An Intrusion Action Based IDS Alert Correlation Analysis and Prediction Framework

Kai Zhang¹, Fei Zhao¹, Shoushan Luo¹, Yang Xin¹,², Hongliang Zhu¹
¹National Engineering Laboratory for Disaster Backup and Recovery, Information Security Center, School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China;
²Guizhou Provincial Key Laboratory of Public Big Data, Guizhou University, Guizhou 550025, China

Corresponding author: Kai Zhang (e-mail: kaikai4006@163.com).

This work was supported in part by the National Key R&D Program of China under Grant 2017YFB0802300, in part Major Scientific and Technological Special Project of Guizhou Province under Grant 20183001, in part Foundation of Guizhou Provincial Key Laboratory of Public Big Data under Grant2018BDKJJ021.

ABSTRACT Since the rapid development of the internet, the emergence of network intrusion has become the focus of studies for scholars and security enterprises. As an important device for detecting and analyzing malicious behaviors in networks, IDS (Intrusion Detection Systems) is widely deployed in enterprises, organizations and plays a very important role in cyberspace security. The massive log data produced by IDS not only contains information about intrusion behaviors but also contains potential intrusion patterns. Through normalizing, correlating, and modeling data, we can obtain the patterns of different intrusion scenarios. Based on the previous works in the area of alert correlation and analyzing, this paper proposed a framework named IACF (Intrusion Action Based Correlation Framework), which improved the process of alert aggregating, action extraction, and scenario discovery, and applied a novel method for extracting intrusion sessions based on temporal metrics. The proposed framework utilized a new grouping method for raw alerts based on the concept of intrinsic strong correlations, rather than the conventional time windows and hyper alerts. For discovering high stable correlations between actions, redundant actions and action link modes are removed from sessions by a pruning algorithm to reduce the impact of false positives, finally, a correlation graph is constructed by fusing the pruned sessions, based on the correlation graph, a prediction method for the future attack is proposed. The experiment result shows that the framework is efficient in alert correlation and intrusion scenario construction.

INDEX TERMS alert correlation, multistep attack, correlation analysis, intrusion scenario, attack prediction, IDS alerts

I. INTRODUCTION

With the rising consciousness of information security, people paid more attention to the network security and established basic network security software and hardware infrastructures, such as IDS (Intrusion Detection Systems), IPS (Intrusion Prevention System), firewalls, and professional security systems e.g. EWS (Early Warning System), and so on.

For large organizations, massive alerts are generated every day. It becomes a difficult task to discover the logical relations between alerts, especially to construct the intrusion scenarios in a higher abstract level. Therefore, studies in this field are focused on how to reduce redundant alerts that heavily impact the result of the analysis, how to extract attacker’s behavior patterns from the raw alerts, and how to understand and explain intrusion incidents in a semantic level.

IDS based on misused detection technology or anomaly-based detection technology both have some shortcomings, such as high false-positive rate, duplicated alerts, and the low detection rate on the multistep attacks. For the NIDS (Network Intrusion Detection System), which is focused on analyzing and matching network packet-level anomalies, they can hardly discover the relations between packets sent by the same attacker for the shortage of context information, as a result, it is difficult to give a whole view of multistep attacks.

Multistep attack refers to the ensemble of attack steps taken by one or several attackers with a single specific objective inside the network, containing at least two distinct actions [1], which is the opposite of a single-step attack.
other words, the attacker needs to perform multiple single-step attacks to achieve the final goal of the intrusion. Malicious attackers tend to carry out a multistep attack for reasons: first, the targeted network or host belongs to a large organization which is protected by layers of security technologies, so attackers need to try different means several times; second, if attack process is decomposed into multiple steps, it will be more difficult to be discovered by victims.

The study of multistep attack detection is confronting the data challenges: first, the increasing complexity of data created by network makes it difficult to filter out data that is not related to security; second, only a small number of intrusion samples are available for training system in a data set because many attacks rarely occur in a long time range, although the data set is large, the samples are limited, which is called rare data problem.

Most detection methods utilize alerts as the input data, which bring some challenges: the huge number of alerts generated in a small-time slice making it difficult to design real-time systems; Alerts that indicating real attacks is often mixed with massive unrelated alerts; IDS from different vendors generate alerts with different structures.

The natures of multistep attack also become a challenge: each of the elementary attacks consisting multistep attack seems to be ordinary behavior; The detection of multistep attacks should not be regarded as the traditional plan recognition, because there are no fully predefined models for multistep attack detection [2]; a multistep attack may be interrupted at any time, only because the intruder loses interest or fails to find the weakness of the target network; There are many good reasons for some elementary attacks that are missed; An intruder does not need to carry out a series of attacks with a strict order, which will lead to a complex alert sequence [3].

To meet the challenges above and enhance the efficiency of analyzing, plenty of works have been done in this paper:

First, raw alerts are normalized and then extracted into intrusion actions by the extraction module. Every intrusion action is composed of a sequence of alerts that having a strong intrinsic correlation, just like the “metadata” of action. The system can compare two actions based on alert types that each action contains, and aggregate actions with the same alert types.

Second, two algorithms are proposed for intrusion session rebuilding: TSS (Time-lag based Sequence Splitting) and SPA (Sequence Pruning Algorithm). TSS is used for splitting a long sequence of intrusion actions into shorter sessions by calculating the time boundaries of actions. The time window technologies involved in many studies are not utilized in TSS, because it can retain more information about the real intrusion process than other approaches. SPA is another important algorithm for pruning sessions extracted by TSS. Duplicated actions or action modes need to be pruned out for a compact and precise sequence. Real correlations between intrusion actions are exposed significantly by compact sessions.

