Predictive Methods in Cyber Defense: Current Experience and Research Challenges

Martin Husák[a,*], Václav Bartoš[b], Pavol Sokol[c], Andrej Gajdoš[c]

[a] *Institute of Computer Science, Masaryk University, Czech Republic*
[b] *CESNET, Czech Republic*
[c] *Faculty of Science, Pavol Jozef Šafárik University in Košice, Slovakia*

Abstract

Predictive analysis allows next-generation cyber defense that is more proactive than current approaches based on intrusion detection. In this paper, we discuss various aspects of predictive methods in cyber defense and illustrate them on three examples of recent approaches. The first approach uses data mining to extract frequent attack scenarios and uses them to project ongoing cyberattacks. The second approach uses a dynamic network entity reputation score to predict malicious actors. The third approach uses time series analysis to forecast attack rates in the network. This paper presents a unique evaluation of the three distinct methods in a common environment of an intrusion detection alert sharing platform, which allows for a comparison of the approaches and illustrates the capabilities of predictive analysis for current and future research and cybersecurity operations. Our experiments show that all three methods achieved a sufficient technology readiness level for experimental deployment in an operational setting with promising accuracy and usability. Namely prediction and projection methods, despite their differences, are highly usable for predictive blacklisting, the first provides a more detailed output, and the second is more extensible. Network security situation forecasting is lightweight and displays very high accuracy, but does not provide details on predicted events.

Keywords: Cybersecurity, Prediction, Forecasting, Data Mining, Machine Learning, Time Series

1. Introduction

Cybersecurity is a continuously evolving field of research that is experiencing frequent shifts in approaches and paradigms. The cybersecurity community acknowledged the fact that cyber threats cannot be fully eliminated and, thus, the research and development are focused on prevention and lowering the impact of security incidents. However, most of the existing approaches are, in essence, reactive. The topics of intrusion detection and incident response were studied intensively in recent years with solid results. Still, they only react to events that already happened [1]. There is a tendency to move towards more proactive approaches [2] that would allow us to prevent or mitigate security incidents before they do any harm. The way for this is paved by cyber threat intelligence [3], cyber situational awareness [4], collaboration and information sharing [5], and other promising directions of research and development.

Methods of predictive analytics are a promising research direction in cybersecurity that would allow for a more proactive approach to security operations [2]. Predictions may serve as an early warning so that the defenders may learn about the threats in advance, set up proper countermeasures, and preemptively mitigate or completely prevent security incidents [6]. Numerous methods and approaches were proposed in the previous work with wildly varying goals and results [1]. The network-wide security situation, such as an increase or decrease of attacks, might be forecasted [7], particular incidents can be predicted by various means [8], and even when an attack is taking place, it is possible to predict the next actions the adversary is going to undertake [4].

Nevertheless, due to the complexity of the continuously changing cyber environment, there are still many challenges that need to be approached, many of them are rooted deeply in the foundations of this research direction. It is challenging even to define what is being predicted and how to use the predictions. Making predictions by using the data from the past is a common approach, which, however, does not reflect the novel forms of attacks and zero-day exploits that appear on a daily basis. Forecasting the increase or decrease in the number of attacks typically does not say much about the actors of those attacks; projecting the next move of an adversary does not say much about the threat landscape [1]. Thus, we find it important to find common ground, on which we may compare and analyze different approaches and methods to learn about strong and weak aspects of each approach. Further, there are huge differences in experimental works and field studies due to the operational environment's numerous problems, such as insufficient or erroneous data on the input, which results in a significant drop in prediction accuracy. This, together with poor evidence or explanation behind the predictive methods, namely in those backed by ma-

*Corresponding author, email: husakm@ics.muni.cz

chine learning, may lead to practitioners' reluctance to use them. Thus, it is also important to continuously evaluate technology readiness levels of the predictive methods and watch for the challenges related to their operational deployment.

To summarize the problems outlined above, we emphasize the following questions that we aim to answer:

1. What is the current state-of-the-art of predictive methods in the cybersecurity domain, such as predicting attacks, projecting the next move of an adversary, and forecasting cybersecurity situation?

2. What are the commonalities and differences between the methods, and how usable are their outputs concerning their use cases? For example, can they be used to mitigate an attack effectively or to prepare for an imminent threat?

3. How effective and efficient are these methods in practice, and how should the researchers learn from it?

To answer the questions, we examine selected examples of various predictive methods proposed in recent years. We selected a sample of substantially different methods that share the same or similar input data. Thus, we illustrate what outputs can various methods produce. Subsequently, we evaluate the outputs of the selected methods in terms of usability. We are especially interested in features such as the success rate of predictions, the timing of predictions, and the level of details that are crucial to the practical usability of the methods. For example, a highly usable predictive method would predict a type of event and its actors, such as IP addresses of attacker and victim, with high accuracy, while leaving enough time for the incident response capabilities to react. We also investigate the methods of predictive blacklisting, which utilizes recent research work while being highly convenient to practitioners.

This paper is structured into eight sections. After the introduction, we present the background and related work, structured by the use cases for predictive methods in cybersecurity, in Section 2. Subsequently, we delve into three examples of current approaches at using predictions in cyber defense. Section 3 briefly describes the environment for which the approaches were designed and in which they were evaluated. Section 4 presents an approach to attack projection based on extracting and matching frequent scenarios. Section 5 presents an approach to attack prediction based on a dynamic network entity reputation and scoring system. Section 6 presents an approach to forecasting based on the analysis of the time series of security events. The summary of each approach is provided at the end of the corresponding section. The final comparison and discussion are presented in Section 7. Finally, Section 8 concludes the paper and discusses the challenges for future research.

Use case	**Task description**
Attack projection (Section 4, [4, 9])	What is an adversary going to do next?
Attack prediction (Section 5, [8, 10])	What type of attack will occur, when, and where?
Network security situation forecasting (Section 6, [7, 11])	How is the overall security situation in the network going to evolve? How many attacks can we expect?

Table 1: Use case characteristics [1].

2. Background and Related Work

In this section, we summarize the state-of-the-art of predictions and forecasting in cybersecurity. First, we comment on the use cases for predictions and forecasting. Subsequently, we present the background and related work on the three main topics of interest: attack projection, attack prediction, and network security situation forecasting. Finally, we discuss the related work and overlaps with other research directions in cybersecurity.

2.1. Use Cases of Predictions in Cybersecurity

The recent survey [1] posed a list and characteristics of the use cases of predictive methods in cybersecurity. The use cases are summed up in Table 2.1. For each use case, we reference an explanatory research paper or a survey on the topic. The first use case is the attack projection [4], in which the problem is to predict the next move of an adversary in a running attack by projecting the series of actions the attacker performs. The projections are based on recent observations and are conducted in real-time. The second use case is the attack prediction [8], in which the problem is to predict what attacks are going to happen, when, and where. Contrary to the first use case, there are no traces of a running attack available; the predicted attacks are yet to start. This use case partially overlaps with risk assessment. Finally, the use case of network security situation forecasting [7] is related to cyber situational awareness. The task is not to predict a particular attack but rather to forecast the situation in the whole network. The outcomes may be a forecast of an increase or decrease in the number of attacks or vulnerabilities present in the network. The following subsections discuss the use cases in more detail.

2.2. Attack Projection and Intention Recognition

Attack projection [4] is the oldest use case for prediction in cybersecurity. It was proposed almost two decades ago [12], the first method started to appear around 2003 [13, 14], and the research still continues [15, 16]. To project the continuation of an attack and predict the upcoming events, we typically need to document the attackers' behavior and establish a description of an attack for later use. Many types of cyberattacks follow a sequence of events,

which can be observed either in the network traffic or on the target system, where intrusion detection systems may be found. For example, an attacker first scans a network horizontally to find active hosts, then scans an active host vertically to find open ports, brute-forces an authenticated service running there, and finally escalates privileges on the exploited system. This is only a generic scenario; specific sequences should contain details on each step, such as port numbers, the software version of the tools, CVE (Common Vulnerabilities and Exposures) number of an exploit, and an attacker's fingerprint. The attack projection is, in essence, very simple. If we observe a sequence of events that fit a known attack scenario, we may assume that the attack will continue in similarly to the scenario, and predict the attacker's next action. Nevertheless, a vague description of an attack is not usable for algorithmic predictions and, thus, a more formal description of an attack is required, e.g., in the form of an attack graph [13] or a probabilistic graph model [14]. Probabilistic models allow us to consider multiple concurrent goals of the attacker. Other possible problems, such as the need to work with unobserved actions or a failure to observe an attacker's action, can be approached by employing Hidden Markov Models [17]. There is also a need to create a model for all the attacks that are going to be projected. Historically, the first methods depended on attack libraries [14] that had to be manually filled, which requires substantial effort and continuous updates [4]. Modern methods more often rely on data mining to automatically extract attack scenarios [18, 17, 19].

2.3. Intrusion Prediction

A more general task is predicting cyberattacks, mostly intrusions [8]. Instead of projecting an already observed attack, we are interested in predicting novel attacks. Minor variations of the task also include predictions of vulnerabilities, attack propagation and multi-stage attacks, and other cybersecurity events. There is also a significant overlap with the research on early warning systems [6], which pose a practical use case for prediction in cybersecurity. Due to the task being too generic, there are a plethora of methods with not many common elements. For example, one may predict the attacks using the same models used for attack projection, such as attack graphs, with only a small variation in prediction start. The prediction may not start with an already observed series of malicious events, but rather with a probability that a particular vulnerability in the network will be exploited. Alternatively, a time series representing a number of attacks on a certain system or network in time may be used to predict a future attack. Advanced methods may calculate with types of attacks and characteristics of attackers and victims, so that they may estimate what type of attack is going to happen, who is going to be the attacker, and who is going to be the victim. Recent approaches often include non-technical data sources in the predictions so that we may see methods based on sentiment analysis on social networks [20, 21] or changes in user behavior [22], thus overcoming the "unpredictability" of cyberattacks.

