Predictive Cyber Situational Awareness and Personalized Blacklisting: A Sequential Rule Mining Approach

MARTIN HUSÁK, Masaryk University, Czech Republic
TOMÁŠ BAJTOŠ, Pavol Jozef Šafárik University in Košice, Slovakia
JAROSLAV KAŠPAR, Masaryk University, Czech Republic
ELIAS BOU-HARB, University of Texas at San Antonio
PAVEL ČELEDA, Masaryk University, Czech Republic

Cybersecurity adopts data mining for its ability to extract concealed and indistinct patterns in the data, such as for the needs of alert correlation. Inferring common attack patterns and rules from the alerts helps in understanding the threat landscape for the defenders and allows for the realization of cyber situational awareness, including the projection of ongoing attacks. In this article, we explore the use of data mining, namely sequential rule mining, in the analysis of intrusion detection alerts. We employed a dataset of 12 million alerts from 34 intrusion detection systems in 3 organizations gathered in an alert sharing platform, and processed it using our analytical framework. We execute the mining of sequential rules that we use to predict security events, which we utilize to create a predictive blacklist. Thus, the recipients of the data from the sharing platform will receive only a small number of alerts of events that are likely to occur instead of a large number of alerts of past events. The predictive blacklist has the size of only 3% of the raw data, and more than 60% of its entries are shown to be successful in performing accurate predictions in operational, real-world settings.

CCS Concepts: • **Information systems** → *Data mining*; • **Security and privacy** → *Intrusion detection systems*; *Network security*;

Additional Key Words and Phrases: Data mining, situational awareness, intrusion detection, attack prediction

ACM Reference format:
Martin Husák, Tomáš Bajtoš, Jaroslav Kašpar, Elias Bou-Harb, and Pavel Čeleda. 2020. Predictive Cyber Situational Awareness and Personalized Blacklisting: A Sequential Rule Mining Approach. *ACM Trans. Manage. Inf. Syst.* 11, 4, Article 19 (September 2020), 16 pages.
https://doi.org/10.1145/3386250

This research was supported by ERDF "CyberSecurity, CyberCrime and Critical Information Infrastructures Center of Excellence" (No.CZ.02.1.01/0.0/0.0/16_019/0000822). The project is co-financed by the Governments of Czechia, Hungary, Poland and Slovakia through Visegrad Grants from International Visegrad Fund. The mission of the fund is to advance ideas for sustainable regional cooperation in Central Europe. Further, the project was also partially funded by the U.S. National Science Foundation (NSF) Office of Advanced Cyberinfrastructure (OAC) #1907821.
Authors' addresses: M. Husák, J. Kašpar, and P. Čeleda, Institute of Computer Science, Masaryk University, Brno, Czech Republic; emails: husakm@ics.muni.cz, kaspar@ics.muni.cz, celeda@ics.muni.cz; T. Bajtoš, Institute of Computer Science, Pavol Jozef šafárik University in Košice, Košice, Slovakia; email: tomas.bajtos@student.upjs.sk; E. Bou-Harb, The Cyber Center for Security and Analytics, University of Texas at San Antonio, San Antonio; email: elias.bouharb@utsa.edu.

2158-656X/2020/09-ART19 $15.00
https://doi.org/10.1145/3386250

1 INTRODUCTION

In recent years, the cybersecurity field started to adopt data mining for its ability to extract concealed and indistinct patterns in the data for various requirements, including alert correlation [3]. Inferring common attack patterns and rules from the alerts helps in understanding the threat landscape for the defenders and allows for achieving cyber situational awareness (CSA) [27] in all three of its stages, namely perception, comprehension, and projection [11]. Perceiving the security situation via intrusion detection systems (IDS) and comprehending cyber attacks via alert correlation allows for the projection of ongoing attacks and prediction of upcoming security events. This article builds foundations from our previous work, in which we tackled various related topics, such as method selection [20], long-term observations [18], and system design [19]. Herein, we present further explorations within such fields and shed light on lessons learned toward the utmost goal of achieving attack projections and building predictive blacklisting capabilities.

Nowadays, data-driven curation and analysis related to intrusion detection alerts could be considered as big data problems due to their volume, variety, and velocity. This is especially apparent in collaborative intrusion detection and information exchange settings [43]. Searching for interesting pieces of information in such distributed data has become complicated for human analysts and, thus, there is a need to investigate machine-supported analytics and systems toward this goal. Raw data is impractical to use in incident response; the number of IP addresses and other network entities is too large to employ them in a blacklist or as firewall rules. In order to make use of intrusion detection alerts at large, there is a need to select a smaller amount of network entities that are likely to continue with malicious activities against particular targets. We typically refer to such attempts as predictive or personalized blacklisting, and we believe that the methods of unsupervised data mining could be used to achieve such goals.

To this end, we state that the main motivation for this work is to distill a small and highly actionable set of network entities from large volumes of alerts generated from heterogeneous and distributed sources. In this context, we propose to achieve this goal via personalized and predictive blacklisting. Namely, we have to:

- Analyze the data from an alert sharing platform using, preferably, an unsupervised data mining approach, which does not require significant human interaction or expert input.
- Predict the continuation of running attacks so that we may estimate future malicious actors.
- Distill a predictive blacklist that contains expected future malicious actors.
- Personalize the blacklist so that recipients receive only the list of malicious actors that are expected to threaten them.

To this end, in this article, we propose an approach based on sequential rule mining to infer knowledge from intrusion detection alerts and use it to create personalized and predictive blacklists. Our work is motivated by the needs of the alert sharing platform SABU [6], which is operated in national research and education network (CESNET), and distributes alerts from various IDS and honeypots located in multiple geographically and organizationally distributed networks. We use Analysis of Intrusion Detection Alerts (AIDA) [21], an alert processing framework that exploits techniques of stream data processing and data mining over the alerts [19]. The AIDA framework is capable of mining sequential rules for use in predicting upcoming security events. The predicted events can then be used to create a blacklist of IP addresses that are likely to persist with malicious activity. Further, our approach also predicts the location of the upcoming security events, which allows for personalized blacklisting for particular organizations participating in information exchange within the SABU platform. The contributions of the article can be summarized as follows:

— We propose a unique combination of information sharing, data mining, and predictive analysis to address an important, applied problem. Specifically, we illustrate the use of the Top-K sequential rule mining [16] in the analysis of intrusion detection alerts in the SABU sharing platform [6]. The results can be directly employed as predictive rules for predicting upcoming security events. We show that the data mining outputs are mostly stable over time and that the predictions leave enough time for the defenders to react to predicted security events.
— We explore the idea of predictive and personalized blacklisting, a novel approach that has highly practical impacts on operational networks. We outline the process of generating predictive and personalized blacklists from the predicted events. Our results show that the predictive blacklist can be significantly smaller than a blacklist made of raw data and achieves the success rate of over 60% in predicting security events.
— We illustrate the use of new and up-to-date datasets and operational tools, such as the AIDA framework [21] and the anonymized, publicly available dataset [24] of alerts from the SABU platform [6], to allow for the reproducibility of our research.