Third, in the phase of constructing intrusion scenarios, a method for session fusion is proposed based on an assumption: intrusion sessions that having one common binary relation tend to integrate the intrusion scenario. The intrusion sessions which are selected to build an intrusion scenario are finally decomposed into binary relations which will be added to a correlation graph. At the same time, the frequencies of binary relations are recorded for the attack prediction.

The rest of this paper is organized as follows: Section 2 summarizes the related works in this field; In Section 3, the details of proposed framework are introduced, along with methods for intrusion action extraction, session rebuilding, correlation graph construction; Section 4 reports the results of evaluation experiments for the framework and algorithms. Finally, in section 5, conclude the paper.

II. RELATED WORKS

A lot of works have been done in the area of alert correlation and scenario construction.

One kind of the most important approaches is similarity-based methods. The similarity between alerts is computed with one or more fields: IP addresses, ports, types and times. Similarity-based techniques reduce the total number of alerts by clustering and aggregating methods. This kind of approach is mostly based on the assumption: similar alerts tend to have the same root causes.

A method is proposed by authors of literature [4] to define the degree of association by calculating the similarity of two alerts, and it used MLP (Multi-Layer Perceptron) and SVM (Support Vector Machine) technologies to improve the accuracy of the result. Reference [5] proposed a method for alert aggregation and correlation based on clustering. The author of [6] proposed an analyzing framework named RTECA which groups alerts into hyper-alerts by alert types and similarities are calculated which are stored in the CCM (Causal Correlation Matrix). In literature [20], a method based on hierarchical clustering is proposed which convert the intrusion cases into graphs and calculate the “Jaccard distance” between them, and finally cluster them into simpler ones.

Some similarity-based techniques utilize temporal constraints to aggregate and correlate alerts. The main idea of temporal correlation is to determine the association strength by analyzing the temporal information (time interval or time-shift or time-lines) of alerts. Most of this kind of methods focused on discovering the association rules in a temporal perspective and relied on time-windows to group the alerts. Ma et al. [7] proposed a method that involves several kinds of time horizons to correlate alerts, to predict the upcoming step of multistep attacks in real-time. Hossein et al. [8] utilize several time-windows to avoid comparing the whole set of received alerts with new alerts, and then involve a probability
calculating process to obtain a threshold value which impacts the correlation result. A method applying a time series-based statistical analysis is proposed by Qin and Lee [2] to determine whether two alerts are correlated or not.

Most of the similarity-based approaches are implemented with light-weight algorithms of less complexity which is based on the simple comparison, and these algorithms show high efficiency in reducing the number of alerts. However, they also have some weaknesses: it is difficult to find a suitable metric of similarity, and the accuracy of similarity-based correlation analysis methods is not high enough.

Another kind of important approach is based on data-mining technologies. The data mining-based technologies aim to find out frequent sequential patterns or sub-patterns that are concealed in the massive alerts. Many attempts have been made on association rule mining and frequent episode mining. Stream mining or Sequential mining technologies are involved in some approaches too.

One stream mining-based method proposed by Farhadi et al. [9] can predict future events by Hidden Markov Model (HMM). By calculating alert frequency in different time windows, they construct attack scenarios and analyze the correlated alerts. Finding the most frequent patterns is also the aim of Brahami and Yah et al. [10] who apply an improved version of the Prefix Span algorithm by distributing the alerts and their attributes in multi-dimensional tables. The RMARS framework is proposed by Bahareth and Bamasak [11], which involves a standard sequential pattern mining algorithm named GSP to extract patterns from a training dataset. The patterns can later be used in real-time detection.

As reported by Martin Husák et al. [12] that data mining technologies are gaining a lot of attention in the fields of cybersecurity and alert association mining. However, some aspects of this kind of technology need to be enhanced: it cannot be ignored that the time-consuming problem which is derived from a large number of patterns being counted and matched by the system. A large number of redundant patterns are discovered with rare interesting patterns.

Sequential-based methods are useful to model and analyze complex attack scenarios from sequences of individual events or steps that are a part of the same attack scenario. Pre/Post-conditions and graphs are two mostly involved technologies [13].

A real-time alert correlation model named RIAC is proposed by Zhaowen et al. [14] to analyze and extract attack scenarios from alerts. It is based on one assumption that there is a prepare-for relation between the elementary attacks which means one attack is a pre-condition for the next attack. To identify multiple attack scenarios, Roschke et al. [15] proposed a Floyd–Warshall algorithm to find out the shortest paths in an attack graph in which, each node represents a single attack step. Zhang Yongtang et al. [16] proposed a mining method for attack sequence. They define the attack scenario as the largest sequence that cannot be “contained” by other sequences.

Time-window technologies ([6], [7], [8], [9], [20], [21], [22], [24]) are widely applied in alert correlation fields for grouping and aggregating alerts. It is efficient and simple to be implemented and is suitable to deal with massive time-related data especially in real-time. However, a weakness of time windows that impact the accuracy of analysis may come up: the size of time-window can hardly be set properly for an intrusion action sequence that constantly changing in the time interval, it may cause an abnormal interruption in a data flow of intrusion session. Li Wang et al. [22], noticed it and proposed extension time window technology to solve the problem, but the threshold for extending the time window is fixed, which reduced the flexibility of the algorithm.