2.4. Network Security Situation Forecasting

The last use case presented here is the forecasting of a global or network-wide security situation [23, 7]. Instead of focusing on an ongoing attack or a particular possible intrusion, the goal is to perceive a holistic state of an information system, network, or the whole Internet. A key concept of a holistic view on cybersecurity is often referenced as cyber situational awareness (CSA) or network security situational awareness (NSSA) that both originate in the general definition of situational awareness defined as: *"Perception of the elements in the environment within a volume of time and space, the comprehension of their meaning and the projection of their status in near future"* [24]. When applied in the cybersecurity field, perception corresponds to the monitoring of cyber systems and intrusion detection. Comprehension corresponds to the understanding of the cybersecurity situation, in our case, represented by modeling of cyber threats or correlating security alerts. Finally, projection is an action of predicting the changes in a cybersecurity situation [4].

Most of the works use quantitative analysis to describe the network security situation at a point in time. The resulting values are then projected into the future. Such an approach does not provide any information about the exact nature of future attacks but can supply warnings about an expected increase or decrease in malicious activity. The quantitative approach allows for efficient application of methods for analysis and projection that have been applied in other fields. The quantitative analysis requires a measure for the evaluation of a network security situation. However, no established canonical measure exists. There are two prevalent approaches: hierarchical method with additive weights and attack intensity estimation method. The hierarchical method evaluates the network security situation bottom up. Initially, a security situation is measured for each host. Subsequently, the values for each host are multiplied by the weight of the host and summed up to compute the overall security of the network. The actual method for estimating host security varies by author. The weight usually expresses the importance of the host. The attack intensity approach fuses information about the ongoing attacks from diverse sources and estimates an overall attack intensity. The overall intensity is derived from the number and severity of attacks against the whole network. The forecast can then give a warning about incoming increase or decrease in the number of cyberattacks [1].

2.5. Related Work

Two recent surveys covered the topics of prediction in cybersecurity in depth. Husák et al. [1] analyzed the use cases and provided a taxonomy and literature review of attack projection, prediction, and forecasting methods. Sun et al. [2] focused on data-driven incident prediction

methods and discussed the paradigm shift from reactive to proactive approaches to cyber defense. Particular areas of predictive approaches in cybersecurity were surveyed in other works as well. For example, Yang et al. [4] formalized the task of attack projection, surveyed literature on the topic, and listed three categories: prediction based on attack plans, estimates of attackers capabilities and intentions, and predictions by learning attack patterns and attacker's behavior. Wei and Jiang [23] analyzed the problem of network security situation forecasting and compared the methods based on neural networks, time series, and support vector machines, while Leau and Manickam [7] surveyed the methods with a theoretical background in machine learning, Markov models, and Grey theory. An overview of intrusion prediction methods was presented by Abdlhamed et al. [25], who split related work into two groups, prediction methods and intrusion detection enhancement. The prediction methods are categorized into three groups, methods using Hidden Markov models, methods based on Bayesian networks, and genetic algorithms. Subsequently, they classify artificial neural networks, data mining, and algorithmic methodologies as three enhancements for intrusion detection, which enhance the effectiveness of prediction systems. The same authors later published a survey of intrusion prediction [8], in which they categorized prediction methodologies and prediction systems. Prediction methodologies can be based on alert correlation, sequences of actions, statistical and probabilistic methods, and feature extraction. Prediction systems are then categorized as based on hidden Markov models, Bayesian networks, genetic algorithms, neural networks, data mining, and algorithmic methods.

The topic of predictions in cybersecurity is still growing. For example, Zhang et al. [16] proposed an IACF framework that further improves alert aggregation, correlation, and attack scenario discovery and prediction. Chen et al. [26] proposed a unique use case for predictions; the authors predict what security events would have been triggered by a security product if it had been present, allowing for adding more context to the security incident handling process.

As we can see in the examples above, predictive methods in cybersecurity are not an isolated research direction, but often overlap with others, such as with the cyber situational awareness [4, 23, 7] and intrusion detection [25, 8]. Attack intention recognition is a very closely related research direction that uses very similar methods and approaches (e.g., graph models) as attack projection. Ahmed and Zaman [15] surveyed the methods of attack intention recognition and recognized four categories: causal networks, path analysis, graphical models, and dynamic Bayesian networks. Methods based on causal networks were evaluated as the most effective. Gheyas and Abdallah [27] surveyed the detection and prediction of insider threats that often combine predictive methods with non-technical inputs, such as changes in user behavior, and offer an opportunity for interdisciplinary research. Finally, from the perspective of use cases for predictive approaches, Ramaki and Atani [6] surveyed early warning systems that often use predictive analytics.

3. Experimental Environment and Setup

Predictive methods in cybersecurity use various inputs, from network traffic records and log data to intrusion detection alerts. Therefore, it is hard to compare particular methods. In this paper, we discuss three methods that share a common data source, which makes them, at least to some extent, comparable. The common data source is SABU [28], a platform for sharing intrusion detection alerts between various organizations. In this section, we briefly introduce the SABU platform and the IDEA format [29], which is used to format and categorize the alerts. Subsequently, we present SABU-specific motivation to conduct research on predictive analyses with the SABU data. Finally, we briefly describe a dataset that was collected in the SABU platform and published for the evaluation of the proposed methods.

3.1. SABU Alert Sharing Platform

Herein, we present the SABU alert sharing platform [28], in which we evaluated several predictive methods. SABU is a platform operated by CESNET, Czech national research and education network (NREN), and used by its partners from academia, public sector, and commerce. The platform allows the partners to exchange alerts from various sources, such as intrusion detection systems, honeypots, and third-party providers. The alerts can be received by security teams (CSIRT, CERT) or by active network defense devices, such as firewalls that automatically add entities (e.g., IP addresses, domain names) 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 [30], a hub that receives the alerts from sending connectors (left) and distributes them to receiving connectors (right). A bi-directional connector iABU is an instance of AIDA [31], an analytical engine for alert processing that hosts attack projection method discussed in Section 4. NERD [32] is a network entity reputation database that also hosts the attack prediction method discussed in Section 5. The security situation forecasting component implements the method discussed in Section 6. Most parts of the platform are available as open-source, including Warden[1], AIDA[2], and NERD[3].

SABU platform uses Intrusion Detection Extensible Alert (IDEA) format [29] for exchanging information on security events. IDEA is inspired by IDMEF [33], but is

[1] `https://warden.cesnet.cz/en/downloads`
[2] `https://github.com/CSIRT-MU/AIDA-Framework`
[3] `https://github.com/CESNET/NERD`

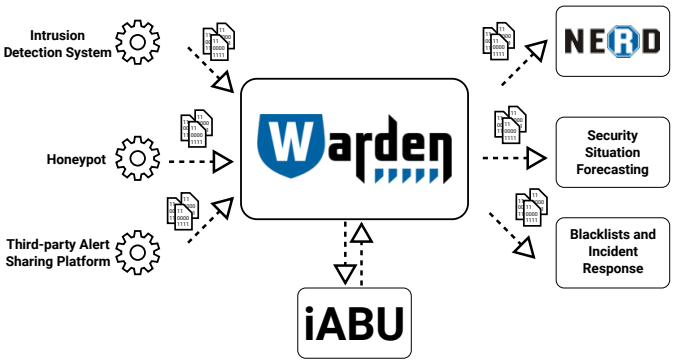

Figure 1: Schema of SABU alert sharing platform.

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 [29].

3.2. The Motivation for Predictions in SABU

The SABU platform is fully operational, and the data shared in it are used in cybersecurity operations and research alike, which provides an opportunity for researchers to evaluate their work in operational settings and practitioners to be in contact with current research, which also facilitates the exchange of ideas and experience that involves the use of predictive analytics.

The participants in the information sharing are interested the most in alerts directly related to their networks, such as malicious action executed by a host in the network of one peer detected in the network of another peer. Such alerts can be easily filtered and distributed, but only a small fraction of the data conforms to this. Most of the malicious actions come from entities in third-party networks and, thus, the peers are also interested in the use of the other alerts. A straightforward usage of the data is blacklisting, in which the network entities (e.g., IP addresses, domain names) are extracted from the alerts and put on a blacklist. There are two limitations to such blacklisting, reactive nature of the approach and large data volume. Predictive analytics may help to resolve both issues.

As mentioned in the introduction of this paper, a shift from reactive to proactive paradigm is welcome. In the blacklisting example, instead of creating blacklists of entities that did malicious actions in the past, we should create a blacklist of entities that are assumed to behave maliciously in the present or future. This way, the blacklist may prevent and mitigate attacks more effectively. Predictive analytics, namely attack prediction and projection, may help significantly in this effort.

