Finding Harmony in the Noise: Blending Security Alerts for Attack Detection

Tom-Martijn Roelofs
ING Bank
Amsterdam, Netherlands

Eduardo Barbaro
ING Bank & TU Delft
Amsterdam, Netherlands

Svetlana Pekarskikh
ING Bank
Amsterdam, Netherlands

Katarzyna Orzechowska
ING Bank
Katowice, Poland

Marta Kwapień
ING Bank
Katowice, Poland

Jakub Tyrlik
ING Bank
Katowice, Poland

Dinu Smadu
ING Bank
Amsterdam, Netherlands

Michel van Eeten
TU Delft
Delft, Netherlands

Yury Zhauniarovich
TU Delft
Delft, Netherlands

ABSTRACT

Large- and medium-sized organizations employ various security systems to protect their assets. These systems, often developed by different vendors, focus on different threats and usually work independently. They generate separate and voluminous alerts that have to be monitored and analyzed by often overburdened security analysts. Prior work has tried to support analysts by better correlating and prioritizing alerts. In this work, we propose to combine the wisdom of individual security systems using an Integration Layer (IL). We validated our idea by deploying the IL in a large global organization (50,000+ employees) running four very different security detection systems. We did so by using end-to-end red-team exercises to generate real attack data. For training, we labeled our dataset with evaluations directly from the incident response team instead of using the escalated decisions of the first/second tier Security Operation Center (SOC) analysts as in prior works. We showed that our approach considerably reduces the number of alerts requiring investigation while maintaining very high performance on multi-step attack detection – Matthews correlation coefficient (MCC) reaches 0.998. The substantial dependence of the model on features derived from the different security systems supports the viability of our integration methodology. The explainability layer added to our system gives analysts insights into why a particular case is marked as an attack or non-attack. Based on the test results, our approach has been added to the production setup.

CCS CONCEPTS

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

SAC '24, April 8–12, 2024, Avila, Spain

ACM ISBN 979-8-4007-0243-3/24/04.
https://doi.org/10.1145/3605098.3635981

KEYWORDS

Intrusion Detection, Security Alert Integration, Machine Learning

ACM Reference Format:
Tom-Martijn Roelofs, Eduardo Barbaro, Svetlana Pekarskikh, Katarzyna Orzechowska, Marta Kwapień, Jakub Tyrlik, Dinu Smadu, Michel van Eeten, and Yury Zhauniarovich. 2024. Finding Harmony in the Noise: Blending Security Alerts for Attack Detection. In *The 39th ACM/SIGAPP Symposium on Applied Computing (SAC '24), April 8–12, 2024, Avila, Spain.* ACM, New York, NY, USA, Article 4, 10 pages. https://doi.org/10.1145/3605098.3635981

1 INTRODUCTION

Modern enterprises rely on a diverse set of monitoring systems, such as Endpoint Detection and Response (EDR), Network Intrusion Detection Systems (NIDS) and User and Entity Behavior Analytics (UEBA), to detect cyber attacks. These systems analyze different data streams, such as network traffic, application logs, endpoint logs, and user behavior. Since these tools are developed by various vendors, they often work independently and generate separate alerts, which are ingested by a Security Operation Center (SOC), where first-tier security analysts analyze and triage them. More sophisticated or higher-priority alerts are escalated directly to second-tier experts, who make the final decision of either closing the alert or escalating it further to the Incident Response Team (IRT).

The high volume of security alerts, which can reach up to 176 per host per day according to Shen et al. [56], can lead to alert fatigue and burnouts of SOC analysts [27, 59]. This can result in attacks going unnoticed. To address this issue, researchers focused on improving automation in handling alert flows. Some approaches aim to eliminate repetitive alerts and tasks [1, 12, 37]. Others work on improving alert triage mechanisms [15, 33, 49]. However, individual alerts, even high-priority ones, are not always indicative of an attack. Therefore, a third approach has been to focus on finding better techniques for aggregating and correlating alerts [17, 31, 32, 36] to create context and link them to actual attacks [61]. In this study, we propose a novel approach within the third category.

The foundation of our method lies in combining the weak signals in the noisy event flows produced by individual independent monitoring systems in a so-called Integration Layer (IL) that deploys machine learning (ML) using Gradient Boosted Trees to detect attacks from the noisy and heterogeneous alert flow. We demonstrate

that ensembling can strengthen the weak signals in voluminous alert data, allowing the scarce attention of security analysts, especially the high-tier ones, to be focused on a much more limited number of alerts with a high probability of being an attack. We evaluate our approach in a real-world enterprise with 50, 000+ employees and find that the integration-layer model can detect all true positives, 592 attacks, with only 2 false positives. Overall, Matthews correlation coefficient (MCC), the better metric [5, 13] for a highly unbalanced dataset case, reaches 0.998 for our model. We also implemented an explainability layer using SHapley Additive exPlanation (SHAP) [55]. This layer helps SOC analysts to understand how the model arrived at a diagnostic. This is critical for analysts [2], but also for compliance purposes. Auditors require an interpretable reason why alerts are being discarded. Our proof-of-concept yielded successful results, leading the company to move the IL into the production environment. Despite the high daily volume of individual alert flows coming into the SIEM, the outcome of the IL remains typically around 2-3 alerts per day, and the system maintains the same level of performance as outlined in this study.

Two recent studies made advances in a similar direction and were also evaluated on real-world enterprise networks, like our approach. These studies take a different technical approach to ensembling. DeepCASE [61] analyzes security events as part of a sequence of relevant prior events using a recurrent neural network with an attention mechanism. The sequences can contain alerts from heterogeneous monitoring systems. If a sequence is new, analysts are asked to label it. Sequences that are sufficiently similar to those that were already labeled, do not have to be investigated again. This approach is different from ours as it does not actually attempt to detect attacks but sets out to reduce the workload of analysts. SIERRA [33] uses unsupervised anomaly detection to identify "time-slots" that likely contain a security incident. In contrast, our approach uses supervised learning, leveraging the labels from incident-response investigations.

Two further points set our approach apart from the state of the art. First, we evaluated the performance with high-quality ground truth data generated by red-team exercises, all mixed in with the regular traffic of the enterprise. Prior work evaluated detection rates using either synthetic attack traces or SOC labels as ground truth. Second, we use training labels (attack/no attack) that are based on the evaluations done by the IRT rather than by the SOC (i.e., first and second-tier analysts). Previous approaches have relied on SOC labels [23, 48] or even simulations [9, 14]. While SOC labels are a reasonable proxy, they obscure the fact that, in most cases, SOC analysts do not actually establish the final verdict on whether an alert sequence actually was an attack. Rather, the SOC analysts escalate potential attack alerts to the IRT, which is responsible for making the final call (i.e., close or respond). The high-quality verdicts by the IRT, compared to the more noisy SOC escalations, ultimately yield better training data for our models. We show that this leads to a reduced number of false positives and false negatives.

Furthermore, we demonstrate that any organization can benefit from a similar IL. In fact, our proof-of-concept is supported by standard components: XGBoost [11] for detection, Matthews correlation coefficient for evaluation, and SHAP values for explainability. These components are highly optimized, enabling the IL to process outputs from individual detection systems efficiently.

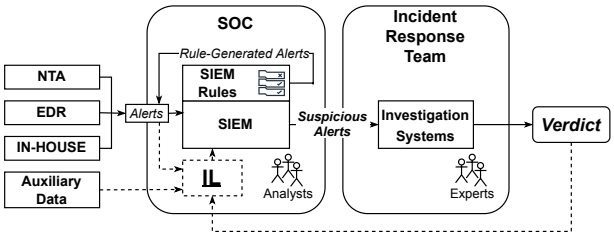


Figure 1: Incident detection and response flow with IL. The IL components are indicated with dashed lines.

