information

Article

Early-Stage Detection of Cyber Attacks

Martina Pivarníková [1], Pavol Sokol [2,*] and Tomáš Bajtoš [2]

[1] Ministry of Investments, Regional Development and Informatization of the Slovak Republic, Computer Security Incident Response Team Slovakia - CSIRT.SK, 811 05 Bratislava, Slovakia; martina.pivarnikova@csirt.sk

[2] Faculty of Science, Pavol Jozef Šafárik University in Košice, 040 01 Košice, Slovakia; tomas.bajtos@upjs.sk

* Correspondence: pavol.sokol@upjs.sk

Received: 1 October 2020; Accepted: 26 November 2020; Published: 29 November 2020

Abstract: Nowadays, systems around the world face many cyber attacks every day. These attacks consist of numerous steps that may occur over an extended period of time. We can learn from them and use this knowledge to create tools to predict and prevent the attacks. In this paper, we introduce a way to sort cyber attacks in stages, which can help with the detection of each stage of cyber attacks. In this way, we can detect the earlier stages of the attack. We propose a solution using Bayesian network algorithms to predict how the attacks proceed. We can use this information for more effective defense against cyber threats.

Keywords: cyber attack; attack prediction; attack projection; early-stage detection; Bayesian network

1. Introduction

Due to the constant development of cyber threats, various defense solutions need to be continuously improved. In addition to developing prevention systems, it is also necessary to focus on detection systems that help to obtain information about threats and attacks. The detection of malicious actions is one of the most critical cybersecurity issues. Intrusion detection refers to the detection of specific patterns or anomaly observations. Nowadays, however, we need to preventively anticipate upcoming harmful activities so that we can react to them and prevent an attack in time before it causes some damage.

Attack prediction study is not as prevalent as detection. Therefore, it is necessary to explore this area of interest because it is beneficial for the entire field of cybersecurity. To predict attacks, it is necessary to examine how they proceed and what steps are being taken. These data can be used to continually improve the systems to detect each phase of the attack. In this way, it is possible to detect the earlier stages of the attacks and predict how they proceed.

Early detection and prediction of cybersecurity incidents, such as attacks, is a challenging task. The threat landscape is continuously evolving, and even with the usage of intrusion detection systems, advanced attackers can spend more than 100 days in a system before being discovered [1]. After the detection of a security incident, we need to determine how the attack will proceed. This is essential because if we can stop the attacker in time, they cannot do as much damage.

It is important to learn from existing attacks so that we can develop tools to find out if such attacks have been repeated. Attack modeling is an intrusion-based methodology that allows one to focus on the different stages of an attack. It is aimed at focusing on different stages of attacks. By identifying attacks at different stages and by implementing tools to disarm the attacks at their various stages, one can take preventive measures to ensure that similar attacks will be detected. It is important to have a layered model to ensure that if one of the defense systems is bypassed, there is another defense line to protect one's organization's assets. That is why we need to establish a multi-layered model of cyber attacks.

In recent years, it has not been sufficient to only be alerted of a security incident. Prevention of the attack altogether has become a necessity. The highest priority in computer security is to prevent an attack and stop the attacker from doing damage. If the path of an attack can be predicted, one has the ability to avoid attacks at every phase. By looking at a survey of the technology, from the host to the network level, one will have an opportunity to study tools or solutions that can be used in protecting against these threats. There are numerous existing prevention methods that are able to stop attacks in progress.

Recognizing an attack's steps is the goal of many cybersecurity analysts. The authors in [2] categorized prediction methods into three categories. An overview can be seen in Table 1.

Table 1. Overview of prediction categories.

Category	**Description**	**Previous Surveys**
Attack projection	What is an adversary going to do next?	Yang et al. [3]
Attack intention recognition	What is the ultimate goal of an adversary?	Ahmed and Zaman [4]
Attack/intrusion prediction	What type of attack will occur, when, and where?	Abdlhamed et al. [5]
Network security situation forecasting	How is the overall situation going to evolve?	Leau and Manickam [6]

The research is focused on early-stage detection and it is based on attack prediction, especially attack projection. This area focuses on the prognosis of the future steps of the attack. The projection of the future stages of an active cyber attack is essential in the context of Cyber Situational Awareness. The attacks often occur over an extended period of time. They involve a lot of steps and use multiple techniques for reconnaissance, exploitation, and obfuscation activities to achieve the attacker's goal. Therefore, it is not sufficient to just detect new or ongoing threats. The projection of future attack steps is deduced from already detected malicious activities. The estimates of current attack tactics may be used to assess imminent threats to critical assets [3].

This paper is based on the previous research of [7] and further develops research conducted by Ramaki et al. [8]. Based on the above-mentioned considerations, we state the following research sub-goals:

1. To propose a multi-stage model suitable for attack projection and early-stage detection, and
2. to design a model for early-stage detection of a cyber attack.

This paper is divided into seven sections. In Section 2, which is focused on related work and existing methods, the analysis of the current approaches of cyber attack prediction is provided. Section 3 presents the drawbacks of existing models and describes the suitable cyber attack model in detail. Subsequently, in Section 4, we propose the approach for early-stage detection of cyber attacks. This includes all of the necessary steps for data processing, alert aggregation, and causal relationship discovery. This section also covers the definition of Bayesian networks. After that, the model for the construction of the Bayesian network and prediction of cyber security alerts is proposed. Section 5 focuses on preprocessing and analysis of the data collection, including the creation of cyber alerts. The example cases of methods for aggregation, causal relationship discovery, and Bayesian network construction are shown. Section 6 presents and discusses the results of the presented methods. Concurrently, it describes groups of alerts and some of the attack paths. In the last section, the conclusion is provided.

2. Related Works

A large number of cyber attack prediction methods use discrete models and graph models, such as attack graphs, Bayesian networks, or Markov models.

In 1998, an attack graph was introduced by Swiler and Phillips [9]. It is a graphical representation of an attack scenario, and it has happened to be a popular method for formal description of attacks. It has become a foundation for other approaches, e.g., methods using Bayesian networks, Markov models, and game-theoretical methods. Their goal was to create a tool for qualitative and quantitative assessment of vulnerabilities. The approach was a great success because it examined a network security state from the system perspective.

Cao et al. [10,11] proposed another variant of the attack graph—the factor graph. It is a probabilistic model that consists of random variables and factor functions. In this paper, it is compared to Bayesian networks and Markov random fields. They used the factor graph to predict attacks with an accuracy of 75% over a dataset of actual security incidents (several years of reports).

The RTECA (Real-Time Episode Correlation Algorithm) was proposed in 2014 by Ramaki et al. [12]. It can be used to detect and predict multi-step attack scenarios. They explain the theoretical and functional implications of the creation of such a tool. Although they propose leveraging the attack graph, the authors have widely used causal correlations in their method.

The authors in [13] developed a method for correlating the intrusion alerts. It produces correlation graphs, which they use for creating attack strategy graphs. They presented techniques for automatically learning attack strategies from alerts raised by intrusion detection systems. These methods extracted attributes relevant to determining an attack strategy, which is represented as a directed acyclic graph, which they called an attack strategy graph. The nodes are known attacks, and the edges between them represent the order of attacks and relationships between them. They also developed a method for easier computer and network forensic analysis. It measures the similarity between sequences of alerts based on their strategies. Their research showed that the proposed methods can successfully extract invariant strategies from alert sequences and can also determine the likeness of those sequences. It can be widely used in identifying attacks that could have been missed by detection systems.

In [14], Li et al. presented another approach based on attack graphs. They described the generation of attack graphs constructed on a data mining approach. The algorithm they proposed uses association rule mining to get multi-step attack scenarios from Intrusion detection system (IDS) alert database. After that, the attack graph is created. The method is also used for calculating the predictability of the attack scenario. It is used for ranking the real-time detection and can help with intrusion prediction.

Liu and Peng [15] developed a game-theoretic framework used for attack prediction. The proposed method can quantitatively predict the probability of attack actions. It can also predict the strategic behavior of the attacker. Thus, it can optimize the precision of correlation-based prediction. This paper presents the first complex framework for motive-based modeling and inference of attackers' intents. In conclusion, the goal of this method is modeling and inference of attack intents, objectives, and strategies.

Wu et al. [16] used another attack prediction method using Bayesian networks. These methods are related to approaches based on attack graphs because a Bayesian network is built from an attack graph. The distinct characteristic of Bayesian networks is the conditional variables and probabilities that are considered in the model.

A Bayesian network is a probabilistic graphical model that describes the variables and the relationships between them. The network is a directed acyclic graph (DAG), where nodes represent the discrete or continuous random variables and edges depict the relationships between them. Each variable has a finite set of mutually exclusive states. The variable and direct edge form a DAG. To each variable A with parents $B_1, B_2, ..., B_n$, there is attached a conditional probability table $P(A|B_1, B_2, ..., B_n)$ [2].

