applied sciences

MDPI

Article

Online Mining Intrusion Patterns from IDS Alerts

Kai Zhang [1,*], Shoushan Luo [1], Yang Xin [1,2,*], Hongliang Zhu [1] and Yuling Chen [2]

[1] National Engineering Laboratory for Disaster Backup and Recovery, Information Security Center, School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China; buptlou@bupt.edu.cn (S.L.); zhuhongliang@bupt.edu.cn (H.Z.)

[2] Guizhou Provincial Key Laboratory of Public Big Data, Guizhou University, Guizhou 550025, China; ylchen3@gzu.edu.cn

* Correspondence: kaikai4006@163.com (K.Z.); yangxin@bupt.edu.cn (Y.X.)

Received: 22 March 2020; Accepted: 19 April 2020; Published: 24 April 2020

Featured Application: In this paper, an influence model is proposed to tackle the sequence data analysis problems such as disordering, element missing and random noises. The proposed method can be used for mining intrusion patterns from the intrusion action sequence extracted from IDS (Intrusion Detection System) alerts.

Abstract: The intrusion detection system (IDS) which is used widely in enterprises, has produced a large number of logs named alerts, from which the intrusion patterns can be mined. These patterns can be used to construct the intrusion scenarios or discover the final objectives of the malicious actors, and even assist the forensic works of network crimes. In this paper, a novel algorithm for the intrusion pattern mining is proposed which aimsto solve the difficult problems of the intrusion action sequence such as the loss of important intrusion actions, the disorder of the action sequence and the random noise actions. These common problems often occur in the real production environment which cause serious performance decrease in the analyzing system. The proposed algorithm is based on the online analysis of the intrusion action sequences extracted from IDS alerts, through calculating the influences of a particular action on the subsequent actions, the real intrusion patterns are discovered. The experimental results show that the method is effective in discovering pattern from the complex intrusion action sequences.

Keywords: IDS (Intrusion Detection System) alerts; intrusion pattern; intrusion scenarios; correlation analysis; intrusion detection; attack scenario; online mining; sequence learning; pattern mining

1. Introduction

According to the kill chain proposed by Bryant et al. [1], the inside multi-step attack has become the main part of the intrusion process, which includethe pre hack (reconnaissance and delivery), hack (installation and privilege escalation), compromise (lateral movement, actions on objective) and theft (exfiltration). Currently, many sophisticated intrusion attacks [2–4] start from the inside of victim networks through the spear-phishing email or the host with vulnerabilities. Once the victim host executes the malicious code, the attacker begins to explore the whole network to discover the resources they need for further attacking. The attacker will hide the traits and avoid detection during intrusion [5,6]. In a word, it is critical to enhancing the existing multi-step attack detection abilities inside the network.

However, the multi-step attack detection approaches are mostly based on the intrusion detection system (IDS) sensors deployed in the hosts or the entrance of the network and are facing more severe challenges: (1) high false positives, which leads to the insertion of irrelevant intrusion actions into the data stream; (2) high redundancy, which is caused by the IDS detection mechanisms or the intended

intrusion strategies; (3) incomplete data, which is a common situation in the real environment due to the network delay or the system error; (4) disordered data, which is caused by the intended intrusion strategies [7] or multiple parallel attacking paths. These issues will cause the failure of the methods based on sequence learning.

Recently, researchers have been making their systems more lightweight, more sensitive to the threats, and capable of online analysis and processing of large-scale streaming data with noises and errors, rather than learning outdated patterns based on limited historical data. They hope that the whole system can automatically adapt to the variances of data and capture the known or unknown threat patterns in an unsupervised way [8]. This is one of the most important studies in the intrusion detection field.

Modeling the multi-step attack is a process of gathering the evidence extracted from the network logs or IDS alerts, to find out how the attack might transpire over time, it is a broader concept than traditional intrusion detection [9]. In the whole process of attack scenario construction, a few kinds of literatures focus on the data preprocessing such as alert normalization, alert aggregation, noise reduction, and hyper-alert extraction, while others focus on the correlation analysis, intrusion pattern discovery, attack scenario construction and attack prediction [10]. Unfortunately, few of them are dedicated to addressing the challenges described above.

Liu et al. [11] proposed a framework for reconstructing the attack scenarios based on the reasoning methods, which can deal with the incomplete evidence using the known vulnerabilities database and other expert knowledge, but it was difficult for them to work out the missing part of the unknown attack scenario.

Angelini et al. [12] proposed a graph-based online multi-step attack detector which can detect the on-going attacks early enough for managers to take proper countermeasures, and a visualization interface was developed to represent comprehensive network situations. The preprocessing of the sensor data was not described, and it was also based on the known vulnerabilities, besides, there was no evaluation based on unknown intrusion patterns.

Shen et al. [13] noticed the problems caused by the incomplete, disordered intrusion action sequences. They proposed a framework named Tiresias which is based on the RNN (Recurrent Neural Networks) algorithm, which can effectively deal with the disordered alert stream. Although Tiresias can calculate the probabilities of multiple actions that may happen in the future, it is trained based on the pre-labeled events, the labeled data imply that the framework is based on the known attack analysis.

Haas et al. [14] proposed a framework for multi-step attack detection by alert correlation process. The alert clustering is leveraged to reduce the number of alerts, and highlight the intrusion actions. The communication patterns are identified based on the clusters, and then the graph-based alert correlation (GAC) algorithm is applied to realize the alert correlations. In addition, the clusters will be labeled based on the vulnerabilities database, and then correlated together with IP addresses. The irrelevant intrusion actions will mix in the discovered attack scenarios due to the correlation based on IP.

In this paper, the backward influence factor (BIF) algorithm is proposed aiming to overcome the problems caused by the disordered, incomplete and noisy IDS logs in a real-time manner. It is a sequence pattern mining algorithm, which is suitable for analyzing the streaming data generated online by IDS or other devices. The BIF algorithm is evaluated in the context of a multi-step attack scenario discovery task. The whole system is based on IDS alert analysis containing five phases: normalizing, intrusion action extraction, intrusion session pruning, correlation discovery, dynamic correlation graph construction. The first three phases are inherited from our previous work [15] because we want to use the data structures and concepts that have been created before. Each phase is summarized as follows:

In the normalizing phase, it unifies the raw alerts from different types of sensors and converts them into the common data structures that can be solved by the system. After normalizing, the raw alerts are converted into alert objects (alert for short).

In the intrusion action extraction phase, it groups alerts by two fields: the source IP address and the destination IP address. Then the intrusion actions are extracted based on the type field and

the destination port field of alerts derived from the same group. In this phase, most redundant and repeated alerts are merged.

