Discerning Wheat from Chaff in SOCs

A Model to Identify ‘Non-Interesting’ Events in Security Operation Centers

Mulders, Tom R.J.

Award date:
2022

Discerning Wheat from Chaff in SOCs: A Model to Identify 'Non-Interesting' Events in Security Operation Centers

Tom Mulders
t.r.j.mulders@student.tue.nl
Eindhoven University of Technology
Eindhoven, The Netherlands

ABSTRACT

The swift identification of non-interesting security events is vital to the quality of service provided by security operation centers. Despite in-depth research into solutions automating the process of tier 1 security analysts, there is little insight into the actual process these analysts perform. Moreover, proposed automations often yield no insight into the reason for the classification of a security event and do not provide evidence. In this work, we design a model consisting of analysis steps, encompassing the analysis process of tier 1 security analysts in a SOC. The model aims to identify non-interesting security event in a swift and methodological, evidence-based manner. We verify the suggested model by running an experiment representative of a real SOC environment, using tier 1 analysts to perform analysis of a range security events. The experiment validates the performance of individual model steps in terms of consistency and accuracy, and validates the accuracy and completeness of the model as a whole. Our experimental results show varying consistency and accuracy for the individual model steps. Moreover, the results confirm the completeness of the model as a whole and we observe high accuracy in identifying non-interesting events. The designed model proves to be an effective tool for a multitude of applications in the SOC as well as in research.

CCS CONCEPTS

• **Security and privacy → Intrusion detection systems**; *Usability in security and privacy.*

KEYWORDS

Security Operations Centers, Network Intrusion Detection Systems, Security analysis

ACM Reference Format:
Tom Mulders. 2022. Discerning Wheat from Chaff in SOCs: A Model to Identify 'Non-Interesting' Events in Security Operation Centers. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The security of organizations is an ever increasingly important topic. At a time where a large number of devices and services are internet-connected, and society depends on these devices and services more than ever, cyber-attacks are becoming more common and sophisticated. At this pivotal point in time, where cyber-criminals are increasingly more capable of monetizing their attacks and a global pandemic provides the context for large phishing campaigns, organizations are spending vast amounts of resources on cyber-defenses. Organizations implement a number of security measures, from technical solutions to prevent or detect attacks, to awareness trainings.

An often implemented security with medium to large organization is a security operations center (SOC). A SOC aims to detect cyber-threats within the customer landscape, by means of various technical solutions. At the heart of the SOC, however, are human analysts with a wide range of expertise and diverse roles, analyzing security events, making decisions about escalation and mitigation, and supporting the owners of the monitored environments. These analysts are nowadays in short supply, as the demand for security analysts increases. Due to this shortage and the nature of the job, security analysts experience high workloads, tight KPIs and repetitive tasks in their daily operations[18] [4]. Specifically, tier 1 security analysts, who are the firstline responders to security events within the SOC, often analyze hundreds or even thousands of security events on a daily basis, and are expected to accurately discern 'interesting security events' (worth of further investigation and potentially of being escalated to the higher tiers in the SOC) from uninteresting security events (that may safely be disregarded). Due to the high demands and repetitive nature of the job, SOCs observe high stress levels for these analysts and suffer from high turnover rate[15][17] that imply additional costs and time to re-hire and re-train new employees on a regular basis. To maintain the effectiveness and efficiency of the SOC, automations should be implemented to support the tier 1 analysts' work. Achieving this, however, requires an in-depth understanding and description of the work of the tier 1 analysts.

Unfortunately, the literature shows a lack of knowledge when it comes to what tier 1 security analysts do specifically, and what drives them to make specific decisions. There is no clear, reproducible process defined, describing the methodology of these analyst, which in turn makes it hard for researchers to reason about the analysis of security events, or the performance of SOCs. This is especially hard, since the job of security analyst is notoriously vague and ill-defined[14]. On the other hand, this is critical to accomplish as it would allow the field to develop further automation without losing 'explainability' of the decisions (e.g. for a SOC to report to

the infrastructure owner the reasons why certain alerts were ignored altogether), and to develop methods to train and support the entry-level workforce in a SOC.

In this thesis we investigate the general analysis process of tier 1 security analysts in a SOC and aims to identify a systematic approach to discern interesting security events from uninteresting events. To this extent, we identify a sequence of steps encompassing the analysis process and derive an evidence-based model applicable for all general security events generated in the SOC. To achieve this we iteratively identify a sequence of analysis steps meant to guide the analyst in looking for evidence that the alert may be 'interesting', based on real security events in a SOC. Secondly we derive a model which is based on the analysis steps and allows for the systematic and quick identification of uninteresting security event. It is key to identify uninteresting security event quickly, because the number of incoming security events is generally too large to investigate everything[12]. The model consists of stages, which aim to identify the necessary evidence to label an alerts as interesting, a lack of this evidence indicates the alert may be uninteresting. We verify our findings in an experiment, where five tier 1 security analysts investigated 200 unique security events consisting of real traffic and injected attacks, using exclusively the proposed model. From the experimental results we obtained that the identified sequence of steps performs varying depending on the model step. However, for the model as a whole we obtained high accuracy across the board indicating the model performs well.

Our contributions are as follows:

- we develop a model that identifies uninteresting security events methodologically using evidence
- we test the model against real SOC data over a period of 10 days showing that it can be used by tier 1 analysts to analyze a range of different network security events
- we evaluate accuracy and consistency of model predictions and discuss implications for research and practice

The remainder of the paper is structured as follows. Section 2 provides background about SOCs and the security analysis process. Section 3 details the related work and section 4 identifies the problem statement and gap. Section 5 explains the methodology utilized for this research and section 6 describes the sequence of analysis steps as well as the model. Then section 7 details the experiment set up and its execution, and section 8 shows the results obtained from this experiment. In section 9 we discuss the results, and identify implications for research and practise and finally in section 10 we provide our conclusions.

2 BACKGROUND

2.1 Security Operation Centers

A security operation center (SOC) is a provider of security services to organizations. These services range from network intrusion detection systems (NIDS), to endpoint detection, firewall monitoring and security information and event management systems (SIEM). The tools providing these services may include static detection mechanisms, dynamic systems, machine learning, artificial intelligence and more. However, all of these technical security solutions generate security events which are in the end evaluated, in first instance, by a so-called Tier 1 analyst in the SOC. The analysis performed by tier 1 analysts aims to discern interesting security events from uninteresting events. Or in other words, identify true attack with impact, from false-positives or attacks with no impact. The interesting security events are escalated to tier 2 and tier 3 security analysts accordingly, for further investigation, classification, communication to customers and mitigation.

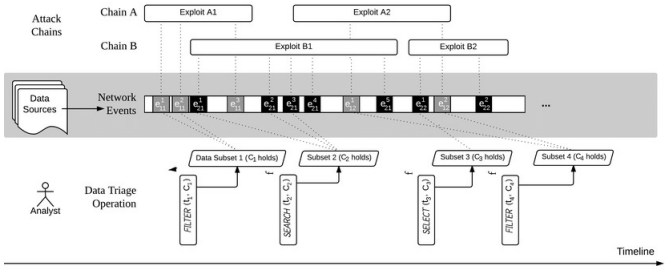

Figure 1: Overview of the analysis process [19]

2.2 The security analysis process

The analysis process of a tier 1 analysts has as input a security event, and as output a classification and/or escalation of this security event. Generally, analysts correlate the security event with other (security) events from any available and relevant source and perform a triage[21] [1] [11]. This triage aims to investigate network events and identify attack chains from the data, as can be seen in figure 1. The figure illustrates that analysts perform operations on network events to identify attack chains and the exploits which make up the chain. It furthermore demonstrates how the analysts must be able to separate attack chains from each other and from benign networks events, using search and filter queries. On a high level, this is the main task performed by security analysts in a SOC.

The investigations into security events as performed by tier 1 analysts are generally repetitive, time-consuming and error prone [18] [4]. Naturally, the quality of the analysis and thus the accuracy of the classification is dependent on the individual skill of the analyst. However, external factors can impact this significantly; for example, the arrival of external information such as the publication of a new vulnerability, or addition of a new customer can drastically impact the workload. Given that this occurs regularly, however not necessarily predictably, the general quality of security event analyses can be quite variable. This combined with a general shortage in security experts, and specifically SOC analysts, makes it difficult to deliver consistent and accurate security services to customers, while also growing and improving the SOC.

Since tier 1 analysts are the first to analyze and classify security event, the accuracy of the analysis is key. However, timeliness is equally important. Responding appropriately to a cyber attack, but in an untimely fashion leaves time for the attack to complete [3]. This can make any mitigation harder or even useless, as the attacker may have already achieved their objective at least partially (e.g. encrypting files to ask a ransom, exfiltrate sensitive data, ..).

3 RELATED WORK

The timeliness and quality of alert analyses, and the general performance of a SOC is dependent on the tooling available to the tier 1 analysts, and on the workload of these analysts. These two aspects are closely related; the introduction of tooling in the SOC may significantly reduce the workload of analysts. Given that SOCs detect vast amounts of alerts and security events, which makes it impossible to investigate everything[12], automations are a necessity in a modern SOC. This is demonstrated by Sundaramurthy et al. in [13], where the authors develop a tool which automates the monotonous workflow of investigating command and control (C&C) traffic from internal hosts. Developed specifically for this workflow, the tool can reduce the workload of tier 1 analysts by orders of magnitude, from 10 minutes on average to a mere 10 seconds. This is significantly impactful, for the analysis of C&C traffic, and allows more time for analysing other alerts. However, the tool is only applicable to the specific workflow, it is not a fully featured solution to any security event. Developing similar tools for all workflows in a SOC, and maintaining those, is not feasible.

Another workflow automation proposed by Van Ede et al., focusses on security event correlation. The authors propose DeepCase, a semi-supervised approach for the correlation of related security events[16]. The clustered alert can then be classified by analyst-driven policies. This approach focusses solely on the clustering of security events and does not encompass the entire analysis process, similar to the earlier proposed tool by Sundaramurthy et al. Also, this semi-supervised approach, fails to highlight the reason for clustering or in other words does not provide evidence that the correlated events are related.

Another type of automation proposed by Du et al. is DeepLog[2], which translates (security) logs into human readable text. This is achieved using a deep neural network, which then also detects anomalies in the logs. These anomalies can then be investigated by security analysts, using a workflow outputted by Deeplog. This automation can best be categorized as an anomaly detection system, not necessarily the automation of the task of an analyst as it takes as input any system or event log, not necessarily security events. In order to incorporate DeepLog into a SOC, the analysts must be trained to handle alarms generated by DeepLog and be able to interpret the workflows outputted by it. This highlights, that any automation within the SOC must either fit in existing workflows, or require additional training and effort of the analysts. In the latter case, one could argue that the workload of the analyst is increased, which is sub-optimal.