Ishida et al. [17] proposed forecast techniques for fluctuation of attacks. They used Bayesian inference for calculating the probability of increase or decrease of the attacks. Two algorithms were considered in this paper—focusing on the attack cycle and the fluctuation range of the number of events. Because the event counts of some attacks change frequently, the proposed algorithms based on

Bayesian inference were used for predicting the probability, since it can calculate event counts directly. Subsequently, they implemented the forecasting system and tested it on real IDS events.

A real-time alert correlation and prediction framework was introduced by Ramaki et al. [8]. The system includes an online and offline mode. In online mode, the attacker's next move is predicted by the Bayesian attack graph. In the offline mode, the Bayesian attack graph is constructed of low-level alerts. The authors used the DARPA 2000 dataset for research. The prediction accuracy was found to increase with the duration of the scenario for the attack. Thus, accuracy ranged from 92.3% when processing the first attack step to 99.2% when processing the fifth attack step.

Okutan et al. [18] used signals unrelated to the target network in their Bayesian-network-based attack prediction process. The signals include mention of Twitter attacks or the total number of Hackmageddon attacks [19]. As was shown in the results, the prediction accuracy differed from 63% to 99%, making it a promising method.

Since probabilistic graphical models are very powerful modeling and reasoning tools, Tabia et al. [20] proposed an efficient approach based on Bayesian networks. It allows the modeling of local influence relationships. It is dedicated to two main problems in alert correlation. Firstly, an approach based on Bayesian multi-nets was designed, which considered the local influence relationships to improve the prediction. The second problem occurs when multiple intrusion detection systems are in use in the network. In this case, too many of the raised alerts are redundant. Therefore, they proposed an approach for handling IDSs' reliability to reduce the number of false alerts. They based this approach on Pearl's virtual evidence [21].

Another widely used approach to predicting attacks is using Markov models. These methods were implemented along with approaches focused on attack graphs and Bayesian networks at the end of 2000. Farhadi et al. [22] proposed a complex system for alert correlation and prediction. Sequential pattern mining was used to collect the attack scenarios, which were then represented using the hidden Markov model, which was used to identify the attack strategy. Markov models perform well in the presence of unobservable states and transitions. They are not reliant on the possession of complete knowledge. This allowed a successful attack prediction, even though some of the attack stages were undetected or absent.

Using hidden Markov models, Sendi et al. [23] proposed a real-time intrusion prediction system. Multi-step attacks were the main interest in this paper. An empirical review showed how their method could anticipate multi-step attacks, which is especially useful in preventing the attacker from taking control of a huge number of hosts in the computer network.

In 2013, Shin et al. [24] introduced a probabilistic approach for the network-based intrusion detection system APAN, which uses a Markov chain for modeling unusual events in the network traffic to predict intrusion. Unlike other Markov-based methods, this method detects network anomalies and does not aim to predict the next step of an attack as different model-checking approaches do.

Holgado et al. [25] proposed a novel method based on a hidden Markov model for multi-step attack prediction using IDS alerts. They considered hidden states as a particular type of attack. At first, the preliminary training phase based on IDS alert information needs to be done. These observations are acquired by pairing the IDS alert information with a previously built database. Unsupervised and supervised methods for learning are performed in the training model. The prediction module can compute the best state sequence using the Viterbi and forward–background algorithms. The success of this method was shown in the successful detection of the distributed denial of service (DDoS) stages, which is a big problem in detection systems nowadays.

Table 2 shows the approaches in the cyber attack prediction methods. The first proposed method that has become popular involves prediction using an attack graph. It is the most transparent and easy-to-understand model for attack step representation. It has become beneficial in predicting the next steps in an attack. One of the lesser-known approaches is game theory. Nevertheless, it can be very useful in detecting DDoS attacks, which are very hard to predict. More commonly used methods include machine learning models. The first of them, the Bayesian network, has excellent accuracy results. However, it is tough to create this model from actual network traffic because the attackers can create loops in security alert data during attack implementation. Less intuitive approaches, but with great results, are the Markov chains and the hidden Markov model. These can be handy in predicting multi-step attacks.

Table 2. Summary of cyber attack prediction methods.

Paper	Approach/Model	Advantages and Limitations
Swiler and Phillips [9]	Attack graph	The first proposed methods
Cao et al. [10,11]	Attack graph	75% accuracy, factor graph
Ramaki et al. [12]	Attack graph	95% accuracy
Ning et al. [13]	Attack graph	Using correlation graphs
Li et al. [14]	Attack graph	Only partial graphs
Liu and Peng [15]	Game theory	Detection of nontrivial cyber attacks (e.g., DDoS)
Wu et al. [16]	Bayesian network	Only model extensions
Ishida et al. [17]	Bayesian network	70% accuracy, prediction of fluctuation of attacks
Ramaki et al. [8]	Bayesian attack graph	92.3–99.2% accuracy, real-time
Okutan et al. [18]	Bayesian network	63–99% accuracy, non-conventional signals
Tabia et al. [20]	Bayesian network	Reducing false alarms
Farhadi et al. [22]	Hidden Markov model	81.33–98.3% accuracy, data mining, illustrative example of a real-time attack projection
Sendi et al. [23]	Hidden Markov model	Prediction of the next step in a multi-step attack
Shin et al. [24]	Markov chain	Improving intrusion detection by predictions
Holgado et al. [25]	Hidden Markov model	Multi-step attack prediction, real-time, able to predict DDoS

On the other hand, Markov chains and the hidden Markov model need specific information. Due to the lack of information provided from the specific type of dataset, it is not possible to determine the values of the observation probability matrix. It is not certain what the probability of an attack is based on an observable alert. Therefore, we have decided to use a Bayesian network to create a method for cyber attack prediction.

3. The Proposed Cyber Attack Model

Cyber attack modeling is an important issue for securing any network and can help save money, time, and other resources. There exist several techniques that are used to model and analyze cyber attacks. The important part of understanding how every cyber attack works is to comprehend the steps that an attacker makes in order to reach their target. The goal of these approaches is to understand cyber attack characteristics to provide better security for a system. To defeat cyber attacks, it is also important to comprehend the attacker's objectives and their means. Understanding the characteristics of attacks is paramount in creating a good security strategy. Attack modeling is important in gaining a perspective on how can a cyber attack be stopped in a coordinated manner.

We considered using one of the following three models for analysis and use in our paper—the kill chain model [26], the model presented in [27], and the Diamond model [28]. The cyber kill chain model defines the path of a cyber attack. In this seven-layered model, each layer is critical for the evaluation of the attack. There are seven stages of the traditional kill chain model—reconnaissance, weaponization, delivery, exploitation, installation, command and control, and acting on the objective. This model is based on the assumption that attackers will seek to penetrate the computer system in a sequential and progressive way.

A sample anatomy of a cyber attack was also presented by Bou Harb et al. in their paper about cyber scanning [27]. The anatomy of the attack consists of the following steps—cyber scanning, enumeration, intrusion attempt, elevation of privilege, performing a malicious task, deploying malware/a backdoor, deleting forensic evidence, and exiting.

The Diamond model is one of the models for intrusion analysis. In this model, an attacker targets a victim on two main occasions, rather than using a sequence of continuous steps like the kill chain. This model consists of four elements—adversary, infrastructure, capability, and the victim [28].

Based on the analysis of the presented models, it was concluded that none of them met the requirements. The first two models contain stages that cannot be detected by IDSs. Since the Diamond model is not focused on attack steps, it is also not relevant to this research. That is why a new model needs to be developed. We introduce a hybrid model that includes four stages. This model can be seen in Figure 1.

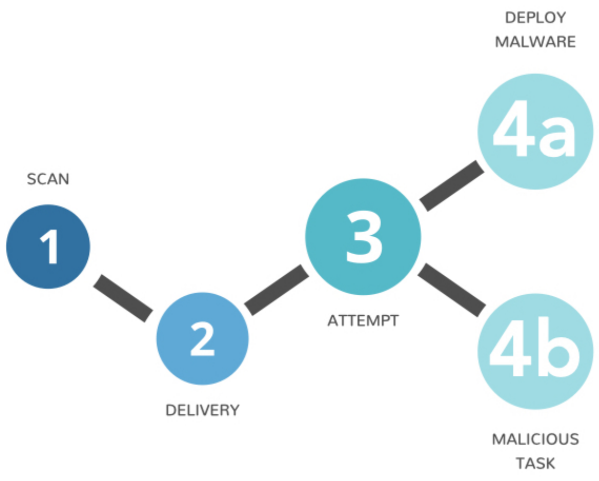

Figure 1. Proposed cyber attack model.