The remainder of this article is organized into five sections. Section 2 presents the related work on sharing intrusion detection alerts, their correlation, and predictive analysis. Section 3 introduces the SABU platform, from which the input data was obtained, and the AIDA framework that we used for processing the data. Section 4 presents the experimental setup and results. Discussion of the results and comparison to related work is presented in Section 5. Section 6 concludes the article and pinpoints a few topics for future work.

2 RELATED WORK

Background to our work consists of three main parts. First, we set our work in the context of collaboration and information exchange in cybersecurity and alert correlation. Second, we summarize related work on attack prediction supported by data mining, which paves the way to our proposed approach. Third, we summarize related work on predictive blacklisting, which shares similar goals as our work. The most substantial related work is summarized in Table 1.

Collaboration and information exchange has become a substantial part of the cybersecurity practice, from the cooperation between IDS [43] to nation-wide information sharing [37]. Research and development have focused on automating the process so that timely and important pieces of information would be exchanged to provide early warning [35], increase the precision of intrusion detection, or simply to announce a threat. The task of alert correlation [9] is to put together corresponding alerts, find relations between alerts, reconstruct the progress of an attack, and also to recognize the intent of an attacker while analyzing the impact of potential security incidents. The detailed tasks and processes of security alert correlation were initially proposed by Valeur et al. [42]. Briefly, the alerts from sensors need to be normalized, fused (aggregated), and verified first. Subsequently, the attack session can be reconstructed for the purposes of attack aim recognition, multi-stage attack correlation, and impact analysis [42]. Cuppens and Miège [7] discussed the problem from the perspective of collaborative intrusion detection, and Elshoush et al. [10] surveyed methods of alert correlation in a collaborative environment. In recent years, alert correlation is often supported by data mining [3]. A review of particular data mining methods applied to the analysis of cybersecurity alerts was presented in our previous work, in which we discussed the method selection process [20].

Predicting cyber attacks is an emerging research topic that often builds upon alert correlation [22]. Herein, we summarize the works that used data mining to produce outputs, such as attack models and predictive rules, which can be used for making predictions. Li et al. [29] and Lei et al.

Table 1. Summary of Related Work

Focus	Author & Year	Dataset	Methodology	Key results
Attack prediction	Li et al. 2007 [29]	DARPA 1999 & DARPA 2000	Association rule mining	Data mining used to construct attack graphs for attack projection
	Lei et al. 2007 [28]	Honeypots	Sequential pattern mining	
	Farhadi et al. 2011 [13]	DARPA 2000	Sequential pattern mining	Efficient real-time implementation, data mining used to construct Markov models for attack projection
	Ramaki et al. 2015 [34, 36]	DARPA 2000 & GCP	Real-time episode correlation	Early warning system with up to 90% accuracy under special circumstances
	Kim and Park, 2014 [26]	Theoretical model	Association rule mining	Estimated 80% accuracy, discussion of time window, and thresholds settings
	Jiang et al. 2016 [25]	Honeypots	Association rule mining	Predictions were shown to reduce the false positive rate of intrusion detection by 30%
Predictive blacklisting	Zhang et al. 2008 [45]	DShield	Relevance ranking, link analysis method similar to Google's PageRank	First experimental work, discussion of the influence of collaboration, median hit rates below 20%
	Soldo et al. 2010 [38]	DShield	Time series analysis, clustering, and similarity search	Various methods were shown to increase the hit count by up to 70%
	Ma et al. 2015 [30]	Honeypots	Enhancing the approaches of Zhang et al. [45]	Personalized, although with hit rates below 15%
	Freudiger et al. 2015 [17]	DShield	Controlled data sharing	Improvements in prediction accuracy and reduction of false positives, privacy-preserving
	Melis et al. 2019 [32]	DShield	A hybrid approach that combines previous works [17, 38]	Balancing the two approaches to achieve accuracy of Ref. [38] and false positive rates of Ref. [17]
	Bartoš et al. 2019 [2]	SABU	Network entity reputation scoring based on long-term observation (weeks, months)	Hit rates up to 43% for 2,000 items in the blacklist, 89% with 500 items in the blacklist, and even 100% with 100 items in the blacklist

[28] used data mining to construct attack graphs for attack projection. Li et al. used association rule mining, while Lei et al. used sequential pattern mining. Farhadi et al. [13] leveraged data mining to construct a Markov model as a means of attack prediction. Association rule mining was used, and algorithms were provided. The authors also proposed using stream-based data processing. RTECA is a framework for early warning based on real-time episode correlation proposed by Ramaki et al. [34] and one of the few examples of an existing tool. Kim and Park [26] used continuous association rule mining algorithm to create sequential rules by applying attack threads and attack sessions to sequential association rules for attack and threat prediction. Recently, Jiang et al. [25] used association rule mining on honeypot logs to predict suspicious network traffic present in the honeypots. Simultaneously, attack prediction methods based on machine learning were proposed. For example, Soska and Christin [39] used machine learning techniques for automated detection of vulnerable websites before they turn malicious. More recently, Veeramachaneni et al. [44] combined supervised and unsupervised methods to improve the detection rate of unsupervised methods alone. Other methods for predicting cyberattacks or cybersecurity situations were discussed in the survey of attack prediction, projection, and forecasting methods [22].

Predictive blacklisting is a proactive variation to blacklisting that focuses on creating a blacklist of network entities (e.g., IP addresses and hostnames) that are likely to behave maliciously. First attempts at predictive blacklisting date back to 2008 in work by Zhang et al. [45], who use data from DShield to create personalized predictive blacklists using the method of link analysis. Variation of other methods, such as time series analysis, clustering, and similarity search, was later proposed by Soldo et al. [38] on the same data, and by Ma et al. [30] on the data from honeypots. Freudiger et al. [17] proposed another approach that is based on controlled data sharing, using the data from DShield, and is privacy-preserving. Finally, Melis et al. [32] combined two of the previous approaches [17, 38], and achieved balance of their strengths and weaknesses, again on DShield data. Predictive blacklisting is also the desired use case of network entity reputation scoring in the work of Bartoš et al. [2], who used the data from the SABU alert sharing platform [6] that was also used in this work. The attention of the researchers in the field is also focused on countermeasure selection, e.g., as surveyed by Nespoli et al. [33]. Similarly to preceding tasks, a large amount of heterogeneous data and events calls for research on selecting optimal countermeasures and development of reaction frameworks that can be combined with Security Information and Event Management (SIEM), early warning, and prediction systems [35].

