applied sciences

MDPI

Article

Online Intrusion Scenario Discovery and Prediction Based on Hierarchical Temporal Memory (HTM)

Kai Zhang [1,*], Fei Zhao [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; zhaofei15206@126.com (F.Z.); 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: 2 March 2020; Accepted: 7 April 2020; Published: 10 April 2020

Abstract: With the development of intrusion detection, a number of the intelligence algorithms (e.g., artificial neural networks) are introduced to enhance the performance of the intrusion detection systems. However, many intelligence algorithms should be trained before being used, and retrained regularly, which is not applicable for continuous online learning and analyzing. In this paper, a new online intrusion scenario discovery framework is proposed and the intelligence algorithm HTM (Hierarchical Temporal Memory) is employed to improve the performance of the online learning ability of the system. The proposed framework can discover and model intrusion scenarios, and the constructed model keeps evolving with the variance of the data. Additionally, a series of data preprocessing methods are introduced to enhance its adaptability to the noisy and twisted data. The experimental results show that the framework is effective in intrusion scenario discovery, and the discovered scenario is more concise and accurate than our previous work.

Keywords: intrusion detection; intrusion scenario discovery; attack prediction; correlation analysis; IDS alerts; HTM; Hierarchical Temporal Memory

1. Introduction

Since the beginning of this century, cyberattacks have threatened users and organizations. They have become more complex and sophisticated with the development of computer networks. Nowadays, multistep attacks mostly aim at stealing important data from organizations and leads to serious data security problems.

A multistep attack, also called a multistage attack, intrusion scenario, refers to the ensemble of steps taken by one or several attackers with a single specific objective inside the network, containing at least two distinct actions [1]. The multistep attack consists of multiple single-step attacks which can be detected by the IDS (intrusion detection system), however, the multistep attack is not easy to discover, having some difficulties:

(1) Although the IDS is efficient in detecting the single-step attack, it can hardly reveal the logical relations among these single-step attacks.

(2) The real attack is a small probability event, and the most dangerous attacks happen only rarely, which mean that in each dataset we have only a few examples of attacks. This problem in intrusion detection research, called the "rare data problem" [2], seems even worse when multistep attacks are considered.

(3) IDS sensors could miss some steps of a multistep attack because these steps may appear normal to them [3]. Thus, we cannot assume that the important alerts related to the real attack will always be there.
(4) High data redundancy and volumes will slow down the analysis process, which is a significant challenge for the development of systems working in real time.

In this paper, an intrusion scenario discovery framework is proposed to tackle these difficulties by data preprocessing and online correlation analysis. The proposed framework was designed based on the following considerations:

First, the framework is a totally online system and has no offline mining or learning part; it is different from other multistep attack detection systems [4,5] that need offline processing. The HTM algorithm can learn the association patterns of intrusion actions in real time and the learned patterns will evolve with the incoming data over time, which is a good characteristic for online learning because it provides a continuous learning mechanism for our framework. However, the "short-term memory" has its negative influence on the prediction accuracy, so, an auxiliary method is proposed to mitigate the influence.

Second, the framework is designed based on the processing of streaming data and does not limit the boundary or the quantity of data, which means the model should be constructed cumulatively.

Third, the framework needs to predict the intrusion at an early stage without waiting for the model to be completely built. Given any intrusion action, the proposed framework will output the predicted action sequence that may happen with a higher probability in the future.

This is the first time to our best knowledge that the HTM algorithm is combined with intrusion scenario discovery. HTM has some attractive features, such as strong noise resistance, continuous learning without retraining, and an unsupervised prediction-driven learning mechanism. The proposed framework can benefit from these features.

In this section, four categories of existing approaches for constructing the intrusion scenarios are discussed and, after that, some HTM use cases are introduced.

(1) Some early prediction models based on prior knowledge are created manually by security experts. The domain knowledge, such as known cyber threats or intrusion scenarios, are leveraged in the modeling process. These models can highly represent real intrusion scenarios, and provide a high accuracy of prediction with no ambiguity, which is the reason why they are popular in the industry. However, it is heavy work for the experts to keep the models always available for attack strategies that are changing all the time. Early studies used the approach proposed by Qin and Lee [6], which can identify attack plans based on isolated attack scenarios, but appreciable adjustments should be made to predict the attacks that are beyond the predefined attack plans.
(2) Two types of graph models, HMM (hidden Markov models) and Bayesian networks, are commonly used in intrusion predictions. Essentially, they are attack graphs that can be constructed automatically. Statistical methods are utilized to obtain the transition probabilities or probability distributions based on which the prediction can be made. Ramaki et al. [7] proposed an attack prediction model based on the previous work [4], which is a real-time IDS alert correlation framework. In their model, Bayes networks and a Bayes attack graph (BAG) are employed to increase the accuracy of the prediction. Unlike the models based on Bayes networks, the HMM-based methods are insensitive to the missed states and transitions, which means it can work well without complete information of the intrusion scenarios [8]. Farhadi et al. [9] proposed the alert correlation and prediction framework utilizing the HMM model as the prediction component, which does not require the information about the network topology, system vulnerabilities, and system configurations, and is robust against over-fitting. However, the model cannot predict patterns of unknown attacks. Additionally, both of the models are sensitive to outliers and the lack of scalability for the various intrusion scenarios.

(3) Data mining is another prevalent technology for discovering correlations within the massive amount of security data before constructing the intrusion prediction models. Husák et al. [10] proposed an attack prediction framework based on data mining to identify the attack patterns from a million alerts. Sequential rule mining is employed to derive rules for prediction. Kim et al. [11] proposed an approach to create an attack graph based on data mining techniques for intrusion prediction. Although data mining can discover unknown intrusion patterns, redundant, small, frequent items would affect the accuracy of the prediction. The most important is that the data mining models cannot discover the critical data items with lower frequency.

(4) Since the ANN (artificial neural network)-related approaches are proposed to meet the intrusion prediction, many studies show that neural networks have a better result in predicting attacks than others. However, neural networks need to be trained offline and retrained in a regular time interval to meet the constantly changing data. Additionally, feature selection and dimensional reduction are two essential and complex tasks to conduct, and the parameters should be learned from the data. In other words, neural networks are heavy models and not optimal for online prediction in real-time. Subba et al. [12] proposed an intrusion detection model which can reduce the computational resources in the training phase and is suitable for real-time deployment. As described above, it needs feature selection and daunting parameter tuning to gain better performance.