3.1. Scan

Cyber scanning is the first step in any sophisticated attack. This step is needed so the attacker can obtain information about their target, e.g., harvesting email addresses and login credentials or finding network vulnerabilities, etc. There are a variety of existing methods that an attacker can use to achieve this goal. There are two types of scanning techniques—passive and active.

An attempt to gain information about a target system or computer network that can be collected without actively engaging with the system is called passive scanning [29]. This can be performed by looking up the information about employees on a company's website. These can be email addresses, personal social media accounts, or phone numbers. LinkedIn and other social media networks can store the information of employees. This can help an attacker to identify their potential goal. In addition, social media accounts of employees can provide information about technologies used by a company. After finding out enough information about a victim, the possibility of success using social engineering techniques increases. Passive scanning is the most difficult thing to detect from the perspective of intrusion detection systems.

Active scanning is an attack in which an attacker engages a targeted network to gain information about vulnerabilities in it [29]. If an attacker is using an automated tool for network scanning, the IDS is likely to detect it and raise an alert. Performing active scanning is very valuable for determining any vulnerabilities that can be used. Network probing can be detected by correlating logs over a period of time. Therefore, it can be determined who may be targeting the system. This paper is focused on network scanning captured by Snort IDS. It can easily detect this type of stage. For example, if an attacker is using the NMAP tool to obtain information about a computer system (open ports or type of operating system), Snort can recognize a large number of various types of incoming packets and, therefore, identify the type of scan. It will raise a network-scan type of alert after recognizing this stage.

3.2. Delivery

Delivery is a critical part of every cyber attack model because it is responsible for an effective cyber attack. In most cyber attacks, it is necessary to have some kind of user cooperation, like downloading and executing malicious files or visiting malicious web pages on the internet. This stage presents a high risk for the attacker because delivery leaves evidence. Multiple delivery methods can be used, such as email attachments, phishing attacks, drive-by download, USB/removal media, or DNS cache poisoning [30].

Snort can detect malicious code by recognizing the transmission of executable code or suspicious strings in network traffic.

3.3. Attempt

Intrusion detection means discovering that some entity—an attacker—has attempted to gain—or has already gained—unauthorized access to the computer system. An intrusion attempt has a potential for a deliberate unauthorized attempt to enter a computer, system, or network to access information and manipulate information or render a system unreliable or unusable [31]. Intrusion attempts are experienced by victims, servers, networks, systems, and computers. These attempts can be discovered by intrusion detection systems. In the best case, it can be a false alarm because detection systems can sometimes raise false positive alerts. In order to determine if this was the case, it is needed to look at the details of the alert. If the attempt came from an infected system in the local network, it could provide information about this system; for example, the IP address that caused the alert. It can be later checked for any malicious activity. The last possibility is that there was an attempt to attack from an outside local network, but it was blocked. There is no way to determine if the attacker did not obtain any information. Detection of intrusion attempts can be helpful in defending a network, for example, by blacklisting IP addresses or updating firewall configurations.

3.4. Deploying Malware/Malicious Tasks

This stage contains the last four stages in the kill chain model [26]. In this phase, the malware is successfully installed on a computer system, or an attacker has obtained rights on the targeted device and is performing some malicious action. It starts with exploitation, which is initiated by installing the malware inside the target computer. The malware or the attacker has the required access rights. If the malware is an executable file or the malicious activity is based on code injection or an insider threat, then the installation is not required. After the malware is installed, it will start communication with the command and control server, which can be an attacker's device, server, or even social media network web server. If the attacker has gained access to a targeted computer system, he/she performs some malicious task; for example, stealing private and intellectual data from the network.

4. The Proposed Model for Early-Stage Detection of Cyber Attacks

This paper focuses on early-stage detection of cyber-attacks and, at the same time, the prediction of the subsequent stages of the attack. These attacks consist of multiple stages and may occur over an extended period of time. In this paper, we study how probabilistic inference can be used to analyze attack scenarios based on the information of the relations between alerts. This section describes a machine learning approach using a Bayesian network to predict cyber attacks' next steps. Algorithms for aggregation, causal relationship discovery, and Bayesian network construction will be introduced in this section.

4.1. Alert Aggregation

The first step in alert processing is aggregation, since intrusion detection systems are susceptible to alert flooding, meaning that they generate a huge number of alerts. Therefore, it is often hard to cope with a big amount of data. This issue can be solved by aggregating all of the alerts. Every aggregated event consists of:

- alert message,
- source IP address,
- destination IP address,
- source port,
- destination port,
- alert counter,
- start timestamp,
- end timestamp.

It is difficult to obtain a bigger picture in a large number of probes. Aggregation reduces the number of redundant alerts generated. This simplifies alert analysis and further processing. Possibly, this will not affect the information obtained in reduced alerts because only alerts that have the same important attributes are merged. Therefore, thousands of generated alerts are aggregated into a hyper-alert. Alerts were aggregated into one based on multiple attributes. All of the subsequent properties have to be met in two alerts for them to be merged into one:

- Time window—alerts have to be in the same time window, which was set to 10 min;
- Alert message and type—both alerts need to have the same message and, therefore, the same type;
- IP addresses—source and destination IP addresses have to be the same in both alerts.

The output of aggregation is stored in a multi-dictionary object: agr_alerts_all. The alerts are added to this data structure based on finding the key, which contains an alert message, source IP address, and destination IP address of the alert. The pseudo-code of the algorithm (Algorithm 1) can be seen below.

Algorithm 1 Aggregation of alerts.

```
    Input $A$ (all alerts generated by Snort)
    Output $agr\_alerts\_all$

 1: procedure AGGREGATION
 2:     for each alert $a$ in $A$ do
 3:         if $a$ not in $agr\_alerts\_all$ then
 4:             $agr\_alerts\_all.add(a)$
 5:         else
 6:             for each alert $agr$ in $agr\_alerts\_all$ with same message, IP as $a$ do
 7:                 if $a$ and $agr$ in same time window then
 8:                     $agr.ports.add(a_{srcport}, a_{dstport})$
 9:                     $agr.count \leftarrow agr.count + 1$
10:                 else
11:                     $agr\_alerts\_all[agr].add(a)$
12:                 end if
13:             end for
14:         end if
15:     end for
16: end procedure
```

4.2. Causal Relationship Discovery

In a causal Bayesian network, each arc is interpreted as a direct causal influence between a parent node and a child node relative to the other nodes in the network [32]. Therefore, after aggregation, the relationships between hyper-alerts are created. A directed edge between hyper-alerts A and B depicts that B occurred after A, and the event in A influences the event in B. This paper assumes that the attacker will not return to earlier stages and, thus, will not repeat certain steps. If the attacker was allowed to go back, it would not be possible to determine the probability deterministically. This would mean that there would be no final number of vertices where the attack could end. Therefore, it is expected that the attacker will end somewhere and, thus, will not perform the same stages over and over. Accordingly, a directed edge between A and B *(A -> B)* is constructed if the following criteria are met:

1. Time window—hyper-alerts A and B have to be within the same time window, which was set to 10 min;
2. Order—B occurred after A;
3. Stages—B is in a lower phase of attack than A;
4. IP addresses:
   - The source and destination IP addresses in A and B are the same, or
   - The destination IP address in A is the same as the source IP address in B.

After discovering all causal relationships between the hyper-alerts, the conditional probabilities need to be determined. The probability of *A -> B* or the probability of B under the condition of A will be denoted with the notation $P(B|A)$. However, before that, a correlation matrix showing the frequency of occurrences of the relationship *A -> B* needs to be created. The given matrix is stored in a pandas DataFrame object—correlated_alerts.