3 SABU PLATFORM AND AIDA FRAMEWORK

Our research was motivated by the needs of the SABU alert sharing platform, which we used to evaluate our proposed approach. We also present AIDA, an analytical framework, which we also leverage to implement the proposed approach. In this section, we briefly describe the SABU platform and the AIDA framework, as well as the dataset of alerts obtained from SABU that we used for the experimentations.

3.1 SABU Alert Sharing Platform

To illustrate the concepts of security alert sharing in practice, we introduce the SABU alert sharing platform [6]. The intended scope of its usage is the network of CESNET, Czech National Research and Education Network, and its partners from academia, the public sector, and industry. The purpose of SABU is to exchange the alerts provided by various IDS, honeypots, and third-party data sources. The alerts should be received by security teams (CSIRT, CERT) and their incident handlers or by active network defense devices, such as firewalls that would automatically add network entities in the alerts on a blacklist. A schematic description of the SABU platform is presented in Figure 1. A central component of the SABU platform is Warden [5], a hub that receives the alerts from sending connectors (left) and distributes them to receiving connectors (right). Sending connectors are various IDS and honeypots. The community around SABU contributes mostly with alerts from network-based IDS and a variety of low-interaction honeypots [5, 6]. Receiving connectors are analytical and incident response tools. In the bottom half of the schema, we can see AIDA [19], an analytical framework that we discuss later in this article.

SABU uses Intrusion Detection Extensible Alert (IDEA) format [4] for exchanging information on security events. IDEA is inspired by Intrusion Detection Message Exchange Format (IDMEF) [8], but is customized to reflect operational needs of network monitoring and incident response community, includes a taxonomy of security events, and prefers data serialization in JSON. A detailed description and examples of messages in the format are available on the IDEA website [4]. The most interesting pieces of information are to be found in *Source* and *Target* fields, where we typically find IP addresses of attackers and victims along with ports and services involved in the event. *Node* contains the identifiers of the data source, such as the name of an IDS that sent the alert. *Category* is a mandatory entry that describes the type of event, such as network scanning (*Recon.Scanning*), brute-force password attacks (*Attempt.Login*), and exploitation attempt

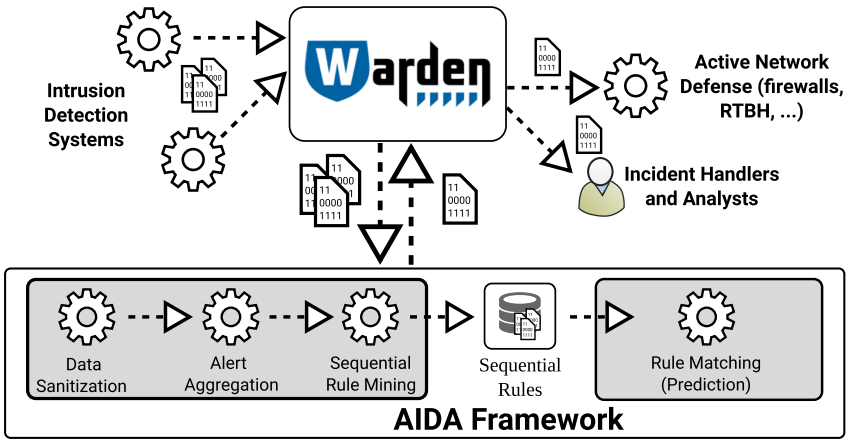

Fig. 1. Schema of SABU alert sharing platform and AIDA framework.

(*Attempt.Exploit*). *DetectTime* is a mandatory entry that indicates the time when the alert was raised, i.e., time of detection. Time of start of a reported event is often hard to set and, thus, this value is only optional.

3.2 AIDA Framework

AIDA [19, 21] is a modular framework for the stream-based analysis of intrusion detection alerts using the concepts of big data processing, data mining, and complex event processing. A simplified schema of the AIDA framework and its deployment in the SABU platform is depicted in Figure 1. The framework receives alerts from Warden, the central component of the SABU platform. The alerts are distributed to processing components of the framework as a data stream using the Kafka message broker [40]. The data is first processed by three components based on the Apache Spark [41] framework for stream-based data processing. The first component performs sanitization and semantic filtering of the data, such as filtering alerts of low interest. The other two components perform aggregation and data mining, as described in previous works [18, 20, 23]. The mined rules are stored in a database, where they are accessed by the rule matching component. The rule matching component is based on Esper [12] and its Event Processing Language (EPL) that allows for translation of the sequential rules into SQL-like queries over the stream of alerts. If the first part of the rule is found using the EPL query, the remainder of the rule is predicted and reported as an alert that is sent back to Warden.

The first application in the processing pipeline is an Alert Aggregation component implemented as a Spark application. Its task is to find and mark redundant alerts that may slow down and influence the result of the subsequent operations, especially the data mining process [20]. It processes the stream of alerts, keeps the vectors of their main features (timestamp, source and target IP addresses, source and target ports, alert source, and event category) in a predefined time window, and matches the alerts with similar features according to a set of rules [23]. The first alert with such features is left untouched; all the others are marked as duplicates of the first alert.

The needs of alert correlation are fulfilled with a Data Mining component. Its purpose is to infer frequent patterns in the alerts so that the patterns may later be used to predict upcoming security events. The component consists of the database generator and the rule miner. The generator is implemented as a Spark application that continuously extracts sequences from the alerts and saves them to a database. A sequence is a vector of itemsets. In our implementation, an itemset is an n-tuple consisting of the name of the sensor, alert type, and target port. Formally:

$$i = (sensor, category, dstport) \tag{1}$$

The itemsets in a sequence share a common property. In our implementation, the common property is the source IP address. A sample sequential database with four sequences can be written as:

$$(i_a, i_b, i_c), (i_a, i_b, i_c), (i_a, i_b, i_d), (i_x, i_y, i_z) \tag{2}$$

The second part of the Data Mining component is the rule miner, a batch script that is executed periodically, typically once in a day, when enough sequences are extracted by the stream-based database generator. The miner uses the SPMF library [14] and mines frequent patterns or rules from the collected sequences, depending on the selected algorithm. For the purposes of security event prediction, we chose TopSeqRules, an effective sequential rule mining algorithm [16], to mine rules that are directly applicable for predictions. Once the mining is completed, the sequential database is cleared for the next collection period. A sequential rule has the form of:

$$((i_a, i_b, \ldots), (i_i, i_j, \ldots), support, confidence), \tag{3}$$

where *support* stands for the frequency of the rule and is defined as the number of sequences corresponding to the rule and divided by the total number of sequences in the sequential database. The value of *confidence* is the percentage of sequences that started with the first part of the rule and continued with the second part of the rule. This can be interpreted as a probability that the second part of the rule will follow the first part of the rule. For example, when processing the sample sequential database in Equation (2), the algorithm would produce rules such as:

$$((i_a, i_b), (i_c), 2/4, 2/3) \tag{4}$$

In this example, (i_a, i_b) and (i_c) are fragments of a frequent sequence (i_a, i_b, i_c). This sequence appeared twice in four sequences and, thus, the *support* value is $2/4$. Three sequences started with (i_a, i_b), but only two ended with (i_c) and, thus, the confidence value is $2/3$. Finally, the TopSeqRules algorithm [16] selects the Top-k rules. Top-k rules are a set of up to k rules, in which each rule has confidence greater than or equal to minimal confidence and greater support than the rules not belonging to the set. In the implementation, k is set to 10, and the minimal confidence is set to 0.5.

