How to Build a SOC on a Budget

Risto Vaarandi and Sten Mäses

How to Build a SOC on a Budget

Risto Vaarandi
Centre for Digital Forensics and Cyber Security
Tallinn University of Technology
Tallinn, Estonia
risto.vaarandi@taltech.ee

Sten Mäses
Centre for Digital Forensics and Cyber Security
Tallinn University of Technology
Tallinn, Estonia
sten.mases@taltech.ee

***Abstract*—During the last decade, many security-aware organizations have built a Security Operations Center (SOC) which refers to security tools and a team of security personnel using these tools according to predefined procedures. However, creating an organizational SOC can involve a significant investment into hardware and software, and setting up a SOC can be a complex and lengthy process. Although SOC related issues have received a considerable amount of attention in recent academic literature, there are very few recommendations on how to build a SOC in a cost-efficient and scalable way with open-source and free solutions. This paper fills this gap and describes the use-case of a SOC in an academic organization, with the main emphasis being on technical details and implementation recommendations.**

***Index Terms*—cyber security, security monitoring, incident detection and response, security operations center, SOC**

I. Introduction

Since the complexity of cyber threats and the number of cyber attacks are steadily increasing, many organizations are running a Security Operations Center (SOC) for timely detection of security incidents [1]. SOC is usually defined as a combination of people (security personnel), technology (security tools), and procedures that define the workflow [2]–[4]. One of the most important SOC technologies is the Security Information and Event Management system (SIEM) that collects, processes, and visualizes security events and other data from the entire organization.

Although several aspects of SOC have received considerable attention, there are almost no papers describing the actual implementation of a SOC with detailed recommendations on how to build it. Other important questions that have received little attention so far are the cost of creating a SOC and its scalability – according to recent studies, limited budget and insufficient scalability of SOC tools are key concerns that SOC managers face [5], [6].

This paper addresses those research gaps and describes a cost-efficient and scalable SOC implementation that relies on open-source and free solutions, and has been operated since 2019 at Tallinn University of Technology (the second largest university in Estonia with 12,000+ users of the organizational network and IT systems). The remainder of this paper is organized as follows – section II discusses related work, section III describes the technology, human resources, and procedures used in SOC, and section IV concludes the paper.

II. Related Work

In recent research literature, several aspects of SOC have received considerable attention. For example, several papers have suggested the use of various machine learning algorithms for event prioritization and anomaly detection [7]–[11]. Also, some papers have discussed the architecture of SOC for generic or specific environments [3], [6] and defined properties for SOC alarms that facilitate efficient alarm handling [12]. There are also papers on security concerns and other technical and non-technical issues that appear in SOCs [1], [2], [5].

Vielberth, Böhm, Fichtinger and Pernul have composed an extensive research literature survey for identifying the nature of a modern SOC and open challenges [1]. The challenges include monotonous tasks of SOC analysts, insufficient sharing of domain knowledge about IT environment and assets between experts and SOC analysts, limited knowledge sharing and collaboration between SOC staff, visualization capabilities that need improvement, insufficient level of automation offered by SOC tools, and the need for privacy related regulations. Since a SOC can handle a wide variety of sensitive private and system data, János and Dai have pointed out the need for properly securing the SOC environment and raising the awareness of SOC staff [2].

Kokulu et al. have conducted a number of semi-structured interviews with SOC managers and analysts for finding technical and non-technical issues in SOCs [5]. Detected issues include low visibility on devices and network topology, insufficient detection and handling of specific attack types, and insufficient training for analysts. Also, SOC managers highlighted limited budget and lack of scalability of SOC technologies as issues. Interestingly, high rate of false positives was not found to be a serious obstacle. This finding has been questioned by Alahmadi, Axon and Martinovic who have described five properties of SOC alarms (Reliable, Explainable, Analytical, Contextual, and Transferable) that facilitate efficient validation of alarms [12].

In [3], Radu has presented a generic SOC architecture where SIEM consists of generation layer (generation of security events on monitored devices), acquisition layer (transmission of events from devices to central SIEM event store), data manipulation layer (central event processing by SIEM), and presentation layer (visualization of events by SIEM). The paper also emphasizes that while dedicated monitoring devices

like IDS appliances are often controlled by SOC staff, network devices like switches, routers, firewalls, and IPS appliances that generate security events are usually managed by network admins outside SOC.