Algorithm 2 Causal relationship discovery.

```
  Input agr_alerts_all
  Output correlated_alerts

procedure CORRELATION
    for each alert $a$ in agr_alerts_all do
        for each alert $b$ in agr_alerts_all do
            if $a_{message}$ != $b_{message}$ and $a_{phase}$ != $b_{phase}$ then
                if $a$ and $b$ in same time window then
                    if $a_{srcIP}, a_{dstIP} == b_{srcIP}, b_{dstIP}$ then
                        if $a$ before $b$ and $a_{phase} < b_{phase}$ then
                            $correlated\_alerts[a][b] \leftarrow correlated\_alerts[a][b] + 1$
                        else if $b$ before $a$ and $b_{phase} < a_{phase}$ then
                            $correlated\_alerts[b][a] \leftarrow correlated\_alerts[b][a] + 1$
                        end if
                    else if $a_{dstIP} == b_{srcIP}$ and $a$ before $b$ and $a_{phase} < b_{phase}$ then
                        $correlated\_alerts[a][b] \leftarrow correlated\_alerts[a][b] + 1$
                    else if $b_{dstIP} == a_{srcIP}$ and $b$ before $a$ and $b_{phase} < a_{phase}$ then
                        $correlated\_alerts[b][a] \leftarrow correlated\_alerts[b][a] + 1$
                    end if
                end if
            end if
        end for
    end for
end procedure
```

Subsequently, two variables representing causal relationships are defined: *Cor(A,B)*, which refers to the number of all causal relationships between alerts *A -> B*, hence the value in matrix:

correlated_alerts[A][B], and *Cor(A,*)*, which represents the number of all causal relationships between A and all of the other hyper-alerts, which is the sum of the row A in the correlated_alerts matrix. Now, the conditional probabilities $P(B|A)$ can be computed as:

$$P(B|A) = \frac{Cor(A,B)}{Cor(A,*)} \quad . \tag{1}$$

We work with table causal_relationships, which shows the results of the causal relationship discovery algorithm (Algorithm 2). It contains the following columns:

- Message of alert A,
- Index of alert A,
- Message of alert B,
- Index of alert B,
- Cor(A,B),
- Cor(A,*), and
- P(B | A).

All the data that we need for the construction of the network are in this table.

4.3. Bayesian Network

Bayesian networks form an important part of artificial intelligence by combining areas of probability theory and graph theory to solve uncertainty and complexity problems. They are one of the graphical probabilistic models, allowing a compact representation of the probability distribution of the simultaneous occurrence of monitored events [33]. The advantage of Bayesian networks is their relative intuitiveness for humans, since it is easier to understand the direct relationships between events and local probability distributions than the resulting probability distributions of occurrence of multiple events simultaneously. Their name is based on the Bayesian statistics on which they are based, which are based on the so-called Bayes' theorem expressing the change of previous assumptions in the light of new facts.

A Bayesian network (or causal network) is represented as a directed acyclic graph (DAG). Each node in this graph is a variable that has certain states. The directed edges in this graph represent relationships between variables—if there is a directed edge between two variables, then one is dependent on another [34]. If there is no connection between the two nodes, it does not mean that they are completely independent, because they can be connected through other nodes. However, they may become dependent or independent depending on the evidence that is established at other nodes. Nodes and connections build the structure of the Bayesian network, and we call it the structural specification.

This model consists of several parameters. The first one is the prior probability of parent nodes, which are not dependent on any states. Every child node has a conditional probability table (CPT) that states the prior knowledge between the node and its parent. An element of the CPT is defined by [35]:

$$CPT_{ij} = P(childstate = j|parentstate = i) \tag{2}$$

Some variables may have specific values that were observed. Let Y be the set of observed variables and Y_0 the corresponding set of values. Let X be the set of variables that are interesting for us. Inference is the process of computing the posterior probability $P(X|Y = Y_0)$. The posterior probability $P(X|Y = Y_0)$ is defined by [36]:

$$P(X|Y = Y_0) = \frac{P(X, Y = Y_0)}{P(Y = Y_0)} \tag{3}$$

In this paper, we will only consider a finite set $U = \{X_1, ..., X_n\}$ of discrete random variables where each variable X_i may take on values 0 or 1. In our case, these random variables will be aggregated alerts—hyper-alerts. We define each node of the causal network to have a binary state, i.e., 1 or 0. The value of 1 represents the alert in the node being raised, while the value of 0 indicates that it was not. Since the variables are discrete, the conditional probability tables contain the probabilities that a variable will contain one of all possible values for each combination of their parents' values.

4.4. Bayesian Network Construction

For Bayesian network generation, the causal relationships and conditional probabilities between hyper-alerts are needed. Therefore, the data from the previous subsection that are captured in the causal_relationships table will be used. The procedure for creating the Bayesian network is shown below. The method for creating the Bayesian network takes three input parameters: (I) agr_alerts_all, (II) correlated_alerts, and (III) causal_relationships. The output of this process is the Bayesian network of the hyper-alerts. The steps for Bayesian network construction are as follows:

1. Step 1: For each causal relationship between hyper-alerts x_i and x_j in the correlated_alerts table, a directed edge x_i -> x_j is generated.
2. Step 2: Each node in the Bayesian network has a conditional probability table (CPT). This table displays the probability of the node given the values of its parent nodes. After all the edges are added to the network in Step 1, a method for generating the conditional probability table of each node was used. This method was introduced by [8]:

$$P(x_j|Pa(x_j)) = \begin{cases} 0, & \forall x_i \in Pa(x_j), x_i = false \\ P(\bigcup_{x_i=true} t_i) = 1 - \prod_{x_i=true}(1 - P(t_i)), & otherwise \end{cases} \quad (4)$$

In this function, x_i indicates values of the variables in the parent nodes $Pa(x_j)$ of x_j. The variable t_i denotes the transition that changes the state of the network from x_i to x_j, where $x_i \in Pa(x_j)$. Using this formula, each field in the conditional probability table in the vertex x_j is calculated based on the values in the parent nodes. It follows from this procedure that if all of the parent nodes have a value of 0 and, thus, no such alerts have occurred, the child node cannot occur either, and the value in the appropriate field will be 0. As a result of this procedure, the Bayesian network with conditional probability tables conditioned on different states of parents in each node has been constructed.

4.5. Alert Prediction

Based on the created Bayesian network, we can further calculate the probability of occurrence of a certain alert under the condition of other alerts using Bayesian inference. After an intrusion system generates one or more alerts, the so-called posterior probability, which is the probability of occurrence of every other state, can be computed. In this paper, only the probabilities of the alerts that belong to the third (Attempt) or fourth stage (Malicious) of the proposed model will be computed. Therefore, the most probable endings of the attack can be determined. With this knowledge, the system administrator can make precautions so they can mitigate the risk of a successful cyber attack that would otherwise cause bigger damage.

5. Data Preprocessing and Data Analysis

5.1. Datataset

For this paper, we work with the Intrusion Detection Evaluation Dataset (CICIDS2017) [37]. It includes benign and the most common attacks, which match the real-world data in (pcap format). It also includes the results of the network traffic analysis using the CICFlowMeter with labeled flows based on the timestamp, source, and destination IPs, source and destination ports, protocols, and attack. The data capturing period started at 9 a.m., Monday, 3 July 2017, and ended at 5 p.m. on Friday, 7 July 2017, for a total of five days. Monday only included regular traffic. The implemented attacks include Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet, and DDoS. They were executed in both the morning and afternoon on Tuesday, Wednesday, Thursday, and Friday [38]. We can see all of the implemented attacks for each day of data collection in Table 3.

Table 3. Overview of the activities in the dataset by day.

Day	Activity
Monday	Benign (Normal human activities)
Tuesday	Brute Force FTP (FTP-Patator)
	Brute Force SSH (SSH-patator)
	NAT Process on Firewall
Wednesday	DoS/DDoS
	DoS Slowloris
	DoS Slowhttptest
	DoS Hulk
	DoS GoldenEye
	NAT Process on Firewal
	Heartbleed Port 444
Thursday	Web Attack—Brute Force
	Web Attack—XSS
	Web Attack—SQL Injection
	NAT Process on Firewall
	Infiltration—Dropbox download
	Meta exploit Win Vista
	Infiltration—Cool disk—MAC
Friday	Botnet ARES
	Port Scan
	NAT Process on Firewall
	DDoS LOIT

This dataset was processed by the Snort intrusion detection system [39], which raised multiple alerts that matched the described attacks. As will be described later in this work, the alerts were aggregated based on the exact criteria into one hyper-alert. They were afterwards correlated—relationships between Hyper-alerts were established based on some properties. After these methods, a directed weighted graph was made, where vertices are hyper-alerts and the weighted edges are relations between them. This graph will be used as a base graph for the cyber attack prediction method using the discrete model—the Bayesian network.

The alerts generated from Thursday's traffic are used as a demonstration of a specific case. This day included multiple web attacks, such as cross-site scripting and SQL injection. Then, in the afternoon, an infiltration was performed. We can see the detailed description in Table 4.

5.2. Alert Preprocessing

For alert processing, the intrusion detection system Snort [40] was used. It analyzes the above-mentioned dataset based on two sets of alert rules. It takes a *.pcap* file as an input and resolves it based on the rules defined by the user in the configuration file. The first rule set was created by Hansson [41]. The company Proofpoint provides the second open rule set [42]. Rules from the first rule set are marked as "NF" and rules from the second rule set are marked as "ET". Once an attack or an abnormal situation is identified, an alert will be raised.

Snort [39] is an open-source network intrusion detection system that is capable of performing real-time traffic analysis and packet logging on IP networks. It can perform protocol analysis and content searching/matching, and can be used to detect a variety of attacks and probes, such as buffer overflows, stealth port scans, Common Gateway Interface (CGI) attacks, SMB probes, OS fingerprinting attempts, and much more.