The large data volume is another aspect that complicates the work with shared data. The alerts in the SABU platform are effectively Big Data; human operators are not capable of processing them all. The capacity of active network devices is also limited and, thus, it is not possible to, for example, convert the whole blacklists into firewall rules. It would be beneficial if the shared data or blacklists were significantly smaller in volume. Predictions may be used to distribute the data only to those recipients who may benefit from it or prioritize the data by a predicted probability of future attacks. For example, if we use attack projection, we may estimate the next victim of a running attack, and send the warning to the victim's network. When predicting attackers, we may prioritize the IP addresses that are most likely to attack in the future, and add them on top of the blacklist. A small blacklist based on projected or predicted security events might be more useful for the recipient than a large blacklist of entities for which we have no estimate if they are going to behave maliciously or not.

3.3. Dataset

The data shared within the SABU platform may be considered confidential and, thus, are not persistently stored, which disallows the reproducibility of the experiments done with such data. Thus, we used the dataset [34] of intrusion detection alerts collected from SABU [28] for one week, from March 11 to March 17, 2019. Almost 12 million alerts were collected from 34 intrusion detection systems (mostly network-based), 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 [29] and categorized using the taxonomy of security events included in the IDEA definition. The IP addresses and other network entities in the alerts are anonymized.

4. Attack Projection: Extraction and Recognition of Attack Sequences

The first method that we include in this paper is the attack projection method based on extracting and recognizing common attack sequences. This method is backed by data mining and complex event processing, and its implementation is publicly available [31]. Several research papers related to this method were already published. First, the selection of the most suitable data mining method was made [35], then the prediction use case was elaborated on [36], and finally, the design of the predictive system was published [31]. Readers interested in the background and details on this method are kindly referred to the research papers mentioned above.

4.1. Method Description

The method processes the alerts from intrusion detection systems and other sources and uses them to extract frequent patterns. The patterns are extracted in the form of predictive rules, i.e., rules that suggest a probable next item in a sequence. The rules are then converted into

search queries that are used to match the rules and predict the next item, i.e., the next malicious action. The list of steps goes as follows:

- Data Preprocessing
  - Input Sanitization
  - Alert Filtering
  - Alert Aggregation
- Data mining
  - Database construction
  - Data mining algorithm
- Rule matching (prediction)

The particular steps are discussed in detail in the remainder of this section.

The preprocessing phase consists of input sanitization, filtering, and aggregation of the alerts on the input. Most of these tasks are implementation-specific. The method processes only the alerts' key elements; the remainders of the alerts can be omitted. Further, not all the alerts are eligible for this task, and there are several scenarios in which there is a need to aggregate data on the input.

The input sanitization and filtering require the definition of the key elements. If the key elements are not present in the alert, the alert is dismissed. Further, if an alert is of a certain category, or does not fulfill other criteria, it is also dismissed. If all the conditions are met, the following 5-tuple is extracted from each alert:

$$(srcip, sensor, category, dstport, timestamp) \quad (1)$$

The *srcip* is the source IP associated with the security event, i.e., IP address of the attacker. The *sensor* field is a unique identifier of the alert's producer, such as an intrusion detection system or a honeypot. The *category* field is an identifier of the type of an event using a predefined taxonomy. Examples from the SABU platform [28] and IDEA format [29] are *Recon.Scanning* for network scanning activities or *Attempt.Login* for brute-force password attacks. The *dstport* stands for the destination port of the malicious activity, i.e., the port number on the victim side. Finally, the *timestamp* is the times of the start of an event or time of its detection, whichever is available.

Alert aggregation is an important step in data preprocessing because processing raw alerts without aggregation would skew the results of data mining. There are two procedures taking place in this step, aggregation of duplicate alerts and aggregation of persisting events [31]. The duplicate entries, i.e., the same alerts that appear in multiple copies, can be seen due to errors in sending and sharing. If multiple alerts share the same key elements (the 5-tuple presented above and the destination IP address), only one of them is accepted, and the others are dismissed. Persisting events cause stateless intrusion detection systems to report the same event multiple times, each with a different timestamp. 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 twelve alerts of the same event, each with a different timestamp. In such a case, only the first alert is processed, and the others are dismissed. Including all of them in the data mining would produce sequences describing the persisting events that are not useful in this use case [35].

The subsequent step is data mining that consists of two parts. First, the sequential database is constructed. Second, the data mining algorithm is executed. When the data mining is completed, the sequential database is deleted so that a new one could be constructed from the new data. In the implementation, this is done once a day [31]. The sequential database is filled with sequences of itemsets derived from alerts that share the same source IP address. An itemset is the n-tuple derived from an alert without the source IP address and the timestamp:

$$i = (sensor, category, dstport) \quad (2)$$

The itemsets are then grouped by the common source IP address and stored in the sequential database. For every unique IP address that appears in multiple alerts, a sequence of itemsets is inserted into the database and sorted by the timestamps. The database content consists of entries, such as:

$$\begin{aligned} &(i_a, i_b, i_c),\\ &(i_i, i_j),\\ &(i_i, i_j, i_k, i_l),\\ &(i_a, i_b, i_c),\\ &(i_a, i_b, i_d),\\ &(i_x, i_y, i_z), \end{aligned} \quad (3)$$

When the database is constructed, the data mining algorithm is executed. Top-K sequential rule mining method [37] was selected as the most suitable for the given use cases [35]. The implementation is using the algorithm implemented in the SPMF library [38] with $K = 10$ [31]. The method selects the ten most frequent sequential rules derived from the sequences in the sequential database. The sequential rule consists of two n-tuples of itemsets and support (s) and confidence (c) values:

$$((i_a, i_b, i_c, ...), (i_i, i_j, i_k, ...), s, c) \quad (4)$$

Support (s) stands for the frequency of the rule and is defined as the number of sequences corresponding to the rule divided by the total number of sequences in the sequential database. Confidence (c) is the percentage of sequences that started with the first part of the rule and continued with the second part of the rule. In terms of predictive analysis, 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 (3), the algorithm would produce rules such as:

$$((i_a, i_b), (i_c), 2/5, 0.6666) \quad (5)$$

The sequential rules are then used for prediction quite straightforwardly. If the rule matching engine detects a sequence of alerts with the same source IP address that is the same as the first part of a rule, then it is predicted that the alerts with the given source IP address and key elements corresponding to the itemsets in the second part of the rule will appear with the probability equal to the $conf$ value of the rule. In the implementation, this is done via complex event processing and conversion of the rules into SQL-like queries over the stream of alerts [31]. The predicted alerts could be seen as expected actions of the attacker behind the particular IP address and, thus, a projection of the attacker's activity.

4.2. Experimental Evaluation

The method and its implementation were evaluated in operational settings [36] and also with the use of the dataset [9]. In both cases, the alerts collected during one week were analyzed, with the data mining executed on data from each day separately. In rough numbers, around 10 million alerts were processed. Out of this number, around 13,000 alerts were dismissed in input sanitization and filtering due to not having any source IP address included (typical, for example, for alerts of phishing attacks) or being irrelevant for the use case, which is the case of alerts of discovered vulnerabilities. Further, we aggregated around 2,500 duplicated alerts (<1 %) and 830,000 persisting alerts (50 %).

The data mining produced ten sequential rules every day with confidence values ranging from 0.6 to 0.9. Most of the rules contained two itemsets in the first part of the rule and only one itemset in the second part of the rule. Further, the results were stable in time; most of the rules appeared in Top-10 several times or every day with similar values of support and confidence [36, 9].

Sequential rules mined on one day were used to predict alerts in the following day. Around 50,000 predictions were made for around 1.6 million alerts on the input on average each day. Around 65 % of the predictions were successful, although the success rate was significantly different for each rule, ranging even more than the confidence values, from 15 to 97 % [9]. The timing of the predictions and predicted alerts was another subject of evaluation. In both experiments, the difference in time, when the prediction was made, and the time, when the predicted alert appeared, was measured. This value represents the time, in which there is a chance to prevent the predicted event. While some predictions do not leave any time to respond (the predicted alert immediately follows the alert that led to the prediction), minimal time differences ranged from tens to hundreds of seconds. The average time difference between predictions and predicted alerts ranged from five minutes to two hours with outliers on both sides [36, 9].

4.3. Summary

The attack projection method and its implementation are aligned with the current research directions in the field, it is backed by unsupervised data mining, and the implementation uses modern stream-processing approaches. Similar methods can be found in related work [17, 19, 39].

The strong aspects of this method are the high accuracy and level of detail of the prediction. Although many methods proposed in related work achieve more than 90 % accuracy in artificial experiments, the accuracy of 65 % is actually pretty good in operational environment [1], where we have to deal with inaccurate, missing, or duplicated data, false positive and false negative alerts, and other aspects that skew the results. From a practical perspective, the predictions describe what is going to happen (category of the alert), where it is going to happen (given by the sensor that is likely to observe the event), and who is the malicious actor (source IP address). Further, the timing of the prediction often leaves enough time to react to the prediction and prevent the malicious event.

The weak aspects of this method are often the result of the "garbage in, garbage out" effect. The quality of the method is, in practice, dependent on the quality of the input data, even with cautious preprocessing. Often the predictive rules may be misleading, and manual inspection is advised if any actions are going to be taken in response to the predictions based on the rules. Another weak aspect is the limitation of operational deployment. Previous works often used long attack scenarios and sequences produced by a combination of network-based and host-based intrusion detection systems or crafted in advance [14, 18]. In practice, however, it is challenging to have a large number of highly accurate sensors of different types that would allow that. Therefore, there are mostly sequences of only three to four steps observed in the field studies [36, 9].