The final application in the alert processing pipeline is a Rule Matching component that processes the stream of alerts and queries them for the presence of predefined patterns (sequences) to perform predictions. This component takes sequential rules from the previous components and converts them into the queries over the stream of alerts. If the first part of a predictive rule is found in the stream of alerts, a new alert is generated using the second part of the predictive rule. The component was implemented as a stand-alone application using Esper, a tool for Complex Event Processing (CEP) over a stream of data. In addition, the component checks if a predicted event appeared in the stream of alerts and measures the time gap between the prediction and the predicted event occurrence. Thus, we may log and measure the precision of predictions and collect feedback on the predictive rules.

3.3 Dataset

To allow for the reproducibility of the results, we used the dataset [24] of intrusion detection alerts collected from SABU. The data was collected continuously as they appeared in the SABU platform for the period of one week, from March 11 to March 17, 2019. Almost 12 million alerts were collected from 34 network-based IDS, honeypots, and other data sources deployed in three distinct organizations: national research and education network, a large campus network, and a commercial Internet service provider. The alerts are stored in the IDEA format and categorized using the taxonomy of security events included in the IDEA definition. The IP addresses in the alerts are anonymized.

4 DATA MINING FOR PREDICTIVE BLACKLISTING

In this section, we provide the description, evaluation, and discussion of an experiment that illustrates the use of data mining for predictive cyber situational awareness and predictive blacklisting in a collaborative environment. First, we describe the setup of the experiment, for which we used the dataset of alerts from the SABU platform and the AIDA framework. The important features are settings of the data sanitization, alert aggregation, and the data mining components of the AIDA framework. Subsequently, we provide the results obtained by processing the dataset. Finally, we discuss the usability of the outputs, their stability in time, and how to distinguish good results from others.

4.1 Experiment Setup

First, we split the dataset into seven parts, one part per day, during the dataset collection period. Subsequently, the first 3 days were primarily used for the learning phase and mining of the rules. The remainder of the dataset was used for the evaluation phase. Nevertheless, we also performed data mining over all 7 days to observe the stability of the data mining outputs during the whole week.

It is important to sanitize the data in the input before processing. In our case, we dismiss the alerts that are either malformed or not relevant for the given use case. The dataset was revealed not to contain any malformed alerts. However, there are numerous alerts that are not usable in the use case of predictive blacklisting. For example, the SABU platform and the dataset contains alerts of vulnerabilities found on hosts in the network or alerts of phishing attacks. Such alerts do not contain the entries required by our proposed data mining approach, most importantly, the source IP address, and it would be hard to correlate them to the vast majority of the alerts.

An important feature of the experiment is alert aggregation. The task of alert aggregation is one of the major steps in alert processing [42]. We evaluated the options of alert aggregation in an alert sharing platform in our previous work [23] and identified four scenarios, in which the alerts may be aggregated. The two basic types of aggregation that we used in the experiment are the aggregation of duplicate entries and persisting events. The duplicate entries, i.e., the same alerts that appear in multiple copies, can be seen in the SABU platform in low numbers (less than 1%) due to errors in sending and sharing. The alerts demonstrating the continuous malicious activity, however, are a major concern, as they stand for around one-half of the alerts. Such alerts are raised in response to long-lasting events by stateless IDS. For example, if a scan of the network takes one hour and the scan detection method is based on a threshold of packets in the last five minutes, the detection system raises 12 alerts of the same event, each with a different timestamp. Multiple alerts of a single long-lasting event skew the results of statistical analysis and data mining. The data mining over unaggregated data produced sequential patterns and rules that described the continuation of an event [20]. Although the number of such patterns and rules on the output might be reduced by certain variants of the mining algorithms, we decided to perform the aggregation first, and, consequently, execute the data mining over aggregated alerts to avoid such outputs completely. Two other scenarios for alert aggregation are alerts of the same event raised by different IDS [23]. Aggregating such alerts might be in conflict with sequence mining because the mined sequences might correspond to the scenarios for aggregation. We did not include such aggregation in our experiments. However, similarities in the results of data mining and potential aggregates are discussed later.

When the data is preprocessed, the remaining task is to select a suitable data mining method that would produce results of reasonable quality in an acceptable time frame. The sequential pattern and rule mining were identified as the most suitable methods for the use cases of alert correlation

and attack prediction in our previous work [20]. Briefly, instances in a sequential pattern and rule mining correspond to the series of actions by the attackers. The actions in the sequences are sorted in time and, thus, illustrate what the attackers do first, how they proceed, and, in case of sequential rules, how they are most probably going to take action next. Because we aim at predictive blacklisting in this article and not alert correlation, we focus only on sequential rule mining methods. Out of the possible options [15], we selected the Top-K sequential rule mining method [16] with K set to 10 and implemented it using the SPMF library [14]. Thus, we obtain the 10 most interesting rules from the dataset. When using plain sequential rule mining, we would have to set thresholds or select the viable results manually.

For completeness, it is also imperative to briefly highlight alternative approaches. For example, association rule mining [1] is a viable alternative to sequence mining that illustrates what actions adversaries are likely to perform simultaneously. However, it does not reflect the timing, which we believe is a strong contribution of the sequence mining methods. Another alternative is frequent episode mining [31], which is very similar to sequence mining but does not consider gaps between events, a condition that would assume and force the events within an episode not to interleave with other events.

4.2 Experimental Results

During the experiment, we processed the whole dataset of 12 million alerts. From the average number of 1.6 million alerts per day, we dismissed around 115 alerts of irrelevant events (reports of discovered vulnerabilities) and 13,000 alerts without any source IP address. Further, we aggregated around 2,500 duplicated alerts (0.15%) and 830,000 persisting alerts (49.61%).

The experimental results are summarized in Tables 2 and 3. Table 2 lists the sequential rules that were mined in the learning phase from the data from the first 3 days. In total, we mined 16 rules during the 3 days, Top-10 rules from the first day, four new rules in Top-10 rules on the second day, and two new rules on the third day. The rules are numbered to connect them with their parameters presented in Table 3. Table 3 lists the values of the parameters of the rules identified by their number. For each rule, we provide the mean values of support and confidence and their deviations. The support and confidence values are parameters of the rule produced during the mining. If a rule was mined multiple times (in multiple days), we calculated the mean values of particular entries and their deviations.