Zhong et al. propose an all-encompassing solution, based on the triage operations performed by security analysts. They observed high-performance and satisfactory false-positive rates. They do note, however, that the quality of the system depends on the quality of the triage traces, which in turn depends on the quality of the analyst. Notably, this approach utilizes the operations of the analysts, such as "searching", "selecting" and "filtering"[19], and does not capture why an analyst performs this action, nor what evidence is obtained from this operation. The usage of traces also suggests, that the system requires a number of triages performed by analysts, before being able to automatically perform triage. Finally, the output of the systems requires verification by human analysts and these analysts need to learn to interpret the output.

In earlier work by Zhong et al.[20], they also captured analysis operations performed by analysts in a SOC and the hypotheses they generate and utilize in this process. Here, they utilize cognitive task analysis (CTA), to capture fine-grained processes performed as part of the analysis. They captured a number analyses of security alerts in terms of actions performed and intermediate hypothesis, this gives great inside into the mind of an analyst. Interestingly, in the discussion the authors highlight that they observed different strategies being utilize to explore the data, and generate the hypotheses. This indicates the lack of a clear predefined process for analysts. In turn, this implies that given the same input security event, one cannot make any assumptions about the trace of the corresponding analysis and that these are fully dependent on the (quality of) security analyst.

Next to designing automation within the SOC, research has also been conducted on how to measure the performance of SOCs or of automations utilized within the SOC. Rosso et al. propose SAIBERSOC[9][10], a tool and methodology which enables researchers and SOC operators to test the performance of the SOC, analysts and tooling. Based on the MITRE ATT&CK framework, it enables the simulation and injection of attacks into SOC enviroments. Utilizing SAIBERSOC, one can design attacks and reason about projected outcomes, and then verify the hypothesis by execution a simulation of the attack. Most importantly, this simulation is as close to being an actual attack as one can achieve without impacting SOC or monitored enviroment systems, and is indistinguishable from other real traffic (and attacks) in the monitored environment.

The importance of tools such as SAIBERSOC for measuring SOC performance, is highlighted in research by Kokulu et al. In their qualitative study on issues within the SOC[5], one of the primary findings is the current metrics for SOC performance are not effective. Moreover, this is a point of contention between security analysts and their managers. They also found the speed of response and the level of automation, to be similarly important and big issues within SOCs. Additionally they noted that poor analyst training and high false-positive rates are issues within the SOCs in their research. This all culminates into poor quality of analysis, if left unaddressed.

Another interesting observation, done by Sunderamurthy et al in [14], is the problem of tacit knowledge within the SOC; decisions made by security analysts are based on intuition and not documented. Often, the security analysts cannot clearly communicate their knowledge related to the incident and the reason for their classification of this incident. This is a key component of the services provided by SOCs, as the contact point for the monitored environment must be provided with evidence of a security incident. Reporting that malware has been installed on a system for example, is often not satisfactory as mitigation may significantly impact the business processes provided by the monitored environment. The contact person must be convinced that mitigation is necessary and warrant a potential interruption of business processes. Similarly, tier 1 security analysts must do this as well when escalating to tier 2 or 3 analysts. Good communication about what a tier 1 analyst has observed, supporting evidence and their decision process is required, as otherwise higher tier analysts must perform this (partial) analysis again. More strongly, "SOC jobs such as incident response

and forensic analysis have become so sophisticated and expertise driven that understanding the process is nearly impossible without doing the job."[14]

To our knowledge there is no automated solution which can be applied to all types of security events and at the same time produces a reproducible, evidence-based analysis of these events. At the same time, the analysis process for tier 1 analysts is ill-defined or not defined at all. This leads to difficulty in communication, as well as to unverifiable alert classifications as well as gives no insight into what tier 1 security analysts do and for what reasons. These are all essential parts required for proper communication to owners of monitored environments. It also makes it hard to measure the performance of SOCs, analysts and tooling.

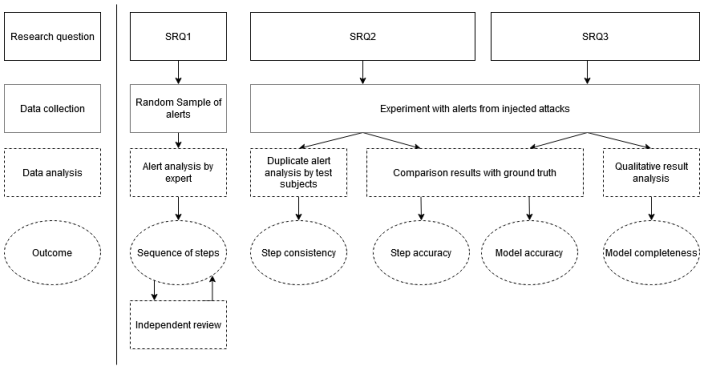

Figure 2: Overview of methodology

4 PROBLEM STATEMENT AND RESEARCH QUESTIONS

4.1 Problem statement

The process of alert investigation is repetitive, time-consuming and error prone[18] [4], which is why automation of these tasks can yield great benefits for the SOC. Much research has been done on individual steps or parts of the investigation, such as correlation and alert reduction, often relying on learning techniques. While these techniques often yield great results in terms of true positives, the number of false negatives can also be quite high, and these techniques often offer little transparency in their decision making as they are grey or black-box solutions, which requires analyst training to interpret. Understanding alerts and attacks is key in the service provided by SOCs which is to inform the customers about concrete risk to their business. The information provided to customers about risk must be factually correct and based on thoughtful analysis of concrete evidence, which is why human analysts are still key to the process.

To move towards the automation of the analysis process, it is first needed to build a model that describes it. Different steps of the model can then be potentially automated, meaning that specific outcomes can then be explained based on the model description of that step. Additionally, such model would provide the backbone for so-called *decision support systems* that guide the human analyst and can at least partially automate away part of the analysis, particularly the more repetitive tasks, while keeping the analyst in the decision loop. The model stages must consist of closely related data points and its content should allow analysts to answer a generic binary question about this particular stage. For example: Is the alert under investigation relevant for the owner of the monitored environment? The final stage of the model should allow for the generic classification "interesting" as well, as opposed to earlier stages. Each stages builds the necessary evidence towards the alert being potentially interesting, a lack of this evidence indicates alerts may be uninteresting.

Further, modeling the analysis process can provide useful insights and data for analysts and researchers. By modeling the analysis process, analysts can utilize a clearly defined process with labeling and distinct steps to perform analysis. This in turn, enables researchers to perform experiments using analysts which produce predictable and clearly defined outcomes, as well as serving as a training tool or guidelines for new analysts. Additionally, the presence of an analysis model may aid performance evaluation of T1 analysts and the SOC as a whole, for example through experiments evaluating analysis accuracy across different phases of the process (e.g. to identify bottle-necks in the analysis process).

As tier 1 analysts are the gateway to escalated events that are then analysed more in depth, in this thesis we focus on modelling the T1 analysis process.

4.2 Research questions

The problem statement above gives raise to the main research question of this thesis:

RQ: How can tier 1 analysts' process be modelled?

To answer the main research question, the following sub-questions have been identified:

SRQ1: Which sequence of steps determine a generic model to analyze alerts for tier 1 analysts?

SRQ2: Can the resulting steps be executed by an analyst in an accurate and consistent manner?

SRQ3: Is the resulting model as a whole accurate and complete in capturing relevant analysis steps?

5 METHODOLOGY

Overview of method. To investigate how a tier 1 analysts' can be generically modeled we first identified a sequence of analysis steps taken by analysts when investigating alerts. This sequence is the basis for the model. Next, we generalized the model and designed an experiment to verify the accuracy and consistency of the steps of the model and to verify the completeness and accuracy of the model as a whole. Figure 2 details the developed method.

5.1 Analysis step identification and categorization

In order to establish a sequence of steps which encompasses the tier 1 analyst's analysis process, we adopt a bottom-up approach and sample a set of 10 security alerts from a the prototype sensor of the ESH-SOC. The 10 alerts consist of malware, exploits, command & control, policy violations and scans. Moreover, some of these were true-positive events with impact, which we refer to as "interesting"

in this paper, and other were true-positive events with no impact or false-positives, these we refer to as "uninteresting" in this paper. We also included 2 occurrences of the same alert, where the classification was different, to ensure the model can differentiate between different network events which end up triggering the same alert. The sampling of these alert occurred between office hours during a weekday, therefore it is representative of the type of alerts that commonly trigger during the operation hours of the ESH-SOC. These alerts are analyzed completely by the authors of this paper, who is a tier 2 analyst with approximately 3 years of experience, with the aim of identifying all information which might be relevant for the analysis of the alert.

Having analyzed and classified the alerts, we established the potential steps of the analysis by reviewing the collection of analysis and identifying related sources of information which are key in the classification of those alerts. The data points identified in the analysis are then labeled and clustered together in related categories. These categories correspond to model steps and are sequenced into model stages, to create a first generic model for alert analysis.

Given the model, we approached two experts within the field of cyber security, one tier 3 security analyst and one experienced security researcher, and asked them to analyze 3 random security alerts, using exclusively the model. These 3 random alerts are sampled from the same environment, but distinct from the alerts used in to construct the model. Based on these analyses and the feedback regarding the completeness (i.e. the extent to which it capture the tier 1 analysts' analysis process), accuracy and usability of the model by these two experts, the model was iteratively refined and reviewed. Prior to the experimental phase of this research, we implemented all feedback provided by the experts, and obtained approval of both experts that the model performs well with regards to the aspects listed above.

5.2 Experiment Design

In order to measure the consistency and accuracy of the model steps, as well as the completeness and accuracy of the model, an experiment was designed. This experiment consists of the analysis of 200 random security alerts generated by a real SOC of which 22 are injected attack, over the course of 10 days. Attacks were injected, since the experiment environment yielded too few "interesting" alerts on a regular basis. The 200 random security alerts are analyzed by two experiment subject, so that we can compare the two analysis of a single alert one-to-one, and determine if the model and its step lead to consistent results. The experiment subjects are 5 tier 1 analysts working as interns at the ESH-SOC at the Eindhoven University of Technology. Given the similar knowledge level and identical data available to the experimental subjects, the participants should have produced the same results.