In the intrusion session pruning phase, a long action sequence can be divided into several short sequences (intrusion sessions) by calculating the average time interval of actions. Then a pruning process begins to remove the repeat sub-patterns from the original sequence. The pruning algorithm can significantly reduce the length of the intrusion sessions without destroying the original associations of actions.

In the correlation discovery phase, all the pruned sessions will be fed to the correlation discovery module according to the start time of the session. The BIF algorithm is applied to calculate the attraction levels between any two actions of the session. The influence factor (IF) values which express the attraction level will increase or decrease with the incoming data over time, then the real association relations are built.

The dynamic correlation graph (DCG) is constructed based on the discovered correlations with higher IF values, the DCG links and nodes are dynamically created or destroyed with the influence factor matrix (IFM) which is the matrix maintaining all IF values and updating them in real-time.

In this paper, the proposed BIF algorithm will be introduced in detail, it can be leveraged either as an optimized method for the intrusion scenario discovery task of our former work [15] or a separate intrusion pattern mining system.

The proposed algorithm is based on the assumptions: the distance of two actions in a session can be used to measure the association strength of them.

2. Materials and Methods

The network environment is always complex and unpredictable, the problems caused by the network environment can badly impact the network security systems. In addition, the IDS can also cause problems such as redundant alerts and repeated patterns.

As shown in Figure 1, S1 is the correct intrusion session extracted under the experimental environment, while S2, S3, and S4 are three different states in a running environment. The action sequence is altered due to the different configurations of the network and the security systems. Therefore, it is necessary to pay attention to these problems and try to minimize their impact.

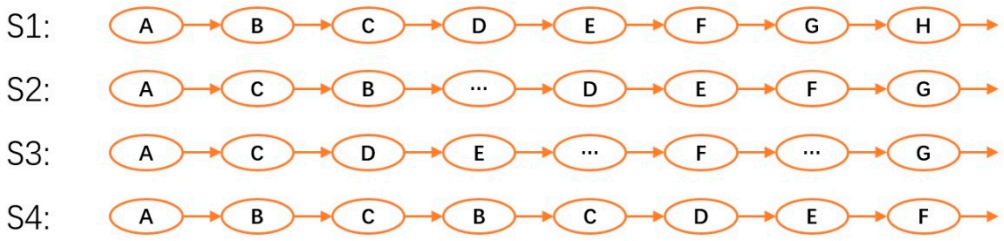

Figure 1. The illustration of the real intrusion sessions. S1, S2, and S3 are three intrusion sessions; A-H is intrusion actions that composed an intrusion scenario; S1 denotes the original session state; S2 is disordered and some irrelevant actions engaged; B is lost in S3; B-C in S4 is repeated due to the intrusion detection system (IDS).

For some IDS introduced problems, a few methods were proposed in our previous work [16,17], the redundant alerts, for example, can be reduced in the action extraction phase, and a few repeated action patterns can be removed in the session pruning phase by the pruning algorithm. In this paper, we mainly focus on the incomplete and disordered session (sequence) learning and attack pattern discovery in real-time.

Definitions

The problem domain can be formalized as follows. An intrusion action $a_i \in A$ consists of a group of alerts, where A denotes the set of all unique actions, and |A| denotes the size of A. An intrusion

session which is a sequence of actions ordered by their time field, $s_i^{(xy)} = \left\{a_{i1}^{(xy)}, a_{i2}^{(xy)}, \ldots, a_{in}^{(xy)}\right\}$ where x and y denote the two hosts from which the session is extracted.

It is assumed that an intrusion action has an attraction effect on the subsequent actions, and a particular action happens due to the attractions of one or more other actions which happened in different sessions. The degree of influence can be calculated and used to measure the association strength between actions.

For a given session $s_i^{(xy)} = \left\{a_{i1}^{(xy)}, a_{i2}^{(xy)}, \ldots, a_{in}^{(xy)}\right\}$, $a_{i1}^{(xy)}$ has a direct influence on $a_{i2}^{(xy)}$, and indirect influence on $a_{i3}^{(xy)}$, the degree of influence will get lower with the longer distance between $a_{i1}^{(xy)}$ and $a_{in}^{(xy)}$. The influence range is $1 \le \omega \le w$, where w is the number of actions influenced. For a given ω, if an intrusion action B falls in the influence range of A, A influents B (A attracts B) which can be denoted as $A \mapsto B$. The algorithm will calculate the influences on the actions that fall in the influence range of specified action in the session. The influence range is shown in Figure 2.

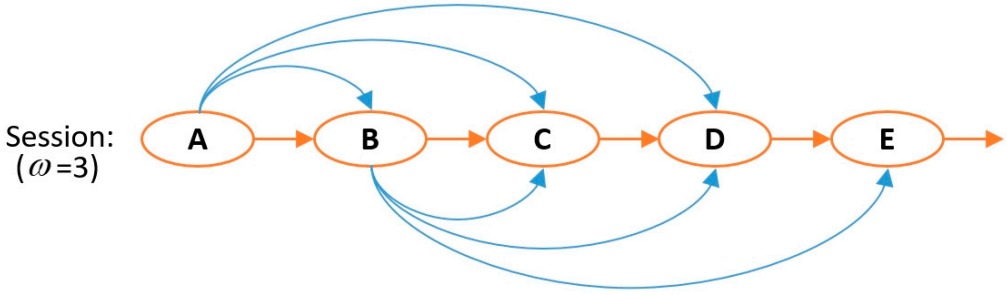

Figure 2. The influence range of A and B in a session when $\omega = 3$.

The influence degree of a_i on a_j can be calculated using Equation (1):

$$F(a_i \mapsto a_j) = \frac{2}{j - i + 1} \quad (j > i), \tag{1}$$

where a_i and a_j denotes the ith and jth action of the session, respectively, ω is the influence range, and |s| denotes the length of the session. The more actions exist between two actions, the lower their influence is and the lower their association strength is. The influence of each pair of actions will update the old values recorded in the influence matrix shown in Figure 3. In the matrix, the values in a row express the attraction strengths of the particular action on other actions, and the values in a column express the attraction strengths of other actions on the particular action.

	a1	a2	a3	a4	a5	a6	a7	a8
a1	0	0.26	0.35	0.21	0.36	0.28	0.1	0.22
a2	0.05	0	0.24	0.37	0.29	0.1	0.33	0.24
a3	0.03	0.31	0	0	0.18	0	0.15	0.38
a4	0.02	0.23	0.39	0	0.21	0	0.3	0.06

Figure 3. The influence matrix. a_1–a_8 indicates intrusion actions. The IF value of a_1 on a_2 is 0.26.