Further, Table 3 contains the results of the evaluation phase. The successful prediction rate is a mean value of successful prediction rates in all days, in which the rule was used for predictions. Min. Δt and Avg. Δt stand for minimal and average time differences (in seconds) between prediction and observation of a predicted event, i.e., its arrival to the analytical framework. In our previous work [18], we predicted alerts in real time and, thus, were able to use the time in which the prediction was made. This is not possible when processing the dataset and, thus, we had to alter the calculation. The time difference is calculated as a difference in timestamps of the last alert in the first part of the rule and the timestamp of the successfully predicted rule.

An interesting problem occurred during the evaluation, in which, in several cases, the time difference between the prediction and the predicted event was negative. This might happen in real-time evaluations, as well. The mandatory timestamp of each alert is the *DetectTime*, a time when the event was detected, not when it started. Varying time windows for the detections and the delays in sending and sharing the alerts cause disordering of the alerts.

Table 4 provides the summary of processed alerts (after sanitization), the number of predictions, and the prediction success rate per day. The number of predictions is the number of alerts predicted using the data from a given day using the rules mined in previous days. The number of successful predictions is the number of predicted alerts, for which we observed the predicted alert in the

Table 2. Mined Sequential Rules

Rule	Input		Output
1	OrgA.tarpit:Recon.Scanning:8000	⇒	OrgA.tarpit:Recon.Scanning:88
2	OrgA.nemea.hoststats:Recon.Scanning:None	⇒	OrgA.hoststats:Recon.Scanning:None
3	OrgA.tarpit:Recon.Scanning:2323	⇒	OrgA.tarpit:Recon.Scanning:23
4	OrgA.tarpit:Recon.Scanning:8000	⇒	OrgA.tarpit:Recon.Scanning:8080
5	OrgA.tarpit:Recon.Scanning:80, OrgA.tarpit:Recon.Scanning:8000	⇒	OrgA.tarpit:Recon.Scanning:8080
6	OrgA.tarpit:Recon.Scanning:88, OrgA.tarpit:Recon.Scanning:8000	⇒	OrgA.tarpit:Recon.Scanning:8080
7	OrgA.nemea.hoststats:Recon.Scanning:None, OrgB.nemea.hoststats:Recon.Scanning:None	⇒	OrgA.hoststats:Recon.Scanning:None
8	OrgA.tarpit:Recon.Scanning:8080, OrgA.tarpit:Recon.Scanning:8000	⇒	OrgA.tarpit:Recon.Scanning:88
9	OrgA.tarpit:Recon.Scanning:8080, OrgA.tarpit:Recon.Scanning:8000	⇒	OrgA.tarpit:Recon.Scanning:80
10	OrgA.tarpit:Recon.Scanning:88, OrgA.tarpit:Recon.Scanning:80	⇒	OrgA.tarpit:Recon.Scanning:8080
11	OrgB.nemea.bruteforce:Attempt.Login:23	⇒	OrgA.tarpit:Recon.Scanning:23
12	OrgA.tarpit:Recon.Scanning:88, OrgA.tarpit:Recon.Scanning:8080	⇒	OrgA.tarpit:Recon.Scanning:80
13	OrgA.tarpit:Recon.Scanning:2222	⇒	OrgA.tarpit:Recon.Scanning:22
14	OrgB.nemea.hoststats:Recon.Scanning:None, OrgA.hoststats:Recon.Scanning:None	⇒	OrgA.nemea.hoststats:Recon.Scanning:None
15	OrgA.tarpit:Recon.Scanning:8080, OrgA.tarpit:Recon.Scanning:81	⇒	OrgA.tarpit:Recon.Scanning:80
16	OrgA.tarpit:Recon.Scanning:80, OrgA.tarpit:Recon.Scanning:81	⇒	OrgA.tarpit:Recon.Scanning:8080

The Rule Items are in the form Organization.Sensor:Alert_type.Alert_subtype:Destination_Port.

Table 3. Parameters of the Mined Rules

Rule	Support	Confidence	Success rate	Min. Δt	Avg. Δt	Times in Top-10
1	0.0052 ± 0	0.5030 ± 0	(15.89 ± 11.83)%	901	23,157	1
2	0.0167 ± 0.0035	0.7800 ± 0.0569	(96.80 ± 0.62)%	1	132	7
3	0.0327 ± 0.0048	0.6043 ± 0.0191	(68.51 ± 1.42)%	901	1,669	7
4	0.0044 ± 0.0010	0.5164 ± 0.0139	(21.33 ± 11.60)%	901	21,404	3
5	0.0033 ± 0.0006	0.5821 ± 0.0069	(36.85 ± 18.80)%	909	17,333	2
6	0.0038 ± 0	0.5765 ± 0	(31.32 ± 18.84)%	901	16,694	1
7	0.0047 ± 0.0008	0.7433 ± 0.0940	(97.60 ± 1.02)%	1	749	6
8	0.0038 ± 0	0.5607 ± 0	(27.86 ± 18.77)%	902	18,781	1
9	0.0036 ± 0	0.5272 ± 0	(27.01 ± 17.02)%	934	18,386	1
10	0.0031 ± 0.0005	0.5408 ± 0.0366	(51.98 ± 10.05)%	279	12,191	3
11	0.0039 ± 0.0014	0.6098 ± 0.0547	(57.43 ± 6.35)%	5	8,197	5
12	0.0028 ± 0.0002	0.5697 ± 0.0049	(40.00 ± 8.75)%	907	14,493	2
13	0.0035 ± 0.0002	0.6438 ± 0.0069	(45.86 ± 3.43)%	901	9,389	6
14	0.0039 ± 0.0004	0.5368 ± 0.0244	(84.14 ± 4.78)%	1	4,313	3
15	0.0026 ± 0	0.5702 ± 0	(31.08 ± 4.68)%	901	14,926	1
16	0.0026 ± 0	0.5623 ± 0	(40.24 ± 3.25)%	901	11,925	1

Table 4. Success Rate and Data Reduction for Predictive Blacklisting

Day	Alerts	Predicted alerts	Successful predictions	Success rate	Reduced data volume
2019-03-11	1,708,733	51,239*	31,958*	62.37%*	1.87%*
2019-03-12	1,664,723	46,721	30,322	64.90%	2.81%
2019-03-13	1,548,186	53,906	32,899	61.03%	3.48%
2019-03-14	1,607,630	45,637	30,039	65.82%	2.84%
2019-03-15	1,710,095	47,481	31,234	65.78%	2.78%
2019-03-16	1,776,168	53,656	35,735	66.60%	3.02%
2019-03-17	1,699,081	50,061	33,325	66.57%	2.95%

(*evaluated only with the rules mined from the same day).

data. The success rate is the ratio of successfully predicted alerts to all predicted alerts. The values are very similar for every day, fluctuating around 50,000 predictions, 32,000 successful predictions, and 65% success rate, except for the first day, which was used only for mining, not for evaluation. Finally, the last column shows the number of predicted alerts as a percentage of the number of all processed alerts, which illustrate the data volume reduction.