Table 4. Description of Thursday's activities in the Intrusion Detection Evaluation Dataset (CICIDS2017).

Type of Attack	Time Interval	Notes
Web Attack —Brute Force	9:20 a.m.–10 a.m.	Attacker: Kali, 205.174.165.73, Victim: WebServer Ubuntu, 205.174.165.68
ine Web Attack—XSS	10:15 p.m.–10:35 a.m.	Attacker: Kali, 205.174.165.73, Victim: WebServer Ubuntu, 205.174.165.68
ine Web Attack—SQL Injection	10:40 a.m.–10:42 a.m.	Attacker: Kali, 205.174.165.73, Victim: WebServer Ubuntu, 205.174.165.68
ine Infiltration—Dropbox download	14:19 p.m–14:21 p.m., 14:33-14:35 p.m.	Attacker: Kali, 205.174.165.73 Victim: Windows Vista, 192.168.10.8
ine Infiltration—Cool disk—MAC	14:53 p.m.–15:00 p.m.	Attacker: Kali, 205.174.165.73 Victim: MAC, 192.168.10.25
ine Infiltration—Dropbox download	15:04 p.m.–15:45 p.m.	
(the first step)		Attacker: Kali, 205.174.165.73, Victim: Windows Vista, 192.168.10.8
(the second step)		Attacker: Vista, 192.168.10.8, Victim: All other clients

As mentioned above, the .pcap files from the CICIDS2017 dataset were analyzed by the Snort intrusion detection system. An output that contains raised alerts was generated into a .csv (comma-separated values) file. Data from this file were inserted into pandas DataFrame objects. Existing intrusion detection systems sometimes generate many false alerts. Therefore, we looked at the benign activity in the network from Monday. Data from each day that contained cyber attacks were filtered with benign activity from Monday's data. Thus, they would contain false positives. That being the case, only suspicious, non-normal cyber alerts were analyzed further. Based on the proposed model, which contains four stages, alerts from the intrusion detection system were assigned into various stages. Raised alerts were sorted based on their types to the proposed stages.

5.3. Alert Generation and Preprocessing

After using the Snort intrusion detection system to analyze these data, the input file in .csv format was created. Next, the data from this file were imported into Python, into the pandas DataFrame data structure. The data were filtered with alerts from Monday so they would not contain any false alerts or false positives. After that, 35 types of alerts remained. As we can see in Table 5, their numbers were very high.

5.4. Alert Aggregation

Therefore, aggregation must be performed on the data. This will significantly reduce the number of redundant alerts so that they can be further processed. As mentioned, an `AgrAlert` class object was created for each alert. The time window for aggregation was set to 10 min. The set of rules determined if the alerts would be merged into one. Two alerts had to be in the same time window and have the same message and the same pair of IP addresses. All of the assembled alerts were stored in the `agr_alerts_all` variable. After this procedure, only 203 hyper-alerts remained for further processing. A preview of them can be seen in Table 6.

5.5. Causal Relationship Discovery

A very important step in creating the Bayesian network is to determine the relationships between hyper-alerts. Therefore, an adjacency matrix `correlated_alerts` with frequencies on the edges was calculated based on a set of rules mentioned in the previous section. The value on the edge depicts how many times one alert occurred after another. A graph was later constructed from this matrix. Number one was added to the field in the matrix, and thus, a directed edge was created between the two alerts if the criteria between two alerts were fulfilled. They must have occurred in the same time window, which was set to an hour and a half. There were four rules that needed to be fulfilled. The first one was that the order had to be maintained. The second one said that the pair of IP addresses was the same or the destination IP address of the first alert was the same as the source IP address of the second one. The other two rules are based on the design of the attack model. The edge A -> B—hence, the edge from A to B—was only created if A and B were not in the same phase, and on top of that, the phase of A had to be in a lower phase than the alert B. In consequence, all of the cycles were removed, and a graph with a tree structure could be created. The table of the `correlated_alerts` can be seen in Table 7. In this table, we omitted the columns with only zero values (no correlation). The alerts in the columns are marked as follows: A3 - GPL NETBIOS SMB-DS IPCshare access; A12 - NF - Echo Reply Payload - Bigger than 100 bytes; A24 - NF - Bad TLD domain - click DNS query - Check domains; and A34 - NF - Echo Request Payload- Bigger than 100 bytes.

Table 5. Numbers of alerts after preprocessing.

Message	Count
NF—SCAN NMAP -sS 1024 Window	128,830
ET TROJAN Windows Microsoft Windows DOS prompt command Error not recognized	27,023
ET WEB_SERVER Script tag in URI Possible Cross Site Scripting Attempt	353
NF—SCAN nmap fingerprint attempt	346
NF—SCAN nmap XMAS	284
ET SCAN NMAP OS Detection Probe	254
NF—Echo Request Payload - Bigger than 100 bytes	254
NF—SCAN NMAP OS Detection Probe	254
NF—Bad TLD domain - click DNS query - Check domains	156
ET SCAN Suspicious inbound to MSSQL port 1433	110
ET SCAN Suspicious inbound to mySQL port 3306	108
ET SCAN Suspicious inbound to Oracle SQL port 1521	108
ET SCAN Suspicious inbound to PostgreSQL port 5432	107
NF—SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	96
ET SCAN Possible Nmap User-Agent Observed	96
ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	96
NF—Echo Reply Payload - Bigger than 100 bytes	88
GPL NETBIOS SMB-DS IPC$ share access	75
ET SCAN Potential VNC Scan 5800-5820	28
NF—SCAN Potential VNC Scan 5800-5820	28
ET SCAN Potential VNC Scan 5900-5920	28
ET WEB_SERVER Possible SQL Injection Attempt UNION SELECT	8
ET WEB_SERVER Possible SQL Injection Attempt SELECT FROM	6
ET POLICY Dropbox.com Offsite File Backup in Use	6
ET WEB_SERVER SELECT USER SQL Injection Attempt in URI	5
GPL ICMP_INFO PING *NIX	5
ET SCAN Behavioral Unusual Port 135 traffic Potential Scan or Infection	4
ET TROJAN Windows dir Microsoft Windows DOS prompt command exit OUTBOUND	3
ET WEB_SERVER Possible MySQL SQLi Attempt Information Schema Access	3
ET WEB_SERVER SQL Errors in HTTP 200 Response (error in your SQL syntax)	3
ET SCAN Behavioral Unusually fast Terminal Server Traffic Potential Scan or Infection (Inbound)	2
ET CURRENT_EVENTS Possible Phishing Redirect Dec 13 2016	2
ET POLICY SSLv3 outbound connection from client vulnerable to POODLE attack	2
ET SCAN Behavioral Unusually fast Terminal Server Traffic Potential Scan or Infection (Outbound)	2
NF—Possible Website defacement - Hacked by - Generic rule - Inbound	1

For the construction of Bayesian network, conditional probabilities between adjacent vertices needed to be calculated. Therefore, two variables were computed—`Cor(A, *)`, which is the number of all alerts originating in A, and `Cor(A, B)`, ergo, the number of times that alert B occurred after alert A. Next, the table of `causal_relationships` was created, containing all the information that will be used in Bayesian network creation.

Table 6. Aggregated alerts.

Message	IP Source	IP Destination
ET CURRENT_EVENTS Possible Phishing Redirect Dec 13 2016	104.23.129.81	192.168.10.17
ET CURRENT_EVENTS Possible Phishing Redirect Dec 13 2016	104.23.129.81	192.168.10.9
ET POLICY Dropbox.com Offsite File Backup in Use	162.125.4.5	192.168.10.8
ET POLICY Dropbox.com Offsite File Backup in Use	162.125.4.5	192.168.10.8
ET POLICY Dropbox.com Offsite File Backup in Use	162.125.18.133	192.168.10.9
...		
NF—SCAN nmap fingerprint attempt	192.168.10.8	192.168.10.12
NF—SCAN nmap fingerprint attempt	192.168.10.8	192.168.10.15
NF—SCAN nmap fingerprint attempt	192.168.10.8	192.168.10.50
NF—SCAN nmap fingerprint attempt	192.168.10.8	192.168.10.16
NF—SCAN nmap fingerprint attempt	192.168.10.8	192.168.10.14

Table 7. Table of correlated alerts.