Given that even large organizations do not generate an excessive volume of alerts, the IL does not demand substantial computational resources and can be deployed on a standard server. Nevertheless, it generates high-quality alerts that warrant prioritization by the company's IRT. For Small and Medium Enterprises (SMEs) relying on Managed Security Service Providers (MSSPs), this approach may reduce response times and facilitate faster threat mitigation.

2 INTEGRATION LAYER

The objective of our work is to develop a method to detect compromised endpoint devices within a large and complex organization exposed to various attack vectors, including zero-days, by consolidating weak signals present in noisy event flows generated by independent monitoring systems.We study an enterprise network with a set of endpoint devices monitored by various *monitoring systems*, such as Security Information and Event Management (SIEM), Endpoint Detection and Response (EDR), Network Traffic Analyser (NTA), and in-house solutions (sets of custom rules run over web-proxy data), alongside providers of auxiliary data. Given that organizations use the best available solutions, these systems are often developed by different vendors. When abnormal activity is detected by the rule- or anomaly-based detection engines of these systems, they generate *alerts*, also referred to as *security events* or *alarms*. A simplified typical process flow of incident detection and response in a large organization is shown in Figure 1.

The alerts from the security monitoring systems are forwarded to the SOC, where they are centrally stored and analyzed. In large SOCs, the security analyst team may be organized into multiple roles. Some analysts perform initial scans, identifying straightforward cases. Other analysts specialize in addressing more complex or escalated incidents. As a result, all alerts are divided into ones that should be ignored (i.e. *non-suspicious*) and those that require further investigation (*suspicious*). The latter is sent to *Incident Response Team* (IRT). There, experts investigate each obtained alert, draw final conclusion if it is related to an attack, and take action. Note that in the case of smaller companies, a Managed Security Service Provider (MSSP) typically acts as a SOC, forwarding the detected alerts back to the organization. There, security professionals, acting as IRT, analyze the obtained alerts and respond to them if the threat is real. Thus, the conclusion about an alert is still made on two levels: SOC analysts report if an alert is suspicious or not, while IRT experts determine the final verdict if an alert is related to a real attack.

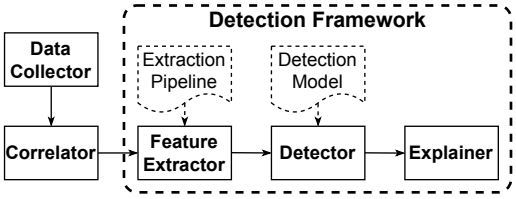


Figure 2: IL components.

This setup, however, has some issues. First, the monitoring systems produce significant numbers of false positives [2]. Second, they are independent. Therefore, events, although connected to the same incident, are not analyzed as such. We may rely on SOC analysts to infer these relations, but given the ever-increasing number of alerts, they will likely be overwhelmed and unable to draw meaningful conclusions, missing alerts or ignoring "less important" ones [27, 59]. Another possible solution is to develop rules for all possible types of attacks. However, it is not viable, given the number of possible attacks and the quadratic dependency of connections to the number of detection systems. Additionally, although a cyber attack consists of similar steps, the timing, techniques, and order are at the attacker's will. Therefore, describing all potential combinations in the first-line detection systems is nearly impossible.

In this work, we propose to add the *Integration Layer* (see dashed-outlined components in Figure 1) to the current security setup of a company. Its goal is to collect alerts and auxiliary data from individual security and monitoring systems, fuse them, and, based on this information, conclude whether a particular endpoint device is compromised. Figure 2 shows the main components of the IL.

The intuition behind our approach is the following: an attacker's malicious activity will likely generate alerts from several detection systems. For instance, an email analysis system may signal if an email contains some specific suspicious words. It may contain a link, which, being checked by a URL-checking system, may generate an alert if the website is co-hosted with other known phishing web pages. Similarly, a SIEM system may raise a security event when a user opens a link outside normal working hours. All these alerts, most probably, will be of low priority (as they also may be caused by a legitimate use) and distinct in time. Thus, they will not attract analysts' attention, and the attack may run undetected. However, if combined and enriched with auxiliary data, these alerts will reinforce the signal of a compromise and allow the IL to make a correct detection even in the case of slow, multi-step attacks [34]. While Security Orchestration Automation and Response (SOAR) systems also correlate alerts and guide security analysts towards task automation, typically, they are limited to vendor (AI) capabilities. Unlike the IL, they cannot be fully tailored to organisations' specific needs and often suffer from a lack of explainability. Further, the architecture design of the organisation may not allow the SOAR to ingest all alerts.

Data Collector. Events from different monitoring systems are collected by *Data Collector*. In our work, we employ alerts generated by a SIEM, NTA, EDR, and an in-house security system developed in the company. In our case, all these systems are developed by different vendors. However, other organizations may accommodate other setups where the same vendor backs several detection systems. Additionally, a company may run other monitoring solutions that may provide *Auxiliary Data* about the context of an attack. For instance, one can collect information about users' requests to a corporate DNS resolver or end-point average CPU load. In our case, we collect information about the users' Internet connections from the corporate web-proxy server. To each event, we add an additional attribute about its source, i.e., what detection system has generated it. This allows us to design and calculate features based on the data yielded by a particular source.

Correlator. Collected events and auxiliary data threads have different attributes, schemas and formats, so we need a component that would combine them into clusters (which we call *data bursts*) that can be bound to or represent attacks. In this work, we define an attack as *a kill-chain related to the compromise of a corporate user during a given period.* Thus, within the IL, an attack is represented by a *data burst* that consists of all security alerts produced by monitoring systems and auxiliary data threads related to an individual user (identified by a *Corporate Key (CK)*) during a particular *Day*. Note that alerts and auxiliary data threads do not always contain those values. For instance, some security systems do not capture CK attribute value. Fortunately, it is usually possible to infer the values of these attributes based on some other attributes' values given some assumptions. For instance, we can obtain the CK value based on the "IP address" or "Hostname" attribute value, given the (straightforward) assumption that during a particular day, only one user uses a single workstation.

Detection Framework. Detection Framework is responsible for detecting alerts related to attacks and explaining diagnoses. It consists of three elements: *Feature Extractor*, *Detector*, and *Explainer*. *Feature Extractor* excerpts a feature vector from each data burst following the procedures defined by *Extraction Pipeline*. *Detector* makes diagnoses using the *Detection Model* whether a feature vector and a corresponding data burst are related to an attack or not. Finally, *Explainer* explains how Detector has reached this conclusion.

3 DATASETS

To design and evaluate our method, we rely on data collected in a large organization (50,000+ employees). We deployed our collector and gathered independent alerts from four security detection systems (SIEM, NTA, EDR, and IN-HOUSE). We captured real alerts during two periods: *05.06.2021-20.07.2021* (*training/validation datasets*) and 01.08.2021-31.08.2021 (*test dataset*). Table 1 shows the number of alerts generated by these systems that are used in our training/validation (first row) and test datasets (second row). In total, the train/validation and test datasets contain 87,747 and 49,613 individual alerts, correspondingly. We aggregate the individual collected alerts by CK and date. After these operations, we obtain the training/validation and test datasets consisting of 55,615 and 30,769 samples correspondingly (see Table 2).