5 DISCUSSION

Herein, we discuss the results of the experimentation in comparison to our previous work and also with respect to the novel topic of predictive blacklisting. The major metrics we are interested in are the prediction accuracy and data volume reduction for predictive blacklisting. Subsequently, we compare our results with the results of related work. Finally, we delve into the rules to discuss the features of a "good" rule, i.e., a rule that is suitable for predictive blacklisting.

5.1 Prediction Accuracy

The first major output of our experiment is the evaluation of the success rate of the predictive rules. The confidence value of the rule can be considered as the expected success rate. However, as we can see in Table 3, the success rate can be substantially different (lower) from the confidence values. This is not necessarily a sign that a rule is bad. Even if we apply the rules mined in one day, we may still see a success rate that is much lower than the confidence (see the first line in Table 4). The major reason for this is hidden in the process of generating sequences for sequential rule mining. One item (alert) may appear in multiple sequences. At the same time, there can be multiple sequences that share one or more items. The data mining also processes the sequences from the whole, so the sequences may also be longer than the aggregation window of the aggregation component and, thus, might add many more sequences to the sequential database. The data mining component collects all the possible sequences, while the rule matching component uses each item (alert) only once for each rule to make predictions.

It is worth noting that we have no information in the dataset nor in the SABU platform about the mitigation actions. Thus, the predictions may be unsuccessful because the malicious actors were already put on blacklists and filtering rules, which prevented them from proceeding with the attack. Such a measurement would require collaboration with a number of organizations and their incident response teams. However, we do not expect this to be a significant factor. Mitigated attacks would rather decrease the confidence value of the mined rules than decrease the success rate.

5.2 Data Volume Reduction

The second major output is data volume reduction. As we can see in the last column of Table 4, the volume of predicted alerts is significantly smaller (around 3%) than the volume of alerts on the input, which is very convenient for the recipients of the data. Using the raw data for blacklisting and mitigation would be very hard due to the limitations of active network defense appliances and frequent changes due to the continuous streaming of the data. Thus, by employing predictive blacklisting, the sharing platform may redistribute only around 50,000 predicted alerts per day, instead of 1.5 million raw alerts. Although the particular prediction might not be accurate, the overall success rate of all predicted events is between 61 and 67%. This is also a hit rate of the blacklist, i.e., a ratio of alerts actually used for mitigating attacks. When using the raw data, the hit ratio would be around 2%. Thus, a personalized blacklist is much smaller in size, yet more actionable for the needs of network defense. The predicted alerts are proactive, i.e., they contain information on events that are likely to happen in the near future. In comparison, the raw data is only reactive, i.e., they refer to actions from the past and do not contain any information about present or future situations.

We may employ the personalization of the blacklist to further reduce the volume of data. A particular recipient may wish to receive only the alerts of events that are predicted to happen in the recipient's network. The number of alerts for particular recipients could be even lower. In the experiment, we processed the data from three organizations. However, the organizations did not contribute equally, and the network of one organization was connected to the Internet via the backbone network of another organization. The third organization did not have an overlap with the other two. More information is provided in the dataset description [24]. Due to the network topology, placement of the IDS, and imbalance between the number of alerts provided by each organization, most of the rules reflected the activity of an attacker in the backbone network and the majority of the predicted alerts were predicted to happen in the backbone network as well. Thus, depending on the circumstances, the personalization of the blacklist might not be balanced well.

The third consideration for the data volume reduction is the number of items in the blacklist that can be lower than the number of alerts. In the experiment, we projected the activities of malicious actors identified by the IP address and, thus, the most important item in a predicted alert is the source IP address. In all stages of alert processing, the number of unique source IP addresses is approximately one half of the number of alerts. Thus, the blacklist distilled from around 50,000 alerts would contain around 25,000 unique source IP addresses. Nevertheless, in operational settings, the predicted alerts are raised continuously, and the blacklist would also be continuously updated. The actual size of the blacklist would depend on the expiration time of the entries in the blacklist. If the entries would expire after one day, then the size of the blacklist would correspond to the number of predicted per day, as summarized in Table 4. When employing the knowledge on time differences between prediction and predicted event, we may set the expiration accordingly so that the item would expire sooner, and the continuous size of the blacklist would be even smaller. However, this should be evaluated in an operational setting.

5.3 Comparison to Previous Results

The results are mostly similar to our previous measurements in the live environment, namely in numbers of dismissed and aggregated alerts on the input [23] and the predictive rules on the output [18]. The major difference is the stability of the rules in time, as presented in our previous work [18], where we observed 8 rules out of 10 to be mined every day while displaying similar values of support and confidence. In our new experiment, we observed only two rules that appeared in Top-10 mined rules in all 7 days. We mined three rules on the first day and two rules on the third

day that did not appear in the outputs anymore. Nevertheless, the remainder of the rules appear at least three times in the outputs, and their values of support and confidence are very similar in all cases. The first experiment and the dataset collection were more than one year apart and, thus, the composition of the data sources and alerts have changed. Long-term monitoring would be more helpful, but we may still assume that the majority of the rules are stable in the short term.

It is also possible to compare our results to the result from related work on predictive blacklisting. The success rate of the prediction is, in essence, the same metric as the hit rate that is used in previous works on predictive blacklisting. The hit rate is the number of hits divided by the size of the blacklist, where the hit is the observation of malicious actions by an entity on the blacklist. Similarly, the hit count is corresponding to the number of successful predictions. Previous results show median hit rates in various experiments under 20% [30, 45]. Further works improved the hit counts but did not discuss the hit rates [17, 32, 38]. The highest hit rates were achieved by Bartoš et al. [2], but the hit rate degrades with the size of the blacklist, as presented in Table 1. The success rate of around 65% in our work outperforms previous works even with large blacklists. Nevertheless, it is important to keep in mind that predictions illustrated in this work apply in the short term (seconds to hours), while previous works often use longer time spans (hours to days and more).

5.4 Selection and Usability of Rules

The final point of discussion is the variety of rules and the differences in their features and usability. The data mining method we used is very convenient; it is fully automated and does not require a training phase or supervision by a human expert. Still, the rules on the outputs are not perfect. Increasing the quality of the input data has a major impact on the quality of the results and, thus, we employed data sanitization and alert aggregation [20, 23]. The SABU platform is further limited by a low variety of alerts due to the fact that most of the data sources are network-based IDS and honeypots that all detect similar events, such as network scanning, brute-force password attack, and exploitations. There are not many finely-grained alerts of particular phases of such events or continuations of the attack observed by host-based IDS.