Alert	A3	A12	A24	A34
A0—NF SCAN NMAP -sS 1024 Window	12	20	2	26
A1—ET SCAN Suspicious inbound to mySQL port 3306	10	16	2	24
A2—ET SCAN Possible Nmap User-Agent Observed	2	0	0	0
A6—ET SCAN Potential VNC Scan 5800-5820	6	12	2	18
A7—NF SCAN nmap XMAS	12	0	2	2
A8—ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap)	2	0	0	0
A10—ET SCAN NMAP OS Detection Probe	12	0	2	2
A12—NF Echo Reply Payload - Bigger than 100 bytes	44	0	0	0
A13—NF SCAN nmap fingerprint attempt	12	0	0	2
A15—NF SCAN NMAP OS Detection Probe	12	0	2	2
A16—ET POLICY SSLv3 outbound connection vulnerable to POODLE attack	0	0	2	0
A19—ET SCAN Suspicious inbound to PostgreSQL port 5432	10	16	2	24
A21—ET SCAN Suspicious inbound to Oracle SQL port 1521	10	16	2	24
A26—ET SCAN Potential VNC Scan 5900-5920	6	12	2	18
A28—NF SCAN Potential VNC Scan 5800-5820	6	12	2	18
A31—ET SCAN Suspicious inbound to MSSQL port 1433	10	16	2	24
A33—NF SCAN Nmap Scripting Engine User-Agent Detected (Nmap)	2	0	0	0
A34—NF Echo Request Payload - Bigger than 100 bytes	12	0	2	0

5.6. Bayesian Network Construction

Since all of the information for making the Bayesian network was obtained, all that is left is to calculate conditional probability tables (CPTs) for each vertex in the graph. The created graph is illustrated in Figure 2. This graph will become the Bayesian network by adding the CPTs. The legend to the graph is shown in Table 8, joining numbers in the graph with alerts (equivalent to ID) and their messages and stages. Stages are numbered as follows: *1—SCAN, 2—DELIVERY, 3—ATTEMPT, 4a—DEPLOY MALWARE, 4b—MALICIOUS TASK.*

All of these alerts were assigned to these stages before. The keyword in the alert message is used to categorize a rule for intrusion detection. For example, most of the alerts that belong to the "Scan" phase contain the keyword "SCAN" in the beginning; therefore, they can be categorized mostly automatically. The alerts that could not be categorized automatically were assigned into stages manually.

Unfortunately, the table shows that no alerts belong to stages 3 (ATTEMPT) and 4a (DEPLOY MALWARE). The fourth phase is present only in 4b (MALICIOUS TASK) because there are no malware execution or distribution actions in this dataset. Due to its nature, there are no attack attempts on the Bayesian network. This is because this dataset was created by collecting and joining simple attacks. Therefore, there is no complex sample of an attack on it. Another reason is that Snort very often detected only an attempt phase, which was not related to other alerts. Therefore, a separate vertex without relationships with others was not added to the graph.

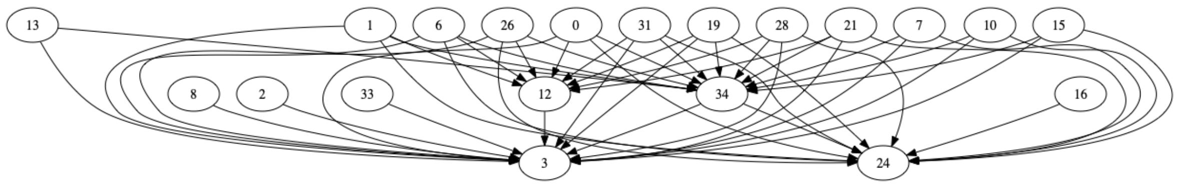

Figure 2. The graph that was created.

Table 8. Description of alerts.

ID	Message	Phase
A0	NF SCAN NMAP -sS 1024 Window	1
A1	ET SCAN Suspicious inbound to mySQL port 3306	1
A2	ET SCAN Possible Nmap User-Agent Observed	1
A3	GPL NETBIOS SMB-DS IPC$ share access	4b
A6	ET SCAN Potential VNC Scan 5800-5820	1
A7	NF SCAN nmap XMAS	1
A8	ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	1
A10	ET SCAN NMAP OS Detection Probe	1
A12	NF Echo Reply Payload - Bigger than 100 bytes	2
A13	NF SCAN nmap fingerprint attempt	1
A15	NF SCAN NMAP OS Detection Probe	1
A16	ET POLICY SSLv3 outbound connection from client vulnerable to POODLE attack	1
A19	ET SCAN Suspicious inbound to PostgreSQL port 5432	1
A21	ET SCAN Suspicious inbound to Oracle SQL port 1521	1
A24	NF Bad TLD domain - click DNS query - Check domains	4b
A26	ET SCAN Potential VNC Scan 5900-5920	1
A28	NF SCAN Potential VNC Scan 5800-5820	1
A31	ET SCAN Suspicious inbound to MSSQL port 1433	1
A33	NF SCAN Nmap Scripting Engine User-Agent Detected (Nmap Scripting Engine)	1
A34	NF - Echo Request Payload - Bigger than 100 bytes	2

The computational complexity of creating a Bayesian network is very high. Therefore, the algorithm will store the network's information, such as vertices and conditional probability tables, in the file. The output file is in BIF format, which is a structure that the `pgmpy` library can work with. We advanced through the vertices in topological order. Topological sorting is a linear ordering of its nodes such that for all directed paths from x to y $(x!=y)$, x comes before y in the ordering. Only the acyclic graph has a topological ordering.

6. Results and Discussion

In this section, the process of the evaluation of the individual methods will be presented. At first, the idea of this paper came from the research made by Ramaki et al. [8]. However, several problems occurred when we tried to follow the steps they took in their approach. The issues within that paper are presented in this section, as well as the modified approach that was developed in order to avoid those problems.

The authors in the mentioned paper tried to create a Bayesian network to predict cyber attack steps. After preprocessing our data from Snort, a similar method of aggregation was used to reduce the number of alerts. Next, the algorithm for causal relationship discovery was evaluated. It was also inspired by the method in the mentioned paper.

At this point, in their proposed method, a problem arose. The occurrence of cycles in the resulting graph was not mentioned in the given paper. However, we succeeded in solving this problem because the two conditions were stated in the causal relationship discovery phase. The attacker can only move across the stages of the cyber attack.

The creation of these conditions was based on the presented model of attack steps. The first condition resulted in eliminating the double-sided edges between the vertices. For example, many vertices from the SCAN phase would be interconnected. The rule said that there should be no edge between two hyper-alerts belonging to the same phase. The second rule resulted in the overall elimination of cycles. The rule was defined so that an oriented edge can only go from an alert that is in a lower phase to an alert in the higher phase. As a result, there are no cycles or bidirectional edges in the resulting graph and the Bayesian network.

After applying the rules, a Bayesian network was created. The following sections display an example of usage of the methods presented in the previous section. The data from Thursday were used to introduce the results of the individual implemented methods. This approach aims to create a Bayesian network designed to predict cyber attacks. The data from this day are described in detail. After that, the aggregation, causal relationship discovery, and Bayesian network construction methods are shown, along with other auxiliary methods.

After the Bayesian network was constructed and written into a file, the last step of the algorithm followed. The Python library pgmpy was used to load the network from the file with BIF format. After that, the Bayesian inference could finally be calculated, and the probabilities of attack steps could be computed. Our Bayesian network tells us how likely it is that an alert from the final stages will occur if the occurrence of an alert from the first phase, i.e., the scanning phase, has been detected. The results are presented in Table 9. In our case, there are two alerts in the Bayesian network belonging to the final phase. These two belong to phase number 4b, which symbolizes some malicious tasks.

The first of these alerts that we can see in the graph under ID - A3 is *GPL NETBIOS SMB-DS IPC$ share access*. This alert symbolizes the establishment of a connection using the samba protocol. It is meant to detect share access from outside the network. In this case, for example, it may be an EternalBlue infection. This exploit uses a vulnerability in Microsoft's implementation of the SMB (Server Message Block) protocol for remote code execution. However, this type of network activity is often a false positive case. It can be a null session attack on samba functionality, which enables anonymous access to hidden administrative shares on a system. On the other hand, it may be legitimate network traffic; for example, traffic to a domain controller. This alert emerges if the rule defined in the Snort detection system is fulfilled.

Table 9. The probability of reaching the final stage provided that specific stages occur.

Late Stage	Condition	Probability	Late Stage	Condition	Probability
A3	A13	0.9998	A24	A13	0.6989
	A1	0.9995		A1	0.7059
	A6	0.9995		A6	0.7081
	A26	0.9995		A26	0.7081
	A0	0.9996		A0	0.7049
	A31	0.9995		A31	0.7059
	A19	0.9996		A19	0.7059
	A28	0.9995		A28	0.7081
	A21	0.9995		A21	0.7059
	A7	0.9997		A7	0.7189
	A10	0.9997		A10	0.7189
	A15	0.9997		A15	0.7189
	A8	1.0000		A8	0.6985
	A2	1.0000		A2	0.6985
	A33	1.0000		A33	0.6985
	A16	0.9992		A16	1.0000

The second alert that belongs to the final phase is *NF - Bad TLD domain - click DNS query - Check domains*. It can be recognized under ID - A24 in the graph. This means that the device has accessed a domain whose TLD is marked as malicious. A top-level domain can be lablled as bad when there were some indications that it was tied to spam or malware dissemination. Websites using the new top-level domains, such as .men, .work, or .click, are some of the riskiest. The rule that was made for this detection is presented next.