The experiment subjects perform the analysis of the random alerts according to the model, and collect the findings of each model step in a standardized results form. The results form contains predefined fields with questions that are designed to capture the outcome of each model step. Collecting the results in this way, allows us to compare the outcome of the model and its steps directly to other analysts and generate general statistics about the model. Additionally, whenever relevant free-text explanation fields are provided to report evidence to the decision as well as to note down any encountered issues with the analysis following the model.

A full description of the experimental setup, subjects, and of the experiment execution is provided in Section 7.

5.3 Verification of the accuracy and consistency of the identified steps

Using the designed experiment, we first verify if the identified model steps are accurate and consistent. In other words, does a model step generate the same outcome, given the same input alert and security data, and is this outcome correct.

5.3.1 Consistency. To measure the consistency of each model step, we compare the experimental results of each step from both experiment participants of the same day. Concretely, we compute the agreement rate between subjects. When the agreement rate is lower than 0.7, we consider the step to be inconsistent. This is based on the "strength of agreement" as defined by Landis at el. in [6], where 0.7 right in the middle of what the authors consider "substantial" agreement.

5.3.2 Accuracy. To measure the accuracy of each model step, we review the results produced during the experiment for every step and compare it to the ground truth. The ground truth of 50 alerts, consists of a random sub-set of the alert analyzed by the test subjects and is produced by a tier 2 analyst. It is recorded in the exact same manner as the results of the experiment as performed by the subjects, and therefore allows for easy comparison. The sub-set of alerts part of the ground truth includes, but is not limited to, all injected attacks. Accuracy is evaluated based on the degree to which the final classification of an alert matches with the associated classification in the ground truth. Concretely, we first verify the general quality of the results by utilizing the Chi-Squared method to see if any test subject performs significantly differently from the others. Further, we evaluate whether accuracy is affected by the specific rule categories. Then, to finally capture the model step accuracy, we compute the overlap with the ground truth. For the accuracy, we count the duplicate alert analysis as 2 distinct analysis, meaning we do not take into account whether the two analyses of the same alert are the same (i.e. consistent).

5.4 Verification of model completeness and accuracy

Having constructed the model from the sequence of steps, we analyze the completeness and accuracy of the model as a whole.

5.4.1 Completeness. As the model is meant to encapsulate the entire analysis of a tier 1 analyst, we must verify that the model is complete. In order to measure the completeness of the model, we again utilize the experimental results. From the results sheet, the completeness can be qualitatively determined using mandatory free-text comment boxes. These comments boxes contain comments about specific model stages, as well as the entire analysis and model limitations are be recorded there.

5.4.2 Accuracy. To measure the accuracy of the model, i.e. to what extent the classifications of alerts are correct, we again compare the experimental results to the ground truth and compute the rate

at which respondents perform the assessment correctly. Here we consider the assessment to be correct, only when both respondents are in agreement. Concretely we utilize a confusion matrix, to compute the sensitivity and specificity of the model. The sensitivity (or true positive rate, TPR) is computed as follows: $TPR = TP/P$, were TP is the number of correctly classified "interesting" alerts, and P is the total number of alerts classified as "interesting" in the ground truth. The specificity (or true negative rate, TNR) is computed as follows: $TNR = TN/N$ were TN is the number of correctly classified "uninteresting" alerts, and N is the total number of alerts classified as "uninteresting" in the ground truth.

6 MODEL DESCRIPTION

6.1 Initial generic sequence of steps

As a first step in identifying a generic sequence of analysis steps, 10 random alerts from the SOC are sampled. These alerts are analyzed and classified by the author of this paper with approximately 3 years of experience, following common practices.

From the resulting analyses, we identified 13 categories of information which form the steps of the model and encapsulate the tier 1 analysts' process. The 13 steps are further combined into a total of 4 sequential model stages. Each stage allows the exploration of closely related steps and the identification of necessary evidence to classify the alert as interesting, a lack of this evidence indicates uninteresting alerts. The following model stage have been identified:

- Relevance Indicators: this stage aims to identify whether the alert under investigation is even relevant for the owner of the monitored environment. This is based on the signature and the scope of the customer, and gives an initial opportunity to prune uninteresting events when one determines that there is no evidence of the alert being relevant.
- Additional Alerts: In this stage one identifies other alerts that have triggered within the monitored environment and reasons about the alert under investigation based on the history of that alert triggering. As well as other alerts which may surround the alert under investigation. This way, alerts that have never been interesting in the past can be identified, which allows one to compare alerts; if exactly the same is observed one can exit the model. The other alerts that may have triggered surrounding the alert investigation, may add evidence that there is indeed a potential attack happening.
- Contextual Information: This stage focuses on the behaviour and observables of the involved internal host. It identifies logs which may be related to the alert under investigation, investigates the traffic stream which triggered the alert and the internal hosts behaviour. Furthermore, if any information about the internal host is known, such as OS, software versions etc. it is utilized at this stage as well.
- Attack evidence: This stage focusses on identifying the potential attack. Firstly information is gathered about the alleged attack and this is matched against the observed behaviour in the evidence collected so far, this way one can see if it is an actual attack happening. Similarly, this stage gather information about what would be observable if the potential attack is successful, and again this is matched against already collected evidence. Then, the attacker is investigated, to identify if the potential attack comes from a known, trusted source. And finally, the collected evidence is reviewed and matches against the use cases of the monitored environment. This way, the owner of said environment can be informed appropriately.

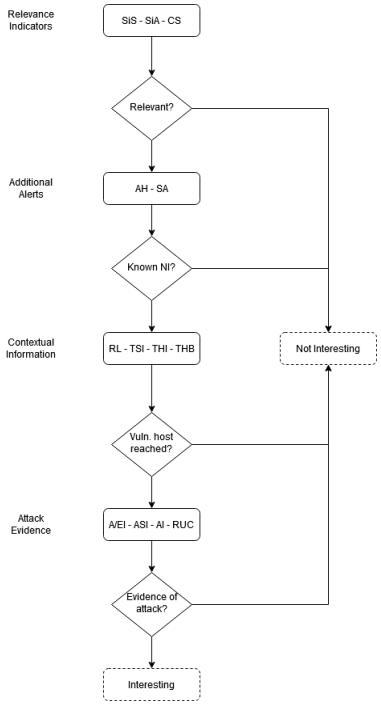

Figure 3: Overview of the model

Each of the stages aims to answer a general question about the alert under investigation, and allows one to exit the model based on the answer given. The questions are listed below, in the order in which they appear in the model:

- Is the alert under investigation relevant to the owner of the monitored environment?
- Is the alert under investigation a known not-interesting?
- Is there evidence that the potential attack reached a vulnerable hosts?
- Is there evidence of a successful attack?

Each model stage, and the steps it includes, is described below. For the steps we use abbreviations throughout the thesis, these are explained in the individual stage sections, but can also be viewed in appendix A.

6.2 Relevance Indicators

The first stage identified as part of the model is relevance indicators, a collection of steps aimed at determining whether a triggered security alert is relevant for the affected customer. This is achieved by three specific steps; signature specificity(**SiS**), signature age(**SiA**) and customer scope(**CS**). This model stage attempts to answer the following general question: Is the alert relevant to the customer?

The is an initial assessment of an alert, without drilling down into the details of the event which triggered it. Purely based on the quality of the signature and the scope of the monitored environment, the analyst determined whether the alert is relevant or not. There is no room for interpretation, if the analyst is unsure it should always be considered related. Further model stages will eliminate the possibility of the alert being interesting, if it turns out the alert was in fact unrelated.

6.2.1 Signature specificity. When estimating the relevance of an alert to the owner of the monitored environment, the analyst investigates the general signature quality of the signature that triggered the alert. To determine 'quality', firstly the analyst determines whether the triggered signature is specific to a certain attack or service, or whether it is only a generic indicator. This step allows one to establish the initial priority of the alert analysis (e.g. specific indicators may be prioritised over generic indicators), as well as determine what to investigate in future steps. For example, a signature for DNS request for a suspicious TLD, would be considered generic, whereas a signature for a DNS request for a phishing domain would be considered specific. Generally, identifying that the signature is specific here increases confidence in the event being interesting. Identifying that the signature is generic may decrease the confidence, depending on the level of generality.

6.2.2 Signature age. When looking further into the signature which generated the alert the analyst can also review the signature creation date and last updated date. Using this data, one can quickly see if the behaviour triggering the alert is a recent or an old threat. Again, this allows one to determine the potential severity depending on the age of associated threat, or indicates a potential uninteresting security event, if the trigger conditions of the indicator are time-dependent (e.g. trigger condition is an IP address, which may have been reassigned between the creation of the indicator and the triggering of the alerts).

6.2.3 Customer scope. The analyst then reviews whether the observed potential threat is within the monitoring scope of the customer. This is determined by reviewing the customer security policy, as well as the service level agreements about sub-nets and reporting. The alerts with no to low impact, in for example a guest network of the customer, can in this way be classified early on in the model, preserving analyses effort for more high impact alerts.

6.3 Additional alerts

The second stage of the model investigates the alert history(**AH**) and how the alert under investigation triggered in the past. Furthermore, it identifies potentially related surrounding alerts(**SA**) by looking at alerts triggered by the same involved hosts around the same time of the original alert. This stage attempts to answer the following general question: Is the alert under investigation a known "not interesting" alert? The model can only be exited at this stage, if the alert history indicates a common "not interesting", and the alert contents match one-to-one. If this is not the case, the presence of previous occurrences of this alert, or of surrounding alerts may add evidence that alert is interesting.

6.3.1 Alert history. First the analyst investigates the history of the alert under investigation; how often has it triggered in the past, how often was it interesting, has it triggered for the same internal host before. Using this information, an analyst can verify quickly whether the observed attack (assuming it is an attack) was not successful in the past. Having verified the past occurrence to be identical and not interesting, the alert under investigation can be classified identically as well.

6.3.2 Surrounding alerts. Besides looking at the history of the alert under investigation, the analyst also investigates similar alerts which were triggered by one or both of the involved hosts, around the time of the potential attack. When looking at these surrounding alerts, analysts may observe different alerts with similar names, indicating the same potential attack. This adds evidence that the event underlying these alerts, is interesting. Or the analyst may observe alerts for different attack stages. For example, investigating a malware alert, a surrounding alert may be for a CnC. Identifying these surrounding alerts allows the analyst to get a complete picture of the attack happening and at the same time it gives the analyst more evidence that there is in fact an ongoing (attempted) attack. It is worth noting that not every surrounding alert is relevant, this depends on the type of alert and it is up to the analyst's expertise to identify those that are, if any.