We identified several features of a good rule that should pave the way for future work. The AIDA framework allows us to manually activate and deactivate what the operator considers as good or bad. Understanding and formalizing the features of a good rule would allow for proper and even automated selection. The first two features, support and confidence values, are set during data mining and are clear to understand. The higher the confidence, the better. Low support would indicate a rare sequence, but, otherwise, is not that important. Another feature of a good rule is its stability in time. If a rule is mined several days in a row with similar support and confidence values, we consider it to be good. Otherwise, it would suggest an anomaly in the input data. Another feature is the success rate that also needs prolonged evaluations in cooperation with the rule matching component of the AIDA framework. These four features are exact and can be formulated into thresholds if needed.

There are two more features of a good rule that are not clear from the values but require an expert or operator's insight. The first would be the rule that overlaps with one of the scenarios for alert aggregation, specifically the detection of the same event by different sensors from different (non-overlapping) networks [23]. Large-scale network scanning is a prime example of such an event. Such a rule would actually be good because it illustrates how the attack spreads over the networks. The value of such a rule is in predicting the next target of the malicious activity, even if it is the same action that the attacker has used before. The second model situation would be a rule that is not necessarily a sequential rule, but rather an association rule in disguise. We can see such rules frequently as combinations of ports scanned by the attackers. For example, rule 3

in Table 2 suggests that a scanner that scans for port 2323 will also scan for port 23. In practice, however, scanners frequently scan port 23 for open Telnet ports and might also scan port 2323 as its alternative binding. The scans are very frequent so the data mining component observes sequences in both directions, 23 => 2332 and 2323 => 23. While both sequences are frequent, the scans of port 2323 are rarer than scans of port 23 and, thus, the rule 2323 => 23 has higher confidence and higher chance to appear in Top-K rules. Association rule mining would find a strong association between the two events, which would be good, but for sequential rules that take timing into account, this would not be a good rule.

6 CONCLUSION

In this article, we presented an approach to creating predictive blacklists from shared intrusion detection alerts using the methods of data mining and predictions of cyberattacks. We briefly described the alert sharing platform, from which we obtained the data, and the analytical framework that was used for data processing. The data was anonymized and made publicly available from Ref. [24] to widely support research reproducibility and auxiliary experimentation by the relevant research communities.

Using the methods of sequential rule mining, we were able to mine predictive rules from the data and use them to predict upcoming cybersecurity events. Over 60% of the predicted alerts were observed to occur and, thus, may be used to create a predictive blacklist of IP addresses that are likely to act maliciously. From a more practical perspective, the predictive blacklist is significantly smaller in volume than the raw data. This fact is important for recipients of the shared data that could otherwise be overwhelmed by the volume of raw data that is hard to process and mostly irrelevant. In this context, the proposed approach provides a smaller volume of data with higher usability. In a production deployment of the SABU alert sharing platform, the AIDA framework uploads the predicted alerts to the sharing platform, and the data consumers may access the predicted alerts and create their own blacklists.

We believe there are many opportunities for future work. First, we identified a set of features of a good rule, i.e., a sequential rule that is most likely going to predict relevant events and leave enough time for response and mitigation. Formalizing the features of a good rule would allow us to create a set of filters or thresholds that would select the most suitable rules. Alternatively, we may train a machine learning algorithm, such as a decision tree, to automatically select the good rules. Second, we would like to continue the measurements in the operational environment and focus on predictive blacklist generation and its usage at the receiving organizations. It would also be imperative to measure the size of a predictive blacklist and its success rate in cyber attack mitigation per organization, hence, providing empirical evidence and measurements of the benefits of information sharing. Although this work mainly involves network security and incident response, we believe that our work may be used in other fields as well. For example, data mining experts may use our results from a specific domain and generalize and optimize them. Machine learning experts may follow the ideas outlined in the discussion on the automated selection of good rules. Further, our work may support future research in threat intelligence and alert correlation by identifying subsets of alerts that are perspective for particular analyses.

REFERENCES

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD'93)*. ACM, New York, NY, 207–216.

[2] Václav Bartoš, Martin Žádník, Sheikh Mahbub Habib, and Emmanouil Vasilomanolakis. 2019. Network entity characterization and attack prediction. *Future Generation Computer Systems* 97 (2019), 674–686.