The mentioned alerts and all others emerged from the rules defined in Snort IDS. As was mentioned before, two sets of rules were used—NF and ET.

Next, the types of paths in the graph will be described based on a certain grouping of alerts. Based on the groups of alerts from the first phase according to the type of attack, paths were created in the graph. Their probabilities were calculated using the Bayesian network. Three groups of alerts belonging to the SCAN phase were created.

6.1. *SCAN Suspicious Inbound to SQL*

The first group of alerts contains all of the SQL-related alerts. All of them cohere with SQL port scans. The alerts belonging to this group are:

- A1—msg: "ET SCAN Suspicious inbound to mySQL port 3306",
- A19—msg: "ET SCAN Suspicious inbound to PostgreSQL port 5432",
- A21—msg: "ET SCAN Suspicious inbound to Oracle SQL port 1521",
- A31—msg: "ET SCAN Suspicious inbound to MSSQL port 1433".

All of the alerts can proceed into both of the second-stage alerts. Alerts in this phase show that a payload of at least 100 bytes has been transmitted into and out of the network:

- A12—msg: "NF - Echo Reply Payload - Bigger than 100 bytes", and
- A34—msg: "NF - Echo Request Payload - Bigger than 100 bytes".

From this point, the attack can end in either one of the final stages. If the attack's path contained an alert with number 12, it will certainly end in an alert with number 3. If the path went through alert 34, the attack can end in any of the final stages:

- A3—msg: "GPL NETBIOS SMB-DS IPC$ share access", and
- A24—msg: "NF - Bad TLD domain - click DNS query - Check domains".

Table 10 shows the probabilities of occurrence of final stages based on some of the paths in the graph that contain the first and the second stages.

Table 10. Probabilities of paths including alerts from the group SCAN Suspicious inbound to SQL.

Observed Alerts		**Final Stage Alert**	**Probability**
The First Stage	**The Second Stage**		
A1	A12	A3	1.0000
A1	A34	A3	0.9998
A1	A12	A24	0.7105

After summarizing the results of these calculations, we get clear paths and ends of probable attacks. In the first phase, the SQL service (ports 3306,5432,1521,1433) is scanned from the external network. Subsequently, the payload (>100 bytes) is downloaded or sent. Next, the malicious activity has been executed—access from the external network to the samba protocol or demand for a bad TLD domain (e.g., .tk, .fit, or .rest [43]).

6.2. *SCAN 1*

The second group contains alerts that did not connect to all of the subsequent stages. All of the alerts only continued into one of the second-stage alerts. The contents of this group include:

- A7—msg: "NF—SCAN nmap XMAS",

- A10—msg: "ET—SCAN NMAP OS Detection Probe", and
- A15—msg: "NF—SCAN NMAP OS Detection Probe".

As mentioned, all of the alerts can only go to a second-stage alert with ID A34. This alert was described in the previous group. The message of this variable is 'NF - Echo Request Payload - Bigger than 100 bytes'.

From this point, the attack can proceed and end in any of the final stages. Both of them have also been described in the previous group's definition. An example of the attack path that contains the first and the second stage and ends in the fourth stage can be seen in Table 11.

Table 11. Probabilities of paths including alerts from the group SCAN 1.

Observed Alerts		**Final Stage Alert**	**Probability**
The First Stage	**The Second Stage**		
A7	A34	A3	0.9999
A7	A12	A24	0.7249

From these findings, it is possible to deduce the path by which an attacker can lead to their malicious activity. The first phase is scanning of some device. It can either be an NMAP scan that sets the FIN, PSH, and URG flags in the packet, or an NMAP scan that tries to detect the target operating system. After this, packets with payloads bigger than 100 bytes are downloaded or sent. The attack can end in two ways—connection to the samba protocol from an external network or requesting one of the bad TLD domains.

6.3. SCAN 2

This group of alerts are connected to every one of the subsequent stages. It contains NMAP scans or VNC scans. This group includes:

- A0—msg: "NF - SCAN NMAP -sS 1024 Window",
- A6—msg: "ET SCAN Potential VNC Scan 5800–5820",
- A13—msg: "NF - SCAN nmap fingerprint attempt",
- A26—msg: "ET SCAN Potential VNC Scan 5900–5920", and
- A28—msg: "NF - SCAN Potential VNC Scan 5800–5820".

As was described before, the attack from this group of alerts can proceed in more than one path. The second stage is the same as that in the SCAN Suspicious inbound to the SQL group. The attack can end in both of the mentioned final stages that were described before. An example of an attack that starts with scanning of the designated ports can be seen in Table 12.

Table 12. Probabilities of paths including alerts from the group SCAN 2.

Observed Alerts		**Final Stage Alert**	**Probability**
The First Stage	**The Second Stage**		
A0	A12	A3	1.0000
A0	A34	A3	0.9998
A0	A12	A24	0.7098

From these data, it was concluded that the attack could proceed as follows. The goal of the first phase of the attack was to scan a wide range of ports to find some open vulnerability. It could be scanning all of the designated ports (0–1024) so the attacker can find out what services run on the device and if any of them are vulnerable. The attacker can also be scanning Virtual Network Computing (VNC) ports, which belong to a graphical desktop sharing system. The second phase of the attack is once again sending or downloading big payloads of data, which is suspicious. The attack could end in any of the final stages. The more probable result is that the attacker found an open, vulnerable Server Message Block (SMB) port and tried to exploit this vulnerability.

Since the dataset was not complex and did not contain a wide range of data, further studies are needed in order to test this approach. A larger dataset with an emphasis on attack stages is needed so the proposed methods can be further verified. In this evaluation, we presented some of the attack paths that are likely to happen in real traffic. From the data that were collected, it was discovered that many of the attackers use malicious domains to execute an attack. It was also concluded that some of the ports, like SMB ports, that are considered vulnerable are very often the target of an attack.

7. Conclusions

Prompt response to security incidents means minimizing the damage caused by the security incidents to the organization. The main goal of the organization is maximum preparation for handling security incidents, as well as their prevention through proactive activities. The transition from reactive activities to proactive activities is currently a challenge in cybersecurity research.

The attack—or the steps of the attackers—can be divided into several stages. Consequently, one of the ways to move from reactive activities to proactive activities is to identify the initial stages of the attack. To be able to make such an identification, it is necessary to predict the next steps of the attacker, which is called the projection of attacks in the current research.

Within this paper, we focused on the projection of attacks, and for this purpose, we defined four stages of attacks. The aim of the paper was the prediction of the final stages (the third and fourth stages), provided that we knew the first two stages of the attack. For this purpose, we chose Bayesian networks.

The aim was to design a model that, based on the projection of the attack, would identify the early stages of the attack. The proposed model includes not only the aggregation of alerts, but also their correlation. We used the proposed attack model for the prediction itself. As research has shown, the paper [8] on which our research is based and was the inspiration for the use of the Bayesian network does not take into account the situation where an attacker can return within a particular stage. The situation creates a cyclic graph, which creates a problem, since the Bayesian network assumes an acyclic graph as the input. For this reason, it was necessary to add a condition to the model under which the attacker does not return within the individual stages.

We tested the proposed model on the publicly available Intrusion Detection Evaluation Dataset CICIDS2017. In the paper, we showed the real application of the proposed model on a prepared dataset, including the creation of alerts, their aggregation and correlation, and the subsequent detection of early stages based on the attack projection.

This research can be expanded in the future. There are several challenges for future work. One of them is the processing and prediction of events, even if there are cycles in the attack graph. This case is problematic because many computational models that take into account acyclic graphs cannot be used. Therefore, it could be appropriate to try another method. For example, using attack graph modeling or a hidden Markov model would be successful. The second challenge that this research presents is creating a complex and comprehensive dataset. Nowadays, to the best of our knowledge, no suitable datasets have been created that emphasize the attacks. Therefore, it is important to work with a dataset that would contain attacks that cover all detectable stages of an attack.

Author Contributions: Conceptualization, M.P., T.B. and P.S.; methodology, M.P. and T.B.; data processing, M.P. and T.B.; results analysis, M.T. and P.S.; writing—original draft preparation, M.P.; writing—review and editing, M.P., P.S. and T.B.; supervision, P.S. All authors have read and agreed to the published version of the manuscript.