In real life, actual attacks are luckily very rare; therefore, to get ground-truth data, we employ a Red Team that regularly performs end-to-end exercises (grounded in the MITRE ATT&CK framework [40]) to collect more events of this nature. The primary difference from an actual attack is the lack of malicious intent from the red team members. The Red Team operates entirely independently

Table 1: Data for train/validate and test sets

Dataset	Period	Number of Individual Alerts				
		SIEM	NTA	EDR	IN-HOUSE	Total
Training	05.06.2021 20.07.2021	13,920	9,121	2,537	62,169	**87,747**
Test	01.08.2021 31.08.2021	6,796	7,693	2,522	32,602	**49,613**

Table 2: Datasets details

Dataset	Number of Samples		
	Non-attack	Attack	Total
Initial	55,615	12	55,627
After Oversampling	55,615	2,952	58,567
Training	44,492	2,360	46,852
Validation	11,123	592	11,715
Test	30,769	4	30,773

from SOC and Incident Response Teams. Its goal (and one of the criteria based on which its performance is evaluated) is to get access to a particular end-point device. Our training and validation dataset comprises 12 attack-labeled rows, while the test dataset includes 4. The attack-labeled samples in the test dataset correspond to an end-to-end Red Team exercise spanning four consecutive days on a single workstation. The Red Team activities during the first two days were stealthy and minimal, while days 3-4 exhibited moderate levels of noise. The aim was to explore the earliest point at which our models could detect these attacks.

As expected in a real environment, even with red-team activities, we have a very unbalanced dataset, where only 0.02% of the total entries are related to attack data. Such an imbalanced dataset can significantly compromise the performance of machine learning algorithms because they cannot effectively learn the decision boundary due to the low number of minority class samples [24]. To make our training dataset more leveled, we perform additional balancing. We use the Synthetic Minority Oversampling TEchnique (SMOTE) proposed by Chawla et al. [10] to create artificial attack-related samples. As a result, we obtain a dataset containing 2, 952 attack-related samples and the same 55, 615 non-attack-related entries (see the row "After Oversampling" in Table 2). After this operation, we get a more balanced training dataset, where around 5% of the entries are attack-related, and the rest belong to the non-attack class.

After the dataset is balanced using the described approach, we split it into training and validation sets in a proportion of 80% and 20%, respectively. The split is stratified[1] according to attack/non-attack and artificial/non-artificial samples (see rows 3 and 4 in Table 2). The attack/non-attack stratification guarantees that the validation set also contains our minority class (attacks). The artificial/non-artificial stratification accounts for the fact that we also want to have an equal distribution of real attacks and attacks generated by SMOTE in the training and validation sets. In that way, we ensure there is no bias toward only detecting SMOTEd attacks.

The last row in Table 2 shows the test dataset details. We utilize this independent set to test the performance of our models on totally unseen data (temporal training consistency [50]), which is also not used to tune any model hyperparameter.

[1]Stratification means that relative class frequencies are approximately preserved.

Table 3: Point distribution per quantile and source

SIEM, NTA, IN-HOUSE	EDR
1^{st} quantile: 1 point	1^{st} quantile: 2 points
2^{nd} quantile: 1.5 point	2^{nd} quantile: 3 points
3^{rd} quantile: 2 points	3^{rd} quantile: 4 points
4^{th} quantile: 2.5 points	4^{th} quantile: 5 points

4 DETECTION APPROACHES

This section explains how we build and fit together the elements that constitute our rule-based and ML-based detection approaches (see Figure 2). There are two main benefits to building a rule-based method. First, it provides explainable and straightforward results. Second, in this study, we use a dataset (see Section 3) that is *unique* (to the best of our knowledge, no similar datasets exist on which we can test our approach) and *private* (as this type of security data is highly sensitive and cannot be shared even in an anonymous form). Therefore, we need a baseline to compare to our ML-based model.

4.1 Rule-Based Approach

In this work, we experimented with six different rule-based models. We started with a very simple model that only performs alert counting. Then, we built the subsequent models, gradually increasing their complexity, taking into account alert importance, data sources, and various weighting schemes:

- R_1 Sum the total number of daily alerts for all data sources;
- R_2 Sum the number of alerts over time for all data sources;
- R_3 Sum the number of alerts over time, weighted per alert importance for every data source;
- R_4 Integrate the total number of daily alerts using quantile attributions to normalize the contribution of each source;
- R_5 Integrate the total number of daily alerts using quantile attributions to normalize the contribution of each source, giving EDR higher importance;
- R_6 Integrate the number of alerts weighted per alert importance and use quantile attributions to normalize the contribution of each source.

Based on our evaluation (see Section 5.1), the best-performing rule-based method is R_5. To build the extraction pipeline for this approach, we undertake several steps. Firstly, we utilize our training dataset to generate the distributions of alerts per CK and day for each source (EDR, SIEM, NTA, and IN-HOUSE). Next, we divide each distribution into quantiles and assign a predetermined number of points to each sample, based on the source and quantile. Based on our team experience, we assign EDR alerts more weight ($x2$) compared to alerts from other sources, given that we focus on endpoint attacks. The final point values are outlined in Table 3. Note that the simplicity of this model facilitates the analyst's ability to provide clear explanations for diagnostic results.

We highlight that the rule-based approaches provide us a ranked view of "how many points" a particular user on a given day scores. For instance, if we take the R_1 rule, which counts the number of alerts per user from all data sources, then the ranked view would show the users sorted in descending order by the number of alerts per day. Hence, the rules themselves do not determine a threshold

that, if crossed, should trigger a detection. The choice of an optimal threshold is a mathematically ill-defined problem [4], raising an *FP-FN* dilemma in our case: we want to avoid False Negatives, FNs (missing an attack) as much as possible but, at the same time, do not want to overload the analysts with False Positives, FPs (non-attacks that the models misjudge as real attacks) [2]. We define the threshold based on the maximum number of FPs the SOC team can investigate within a day while minimizing the number of FNs. Based on the interaction with the SOC team, we estimate that the analysts can handle around 5 extra FPs per day coming from the IL without disrupting the other daily activities.

4.2 Machine-Learning-Based Approach

Due to the lack of explainability in the past, cybersecurity and machine learning were not viewed as compatible until recently. However, with the advances in this area [44], nowadays, more and more security detection solutions employ a machine-learning engine. In this work, we rely on a supervised machine-learning algorithm performing two-class classification. If designed properly, supervised machine-learning algorithms outperform unsupervised ones, but they require careful design to detect previously unseen attack patterns. Further, we show how we achieved this goal.

Feature Extractor. Based on the collected data, relevant literature review and our experience, we have engineered an initial list of features. The majority of the features are trivial and mentioned in the literature: the daily number of alerts from each source; the number of important alerts[2] from each source and totals; the number of bytes-in and bytes-out; flags if specific detection system's rules are triggered (e.g., data exfiltration or beaconing [25]), etc. At the same time, several features are unique and, to the best of our knowledge, have not been mentioned in literature before. One such feature is $randomness\_uniformity\_min$, which is calculated as follows. At first, for each CK, we obtain a list of domains the corresponding user visited during the day. For each domain, we calculate the $randomness\_uniformity$ of content types – counting how many times each content type is used to request data from the domain and extracting the maximum value (percentage). Then, we take the minimum value for all domains visited by that CK. Table 4 lists examples of the designed features, their types, and a more detailed explanation of how to calculate them.

After obtaining the initial list of the features, we apply a feature reduction process. There are several reasons for doing so: first, it reduces noise, thus allowing our model to learn the right patterns in the data. Second, a smaller number of features increases the performance of ML models [3] because they need to take fewer dimensions into account. Last but not least, it increases the explainability of our ML model results (see Section 5.2), which improves usability and augments the trust in the model by security analysts.