5. Attack Prediction: Network Entity Reputation and Scoring

The second method included in this paper is a specific kind of attack prediction. It does not predict exact attacks that are going to happen; instead, it estimates, for each IP address (or, potentially, another type of network identifier, an *entity*), the *probability* that it will attack in an immediate future time window (e.g., next 24 hours). It was designed primarily to summarize all known information about previous behavior of malicious entities into a numeric score representing the level of threat they currently pose. However, since the score is based on a prediction of future attacks, it can also be considered an attack prediction algorithm.

The score, called *Future Misbehavior Probability (FMP)*, can be used, for example, to sort the malicious IP addresses and create short but effective blacklists blocking the sources with the highest attack probability, or it can serve as one of the inputs in multi-criteria alert prioritization algorithms which help human operators to decide

which alerts are the most important. Full details of the method were presented in [10].

The FMP score may be general, predicting any kind of malicious behavior, or specific to a particular type of activity. For example, there can be an FMP score in the context of DDoS attacks and a different one in the context of port scans, each estimating probability of different types of behavior. It is also possible to compute the FMP score for specific targets, e.g., for specific networks or destination ports. In the variant shown in this paper, the model computes two FMP scores for each IP address, each predicting alerts of one of the following two categories – scanning activity (further referred to as *scan*) and attempts of unauthorized access, e.g., by dictionary attacks or exploit attempts (*access*). These categories were chosen since they are the most frequent ones in SABU data.

5.1. Method Description

The goal is to estimate, for each source (IP address), the probability that an alert of a specific type will be observed within the next 24 hours. This probability estimation is performed by a machine learning (ML) model trained on historical data. The particular model type used is Gradient Boosted Decision Trees (GBDT), although neural networks were also tested with very similar results [10]. The input of the model consists primarily of data about historical alerts. These are supplemented by additional data from external sources, such as the presence of the IP on public blacklists, its geographic location, or a heuristical guess whether it is statically or dynamically assigned IP address.

Formally, for this method, an alert is represented by tuple $a = (t, e, c, v, d)$, where items correspond to the following fields: detection time, source IP (entity), category, attack volume and detector name. The set of all available alerts is denoted by A. The time at which prediction is computed (*current time* or *prediction time*) is marked as t_0. The *prediction window*, T_p, is the time window of length w_p immediately following t_0, $T_p = (t_0, t_0 + w_p)$. The predictor uses information about the past alerts from *history window*, $T_h = (t_0 - w_h, t_0)$, where w_h is the history window length. For the evaluation below, we set $w_h = 7$ days, $w_p = 1$ day.

Since the machine learning model is not able to process a sequence of historical alerts directly, it has to be transformed into a fixed number of features first. So, for a given prediction time t_0, a feature vector $\mathbf{x}_{e,t_0} = (x_1, x_2, ..., x_k)_{e,t_0}$ is computed for each source (entity) e from the alerts reporting it as malicious and from the supplementary data. The particular set of input features contains:

1, 2 number of alerts in the last day and the last week,

3, 4 the total number of connection attempts (attack volume) in the last day and the last week,

5, 6 number of detectors reporting the address in the last day and the last week,

7 EWMA[4] of number of alerts per day over the previous week,

8 EWMA of the total number of connection attempts per day over the last week,

9 EWMA of a binary signal (0 or 1) expressing the presence of an alert in every day over the last week,

10 time from the last alert,

11, 12 mean and median intervals between alerts in the last week.

In order to leverage spatial correlations (i.e., the observation that malicious IP addresses are often close to each other in IP address space [40]), a similar set of features is also computed by taking into account all alerts related to the same /24 prefix as the evaluated IP address. This *prefix set* contains features 1–9 from the previous list and two new ones:

- number of distinct IP addresses in the prefix reported on the last day,
- number of distinct IP addresses in the prefix reported in the last week.

All the above-mentioned features are always computed from *scan* and *access* alerts separately, making a total of $2 \times (12 + 11) = 46$ features directly derived from alerts. Since there are significant correlations between the two categories, both subsets are used as inputs regardless of which alert category is predicted.

These alert-based features are supplemented by geolocation information – for each IP address, its country of origin is queried in geolocation database[5], and for each country the number of malicious IP addresses relative to its total IP address space is computed. The resulting number is used as an input feature for all IP addresses from a given country. An analogous procedure is performed with autonomous system numbers (ASN) instead of countries to get another input feature.

Finally, ten binary features are added by querying the IP address on several public blacklists and looking at its associated hostname. In total, each IP address is represented by a vector of 58 features.

The output (class label) to be predicted, y_{e,t_0}, is binary. It depicts whether or not there is an alert reporting the IP address within the prediction time window:

$$y^c_{e,t_0} = \begin{cases} 1 & \text{if } \exists\ a \in A : a = (t, e, c, \cdot, \cdot), t \in T_p \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where c is the alert category (*scan*/*access*) to be predicted. Samples with $y_{e,t_0} = 1$ are deemed to belong to *positive class*, others form the *negative class*.

[4] Exponentially weighted moving average with $\alpha = 0.25$.

[5] GeoLite2 database from MaxMind, `http://www.maxmind.com/`

The ML task is to create an estimator which, for a given feature vector $\mathbf{x}_{e,t_0}$, is able to estimate the probability that $y^c_{e,t_0} = 1$, i.e., that the IP address will be reported as malicious in the prediction window. This is known as *binary class probability estimation*. Output of the estimator is denoted as $\hat{y}_{e,t_0}$ and denotes the estimated probability of the positive class for the feature vector,

$$\hat{y}^c_{e,t_0} \approx p(y^c_{e,t_0} = 1|\mathbf{x}_{e,t_0}). \tag{7}$$

To train the ML model, a sample of historical data from at least a few weeks is taken, several "prediction times" t_0 within this period are selected, and a set of training vectors $(\mathbf{x}_{e,t_0}, y^c_{e,t_0})$ is computed, one for each IP address reported as malicious within a history window preceding each t_0. Non-linear scaling of some features and other transformations are then performed on the data (see Sec. 5.2.3 in [10] for details) and, finally, the samples are passed to the ML model to train (in fact, two sets of samples are used, and two separate models are trained in our case, one for each attack category).

A metric commonly used to evaluate probability estimation models is the Brier score (BS). For our binary case, with classes labeled 0 and 1, the BS can be described as a mean squared difference of the predicted probability of the positive class and value of the real class:

$$BS = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \tag{8}$$

where N is the number of samples. The BS can have values between 0 and 1, with lower values being better.

After the model is trained and its performance (checked on a separate testing dataset) is satisfactory, it can be used to assign FMP scores to new IP addresses in practice. First, a feature vector $\mathbf{x}_{e,t_0}$ representing the previous alerts and other information related to the IP address is computed. Then it is passed to the trained model, and its output value, $\hat{y}^c_{e,t_0}$, is directly used as the FMP score (for attack category c) of the IP address.

5.2. Experimental Evaluation

The method was evaluated on a dataset consisting of three months of data from the SABU platform [28] from September to November 2017. Note that the date range is different from the one used in the other two sections. That is because this method needs much more data than what is available in the dataset [34] (one week), so we had to reuse the results of experiments made on an older dataset. Nevertheless, no significant changes were made to the platform and its data sources between the two date ranges, so the characteristics of the data are very similar. The method was also implemented and tested in operational settings in the NERD system [32], although currently only in a simplified form using just some of the input features. Here we briefly present results of the evaluation on the dataset borrowed from [10].

As mentioned above, a separate prediction model is created for each of the two broad groups of alerts – scanning activity and attempts of unauthorized access. Other attack types, e.g., DDoS attacks, are too infrequent in the dataset (less than 1 %), so there are not enough samples to build an accurate prediction model using this method.

Training samples were computed at 24 different points in time within the three months, always for all the IP addresses deemed as malicious in the preceding week. Thus a total of 12.3 million samples related to *scan* alerts (the *scan dataset*) and 765,000 samples related to *access* alerts (the *access dataset*) were obtained. A random subset of samples was put aside from each dataset as testing data (600,000 samples in scan dataset, 100,000 samples in access dataset). The rest was used to train the ML model.

The best performing models were based on gradient boosted decision trees (GBDT) with 200 trees, each having the depth of up to 7. Their Brier scores on the testing datasets were 0.0628 (*scan* data) and 0.0507 (*access* data). Both values are close to zero, which means the model output approximates the real probability of observing a future alert very well. For detailed results, including a comparison of different ML models or using various subsets of the input features only, please refer to [10].

One of the possible use cases for the attack probability prediction represented by the FMP score is the creation of highly efficient predictive blacklists. Consider a case when a network operator wants to block traffic from known malicious IP addresses, but due to technical limitations (e.g., a limit on the number of firewall rules), only a small subset of such addresses can be blocked. Obviously, the optimal way is to block those addresses with the highest probability of attacking in the near future. Since the FMP score approximates exactly this probability, the blacklist can be created by simply taking top-N IP addresses with the highest score.

Effectivity of such blacklists was evaluated by computing FMP scores (for *access* data only) of all addresses a few days after the training period, creating top-N blacklists of different sizes and checking how many of the blacklisted addresses were actually reported as attacking within the next day. For comparison, a set of blacklists with the same size was created in a traditional way by simply listing the top-N addresses with the highest number of reported attacks in the previous week.

The results are shown in Table 2. N is the number of entries in the blacklist, *hit count* is the number of entries that were actually useful, i.e., the blacklisted address indeed attacked on that date and would be blocked by the blacklist. The *hit rate* is simply hit count divided by N and represents the efficiency of the blacklist. We can see that in all cases, the FMP based blacklists were much more efficient.