With the continuous arrival of the intrusion sessions, the two actions A and B may have different influence values in different sessions, the old influence in the matrix should be updated with the new values using Equation (2):

$$F_o = F_o + \alpha(F_n - F_o) \quad (0 < \alpha < 1), \tag{2}$$

where α indicates the refreshing rate, F_o denotes the original value of influence recorded in the matrix, F_n denotes the newly calculated influence value. Note that, the calculated F_o may increase or decrease.

The number of sessions contains $A \mapsto B$ can be denoted as $f(A \mapsto B)$, and the total number of sessions in an analyzing period can be denoted as $|S|$. The probability of $A \mapsto B$ can be calculated using Equation (3):

$$P(A \mapsto B) = \frac{f(A \mapsto B)}{|S|}, \tag{3}$$

The comprehensive influence of intrusion action A on B can be calculated using Equation (4):

$$CF(A \mapsto B) = F(A \mapsto B) \times P(A \mapsto B), \tag{4}$$

where with Equation (4), the influence strength of A on B can be calculated based on the historical observation.

For a given intrusion action A, what predictions will the algorithm make? First, the influenced actions by A will be collected as the candidate predictions, second, the comprehensive influences are calculated, and the strongly influenced actions will be selected and predicted.

3. Results

In this section, the evaluation results of the proposed algorithm are discussed. Four types of datasets are generated automatically and the algorithm is tested based on them. The experimental results are discussed separately.

3.1. Evaluations On The Automatically Generated Dataset

We extracted the intrusion sessions based on the CICIDS2017 intrusion detection evaluation dataset [18] in the previous work [15], but the extracted sessions are not complicated and varied enough to evaluate the proposed method. In addition, we mainly focused on the sequence analyzing performance and intrusion pattern discovery accuracy, rather than the data preprocessing, however, an effective data preprocessing can increase the capacity of the proposed method.

According to the characteristics of intrusion patterns extracted from real data in our former work, we generated multiple datasets with each having different types of defects described in Section 2.

3.1.1. Baseline Dataset

First, we generate a standard baseline dataset for comparison tests with other datasets. The baseline dataset contains 10 different intrusion patterns (action sequences) involving 90 unique actions and 500 random background action sequences. Each intrusion pattern shares no common actions with others, and contains no irrelevant actions, in a word, they have no data problem. The details of the generated dataset are listed in Table 1.

Table 1. The baseline dataset.

Characteristics	Value	Note
Pattern sessions	10	Need to be learned and predicted
Random traffic	500	Background sessions
Unique actions	190	Total number of unique actions in the dataset
Missed actions	0	Number of actions lost in each pattern
Disordered actions	0	Actions changed their positions in the session
Shared actions	0	Actions shared in any two pattern sessions

The pattern sessions used to generate the baseline dataset are listed in Table 2:

Table 2. Generated pattern sessions.

Pattern Sessions	Appear Times [1]
a1, a2, a3, a4, a5, a6, a7, a8, a9, a10, a11, a12, a13, a14	5
a15, a16, a17, a18, a19, a20	5
a21, a22, a23, a24, a25, a26, a27, a28, a29	5
a30, a31, a32, a33, a34, a35, a36, a37, a38, a39	4
a40, a41, a42, a43, a44, a45, a46, a47, a48, a49	5
a50, a51, a52, a53, a54, a55	4
a56, a57, a58, a59, a60, a61, a62, a63, a64	4
a65, a66, a67, a68, a69, a70, a71, a72	5
a73, a74, a75, a76	3
a77, a78, a79, a80, a81, a82, a83, a84, a85, a86, a87, a88, a89, a90	5

[1] Appear times are random in each round of test.

We first tested the method with the baseline data set. After learning the dataset, the different actions were fed to the algorithm to check the prediction results by calculating the top average influences. The result shows that the algorithm can predict the future action sequence accurately due to the clean session patterns. The test results are listed in Table 3.

Table 3. Prediction results based on the baseline dataset.

Inputs	TopPredicted Actions (Average Influence Factor)
a1	a2(0.020), a3(0.013), a4(0.010), a5(0.008), a6(0.007)
a6	a7(0.020), a8(0.013), a9(0.010), a10(0.008), a11(0.007)
a16	a17(0.027), a18(0.018), a19(0.013), a20(0.011)
a24	a25(0.033), a26(0.022), a27(0.017), a28(0.013), a29(0.011)
a33	a34(0.027), a35(0.018), a36(0.013), a37(0.011), a38(0.009)
a40	a41(0.032), a42(0.027), a43(0.023), a44(0.020)
a52	a53(0.027), a54(0.018), a55(0.013)
a61	a62(0.033), a63(0.022), a64(0.017)
a68	a69(0.020), a70(0.013), a71(0.010), a72(0.008)
a83	a84(0.027), a85(0.018), a86(0.013), a87(0.011), a88(0.009)

Based on the data shown in Table 3, the algorithm works well with the baseline dataset. The top average influence will be used to determine what is probably the next action. The baseline dataset is only used to check whether our thinking is feasible.

In the following evaluations, three datasets each with a different data defect are generated and used to test the proposed algorithm. In the end, a dataset with all three types of data defects are generated to simulate the real data environment, and the BIF algorithm is fully tested.

3.1.2. Noisy Dataset

By only inserting random irrelevant actions into each pattern session of the baseline dataset to generate a noisy dataset, the number of noise action inserts into each of the pattern sessions can be specified. We inserted 1–10 noise actions (10 noise levels) into the random positions of each pattern session to observe the prediction errors which are compared to the baseline dataset. There were 10 tests for each noise level, for each test, the data set was regenerated to ensure sufficient randomness. The system made 10 predictions based on different inputs and the prediction errors were counted. The average influences of each candidate action were calculated and the top n actions with the highest average influences were selected as the prediction results which were compared with the predictions shown in Table 3.