In [6], Weissman and Jayasumana have proposed a SOC architecture for monitoring IoT devices, and have highlighted key challenges – limited budget, insufficient scalability of SOC event databases for receiving large volumes of events, insufficient analyst expertise and alert fatigue.

Demertzis et al. have presented a network anomaly detection method for use in SOC which employs an ensemble of support vector machine, neural network, random forest, and k-nearest neighbors based classifiers [11]. The ensemble is used for processing the data points that represent network flows. According to the experiments conducted by the authors on five data sets, the ensemble had the same performance as its best classifier. In [8], Bienias, Kołaczek and Warzińsky have proposed an architecture for a network anomaly detection module which is designed for use in SOC.

Feng, Wu and Liu have described a machine learning framework for SOC that assesses the risks related to individual users [7]. First, a training data set is created for supervised machine learning algorithms, where each data point describes some user, with features of the data point derived from alerts for that user, user's social connections graph analysis, etc. Also, the data point labels are automatically derived by applying text mining algorithms to SOC analyst notes. The authors have evaluated four classifiers that are based on multi-layer neural network with two hidden layers, random forest with 100 estimators, support vector machine with radial basis function kernel, and logistic regression. According to the authors, multi-layer neural network and random forest based classifiers achieved the best performance during the experiments.

Najafi et al. have suggested the MalRank anomaly detection algorithm for domain names, IP addresses, and other entities extracted from SOC event logs [10]. After extraction, entities are organized into a graph as nodes, with graph edges representing relations between the entities. MalRank uses information about known malicious nodes (e.g., obtained from threat intelligence) for calculating maliciousness score for other nodes in the graph.

Gupta, Traore and Quinan have presented a supervised method for classifying SOC events that relies on past event data that have been labeled by SOC analysts [9]. First, categorical features of past event data are converted into numerical features with one-hot encoding, and additional features are derived from existing ones. After that, a classifier is trained on this data set, and the classifier is then used for detecting future high-priority SOC events. The authors experimented with XG Boost, logistic regression, and deep neural network based classifiers, with the latter having the best performance.

Although a number of papers have been published on the design, architecture, and other aspects of SOC, there are very few papers that provide detailed recommendations on how to build a production SOC. The paper by de Céspedes and Dimitoglou is one notable exception [13]. The authors have created a SOC for a university department with under 50 employees, with the department's network containing a firewall and a number of servers, network devices, and other network nodes. For SOC technology, only open-source tools were used for reducing the cost of building the SOC, with Graylog log management system, Grafana visualization framework, OSSIM SIEM, Snort IDS, and pfSense firewall being the main SOC tools. In order to reduce the cost of SOC even further, SOC tools were implemented on virtual machines. As mentioned in the paper, the SOC was built as a proof-of-concept solution for future research and teaching activities, and the paper does not provide any details about its production use over a longer period of time. In the following section, we will describe a low-cost production SOC for a large academic institution that has been operational since 2019.

III. Description of SOC

A. Background

The SOC described in this section has been operational since April 2019 at Tallinn University of Technology, Estonia (TalTech). TalTech is the second largest university in Estonia, having about 2,000 employees and over 10,000 students. The university's infrastructure is complex and largely managed by the IT support unit that exercises control over network infrastructure, firewalls, and most critical servers. However, some parts of the infrastructure (e.g., academic computing clusters and specific departmental sub-networks) are maintained by academic personnel. The management of the university's network is further complicated by the bring-your-own-device policy, with a significant amount of end-user devices in the university's private network being not maintained by the IT support unit. During everyday use, the university's infrastructure sees considerable amount of network traffic, with the network bandwidth consumption on the outer network perimeter routinely being 1.5–2 Gbit/s.

The SOC of TalTech was created for achieving the following three objectives:

- improving organizational security posture by advancing threat detection and analysis,
- offering learning opportunities to students by involving them as security analysts,
- facilitating academic cyber security research by using collected security data for research experiments.

In recent research literature, limited budget and insufficient scalability of SOC tools have been highlighted as major challenges when implementing and running a SOC [5], [6], [13]. For addressing these challenges, our SOC employs open-source and free tools that have been combined in a scalable way.