The FMP blacklists were also compared to several real public blacklists. For example, the SSH blacklist from *blocklist.de*, which is focused on similar types of attacks (login attempts), can block 336 attackers on the evaluated

N	blacklist type	hit count	hit rate
100	FMP	100	100 %
	traditional	71	71 %
500	FMP	443	89 %
	traditional	233	47 %
2000	FMP	862	43 %
	traditional	579	29 %

Table 2: Comparison of efficiency of FMP-based and traditional blacklists of different sizes [10]

day. However, with 8063 entries, its hit rate (efficiency) is only 4.2 %, much less than that of FMP blacklists with comparable hit count.

5.3. Summary

The presented attack prediction method provides a score for each malicious entity, which summarizes its reputation into a number expressing the probability of future attacks. Even though it cannot predict exactly whether an attack will occur or not, nor its detailed characteristics, knowledge of the probability can still be very useful. It was shown to be especially suitable for predictive blacklisting, as suggested by experiment results, but can also be used as easily comprehensible information for a human operator investigating a suspicious IP address.

The main advantages of the method include the ease of use for the creation of predictive blacklists as well as clear interpretability of the resulting FMP score. Another advantage in comparison to other methods is the ability to evaluate *any* IP address, regardless of the number of previous alerts or matching any patterns. The method is also easily adaptable to different needs or different availability of input data. Many variants can be created by narrowing down which attacks are predicted so that probabilities are computed separately for specific attack types, target networks, or services. Finally, it is straightforward to add any additional input features that may be available, which can improve the estimation accuracy. Simultaneously, as further experiments show [10], it works reasonably well with alert data only, without any supplementary features.

The most important disadvantage of this method is that it does not provide any details about the anticipated attacks, such as their targets, volume, or approximate time. While this can be partially mitigated by estimating multiple FMP scores not only for separate attack types, but also for different targets, time ranges, or combinations of multiple such characteristics, it is only feasible to do it for a small number of such categories, since there must be enough training samples for each of them. Generally, the need for a large amount of training data can be considered as another disadvantage of this method. As experiments show, at least several hundreds of thousands (ideally millions) of samples of given attack category (or target, or combination of both) are needed to get a good prediction performance, and they should span at least a few weeks in time. The next issue is that the method, in fact, predicts future alerts, not attacks, so potential false-negative (undetected attacks) or false-positive results (false alerts) of the detectors reporting alerts can affect the results. Theoretically, the probability estimate can be corrected if false-positive and false-negative rates of the detectors are known, but this has never been implemented and evaluated since these rates are generally unknown in practice. Another problem may arise if the method would be used to evaluate IP addresses that have not been detected as malicious (yet), which is theoretically possible and may lead to non-zero FMP scores thanks to the "prefix" features and other data not derived from previous alerts of the IP. In such a case, one can run into legal issues, since such activity can be seen as "profiling," and performing some actions, such as blocking traffic, based on it might be illegal in some jurisdictions, such as EU's GDPR [41].

6. Forecasting: Time Series Analysis

The last method we include in this paper is the attack forecasting based on time series. This subsection is based on the results of our previous research [11]. Time series models "attempt to make use of the time dependent structure present in a set of observations" [42]. The appropriate forecasting methods depend largely on what type of data are available. We have the choice of either qualitative forecasting methods (in cases when available data are not relevant to the forecasts) or quantitative forecasting methods.

Quantitative forecasting methods can be applied if two conditions are satisfied [43]:

- there is numerical information about the past – we have available data from Warden and SABU system,
- it is reasonable to assume that some aspects of the past patterns will continue – detecting security attacks and alerting is an ongoing activity.

The outcome of quantitative forecasting methods using the dataset mentioned above is the number of security alerts at a particular point in time.

6.1. Method Description

There is a wide range of quantitative forecasting methods, and their usage often depends on the specific disciplines or specific purposes. For choosing a particular method, properties, accuracies, and costs must be considered. In our research, we have considered four approaches to time series forecasting:

- ARIMA models,
- exponential smoothing models (state space models),
- the naive approach,

- combination (average) of ARIMA and exponential smoothing models.

ARIMA and exponential smoothing (ETS) models are one of the most widespread approaches to time series forecasting [43]. Moreover, we have compared these wide-used forecasting methods with the naive method [44, 45] which is a computationally undemanding approach and can handle large data. For our research, it is a certain basis for predictions. We have also added a combination (average) of ARIMA and ETS methods to compare standard methods with their combination. The idea of averaging or boosting is very popular in machine learning nowadays [1].

The models from the ETS family are characterized by the predictions being weighted combinations of previous observations with newer ones, which have a relatively higher weight compared to older observations. Exponential equalization reflects the fact that weights decrease exponentially with aging observations [43, 44].

The ARIMA models represent a generalization of the class of ARMA models that incorporate a wide range of non-stationary series. These models, by finite number of differentiations, ensure time series stationarity, allowing the use of ARMA models. ARMA models are a combination of auto-regression (AR) and moving average (MA) [45]. ARIMA models provide another approach to time series forecasting. While ETS models are based on a description of the trend and seasonality in the data, ARIMA models aim to describe the autocorrelations in the data [43]. Both classes of models can reflect seasonality in the data. ARIMA model can be expressed by the formula:

$$y_t^{(d)} = c + \sum_{i=1}^{p} \phi_i y_{ti}^{(d)} + \sum_{j=1}^{q} \theta_j \varepsilon_{tj} + \varepsilon_t \qquad (9)$$

where y is the series differentiated d times, c is the constant, p is the degree of autoregression, ϕ_i are the coefficients of autoregression, q is the degree of moving averages, θ_J are the coefficients of moving averages and ε_t are errors.

6.2. Experimental Evaluation

The forecasting methods were evaluated using a dataset from the SABU platform [34]. We focused on the total number of alerts and did not address the specific types of alerts, such as network scanning or login attempts. Given the fact that the dataset consists of data for one week, we consider 24 hours period and four different time units (5 minutes, 15 minutes, 30 minutes, and 60 minutes). According to this time unit, we created four separate time series for the total number of alerts (see Table 3).

We evaluated the method using the implementation presented in our previous work [11]. In our extensive calculations, we used R functions from one of the most widespread R packages intended for time-series predictions called *forecast* [46]. Beneficial functions when working with big data sets or potentially in real-time predictions are the ones for the automatic fitting of ARIMA and ETS.

Time series	Time interval	Number of units	Period
1.	5 min.	1729	192
2.	15 min.	577	96
3.	30 min.	289	48
4.	60 min.	145	24

Table 3: Description of time series. The 1st column represents time series, the 2nd column represents time interval, the 3rd column represents number of all time units in time series, and the 4th column represents number of time unit in one period (24 hours).

These functions are designed to automatically choose the best model from the particular class, considering, for example, the information criteria [46].

All the above-mentioned types of models were evaluated and compared according to several criteria based on the quality of predictions. For each class of models, particular models were fitted in the seasonal and non-seasonal settings. The addition of this classification is intended to help determine whether seasonality needs to be incorporated into forecasting models. If this is not confirmed, it simplifies the usage of these models and their automation itself (e.g., in the case of in case of the real-time forecasting). We considered one, two, five, and ten steps ahead predictions, which were compared to true values included in the test set.

Moreover, we considered two cases of model fitting. The first one was the "classical" one; when we kept the whole training data set and step by step, we added one more observation from the test set to the training set in each round of evaluation. Since the practice requires the forecasting methods to be time-saving and it also requires the possibility of running (forecasting) in real-time, in the second case, we consider the situation where we take into account not the whole training set but just a part of it. For this reason, In the second case, we used the so-called "rolling window" or "one in, one out," which means that in each round of evaluation we remove the oldest observation from the training set and at the same time we add one new observation from the test set to the training set.

At first, we calculated 95 % (bootstrap - because the normality in data is not satisfied) prediction intervals and the average length of these prediction intervals. The narrower the intervals, the more precise information they provide. We also took a look at the average coverage yield by these prediction intervals, and it should be approximately 95 %. If the coverage is lower than 95 %, the intervals are not very precise (maybe biased), and if the coverage is higher than 95 %, the intervals might be uselessly too wide.

Table 4 shows the results of the average coverage by 95 % prediction intervals. As a part of the evaluation, we tested all the methods on four data sets (with time-series observations in different time intervals (5 min., 15 min., 30 min., and 60 min.) and we compared the introduced forecasting approaches on the basis of different prediction steps – the next one, the 2nd one, the 5th one, and the 10th

Forecast	5 min.	15 min.	30 min.	60 min.
One-step	96.53 % A,As,AE, AEsw	97.39 % E,Aew, Asw	94.83 % All-Ew	96.55 % All
Two-steps	96.52 % Asw	97.37 % Nw	95.61 % All-Ns	98.21 % All
Five-steps	96.61 % Aw	98.20 % Asw	95.19 % Ew Esw	99.2 % All
Ten-steps	96.91 % Asw	98.49 % Asw	96.94 % Ew	99.5 % All

Table 4: Average percentage coverage of actual (future) time series values by 95 % prediction intervals. Notes: A – ARIMA model; E – exponential smoothing (state space models); N – naive model; AE – ARIMA + exponential smoothing (average); ALL – all models; (-) – without; s – with seasonality; w – rolling window.