Our feature selection process consists of two stages. During the first stage, we remove features with high mutual correlation. During the second stage, we use Principal Component Analysis (PCA), a dimensionality reduction algorithm, to guarantee we obtain the smallest set of features explaining the largest portion of the data

[2]We have created a list of important alert types for each source and count the number of alerts that fall into this category.

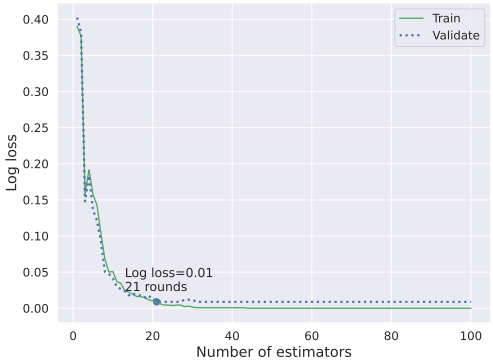


Figure 3: Logarithmic loss per number of estimators.

variance [53], more than 99% in our case. As a result, we obtained a list of 32 features that we use further (marked as "Final" in Table 4).

Detector. As a base for our ML-based model, we use a distributed, efficient, and flexible implementation of a decision-tree-based gradient boosting algorithm [21] called XGBoost [11]. We chose this implementation due to its high performance and low memory footprint [47]. We use a logistic regression objective function for binary classification, obtaining an output with boundary conditions [0,1]. As a loss function, we use logarithmic loss. XGBoost uses the logarithmic loss to learn the relation between input and output. Hence, the better the model learns the relationships, the more reliable the results. In Figure 3, we show that our logarithmic loss drops rapidly (around 20 rounds) close to zero for both train and validation sets and remains constant after that. The fact that the error for the validation set does not increase with the number of estimators suggests that our model does not overfit. To choose hyperparameters' values (learning rate, minimum loss reduction, number of trees, and maximum depth), we used a grid search using a 3-fold cross-validation method over the training data, applying a random sub-sampling of 80% and L1 regularization to avoid overfitting.

Explainer. Compared to a simple rule-based approach (e.g., R_5), which is formulaic and therefore fully explainable, the outcome of a boosted-tree algorithm (or any ML-based model) is, by definition, less explainable. XGBoost provides users with a global feature importance view that tells what features, on average, are the most important ones. However, it does not provide an explanation of how and why the model comes to every individual diagnosis that is required for a security analyst to explain a particular result. Hence, we also utilize SHAP values [55] to provide security analysts with insights on an individual sample basis. SHAP values show the positive and negative relationships between features and ML output and the most important features contributing to such output.

5 EVALUATION

Attacks per se, and especially according to our definition (see Section 2), are very rare events. However, due to the presence of SMOTEd samples, our validation set contains several attacks. That allows us to evaluate if and to what extent our models are able to detect unseen artificially generated attacks as well as real red-team

Table 4: Feature list

#	Final	Feature Name	Data Type	Description
Features from Alerts Data				
1		ck_occ	Integer, $[1, \infty)$	Total number of alert occurences from four detection systems for the given CK per given date
2	✓	ck_occ_ratio	Float, $[1.0, \infty)$	Ratio for the current CK of all alerts from the beginning of the period including the current date and all alerts from the beginning of the period excluding the current date
3	✓	edr	Integer, $[0, ck\_occ]$	Number of alerts from the EDR system per given date
4	✓	edr_important_alert	Integer, $[0, edr]$	Number of alerts triggered from a list of EDR important alerts per given date
5	✓	edr_ratio	Float, $[1.0, \infty)$	Ratio for the current CK of EDR alerts from the beginning of the period including the current date and EDR alerts from the beginning of the period excluding the current date
6		siem	Integer, $[0, ck\_occ]$	Number of alerts from the SIEM system per given date
7	✓	siem_important_alert	Integer, $[0, siem]$	Number of alerts triggered from a list of SIEM important alerts per given date
8	✓	nta	Integer, $[0, ck\_occ]$	Number of alerts from the NTA system per given date
9	✓	nta_important_alert	Integer, $[0, nta]$	Number of alerts triggered from a list of NTA important alerts per given date
10	✓	nta_ratio	Float, $[1.0, \infty)$	Ratio for the current CK of NTA alerts from the beginning of the period including the current date and NTA alerts from the beginning of the period excluding the current date
11	✓	nta_total	Integer, $[0, n^1]$	Sum of NTA alerts from the beginning of the period until a given date for a given CK
12	✓	in_house	Integer, $[0, ck\_occ]$	Number of alerts from the IN-HOUSE system per given date
13	✓	in_house_important_alert	Integer, $[0, in\_house]$	Number of alerts triggered from a list of IN-HOUSE important alerts per given date
14	✓	in_house_ratio	Float, $[1, \infty)$	Ratio for the current CK of IN-HOUSE alerts from the beginning of the period including the current date and IN-HOUSE alerts from the beginning of the period excluding the current date
15	✓	in_house_total	Integer, $[0, n^1]$	Sum of IN-HOUSE alerts from the beginning of the period until a given date for a given CK
16	✓	deprecated_useragent_seen	Binary, $\{0, 1\}$	Did "Deprecated User-Agent" IN-HOUSE model trigger
17	✓	data_exfiltration_seen	Binary, $\{0, 1\}$	Did "Data Exfiltration" IN-HOUSE model trigger
18	✓	num_diff_src	Integer, $[1, 4]$	Distinct number of detection systems that triggered at least once per given date
Auxiliary Data Features				
19	✓	bytes_out_sum	Integer, $[0, \infty)$	Sum of bytes-out in Proxy data per CK per date
20		bytes_in_sum_agg	Integer, $[0, \infty)$	Sum of bytes-in in Proxy data
21	✓	bytes_out_sum_avg	Float, $[0.0, \infty)$	Average of bytes-out in Proxy data per CK per date
22	✓	bytes_out_sum_non_categorized	Integer, $[0.0, \infty)$	Sum of bytes-out to domains not categorized by Proxy per CK per date
23	✓	bytes_out_sum_q25	Integer, $[0, \infty)$	First quartile of $bytes\_out\_sum$
24	✓	bytes_out_sum_q75	Integer, $[0, \infty)$	Third quartile of $bytes\_out\_sum$
25	✓	bytes_out_sum_std	Float, $[0.0, \infty)$	Standard deviation of $bytes\_out\_sum$
26	✓	bytes_out_sum_var	Float, $[0.0, \infty)$	Variance of bytes_out_sum
27	✓	get_bytes_ratio	Float, $[0.0, 1.0]$	Ratio of $bytes\_in\_sum/(bytes\_in\_sum + bytes\_out\_sum)$ of connections with "GET" requests
28	✓	no_of_rare_uas	Integer, $[0, \infty)$	Number of rare user agents observed in Proxy traffic
29	✓	url_substring_max_cnt	Integer, $[0, s^2]$	Max size of a sub-string in a URL
30		domains_dst_cnt	Integer, $[0, \infty)$	Count of distinct domains CK connected to that day
31		uncategorized_domains_cnt	Integer, $[0, \infty)$	Number of uncategorized domains user connected to that day
32	✓	uncategorized_domains_percent	Float, $[0.0, 100.0]$	Ratio $uncategorized\_domains\_cnt/domains\_dst\_cnt$ in percent
33	✓	suspicious_tld_dst_cnt	Integer, $[0, \infty)$	Number of distinct domains with suspicious Top Level Domain that a user connected to that day
34	✓	suspicious_tld_percent	Float, $[0.0, 100.0]$	Ratio $suspicious\_tld\_dst\_cnt/domains\_dst\_cnt$ in percent
35	✓	ssl_inspected_dst_cnt	Integer, $[0, \infty)$	Number of distinct SSL inspected domains user was connected to
36	✓	randomness_uniformity_min	Float, $[0.0, 100.0]$	$randomness\_uniformity$ is calculated by counting distinct content type connections per domain and getting maximum value in percent. For instance, if a user connects to domain1.com 10 times, each time requesting data with content type 'text/plain', then the $randomness\_uniformity$ is 100. If the user connects to domain2.com 10 times, 8 times requesting data with the 'text/plain' content type and 2 times with 'text/javascript', then $randomness\_uniformity$ is 80. Accordingly, $randomness\_uniformity\_min$ for this user is 80.
37	✓	randomness_uniformity_q95	Float, $[0.0, 100.0]$	Count distinct content types per domain and get the maximum value in percent. Get the 95th quantile of the obtained values.