Our SIEM is based on a non-commercial installation of Elastic Stack [14] which was chosen because of its popularity as a log management and SIEM platform [15], [16]. In addition, other log management and SIEM platforms like Graylog and Wazuh are relying on Elastic Stack technologies, most notably the Elasticsearch database backend [17], [18]. Since

one of the SOC goals is to educate the student analysts, using widely accepted technologies in SOC increases the future value and usefulness of the knowledge the students obtain during the learning process. Also, the Elasticsearch database engine has a distributed nature and can be easily extended from a single node installation to a cluster of many nodes, allowing to scale it up for handling growing workloads.

B. Experiments for Evaluating SIEM Tools

This subsection presents performance experiments for finding the most scalable deployment scenario for Elastic Stack based SIEM. For collecting events from monitored nodes to SIEM, two deployment scenarios have been suggested in Elastic Stack documentation (see Fig. 1 and Fig. 2). In the case of both scenarios, events originate from Beat agents [19] that are lightweight data shippers for sending log file data and other relevant data to SIEM. One of the most commonly used Beat agents is Filebeat [20] which is able to track log files in real time, sending events extracted from log files to a configured destination.

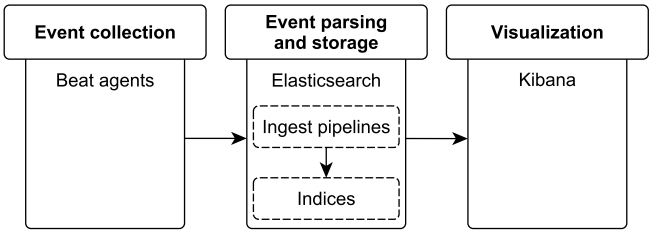

Fig. 1. Standard implementation of Elastic Stack

In the case of the first scenario (see Fig. 1) [21], Beat agents like Filebeat are sending security events from monitored nodes directly to the Elasticsearch database, where events are parsed by ingest pipelines and then stored in Elasticsearch indices. Although access to a properly secured Elasticsearch database requires authentication, allowing any remote node from the organizational network to establish a TCP connection to a database backend is not a good security practice [22]. For this reason, most security-aware organizations prevent unrestricted access to database backends by network segmentation, allowing network connections only from a few selected nodes. Furthermore, sending events from Beat agents directly to Elasticsearch does not allow for the creation of regular log files on a central log server. However, compressed log files consume much less disk space than Elasticsearch indices and are more convenient for long term storage of log data.

For addressing these issues, Elastic Stack documentation describes another deployment scenario that involves the use of Logstash [23] event processing tool as a gateway between Beat agents and Elasticsearch (see Fig. 2) [24]. Although Logstash is able to write received events to regular log files, Logstash is known to consume a lot of computing resources and a widely used *syslog* server Rsyslog [25] has been found to be the most efficient alternative [26], [27]. Since Logstash

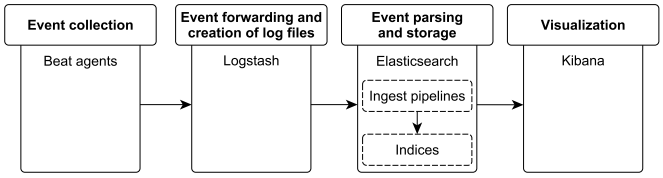

Fig. 2. Elastic Stack implementation with Logstash

and Beat agents communicate with a proprietary Lumberjack protocol, Beat agents cannot send events to Rsyslog directly, but can be configured to communicate with Rsyslog via Kafka or Redis messaging infrastructure. As an alternative, Logstash gateways can be installed in organizational subnetworks for local Beat agents, so that gateways are forwarding events to Rsyslog over the *syslog* protocol. We have released example configurations for Filebeat, Logstash, and Rsyslog at *https://github.com/ristov/elastic-examples*, and these configurations have also been used for the experiments outlined below.