Forecast	5 min.	15 min.	30 min.	60 min.
One-step	0,576 Aw, Aws	0,667 A,E,AE, A,E,AE(s)	0,727 AEw, AEsw	0,832 Ew, Esw
Two-steps	0,574 A, As	0,667 A,E,AE, A,E,AE(s)	0,677 Ew, Esw	0,790 Ns
Five-steps	0,586 A As	0,659 A,E,AE A,E,AE(s)	0,736 Ew, Esw	1,102 Ew
Ten-steps	0,586 A As	0,698 A,E,AE A,E,AE(s)	0,830 Ew, Esw	1,441 Ew, Esw

Table 5: Results of the MASE values for forecasting methods. Notes: A – ARIMA model; E – exponential smoothing (state space models); N – naive model; AE – ARIMA + exponential smoothing (average); s – with seasonality; (s) – with seasonality for each model in cell; w – rolling window.

step ahead. The values that approach 95 % indicate that the 95 % prediction interval has been correctly selected for a specific case (method, number of forecast steps, and time interval). If the value of coverage is lower than 95 %), it means that the prediction interval is less reliable. Conversely, the higher the value of coverage (more than 95 %) implies that the confidence interval is broader and less informative. As we can see from the table, a reasonably chosen time interval seems to be 30 minutes, and the ETS method with seasonality and rolling window.

Another approach to evaluate the predictions' quality of particular models is the so-called cross-validation [43]. We employed two variations of this approach. In the first case, we calculated the predictions at specific future times (separately for one or two or five or ten steps ahead) over the whole test set, and at the end, we calculated the metric value from all predictions for different prediction steps separately. A metric commonly used to evaluate the accuracy of forecasts is Mean absolute scaled error (MASE) [47]. MASE is defined as [43]:

$$\mathrm{MASE} = \mathrm{mean}(|q_j|) \tag{10}$$

where, for non-seasonal time series, q_j is:

$$q_j = \frac{e_j}{\frac{1}{T-1}\sum_{i=2}^{T}|y_i - y_{i-1}|}, \tag{11}$$

and, for seasonal time series, q_j is:

$$q_j = \frac{e_j}{\frac{1}{T-m}\sum_{i=m+1}^{T}|y_i - y_{i-m}|} \tag{12}$$

In both cases, e_j is forecast error, i.e., the difference between an observed value and its forecast, y_j represents observed value, T is the length of time series, and m is seasonality parameter (period).

In the second type of cross-validation, we considered all predictions up to the 2^{nd}, 5^{th}, and 10^{th} step ahead forecast in each round of evaluation we calculate MASE (not just the last one as in the previous case). Table 5 shows the results of the average MASE values for the selected four types of time intervals (of time series observations) and the MASE of point predictions of the 2^{nd}, 5^{th}, and 10^{th} step calculated over the whole training set. According to this criterion, the prediction is more accurate, when the lower value (ideally below 1) of MASE is achieved. A scaled error is less than one if it arises from a better forecast than the average naive forecast computed on the training data. Conversely, it is greater than one if the forecast is worse than the average naive forecast computed on the training data [48]. As can be seen from this table, except for two cases, all values were below 1. ARIMA model in 5 minute time interval shows the best values, but in the case of larger intervals, the use of the ETS model with a rolling window is better. It is also interesting to read from the table that the methods that used the rolling window have comparable results to the methods without using this approach. This fact is also demonstrated in Figures 2 and 3 and suggests that older values do not significantly affect on the prediction using selected methods (e.g., exponential equalization assigns low weights to small observations – low importance). For this reason, a reasonable sliding window will suffice. This result is important for the usage of forecasting based on time series without the need for increasing memory requirements (e.g., in case of the real-time forecasting).

Moreover, during the computations on test data set together with models fitting, we verified the suitability of fitted models by testing the residuals. In the case of testing the independence of residuals, we employed the Box–Pierce and Ljung–Box tests [49].

When it comes to ARIMA models and the combinations of ARIMA and ETS models, almost 100 % of the fitted ARIMA and combined (ARIMA and ETS) models passed the independence test of residuals. If we take a look at the ETS models separately, 73 %–100 % of models

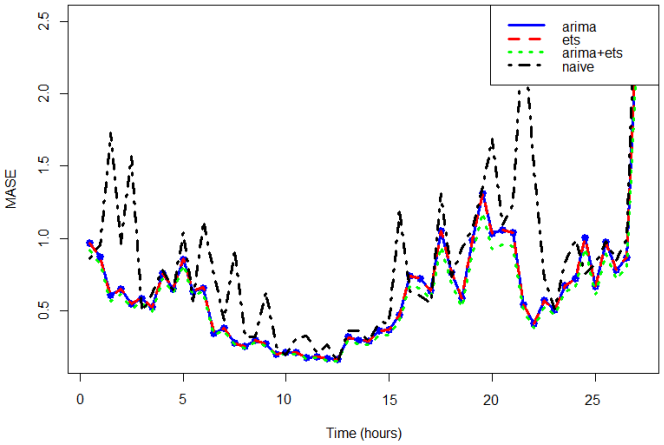

Figure 2: The accuracy of forecasting methods according to MASE criterion in the 5-step forecast measurements at thirty-minute intervals.

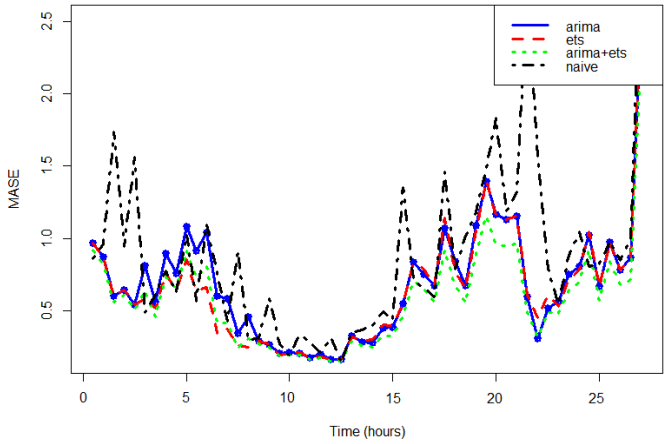

Figure 3: The accuracy of forecasting methods according to MASE criterion in the 5-step forecast measurements at thirty-minute intervals and rolling window.

passed the same criteria. It is quite enough. Employing the Shapiro-Wilk test [50], we also tested the normality of residuals, but here the situation is different: 0 %–50 % successfully passed this test. As a consequence, we had to compute bootstrap prediction intervals [51, 52, 53] instead of approximate normal prediction intervals.

An example of usage of the forecasting method based on time series is shown in Figure 4, where we can see 10-step forecasting based ARIMA method measurements at thirty-minute intervals.

6.3. Summary

Time series constructed from historical records, such as intrusion detection alerts, pose an interesting method of forecasting cyberattacks. They can be used to make long-term and short-term predictions of an increase or decrease in the number of cyberattacks and alerts.

These methods offer several advantages. For example, the time-series data are easy to obtain and process. Only the timestamps or counts (measurements) in particular time points are needed. As the next advantage, we pose the fact that there is a plethora of ready-to-use implementations [45, 44] that provide autofit of the models and their diagnostics. Moreover, these methods can achieve quite high precision in practice, as was shown in the evaluation. When it comes to technical or computational issues, time series forecasting approaches are fast and stable with respect to memory requirements. As we have shown, another good point is the observation that we do not need to keep the whole training data set and just keep adding new time series values, but instead, we can employ the so-called rolling window of some appropriate size and consequently we can reduce the memory usage and make the calculations faster. Therefore the introduced methods seem to be useful in real-time predictions.

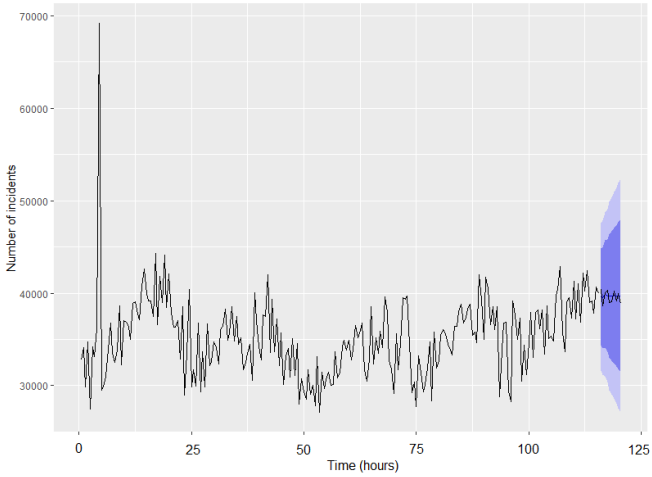

Figure 4: The 10-step forecast based ARIMA method measurements at thirty-minute intervals and rolling window.

The usage of time series forecasting approaches also has its disadvantages. For example, the time series does not provide a very detailed prediction. It is usually one or more numbers in case of point prediction and a prediction interval with particular confidence in the second case. This can be partially compensated by creating multiple time series characterizing the same development within a particular network and consequently using multidimensional time series. The next issue is that some models can not handle non-stationary data, and some preprocessing must be done (for example, the usage of the mathematical transformation of data). It is not our case here, but also a serious problem can be the occurrence of many small counts (less than one hundred) or zeros. In such a situation, the different methodology is needed, for example, the so-called INAR models [43] or some other approaches developed to

deal with integer time series. However, many of these models seem not to have any good implementation, or the implementation is missing completely. Tthis problem is also evident in the models we used. For our purposes, more clever algorithms, functions, and other options have been implemented in R than in any other open source programming language or software. Perhaps it is just a matter of time while these computational tools spread into other software packages.