The HTM (Hierarchical Temporal Memory) algorithm is a new type of intelligent learning system inspired by the structures and mechanism of data processing of the mammalian neocortex. Although the HTM is still evolving with the latest discoveries in neurobiology, many applications of it are proposed for its outstanding characteristics. The existing use cases based on HTM theory are discussed in detail.

Cui et al. [13] analyzed the properties of HTM sequence memory in the sequence learning and prediction tasks and compared the HTM sequence memory with other sequence learning algorithms. The model has better performance in continuous online learning, robustness to noise, fault tolerance, and no parameter tunings.

El-Ganainy et al. [14] proposed a method to predict medical streams in real-time; the paper employs two machine learning algorithms: HTM (Hierarchical Temporal Memory) and LSTM (Long Short-Term Memory) to model the same medical data stream, and the experimental results show that the HTM algorithm is more effective than the LSTM algorithm in three aspects: no training process, no domain knowledge, and the ability to adapt to variations in the data.

Li et al. [15] proposed a framework to detect the anomaly in the stream of air traffic data broadcasted between a flight and the air traffic control center. The HTM is used to encode the data and learns the patterns from the sequential data for the anomaly prediction, and then conducts the deviation analysis to identify the attack based on the former predictions.

Wang et al. [16] proposed a distributed vehicle network anomaly detection framework with the abilities of continuous online learning and exception scoring. The framework can learn the network data sequence and detect the abnormal states of the vehicle networks.

Shah et al. [17] proposed a framework to categorize documents using the SP (Spatial Pooling) algorithm of the HTM theory, and LSI (Latent Semantic Indexing) technology is used for extracting the top features from the input and converting them into binary format. The assessment shows that the HTM model is accurate compared to the conventional technologies used in text classification.

Shen et al. [18] have proposed a gait pattern recognition approach using the extended HTM algorithm. In the paper, the authors decomposed the gait image into sequences of movements of each body parts, and learns the features through three levels of neurons. A Markov chain is used to discover the feature groups in which the mutual transition probabilities are above the threshold, then the HTM algorithm, which is extended with a dynamic programming algorithm, is used to infer the complex sequences. The HTM algorithm proves the effectiveness of visual problems.

The main contributions of this paper are: (1) to introduce a series of data preprocessing methods to refine the streaming data; (2) the HTM algorithm is introduced to enhance the ability of online learning; and (3) an auxiliary method is proposed to improve the accuracy of intrusion scenario discovery.

2. Materials and Methods

In this section, the basic concepts and the theory of the HTM are introduced, and then the intrusion action clustering and the session reconstruction of the proposed intrusion scenario discovery framework is described in detail.

2.1. Preliminaries of HTM

The formal name of the HTM is HTM cortical learning algorithms, which is a machine learning technology that aims to capture the structural and algorithmic properties of the neocortex. The HTM models neurons (cells), which are arranged in columns, in layers, in regions, and in a hierarchy [19]. The HTM learns and memorizes the patterns by building and destroying the synapse connections around the neuron bodies. The HTM theory and basic concepts are explained in this section.

The software implementation of the HTM algorithm was published in 2011, which may be the real birth of the HTM. The theory of the HTM is developing quickly and the related applications are increasing every year for its distinct properties in online learning and prediction, especially in anomaly detection. More applications are being explored by academics and industry.

The HTM model is a multi-layer structure that each layer contains a number of columns and each column possesses several neuron cells. The model extracts and represents the features of the outside world by checking the activated column locations within the whole layer. The process of converting the outside data into the inner representation (sparse distributed representations, SDR for short) is called SP (spatial pooling), which is the first phase of the whole learning process. The column can be activated as long as one of the cells composing the column is activated. Under two situations, the neuron cells will get fired into an active state. First, the specified cell is activated in the bursting phase. Second, if the cell is in the predictive state, it will be active in the next time step (for more details, please see [19]). The structure of the HTM is shown in Figure 1.

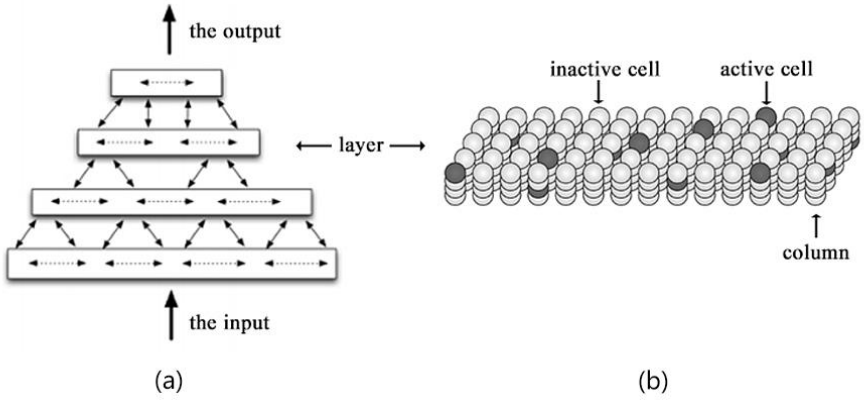


Figure 1. The structure of the HTM model. (**a**) The hierarchy structure of HTM; (**b**) the neuron cells in a layer of columns.

Each layer not only represents the different data features but also accumulates the feature transitions over time through constructing or destroying the connections among cells of different columns in the same layer. The permanence value of the specified connection is increased when the connected two cells are successively activated in two adjacent time steps, otherwise, it decreased. This is the second process of HTM called TM (temporal memory), which can store and predict the most probable features based on strong connections. Each lower layer in the hierarchy is the input

representation of the higher layer, as a result, the highly abstracted patterns of the original data are obtained by the top layer.

In a word, the learning process contains two phases: the SP (spatial pooling) and TM (Temporal memory). In the SP phase, the original data are first converted into bit array and then abstracted into an inner representation called SDR. In the TM phase, the temporal correlation of SDRs are accumulated over time. Finally, the SDRs that are predicted by TM are classified by the classifier in order to map the inner representation to the outer presentation. The learning phases are illustrated in Figure 2.