6.4 Contextual information

The third stage of the model investigates the context around the alert under investigation. The context provides meaningful insight into what type of host is being attacked, what the actual traffic generating the alert is and whether there is any follow-up traffic which may be generated as a result of the alert under investigation. More specifically, the analyst determines a set of related logs(**RL**), which contain information about the cause of an alert, or the result/impact of that alert. They investigate the traffic stream information(**TSI**) which triggered the alert and finally review any target host information(**THI**) available. Having done so, the analyst finally looks at the target hosts behaviour (**THB**) to determine if the system was impacted by the attack. The general question we attempt to answer in this model stage is the following: Was a vulnerable host reached by a potential attack? At this stage, the analyst collects concrete evidence generated by the security systems (Zeek, in the experimental setup), and information provided by the owner of the monitored environment. Only if at this stage the analyst determines there is zero evidence of potential attack reaching a vulnerable host is the model exited. Otherwise, evidence to the contrary is identified and the model is further executed.

6.4.1 Related logs. To get a clear and complete picture of what is happening on a monitored host, this model step focuses on identifying logs which may impact the cause or the outcome of the attack under investigation. This selection is largely dependent on the type of alert, and the type of traffic it is triggered on. In general, the **RL** consists of at least a connection log, a protocol specific log (such as HTTP, or SSH), in addition to the alert log. Furthermore, any other logs generated by the defending host of the protocol in the alert or DNS logs are typically related. Generally, the **RL** are from the time around the alert. When identifying **RL**, a full overview of all logs

surrounding the alert is also viewed, and from this any unexpected logs may also be added to the set of **RL**. For example, investigating an exploit attempt against a server in the customer data center, a log of a successful SSH connection from the internet, generated just after the attack, could very well be included in the set of **RL**.

6.4.2 Traffic stream information. Having identified a set of **RL**, the analyst has now an idea of the context of the attack and the traffic which triggered the alert. The analyst can view more detailed information about the number of bytes and packets sent by the attacker and defender, by reviewing the **TSI**. The aim is to identify if the amount of traffic can be considered normal in this context. This allows analysts to easily identify failed port scans by viewing if a response packet was sent to the scanner, this also allows analysts to determine whether any 'lucky hits' were possibly generated. A 'lucky hit' occurs when the trigger conditions of a signature are met by pure chance on a random sequence of bytes or characters, and thus trigger on traffic they are not supposed to trigger on producing false-positives (or "uninsteresting" to stick with the convention of the thesis). Observing a very large traffic stream, indicates to the analyst that the chances of a lucky hit larger than compared with small traffic streams, so this would indicate that an analyst should take the possibility of lucky hits into account. Moreover, the number of packets and bytes in a traffic stream, may indicate that the connection associated with that stream is successful or not successful, which may later be used to reason about **A/EI** or **ASI**.

6.4.3 Target host information. This stage considers information about the targeted host. This information can vary greatly from SOC to SOC and even from monitored environment to monitored environment. In general, however, the analyst can utilize information about the host, such as whether it is a desktop or a server, its purpose (for example DNS server), its OS, its associated sub-net, host name, open ports and so on, to reason about whether the attack under investigation can ever lead to successful violation of corporate policies. Clearly, investigating a Linux-specific exploit while the targeted host is running, say, Windows, is valuable information that allows an analyst to quickly eliminate cases where non-vulnerable hosts were reached by an attack.

6.4.4 Target host behaviour. Next to utilizing known information about the targeted host, the analyst can also review the target hosts behaviour. For this, the logs produced by the IDS and network sniffer are utilized and are used to review the behaviour of the host before and after the attack. If there is a significant difference, this may indicate that the host was impacted by the attack. We can also more generally review the hosts behaviour, for example weekly occurring backups or updates can be identified and this can help in identifying false positives.

6.5 Attack evidence

In the fourth and final model stage the analyst knows a potential attack reached a vulnerable host, and investigates the presence of attack evidence. First the analyst identifies the exact type of attack by looking at the attack/exploit information(**A/EI**). From the attack/exploit information, they investigate potential indicators of a successful attack by reviewing the attack success indicators (**ASI**) for this specific attack and the attacker information (**AI**). Finally, knowing whether an attack took place, and whether it was successful, they review its relation to the use cases(**RUC**) for the associated environment. This final step allows the analyst to correctly classify the alert and determine its impact on the environment. In this final model stage, the model answers the following general question: Is there evidence of a successful attack? At this stage, the model can be exited if there is no evidence of an attack, or if the attack was not covered by the use cases or unsuccessful.

6.5.1 Attack/Exploit information. In this model step, the analyst determines the exact attack, tools involved and associated hacking groups. The information required for determining this mainly originates from the signature which triggered the alert, and often a reference to source about the attack is included as well. Furthermore, an analyst utilizes any open sources available to get a full picture of what attack is occurring. From this step, it should be clear whether there is an attack, and if so what attack specifically. Using this information, again the analyst estimates the impact on the customer.

6.5.2 Attacker information. Similar to investigating the attack, the analyst can also investigate the behaviour of the attacker (or at least of the attacking system). Using the logs generated as a result of the attacker behaviour, as well as using public sources, the analyst can determine whether the attacker is an actual attacker. This step is needed to distinguish actual attackers from known and trusted sources such as (vulnerability) scanners, as well as help identify false positive alerts generating 'hits' on backup streams, update and executable downloads.

6.5.3 Attack success indicators. Having determined which attack reached a vulnerable host, the analyst now investigates whether the attack was successful. Again, analysts use information obtained from former stages and open sources that identify clear indicators of successful attacks. For example, when investigating a Mirai botnet alert, an analyst may observe a second-stage download location in the payload of the request when reviewing the related logs, or surrounding alerts. This download location, typically an IP but sometimes a domain, can then be used as an attack success indicator; if the analyst sees a connection to this IP/domain after the alert has been set up from the defending host, they can be confident that the initial exploitation was successful. Generally, the attack success indicators are highly dependent on the specific attack, however, generic indicators such as DNS request for strange top-level domains, rapid connections or internal scanning can often be used as well.

6.5.4 Relation to use cases. Finally, knowing a successful attack took place, the analyst consults the use cases for the affected environment. This step helps them to correctly identify the full impact for the environment, and thus the final classification of the alert. It also eliminates any alerts which are not important to the environment. For example, investigating a generic malware alert, having determined that it is actually adware on a desktop, the use cases may call for no action at all, depending on the environment. Or a non-successful port scan, in a highly secure sub-net, might be highly interesting in a specific environment and may therefore be escalated. Finally, the use cases provide useful information and

guidance on what and how to report to higher tier analysts or the affected customer.

6.6 Alert classification

Using the model detailed above, analysts can systematically classify alerts; at any model stage, the model can be exited. Exiting the model prior to the final stage (attack evidence), implies the alert is classified as "not interesting". If the final stage of the model is reached, then an alert is only classified as interesting if there are actual indicators of a successful attack and the attack is covered by the user cases. As highlighted earlier, this is the classification of a tier 1 analyst. Further classification of the alert may be performed by higher tier analysts, and the classification may differ from the original one.

7 EXPERIMENT EXECUTION

7.1 Experiment participants

For this experiment, five analysts in an operational SOC have been recruited. The analysts are SOC intern computer science students specializing in cyber security. These students have been trained to be tier 1 analysts and have approximately 3 months of part-time experience in that role. The subjects exclusively have experience analyzing network security events generated by Suricata, network logs produced by Zeek and the Security Onion Dashboard and as such the experiment environment as exposed to the subjects is equivalent to the real environment they are used to. This reduces the chance of errors and eliminates a learning curve for new tools.

7.2 Experiment preparation

To prepare the experiment participants, an hour-long training was prepared explaining the model, its stages and steps, in great detail. The experimental setup and constrains and the results sheet were conveyed as well. The slides from this training were provided to the subjects after the training, to be used as a guide during the experiment.

For the duration of the experiment, the subjects are forbidden from communicating with each other about the experiment. To enforce this, a schedule was maintained which assures no subjects work on the same day as another subject twice, i.e. having worked on the same day as another subject, these two subjects will not work again on the same day. Furthermore, this schedule was distributed to the subjects individually, assuring the subjects are not aware of which subject is working on the same day as them. Additionally, the subjects have no knowledge about alerts assigned to other subjects.

Given that the subjects have experience as tier 1 analysts, for the accuracy of the experiment, they were not allowed to ask questions about how to analyze an alert, as the model should provide enough basis for this next to their previous experience. The subject is only permitted to ask questions about unclear details of the model or the provided results sheet. These questions are exclusively allowed to the researcher supervising the experiment.

7.3 Environment

The environment for the experiment was designed to be as representative of a real operational SOC as possible. To this extent, a slightly customized Security Onion environment was set up. Security Onion is an all-in-one solution for SOCs, it includes a Suricata for detecting network attacks, and Zeek for logging all traffic. For Suricata we deployed the open source Emerging Threat Open ruleset, as well as the licensed Emerging Threat PRO ruleset. From these signatures a subset was disabled according to a configuration used in production in the SOC the subjects are recruited from; these signatures typically generate false positives or simply too many alerts. The disabled signatures include, but are not limited to, any hunting, policy or info signatures.

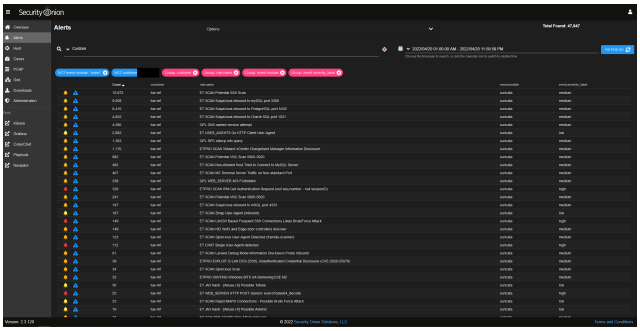

Figure 4: Security Onion Dashboard

Furthermore, the environment includes the Security Onion Dashboard which is visible in figure 4, which allows analysts to interact with alerts and logs and query for data. The Security Onion Dashboard is the primary tool for investigation in the experiment, and displays the current unresolved alerts, along with the alert name, customer and alert severity. Connected to the Security Onion sensor is a real office network segment of the university, with over 500 unique hosts and multiple DNS and file servers. Test subjects exclusively use Suricata or Zeek logs, or public sources for the investigation of alerts in the experiment.[1]

Next to monitoring a real office environment, the Security Onion instance was also connected to a local deployment of SAIBERSOC. [9] SAIBERSOC is a tool that allows researches to inject attacks in environments, while customizing the traffic to be indistinguishable from real traffic. This feature is key, for we must inject attacks into the environment in order to test the model on true-positive successful attacks. And these injected attack must be indistinguishable in terms of source (sensor) and destination (in customer IP range) from the real alerts generated by actual traffic, otherwise the experiment results will be unreliable.