7. Comparison and Discussion

In this section, we compare the three methods presented earlier and discuss their usability, effectiveness, and efficiency in the common environment. We want to highlight the different ways in which each method benefits security operations. The most important features of the three methods, along with a brief recapitulation of their strong and weak aspects, are presented in Table 7.

The three methods, despite their differences, share a few common features. All the three methods are prone to the "garbage-in, garbage-out" effect. If the input data are insufficient in volume, variety, and quality, we may not expect trustworthy outcomes. For example, having mostly alerts of network-based events, such as scans, on the input is sufficient to forecast scanning activity in the future, but not proper attack projection due to the lack of information from host-based sensors. This is one of the issues we faced in the evaluation in the operational setting. False-positive and false-negative detections are a well-known issue in the field [1]. False-positive alerts may lead the attack projection and prediction methods to false-positive predictions, while false-negative detections are problematic for network security situation forecasting.

From a research perspective, there are interesting differences in the reproducibility of the experiments. We are not aware of any prior work that would compare several predictive methods for cybersecurity in an operational setting. Although datasets were often used to evaluate methods proposed in previous works, the datasets are often old and unreliable, the new datasets were not used often yet, and the datasets were mostly created to evaluate methods of intrusion detection, not prediction, and, thus, substantial features can be missing [1]. Another problem is the feasibility of evaluation using datasets. It is certainly possible for short-term predictions and forecasts, such as in the attack projection, but not for long-term forecasts and predictions based on long-term observations. Such methods require large volumes of data from long time frames on the input, which is impractical for experimental evaluation.

Predictive blacklisting is a promising application of predictive methods, namely, attack projection and attack prediction. If we extracted all the network entities, such as IP addresses, from the input data, we might use them to create a blacklist. However, such a blacklist would be too large to be effectively used, and its hit rate, i.e., the percentage of entries that would be used for successful attack mitigation, would be lower. However, this could be significantly improved by creating a blacklist of entities, for which we predict a malicious action [9, 10]. The results show that we may achieve a very high hit rate with a few hundreds of entries in the blacklist (see Table 2) [10] or generate a predictive blacklist of only 3 % volume of the blacklist based on sources data, but with 65 % hit rate [9]. Predictive blacklisting can be seen as the most perspective use of predictive methods in cybersecurity. In practice, a predictive blacklist may be used as any other blacklist, but with a significantly increased effectiveness.

An interesting non-technical issue is a risk associated with processing private information. There is a risk in profiling [41]. The risk may be a concern for the first two methods; the third method is not processing network entities. The attack projection method works solely with network entities that were recently detected as malicious and, thus, it is backed by some evidence of malicious activity. However, reputation scoring in the attack prediction methods poses a high risk of profiling when including auxiliary input data. It is theoretically possible, although not implemented, to assign a reputation score to a network entity that was not detected as malicious in the past. For example, if it shares similar characteristics with malicious actors. Limiting access to the defended system from such a network entity would cross the law [41].

Finally, it is worth mentioning the dependency of the discussed methods on the SABU platform, where the experiments were executed, and from which the data were collected. Naturally, the implementations of the predictive methods used technologies and data formats from the SABU platform, such as Warden communication channels and the IDEA format. Further, the data in the platform are imbalanced due to the reliance on network-based intrusion detection systems and honeypots that report different types of alerts than host-based intrusion detection systems and other possible sources of data. However, the methods are oblivious to the content of the alerts and would work equally if other technologies and formats were used or if a composition of the alert types was different.

8. Conclusion

In this paper, we evaluated the capabilities of predictive methods in cybersecurity. We summarized use cases and state-of-the-art methods, selected three methods as representatives of the use cases, and compared them in a common environment of the SABU alert sharing platform [28]. The methods of attack projection [31], attack prediction [10], and network security situation forecasting [11] were all adapted to the common environment, which facilitated the comparison of the approaches and highlighted the differences between the outputs of the methods and ways in which the predictive methods benefit the cyber defense.

Method	Output	Parameters	Strong aspects	Weak aspects
Attack projection	Alert of the expected next action of an adversary.	Accuracy around 65 %, events are predicted from a few seconds to several hours in advance.	+ Very detailed and on-time prediction. + Reasonably high accuracy in practice. + Suitable for predictive blacklisting. + No issues with profiling, backed by recent evidence.	– Timing information is not part of the rule mining; timing needs to be estimated afterward. – Time to react may be too short. – Some rules might not be usable, manual inspection is advised. – Attack sequences are too short in practice.
Attack prediction	Estimated probability that a network entity will behave maliciously within a given time range.	Estimation closely approximates true probability (Brier score[6] $\leq$ 0.063). Precision of top-N blacklists depends on their size, 89 % for $N = 500$, 100 % for $N = 100$.	+ Suitable for predictive and highly efficient blacklisting. + Easily adaptable – slight variations may predict probability per event category, target network, target service, etc. + Easily extensible when new input features are available; works reasonably well even without the supplementary data.	– No information on the attack, only a probability that there will be some. – Can only calculate probabilities for known malicious entities; cannot predict new ones unless risking issues with profiling. – Large amounts of data and a longer collection period are required, complicated evaluation with datasets.
Network security situation forecasting	The number of alerts of adversaries' actions.	Accuracy of prediction from 0.574 to 1.441 according to MASE criteria, accuracy of 95 % (bootstrap) prediction intervals from 94.83 % to 99.5 %, point prediction and prediction intervals depending on parameters.	+ Very high accuracy. + Many methods of time series analysis and their implementations are available. + Seasonality taken in to account. + Low and stable memory and time complexity. + No issues with profiling.	– Not a very detailed prediction – no information on the attacks, only their expected number. – Possible problems with some type of data (e.g., significant number of zero values). – Possible problems with implementations of the methods in particular programming languages or libraries (e.g., missing auto-fitting feature, complicated metrics usage and evaluation).

Table 6: Comparison of predictive methods.

We stated three questions in the introduction that we aim to answer here. In the first question, we were interested in the current state of predictive methods in cybersecurity. As shown in this paper, the approaches proposed in previous works achieved sufficient technology readiness levels to be implemented and deployed in an operational environment, although still in more of an experimental and evaluation mode than as an operational service.

In the second question, we were interested in the outputs of the three evaluated methods. Attack projection and prediction methods were showed to be highly suitable for predictive blacklisting, which provides valuable input for existing cyber defense capabilities. The third method, network security situation forecasting, is helpful in estimating the number of attacks in the near future, which may be used to optimize cybersecurity operations. However, the methods do not provide any specific suggestions; it is up to the operators to react to the results.

In the third question, we were interested in the effectiveness and efficiency of the methods in practice and possible pitfalls. Namely, the blacklists consisting of network entities that are predicted to behave maliciously achieve very high hit rates. Further, the data volume in a predictive blacklist is substantially smaller than in the blacklist based on all past observations. However, all three methods face problems with the quality of the data on the input, which might skew the outputs and decrease their usability.

We believe there are many opportunities for future work. First, there is a need to address the data quality issues, not only for predictive methods but also for other uses in cybersecurity. A practical insight into this is that the variability and heterogeneity of the input data are not satisfactory in an operational environment. For example, our testing environment is heavily biased towards network-based intrusion detection and does not provide many opportunities to process alerts from host-based intrusion detection systems. Second, the reproducibility of research and experiments remains an open problem. Although we were able to compare three different approaches in a common operational setting, some methods are difficult to evaluate in a controlled manner, such as with a dataset. Third, the human factor is likely to become an important issue in future work if the predictive methods are used as a decision support system or will interact with

[6] Mean squared difference of the predicted probability of the positive class and value of the real class (0 or 1), see Eq. 8.

cybersecurity operators. Namely, human decision-making may play a role as a potential source of errors in the predictions' use and interpretation.

Acknowledgment

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, by the Slovak Research and Development Agency under the contract No. APVV-17-0568 (Application of mathematical methods in economic and medical decision-making), by European Union's Horizon 2020 research and innovation programme under grant agreement No 833418, and by the project Reg. No. CZ.02.1.01/0.0/0.0/16_013/0001797 co-funded by the MEYS of the Czech Republic and ERDF.

References

[1] M. Husák, J. Komárková, E. Bou-Harb, P. Čeleda, Survey of attack projection, prediction, and forecasting in cyber security, IEEE Communications Surveys & Tutorials 21 (1) (2019) 640–660 (Firstquarter 2019).

[2] N. Sun, J. Zhang, P. Rimba, S. Gao, L. Y. Zhang, Y. Xiang, Data-driven cybersecurity incident prediction: A survey, IEEE Communications Surveys & Tutorials 21 (2) (2019) 1744–1772 (2019).

[3] T. D. Wagner, K. Mahbub, E. Palomar, A. E. Abdallah, Cyber threat intelligence sharing: Survey and research directions, Computers & Security 87 (2019) 101589 (2019).

[4] S. J. Yang, H. Du, J. Holsopple, M. Sudit, Attack Projection, Springer International Publishing, Cham, 2014, pp. 239–261 (2014).

[5] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, M. Fischer, Taxonomy and Survey of Collaborative Intrusion Detection, ACM Computing Surveys 47 (4) (2015) 55:1–55:33 (May 2015).