First, experiments were conducted with Rsyslog and Logstash to measure their resource consumption[1]. For the workload, around 1,200 events per second during 3 hours were generated in EVE (JSON) format by Suricata IDS [29] in a production network. Performance tests were conducted on a server with 24 logical CPUs (Intel Xeon E5-2630Lv2), 64GB of memory, and Linux operating system. Since the server was manufactured in 2013, the test results reflect the performance of evaluated solutions on commodity hardware. Elastic Stack 7.16.3 and Rsyslog 8.2202.0 were installed on the server, and for making Logstash more efficient according to vendor recommendations [30], we configured it to use 4GB of memory and write events to Elasticsearch by larger batches of 500 events. During the experiments, the CPU utilization of a running program was measured as follows:

$$\frac{CPU\ time\ used\ by\ program\ in\ seconds}{program\ execution\ time\ in\ seconds} * 100\% \quad (1)$$

Note that the CPU utilization can exceed 100% if a multithreaded program has fully consumed the resources of more than one CPU. According to our tests, the CPU utilization of Rsyslog and Logstash was 23.9% and 48.4% respectively (see the first data row in Table I), with the peak memory usage of Rsyslog reaching 1.4GB.

TABLE I
CPU UTILIZATION OF RSYSLOG AND LOGSTASH

Additional parsing	Rsyslog	Logstash
no	23.9%	48.4%
yes	24.4%	58.1%

[1]Like Rsyslog, another widely used *syslog* server Syslog-ng can send events to Elasticsearch. However, it has limited support for ingest pipelines which prevents its use as a gateway between Beat agents and Elasticsearch [28], and therefore we have not evaluated Syslog-ng during the experiments.

Quite often, the event parsing and enrichment done in Elasticsearch ingest pipelines is not sufficient and events require some additional preprocessing. For evaluating the cost of additional event parsing for Logstash and Rsyslog, we configured both tools to parse out source and destination IP addresses from incoming Suricata IDS events, treating the events in EVE format as flat strings. The tools were also configured to create two additional fields based on the values of source and destination IP addresses. In the case of Logstash, event parsing was addressed with a *grok* filter, while in the case of Rsyslog a *liblognorm* rule was used (*grok* and *liblognorm* are standard event parsing measures of respective tools). According to Table I, the CPU utilization of Logstash increased by almost 10% (from 48.4% to 58.1%), while in the case of Rsyslog the increase was marginal (from 23.9% to 24.4%). These results are in line with findings by other researchers who have reported a significant performance gap between *grok* and *liblognorm* [26], [31]. Also, the performance of *liblognorm* is much less influenced by the rulebase size [26], [31]. For example, during an experiment described in [31], increasing the *liblognorm* rulebase from 37 rules to 1,161 rules increased the CPU time consumption only by 20%.

One of the main advantages of Elastic Stack predefined ingest pipelines is the parsing of events according to Elastic Common Schema (ECS) which introduces specific field names. Since Elastic Stack comes with many predefined event correlation rules and dashboards that have been designed for ECS, ingest pipelines help to prepare event data for these rules and dashboards. However, although ingest pipelines exist for a number of common event formats, there are many formats for which pipelines are currently not provided by Elastic Stack. Also, if the end user does not wish to use predefined event correlation rules and dashboards but rather create a custom setup, an ingest pipeline is not required, since event parsing can be entirely delegated to Logstash or Rsyslog.

To assess the CPU consumption for event parsing with and without an ingest pipeline, we measured the CPU utilization of Elasticsearch under the workload of 2,000 EVE messages per second from Suricata IDS during 3 hours, using Rsyslog to write events to Elasticsearch. When a predefined ingest pipeline for Suricata events was employed for parsing, the CPU utilization of Elasticsearch reached 444.6%, and when parsing was done by Rsyslog, Elasticsearch CPU utilization dropped to 59.2% (in both cases, Rsyslog CPU utilization was around 40%). Despite the pipeline offering some advantages (e.g., GeoIP enrichment), parsing with Rsyslog reduced the CPU consumption of Elasticsearch about 7.5 times. Also, although Elastic Stack was deployed on commodity hardware, the combined CPU utilization of Rsyslog and Elasticsearch was about 100% (i.e., just 1 CPU out of 24 was fully loaded).

The experiment results from this subsection illustrate that combining Rsyslog with Elastic Stack allows to considerably decrease the resource consumption and increase the system scalability. These considerations are particularly important when Elastic Stack based SIEM has to be deployed as a single node installation or on nodes with limited computing power.