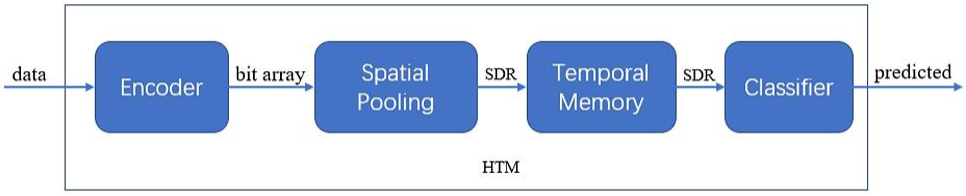


Figure 2. The learning process of the HTM.

In this paper, we will not discuss the details of the HTM model itself. For more information about the concepts, functions, and mathematical definitions used in the HTM, please see [13,20–24].

2.2. The Intrusion Scenario Discovery and Prediction Framework

The main components (phases) of the proposed framework is illustrated in Figure 3:

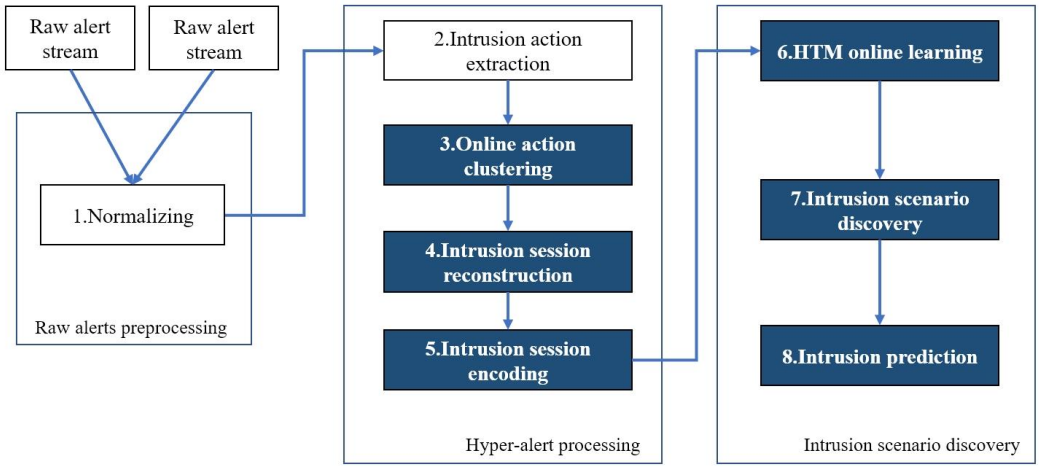


Figure 3. The components of the proposed framework. The rectangles in blue are the main contributions of this paper.

The descriptions of each component are listed as follows:

1. Normalizing: Raw alerts are converted into a unified data structure for future analyzing;
2. Intrusion action extraction: A group of normalized alerts can correlated into a hyper-alert, which is called intrusion action in this paper, and should meet three requirements: first, they have the same source IP address and destination IP address; second, they have the same destination port or the same intrusion type; and last, they occur continuously in time order. The massive redundant raw alerts are aggregated in this process;
3. Online action clustering: Many extracted actions possess similar semantics, which will be merged based on the similarity calculating;
4. Intrusion session reconstruction: The time-ordered action sequence will be split into many subsequences by calculating the variance of the time gaps between actions, each subsequence

is named the intrusion session. Note that all the actions of a particular intrusion session are extracted from the same pair of physical devices. The extracted session will be refined in order to reduce more noise actions;

5. Intrusion session encoding: The intrusion session should be encoded before feed to the HTM algorithm. The session can be first converted into the action ID sequence, and the ID will be encoded into a bit array which is the HTM inner representation of that action. The learning phase of the HTM starts with the encoding of the ID sequences;
6. HTM online learning: The HTM learns the patterns of the action sequences and predicts the next step according to the current input, and the anomaly score which indicates the discrepancy between the predicted value and the input value. The predicted values and the anomaly scores will be updated in a particular matrix;
7. Intrusion scenario discovery: Based on the analyzing of historical predicted actions and the anomaly scores in the correlation matrix, the correlation strengths of intrusion actions are calculated by the auxiliary method; and
8. Intrusion prediction: The system will search through the matrix to determine the attacking paths that may happen in the future.

The system is proposed based on our previous work [24], a few concepts, such as intrusion action, intrusion session, correlation graph, and methods, such as normalizing and intrusion action extraction, are inherited. In this paper, the data preprocessing is not discussed in detail.

2.2.1. Intrusion Action Clustering

To simplify the associations between actions, the online action clustering is introduced, which can reduce the classes of actions. The definition of intrusion action is modified for the consequential processes.

Intrusion action, which referrs to a group of attack types derived from the raw IDS alerts, can be denoted as a tuple A: <time, types, cid, srcIp, dstIp>. The *time* field indicates the creating time, the types field contains a group of alert types expressing the same malicious behavior, the cid field indicates the class id, and the fields srcIp and dstIp indicate the source IP address and the destination IP address, respectively. The actions are extracted by the intrusion action extracting process.

The intrusion actions can be compared with each other based on the similarity of the types field. In this paper, the edit distance of two strings is used to calculate the similarity of the actions. Consider two intrusion actions A and $A^{'}$, the types field can be denoted as $T(X) = \{t_n | t_n \in X,\ n = 1, 2, \ldots, |X|\}$, the similarity of two attack types which are strings can be calculated with Equation (1):

$$\Delta(tx, ty) = \frac{1}{\mu LD(tx, ty) + 1} \qquad (0 < \mu \leq 1) \tag{1}$$

where LD is the Levenshtein Distance function that returns the distance of two attack types: t_x and t_y. The parameter μ is a factor that determines the capacity of a potential cluster. Thus, the similarity of action A and A' is calculated with Equation (2):

$$sim(A, A') = \frac{\sum_{k=1}^{p} \Delta(A(t_k), A'(t_k))}{p} \tag{2}$$

where $A(t_k)$ indicates the kth attack-type of A, and $p = max\ (|A|, |A'|)$, if $sim\ (A, A') = 1$, A is the same as A'. Based on the *sim* function, online clustering can be achieved. The *cid* field will be set to the cluster id for each action in the same cluster.

For a cluster $C = \{A_k | k \in Z,\ 1 \leq k \leq |C|\}$, the average similarity between an action X and cluster C can be calculated with Equation (3):

$$avgsim(X, C) = \frac{\sum_{k=1}^{n} sim(X, C[k])}{n} \tag{3}$$

where $C[k]$ denotes the kth action in cluster C. If the value of *avgsim (X, C)* is higher than the threshold, then X will be put into cluster C, otherwise, a new cluster will be created.