[6] A. A. Ramaki, R. E. Atani, A survey of it early warning systems: architectures, challenges, and solutions, Security and Communication Networks 9 (17) (2016) 4751–4776 (2016).

[7] Y.-B. Leau, S. Manickam, Network Security Situation Prediction: A Review and Discussion, Springer Berlin Heidelberg, 2015, pp. 424–435 (2015).

[8] M. Abdlhamed, K. Kifayat, Q. Shi, W. Hurst, Intrusion Prediction Systems, Springer International Publishing, Cham, 2017, pp. 155–174 (2017).

[9] M. Husák, T. Bajtoš, J. Kašpar, E. Bou-Harb, P. Čeleda, Predictive cyber situational awareness and personalized blacklisting: A sequential rule mining approach, ACM Transactions on Management Information Systems. Forthcoming (2020).

[10] V. Bartoš, M. Žádník, S. M. Habib, E. Vasilomanolakis, Network entity characterization and attack prediction, Future Generation Computer Systems 97 (2019) 674–686 (2019).

[11] P. Sokol, A. Gajdoš, Prediction of Attacks Against Honeynet Based on Time Series Modeling, Springer International Publishing, Cham, 2018, pp. 360–371 (2018).

[12] C. W. Geib, R. P. Goldman, Plan recognition in intrusion detection systems, in: DARPA Information Survivability Conference amp; Exposition II, 2001. DISCEX '01. Proceedings, Vol. 1, 2001, pp. 46–55 vol.1 (2001).

[13] T. Hughes, O. Sheyner, Attack scenario graphs for computer network threat analysis and prediction, Complexity 9 (2) (2003) 15–18 (2003).

[14] X. Qin, W. Lee, Attack plan recognition and prediction using causal networks, in: Computer Security Applications Conference, 2004. 20th Annual, 2004, pp. 370–379 (Dec 2004).

[15] A. A. Ahmed, N. A. K. Zaman, Attack intention recognition: A review, IJ Network Security 19 (2) (2017) 244–250 (2017).

[16] K. Zhang, F. Zhao, S. Luo, Y. Xin, H. Zhu, An Intrusion Action-Based IDS Alert Correlation Analysis and Prediction Framework, IEEE Access 7 (2019) 150540–150551 (2019).

[17] H. Farhadi, M. AmirHaeri, M. Khansari, Alert Correlation and Prediction Using Data Mining and HMM, ISeCure 3 (2) (2011).

[18] Z. t. Li, J. Lei, L. Wang, D. Li, A data mining approach to generating network attack graph for intrusion prediction, in: Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on, Vol. 4, 2007, pp. 307–311 (Aug 2007).

[19] Y.-H. Kim, W. H. Park, A study on cyber threat prediction based on intrusion detection event for APT attack detection, Multimedia Tools and Applications 71 (2) (2014) 685–698 (2014).

[20] A. Hernández, V. Sanchez, G. Sánchez, H. Pérez, J. Olivares, K. Toscano, M. Nakano, V. Martinez, Security attack prediction based on user sentiment analysis of Twitter data, in: 2016 IEEE International Conference on Industrial Technology (ICIT), 2016, pp. 610–617 (March 2016).

[21] K. Shu, A. Sliva, J. Sampson, H. Liu, Understanding cyber attack behaviors with sentiment information on social media, in: Social, Cultural, and Behavioral Modeling, Springer International Publishing, Cham, 2018, pp. 377–388 (2018).

[22] P. Shao, J. Lu, R. K. Wong, W. Yang, A transparent learning approach for attack prediction based on user behavior analysis, in: Information and Communications Security, Springer International Publishing, Cham, 2016, pp. 159–172 (2016).

[23] X. Wei, X. Jiang, Comprehensive analysis of network security situational awareness methods and models, in: Instrumentation and Measurement, Sensor Network and Automation (IMSNA), 2013 2nd International Symposium on, IEEE, 2013, pp. 176–179 (2013).

[24] M. R. Endsley, Situation awareness global assessment technique (SAGAT), in: Aerospace and Electronics Conference, 1988. NAECON 1988., Proceedings of the IEEE 1988 National, IEEE, 1988, pp. 789–795 (1988).

[25] M. Abdlhamed, K. Kifayat, Q. Shi, W. Hurst, A system for intrusion prediction in cloud computing, in: Proceedings of the International Conference on Internet of Things and Cloud Computing, ICC '16, ACM, New York, NY, USA, 2016, pp. 35:1–35:9 (2016).

[26] S.-T. Chen, Y. Han, D. H. Chau, C. Gates, M. Hart, K. A. Roundy, Predicting cyber threats with virtual security products, in: Proceedings of the 33rd Annual Computer Security Applications Conference, ACSAC 2017, Association for Computing Machinery, New York, NY, USA, 2017, p. 189–199 (2017).

[27] I. A. Gheyas, A. E. Abdallah, Detection and prediction of insider threats to cyber security: a systematic literature review and meta-analysis, Big Data Analytics 1 (1) (2016) 6 (Aug 2016).

[28] CESNET and Masaryk University, SABU (Jul. 2016).
URL `https://sabu.cesnet.cz/en/start`

[29] CESNET, Intrusion Detection Extensible Alert (Dec. 2016).
URL `https://idea.cesnet.cz/en/index`

[30] CESNET, Warden (Jul. 2017).
URL `https://warden.cesnet.cz/en/index`

[31] M. Husák, J. Kašpar, 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, 2019, pp. 81:1–81:8 (2019).

[32] V. Bartoš, NERD: Network Entity Reputation Database, in: Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES '19, ACM, 2019, pp. 84:1–84:7 (2019).

[33] H. Debar, D. A. Curry, B. S. Feinstein, The Intrusion Detection Message Exchange Format (IDMEF), RFC 4765 (Experimental) (Mar. 2007).
URL `http://www.ietf.org/rfc/rfc4765.txt`

[34] M. Husák, M. Žádník, V. Bartoš, P. Sokol, Dataset of intrusion

detection alerts from a sharing platform, Mendeley Data, v1 (June 2019).
URL http://dx.doi.org/10.17632/p6tym3fghz.1
[35] M. Husák, J. Kašpar, E. Bou-Harb, P. Čeleda, 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, 2017, pp. 22:1–22:10 (2017).
[36] M. Husák, J. Kašpar, Towards predicting cyber attacks using information exchange and data mining, in: 2018 14th International Wireless Communications Mobile Computing Conference (IWCMC), 2018, pp. 536–541 (June 2018).
[37] P. Fournier-Viger, V. S. Tseng, Mining top-k sequential rules, in: International Conference on Advanced Data Mining and Applications, Springer, 2011, pp. 180–194 (2011).
[38] P. Fournier-Viger, J. C.-W. Lin, A. Gomariz, T. Gueniche, A. Soltani, Z. Deng, H. T. Lam, The SPMF Open-Source Data Mining Library Version 2, in: Proc. 19th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2016), Part III, Springer LNCS 9853, 2016, pp. 36–40 (2016).
[39] A. A. Ramaki, M. Khosravi-Farmad, A. G. Bafghi, Real time alert correlation and prediction using bayesian networks, in: 2015 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC), 2015, pp. 98–103 (Sep. 2015).
[40] G. Moura, R. Sadre, A. Pras, Bad neighborhoods on the internet, IEEE Communications Magazine 52 (7) (2014) 132–139 (2014).
[41] V. Stupka, M. Horák, M. Husák, Protection of personal data in security alert sharing platforms, in: Proceedings of the 12th International Conference on Availability, Reliability and Security, ACM, Reggio Calabria, 2017, pp. 65:1–65:8 (2017).
[42] E. Condon, A. He, M. Cukier, Analysis of computer security incident data using time series models, in: Software Reliability Engineering, 2008. ISSRE 2008. 19th International Symposium on, IEEE, 2008, pp. 77–86 (2008).
[43] R. J. Hyndman, G. Athanasopoulos, Forecasting: Principles and Practice, OTexts, 2018 (2018).
[44] P. J. Brockwell, R. A. Davis, Introduction to time series and forecasting, Springer, 2016 (2016).
[45] G. E. Box, G. M. Jenkins, G. C. Reinsel, G. M. Ljung, Time series analysis, control, and forecasting (2015).
[46] R. J. Hyndman, Y. Khandakar, Automatic Time Series for Forecasting: The forecast Package for R, Journal of Statistical Software 27 (July 2007).
[47] R. J. Hyndman, A. B. Koehler, Another look at measures of forecast accuracy, International journal of forecasting 22 (4) (2006) 679–688 (2006).
[48] R. J. Hyndman, Measuring forecast accuracy, Business forecasting: Practical problems and solutions (2014) 177–183 (2014).
[49] A. C. Harvey, Time Series Models, MIT press, 1993 (1993).
[50] P. Royston, Remark AS R94: A remark on algorithm AS 181: The W-test for normality, Journal of the Royal Statistical Society. Series C (Applied Statistics) 44 (4) (1995) 547–551 (1995).
[51] A. Rodriguez, E. Ruiz, Bootstrap prediction intervals in state-space models, Journal of time series analysis 30 (2) (2009) 167–178 (2009).
[52] L. A. Thombs, W. R. Schucany, Bootstrap prediction intervals for autoregression, Journal of the American Statistical Association 85 (410) (1990) 486–492 (1990).
[53] G. Masarotto, Bootstrap prediction intervals for autoregressions, International Journal of Forecasting 6 (2) (1990) 229–239 (1990).