Hyper alert ([2], [6], [20], [22]) is another basic method applied in many kinds of literature for aggregating alerts by alert type. Although it is efficient in reducing the homologous alerts, it destroyed the logical correlations of alerts that belong to the same intrusion session.

This paper is inspired by the above literature and proposed several improved methods for alert grouping, aggregating, and an efficient framework for correlating IDS alerts is proposed which can predict the upcoming steps of attack with high accuracy.

III. PROPOSED ALERT CORRELATION FRAMEWORK

In this section, five main aspects of the proposed framework are introduced: intrusion action extraction, intrusion session rebuilding, session pruning algorithm, intrusion scenario construction, attack prediction. These functions are built according to the procedures of alert correlation analysis and prediction.

A. SOME TERMS AND EXPLANATIONS

There are various terms appeared in published literature describing similar concepts. In order not to confuse them, a short explanation for each term appeared in this paper are listed below.

alert(s): an alarm text generated by IDS when discovered a suspicious event in the network, which can be recorded in IDS log files or database. Sometimes we use raw alert(s) instead in the rest of this paper.

Alert (the first letter is capitalized): an object that is normalized from the raw alert. It can be stored or managed by the computer system in a unified way. Other literature calls it “Event,” “Alarm,” “Warnings,” and so on. Sometimes, the term Alert object is used as the same meaning in this paper.

intrusion(s): another shorter name for a multistep attack in this paper which describes a kind of complex attacking process that contains a series of intrusion actions with the same purpose.

intrusion action(s): that indicates a step of the complex intrusion process. Intrusion action can be represented by a
series of Alert objects. Sometimes action for short in this paper.

intrusion session(s): a sequence of intrusion actions that happens between a pair of hosts. Intrusion sessions are rebuilt with a temporal constraint. Sometimes session for short in this paper.

B. FRAMEWORK INTRODUCTION

As reported by Julio Navarro et al. [1] in their article, multistep attack detection is an active field of research, IDS alert correlation and analysis is one of the most important technologies for multistep attack detection. Various methods are proposed to deal with the problems in alert correlation and attack prediction. However, the methods that both have a high performance in analysis and prediction are hardly to achieve, that is why this field keeps active for a long time until now.

The framework IACF (Intrusion Action Based Correlation Framework) proposed in this paper, contains two main phases: extraction phase and modeling phase. In the extraction phase, the framework extracts intrusion actions and sessions; In the modeling phase, the correlation graph is constructed by sessions that extracted in the extraction phase.

The components of the framework are shown in Fig. 1:

In the beginning, we assume that alerts generated by multiple security sensors deployed in different security domains continuously arrive at the system. The alerts need to be normalized into the same structure for the sake of consequential proceedings. The normalized alerts are then sent to the action extraction module.

In the process of action extraction, alerts are grouped by source IP address and destination IP address, as a result, the alerts in the same group are derived from the communications between a pair of IP addresses. The framework then starts to extract intrusion actions following a set of rules from each group of alerts. At the end of this process, extracted actions are verified to aggregate the ones indicating the same action but with different alert sequence. The extracted actions are finally sent to the module for session extraction.

In the session extraction process, action sequences are split into shorter ones from a temporal perspective. Note that IACF will create a FIFO queue (First-In-First-Out Queue) to cache actions for each pair of IP addresses, and newly extracted actions will append to that queue. IACF will check the time intervals between actions to identify the start or endpoint of a real intrusion session. The extracted sessions will be pruned before sent to the modeling component.

In the modeling phase, one correlation engine begins to deal with the pruned sessions. It groups the sessions that have common binary relations, and each group of sessions is decomposed into binary relations to construct a correlation graph. The existing graph model can be updated with the new extracted sessions in a training phase until the graph model is stable enough for prediction.

For predicting a future attack, IACF can give out several prediction results indicating the probabilities of the next attack intention. Unlike other studies, the prediction results exported by IACF, which indicates some kinds of intentions of an attacker that may happen in the future but not an exact intrusion action for the next step even though the prediction is described as a sequence of actions. The human security experts can easily infer the intruder’s real intention through the prediction results.

In this paper, we will explain the details of the IACF. Some useful algorithms are proposed, such as session pruning algorithm and prediction algorithm and other optimized processes for correlation and analyzing. As the evaluation experiments show, the framework is efficient and accurate in alert correlation and attack prediction.

C. INTRUSION ACTION EXTRACTION

Many kinds of literature calculate the association strength between alerts or hyper-alerts by similarity-based or data mining-related technologies. Most of these methods are based on statistical methods, through discovering the frequent items and calculating the confidence factor to measure association strength. Although these frequency-based methods have a good performance in correlation analysis and intrusion scenario construction, they are used in an apparently unreasonable way, because they ignored an important feature of raw alerts: “intrinsic correlation.”

There is a strong correlation within alerts, which can also be called “atomicity”. This important feature is derived from IDS detecting and alarming mechanism. As we know, misuse-based IDS detects anomaly behaviors by comparing network packets with the predefined signatures. One situation is that there is a big probability that one packet matches more than one signatures, which will trigger multiple different alerts. Another situation is that one piece of attack may contain several instructions encapsulated in different packets that also triggers multiple alerts. A group of alerts triggered by these two situations have an “intrinsic correlation” between them, in other words, they are indivisible to describe one specific intrusion action. Many other technologies, find correlations by applying the statistical methods directly on the raw alerts or hyper-alerts which ignored the atomicity of alerts, that is why they are unreasonable.