C. SOC Tools and Architecture

The SOC architecture is depicted in Fig. 3. For the sake of cost-efficiency, the SIEM and other SOC tools have been deployed on virtual machines, unless the tool vendor recommends the use of physical machines for high-performance setups (e.g., see [32]). The SIEM employs Elasticsearch backend and Kibana visualization interface. For performance reasons, Elasticsearch data nodes are running on bare-metal servers with large SSD disks, with data being replicated over several nodes for fault tolerance.

According to experiment results from the previous subsection, Rsyslog is utilized as the main tool for parsing and storing events into Elasticsearch, with about 0.1% of events being directed to Logstash for more complex and expensive parsing. For collecting security events, *syslog* protocol is used which is supported by all network devices and servers in the organizational network, and also allows to collect events from Beat agents via Logstash gateways. Events are collected over encrypted and authenticated network connections (e.g., using TLS-based *syslog* protocol [33], apart from few network devices which don't support this functionality). Events are received with Syslog-ng [34] which is used for creating regular log files and for elaborate event preprocessing. Syslog-ng supports advanced event rewriting rules that employ Perl-Compatible Regular Expressions, and we use these rules for some complex event normalization and modification tasks (e.g., removing sensitive data from events that are used during academic research experiments in real-time).

One important SIEM tool is the event correlation engine [35]. Elastic Stack has its own security event correlation engine that comes with over 600 deactivated rule examples for correlating events from many different applications and other sources. The engine is featuring Event Query Language (EQL) for detecting event sequences by frequent database queries (e.g., executed once a minute). This *batch processing* based event correlation technique has one significant drawback – when the number of rules increases and they involve complex database queries that search events from larger time frames, it can impose a significant load on the database. Because of this limitation, vast majority of 600+ example rules involve small time frames for searching event sequences or no time frames at all (e.g., we found only 5 rule examples using a time frame larger than 5 minutes). Also, generating output notifications from Elastic Stack event correlation engine requires a commercial license which is another serious limitation.

For these reasons, our SOC employs Simple Event Correlator (SEC) [36] for more demanding event correlation tasks. Unlike Elastic Stack native event correlator, SEC employs the *stream processing* paradigm that involves immediate correlation of incoming events with the help of compact memory based data structures representing information about past events. A single SEC instance can handle hundreds of events per second while consuming a small amount of CPU time [36], and since SEC memory footprint is very modest (typically about 20–30MB), it is straightforward to run several