A part of the generated noisy dataset is shown in Figure 4, of which each row contains an intrusion session. The actions that start with “b” are the noise actions, and all the actions are divided by a comma.

a43,b28,a44,b30,b30,b4,b32,b18,b17,a45,b47,b2,b32,a46,b32,b29,b16,b41,b13,a47,a48,b2,a49,b15,b1,b43
a40,b34,b20,b40,b24,b19,a41,b39,b18,b19,b46,b26,b36,b33,b46,b44,b15,b17,a42,b17,b3,b41
a43,b38,b40,b20,b25,b22,b40,a44,b16,b36,b18,b34,b21,a45,b8,a46,b36,b30,a47,a48,b7,b49,a49,b10,b41,b45
a56,a57,b22,b26,b16,a58,b42,b46,b5,b10,a59,b11,b5,b22,a60,b27,b47,b20,b30,a61,b46,b44,a62,b19,a63,a64,b25,b25
a40,b23,b47,b45,b32,b23,b10,b18,b29,b7,b21,b35,b32,b22,b36,b7,b14,b42,a41,b33,b40,a42
a65,b30,a66,b25,b27,a67,b32,b8,b9,b47,b37,b16,b7,a68,b18,b38,a69,a70,b19,a71,b16,b29,b8,b50,a72,b19,b11
a73,b14,b48,a74,b4,b29,b22,b24,b5,b5,b49,a75,b4,b43,b40,b16,b13,b7,b22,b12,b39,b15,a76
a73,b48,b49,b39,b14,b33,b19,b34,b42,a74,b26,b14,b42,b38,b36,b47,b28,b32,a75,b7,a76,b45,b45
a15,b17,b39,b34,a16,b31,b46,a17,b35,b34,b12,b47,a18,b31,b33,b30,b21,b26,b2,b3,b23,a19,b35,b13,a20
a21,b38,a22,b27,b28,a23,b18,a24,a25,b40,b14,b29,b44,b13,b31,b28,b33,b37,b25,b46,a26,a27,b9,b35,b28,b13,a28,a29
a56,b28,b11,b5,b31,a57,a58,a59,b22,b12,a60,b5,b23,b6,b33,b46,b27,a61,b21,a62,b40,b40,b38,a63,b17,a64,b23,b34
a30,b44,b9,b31,b44,a31,a32,b9,a33,b9,b36,a34,b48,a35,b5,b2,b5,a36,a37,b10,b20,b33,b28,a38,b32,b22,a39,b15,b25
a50,b21,b26,b19,b40,b12,b39,a51,b43,b32,b31,b2,b37,a52,b38,b13,a53,b46,b34,b41,a54,b7,b15,b14,a55
a50,b4,b26,b5,a51,b21,a52,b36,b41,b17,b2,b3,a53,b6,b25,b20,b31,b39,b46,a54,a55,b4,b29,b20,b47
a56,b8,b35,b49,b32,a57,b15,b38,b49,a58,b36,b10,b32,a59,b38,b12,b44,b27,b24,b49,a60,a61,b40,a62,a63,b14,b36,a64
a15,b10,a16,b19,a17,b47,b25,b21,b22,a18,b19,b29,b8,b40,a19,b10,b35,b3,b19,b36,b29,b6,a20,b2,b5
a73,b40,b8,b33,b44,b14,b29,a74,b41,b46,b45,b23,a75,b24,b6,b22,b9,b7,a76,b22,b5,b14,b38
a50,b30,b18,a51,b43,b16,b9,a52,b21,b44,b37,b49,b29,b12,b29,a53,b35,a54,b24,b39,b44,b12,b14,b9,a55
a30,b15,b28,a31,b34,b44,a32,b50,b49,b22,b8,b33,a33,b40,a34,b32,a35,b49,a36,a37,b36,b30,b46,a38,b25,a39,b18,b23,b7
a65,a66,b21,b27,a67,b25,a68,b39,b2,b47,b4,a69,b36,b12,b10,b29,b32,a70,a71,b6,a72,b41,b19,b31,b12,b2,b15
a15,a16,b13,b39,a17,b5,b20,b34,a18,a19,b35,b22,b27,b26,b16,b26,b49,a20,b43,b19,b49,b34,b11,b28,b20
a65,a66,b2,b4,b38,a67,a68,b10,b18,b38,a69,a70,b14,b32,b21,b2,b14,b20,b33,b48,b33,a71,b20,b22,a72,b25,b40
a77,a78,b46,b13,b28,b2,b17,b41,b35,b1,b44,a79,b38,a80,a81,b19,a82,a83,a84,a85,a86,b48,a87,a88,b23,b16,b16,b16,b29,a89,b19,a90,b43

Figure 4. The generated noisy sessions.

The accuracy of the analysis based on the noisy dataset is shown in Table 4.

Table 4. Prediction results based on the noisy dataset.

Inputs	Prediction Errors	Prediction Accuracy [1]
a1	(0,0,0,0,3,3,4,0,3,7)	98.5%
a6	(0,0,0,1,2,1,0,0,1,2)	99%
a16	(0,0,4,0,3,1,7,6,10,0)	92.3%
a24	(1,0,0,0,1,2,2,2,5,10)	95.4%
a33	(0,0,4,2,1,7,3,12,2,8)	93.3%
a40	(1,0,2,0,1,4,4,2,8,0)	97.6%
a52	(0,0,1,0,1,2,4,2,2,5)	94.3%
a61	(0,0,0,1,0,1,0,2,0,1)	98.3%
a68	(0,0,1,0,0,0,5,4,6,5)	94.8%
a83	(0,0,0,0,0,1,0,2,0,1)	99.4%

[1] $accuracy(action) = (1 - \frac{\#actions\ wrongly\ predicted}{\#actions\ should\ be\ correctly\ predicted}) \times 100\%$.

The evaluation results are shown in Figure 5:

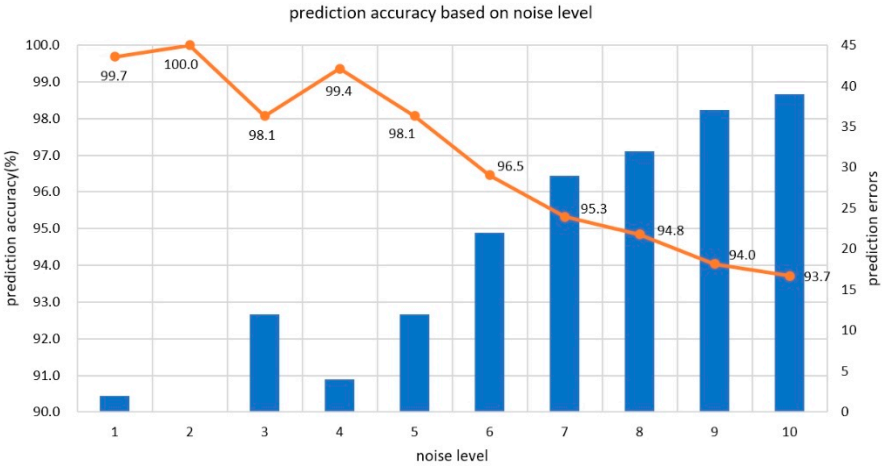

Figure 5. The prediction accuracy based on different noisy levels.

As shown in Figure 5, the average accuracy of the prediction decreases with the increasing of the noise level. When inserting 10 noise actions into each pattern sessions, the prediction accuracy stays above 93.7%. The average accuracy of predictions based on the input actions is 96.3%. In a word, the proposed algorithm can learn the patterns from the noisy sequences with a higher accuracy.