1. n is a sum of ck_occ up to the current date for a given CK
2. https://stackoverflow.com/a/417184/1108213

attacks. In addition, we evaluate our system on the test set that only contains real data (i.e., no SMOTEd samples).

Unfortunately, the commonly used metrics like precision, recall, accuracy, or F1-measure can be either overly optimistic about the results or very difficult to directly compare [5, 13]. Therefore, to compare the performance of our models, we have chosen the metrics that account for the fact that our dataset is very unbalanced: *Matthews correlation coefficient* (MCC) [38] and the *Area Under the Precision-Recall Curve* (PR-AUC) metrics. MCC is a re-definition of Pearson's coefficient that measures the quality of a binary classifier with the advantage of accounting for the weight of all classes. The PR-AUC metric is used instead of the more common area under the receiver operating characteristic (ROC-AUC) curve because the latter is overly optimistic for unbalanced datasets, and even a small number of (in)correct results can result in a significant change in ROC-AUC [54]. Additionally, we use *McNemar's test* [39], a paired and non-parametric nominal test, to claim that one model is different than another with at least 95% confidence. Finally, we also report the commonly used metrics for reference.

5.1 Rule-based Approach

Before proceeding with the evaluation of rule-based methods, it is necessary to define the threshold values that align with the company's business requirement. To determine the appropriate threshold value for each rule-based model, we evaluated the performance of each model on our training dataset, minimizing the number

Table 5: Metrics for all 6 different rule-based models.

Metrics	Rule-based Models					
	R1	R2	R3	R4	R5	R6
Precision (P)	0.596	0.909	0.667	1.000	**0.981**	0.995
Recall (R)	0.095	0.221	0.129	0.160	**0.438**	0.312
Accuracy (Acc)	0.951	0.960	0.953	0.958	**0.971**	0.965
MCC	0.224	0.437	0.280	0.391	**0.645**	0.547

Table 6: Confusion matrix and metrics values for the best rule-based and ML-based models

Model	Number of				P	R	Acc	MCC
	TP	FP	TN	FN				
R_5	259	5	11,118	333	0.981	0.438	0.971	0.645
ML-based (IRT)	592	2	11,121	0	0.995	1.0	1.0	0.998
ML-based (SOC)	597	31	11,090	7	0.933	0.995	0.996	0.968

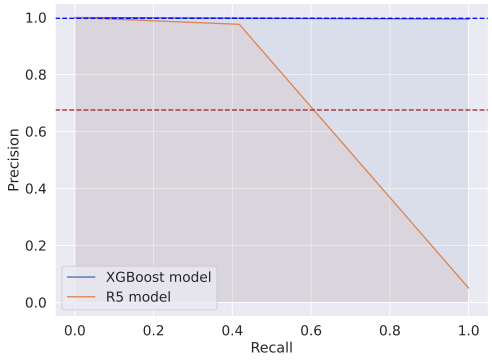


Figure 4: Precision-recall curve for ML-based XGBoost (blue) and rule-based R_5 (orange) models. The horizontal lines represent their respective average precision (or PR-AUC).

of false negatives while ensuring that the number of false positives remains limited to a maximum of 5 per day according to the company's business requirement. After finding the threshold, we evaluated each model on the validation dataset. The results of this evaluation are reported in Table 5. As we can see, the rule-based method R_5 shows the highest MCC (precision/recall and accuracy as well). In addition, McNemar's test shows that with at least 95% confidence, R_5's results are statistically different than R_1, R_2, R_3, and R_4 and comparable to R_6. Given the highest MCC, R_5 is considered forward as our baseline rule-based model.

The first row in Table 6 shows the number of TP, FP, TN, and FN for the best rule-based model (R_5). As can be seen, it performs moderately, correctly capturing 259 out of 592 attacks. It misses 333 attacks and wrongly classifies 5 samples as attacks. The remainder of the data, 11, 118 entries, are correctly classified as non-attacks.

The metrics for R_5 were determined using a threshold value of 8.5. However, reducing this threshold to 8.0 caused a significant spike in the number of false positives, reaching hundreds. This sudden rise in false positives highlights the delicate balance between differentiating attack and non-attack classes and the challenges in accurately determining legitimate behavior versus an attack. This observation suggests that "anomaly detection" strategies may not be effective in this domain, as they primarily aim to identify outliers that are significantly separated from "normal traffic". The sharp increase in false positives also indicates that the model is too simple to understand the nuances of less obvious entries.

Note that creating rule-based models requires expert knowledge of how detection systems results correlate and what type of alerts should be prioritized. For instance, based on our experience and the interaction with the SOC team, we know that the EDR system deployed in the organization produces stronger evidence of compromise. Hence, we assign higher weights to them. Obviously, given the complex nature of the cyber detection space, an expert cannot grasp all subtle connections between different types of alerts and their influence on the attacks. That and the moderate results achieved by rule-based approaches further suggest that ML-based models will perform much better in this task. Still, as we already mentioned, the rule-based model gives a baseline that can be compared against.

5.2 ML-based Approach

Our ML-based model correctly captures 100% of the 592 attacks (see Table 6). Of the 594 (592 + 2) alerts classified as attacks, solely 2 are incorrectly classified (False Positives). The remaining 11, 121 entries are correctly classified as non-attacks. Both precision and recall metrics converge to 1, which makes them impractical to be used for comparison. As noted in Section 5, we use the MCC metric instead, which is equal for our ML-based model to 0.998.

One of the novelties of our approach is to utilize labels obtained directly by the IRT instead of SOC escalations. To quantify that benefit, we ran the same case, but instead of labeling the data using the outcome of the IRT verdicts, we utilized SOC escalations (as positives). We highlight that the IRT labels are a subset of the SOC escalations, i.e., the IRT receives all escalations from SOC and decides if they need to be actioned (positives) or closed (negatives). The results show a significant increase in the number of false positives (from 2 for IRT to 31 for SOC, see Table 6). The small difference in the total number of cases for IRT and SOC, 11, 715 and 11, 725, respectively, is explained by SMOTE and undersampling being done on sets with a different number of cases labeled as attacks to start with (12 for IRT — see Table 2, and 71 for SOC).

Figure 4 shows the PR-curve for both R_5 and ML models, highlighting the better performance obtained by the ML-based model. Indeed, PR-AUC, which indicates to what extent a model is better than a random guess (for which PR-AUC is 0.5), is equal to 0.68 and close to 1 for the R_5 and ML-based models, respectively.