Based on the equations described above, the actions with similar semantics can be merged into one class, and the number of actions will be reduced significantly.

2.2.2. Intrusion Session Reconstruction

An intrusion session is a sequence with all the unique actions extracted from the communication of a pair of devices in it. As described before, the time-ordered action sequence will be split into many subsequences by calculating the variance of the time gaps between actions. Each subsequence is named intrusion session in this paper.

In practice, the intrusion session always contains long, repeated, and disordered actions, which lower the performance of the sequence learning system. In most situations, the types of actions involved in a particular intrusion are more important than the order of them; in other words, the order of actions can be altered due to variety of situations, such as the network delay, the system error, even the intended disorders made by the attacker. Thus, the session reconstruction is proposed to refine the sessions.

The extracted session in Figure 4 is the ordinary state after applied the pruning algorithm proposed in [24]. If the session is converted into a more precise and clearer one, the performance of the system can be highly enhanced.

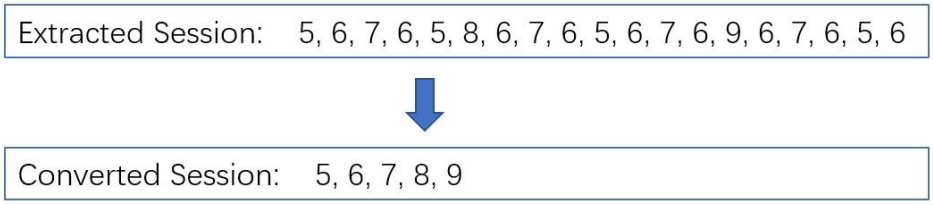


Figure 4. The extracted session and the converted session. The numbers denote the intrusion actions.

In order to discover the potential skeleton of the intrusion, a session refining method is proposed in this paper to highlight the real information hiding in the sessions. It removes the repeated actions from intrusion sessions by calculating the similarity of actions. When reconstructing the session, the new arrived action will be compared with the existing actions of the converted session, and it will discard the actions if they have been in the converted session. The reconstruction process keeps all the action information and the basic orders of actions, and it is more effective than our pruning algorithm proposed in our previous work [24]. Figure 5 shows how the method works.

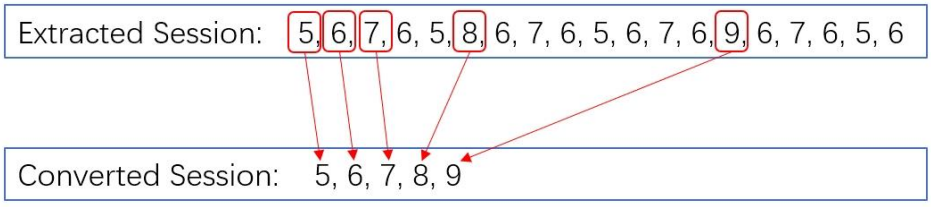


Figure 5. The proposed session refining method.

Although the session refining method is simple, it is effective in discovering intrusion patterns. The most important is that it can stabilize the association patterns of actions in the session, and provide a clean sequence for the HTM algorithm to learn.

2.2.3. Online Learning and Prediction

The HTM algorithm can provide some advantages for the proposed framework: (1) It learns the patterns and trends based on the variance of the features over time, which makes it suitable for sequence learning, and it has a better performance than other sequence learning algorithm [13]; (2) it further reduces the influence of the noise due to the special representations of the data; and (3) it can be used to reduce the negative influences of data problems, such as sudden changes in the pattern, and it highlights the real patterns that should be focused on.

The association relation of actions is the main objective to be learned by HTM algorithm. The correlation between two actions A and B can be measured by their temporal orders in the sequence, and the TM (Temporal Memory) learning process of HTM which is described in Section 2.1 can discover and accumulate the temporal association strength between A and B. Thus, the intrusion sessions are first converted into the action ID sequences and the IDs are fed to the HTM by their occurrence order.

The backend system will record and analyze the output of the HTM in real-time. There are three main output values are recorded for analyzing: The current input value, the predicted value, and the anomaly score calculated with the two values. The anomaly score assesses the differences between the predicted SDR and the input SDR.

Considering P'_{t-1} as the predicted SDR at time step $t-1$, and P_t as the input SDR at time step t. The anomaly score at time step t can be calculated as:

$$\omega_t = 1 - \frac{p'_{t-1} \bullet p_t}{|p_t|}, \tag{4}$$

where $|P_t|$ denotes the number of one bit in P_t, the symbol $\bullet$ computes the number of one bit of the SDR that is different from the other one. ω_t will be 0 if the predicted SDR is totally different from the input SDR.

The prediction made by HTM can be denoted as a triple: $<p, p', \omega>$ where p indicates the current input value, p' indicates the predicted value at the previous time step, and ω indicates the anomaly score. The records with $\omega \geq 0.6$ will be selected to update the correlation matrix.

Although the introduction of the HTM algorithm enhances the framework in the ability of online learning and analyzing, the learned patterns will evolve with the data variance in the long-term continuous learning process. In other words, the learned patterns should be kept stable in a way. Thus, the correlation matrix (CM) is proposed.

For given actions $A_1, A_2, A_3, \ldots, A_n$, the correlation matrix can be defined as an $n \times n$ matrix $M(A_i, A_j) = <|A_i \rightarrow A_j|, \overline{\omega_0}> \quad (i, j < n)$ where A_i indicates the input action, and A_j indicates the predicted action. The value of $M(A_i, A_j)$ is a tuple, $|A_i \rightarrow A_j|$ denotes that the number of $A_i \rightarrow A_j$ has been predicted, and $\overline{\omega_0}$ is a single exponential smoothed anomaly score. $\overline{\omega_0}$ can be calculated by Equation (5):

$$\overline{\omega_0} = \overline{\omega_0} + \alpha(\omega_n - \overline{\omega_0}) \tag{5}$$

where ω_n indicates the anomaly score of the latest prediction. α denotes the smooth factor.

In the prediction phase, the correlation strengths (CS values) between actions are calculated by the historical predictions as:

$$CS(Ai, Aj) = \frac{|A_i \rightarrow A_j|}{|A_i \rightarrow *|} \times \overline{\omega_0} \tag{6}$$

where $|A_i \rightarrow *|$ denotes the total number of the predictions with A_i as the input action.