Based on the concept of atomicity, we can group alerts differently: the intrusion action-base grouping method. This method takes full account of atomicity and assumes that one
class of attack instructions may trigger the same set of alerts no matter how they are ordered.

An Alert object which is described in section 3.1 can be defined as Alert<time, type, source-ip, source-port, dst-ip, dst-port, protocol>. Sometimes, we use the field type to denote the name of the Alert object.

An intrusion action object can be defined as Action<start-time, end-time, alerts>. The field alerts is a sequence with a set of Alert objects in it; the value of start-time is derived from the time field of the first Alert object in traces, and the value of end-time is derived from the time field of the last trace object in traces.

This paper assumes that raw alerts arrive at system in real-time; two points need to be noted when extracting intrusion actions: 1) source-ip and source-port are not used for extracting actions, because they are easy to be tampered with by malicious attacker; 2) protocol is also excluded for action extraction, because most of the alerts involve only a few kinds of protocols, in other words, this field makes action extraction process coarse-grained.

Based on the observation of data, the fields dst-port and type can be used as combination constraints for extracting actions. For example, if two alerts with the same dst-port or alert type that occur successively in a sequence will be grouped into the same action, otherwise, they will be grouped into different actions.

FIGURE 2. Raw alerts triggered by 202.77.162.213 and 172.16.112.10

When the first alert (row no. 302, r302) arrives, it is appended to a new action: A: A = \{r302\}, when the second alert (r303) arrives, system compares r303 with r302 which is the last alert of A, r303 is appended to A because it has the same dst-port as r302, and r304 is also appended to A for the same reason, and now A = \{r302, r303, r304\}. When r305 arrives, it compares r305 with r304 which is different in both type and dst-port. In this situation, a new action B is created and r305 is appended to B rather than A, and B = \{r305\}. Following the same rules, intrusion actions are extracted one by one.

Many duplicated actions are created because of duplicated alerts and false positives. A matching method is needed for comparing or searching extracted actions. The essential differences between the two actions exist in their Alert sequence.

“Alert Set”: A minimum set of Alert types based on which various Alert sequences are constructed by duplicating one or several subsets of it. For example, if \(X = \{a, b, b, a\}\), its Alert set can be denoted: \(|X| = \{a, b\}\). An action’s Alert set is not the same as it’s Alert sequence because no repeated elements are allowed in the Alert set. In fact, the Alert set contains all distinct elements of an Alert sequence that representing an action.

For two actions \(X, Y\), if \(|X| \subseteq |Y|\) and \(|X| \subseteq |Y|\), \(X\) and \(Y\) are the same actions. It can be explained that the same intrusion action tends to trigger a similar type of alert. Two actions are the same because of the same type of Alerts rather than the fully equaled Alert sequence. With this rule, actions can be matched by comparing their Alert sets.

D. INTRUSION SESSION REBUILDING

Intrusion session rebuilding aims to expose real correlations hidden in actions. The association relation between two actions is significant when they take place one after another in the same intrusion session, therefore, it is an important task to find out intrusion sessions composing an intrusion scenario.

During the action extraction process, actions are extracted from an alert sequence derived from a pair of hosts in real-time. That means extracted actions also belong to the session of that pair of hosts.

SRC (Session Rebuilding Component) works based on an assumption: each intrusion action belongs to an intrusion session. SRC has a FIFO queue to which newly extracted action is appended, and a set of actions that indicating a new intrusion session can be cut off the head of the queue. The key problem is how to identify the cutting point.

A series of intrusion actions done by an attacker is more concentrative in the temporal dimension than random false positives. Based on this, SRC can extract intrusion sessions by calculating the discreteness of time intervals between actions. Although a large number of alerts can be generated in a small-time slice by a high-performance IDS, the number of alerts triggered by a pair of hosts is smaller and their time intervals can be measured especially after the process of action extraction.

For session rebuilding, TSS (Time-lag based Sequence Splitting) algorithm is proposed which can determine the cutting point for a session. TSS is not an accurate algorithm that can extract real attack sessions precisely, but a method more reasonable for organizing intrusion actions than time-windows for its effort in preventing irrational division.

Fig. 3 shows the distribution of intrusion actions on the timeline.

A long sequence of actions on the time line

FIGURE 3. Distribution of actions on the timeline. Arrow denotes a timeline; Black round dots denote actions; Double slashes denote the cutting point.
TSS works on a flow of actions as input and output intrusion sessions. Consider an input sequence with more than two actions: \( S = \{ A_1, A_2, \ldots, A_n \} \), TSS needs to determine whether an action \( A_p \) of \( S \) belongs to the subsequence \( S' = \{ A_1, A_2, \ldots, A_p \} \) or \( S'' = \{ A_{p+1}, A_{p+2}, \ldots, A_n \} \). To meet this question, some equations are presented below:

**Equation 1:** \( T_{\text{lag}} \) function is used to calculate the time interval between two actions: \( A_k \) and \( A_m \) in a sequence (\( A_k \) happens before \( A_m \)), and \( \text{start}_t() \) and \( \text{end}_t() \) are two functions to retrieve the start-time and end-time of an action object.

\[
T_{\text{lag}}(A_k, A_m) = \text{start}_t(A_m) - \text{end}_t(A_k)
\]