Finally, test subjects use their personal computers to access the experiment environment over VPN, and next to the Security Onion Dashboard they may access the internet for any OSINT investigations.

7.4 Sampling

To validate the model, a sample of the alerts generated and injected into the experiment SOC must be analyzed. This sample

[1]The Security Onion enviroment (sensor, storage and master node) were deployed in a distributed manner on left-over hardware. This meant that PCAP retention is severely limited, to approximately one hour. For this reason, as well as that tier 1 analysts in this environment do not necessarily require PCAPS, led to the decision to exclude PCAPS altogether from the experiment.

must includes a range of alert types, and most importantly, actual successful attacks.

Analysis of the data collected by Security Onion over the course of two-and-a-half weeks, identified the generation 150 unique alerts in total and on average 60 unique alerts per day. In total, 350.000 alerts were generated and over 160 million logs were created. Close to 100 million connections were attempted or established and 48 million DNS requests were made.

To generate our sample to give to the analysts, we sample alerts from 9 AM to 5 PM, ensuring representative traffic of an office environment. The experimental sample was chosen to be 200 alerts, to be analyzed twice for measuring consistency. Because each alert is analysed independently by two analysts, in total 400 alert analyses are conducted.

From analysis of the experiment sensor data, it was observed that the alerts generated are predominantly scans, and exploit attempts from the internet, exceeding 90% of all alerts generated. For this reason, a stratified random sampling method was devised to assure good coverage of all alerts.

Firstly, we categorize all triggered alerts within the sampling window based on their included rule category[8]. The rule category captures the type of threat, such as scan, malware or exploit or it refers to a tactic or technique used by attackers. From the rule category, the analysts prioritize open alerts based on overall severity and the policy associated with the monitored environment. It also can also be used as indication of the type of traffic which triggered the alert. For each of the rule categories, we randomly select a unique rule, and from that we sample a random alert generated by that rule. This way, for each category at least one alert will be included. Since the injected attacks generate alerts of rule categories only seldom triggered, at least one alert of the injected attack is present in the sample as there will be no other (non-injected) occurrences of these alert categories.

Multiple alerts of the injected attack may be included, if the alerts result from randomly sampling more common rule categories. Additionally, as far as possible, five new instances of alerts from the previous experiment day, are included in the next days sample. These are randomly sampled from the previous day, and if the alert did not occur on the next day, we re-sampled until one was found which did occur. This allows us to compare the analyses of multiple unique occurrences of an alert; further, this replicates a real environment in which this is often the case.

7.5 Attack Injection

To investigate the validity of the model for true-positive successful attacks, on every experiment day a (multi-stage) attack is injected. The attacks are acquired from malware-traffic-analysis.net[7]. This website hosts many PCAPs with malicious network traffic and analysis of the associated (multi-stage) attacks. The injected attack were selected on being multi-stage if possible, but most importantly they should generate meaningful alerts and logs in the experiment environment. The attack are detailed in appendix C.

Having verified the attacks generated alerts in the experiment environment, the environment was fully cleaned before experiment start to leave no trace of the injected attacks. The attacks are injected at random times within the sampling window of the associated experiment day. Additionally, the internal IP addresses in the attacks was modified to be random unassigned IP addresses within the experiment network. An exception to this are the DNS servers used in the attacks, these were re-set to be the actual internal DNS servers in that sub-net, as this is the corporate policy for that sub-net.

Unassigned IPs were chosen, to avoid conflicting real information in the logs of that host impacting the analyses of the alerts in the sample. External IP addresses were not changed, to allow real investigation of the external hosts conducting the attack, using OSINT. Using SAIBERSOC, the attack are injected on the same interface the office network is connected to, to make these injected attacks appear to be originating from the office network as well.

7.6 Experiment execution

Alerts to investigate are passed to the analysts as entered into a standardized Excel sheet for that day mentioning the rule name, category and a direct URL to the selected alert in the experiment environment. Furthermore, the Excel sheet contains criteria to include to measure the model, more on this in another chapter. After producing the sample and the Excel sheet, the next day the experiment subjects perform the analyses.

In order to balance the workload of the experiment participants, and control the time when the analyses are performed, the experiment took place over the course of 10 days. On each day, two analysts analyze the same 20 alerts. At the end of the day, the results are collected and verified to be complete and corresponding to the model steps.

7.7 Experiment results collection

In order to capture the results of the experiment in a meaningful way, which allows us to compare analyses directly, a standardized results sheet was created. The results sheet, delivered to the experiment subjects at the start of each experiment day, requires the collections of predefined pieces of information. For each stage in the model, as well as for each category of the model, questions have been designed with multiple-choice answers. The questions are aimed to collect and summarize the particular model step outcomes.

Additionally, for steps where a multiple-choice answer was not sufficient and/or the step is complex, an additional free text field is included to provided evidence for the multiple-choice answer chosen. This is also where test subjects record any missing model features, or other limitations in their ability to analyze the alert. A detailed overview of the results sheet formulation is provided in appendix B.

All fields in the results sheet are mandatory and must be filled in. The experimental results as produced by the experiment subject, are not edited by anyone except the original subject. When verification of the results shows horizontal inconsistencies or missing data, test subjects are asked to correct the errors themselves, immediately. Only after approval by the experiment supervisor are the results of an experiment day final. After receiving all final experiment results, these are combined into one results sheet on which the data analysis is performed.

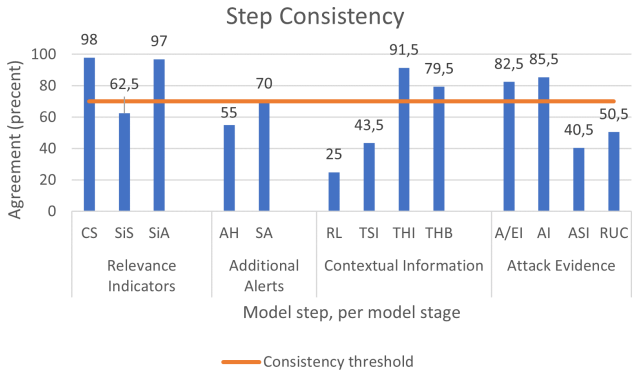

Figure 5: Consistency rate of model steps

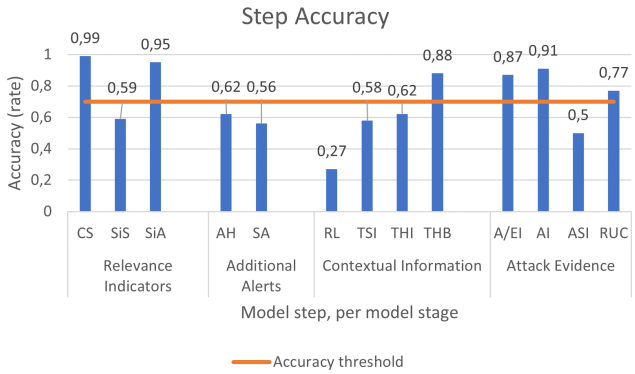

Figure 6: Model step accuracy

8 EXPERIMENT RESULTS

The experiment as described above was executed without technical or procedural issues. All 400 alert analyses were collected and a ground truth of 50 alerts analyses was created for result verification.

8.1 Step consistency

In order to measure the consistency of the model steps, we compute how often test subjects agreed in their analyses. Only when the two subjects of the same day agree, the result is considered consistent. The computed step consistency rate is visible in figure 5. This bar chart shows the agreement rate between the 2 experiment participant who analyzed the same alert, per model step. The horizontal orange bar indicates the consistency threshold of 70%.

8.1.1 Relevance indicators. First of all, for **CS** we recorded the agreement rate at 98%, so we can consider this step to be consistent. Then for **SiS** we achieved an agreement rate of 62.5%, so by our definition this step is inconsistent. For **SiA** we found an agreement rate of 97%, indicating this step is highly consistent. Overall this model stage performs well, though improvements should be made to **SiS**.

8.1.2 Surrounding alerts. Firstly for the **AH** we achieved an agreement rate of 55%, meaning this step is inconsistent. The for **SA**, we observed an agreement rate of 70%, which makes this step consistent. Given this poor to mediocre overall performance, this model stage requires improvement before it we can utilize it for further research or in practise.

8.1.3 Contextual information. The **RL** proved to be the worst performing step in the model, with an agreement rate of 25%, thus this step is highly inconsistent. The **TSI** performed slightly better, with an agreement rate of 43.5%, meaning that also this step is inconsistent. For **THI**, we achieved an agreement rate of 91.5%, so the **THI** can be considered consistent. Finally, for the **THB** we achieved an agreement rate of 79.5%, therefore we consider **THB** to be a consistent step of the model. While the steps of this stage related to the targeted host perform well, the **RL** and **TSI** step perform poorly and requires improvement.

8.1.4 Attack evidence. First of all, for **A/EI** we found an agreement rate of 82.5% indicating consistency. For **AI** we observed similar results, with an agreement rate of 85.5%, making this a consistent step as well. Then for **ASI** we found an agreement rate of 40.5%, indicating this step is highly inconsistent. Then finally for **RUC** we observed an agreement rate of 50%, making this step inconsistent as well. Overall the steps in this stage related to attacker or attack information perform well, with the exception of success indicators, which requires improvement. Additionally, the relation to the use cases also performed poorly and will require adjustments.

Overall, we observe that 6 out the 13 steps in the model yield inconsistent results. Out of the model stages, we found the steps in the stage surrounding alerts to be the most inconsistent.

8.2 Step accuracy

We have found that the step accuracy was highly correlated with the step consistency, i.e. an inconsistent step is also mostly inaccurate. This is unsurprising as for each step only one assessment is deemed as 'correct', meaning that any inconsistency would also lead to a lower overall accuracy in the assessment of that step. The step accuracy is reported in figure 6. The figure shows the agreement between an experiment participant and the ground truth. The horizontal orange bar corresponds to the accuracy threshold of 70%. For each model stage, we describe the accuracy per step below.

8.2.1 Relevance indicators. Firstly for **CS** we found an accuracy of 99%, with a single result being different from the ground truth. For **SiS** we observed an accuracy of 59% making this an inaccurate step. Finally for **SiA** we observed an accuracy of 95%, with only a single occurrence where both test subjects were incorrect in their analysis.

8.2.2 Additional alerts. The **AH** proved to be inaccurate with a rate of 62%. Similarly, for **SA** we observed an accuracy of 56%.