For a particular input action *A*, the actions predicted by HTM can be searched in the matrix and the CS values between *A* and the predicted actions are calculated, the action *B* will be selected if CS *(A, B)* is the max value among the CS values. Then, the framework iteratively finds the action *C* that makes *CS (B, C)* the max value among CS values calculated with *B* as the input action, and so on. Thus, the prediction sequence is constructed. In order to avoid the system falling into an endless loop, the searching progress will be ended if the newly searched action has appeared in the prediction sequence.

3. Results

In this section, the experimental evaluations are performed. Two main aspects of the framework will be evaluated: (1) The accuracy of intrusion pattern learning; and (2) the ability of multistep attack scenario discovery.

3.1. Experiment Setup

The proposed system is implemented with Java programming language and runs on a server with 8 GB RAM and 2.6 GHz Intel CPU, on which Windows 10 operating system is installed. In addition, the standard Java implementation of the HTM framework is employed and the free open source IDS software Snort (Sourcefire Columbia, Maryland, USA) is used in reading, detecting network traffic data, and generating raw alerts. The initial parameters of the system are summarized in Table 1.

Table 1. Parameters set for the system.

Parameters	Values	Notes
Ls	200	Action queue length
Lc	2000	Alert cache
μ	0.1	Similarity factor
s_0	0.6	Similarity threshold
ω	0.6	Anomaly score threshold
N	500	Encoder bit array size
W	21	Encoder bucket width
Cl	2048	Number of columns in SP
ce	10	Number of cells per column
α	0.3	Smooth factor
others	-	Other parameters are default

3.2. Evaluations on CICIDS2017 Dataset

CICIDS2017 intrusion detection evaluation dataset [25] was published by the Canadian Institute for Cybersecurity in 2017 which covers a variety of the most up-to-date well-known attacks and it is free to download for studying. We adopt the dataset for the reasons listed below:

1. The dataset resembles real-world network traffic data that contains realistic and naturalistic benign background traffic that simulates everyday interaction behaviors of 25 users;
2. The data was continuously captured for five days, from Monday to Friday in July 2017. About 10 GB of data was created per day and was labeled for training. It needs to noted that the Monday dataset only contains benign interactions of daily work;
3. The performed attacks including brute force FTP, brute force SSH, DoS, Heartbleed, web attack, infiltration, botnet, and DDoS; and
4. The network traffic was captured in a completely configured network, including modem, firewall, switches, routers, and PCs, with a variety of operating systems, such as Windows, Ubuntu, and Mac OS X, and most of the traffic is based on the HTTP, HTTPS, FTP, SSH, and email protocols.

To evaluate the proposed system, we used the data captured on Thursday, containing two attacking processes which can be seen as two scenarios. First, in the morning, the attacker executed three web

attacks: brute force, XSS (Cross Site Scripting), and SQL injection. Second, in the afternoon, the attacker performed a series of infiltrations.

The evaluation process is divided into two parts: first, we will assess the abilities of the intrusion sequences extracting and reconstructing for the learning process. In the second part, we will evaluate the predictions of the system.

3.2.1. Data Preprocessing Results

We split the Thursday data file into four parts for snort to read and detect in a continuous way. More than 64,000 alerts generated by snort are recorded in a file. About 5876 intrusion actions are extracted and 39 classes of intrusion actions remain after clustering. After that, 780 intrusion sessions are extracted and only 28 sessions have more than two actions. After data preprocessing, 28 distinct intrusion sessions consisting of 39 classes of actions are extracted. The statistical results are listed in Table 2.

Table 2. Statistical results of the data preprocessing.

Names	Values
Total raw alerts	64,000
Total alerts used to construct the sessions	2040 (3.2% of total alerts)
Alert groups	769
Extracted actions (repeated)	5876
Action classes (after clustering)	39
Extracted sessions	780
Sessions with only one action	702 (90% of total sessions)
Sessions with more than two actions	28 (3.5% of total sessions)
Session refining rate [1]	82%
Sessions refined	38 (49% of sessions with more than one actions)

[1] $refining = \frac{\#total\ actions\ removed\ from\ refined\ sessions}{\#total\ actions\ of\ refined\ sessions} \times 100\%$.

In Table 2, about 2040 raw alerts (3.2% of total alerts) are used to construct the actions and sessions, the rest of alerts are merged or discarded by system. The system reduces the number of alerts by two ways: (1) intrusion action extraction, the similar alerts (with small edit distance of the alert type) will be merged to express the same action; (2) action clustering, 5876 actions are clustered into 39 classes, which can significantly reduce the number of alerts; and (3) intrusion session reconstruction, which can reduce the actions in a session. For the statistical requirements of the experiments, each intermediate process' results are saved in the files. The data related to the real intrusion can be very sparse compared to the whole dataset. The extracted intrusion actions are listed in Table 3, and some actions consist of multiple alerts.

Table 3. Extracted intrusion sessions.

Scenarios	Actions [1]	Alerts
Infiltration	A	protocol-snmp request tcp
	B	protocol-snmp agentX/tcp request
	C	protocol-icmp ping protocol-icmp ping undefined code protocol-icmp echo reply undefined code
	D	policy-other tcp packet with urgent flag attempt server-other winnuke attack
	E	policy-other tcp packet with urgent flag attempt
	F	server-webapp robots.txt access
	G	policy-other ftp anonymous login attempt
	H	app-detect failed ftp login attempt
	I	openssh maxstartup threshold connection exhaustion denial of service attempt
	J	indicator-scan ssh brute force login attempt
	K	server-other realnetworks helix server ntlm authentication heap overflow attempt
Web attack	L	server-webapp password sent via post parameter
	M	server-webapp flexense diskpulse disk change monitor login buffer overflow attempt
	N	policy-other script tag in uri—likely cross-site scripting attempt
	O	sql 1 = 1 − possible sql injection attempt

[1] Not all the actions are listed in the table for the limitation of this paper.

Compared with the previous work [24], there are two improvements in the data processing: (1) The total number of the extracted intrusion actions decreased significantly due to the action clustering which can reduce the similar actions in a session and make the semantic information more clearly, while, at the same time, it saves many computing resources; and (2) the rebuilt sessions are more accurate in expressing the intrusion strategies. This is because the complicated binary relations are simplified. Most importantly, it highlights the semantics hiding in the sessions.