**Equation 2:** \( T_{\text{avg}} \) function is used to calculate the average time interval of a sequence with a header \( A_k \) and a tail \( A_m \).

\[
T_{\text{avg}}(A_k, A_m) = \frac{\sum(T_{\text{lag}}(A_k, A_{i+1}))(m-k)}{m > k}
\]

**Equation 3:** \( T_{\sigma} \) function is used to calculate the standard deviation of time intervals of a sequence with a header \( A_k \) and a tail \( A_m \) (\( A_k \) happens before \( A_m \)).

\[
T_{\sigma}(A_k, A_m) = \sqrt{\frac{\sum (T_{\text{lag}}(A_k, A_{i+1})-T_{\text{avg}}(A_k, A_m))^2}{(m-k)}}
\]

If the variation of the time interval between \( A \) and \( B \) is more than \( N (N > 1, N \in \mathbb{Z}) \) times of \( T_{\sigma} \), which means a cutting point of current intrusion session has come out, and TSS will end the current session with \( A \) and create a new session with \( B \) as its first action. The value of \( N \) is preset by the user when the system starts up. The operations of TSS are as follows:

1. Create a new session: \( S_m \) (\( m \geq 1, m \in \mathbb{Z} \));
2. Append \( A_k \) and \( A_k+1 \) to \( S_m \) (\( k \geq 1, k \in \mathbb{Z} \));
3. When \( A_{k+2} \) arrives, it calculates the standard deviation: \( E = T_{\sigma}(A_k, A_{k+1}) \) and \( \Delta t = |T_{\text{lag}}(A_{k+1}, A_{k+2}) - T_{\text{avg}}(A_k, A_{k+1})| \);
4. If \( \Delta t = N \times E \), append \( A_{k+2} \) to \( S_m \) and set \( k = k + 1 \) then go to step 3);
5. If \( \Delta t > N \times E \), set \( m = m + 1 \) and go to step 1);
6. TSS will not stop until it arrives at the end of the sequence;

The generated sessions will be used to construct intrusion scenarios after the pruning process which is described in the next subsection.

### D. INTRUSION SESSION PRUNING

The “action mode,” which refers to a subsequence of actions that continuously appears in an intrusion session more than once, should be pruned off to expose the essential correlations of actions. It is an effective method to investigate potential relations in a long and complex sequence. It is of benefit to pruning off redundant action modes hidden in a session for three reasons: 1) The duplicated actions cannot provide more information about an intruder; 2) Minimize the impact of redundant modes and false positives; 3) Shorter action sequences make a good performance of the system.

One requirement that should be met by the pruning algorithm is that no more redundant modes exist in a pruned session. In fact, it is a difficult task to find out all the redundant modes hiding in a sequence, because a new mode may be formed after a round of pruning.

Table I shows several sequences that should be pruned. In fact, the potential algorithm can prune any sequences not limited to the ones listed in Table I.

<table>
<thead>
<tr>
<th>Sessions BEFORE and AFTER PRUNING</th>
<th>Before Pruning</th>
<th>After Pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A \rightarrow B \rightarrow C )</td>
<td>( A \rightarrow B \rightarrow C )</td>
<td>( A \rightarrow B \rightarrow C )</td>
</tr>
<tr>
<td>( A \rightarrow A \rightarrow B \rightarrow C \rightarrow C )</td>
<td>( A \rightarrow B \rightarrow C )</td>
<td>( A \rightarrow B \rightarrow C )</td>
</tr>
<tr>
<td>( A \rightarrow B \rightarrow C \rightarrow A \rightarrow B \rightarrow C )</td>
<td>( A \rightarrow B \rightarrow C )</td>
<td>( A \rightarrow B \rightarrow C )</td>
</tr>
<tr>
<td>( A \rightarrow B \rightarrow C \rightarrow D \rightarrow A \rightarrow B )</td>
<td>( A \rightarrow B \rightarrow C \rightarrow D \rightarrow A \rightarrow B )</td>
<td>( A \rightarrow B \rightarrow C \rightarrow D \rightarrow A \rightarrow B )</td>
</tr>
</tbody>
</table>

A Sequence Pruning Algorithm (SPA for short) is proposed to achieve the goals described above. The basic idea of SPA is to remove the adjacent duplicate modes of a sequence iteratively until no more duplicate modes appear again. Fig. 4 demonstrates how SPA works:

SPA starts an iteration when \( m = 1 \), the rest procedures are as follows:

When \( m = 1 \) (Fig. 4(a)), the sliding window size is set to \( w = m + m = 2 \). The sliding window (the rectangle box in Fig. 4) slides one element each time from left to right side of the sequence \( S_m \). If the two sets of element (separated by vertical dotted line) in sliding window are identical to each other, the positions of the first set of elements (letters with backslash in Fig. 4) are recorded and will be removed in the future when current iteration is over, and the sliding window goes on to slide to the next element and continues to perform another matching.

When \( m = 2 \) (Fig. 4(b)), SPA starts the second iteration and sets the sliding window size \( w = m + m = 4 \), the rest operations are the same as \( m = 1 \). SPA goes on until \( m = 9 \) which is more than half of the input sequence.