[3] Anna L. Buczak and Erhan Guven. 2016. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys Tutorials* 18, 2 (Secondquarter 2016), 1153–1176.
[4] CESNET. 2016. Intrusion Detection Extensible Alert. Retrieved November 15, 2019 from https://idea.cesnet.cz/en/index.
[5] CESNET. 2017. Warden. Retrieved November 15, 2019 from https://warden.cesnet.cz/en/index.
[6] CESNET and Masaryk University. 2016. SABU. Retrieved November 15, 2019 from https://sabu.cesnet.cz/en/start.
[7] Frédéric Cuppens and Alexander Miège. 2002. Alert correlation in a cooperative intrusion detection framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*. 202–215.
[8] H. Debar, D. Curry, and B. Feinstein. 2007. The Intrusion Detection Message Exchange Format (IDMEF). RFC 4765 (Experimental). http://www.ietf.org/rfc/rfc4765.txt.
[9] Hervé Debar and Andreas Wespi. 2001. Aggregation and correlation of intrusion-detection alerts. In *Proceedings of the International Workshop on Recent Advances in Intrusion Detection*. Springer, 85–103.
[10] Huwaida Tagelsir Elshoush and Izzeldin Mohamed Osman. 2011. Alert correlation in collaborative intelligent intrusion detection systems—A survey. *Applied Soft Computing* 11, 7 (2011), 4349–4365.
[11] Mica R. Endsley. 1995. Toward a theory of situation awareness in dynamic systems. *Human Factors* 37, 1 (1995), 32–64.
[12] EsperTech Inc.2019. Esper. Retrieved November 15, 2019 from http://www.espertech.com/esper/.
[13] Hamid Farhadi, Maryam AmirHaeri, and Mohammad Khansari. 2011. Alert correlation and prediction using data mining and HMM. *ISeCure* 3, 2 (2011).
[14] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Antonio Gomariz, Ted Gueniche, Azadeh Soltani, Zhihong Deng, and Hoang Thanh Lam. 2016. The SPMF open-source data mining library version 2. In *Proceedings of the 19th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2016)*. Part III, Springer LNCS 9853, 36–40.
[15] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Rage Uday Kiran, Yun Sing Koh, and Rincy Thomas. 2017. A survey of sequential pattern mining. *Data Science and Pattern Recognition* 1, 1 (2017), 54–77.
[16] Philippe Fournier-Viger and Vincent S. Tseng. 2011. Mining top-k sequential rules. In *Proceedings of the International Conference on Advanced Data Mining and Applications*. Springer, 180–194.
[17] Julien Freudiger, Emiliano De Cristofaro, and Alejandro E. Brito. 2015. *Controlled Data Sharing for Collaborative Predictive Blacklisting*. Springer International Publishing, Cham, 327–349.
[18] Martin Husák and Jaroslav Kašpar. 2018. Towards predicting cyber attacks using information exchange and data mining. In *Proceedings of the 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*. 536–541.
[19] Martin Husák and Jaroslav Kašpar. 2019. AIDA framework: Real-time correlation and prediction of intrusion detection alerts. In *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES'19)*. ACM, New York, NY, Article 81, 8 pages.
[20] Martin Husák, Jaroslav Kašpar, Elias Bou-Harb, and Pavel Čeleda. 2017. On the sequential pattern and rule mining in the analysis of cyber security alerts. In *Proceedings of the 12th International Conference on Availability, Reliability and Security*. ACM, Reggio Calabria, 22:1–22:10.
[21] Martin Husák, Jaroslav Kašpar, and Milan Žiaran. 2019. AIDA Framework. Retrieved February 5, 2020 from https://github.com/CSIRT-MU/AIDA-Framework.
[22] Martin Husák, Jana Komárková, Elias Bou-Harb, and Pavel Čeleda. 2019. Survey of attack projection, prediction, and forecasting in cyber security. *IEEE Communications Surveys & Tutorials* 21, 1 (Firstquarter 2019), 640–660.
[23] Martin Husák, Milan Čermák, Martin Laštovička, and Jan Vykopal. 2017. Exchanging security events: Which and how many alerts can we aggregate? In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, Lisbon, 604–607.
[24] Martin Husák, Martin Žádník, Václav Bartoš, and Pavol Sokol. 2019. Dataset of intrusion detection alerts from a sharing platform. Retrieved November 15, 2019 from http://dx.doi.org/10.17632/p6tym3fghz.1 Mendeley Data, v1.
[25] Ci-Bin Jiang, I-Hsien Liu, Yao-Nien Chung, and Jung-Shian Li. 2016. Novel intrusion prediction mechanism based on honeypot log similarity. *International Journal of Network Management* 26, 3 (2016), 156–175.
[26] Yong-Ho Kim and Won Hyung Park. 2014. A study on cyber threat prediction based on intrusion detection event for APT attack detection. *Multimedia Tools and Applications* 71, 2 (2014), 685–698.
[27] Alexander Kott, Cliff Wang, and Robert F. Erbacher. 2015. *Cyber Defense and Situational Awareness*. Vol. 62. Springer.
[28] Jie Lei and Zhi tang Li. 2007. Using network attack graph to predict the future attacks. In *2nd International Conference on Communications and Networking in China, 2007. CHINACOM'07*. 403–407.
[29] Zhitang Li, Jie Lei, Li Wang, and Dong Li. 2007. A data mining approach to generating network attack graph for intrusion prediction. In *Proceedings of the 4th International Conference on Fuzzy Systems and Knowledge Discovery - Volume 04 (FSKD'07)*. IEEE Computer Society, Washington, DC, 307–311.
[30] Xiaobo Ma, Jiahong Zhu, Zhiyu Wan, Jing Tao, Xiaohong Guan, and Qinghua Zheng. 2010. Honeynet-based collaborative defense using improved highly predictive blacklisting algorithm. In *Proceedings of the 2010 8th World Congress on Intelligent Control and Automation*. 1283–1288.

[31] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. 1997. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery* 1, 3 (1997), 259–289.
[32] Luca Melis, Apostolos Pyrgelis, and Emiliano De Cristofaro. 2019. On collaborative predictive blacklisting. *ACM SIGCOMM Computer Communication Review* 48, 5 (2019), 9–20.
[33] Pantaleone Nespoli, Dimitrios Papamartzivanos, Félix Gómez Mármol, and Georgios Kambourakis. 2018. Optimal countermeasures selection against cyber attacks: A comprehensive survey on reaction frameworks. *IEEE Communications Surveys Tutorials* 20, 2 (Secondquarter 2018), 1361–1396.
[34] Ali Ahmadian Ramaki, Morteza Amini, and Reza Ebrahimi Atani. 2015. RTECA: Real time episode correlation algorithm for multi-step attack scenarios detection. *Computers & Security* 49 (2015), 206–219.
[35] Ali Ahmadian Ramaki and Reza Ebrahimi Atani. 2016. A survey of IT early warning systems: Architectures, challenges, and solutions. *Security and Communication Networks* 9, 17 (2016), 4751–4776.
[36] Ali Ahmadian Ramaki, Massoud Khosravi-Farmad, and Abbas Ghaemi Bafghi. 2015. Real time alert correlation and prediction using Bayesian networks. In *Proceedings of the 2015 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC)*. 98–103.
[37] Florian Skopik. 2018. *Collaborative Cyber Threat Intelligence: Detecting and Responding to Advanced Cyber Attacks at the National Level*. CRC Press.
[38] Fabio Soldo, Anh Le, and Athina Markopoulou. 2010. Predictive blacklisting as an implicit recommendation system. In *2010 Proceedings IEEE INFOCOM*. IEEE, 1–9.
[39] Kyle Soska and Nicolas Christin. 2014. Automatically detecting vulnerable websites before they turn malicious. In *USENIX Security Symposium*. 625–640.
[40] The Apache Software Foundation. 2017. Apache Kafka. Retrieved November 15, 2019 from https://kafka.apache.org/.
[41] The Apache Software Foundation. 2018. Apache Spark. Retrieved November 15, 2019 from https://spark.apache.org/.
[42] Fredrik Valeur, Giovanni Vigna, Christopher Kruegel, and Richard A. Kemmerer. 2004. Comprehensive approach to intrusion detection alert correlation. *IEEE Transactions on Dependable and Secure Computing* 1, 3 (July 2004), 146–169.
[43] Emmanouil Vasilomanolakis, Shankar Karuppayah, Max Mühlhäuser, and Mathias Fischer. 2015. Taxonomy and survey of collaborative intrusion detection. *ACM Computing Surveys* 47, 4, Article 55 (May 2015), 33 pages.
[44] Kalyan Veeramachaneni, Ignacio Arnaldo, Vamsi Korrapati, Constantinos Bassias, and Ke Li. 2016. AI^2: Training a big data machine to defend. In *Proceedings of the 2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*. 49–54.
[45] Jian Zhang, Phillip A. Porras, and Johannes Ullrich. 2008. Highly predictive blacklisting. In *USENIX Security Symposium*. 107–122.

Received November 2019; revised February 2020; accepted February 2020