3.1.3. Disordered Dataset

In this section, we will test the algorithm using the disordered dataset generated based on the baseline data. The term disordered in this paper means the action in an intrusion session (action sequence) changed their position due to various reasons—the session is called the disordered session. The action that changed its original position is called the disordered action. The disorder level of a dataset is the average number of disordered actions in each pattern session.

There are five disorder levels, and 10 tests for each disorder level. For each test, two datasets will be generated: the training dataset and the testing dataset, each dataset is generated separately. The disordered actions in each pattern session are randomly selected and moved to a random position in the session. The pattern sessions each with 10 actions are shown in Table 5.

Table 5. The pattern sessions should be discovered by system.

No.	Pattern Sessions	Appear Times
1	a1, a2, a3, a4, a5, a6, a7, a8, a9, a10	5
2	a11, a12, a13, a14, a15, a16, a17, a18, a19, a20	5
3	a21, a22, a23, a24, a25, a26, a27, a28, a29, a30	5
4	a31, a32, a33, a34, a35, a36, a37, a38, a39, a40	5
5	a41, a42, a43, a44, a45, a46, a47, a48, a49, a50	5
6	a51, a52, a53, a54, a55, a56, a57, a58, a59, a60	5

The appear times means the number of appearances of each disordered pattern sessions in the dataset. For each test, an input action will be fed to the system for prediction. The testing results are shown in Table 6.

Table 6. Testing results based on the disordered dataset.

Inputs	Prediction Errors	Prediction Accuracy [1]
a1	(9,7,9,14,10)	89.1%
a11	(7,7,8,11,13)	89.8%
a21	(2,8,8,14,12)	90.2%
a31	(3,7,12,11,15)	89.3%
a41	(5,7,7,8,18)	90%
a51	(5,7,11,11,13)	89.6%

[1] $accuracy(action) = (1 - \frac{\#actions\ wrongly\ predicted}{\#actions\ should\ be\ correctly\ predicted}) \times 100\%$.

There are more prediction errors than the result of the noisy data evaluation. The average accuracy of prediction for each pattern session is 89.7%. The relation between prediction accuracy and the disorder level is illustrated in Figure 6.

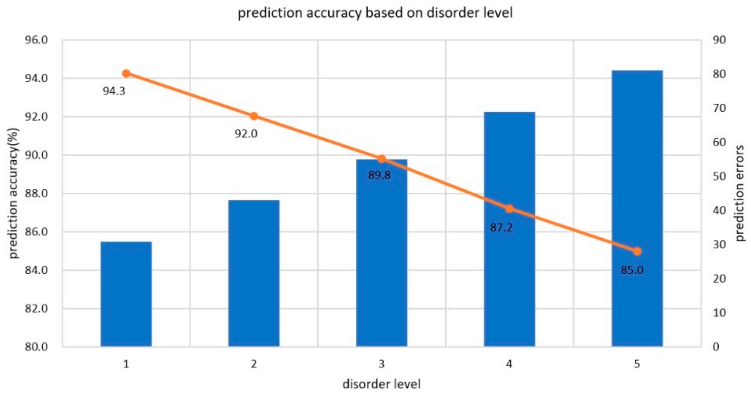

Figure 6. The prediction accuracy based on 5 disordered levels.

The proposed algorithm is more sensitive to the disordered data than the noisy data, that is because it is the major method for the algorithm to learn patterns by accumulating the influence between ordered actions, if the orders of actions are changing frequently, it will be hard for the algorithm to learn any patterns. With 50% disordered actions in each session, the prediction accuracy is 85% which is an acceptable result.

3.1.4. Incomplete Dataset

For the incomplete dataset, which refers to the situation that actions in an intrusion session are lost for various reasons. The missing action will not change the order of other actions in the session; however, it will cause serious problems in the data mining systems and the statistical systems, especially the sequential learning systems.

The proposed method can solve the missing data problem by creating the association relations between actions. The influence relation of two actions becomes stronger if they always appear in the same session and are close to each other, otherwise, the influence relation will weaken. So, the missing action will not impact the learned patterns.

The evaluation test is similar to the disordered data evaluation. The six types of pattern sessions are the same as the ones in Table 5. There are five missing-levels for each intrusion sessions. For instance, level 2 means two random actions of the pattern sessions are lost. For each missing level, 10 predictions are made for the particular input by the system. The prediction results are shown in Table 7.

Table 7. Testing results based on the incomplete dataset.

Inputs	Prediction Errors	Prediction Accuracy [1]
a1	(2,5,4,6,7)	94.7%
a11	(0,4,1,4,5)	96.9%
a21	(3,3,3,7,6)	95.1%
a31	(1,2,4,5,7)	95.8%
a41	(0,0,2,3,7)	97.3%
a51	(0,2,5,4,7)	96%

[1] $accuracy(action) = (1 - \frac{\#actions\ wrongly\ predicted}{\#actions\ should\ be\ correctly\ predicted}) \times 100\%$.

The average accuracy of the predictions made for particular input is 96%, which means the algorithm has a strong resistance to the incomplete data. The relation between the missing level and the prediction accuracy is shown in Figure 7.

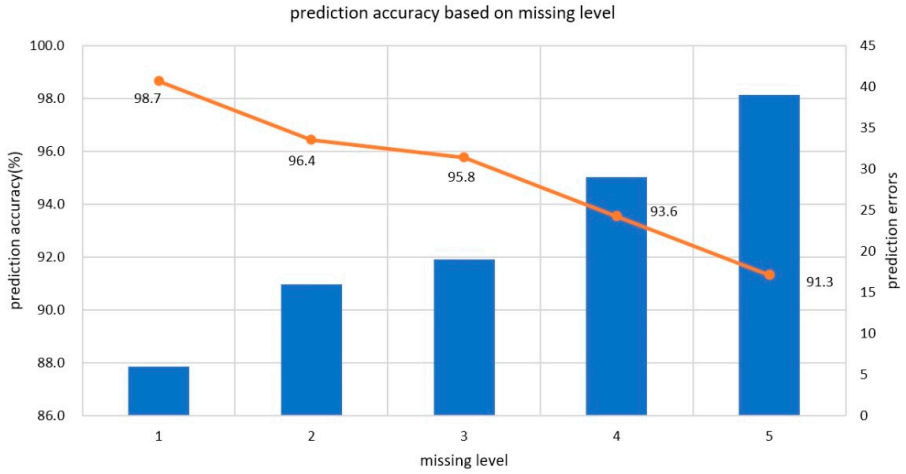

Figure 7. The prediction accuracy based on 5 missing levels.