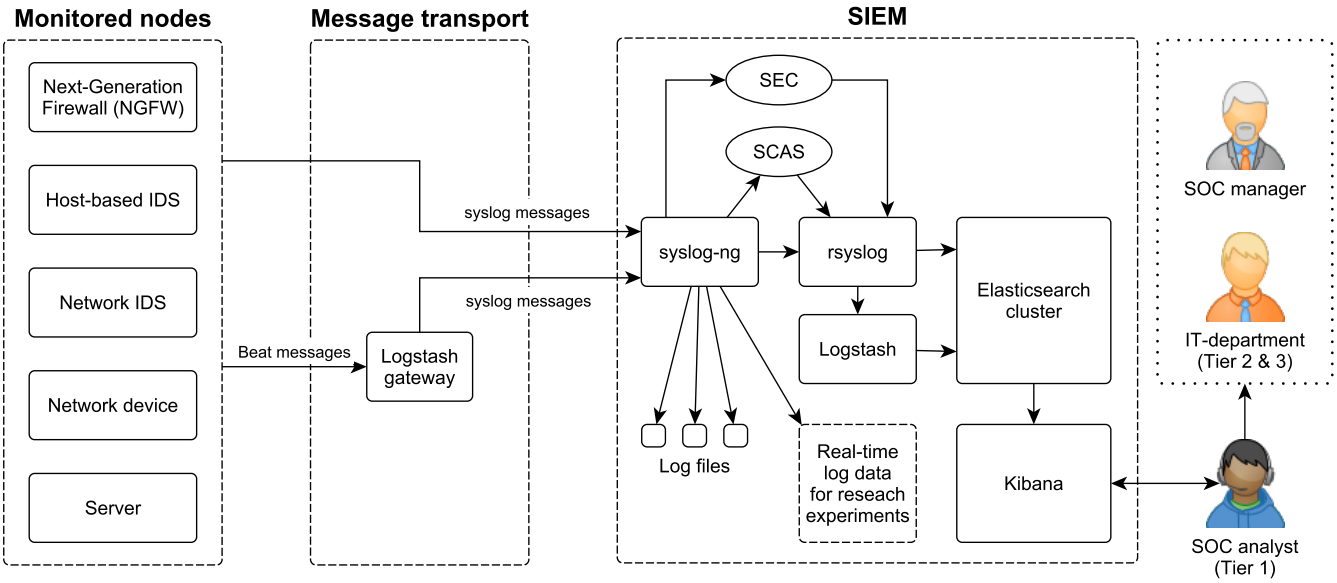

Fig. 3. SOC architecture

instances on the same machine if parallel processing is needed. Also, we have found the functionality of EQL quite limited when compared to the SEC rule language, and the use of SEC has allowed for implementing more complex event correlation schemes without imposing any load to Elasticsearch database.

As for nodes that send data to Elastic Stack based SIEM, we are following the shared responsibility model suggested in [3] – dedicated monitoring devices like network IDS are controlled by SOC staff, while other nodes are managed by personnel outside SOC. We employ Suricata IDS as our primary open-source network monitoring tool. Suricata was selected because of its scalable multi-threading based architecture and ability to cope with high network loads that exceed 10 Gbit/s on regular server hardware [37]. Suricata appliances have been configured to use Emerging Threats signatures and in addition to IDS alerts, they also produce events about observed network flows and application layer transactions (e.g., DNS queries and responses).

The SIEM is also collecting security events from regular computing infrastructure such as next generation firewalls that implement IPS functionality, network devices, host-based security management and IDS modules, etc.

D. People and Processes

A SOC organization usually comprises the following roles – SOC analysts (e.g., arranged into two or three tiers), security experts (e.g., malware analysts), and the SOC manager [1], [35]. Our SOC employs 2–3 student volunteers as Tier 1 analysts. All the other SOC roles are fulfilled by 3–4 regular employees (according to a recent study [4], majority of organizations have 2–5 analyst positions). The work of Tier 1 student analysts is not remunerated, but they receive academic credit points for one semester of work. This arrangement not only reduces the SOC personnel costs, but also allows to use the SOC for teaching purposes.

In recent literature, insufficient training, monotonous work and alert fatigue that can lead to burnout, insufficient sharing of domain knowledge about IT environment and assets between experts and analysts, and limited knowledge sharing and collaboration between SOC staff have been identified as analyst related challenges [1], [5], [6]. For addressing these issues, the students are recruited from cyber security master's program of TalTech which offers a number of highly technical courses and training relevant to analyst work. Also, student analysts are rotated after every semester for preventing analyst burnout.

For knowledge transfer, students from the previous semester train the new analysts during a 2–3 weeks long handover period. In addition, the analysts and experts are maintaining a Wiki based knowledge repository which includes documentation and usage tips for SOC tools, past daily and weekly reports with details about alert analysis, and information about IT assets. Finally, for facilitating cooperation and timely communication between SOC staff, an MS Teams channel is maintained.

Other challenges related to SOC and its staff are proper handling of sensitive data, the need for relevant privacy regulations, security awareness of the SOC staff, and the need for properly securing the SOC environment [1], [2]. For addressing these challenges, new analysts go through security briefings during the recruitment process. During the briefings, the rules for handling sensitive data and using the SOC environment are explained. For example, the rules do not allow copying any SOC data to external media or outside the SOC environment, and the use of personal devices in SOC environment is prohibited. After the briefings, new analysts

sign a Non-Disclosure Agreement (NDA) that describes these rules (similar NDAs are signed by researchers who wish to use the SOC data for academic research projects). Finally, the SOC environment resides in dedicated networks that are separated from rest of the campus network, and physical access to the SOC analyst room is limited.

One of the most important SOC work processes is the alert handling process [1]. During this process, alerts are prioritized by Tier 1 analysts, with high-priority alerts being investigated first. Depending on the alert impact, Tier 1 analysts will contact Tier 2+ analysts and security experts for further analysis and response. As an important part of the alert handling process in our SOC, the results of the analysis are documented in the Wiki based knowledge repository. Daily and weekly reports are another major deliverable from Tier 1 analysts that are also stored in the knowledge repository.