Funding: This paper is funded by the project VVGS-PF-2019-1062, project VVGS-PF-2020-1427, and the Slovak APVV project under contract No. APVV-17-0561.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. FireEye. Common Vulnerability Scoring System. 2018. Available online: https://www.fireeye.com/content/dam/collateral/en/mtrends-2018.pdf (accessed on 19 November 2020).
2. Husák, M.; Komárková, J.; Bou-Harb, E.; Čeleda, P. Survey of attack projection, prediction, and forecasting in cyber security. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 640–660. [CrossRef]
3. Yang, S.J.; Du, H.; Holsopple, J.; Sudit, M. Attack projection. In *Cyber Defense and Situational Awareness*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 239–261.
4. Ahmed, A.A.; Zaman, N.A.K. Attack Intention Recognition: A Review. *IJ Netw. Secur.* **2017**, *19*, 244–250.
5. Abdlhamed, M.; Kifayat, K.; Shi, Q.; Hurst, W. Intrusion prediction systems. In *Information Fusion for Cyber-Security Analytics*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 155–174.
6. Leau, Y.B.; Manickam, S. Network security situation prediction: A review and discussion. In *International Conference on Soft Computing, Intelligence Systems, and Information Technology*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 424–435.
7. Mézešová, T.; Sokol, P.; Bajtoš, T. Evaluation of Attacker Skill Level for Multi-stage Attacks. In Proceedings of the 2019 11th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Pitesti, Romania, 27–29 June 2019; pp. 1–6.
8. Ramaki, A.A.; Khosravi-Farmad, M.; Bafghi, A.G. 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), Rasht, Iran, 2–3 September 2015; pp. 98–103.
9. Swiler, L.P.; Phillips, C. *A Graph-Based System for Network-Vulnerability Analysis*; Technical Report; Sandia National Labs.: Albuquerque, NM, USA, 1998.
10. Cao, P.; Chung, K.W.; Kalbarczyk, Z.; Iyer, R.; Slagell, A.J. Preemptive intrusion detection. In Proceedings of the 2014 Symposium and Bootcamp on the Science of Security, Raleigh, NC, USA, 8–9 April 2014; p. 21.
11. Cao, P.; Badger, E.; Kalbarczyk, Z.; Iyer, R.; Slagell, A. Preemptive intrusion detection: Theoretical framework and real-world measurements. In Proceedings of the 2015 Symposium and Bootcamp on the Science of Security, Urbana-Champaign, IL, USA, 21–22 April 2015; p. 5.
12. Ramaki, A.A.; Amini, M.; Atani, R.E. RTECA: Real time episode correlation algorithm for multi-step attack scenarios detection. *Comput. Secur.* **2015**, *49*, 206–219. [CrossRef]
13. Ning, P.; Xu, D. Learning attack strategies from intrusion alerts. In Proceedings of the 10th ACM Conference on Computer and Communications Security, Washington, DC, USA, 27–30 October 2003, pp. 200–209.
14. Li, Z.; Lei, J.; Wang, L.; Li, D. A Data Mining Approach to Generating Network Attack Graph for Intrusion Prediction. In Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007), Haikou, China, 24–27 August 2007; Volume 4, pp. 307–311.
15. Liu, P. *A Game Theoretic Approach to Cyber Attack Prediction*; Technical Report; Pennsylvania State University: University Park, PA, USA, 2005.
16. Wu, J.; Yin, L.; Guo, Y. Cyber attacks prediction model based on Bayesian network. In Proceedings of the 2012 IEEE 18th International Conference on Parallel and Distributed Systems, Singapore, 17–19 December 2012; pp. 730–731.
17. Ishida, C.; Arakawa, Y.; Sasase, I.; Takemori, K. Forecast techniques for predicting increase or decrease of attacks using bayesian inference. PACRIM. In Proceedings of the 2005 IEEE Pacific Rim Conference on Communications, Computers and signal Processing, Victoria, BC, Canada, 24–26 August 2005; pp. 450–453.
18. Okutan, A.; Yang, S.J.; McConky, K. Predicting cyber attacks with bayesian networks using unconventional signals. In Proceedings of the 12th Annual Conference on Cyber and Information Security Research, Kowloon, Hong Kong, China, 15–16 August 2017; p. 13.
19. Passeri, P. HACKMAGEDDON, Information Security Timelines and Statistics. 2016. Available online: https://www.hackmageddon.com/ (accessed on 19 November 2020).
20. Tabia, K.; Leray, P. Bayesian network-based approaches for severe attack prediction and handling IDSs' reliability. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*; Springer: Berlin/Heidelberg, Germany, 2010, pp. 632–642.
21. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*; Elsevier: Amsterdam, The Netherlands, 2014.

22. Farhadi, H.; AmirHaeri, M.; Khansari, M. Alert correlation and prediction using data mining and HMM. *ISeCure* **2011**, *3*, 77–101.
23. Sendi, A.S.; Dagenais, M.; Jabbarifar, M.; Couture, M. Real time intrusion prediction based on optimized alerts with hidden Markov model. *J. Networks* **2012**, *7*, 311.
24. Shin, S.; Lee, S.; Kim, H.; Kim, S. Advanced probabilistic approach for network intrusion forecasting and detection. *Expert Syst. Appl.* **2013**, *40*, 315–322. [CrossRef]
25. Holgado, P.; Villagrá, V.A.; Vazquez, L. Real-time multistep attack prediction based on hidden markov models. *IEEE Trans. Dependable Secur. Comput.* **2017**, *17*, 134–147. [CrossRef]
26. Hutchins, E.M.; Cloppert, M.J.; Amin, R.M. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. *Lead. Issues Inf. Warf. Secur. Res.* **2011**, *1*, 80.
27. Bou-Harb, E.; Debbabi, M.; Assi, C. A systematic approach for detecting and clustering distributed cyber scanning. *Comput. Networks* **2013**, *57*, 3826–3839. [CrossRef]
28. Caltagirone, S.; Pendergast, A.; Betz, C. *The Diamond Model of Intrusion Analysis*; Technical Report; Center For Cyber Intelligence Analysis and Threat Research: Hanover, MD, USA, 2013.
29. Bou-Harb, E.; Debbabi, M.; Assi, C. Cyber scanning: a comprehensive survey. *IEEE Commun. Surv. Tutor.* **2013**, *16*, 1496–1519. [CrossRef]
30. Yadav, T.; Rao, A.M. Technical aspects of cyber kill chain. In *International Symposium on Security in Computing and Communication*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 438–452.
31. Available online: https://www.mycert.org.my (accessed on 19 November 2020).
32. Cooper, G. An overview of the representation and discovery of causal relationships using Bayesian networks. In *Computation, Causation, and Discovery*; MIT Press: Menlo, CA, USA, 1999; pp. 4–62.
33. Guo, H.; Hsu, W. A Survey of Algorithms for Real-Time Bayesian Network Inference. 2002. Available online: https://www.aaai.org/Papers/Workshops/2002/WS-02-15/WS02-15-001.pdf (accessed on 19 November 2020).
34. Friedman, N.; Geiger, D.; Goldszmidt, M. Bayesian network classifiers. *Mach. Learn.* **1997**, *29*, 131–163. [CrossRef]
35. Qin, X.; Lee, W. Attack plan recognition and prediction using causal networks. In Proceedings of the 20th Annual Computer Security Applications Conference, Tucson, AZ, USA, 6–10 December 2004; pp. 370–379.
36. Zhang, N.L.; Poole, D. A simple approach to Bayesian network computations. In Proceedings of the Biennial Conference-Canadian Society for Computational Studies of Intelligence, Banff Park Lodge, Banff, Alberta, 16–20 May 1994; pp. 171–178.
37. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. 2018. Available online: https://www.scitepress.org/Papers/2018/66398/66398.pdf (accessed on 19 November 2020).
38. Intrusion Detection Evaluation Dataset (CICIDS2017), U.o.N. Brunswick. 2017. Available online: https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 19 November 2020).
39. Snort - Network Intrusion Detection & Prevention System. Available online: https://www.snort.org (accessed on 19 November 2020).
40. SNORT Users Manual 2.9.13. The Snort Project 2019. 2019. Available online: https://www.snort.org/documents/snort-users-manual (accessed on 19 November 2020).
41. Hansson, L. NF IDS Rules. Available online: https://networkforensic.dk/SNORT/ (accessed on 19 November 2020).
42. Emerging Threats rules. Available online: https://rules.emergingthreats.net/ (accessed on 19 November 2020).
43. What Is Cyber Attack? 2018. Available online: https://www.upguard.com/blog/cyber-attack (accessed on 19 November 2020).