Contrary to the rule-based method, where we rely on the security team's experience to define the importance of individual features, the ML-based model can detect the feature importance using the labeled training data. We can use this aspect to understand what features are the most important for the model to make its decision. The top features impacting the ML model's verdict are: the number of rare user agents (10%), ratio bytes in/out for connections with "GET" responses (9%), number of EDR alerts (8%), number of important EDR alerts (6%), randomness uniformity (6%), number of different sources (2%), number of in-house system alerts (1%). These features alone contribute significantly to the model's performance.

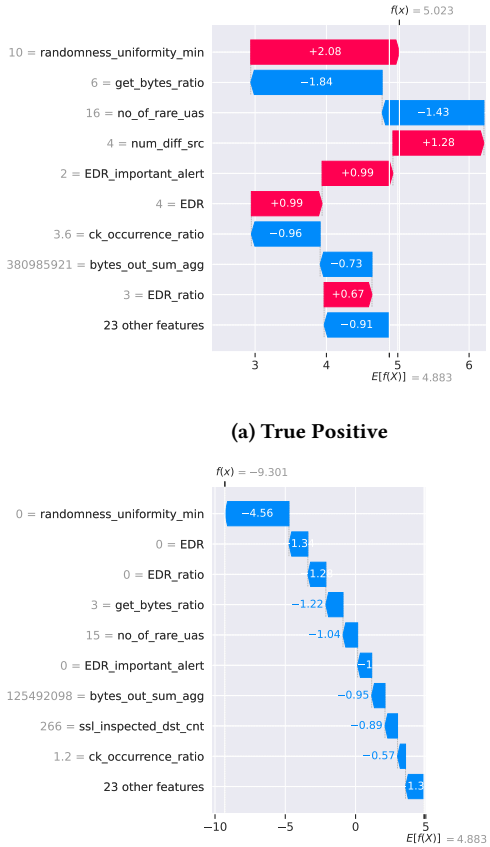
Note that, however, it does not mean removing them would yield to a total loss of performance.

There are several conclusions that we can make from this list of important features. First, it indicates that proxy and EDR alerts are significantly more important than the ones coming from other sources (e.g., NTA and SIEM). That makes sense, as in this work, we aim at detecting endpoint attacks. Second, it shows that combining multiple data sources allows for better model performance. This multi-source dependence validates the need for getting the data from diverse security systems that monitor different aspects of an entity's operation (network, proxy, applications, behavior, etc.), thus confirming the viability of the IL idea.

We also evaluated our ML-based approach on the test dataset. Out of 4 attacks (the Red Team performed the attack during 4 consecutive days on the same machine), the IL managed to detect the attack on the third and fourth day (thus, 2 attacks are detected). We judge this to be a good result because, during days 1 and 2, the Red Team was performing just reconnaissance and, therefore, was very quiet, only triggering 3 and 2 independent alerts, respectively.

For each observation, the IL provides an explanation of its verdict delivered in the form of SHAP values waterfall representation. Figure 5 shows these representations for two samples (one TP and one TN) in the test set. These SHAP values waterfall representations are presented to IRT and SOC experts to provide insight into how the IL arrives at its conclusions.

The use of SHAP values as a method for explanation has proven to be effective during the testing phase, receiving positive feedback from IRT and SOC experts. As a result, it is currently utilized as the primary mechanism for explaining diagnoses made by the ML-based approach. The key advantage of this method lies in its ability to provide local explanations for each individual conclusion, as opposed to global explanations of feature importance offered by boosting-tree implementations.



(a) True Positive

(b) True Negative

Figure 5: SHAP values for test set samples. Red (blue) bars represent features positively (negatively) correlated with the target variable.

6 DISCUSSION AND LIMITATIONS

We show the advantages of the IL by implementing it to detect attacks on endpoints. We do not recommend developing a single "master" model - encompassing every type of attack. Instead, according to our research, it is better to build distinct detection models for every kind of attack, e.g., for attacks on endpoints, servers, or applications. The reason is that features covering the systems mentioned above vary significantly, meaning important features for endpoint-type detection likely will not be important for an attack on a server or application. Thus, a natural extension of this work is to cover other types of attacks, for example, on servers or particular applications. We highlight that while the features used for detection on servers or applications may differ, the concept of integrating data from multiple sources will remain. Academic literature [57] also suggests tailoring detection models to particular types of attacks.

Utilizing, combining, and correlating different features in an IL is a generic concept and can be applied to any attack. That said, the results we report here remain specific to our framework. We acknowledge that other organizations may have different individual systems, and that may yield different results.

Currently, we aggregate events daily, i.e. within a day, an attacker should produce enough malicious activity to trigger the IL. While this is a realistic scenario (average penetration time is about two days [51]), some "low-slow" attacks are designed to last multiple weeks while only a few actions are executed at a time. For instance, our test dataset includes an attack and during the first two days, the Red Team performed reconnaissance and executed penetration only during the third day. Unsurprisingly, our model detects the attack on the third day. Therefore, another extension of this work is to focus on modeling these "low-slow" attacks.

There are limitations related to our datasets. Due to privacy and the sensitive nature of the data, we cannot share our dataset. Therefore, it is not trivial to compare our approach to others. At the same time, this is a typical limitation of many works in this area [31]. So far, no work has been able to set up a true comparative evaluation on the same enterprise data. However, we can highlight two points supporting our conclusions. First, our organization already employs several state-of-the-art detection systems from leading industry vendors. Despite the presence of those systems, our IL proved its effectiveness and is now used as the main system to generate alerts. Second, in this paper, we did our best to report all the details about our system. Therefore, other companies can replicate and verify

the approach. Note that running a similar system should not take a lot of resources, thus also useful for smaller companies.

The other limitation is the low number of attack samples in the test data. On the one hand, this result shows that the systems and procedures developed and employed in the organization have a positive impact on its security. That said, it is counterintuitive to wish for more positive samples because they correspond to successful attacks. On the other, it does not allow us to determine the true performance of our approach. Thus, it is impossible to evaluate its real efficiency given only four positive attack samples, two of which represent "low-slow" cases. Since the IL has been in production, we can report similar results to the ones presented in this work.

7 RELATED WORK

The modern cyber-attack detection landscape is diverse, with a wide range of solutions. Signature-based systems, such as SNORT and Bro, use known attack patterns to detect intrusions. Case-based systems, as proposed in [19], rely on past experiences to detect new attacks. Knowledge-based systems, such as those based on the MITRE's ATT&CK framework, use knowledge of attacker's tactics, techniques, and procedures to detect intrusions. Detection methods can also include rule-based [41], statistical [6], and data mining or AI-based [7, 16, 18, 29, 30, 33, 42, 43, 46, 56, 61, 62] approaches. Despite the wealth of options available, there is no widely accepted approach, set of tools, or underlying strategy for monitoring, detecting, and responding to cyber-attacks [8, 22, 52, 58, 60].

The use of statistical or rule-based approaches for attack detection is attractive because these models are easy to understand, require minimal computational resources, and are relatively simple to implement. However, they may not be able to detect more complex attacks and produce too many alerts. Additionally, setting the threshold for these models can be an arbitrary or knowledge-based process, introducing bias and uncertainty [4, 6], as we also show here. The use of AI in cyber-attack detection and prevention is gaining traction recently [18, 29, 33, 43, 56, 61, 62]. These approaches tend to be more accurate and precise, detecting even the most complex activities. However, they come at the cost of increased computational complexity and longer processing times. Additionally, the process of data collection and model deployment is not straightforward, and explaining a machine-learning outcome is less transparent than in the case of rule-based methods [44].