Insufficient level of automation in the alert handling process has been identified as a major issue that causes alert fatigue and demotivates the analysts [1], [6]. As discussed in section II, machine learning approaches have been suggested for improving automation. Unfortunately, implementations of most approaches have either not been tested over longer periods of time in production environments or are not publicly available. For example, from machine learning approaches discussed in section II, none have publicly available implementations, and only one is actually used in an organizational SOC [7].

In our environment, handling IDS alerts from the external network perimeter is the most challenging issue, since about 200,000 alerts are triggered each day (typically, there are less than 100 alerts of other types per day, and they can be easily handled by analysts). For addressing this problem, we have created an open-source tool called SCAS which implements an unsupervised stream clustering algorithm for automated prioritization of IDS alerts [38].

SCAS processes alerts in real time and first arranges them into alert groups, where each group represents alerts for the same external host observed during a short time frame (e.g., 5 minutes). Alert groups are then clustered, so that groups belonging to clusters match frequently occurring IDS alert patterns representing threats of low importance. Also, unusual alert groups are classified as outliers. Finally, a similarity score from range 0..1 is calculated for each alert group assigned to a cluster which reflects the similarity between the group and its cluster centroid (outliers are receiving the score of -1). Alert groups with lower similarity scores are thus outliers or dissimilar to their cluster centroids, deserving closer attention.

SCAS does not require any human interaction and is able to compress and filter out large IDS alert volumes of low importance. For example, in January – February 2022 SCAS reduced 11,484,738 IDS alerts to 71,033 IDS alert groups of higher importance (i.e., the number of events that required inspection was reduced by more than 99%).

For the automation of other IDS alert analysis tasks (e.g., the identification of aggressive attackers who trigger many alerts during short time frames), SEC event correlation rules are employed.

IV. Conclusion and Future Work

This paper describes a cost-efficient production SOC that operates in a large academic institution, and also provides detailed technical recommendations on how to use open-source and free tools for building a low-cost and scalable SIEM solution for a SOC.

As for future work, we plan to continue research on unsupervised machine learning and active learning algorithms for automated classification and prioritization of SOC alerts. We also plan to study the human aspects in SOC, e.g., the efficiency of knowledge transfer from experienced analysts to newcomers and methods for improving it.

Acknowledgment

The authors express their gratitude to Mr. Priit Pennula, Prof. Rain Ottis and Prof. Olaf M. Maennel for their support.

References

[1] M. Vielberth, F. Böhm, I. Fichtinger, and G. Pernul, "Security operations center: A systematic study and open challenges," *IEEE Access*, vol. 8, pp. 227 756–227 779, 2020.

[2] F. D. János and N. H. P. Dai, "Security concerns towards security operations centers," in *2018 IEEE 12th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, 2018, pp. 000 273–000 278.

[3] S. G. Radu, "Comparative analysis of security operations centre architectures; proposals and architectural considerations for frameworks and operating models," in *International Conference for Information Technology and Communications*. Springer, 2016, pp. 248–260.

[4] C. Crowley and J. Pescatore, "Common and best practices for security operations centers: Results of the 2019 soc survey," *SANS, Bethesda, MD, USA, Tech. Rep*, 2019.

[5] F. B. Kokulu, A. Soneji, T. Bao, Y. Shoshitaishvili, Z. Zhao, A. Doupé, and G.-J. Ahn, "Matched and mismatched socs: A qualitative study on security operations center issues," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 1955–1970.

[6] D. Weissman and A. Jayasumana, "Integrating iot monitoring for security operation center," in *2020 Global Internet of Things Summit (GIoTS)*. IEEE, 2020, pp. 1–6.

[7] C. Feng, S. Wu, and N. Liu, "A user-centric machine learning framework for cyber security operations center," in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2017, pp. 173–175.

[8] P. Bienias, G. Kołaczek, and A. Warzyński, "Architecture of anomaly detection module for the security operations center," in *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. IEEE, 2019, pp. 126–131.

[9] N. Gupta, I. Traore, and P. M. F. de Quinan, "Automated event prioritization for security operation center using deep learning," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 5864–5872.

[10] P. Najafi, A. Mühle, W. Pünter, F. Cheng, and C. Meinel, "Malrank: a measure of maliciousness in siem-based knowledge graphs," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, pp. 417–429.