The code of SPA algorithm contains two functions: main function and the pruning function as follows:
Algorithm 1-1. Main Function of SPA

Require: \( S_n = \emptyset; S_m = \emptyset; S_{\text{size}} \geq 2; \)
set \( m = 1; n = S_{\text{size}} + 2; \)
while \( m \leq n \) do
    \( S_n = \text{prune} (S_n, m); \)
    \( m = m + 1; \)
    \( n = S_{\text{size}} + 2; \)
end while;

Algorithm 1-2. Prune Function of SPA (optimized)

Require: \( S_n \neq \emptyset; S_{\text{size}} \geq 2; m > 0; \)
set header = 0; tail = header + m; times = 0;
set sames = \emptyset; removes = \emptyset; nOuterWhile = false;
while tail < \( S_n \cdot \text{size} \) do
    set fullmatch = false;
    While true do
        remove all elements of sames;
        if tail > \( (S_n \cdot \text{size}) \cdot (-m) \)
            endOuterWhile = true;
        break;
    end if
    set i = m - 1; matchtimes = 0;
    While i \geq 0 do
        if \( S_n \cdot [\text{header} + i] = S_n \cdot [\text{tail} + i] \)
            append (header + i) to sames;
            matchtimes = matchtimes + 1;
        else
            mismatch = header + i;
            break;
        end if
    end while
    if matchtimes \( = = m \)
        append all elements of sames to removes;
        remove all elements of sames;
        header = header + m;
        tail = header + m;
        continue;
    else
        fullmatch = false;
        break;
    end if
end while
if endOuterWhile \( = = \) true
    break;
end if
if fullmatch \( = = \) false
    header = mismatch + 1;
    tail = header + m;
end if
end while
remove all elements of \( S_n \) appeared in removes;
return \( S_n \);

To improve the efficiency of the SPA algorithm, two optimizing suggestions that can greatly reduce the matching times are given below:

1) When matching two sets of elements in sliding window, it is advised to start matching the last element of each set (the elements pointed at by arrows in Fig. 5), because if they do not match, sliding window can directly slide to the position of the dotted rectangle box that is next to the first unmatched element in Fig. 5, rather than sliding one element each time without skips.

2) For a long single-element-sequence that exceeds the sliding window size, the sliding window can temporarily extend for \( m \) each time until a different element appears in the window, and shrinks back after recording all the repeated elements in it.

FIGURE 5. Optimizing strategy for the SPA algorithm. Start matching the last pair of actions at which arrows are pointing rather than the first pair of actions of each set in the sliding window.

The evaluation experiment shows that the performance of the optimized SPA is significantly improved due to the skipping sliding window mechanism. The algorithm has several features as bellows: 1) With a high compression ratio because newly formed modes are considered; 2) Simple enough to be implemented and understood; 3) In the worst case: there is no redundant mode exiting in a sequence which means SPA need to do the largest number of matching. The time complexity is \( O(n^2) \).

The pruned intrusion sessions that contain the rich type of actions have a high probability to be a real intrusion session. The pruned sessions with few kinds of actions may be a single-step attack or false-positive which will be filtered out by the system.

E. INTRUSION SCENARIO CONSTRUCTION

In the multistep intrusion scenario, intruders usually communicate with multiple targets and generate multiple intrusion sessions. These sessions should be fused in a way to construct the intrusion scenario. Many methods are proposed for graph-based scenario construction, Ali Ahmadian Ramaki et al. [6] define the intrusion scenario as an attack tree. Bo-Chao Cheng et al. [17] convert an intrusion scenario into a special line chart which can be used to match a potential scenario with a known one. Feng Xuewei et al. [18] leverage the Markov chain to form a graph of a scenario with causal knowledge.

In this paper, a kind of correlation graph model is proposed to represent an intrusion scenario, which is constructed by binary correlations extracted from the pruned sessions.

Binary correlation is an important concept which refers to the adjacent relationship of two actions that happen successively in a session. If \( A \) happens before \( B \), it is denoted as \( A > B \) (\( A \) is the necessary condition of \( B \)), whereas \( B > A \).
(B is the necessary condition of A); if $A > B$, and $B > C$, the relationship between $A$, $B$ and $C$ can be denoted as $A > B > C$. Especially, if $A > B$ happens in session $S$ and $B > C$ happens in session $S'$, the conclusion: $A > B > C$ can also be obtained, because when an intruder successfully perform $B$, the potential condition for $C$ is available too, in other words, if $A > B$ happens, $C$ has a certain probability to happen as a candidate action.

Note that a session can be decomposed into several binary correlations, for example, session $S$: $A > B > C > D$ can be decomposed into three binary correlations: $A > B$, $B > C$, $C > D$, and we can say $S$ contains a binary correlation $B > C$.

The correlation graph can be defined as a directed acyclic graph which is denoted as $G = \langle V(n), E(f) \rangle$, where $V$ indicates the node-set and $E$ indicates the edge set of a graph. Each node in the graph represents an intrusion action, while the edge between two nodes indicates a binary correlation between two actions. For node $V(n)$, $n$ denotes the action’s name and $f$ in $E(f)$ denotes the frequency of a binary correlation.

The construction of the correlation graph is based on an assumption that: if two sessions share at least one action with each other, they belong to the same scenario and should be added to the same graph. There are two phases for constructing a correlation graph: the training phase and the pruning phase. In the training phase which can be implemented as a mining process offline, the system constructs the graph through analyzing and correlating the historical data. In the pruning phase, which is implemented as a modification process online.