8.2.3 Contextual information. For the contextual information we observed mostly inaccurate results. The **RL** yielded an accuracy of just 27%, where only 4 out of the 50 alerts of the ground truth were correctly labeled by both test subjects. This is by far the worst performing model step. For **THI** we observed an accuracy of 62%

Table 1: Confusion matrix based on ground truth

n=50	pred. interesting: 21	pred. uninteresting: 29
actual interesting: 22	18	4
actual uninteresting:28	3	25

and for **TSI** we observed inaccurate results, with a rate of 58%. Finally for **THB** we obtained an accuracy of 88%.

8.2.4 Attack evidence. Firstly for the **A/EI** we found an accuracy of 87%. For the **AI**, we achieved an accuracy of 91%. Then for the **ASI** we observed poor accuracy of only 50% and finally of the **RUC** a good accuracy of 77%.

Overall, we observe that 7 out of the 13 steps in the model yield inaccurate results. Similar to the step consistency, for step accuracy we observe that the steps in the stage additional alerts mostly yields inaccurate results. Additionally for the step **RL** we found very poor accuracy at just 27%. Interestingly, **THI** is consistent, but not accurate. And for **RUC** we obtained inconsistent, but accurate results.

8.3 Model accuracy

Table 1 reports the confusion matrix of the results against the ground truth. A Chi-Squared test shows that the distribution of outcomes is significantly different from what expected from a random association, suggesting that analysts where better than chance at identifying the correct category for the classification ($p = 1.861e^{-6}, df = 1, \chi^2 = 22.734$). We can determine the accuracy of the model by computing the sensitivity (i.e. true positive rate) and specificity (i.e. true negative rate) using the confusion matrix in table 1. From this we obtain a sensitivity of 0.818 and a specificity of 0.893 which indicate great accuracy.

8.4 Model completeness

For the completeness of the model, the free-text comment fields were analyzed for model steps as well as for the final classification. Here it was found that experiment subjects occasionally required PCAPs for analysis. This can be classified as a comment on the available data to the test subjects and is not a model limitation. The majority of the comment fields contain comments about which data source was used to perform a given model step. While some of these comments did include mentions of missing logs, and therefore the inability to classify an alert accurately, this again is not a model limitation but and issue of available data. This corresponds with the initial findings by the two experts, who gave their sign of approval that the model was complete, and capable of capturing all relevant steps of the analysis process of tier 1 analysts. Given this, we consider the model to be complete.

9 DISCUSSION

The results above show varying levels of reliability of the model. In this section we discuss these in detail, as well as providing a perspective on implications of this work on both research and practice, and discuss future steps to improve the model.

9.1 Which sequence of steps determine a generic model to analyze alerts for tier 1 analysts?

The sequence of steps identified at the start of the research is verified by two independent experts within the cyber security field. The experts conducted analysis of security events using exclusively the model and commented on its performance. Firstly, they identified that although the model was complete, the question corresponding to the model stage "contextual information" was too narrow to be applicable to all alerts in some cases. A new question was identified based on this feedback, which allows more (later verified to be all) alerts to be captured.

Next, they identified that several model steps' definitions were unclear or overlapping. This resulted in a more concrete definition of several of these labels, as well as several model steps being split up into multiple steps. This aided in the atomicity and understandability of the steps.

The overall changes as a result of the iterative verification process with the experts were quite minor, and having implemented the changes, the experts agreed that the model is complete and encapsulates the general analysis process of tier 1 security analysts in a SOC. The concrete changes made to the model as result of the experts can be viewed in appendex D.

9.2 Are the resulting steps accurate and consistent?

From the results above, we can observe that the many of the steps produce inconsistent results and that accuracy is largely the same as consistency. In order to explain this, we qualitatively analyzed the results and the comments.

9.2.1 Relevance indicators. For relevance indicators we found that the **CS** is highly consistent and accurate. This is believed to be largely due to the definition of "in scope" and the experiment set up. In a real SOC environment, however, the scope is also clearly defined, so we believe the performance of **CS** in the experiment to be representative of real world performance.

For **SiS**, we collected primarily inconsistent and inaccurate results. The inaccuracy and inconsistency can be attributed to an unclear definition of specific versus generic. We believe the general idea behind this step was clear, however, the edge cases caused some confusion. To mitigate this a set of clear conditions to identify a specific signature should be identified.

For **SiA** we found highly consistent and accurate results, possibly due to the clear definition of the step.

Given the inaccuracy and inconsistency of **SiS**, the definition of generic versus specific signatures should be concretized and made more clear. Afterwards, the experiment should be repeated in order to identify whether this model step is consistent and accurate. For **CS** more research can be performed in to how to automatically tune a ruleset to subnet or customer, by for example labeling subnets at a certain level of risk or trust, and correlating this with rule severity as defined in the signatures. For **SiA** more research can be done into assocating the dates in the signature with the relevance of the threat they are detecting. For example, an old signature as determined by the dates in the signature, could be enriched with the

release date of associated CVE number, to give a better indication of the age of the threat as well.

9.2.2 Additional alerts. For the additional alerts we captured results by means of questions and free-text comment box per step. For **AH**, we found primarily highly inconsistent results. The comment field revealed that the definition of the **AH** was not clear. There was confusion between whether it was the very first time an alert triggered in the SOC altogether, or whether it was the first time of that analyst seeing that alert. This lead to a significant difference in results: one alert was classified as "first occurrence" by one test subject, and "as Typically NI" by the other, stating over 6000 of these alerts had triggered in the past. Due to this we cannot draw any definite conclusion on the performance of this step. For the **SA**, we again found inconsistent results though less so than **AH**. Also, the accuracy was bad. From the comment field we identified that there was a misinterpretation of how the **SA** add evidence. Often, when classified as "adds evidence" the comment is "multiple alerts of same attack or scan", indicating that multiple of occurrences of the same alert adds evidence of a successful attack. This is not strictly true and not the intention of the this category; only different alerts about the same attack, or alerts about different attack stages really add such evidence. Also, often this category was (partially) confused with **AH**. We believe this issue partially arose as a consequence of the difference in knowledge by the model designer and verifiers and the test subjects. Future designs and adjustments should therefore include tier 1 analysts to verify the definition and scope of steps and stages.

Having observed poor performance for the steps in additional alerts, it is clear that clearer definition of **AH** and **SA** must be provided. Additionally, for the future we aim to include tier 1 analysts into the design process as well, as **SA** highlighted that there is a significant knowledge gap between analysts of different tiers which warrant collaboration to ensure usability of the model.

9.2.3 Contextual information. Firstly, for the **RL** we observed inconsistent results. It is worth reviewing the possible results here; "nothing", "logs which indicate the event which caused the alert", "logs which indicate result/impact of the event causing the alert" and "both". Most occurrences where the test subjects did not agree, are answers where one test subject answered "both" and the other answered "logs which indicate the event which caused the alert" or "logs which indicate result/impact of the event causing the alert". We observed only a few cases where one of the subject answered "nothing" and the other subject did not answer "nothing". This indicates to us that if there are **RL**, they will be identified by analysts using the model, however the completeness of the set of **RL** is not consistent. We believe adjusting the possible results to reduce the complexity will improve the consistency. For this, we should first ask for logs indicating the event which caused the alert, and then ask for the logs indicating the result/impact of the event causing the alert, separately. Similarly for the accuracy of the **RL**, this is quite poor yet there were but a few cases where "nothing" was identified by subjects while the ground truth states another outcome.

Secondly, for the **TSI** we obtained inconsistent results. Here, we clearly observed a misinterpretation as visible from the comment fields of **RL**. Here, subjects often claimed a "normal" sized connection, explaining in the comment field that a single SYN scan packet was not met with a response packet. Clearly, a single packet "connection" is not a normal sized connection for TCP connections. This also compromises the accuracy of this category. The combination of misinterpretation and general knowledge level of the experiment subjects, again leads us to believe including tier 1 analysts in the design and verification process is necessary. Moreover, the definition should be more clearly defined.

Finally, for the **THI** we observed good consistency but poor accuracy. The consistency is likely due to the fact that this is a clear process, with predefined information sources and assumptions by the experiment environment. Yet it is representative of real performance, as the aforementioned process, information sources and assumptions are based on the real SOC environment. The accuracy however is at 62% which is just below our accuracy threshold. We believe this might be due to the limited information in the aforementioned information sources, where experience and knowledge is required to correctly identify this. For the **THB** the results also proved to be accurate and consistent. Again, the definition and scope aided in this.

From the results of **RL** it is clear that this model step needs to be more concrete and defined more clearly. To this extent, clear constraints for "related" should be defined. Having this definition, executing the experiment will lead to more consistent accurate results in our opinion. Next, for **TSI** again there should be clear constraints and definitions of when a traffic stream is considered to be small, normal or large. With the new definition we expect the results of this step to be consistent and accurate.

9.2.4 Attack evidence. For the first category **A/EI**, we found satisfactory consistency and accuracy. As the test subjects have access to any information included in the signatures, and any open sources, they are capable of performing in-depth research about attack indicators. Together with the available logs, we believe the test subject were well equipped to assess whether there was an actual attack. In cases of doubt, the subject should err on the side of caution and answer "attack", as to escalate it to a tier 2 analyst, given that false-negatives are unacceptable. We believe this contributed to the consistency and accuracy, and again this is representative of real SOC operations. For the **AI** we also observe high consistency and accuracy. This can be explained by the availability of open sources and again airing on the side of caution in the classifications. Only if a host is undeniably trusted, should this option be chosen. This strong constraint aided particularly in the accuracy of the outcomes.

Then for **ASI** we observed inconsistent and thus inaccurate results. We believe this is due to the way results were captured, the options being: Definitely unsuccessful, successful and unknown. Including the option "unknown", led to the individual test subject's knowledge level and experience being a big factor. Test subjects were observed answering "unknown" when they did not know how do perform the analyses of an alert. Meanwhile, the goal of this outcome is to capture instances where the success of an attack cannot be determined from the logs. To better capture this category's performance, this option should be removed and test subjects should output "successful" when they are unable to determine that it is "definitely unsuccessful". That being said, there were only a few occurrences were one test subject classified the **ASI** as "successful"

and the other subject answered "definitely unsuccessful". Therefore, we believe this category performs better than the data suggests.

Lastly, for the **RUC** we observed inconsistent but accurate results. We believe this is mainly due to the the lack of there being an actual customer policy and use cases for the experiment setup. A clear set of use cases alongside a customer security policy, will allow analysts to perform this analysis fully, leading to more consistent results. A possible reason for this step being accurate despite the inconsistency is that on average the test subjects were knowledgeable enough to make the distinction based on their past experience. Implementing the changes for the consistency, might improve the accuracy as well.