The algorithm will first analyze the training dataset and then load the testing dataset which is generated with the same parameters of the training dataset, all the datasets contain the pattern sessions randomly generated and the background sessions. The experimental results show that the missing actions can hardly impact the proposed algorithm. The average prediction accuracy has been kept above 93% when even 40% of the actions are missing.

3.1.5. The Heterogeneous Dataset

Combined with the three types of data defects described above, we generated the final version of the heterogeneous dataset containing 50% of noise actions, 20% of disordered actions and 20% missing actions. There is only one type of intrusion session that expresses the unknown intrusion pattern in the dataset. The BIF algorithm should analyze the overall data to discover the unknown intrusion pattern. The actions of the unknown intrusion pattern will also appear randomly in other sessions. Finally, the discovered pattern will be compared with the initial session.

In this experiment, the dataset will be firstly described, and then, the experimental results will be discussed. We want to know how bad data the algorithm can analyze to discover the whole unknown pattern and the performance of finding effective patterns.

First, the composition of the generated dataset is listed in Table 8:

Table 8. The heterogeneous dataset for analyzing.

Sessions	Types	Amount	Note
Pattern sessions	1	10–20	The target that should be discovered
Irrelevant sessions	50	150	The other ordinary irrelevant patterns
Background sessions	Not limited	2000	The background traffic

As shown in Table 8, the first type data are the pattern session which is the target pattern <a_1, a_2, . . . , a_{10}>, the actions which are related to the target pattern will be named with the prefix "a" to distinguish them. When generating the pattern sessions, it will obfuscate these sessions with the following steps:

- Inserting noise actions. The noise actions are named with the prefix "b", for example, b_1, b_2.
- Removing actions. The pattern related actions may be removed from the session randomly to simulate the incomplete data.
- Changing positions. The positions of a few actions may be changed randomly in the session to simulate the disordered situations.

Figure 8 shows a part of the pattern sessions with different types of defects.

```
b93, a1, b10, b9, a2, b1, a4, a6, b32, a7, a8, a9, a3, a10
a1, b18, b45, a2, a3, b83, a8, b73, a5, a6, a7, b41, a9, b38, b78
b45, a1, a2, b80, a4, a5, b1, a6, a7, a8, a3, a9, a10
b92, a1, a6, a2, b33, a3, a4, b57, a5, b68, b32, a7, a8, b52, a9
a1, b26, a2, a4, a3, b38, a6, b86, b10, a7, b89, a8, a9, a10
a1, a3, a4, a5, b37, b54, a2, a6, b62, a7, b59, a8, a9, b13, b30, a10
a2, a3, b58, b21, a4, a5, a6, a7, a9, b23, b87, a8
a4, a1, a2, b52, a3, a5, a6, b57, b45, b41, a8, a9, a10, b71
a2, a3, a4, b70, b42, a10, a1, a5, b58, b8, b66, b75, a7, a8, a9
b73, b66, b59, a1, a2, b64, a6, b39, a4, a5, a7, a8, b45, a9, a10
a1, b13, a3, b39, a4, a5, b11, a6, a7, b77, b22, b34, a8, b4, a9, a10
b67, a1, b86, a9, a4, a5, a2, b39, b27, a6, a7, b24, a8, b7, a10
a1, a2, b28, b79, a5, b97, b82, a3, a6, a7, a8, a9, a10
a1, b59, a2, a3, a4, a5, b40, a7, b89, b16, a6, b82, a9, a10
b14, a8, a1, a2, a3, b92, a4, a5, b52, a7, a9, a10, b1
```

Figure 8. A fragment of the first type of data. Each row denotes a pattern session.

There are 50 types of sessions contained in the irrelevant sessions shown in Table 8;, they are ordinary patterns used to mislead the algorithm by containing a few actions appeared in the target pattern sessions at a specified percentage to simulate the real data environment. Intrusion patterns always share common actions with other sessions. The actions of the irrelevant sessions are named with the prefix "c". These irrelevant intrusion patterns are also obfuscated through the steps described above. The generated data are shown in Figure 9.

```
c32,  c33,  c34,  a7,  c36,  b116,  b65,  b44,  a5,  c37,  b158,  c38,  c39
c1,  b39,  a1,  c3,  c4,  c5,  a3,  c6,  c7,  b110,  c8,  b54,  c2,  c9,  c10
b177,  c10,  c12,  b121,  a2,  b157,  a8,  b6,  c13,  c14,  c15,  c16,  b145
b196,  b20,  b89,  c61,  c60,  a2,  b138,  a1,  c62,  c63,  b117,  c64,  c65
a2,  c82,  c83,  a3,  c85,  b125,  c86,  b60,  b116,  b16,  c87,  b1,  b118,  b78,  c89
c72,  a5,  a7,  c73,  c74,  c75,  b134,  b91,  b73,  c76,  b163,  c77,  c78,  b192
a9,  c40,  c41,  c42,  c44,  b109,  c43,  b150,  a10,  c45,  b175,  c47,  b27,  c48,  c49
a7,  b24,  b37,  a8,  c52,  b187,  c53,  c51,  b157,  c54,  c55,  c56,  b161,  b80,  c58,  c59
b138,  b92,  c22,  c23,  b26,  a3,  c24,  c25,  c26,  b173,  c27,  a8,  b117
c64,  c60,  c61,  a1,  c62,  b17,  c63,  c65,  b182,  b78,  b139
c51,  b137,  a7,  a8,  c52,  b104,  c53,  b94,  c54,  c55,  c56,  b177,  b76,  b27,  c58,  c59
c96,  c91,  b198,  a6,  c94,  b46,  c90,  c95,  a9,  c97,  b172,  c98,  b93,  c99,  b153,  b146
b56,  c10,  b71,  b150,  c12,  b20,  a2,  a8,  c14,  c16
```

Figure 9. A fragment of the second type of data. Each row denotes an intrusion session.

The third type of data are generated in a random way to increase the total data volume and introduce the data sparsity problem. In addition, these sessions have a specified probability to contain the actions appeared in the first type of data. The generated data of this type are shown in Figure 10.

```
b651,  b1290,  b1230,  b1253,  b1202,  b28,  b623,  a6,  b1,  b221,  b26,  b268,  b1133
b409,  b1278,  b802,  b1297,  a2,  b901,  b813,  b508,  b456,  b301,  b329,  b51
b261,  b488,  b31,  b504,  b591,  a2,  b1190,  b330
b312,  b1059,  b1272,  a5,  b83,  b221,  b1117
b872,  b516,  b713,  b743,  b74,  b355,  b180,  b565,  b771,  b635,  b638,  a5,  b1084,  b1263,  b955
b895,  a10,  b1156,  b595,  b285,  b1263,  b665,  b306,  b445,  b73,  b996,  b424,  b336,  b1227,  b480,  b1150
b139,  b780,  b896,  b433,  b99,  b1271,  b1165,  a5,  b820
b1023,  b637,  b851,  b183,  b1047,  a3
b206,  b1234,  b649,  b746,  b1198,  b1252,  b623,  a7,  b666,  b613,  b1183,  b1195
b1,  b1231,  b185,  b29,  b161,  b871,  a8,  b294,  b277,  b847
b353,  b1178,  b356,  b1179,  a2,  b1084,  b1130,  b963,  b791,  b760,  b222,  b267,  b1020,  b246
b1108,  b773,  b1225,  b1141,  a1,  b307,  b538,  b1191,  b192,  b547,  b101
```

Figure 10. A fragment of the third type of data. Each row denotes a normal session.