The capability to detect both known and unknown attacks is crucial for a modern cyber-attack detection system. However, achieving this level of generalization can be challenging as it often comes at the cost of negatively impacting the system's ability to detect known attacks. For example, systems like DeepCASE [61] shine at identifying attacks that are similar to previous ones but require manual intervention for new, unknown attacks. On the other hand, SIERRA [33] excels at detecting anomalies but also prioritizes a large number of non-attack-related alerts for investigation.

One significant challenge in detecting actual attacks among a large number of alerts is their needle-in-a-haystack nature. This means that only a small percentage of total alerts correspond to real attacks, which poses a problem when models rely on knowing the outcome of an alert (i.e. attack or not) to learn and improve. This problem is not unique to cybersecurity [20, 24] but is crucial because false negatives have a disproportionately high cost. Unfortunately, only a few intrusion detection works mention that and try to balance datasets and use appropriate metrics. For instance, Jia et al. [28] also use SMOTE to increase the number of unrepresented class samples and random downsampling to decrease the number of overrepresented class samples. Interestingly, in our work, we also experimented with the random downsampling technique; however, our results did not show any increase in the performance of our models. Li et al. [35] combined SMOTE with a Random Forest algorithm to automatically oversample minority classes' samples.

Many attempts have been made to reduce the number of alerts that a security analyst needs to review, including alert verification, clustering, and re-prioritization. We refer the interested reader to the survey works [26, 31, 34, 36] that summarized different approaches and the corresponding seminal works in this area. According to the developed classification, our approach exploits a similarity-based correlation technique to combine alerts. However, in our work, we focus on multi-stage attacks. Their specifics and other seminal works are surveyed by Navarro et al. [45].

8 CONCLUSION

The overwhelming amount of security alerts generated by detection systems is a significant challenge faced by security teams in large organizations. Alahmadi et al. [2] report that a staggering 99% of these alerts are FPs, making it extremely difficult for security analysts to efficiently assess the true security risks. To address this issue, this work introduces the Integration Layer that combines weak signals from various independent detection systems to increase the accuracy of attack detection and reduce the number of alerts security analysts need to evaluate. The inclusion of an explainability component aided security analysts in understanding the IL model's decisions and provided evidence for audits.

The evaluation of the IL on the real-life data collected from a large organization with 50, 000+ employees shows the viability of our approach: the MCC reaches 0.998 for the ML-based IL. Not surprisingly, the IL, since its implementation, has been fully integrated into the organization's security system, and, to date, results have been consistent with those reported in this study.

REFERENCES

[1] K.H. Al-Saedi, S. Ramadass, A. Almomani, S. Manickam, and W. Alsalihy. 2012. Collection mechanism and reduction of IDS alert. *International Journal of Computer Applications* 975 (2012), 8887.
[2] B. Alahmadi, L. Axon, and I. Martinovic. 2022. 99% False Positives: A Qualitative Study of SOC Analysts' Perspectives on Security Alarms. In *USENIX Security*. 2783–2800.
[3] E. Alpaydin. 2014. *Introduction to Machine Learning* (3 ed.).
[4] E. Barbaro, E.M. Grua, I. Malavolta, M. Stercevic, E. Weusthof, and J. van den Hoven. 2020. Modelling and predicting User Engagement in mobile applications. *Data Sci.* 3 (2020), 61–77.
[5] S. Boughorbel, F. Jarray, and M. El-Anbari. 2017. Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric. *PLOS ONE* 12, 6 (06 2017), 1–17.
[6] B. Bouyeddou, F. Harrou, B. Kadri, and Y. Sun. 2021. Detecting network cyber-attacks using an integrated statistical approach. *Cluster Computing* 24 (2021), 1435–1453.
[7] A.L. Buczak and E. Guven. 2016. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys & Tutorials* 18, 2 (2016), 1153–1176.
[8] S.E. Chandy, A. Rasekh, Z.A. Barker, and M. Ehsan Shafiee. 2019. Cyberattack Detection Using Deep Generative Models with Variational Inference. *Journal of*