We note that the tier 1 analyst performed well on this stage, where several OSINT investigation had to be performed. This indicates that though a very important stage, it is not as complex as thought earlier. However, for **RUC** we found very poor results. For this reason, we must incorporate the use cases of customer better into the process of tier 1 analysts. Also, the use case must be properly defined, to allow analysts to make use of them. It is also this model step, that should be focused on when training new analysts.

9.3 Is the resulting model accurate and complete?

As visible from the results, the model overall is accurate with a sensitivity of 0.818 and specificity of 0.893. These results differ significantly from the accuracy of the individual model steps. This difference can be explained by the model ability to handle errors; whenever an error is made at an early step, and the model is not exited with a classification of "not interesting", at a later model step conflicting or no evidence will be found, which in turn means the model will be exited anyway. Due to the model natural resilience to errors in the steps, the overall accuracy of the model improves at the cost of individual step accuracy. This is a positive feature of the model, given that the overall classification of an alerts outweighs the intermediate classifications made throughout the analysis process.

Furthermore, we have also determined the model to be complete from the qualitative analysis of the experiment results. Participant of the experiment did comment on the absence of PCAPs. This is not a limitation of the model however, as PCAPs can be considered to be part of **A/EI** and **ASI**. Also occasionally comment were left about the absence of logs, again this is not a model limitation. Very rarely there were comments about an experiment participant not being able to analyze an alert, due to limitations in their knowledge. Given that the model is designed to guide an analyst, the model should support the experiment participants in these cases. Given that we observed high accuracy, we believe that the model is sufficiently capable of this and therefore we also consider the model to be complete.

9.4 Implications for research and practise

9.4.1 Implications for research. The identified sequence of steps lays the foundation for various further research possibilities. Firstly, having a complete and accurate sequence, allows researchers to capture attack and analysis information about specific steps in the analysis process. This, in turn, allows the identification of bottlenecks or easily automatable steps. It also allows researchers to measure the quality of new solutions or products. For example, the benefit of a new threat intelligence dashboard could be measured, by analyzing the performance of the **A/EI**, **AI** and **ASI** with and without the dashboard. The model can be further employed as a prioritization tool whereby alerts associated with more evidence that a security violation happened can be discerned from those for which no or little evidence is found. Automated methods for the collection of this information should be developed.

For **A/EI**, given the great accuracy and consistency, this step can potentially be automated. As an automation, we can make use of NLP to interpret the attack or exploit of the signature. Next, using this information, we can query public sources, such as VirusTotal to, security providers' blog posts for attack or exploit information or threat intelligence providers. The exact implementation of this will have to results from future research. Given that the **ASI** is fairly similar to **A/EI** and **AI** we can enrich this model step in a similar fashion, after having verified its performance using the suggested changes to the experiment result capture. Finally, to properly measure the performance of **RUC** we must re-run the experiment with a properly defined customer.

9.4.2 Implication for practice. For use in practise, the sequence of steps proves useful as well. The sequence of steps is already used in training new tier 1 analysts in the ESH-SOC, and it provides a structured method of analyzing to existing tier 1 analysts. Due to this, collaboration becomes easier as pieces of information can be identified by the sequence step it is associated with. Further, the presence of a model over which to run specific automation processes allows outcomes of that automation to be mapped to the reasons why a certain overall classification was given to that input. Similarly, one can use the identified sequence of steps in a decision support system for tier 1 analysts. For example, the application of the model provides a clear progression in the analysis that adds evidence to the potential 'relevance' of the investigated alert. This may serve as a means to prioritize further investigations based on the (automated) analysis performed. As the model highlights which types of information are needed for alert analysis, it can also be utilised to enrich automated reporting of incident and alerts to customers with evidence of attacks. It can also be utilized to automatically generate a likelihood of an alert being interesting, which can be used by analysts to prioritize which alerts to analyze first.

From the relevance indicator results it is clear that **CS** and **SiA** appear to be automatable model steps, due to their high consistency and accuracy. An implementation of such automation of **CS** can be realised by defining rulesets specific to customers and their subnets, this is common practise in the industry. Iteratively, indicators can be enabled or disabled based on alert generation and classification. Furthermore, for **SiA** one can create an automation which parses rules, and computes the age based on the creation and last modification date of the rule. This way, aged signatures which match other conditions (for example, the rule being generic), can also automatically be disabled.

Given the great performance of **THI** and **THB**, automation for these steps appear to be feasible. Firstly, for **THI**, a system

can be designed which intakes information about the customer hosts and networks. This system is most similar to a vulnerability management system combined with a service description per host and network. Using this, the **THB** can also be determined more accurately. The implementation can be a query-able database, or a fully integrated database in the SOC dashboard.

9.5 Limitations

One of the primary limitations of the research is the fact that a bottom-up approach was taken, using 10 alerts from a single environment. Moreover, only 3 experts were available to confirm the validity of the model, associated with the SOC from which the original 10 alerts were samples. This may pose a threat to the validity of the model across other environments, across all alerts which can be observed in a SOC or experts from other SOCs. This limitation was partially mitigated by the experiment, where the alerts were sampled from a different environment and a larger number was sampled. Future research will be required to mitigate the limitation with regards to expert opinion, where other experts from different SOCs apply the model or are interviewed to gain a broader insight into the model performance.

As preluded to in the discussion of the individual model steps above, several results were impacted by limitations of the experiment. This section identifies such limitation and suggests corrections for future experiment executions. Firstly for the **CS** we found that in the experiment environment, it was not possible to properly define a distinct customer scope, as the IPs in the network were quite arbitrary and no information was available for the majority of the hosts. There were no clear subnets were similarly functioning hosts operated, such as a data center. For this reason, test subjects should only answer "not in scope" if the IPs involved in the alert were not within the customer scope. Due to the technical set up, this is near-impossible to occur, resulting in almost exclusively "in scope" results. While this is also expected in most real-world SOC but the least mature ones, a new execution of the experiment, with clearly defined subnets and policies will allow us to properly measure this model step.

Next, for **AH** we found that the method of capturing the answer was resulting in inaccurate results; not all answers were distinct, which meant we cannot clearly distinguish why the results are as they are. Also the definition of those answers should be made more clear; for example, "first occurrence" means first occurrence in the SOC, not the analyst first interaction with this alert. In the future we would split this question up into a yes/no question about it being the first occurrence in the SOC, and a separate question about it being typically FP, or NI (or n/a). This does not affect the model, only the method of results capture. Furthermore, due to the experiment being executed in an experimental setup, there is no proper history of alert classification for this environment. In a real SOC environment an analyst may be able to review previous alerts analyses to aid in this model step. Then, for **SA** we found that mostly the results were impacted by a misinterpretation of the step. A redefinition and scoping of **SA** must clarify this, as it was often confused with **AH**. Furthermore, test subjects were observed to incorrectly label other alerts as adding evidence. This is most likely due to a knowledge gap between model designers and test subjects. In or to mitigate this, a clear definition of when an alert adds evidence to another alert must be defined.

For **RL** we found the method of capturing results to be insufficient for measure the performance of this step. Having an answer encompassing both other answers obscures the true performance of this step. To mitigate this, we split up the question into a question for logs indicating the cause and a question for logs indicating the cause/result of an alert. For **THI** we found inconsistent results, just below our threshold for accuracy. This might be impacted by the limited information known about hosts in the experiment environment.

Then for **TSI** we found that the results as reported by the test subjects are consistently incorrent. This is due to the knowledge gap between analyst tiers. During the experiment introduction and training this step was left too implicit, so to improve this step we must clearly define when which answers is expected.

For **ASI** we also encountered issues in the method of result capture, where an aswer "unknown" was possible. Again, this was misinterpreted; if a test subject did not know what to do, they answered "unknown", despite this being exclusively for cases where the success of an attack could not be determined, due to lack of logs, or encryption for example. A next iteration of the experiment will redefine the "unknown" answer.

Finally, for **RUC** we found poor results due to the lack of an actual customer and associated use cases in the experiment set up. In the future, a (fictional) customer security policy and associated use case should be defined to properly measure the performance of this model step.

Furthermore, due to technical limitations no PCAPs were not available to be used for the analysis of alerts. PCAPs can be considered to be part of **A/EI** and **ASI**, yet they are not always available in real SOCs, nor do they allow for automation as well as logs do. Test subjects only occasionally commented they would have liked the PCAP for an analysis. There was no consistency in when or why they preferred this. Given this, we believe the absence of PCAPs did not impact the performance significantly, and should not be considered essential for this model to perform well.

10 CONCLUSION

Our research shows that the general analysis process of tier 1 analysts, with as goal to identify non-interesting security events in a SOC, can be modeled. The experimental results shows that the designed model is complete and overall yields reliable results, both in term of true positives as well as true negatives. We found that the model can be used for training analysts, or as the basis for technical automations, such as decision support systems. Additionally, the model provides researchers with a basis on which they can perform research on specific SOC elements, for example the performance of new tooling. We furthermore identified that several model steps do yield unreliable results and need to be improved in future research. However, for the model step that perform well, we identified possible automations which can aid in the performance of the SOC.

11 APPENDICES

12 ACKNOWLEDGMENTS

This work is supervised by Luca Allodi.

REFERENCES

[1] A. D'Amico and K. Whitley. 2008. The Real Work of Computer Network Defense Analysts. Springer Berlin Heidelberg, Berlin, Heidelberg, 19–37. https://doi.org/10.1007/978-3-540-78243-8_2

[2] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. In Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (Dallas, Texas, USA) (CCS '17). Association for Computing Machinery, New York, NY, USA, 1285–1298. https://doi.org/10.1145/3133956.3134015

[3] Varun Dutt, Young-Suk Ahn, and Cleotilde Gonzalez. 2011. Cyber Situation Awareness: Modeling the Security Analyst in a Cyber-Attack Scenario through Instance-Based Learning. 20th Annual Conference on Behavior Representation in Modeling and Simulation 2011, BRiMS 2011, 280–292. https://doi.org/10.1007/978-3-642-22348-8_24

[4] Eric T. Greenlee, Gregory J. Funke, Joel S. Warm, Ben D. Sawyer, Victor S. Finomore, Vince F. Mancuso, Matthew E. Funke, and Gerald Matthews. 2016. Stress and Workload Profiles of Network Analysis: Not All Tasks Are Created Equal. In Advances in Human Factors in Cybersecurity, Denise Nicholson (Ed.). Springer International Publishing, Cham, 153–166.