The three types of data are generated separately, however, they will be randomly mixed into the data stream fed to the analyzing algorithm. About 2350 sessions are generated, and more than 20,000 actions are created in the dataset. It takes about 600 milliseconds to finish the whole process including data-generating, analyzing, file writing and results calculating. All the tests in this paper are running on a server with 8 GB RAM and 2.6 GHz Intel CPU, on which Windows 10 operating system is installed, and the Java programing language is used to implement all the tests.

Based on the dataset with 20% disordered actions, 50% noise actions and 20% missing actions, and the pattern sessions sharing 20% of actions with the ordinary patterns sessions, the target intrusion pattern discovery result is illustrated in Figure 11:

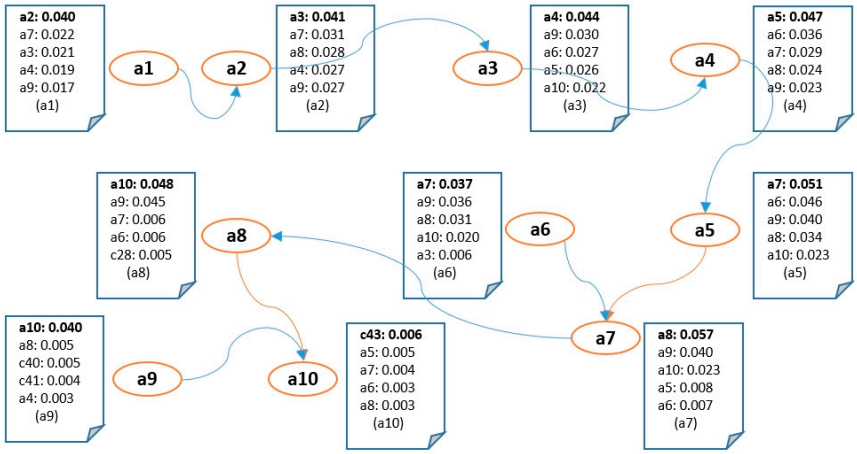

Figure 11. The dynamic correlation graph based on the analysis results.

As shown in Figure 12, the orange circle denotes the action, and the attraction list lists the top five actions that are heavily influenced by the specified action. The first bold action in the list is the next action that the particular action will point to and the float numbers in the list are the comprehensive influence factors. There are two orange arrows in Figure 12 pointing at the wrong actions comparing with the initial pattern which means the proposed method needs more adjustment and optimization. The influence threshold $\beta = 0.01$ can be used to filter out the actions with influences below β, thus, there are no subsequent actions after a10. The correlation graph can be updated over time and the links between actions will be altered too.

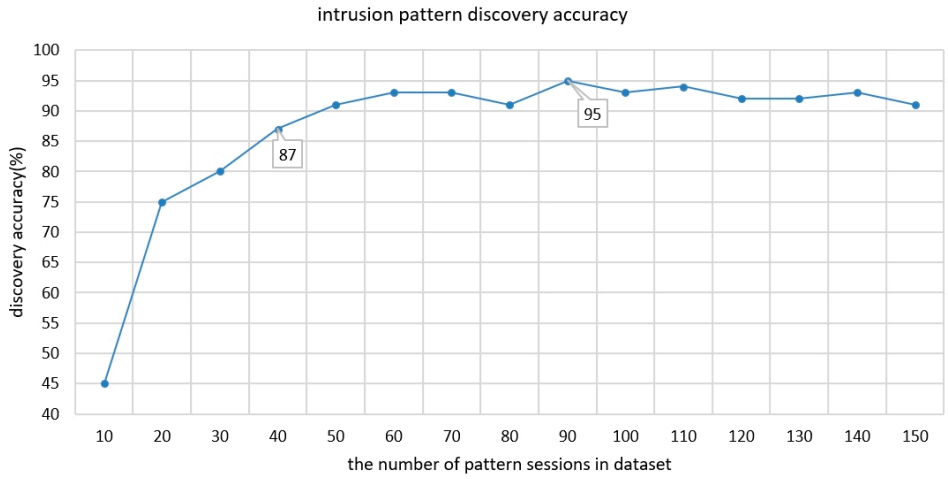

Figure 12. Intrusion pattern discovery accuracy of the proposed method.

The discovery accuracy is used to measure the ability of the system to discover patterns. It can be calculated by $\delta = |D|/|E| \times 100\%$ where $|D|$ denotes the correct relations discovered by system, and $|E|$ denotes all the relations the system should correctly discover.

The quantitative testing method is used to find out which factors have a potential impact on the accuracy of the proposed method. For the three types of data listed in Table 8, the relationship between the proportion of them in the dataset and the pattern discovery accuracy is tested. The results show that:

- The number of background sessions has little effect on the discovery accuracy;
- The irrelevant pattern sessions will impact the accuracy depending on the number of common actions sharing the target pattern sessions;

- With the fixed percentage of noise data, disordered data and incomplete data, the discovery accuracy is mainly impacted by the appearance frequency of pattern sessions.

The experimental results are shown in Figure 12. If an intrusion pattern appears above 50 times in the dataset with the three types of data defects, the discovery accuracy can be 91% and remains stable.

If the percentage of actions sharing between target pattern sessions and the irrelevant pattern sessions is lower than 40%, the pattern identifying accuracy can be kept above 90%. This is effective for distinguishing the particular intrusion patterns from other similar intrusion patterns.

4. Discussion

In this paper, a novel sequence learning and mining algorithm is proposed to meet the challenges of network log (or the IDS logs) defects caused by various environmental problems. The algorithm is simple, lightweight and effective in discovering the patterns hiding in the network logs.

The proposed algorithm is based on the backward attraction calculation, which means the association relation between two intrusion actions can be measured by the distance of their indices in the intrusion session (action sequence). The nearer of their position in the session, the stronger the attraction between them. The attraction strength is updated and accumulated with the different distances of actions in different sessions. The actions with higher attractions are selected to construct the correlation graph which is used for attack prediction or attack scenario recognition.