3.2.2. Data Learning Results

The accuracy of the framework in discovering potential intrusion patterns and the ability of discovering the intrusion scenarios will be evaluated in this section. First, the framework will be tested on an automatically generated dataset by programming. Second, the system will be tested on a series of intrusion sessions extracted in the data preprocessing evaluation, and these intrusion sessions contain the real intrusion scenarios.

In order to test the core algorithm of the system in a flexible way, the automatically generated dataset are used by programming. Five intrusion patterns are generated with about 10 actions in each pattern. Some irrelevant actions are inserted into these patterns randomly to mislead the system. The target of the system is to discover these unknown patterns in an unsupervised way. After training, some actions are provided to the system to make predictions. The predicted actions will be compared with the five patterns, and all the predicted actions that are not match with the patterns will be recorded for statistics.

The details about the dataset are listed in Table 4. The whole evaluation contains 10 tests, and each test is conducted separately with the testing data regenerated. The test data contains three parts: (1) The intrusion patterns to be discovered by the system; (2) the background traffic; and (3) the gradually increased noise actions inserted into each pattern.

Table 4. Intrusion patterns for testing. The system makes predictions for input actions.

Intrusion Patterns	Input Action	System Prediction
1-2-3-4-5-6-7-8-9-10	1	2-3-4-5-6-7-8-9-10
11-12-13-14-11-12-13-14-15-16	12	13-14-11(13-14-15-16)
17-18-19-20-24-28-29	17	18-19-20-24-28-29
21-22-23-24-28-29-30	22	23-24-28-29-30
25-26-27-28-29-30-21-24	26	27-28-29-30-21-24

The generated data are shown in Figure 6, the number denotes the action IDs, and the actions with ID more than 100 are irrelevant actions. For the system, the frequency of one or more actions appearing together in the sequence will be learned by the HTM algorithm through the prediction and revising process.

S1 (Noise=2) :
21,**22**,**23**,**24**,162,**28**,**29**,**30**,111,135,126,**1**,**2**,131,**3**,**4**,**5**,**6**,165,**7**,**8**,**9**,**10**,108,198

S2 (Noise=5) :
124,385,**17**,**18**,221,**19**,**20**,357,**24**,**28**,154,**29**,108,125,187,**25**,**26**,122,126,**27**,
208,**28**,**29**,326,**30**,**21**,**24**,289,166,149,299

Figure 6. The generated action sequences with different noise level. The bolded numbers denote the pattern action, the grey numbers denote the background traffic, and the red numbers denote the noise actions.

The testing program will create the data file for each test and load the data into the framework. Each data file contains about 2000 actions and 20 randomly-selected intrusion patterns with noise. The noise actions will increase with testing.

For each test, the framework will correlate the actions based on their high CS value. If two actions appear together only once, the anomaly score of the HTM for this prediction will be very low and the CS value too, and if one particular action frequently appears with many other different actions, the HTM will keep a low prediction accuracy for this action because it cannot decide which action will appear after it, Additionally, the confidence will decrease. Thus, only stable patterns will be discovered. In the test, the system will start searching with the given action and only select the action with the highest CS value among the candidate values which are greater than 0.6.

The prediction errors for each pattern in each test are listed in Table 5. The prediction accuracy for each pattern is an average value of 10 tests. The prediction errors will increase with the noise level.

Table 5. Intrusion pattern discovering results.

Intrusion Patterns	Errors	Accuracy [1]
1-2-3-4-5-6-7-8-9-10	(0,0,0,0,0,0,0,1,3,4)	92%
11-12-13-14-11-12-13-14-15-16	(0,0,0,0,0,1,0,0,0,2)	92.5%
17-18-19-20-24-28-29	(0,0,1,0,0,0,1,0,2,3)	89%
21-22-23-24-28-29-30	(0,0,0,0,0,0,0,0,1,4)	90%
25-26-27-28-29-30-21-24	(0,0,0,0,0,0,0,1,0,3)	93%

[1] $accuracy = 1 - \frac{\#false\ predicted\ actions}{\#all\ should\ be\ predicted} \times 100\%$.

The false positive of the framework can be calculated as *FP = #prediction errors/#all actions predicted* × 100%. Figure 7 shows the false positives at different noise levels. If each pattern contains less than five noise actions (50% of noise) the false positives can be lower than 3%; if the noise goes up to 80%, the false positives will increase significantly. Thus, the framework can work at a noise level lower than 80%.

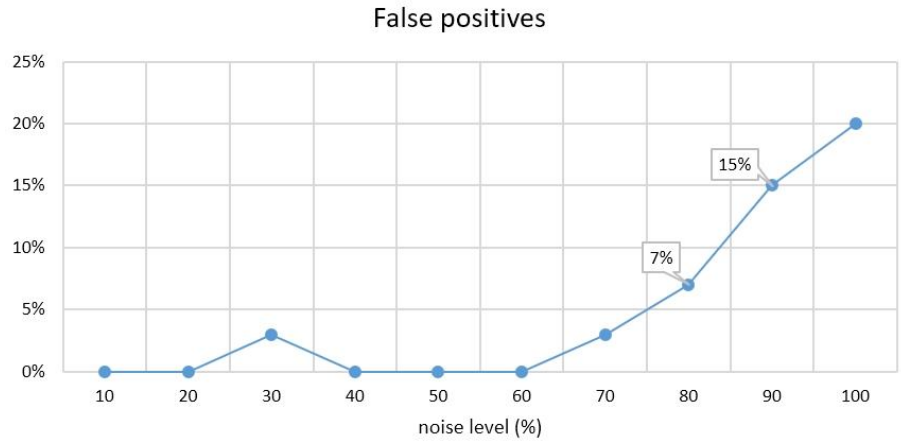


Figure 7. The false positives of the framework.

In the tests, the anomaly score threshold is set to 0.6, 30% of the data are filtered out and not recorded in the correlation matrix, which reduces the size of the matrix and the searching time. When actions 1–10 are correlated by the system, the CS values are shown in Figure 8.