Fig. 6 shows the main operations of graph construction in the training phase. The letter “x” in Fig. 6 denotes an action of a randomly selected session and “g” denotes a correlation graph. The graph is constructed by searching actions in the session pool that have a binary correlation with $x$ and add them to the graph iteratively. In the training phase, all actions of sessions with at least one common action will be added to the same graph.

For example, Fig. 7 shows four intrusion sessions ($S_1$, $S_2$, $S_3$, $S_4$) selected to build a graph for they all share a common action $D$. Firstly, the system adds $A$ to a newly created graph $g$, and search the actions having a binary correlation with $A$ in these four sessions. Only $B$ has a binary correlation with $A$, so $B$ is added to $g$, and $g = \{A, B\}$ as shown in Fig. 7(a); Secondly, system begins to search actions having binary correlation with $B$ in the four sessions, only $C$ is added to $g$, and $g = \{A, B, C\}$ (Fig. 7(b)); Both $H$ and $D$ have a binary correlation with $C$, so $H$ and $D$ are added to $g$, and $g = \{A, B, C, D, H\}$ (Fig. 7(c)); The constructed graph is the same as Fig. 7(d).

![FIGURE 7. Procedures of scenario construction in the training phase. (a), (b), (c), and (d) are four steps of the construction.](image)

It does not affect the structure of the graph to start the construction with which action. Finally, all constructed graphs are stored in a database for future retrieving.

The graph pruning phase which is an online process to update and improve the graph model in real-time. The tasks for the pruning phase are to fuse new extracted sessions into the graph and remove the leaf nodes that rarely happened within a long time.

**F. INTRUSION PREDICTION**

The frequencies of each binary correlation are collected for predicting future attacks while building a correlation graph. In Fig. 7 (d), the number next to the edge denotes the frequency of the binary correlation.

We use $F(A > B)$ to denote the frequency of correlation $A > B$, and $F(A > *)$ to denote the total number of the binary correlations that starting with $A$ in a graph, as shown in Fig. 7(d), $F(D > *) = 3$. $F(*) > B$ denotes the total number of binary correlations ending with $B$, In Fig. 7(d), $F(*) > C) = 3$. $P(A, B)$ denotes the probability of the correlation $A > B$ which can be calculated based on correlation graph with Equation 4 as follows:

$$P(A, B) = \frac{F(A > B)^2}{F(A > *) \times F(*) > B} \quad (4)$$

As in Fig. 7(d), $P(B, C)$ can be calculated: $F(B > C) = 2$; $F(B > *) = 2$; $F(*) > C) = F(B > C) + F(H > C) = 3$; finally, $P(B, C) = 2/3 = 66.7%$. Similarly, $P(D, F) = 1/3 = 33.3%$. 

![FIGURE 6. Procedures of scenario construction in the training phase. It constructs correlation graphs by adding distinct action nodes that having a binary correlation with the just added nodes to the graph iteratively.](image)
Sometimes, we want to calculate the probability of a sequence \( BCDE \) happens after \( A \), which can be denoted as \( P(A, BCDE) \). Equation 5 is as follows:

\[
P(A, B_1...B_n) = P(A, B_1) \times P(B_1, B_2) \times ... \times P(B_{n-1}, B_n) (5)
\]

In Fig. 7(d), \( P(A, BCDE) \) can be calculated: \( P(A, B) = 1; P(B, C) = 2/3; P(C, D) = 1; P(D, E) = 2/3 \); finally, \( P(A, BCDE) = 44.4\% \).

IV. IMPLEMENTATION AND EVALUATION

In this part, the experimental evaluations of the proposed framework are performed and the reports are shown. Three dimensions are used to evaluate the framework: the accuracy of multistep attack scenario recognition, the performance evaluation of related algorithms and the accuracy of prediction.

A. EXPERIMENTS SETUP

The preset values of the framework are listed in Table II. The framework is implemented with Java programming language and runs on a server with 8GB RAM and 2.6GHz Intel CPU, on which Windows 10 operating system is installed. In addition, snort (a free IDS software) is used to read, detect network data and generate raw alerts.

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Parameter Names</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_s )</td>
<td>Alert type similarity threshold</td>
<td>0.8</td>
</tr>
<tr>
<td>( L_c )</td>
<td>Alert cache</td>
<td>1000</td>
</tr>
<tr>
<td>( P_s )</td>
<td>Pool size</td>
<td>500</td>
</tr>
<tr>
<td>( Q_s )</td>
<td>FIFO queue size</td>
<td>100</td>
</tr>
<tr>
<td>( T_o )</td>
<td>Initial time interval standard deviation (minute)</td>
<td>10</td>
</tr>
<tr>
<td>( N )</td>
<td>N (multiple of standard deviation)</td>
<td>3</td>
</tr>
<tr>
<td>( M_s )</td>
<td>Minimum session size</td>
<td>2</td>
</tr>
</tbody>
</table>

TABLE II

PRESET VALUES FOR Initializing THE FRAMEWORK

B. EVALUATION ON LLSDOS 1.0

DARPA 2000 intrusion detection evaluation data set [19] is often used to test and evaluate the correlation analysis system for discovering multistep attack scenarios. It has two versions of intrusion scenario data sets: LLSDOS 1.0 and LLSDOS 2.0. Snort will read in the data set and generate alerts that are finally recorded in a log file. The data set contains a complete multistep intrusion scenario: three victims inside the network are intruded by the attacker outside the network and launch a DDOS attack. The steps of the scenario are as follows:

1) Through sending ICMP echo-requests, attacker discovered the hosts that alive;
2) The hosts are probed to find the ones running “sadmind” remote administration tools that might be vulnerable;
3) Three hosts are intruded through the remote buffer-overflow attack, and sadmind remote-to-root exploit;
4) Perform a telnet login and install DDOS control software;
5) Launch a DDOS attack by sending “mstream” instructions through a telnet session.