Water Resources Planning and Management 145, 2 (2019), 04018093.
[9] J. Chang, J. Yu, and Y. Pei. 2010. MS²IFS: A Multiple Source-Based Security Information Fusion System. In *International Conference on Communications and Intelligence Information Security*. 215–219.
[10] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *J. Artif. Intell. Res. (JAIR)* 16 (06 2002), 321–357.
[11] T. Chen and C. Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 785–794.
[12] D. Chiba, M. Akiyama, Y. Otsuki, H. Hada, T. Yagi, T. Fiebig, and M. Van Eeten. 2022. DomainPrio: Prioritizing Domain Name Investigations to Improve SOC Efficiency. *IEEE Access* 10 (2022), 34352–34368.
[13] D. Chicco and G. Jurman. 2020. The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation. *BMC Genomics* 21 (01 2020).
[14] T. Chyssler, S. Nadjm-Tehrani, S. Burschka, and K. Burbeck. 2004. Alarm reduction and correlation in defence of IP networks. In *IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. 229–234.
[15] M. Cinque, R. Della Corte, and A. Pecchia. 2020. Contextual filtering and prioritization of computer application logs for security situational awareness. *Future Generation Computer Systems* 111 (2020), 668–680.
[16] M. Cui, J. Wang, and B. Chen. 2020. Flexible Machine Learning-Based Cyberattack Detection Using Spatiotemporal Patterns for Distribution Systems. *IEEE Transactions on Smart Grid* 11 (2020), 1805–1808.
[17] F. Cuppens and A. Miege. 2002. Alert correlation in a cooperative intrusion detection framework. In *IEEE Symposium on Security and Privacy*. 202–215.
[18] A. Delplace, S. Hermoso, and K. Anandita. 2020. Cyber Attack Detection thanks to Machine Learning Algorithms. https://arxiv.org/abs/2001.06309
[19] M. Esmaili, R. Safavi-Naini, and J. Pieprzyk. 1996. Evidential reasoning in network intrusion detection systems. In *Information Security and Privacy*. 253–265.
[20] A. Fernández, S. García, F. Herrera, and N.V. Chawla. 2018. SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-Year Anniversary. *J. Artif. Int. Res.* 61, 1 (jan 2018), 863–905.
[21] J.H. Friedman. 2001. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics* 29, 5 (2001), 1189–1232.
[22] G. González-Granadillo, S. González-Zarzosa, and R. Diaz. 2021. Security Information and Event Management (SIEM): Analysis, Trends, and Usage in Critical Infrastructures. *Sensors* 21, 14 (2021).
[23] J.L. Guerra, C. Catania, and E. Veas. 2022. Datasets Are Not Enough: Challenges in Labeling Network Traffic. *Comput. Secur.* 120 (sep 2022).
[24] H. He and E.A. Garcia. 2009. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering* 21, 9 (2009), 1263–1284.
[25] X. Hu, J. Jang, M.P. Stoecklin, T. Wang, D.L. Schales, D. Kirat, and J.R. Rao. 2016. BAYWATCH: Robust Beaconing Detection to Identify Infected Hosts in Large-Scale Enterprise Networks. In *IEEE/IFIP International Conference on Dependable Systems and Networks*. 479–490.
[26] N. Hubballi and V. Suryanarayanan. 2014. False alarm minimization techniques in signature-based intrusion detection systems: A survey. *Computer Communications* 49 (2014), 1–17.
[27] O. Hughes. 2022. Cybersecurity burnout is real. And it's going to be a problem for all of us. Retrieved 12.07.2022 from https://www.zdnet.com/article/cybersecurity-burnout-is-real-and-its-going-to-be-a-problem-for-all-of-us/
[28] B. Jia, Y. Tian, D. Zhao, X. Wang, C. Li, W. Niu, E. Tong, and J. Liu. 2021. Bidirectional RNN-Based Few-Shot Training for Detecting Multi-stage Attack. In *Information Security and Cryptology*. 37–52.
[29] D. Jin, Y. Lu, J. Qin, Z. Cheng, and Z. Mao. 2020. SwiftIDS: Real-time intrusion detection system based on LightGBM and parallel intrusion detection mechanism. *Computers & Security* 97 (2020), 101984.
[30] H. Karimipour, A. Dehghantanha, R.M. Parizi, K.R. Choo, and H. Leung. 2019. A Deep and Scalable Unsupervised Machine Learning System for Cyber-Attack Detection in Large-Scale Smart Grids. *IEEE Access* 7 (2019), 80778–80788.
[31] I. Kotenko, D. Gaifulina, and I. Zelichenok. 2022. Systematic Literature Review of Security Event Correlation Methods. *IEEE Access* 10 (2022), 43387–43420.
[32] M. Landauer, F. Skopik, M. Wurzenberger, and A. Rauber. 2022. Dealing with Security Alert Flooding: Using Machine Learning for Domain-Independent Alert Aggregation. *ACM Trans. Priv. Secur.* 25, 3, Article 18 (apr 2022).
[33] J. Lee, F. Tang, P. Thet, D. Yeoh, M. Rybczynski, and D. Divakaran. 2022. SIERRA: Ranking Anomalous Activities in Enterprise Networks. In *IEEE European Symposium on Security and Privacy*. 44–59.
[34] D. Levshun and I. Kotenko. 2023. A survey on artificial intelligence techniques for security event correlation: models, challenges, and opportunities. *Artificial Intelligence Review* (2023), 1–44.
[35] S. Li, Q. Zhang, X. Wu, W. Han, Z. Tian, and S. Yu. 2021. Attribution Classification Method of APT Malware in IoT Using Machine Learning Techniques. *Sec. and Commun. Netw.* (jan 2021), 12 pages.
[36] T. Limmer and F Dressler. 2008. *Survey of Event Correlation Techniques for Attack Detection in Early Warning Systems.* Technical Report. TU Dresden.
[37] X. Lu, X. Du, and W. Wang. 2018. Network IDS Duplicate Alarm Reduction Using Improved SNM Algorithm. In *IEEE International Conference on Image, Vision and Computing*. 767–774.
[38] B.W. Matthews. 1975. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA) - Protein Structure* 405, 2 (1975), 442–451.
[39] Q. Mcnemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* 12 (1947), 153–157.
[40] Mitre. [n. d.]. The MITRE's ATT&CK framework. Retrieved 04.07.2022 from https://attack.mitre.org/
[41] A. Mounji. 1997. *Rule-based distributed intrusion detection.* Ph. D. Dissertation. Institute d'Informatique, University of Namur, Belgium.
[42] N.R. Moşteanu. 2020. Artificial intelligence and cyber security - face to face with cyber-attack - A Maltese case of risk management. *ECOFORUM* 9 (2020). Issue 2.
[43] M. Musser and A. Garriott. 2021. *Machine Learning and Cybersecurity: Hype and Reality.* Technical Report. Center for Security and Emerging Technology. https://cset.georgetown.edu/publication/machine-learning-and-cybersecurity/
[44] A. Nadeem, D. Vos, C. Cao, L. Pajola, S. Dieck, R. Baumgartner, and S. Verwer. 2023. SoK: Explainable Machine Learning for Computer Security Applications. In *IEEE European Symposium on Security and Privacy*. 221–240.
[45] J. Navarro, A. Deruyver, and P. Parrend. 2018. A systematic survey on multi-step attack detection. *Computers & Security* 76 (2018), 214–249.
[46] K.K. Nguyen, D.T. Hoang, D. Niyato, P. Wang, D. Nguyen, and E. Dutkiewicz. 2018. Cyberattack detection in mobile cloud computing: A deep learning approach. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*. 1–6.
[47] D. Nielsen. 2016. *Tree Boosting With XGBoost. Why Does XGBoost Win "Every" Machine Learning Competition?* Master's thesis. NTNU.
[48] T. Nishiyama, A. Kumagai, K. Kamiya, and K. Takahashi. 2020. SILU: Strategy Involving Large-scale Unlabeled Logs for Improving Malware Detector. In *IEEE Symposium on Computers and Communications*. 1–7.
[49] A. Oprea, Z. Li, R. Norris, and K. Bowers. 2018. MADE: Security Analytics for Enterprise Threat Detection. In *Annual Computer Security Applications Conference*. 124–136.
[50] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro. 2019. TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time. In *USENIX Security*. 729–746.
[51] Positive Technologies. 2021. Cybercriminals Can Penetrate 93% of Local Company Networks, and Trigger 71% of Events Deemed 'Unacceptable' For Their Businesses. https://www.ptsecurity.com/ww-en/about/news/positive-technologies-cybercriminals-can-penetrate-93-of-local-company-networks-and-trigger-71-of-events-deemed-unacceptable-for-their-businesses/
[52] J. Raiyn. 2014. A survey of Cyber Attack Detection Strategies. *International Journal of Security and Its Applications* 8 (01 2014), 247–256.
[53] G.T. Reddy, M.P.K. Reddy, K. Lakshmanna, R. Kaluri, D.S. Rajput, G. Srivastava, and T. Baker. 2020. Analysis of Dimensionality Reduction Techniques on Big Data. *IEEE Access* 8 (2020), 54776–54788.
[54] D. Rosenberg. 2022. Unbalanced Data? Stop Using ROC-AUC and Use AUPRC Instead. Retrieved 05.12.2022 from https://towardsdatascience.com/imbalanced-data-stop-using-roc-auc-and-use-auprc-instead-46af4910a494
[55] L.S. Shapley. 1951. *Notes on the N-Person Game - I: Characteristic-Point Solutions of the Four-Person Game.* RAND Corporation.
[56] Y. Shen, E. Mariconti, P.A. Vervier, and G. Stringhini. 2018. Tiresias: Predicting Security Events Through Deep Learning. In *ACM SIGSAC Conference on Computer and Communications Security*. 592–605.
[57] R. Sommer and V. Paxson. 2010. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. In *IEEE Symposium on Security and Privacy*. 305–316.
[58] M.R. Stytz, D.E. Lichtblau, and S.B. Banks. 2005. *Toward Using Intelligent Agents to Detect, Assess, and Counter Cyberattacks in a Network-Centric Environment.* Technical Report. Institute for Defense Analyses.
[59] S.C. Sundaramurthy, A.G. Bardas, J. Case, X. Ou, M. Wesch, J. McHugh, and S.R. Rajagopalan. 2015. A Human Capital Model for Mitigating Security Analyst Burnout. In *Symposium On Usable Privacy and Security*. 347–359.
[60] F. Ullah and M. Ali Babar. 2019. Architectural Tactics for Big Data Cybersecurity Analytics Systems: A Review. *Journal of Systems and Software* 151 (2019), 81–118.
[61] T. van Ede, H. Aghakhani, N. Spahn, R. Bortolameotti, M. Cova, A. Continella, M. van Steen, A. Peter, C. Kruegel, and G. Vigna. 2022. DEEPCASE: Semi-Supervised Contextual Analysis of Security Events. In *IEEE Symposium on Security and Privacy*. 522–539.
[62] R. Vinayakumar, Mamoun Alazab, K. P. Soman, Prabaharan Poornachandran, Ameer Al-Nemrat, and Sitalakshmi Venkatraman. 2019. Deep Learning Approach for Intelligent Intrusion Detection System. *IEEE Access* 7 (2019), 41525–41550.