviour due to its anomaly detection based approach. By integrating information from an arbitrarily wide range of input sensors and by using novel multidimensional anomaly detection algorithms the system is able to detect modifications that could not have been detected previously. In addition the systems ability to be scaled horizontally facilitates analysis of very large sets of input data. Finally, the SIEM GUI can display complex analysis results as well as context data to security operators in an easy-to-use manner, thereby helping them to address threats effectively.

Acknowledgment

The authors would like to thank the German Federal Ministry of Education and Research (BMBF) [18] 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] GLACIER project website: https://www.glacier-project.de.

[2] Michael Fiedler: *Cyberangriffe: dramatische Zunahme und Rekordschäden*. procontra-online.de, Alsterspree Verlag GmbH, Berlin, 15.11.2019, accessed: 2021-04-23.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

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

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] C. Ordonez, Z. Chen, and J. Garc´ıa-Garc´ıa: *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.

[17] 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.

[18] Federal Ministry of Education and Research (BMBF): https://www.bmbf.de.

[19] Hamolia, V., Melnyk, V., Zhezhnych, P., & Shilinh, A. (2020). *Intrusion Detection in Computer Networks using Latent Space Representation and Machine Learning*. International Journal of Computing, 19(3), 442-448. https://doi.org/10.47839/ijc.19.3.1893

[20] Heine F. (2017) *Outlier Detection in Data Streams Using OLAP Cubes.* In: Kirikova M. et al. (eds) New Trends in Databases and Information Systems. ADBIS 2017. Communications in Computer and Information Science, vol 767. Springer, Cham