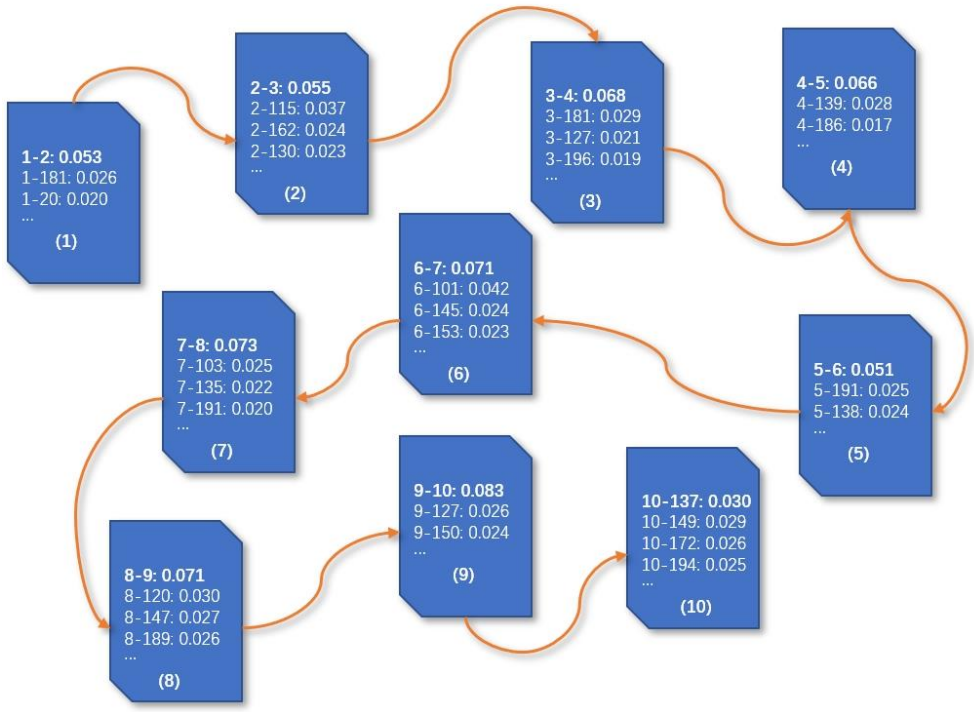


Figure 8. The correlated actions 1–10, and their CS values.

With the complex intrusion scenario, the framework tends to correlate multiple intrusion patterns together based on the common actions they are sharing. For instance, the three patterns (17~29, 21~30, 25~24) listed in Table 5 will be correlated because they share the actions 24, 28, and 29. Figure 9 illustrates the correlate graph.

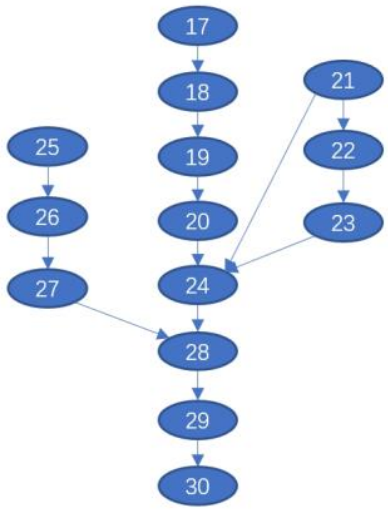


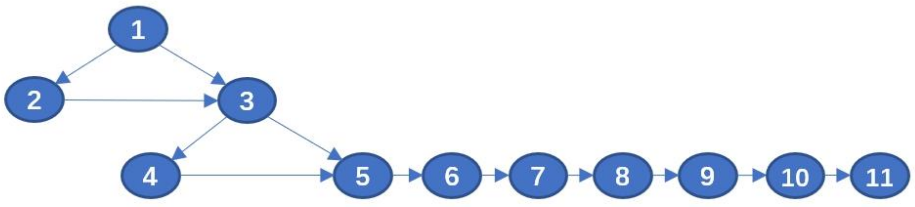
Figure 9. The correlate graph based on the three patterns (17~29, 21~30, 25~24).

To evaluate the whole system based on a real intrusion scenario, the infiltration attack scenario of the CICIDS2017 dataset is used. The scenario is about a Dropbox download attack, which leveraged the technologies of overflow, port scanning, and Nmap. There are 13,700 IDS alerts generated by snort, 1647 intrusion actions, 163 sessions are extracted, and 22 sessions have more than two actions. All the extracted sessions are loaded into the framework. A part of the sessions is listed in the Table 6.

Table 6. A part of the extracted intrusion sessions of an infiltration attack.

Sessions	Sessions	Sessions
1-2-3-4-5-6-7	2-1-3-4-5-6-7-14	23-24-25
8-9-10-3	7-8-18-17-19-20-21	3-8-9-10-11-26
8-10-3-9	8-9-3-10-11	1-2-3-4-5-7-9-10
1-2-4-5-14	2-1-3-4-5-6-14-7	1-2-27-13-6-14
10-11-12-8-9	2-1-3-4-5-14-7-15-16-17-18-19-20	28-25-29

The discovered intrusion scenario is shown in Figure 10. Compared with our previous work, the number of nodes in the correlation graph is reduced and the semantic of the correlation graph is closer to the real scenario. However, the samples related to the real scenario are very limited, although there are more than 13,700 alerts, only 720 alerts (5%) are used to construct the actions.



1: snmp request;
2: snmp agentx/tcp request;
3: icmp ping undefined code;
4: winnuke attack;
5: tcp packet with urgent flag attempt;
6: robots.txt access;
7: ftp anonymous login attempt;
8: failed ftp login attempt;
9: openssh maxstartup threshold connection exhaustion denial of service attempt;
10: ssh brute force login attempt;
11: realnetworks helix server ntlm authentication heap overflow attempt.

Figure 10. The discovered intrusion scenario.

Compared with our former work, the new system gains four new better characters: (1) there are no more complex branches in the correlation graph because all the sessions are reconstructed to remove the ambiguous relations; (2) the new system is focused on predicting the future attack trends rather than the particular next attack. The action nodes in the graph are the abstraction of a group of malicious behaviors with the same purpose; (3) the correlation graph is constructed based on the prediction results of HTM, which means the graph is representing one of the intrusion scenarios previously learned by HTM; and (4) the correlation graph is updated over time, which will be beneficial for learning new patterns and predicting unknown attacks.

4. Discussion

The discussions of the experimental results are given in Section 3.2. In this section, our works are concluded. The unsupervised online sequence learning and prediction system is proposed. Several measures have been introduced to enhance the performance of the system, such as intrusion action clustering, intrusion session reconstruction, and the optimizations of data processing. These measures can effectively reduce the data noises, redundancies, and false positives, making it easier to discover the potential intrusion behaviors and the correlations between them. The reduction of intrusion action types and unified intrusion sessions have greatly reduced the ambiguous correlations and further simplified the correlation graphs. We introduced the HTM cortical learning algorithms to improve the online sequence learning, based on the predictions of HTM, the correlation graph can be constructed in real-time and keeps updating over time. Offline training is not needed anymore, which makes it lightweight for online work.