Three types of automatically generated datasets each with a different type of data defects are tested on the algorithm, the results show that the proposed algorithm is effective in learning and mining the sequence patterns from the disordered, noisy and incomplete session data. The prediction tests are used to measure the learning ability of the algorithm, and the average prediction accuracy is kept above 90%. Finally, the heterogeneous dataset is generated by combining all data defects to simulate the real data environment, and the pattern discovery ability of the algorithm is measured in different conditions. The experimental results show that the algorithm can discover the unknown intrusion pattern in the dataset containing 50 other different intrusion patterns sharing 40% actions with the target pattern session, and the discovery accuracy is kept above 91%.

Although the algorithm is effective in mining the intrusion patterns in the complex dataset, it is still impacted by the high disordered sessions, if 50% of actions in each session is randomly changed their position, the accuracy of the algorithm will go down to 85%. Solving this problem will be the aim of subsequent studies.

The proposed algorithm is designed based on online unsupervised learning and with no complex parameter tuning and retraining. In addition, the network logs normalization, aggregation, and other clustering processes are preferred to prevent the explosive growth of the action types.

Author Contributions: K.Z. conceived and designed the experiments; K.Z. performed the experiments; K.Z. analyzed the data; S.L. and H.Z. contributed environment; Y.X. and Y.C. provide resources; K.Z. wrote the paper. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Key R&D Program of China, grant number 2017YFB0802300, the Major Scientific and Technological Special Project of Guizhou Province, grant number 20183001, and the Foundation of Guizhou Provincial Key Laboratory of Public Big Data, grant numbers 2018BDKFJJ008, 2018BDKFJJ020, and 2018BDKFJJ021 And The APC was funded by Y.X. and Y.C.

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

References

1. Bryant, B.D.; Saiedian, H. A novel kill-chain framework for remote security log analysis with SIEM software. *Comput. Secur.* **2017**, *67*, 198–210. [CrossRef]
2. Stringhini, G.; Shen, Y.; Han, Y.; Zhang, X. Marmite: Spreading Malicious File Reputation through Download Graphs. In Proceedings of the 33rd Annual Computer Security Applications Conference, Orlando, FL, USA, 4–8 December 2017; pp. 91–102.

3. Blond, S.L.; Uritesc, A.; Gilbert, C.; Chua, Z.L.; Saxena, P.; Kirda, E. A look at targeted attacks through the lense of an NGO. In Proceedings of the 23rd USENIX conference on Security Symposium, San Diego, CA, USA, 20–22 August 2014; pp. 543–558.
4. Farinholt, B.; Rezaeirad, M.; Pearce, P.; Dharmdasani, H.; Yin, H.; Blond, S.L.; McCoy, D.; Levchenko, K. To Catch a Ratter: Monitoring the Behavior of Amateur DarkComet RAT Operators in the Wild. In Proceedings of the 2017 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2017; pp. 770–787.
5. Lu, G.; Guo, R.; Wang, J. An Analysis of the Behavior of APT Attack in the Ngay Campaign. In Proceedings of the 2018 IEEE 18th International Conference on Communication Technology (ICCT), Chongqing, China, 8–11 October 2018; pp. 1115–1122.
6. Ghafir, I.; Kyriakopoulos, K.G.; Lambotharan, S.; Aparicio-Navarro, F.J.; Assadhan, B.; Binsalleeh, H.; Diab, D.M. Hidden Markov Models and Alert Correlations for the Prediction of Advanced Persistent Threats. *IEEE Access* **2019**, *7*, 99508–99520. [CrossRef]
7. Navarro, J.; Deruyver, A.; Parrend, P. A systematic survey on multi-step attack detection. *Comput. Secur.* **2018**, *76*, 214–249. [CrossRef]
8. Husák, M.; Komárková, J.; Bou-Harb, E.; Čeleda, P. Survey of Attack Projection, Prediction, and Forecasting in Cyber Security. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 640–660. [CrossRef]
9. Yang, S.J.; Du, H.; Holsopple, J.; Sudit, M. Attack Projection. In *Cyber Defense and Situational Awareness*; Kott, A., Wang, C., Erbacher, R.F., Eds.; Springer: Cham, Switzerland, 2014; pp. 239–261.
10. Ramaki, A.A.; Rasoolzadegan, A.; Bafghi, A.G. A Systematic Mapping Study on Intrusion Alert Analysis in Intrusion Detection Systems. *ACM Comput. Surv.* **2018**, *51*, 1–41. [CrossRef]
11. Liu, C.; Singhal, A.; Wijesekera, D. A logic-based network forensic model for evidence analysis. In Proceedings of the 11th IFIP International Conference on Digital Forensics (DF), Orlando, FL, USA, October 2015; Volume 462, pp. 129–145.
12. Angelini, M.; Bonomi, S.; Borzi, E.; Pozzo, A.D.; Lenti, S.; Santucci, G. An Attack Graph-based On-line Multi-step Attack Detector. In Proceedings of the 19th International Conference on Distributed Computing and Networking, Varanasi, India, 4–7 January 2018; pp. 1–10.
13. Shen, Y.; Mariconti, E.; Vervier, P.-A.; Stringhini, G. TIRESIAS: Predicting Security Events through Deep Learning. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15 January 2018; pp. 592–605.
14. Haas, S.; Fischer, M. On the alert correlation process for the detection of multi-step attacks and a graph-based realization. *SIGAPP Appl. Comput. Rev.* **2019**, *19*, 5–19. [CrossRef]
15. Zhang, K.; Zhao, F.; Luo, S.; Xin, Y.; Zhu, H. An Intrusion Action-Based IDS Alert Correlation Analysis and Prediction Framework. *IEEE Access* **2019**, *7*, 150540–150551. [CrossRef]
16. Su, Y.-H.; Cho, M.C.Y.; Huang, H.-C. False Alert Buster: An Adaptive Approach for NIDS False Alert Filtering. In Proceedings of the 2nd International Conference on Computing and Big Data, Taichung, Taiwan, 18–20 October 2019; 2019; pp. 58–62.
17. Kawakani, C.T.; Junior, S.B.; Miani, R.S. Intrusion Alert Correlation to Support Security Management. In Proceedings of the XII Brazilian Symposium on Information Systems on Brazilian Symposium on Information Systems: Information Systems in the Cloud Computing Era-Volume 1, Florianopolis, Santa Catarina, Brazil, 17 May 2016; pp. 313–320.
18. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), Funchal, Madeira, Portugal, 22–24 January 2018; Volume 1, pp. 108–116.