[5] Faris Bugra Kokulu, Ananta Soneji, Tiffany Bao, Yan Shoshitaishvili, Ziming Zhao, Adam Doupé, and Gail-Joon Ahn. 2019. 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 (London, United Kingdom) (CCS '19). Association for Computing Machinery, New York, NY, USA, 1955–1970. https://doi.org/10.1145/3319535.3354239

[6] J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. biometrics (1977), 159–174.

[7] @malware_traffic. [n. d.]. My Technical Blog Posts. Retrieved May 5, 2022 from https://www.malware-traffic-analysis.net/

[8] Proofpoint. [n. d.]. TECH BRIEFET Category Descriptions. Retrieved August 3, 2022 from https://tools.emergingthreats.net/docs/ETPro%20Rule%20Categories.pdf

[9] Martin Rosso, Michele Campobasso, Ganduulga Gankhuyag, and Luca Allodi. 2020. SAIBERSOC: Synthetic Attack Injection to Benchmark and Evaluate the Performance of Security Operation Centers. In Annual Computer Security Applications Conference (Austin, USA) (ACSAC '20). Association for Computing Machinery, New York, NY, USA, 141–153. https://doi.org/10.1145/3427228.3427233

[10] Martin Rosso, Michele Campobasso, Ganduulga Gankhuyag, and Luca Allodi. 2022. SAIBERSOC: A Methodology and Tool for Experimenting with Security Operation Centers. Digital Threats 3, 2, Article 14 (feb 2022), 29 pages. https://doi.org/10.1145/3491266

[11] Reza Sadoddin and Ali Ghorbani. 2006. Alert correlation survey: framework and techniques. 37. https://doi.org/10.1145/1501434.1501479

[12] Ankit Shah, Rajesh Ganesan, Sushil Jajodia, and Hasan Cam. 2019. Understanding Tradeoffs Between Throughput, Quality, and Cost of Alert Analysis in a CSOC. IEEE Transactions on Information Forensics and Security 14, 5 (2019), 1155–1170. https://doi.org/10.1109/TIFS.2018.2871744

[13] Sathya Chandran Sundaramurthy, John McHugh, Xinming Ou, Michael Wesch, Alexandru G. Bardas, and S. Raj Rajagopalan. 2016. Turning Contradictions into Innovations or: How We Learned to Stop Whining and Improve Security Operations. In Twelfth Symposium on Usable Privacy and Security (SOUPS 2016). USENIX Association, Denver, CO, 237–251. https://www.usenix.org/conference/soups2016/technical-sessions/presentation/sundaramurthy

[14] Sathya Chandran Sundaramurthy, John McHugh, Xinming Simon Ou, S. Raj Rajagopalan, and Michael Wesch. 2014. An Anthropological Approach to Studying CSIRTs. IEEE Security Privacy 12, 5 (2014), 52–60. https://doi.org/10.1109/MSP.2014.84

[15] Sathya Chandran Sundaramurthy, Michael Wesch, Xinming Ou, John McHugh, Siva Rajagopalan, and Alexandru Bardas. 2017. Humans are dynamic. Our tools should be too. Innovations from the Anthropological Study of Security Operations Centers. IEEE Internet Computing PP (06 2017), 1–1. https://doi.org/10.1109/MIC.2017.265103212

[16] Thijs van Ede, Hojjat Aghakhani, Noah Spahn, Riccardo Bortolameotti, Marco Cova, Andrea Continella, Maarten van Steen, Andreas Peter, Christopher Kruegel, and Giovanni Vigna. 2022. DeepCASE: Semi-Supervised Contextual Analysis of Security Events. In Proceedings of the IEEE Symposium on Security and Privacy (SP). IEEE.

[17] Manfred Vielberth, Fabian Böhm, Ines Fichtinger, and Günther Pernul. 2020. Security Operations Center: A Systematic Study and Open Challenges. IEEE Access 8 (2020), 227756–227779. https://doi.org/10.1109/ACCESS.2020.3045514

[18] Chen Zhong, John Yen, Peng Liu, and Robert Erbacher. 2016. Automate Cybersecurity Data Triage by Leveraging Human Analysts' Cognitive Process. 357–363. https://doi.org/10.1109/BigDataSecurity-HPSC-IDS.2016.41

[19] Chen Zhong, John Yen, Peng Liu, and Robert Erbacher. 2018. Learning From Experts' Experience: Toward Automated Cyber Security Data Triage. IEEE Systems Journal PP (05 2018), 1–12. https://doi.org/10.1109/JSYST.2018.2828832

[20] Chen Zhong, John Yen, Peng Liu, Rob Erbacher, Renee Etoty, and Christopher Garneau. 2015. An Integrated Computer-Aided Cognitive Task Analysis Method for Tracing Cyber-Attack Analysis Processes. In Proceedings of the 2015 Symposium and Bootcamp on the Science of Security (Urbana, Illinois) (HotSoS '15). Association for Computing Machinery, New York, NY, USA, Article 9, 11 pages. https://doi.org/10.1145/2746194.2746203

[21] Chen Zhong, John Yen, Peng Liu, Rob F. Erbacher, Christopher Garneau, and Bo Chen. 2017. Studying Analysts' Data Triage Operations in Cyber Defense Situational Analysis. Springer International Publishing, Cham, 128–169. https://doi.org/10.1007/978-3-319-61152-5_6

Abbreviation	Step Name
SiS	Signature Specificity
SiA	Signature Age
CS	Customer Scope
AH	Alert History
SA	Surrounding Alerts
RL	Related Logs
TSI	Traffic Stream Information
THI	Target Host Information
THB	Target Host Behaviour
A/EI	Attack/Exploit Indicators
ASI	Attack Success Indicators
AI	Attacker Information
RUC	Relation to Use Cases

A GLOSSARY

B RESULTS SHEET FORMULATION

B.1 Relevance indicators

For the relevance indicators, we collect three multiple-choice results in total. For the **CS** we have the binary question of whether the alert is within the customer scope or not. For **SiS**, we ask whether the signature which generated the alert is generic or specific to a certain attack. For **SiA** we ask if the signature is old or new; if it updated last before 2020 we consider it old, otherwise we classify it as new.

B.2 Additional alerts

For the additional alerts stage of the model, we collect the results of the **AH** category, and the **SA**. For **AH**, we ask the experiment subject to indicate whether the alert is the first occurrence within the SOC, whether it is typically false positive, or not interesting or if the **AH** is inconclusive. Here we also ask for a free-text comment, to indicate exactly which information led the subject to this conclusion. Then for the **SA**, we ask the binary question of whether the **SA** add evidence, in addition to the original alert, that an attack is happening or has been successful. Again, we ask for a free-text comment to capture the exact reasoning of the subject.

B.3 Contextual information

For contextual information, we capture results for the **RL**, **TSI**, **THI** and **THB** steps. When reviewing the related logs, experiment subject are asked to indicate whether the **RL** show evidence of the cause of an alert, evidence of the impact/result of the alert, both, or whether they do not add any evidence. Here we also ask for a free-text comment about which log provided which type of information,

if any. For the **TSI**, the test subject estimate whether the size of the connection (i.e. number of packet/bytes) which triggered the alert, is normal for that type of connection, or whether it particularly small or large. For **THI**, we ask the question: from the information available from the customer and from the observed traffic, is the targeted host vulnerable to the potential attack? And finally for **THB**: does the targeted host exhibit unusual behaviour surrounding, but particularly, after the alert, or is there only normal behaviour. For the latter question, we also require a free-text comment about why the behaviour was normal or unusual.

B.4 Attack evidence

In order to capture the attack evidence in the results, we record findings about the **A/EI**, **AI**, **ASI** and **RUC**. Firstly, for the **A/EI** we ask the question: from the attack evidence in the alert, the logs and in public sources, is the observed traffic an attack or not. And for **AI** we ask: is the attacker a known trusted host, or an unknown host. Then, for the **ASI**, given the success indicators for the observed attack, was the attack successful, definitely unsuccessful or unknown. It should be noted, that in case in **A/EI** "no attack" was recorded in the results, that here the test subject always enters "unknown". For this question we also ask a free-text explanation to identify which (if any) success indicators were observed. Finally, for **RUC** we ask the binary question of whether the alert is covered by the use cases for the specific customer, given the evidence collected through-out the model.

B.5 Alert classification

Besides collecting information about the model stages and steps, at the end the test subjects must also classify the alerts as either *Interesting* or *Not Interesting*. Where *Interesting* is defined as: potentially a successful attack with impact for the customer. Not interesting can be defined as either a false-positive or a true-positive alert with no impact.

These rather generic classifications were chosen, since tier 1 analysts generally do not analyze and classify alerts to completion; their job description prescribes escalation to tier 2 or 3 analysts as soon as they identify something out of the ordinary.

There is also a final comment field, to indicate anything not covered by the model or results sheet, as well other issues occurring during the experiment.

C INJECTED ATTACKS

The list below identifies the attacks that were injected as part of the experiment, and the general behaviour that could be determined from the logs and alerts the attacks generated in the experiment environment.

(1) Remcos RAT: installation, command & control, lateral movement
(2) RIG Exploit Kit and Dridex: installation, command & control
(3) Emotet and Trickbot: command & control, lateral movement
(4) Qakbot and Cobalt Strike: installation, command & control, lateral movement
(5) Qakbot and Spambot: installation, command & control
(6) Hancitor and Cobalt Strike: installation, command & control, lateral movement
(7) Ghost RAT: command & control
(8) BazaarLoader and Cobalt Strike: installation, command & control, lateral movement
(9) MalSpam Brazil: installation, command & control
(10) Ursnif: installation, command & control

D MODEL CHANGES AS A RESULT OF EXPERT FEEDBACK

The initial sequence steps identified was extended with **CS** after the first round of verification, and the **RUC** was moved from the start of the sequence to its tail. **CS** was previously covered by **RUC**, but was found to be atomic and impactful enough to justify its own step. Additionally, **CS** can be determined more easily and earlier on in the process, than **RUC**.

As discovered during the verification by the experts, **RUC** requires details about the attack, the affected system and the impact, which are not available early on in the analysis process. For this reason as well, **RUC** was moved to the end of the sequence of steps. The adjustments detailed above were implemented before the experiment design and execution.

The question corresponding to the stage contextual information was changed from "2-way communication established between attacker and attacked host" to "Vulnerable host reached by potential attack", to capture cases were attacks or exploits do not result in two-way communication or no two-way communication between attacker and attacked host.

Finally, a model step "signature quality" (now omitted) was split up into "signature specificity" and "signature age", the step "target host information" was split up into "target host information" and "target host behaviour" and the step "attack/exploit information" was split into "attack/exploit information" and "attacker information". These changes were made to make the steps more atomic and easier to capture results for.