Many aspects of the system should be studied and enhanced in the future: (1) the orders of actions in a session are important for the system; if all actions randomly change their relative positions frequently, the association patterns between actions are difficult to discover for the system, thus, a pattern stabilization method (maybe based on the statistics) which can correct the disordered actions before learning is appropriate for the system; (2) if multiple intrusion scenarios occurred in parallel and share some common intrusion actions, the framework will correlate them into one large scenario, which leads to the complex model, so the studies of parallel intrusion scenario construction is a direction; and (3) although the HTM algorithm can learn the unknown patterns of the sequence, it can only predict the next time step, rather than predicting a sequence of future actions. The parameters of the HTM algorithm should be carefully set, though it is easier to set than other machine learning algorithms.

Author Contributions: K.Z. conceived and designed the experiments; K.Z. performed the experiments; K.Z. and F.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. Navarro, J.; Deruyver, A.; Parrend, P. A Systematic Survey on Multi-step Attack Detection. *Comput. Secur.* **2018**, *76*, 214–249. [CrossRef]
2. Ourston, D.; Matzner, S.; Stump, W.; Hopkins, B. Applications of Hidden Markov Models to Detecting Multi-Stage Network Attacks. In Proceedings of the 36th Annual Hawaii International Conference on System Sciences, Big Island, HI, USA, 6–9 January 2003; p. 10.
3. Bing, C.; Lee, J.; Wu, A.S. Active Event Correlation in Bro IDS to Detect Multi-Stage Attacks. In Proceedings of the Fourth IEEE International Workshop on Information Assurance (IWIA'06), London, UK, 13–14 April 2006; pp. 16–50.

4. 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]
5. Ren, H.; Stakhanova, N.; Ghorbani, A.A. An Online Adaptive Approach to Alert Correlation. In Proceedings of the Detection of Intrusions and Malware, and Vulnerability Assessment, Bonn, Germany, 8–9 July 2010; pp. 153–172.
6. 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.
7. Ramaki, A.A.; Khosravi-Farmad, K.; Bafghi, A.G. Real Time Alert Correlation and Prediction Using Bayesian Networks. In Proceedings of the 12th International Iranian Society of Cryptology Conference on Information Security and Cryptology (ISCISC), Rasht, Iran, 8–10 September 2015; pp. 98–103.
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. Farhadi, H.; AmirHaeri, M.; Khansari, M. Alert Correlation and Prediction Using Data Mining and HMM. *ISeCure* **2011**, *3*, 77–101.
10. Husák, M.; Kašpar, J. Towards Predicting Cyber Attacks Using Information Exchange and Data Mining. In Proceedings of the 14th International Wireless Communications & Mobile Computing Conference (IWCMC), Limassol, Cyprus, 25–29 June 2018; pp. 536–541.
11. Kim, Y.; Park, W.H. A study on cyber threat prediction based on intrusion detection event for APT attack detection. *Multimed. Tools Appl.* **2014**, *71*, 685–698. [CrossRef]
12. Subba, B.; Biswas, S.; Karmakar, S. A Neural Network based system for intrusion detection and attack classification. In Proceedings of the 2016 Twenty Second National Conference on Communication (NCC), Guwahati, India, 4–6 March 2016; pp. 1–6.
13. Cui, Y.; Ahmad, S.; Hawkins, J. Continuous Online Sequence Learning with an Unsupervised Neural Network Model. April 2016. Available online: https://arxiv.org/abs/1512.05463v2 (accessed on 15 July 2019).
14. El-Ganainy, N.O.; Balasingham, I.; Halvorsen, P.S.; Rosseland, L.A. On the Performance of Hierarchical Temporal Memory Predictions of Medical Streams in Real Time. In Proceedings of the 2019 13th International Symposium on Medical Information and Communication Technology (ISMICT), Oslo, Norway, 8–10 May 2019; pp. 1–6.
15. Li, T.; Wang, B.; Shang, F.; Tian, J.; Cao, K. Online sequential attack detection for ADS-B data based on hierarchical temporal memory. *Comput. Secur.* **2019**, *87*, 101599. [CrossRef]
16. Wang, C.; Zhao, Z.; Gong, L.; Zhu, L.; Liu, Z.; Cheng, X. A Distributed Anomaly Detection System for In-Vehicle Network Using HTM. *IEEE Access* **2018**, *6*, 9091–9098. [CrossRef]
17. Shah, D.; Ghate, P.; Paranjape, M.; Kumar, A. Application of hierarchical temporal memory theory for document categorization. In Proceedings of the 2017 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computed, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI), San Francisco, CA, USA, 4–8 August 2017; pp. 1–6.
18. Shen, J.; Loew, M. Hierarchical temporal and spatial memory for gait pattern recognition. In Proceedings of the IEEE Computer Society, IEEE Applied Imagery Pattern Recognition Workshop, Washington, DC, USA, 18–20 October 2016; pp. 1–9.
19. Hawkins, J.; Ahmad, S.; Dubinsky, D. Hierarchical Temporal Memory Including HTM Cortical Learning Algorithms. September 2011. Available online: https://numenta.com/assets/pdf/whitepapers/hierarchical-temporal-memory-cortical-learning-algorithm-0.2.1-en.pdf (accessed on 11 July 2019).
20. Wu, J.; Zeng, W.; Chen, Z.; Tang, X. Hierarchical Temporal Memory Method for Time-Series-Based Anomaly Detection. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW), Barcelona, Spain, 12–15 December 2016; pp. 1167–1172.
21. Ahmad, S.; Lavin, A.; Purdy, S.; Agha, Z. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing* **2017**, *262*, 134–147. [CrossRef]
22. Purdy, S. Encoding Data for HTM Systems. February 2016. Available online: https://arxiv.org/abs/1602.05925 (accessed on 14 July 2019).
23. Cui, Y.; Ahmad, S.; Hawkins, J. The HTM spatial pooler–a neocortical algorithm for online sparse distributed coding. *Front. Comput. Neurosci.* **2017**, *11*, 111. [CrossRef]

24. 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]
25. 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.