[11] K. Demertzis, P. Kikiras, N. Tziritas, S. L. Sanchez, and L. Iliadis, "The next generation cognitive security operations center: network flow forensics using cybersecurity intelligence," *Big Data and Cognitive Computing*, vol. 2, no. 4, p. 35, 2018.

[12] B. A. Alahmadi, L. Axon, and I. Martinovic, "99% false positives: A qualitative study of soc analysts' perspectives on security alarms," in *Proceedings of the 31st USENIX Security Symposium (USENIX Security), Boston, MA, USA*, 2022, pp. 1–18.

[13] R. de Céspedes III and G. Dimitoglou, "Development of a virtualized security operations center," *Journal of Computing Sciences in Colleges*, vol. 37, no. 3, pp. 108–119, 2021.

[14] "Elastic stack," Elastic Stack home page. [Online]. Available: https://www.elastic.co/elastic-stack/
[15] K. M. Kavanagh, T. Bussa, and J. Collins, "Gartner magic quadrant for security information and event management," *Gartner Inc*, 2021.
[16] D. Horovits, "The complete guide to the ELK stack," Jun. 9, 2020. [Online]. Available: https://logz.io/learn/complete-guide-elk-stack/
[17] "Graylog - industry leading log management." [Online]. Available: https://www.graylog.org
[18] "Wazuh - the open source security platform." [Online]. Available: https://wazuh.com
[19] "Beats," Beats home page. [Online]. Available: https://www.elastic.co/beats/
[20] "Filebeat," Filebeat home page. [Online]. Available: https://www.elastic.co/beats/filebeat
[21] "Configure the elasticsearch output," Filebeat documentation. [Online]. Available: https://www.elastic.co/guide/en/beats/filebeat/current/elasticsearch-output.html
[22] "Database security cheat sheet," OWASP Cheat Sheet Series. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Database_Security_Cheat_Sheet.html
[23] "Logstash," Logstash home page. [Online]. Available: https://www.elastic.co/logstash/
[24] "Configure the logstash output," Filebeat documentation. [Online]. Available: https://www.elastic.co/guide/en/beats/filebeat/current/logstash-output.html
[25] "Rsyslog," Rsyslog home page. [Online]. Available: https://www.rsyslog.com
[26] R. Gheorghe, "5 logstash alternatives (2022 comparison)," Sematext Blog, Jan. 4, 2022. [Online]. Available: https://sematext.com/blog/logstash-alternatives/
[27] R. Vaarandi and P. Niziński, "Comparative analysis of open-source log management solutions for security monitoring and network forensics," in *Proceedings of the 2013 European conference on information warfare and security*, 2013, pp. 278–287.
[28] "A question about the elasticsearch-http driver," Syslog-ng mailing list. [Online]. Available: https://lists.balabit.hu/pipermail/syslog-ng/2022-March/026409.html
[29] "Suricata," Suricata home page. [Online]. Available: https://suricata.io
[30] "Logstash performance tuning," Logstash documentation. [Online]. Available: https://www.elastic.co/guide/en/logstash/7.16/performance-tuning.html
[31] R. Gerhards, "Efficient normalization of it log messages under realtime conditions," 2016.
[32] "High performance configuration," Suricata documentation. [Online]. Available: https://suricata.readthedocs.io/en/suricata-6.0.5/performance/high-performance-config.html
[33] F. Miao, Y. Ma, and J. Salowey, "Rfc5425: Transport layer security (tls) transport mapping for syslog," Mar. 2009. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc5425
[34] "Syslog-ng," Syslog-ng home page. [Online]. Available: https://www.syslog-ng.com
[35] C. Zimmerman, "Ten strategies of a world-class cybersecurity operations centre. the mitre corporation," 2014.
[36] R. Vaarandi, B. Blumbergs, and E. Çalışkan, "Simple event correlator - best practices for creating scalable configurations," in *2015 IEEE International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*. IEEE, 2015, pp. 96–100.
[37] T. Appel, "Pushing suricata towards 80Gbps and more," Oct. 30, 2019. [Online]. Available: https://suricon.net/wp-content/uploads/2019/11/SURICON2019_Pushing-Suricata-Towards-90-Gbit_s-and-More.pdf
[38] R. Vaarandi, "A stream clustering algorithm for classifying network ids alerts," in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*. IEEE, 2021, pp. 14–19.