In this scenario, the first two steps are performed by intruders targeting almost all the hosts inside the network. The last three steps are performed targeting three victims: 172.16.115.20, 172.16.112.10 and 172.16.112.50.

Based on the data set, snort generated about 23300 raw alerts, 80% of which are background noises and false positives, and are grouped into 325 sets by source IP and destination IP. After the process of action extraction, 3934 intrusion actions of 31 different types are extracted. 336 intrusion sessions are generated, and 321 sessions contain no more than 3 actions after pruning and only 3 sessions have more than 5 actions, however, they provide the most information about the DDOS scenario. The framework based on the SPA algorithm can be greatly efficient in discovering intrusion scenarios. At last, three correlation graphs are generated based on the data set, and only one graph contains more than three nodes. Statistical information is listed in Table III as follows:

<table>
<thead>
<tr>
<th>TABLE III</th>
<th>RUNNING RESULTS OF PROPOSED FRAMEWORK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Names</td>
<td>Values</td>
</tr>
<tr>
<td>Total raw alerts</td>
<td>23300</td>
</tr>
<tr>
<td>Alert groups</td>
<td>325</td>
</tr>
<tr>
<td>Extracted actions</td>
<td>3934</td>
</tr>
<tr>
<td>Action types</td>
<td>31 (0.79% of total actions)</td>
</tr>
<tr>
<td>Distinct raw alerts used to construct actions</td>
<td>40 (0.17% of total alerts)</td>
</tr>
<tr>
<td>Extracted sessions</td>
<td>336</td>
</tr>
<tr>
<td>Sessions with more than 5 actions(pruned)</td>
<td>3</td>
</tr>
<tr>
<td>Sessions with less than 3 actions(pruned)</td>
<td>321</td>
</tr>
<tr>
<td>Session pruning ratio</td>
<td>64%</td>
</tr>
<tr>
<td>Generated correlation graphs</td>
<td>3</td>
</tr>
<tr>
<td>Graph with more than 3 nodes</td>
<td>1</td>
</tr>
</tbody>
</table>

\( \text{session pruning ratio} = (\text{actions pruned from sessions}) / (\text{total actions of session before pruned}) \times 100\% \)

Extracted actions are assigned with distinct labels generated by a symbol-space component for the convenience of representation and discussions. As shown in Table IV, the actions with a symbol are listed in the left column and alert composing the corresponding action in the right. More detailed information is shown in Table IV as bellow:

<table>
<thead>
<tr>
<th>TABLE IV</th>
<th>IMPORTANT EXTRACTED ACTIONS AND ALERTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action Symbol</td>
<td>Alerts of action</td>
</tr>
<tr>
<td>A</td>
<td>icmp ping</td>
</tr>
<tr>
<td>B</td>
<td>sunrpc portmap getport request</td>
</tr>
<tr>
<td></td>
<td>solaris udp portmap sadmin port query request</td>
</tr>
<tr>
<td></td>
<td>portmap sadmind request udp attempt</td>
</tr>
<tr>
<td>C</td>
<td>sadmind udp ping</td>
</tr>
<tr>
<td>D</td>
<td>sadmind query with root credentials attempt</td>
</tr>
<tr>
<td></td>
<td>solaris udp portmapper sadmin port query</td>
</tr>
<tr>
<td></td>
<td>sadmind netmgt proc service client domain overflow</td>
</tr>
<tr>
<td>E</td>
<td>telnet login incorrect</td>
</tr>
<tr>
<td></td>
<td>telnet login failed</td>
</tr>
<tr>
<td>F</td>
<td>ccproxy telnet ping buffer overflow</td>
</tr>
<tr>
<td>G</td>
<td>rsh root</td>
</tr>
<tr>
<td>H</td>
<td>ddos mstream</td>
</tr>
</tbody>
</table>
The session objects extracted by IACF are represented as a sequence of action symbols with binary correlation characters between them. Each session object is extracted based on the raw alerts triggered by the communications between a pair of hosts which are the extracting source of it. Table V shows a part of the sessions for constructing the correlation graph as follows:

<table>
<thead>
<tr>
<th>Sessions (pruned by SPA)</th>
<th>Extracting sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>A&gt;B&gt;C&gt;B&gt;D&gt;E=B=D&gt;G</td>
<td>202.77.162.213-172.16.112.10</td>
</tr>
<tr>
<td>A&gt;B&gt;C&gt;B&gt;D=E=B&gt;D&gt;H=G</td>
<td>202.77.162.213-172.16.112.50</td>
</tr>
<tr>
<td>A&gt;B&gt;C&gt;B=D=E&gt;B&gt;D&gt;H=G</td>
<td>202.77.162.213-172.16.115.20</td>
</tr>
<tr>
<td>F&gt;E=F</td>
<td>172.16.113.84-172.16.112.50</td>
</tr>
<tr>
<td>A&gt;B</td>
<td>195.115.218.108-172.16.113.50</td>
</tr>
<tr>
<td>202.77.162.213-172.16.112.105</td>
<td></td>
</tr>
<tr>
<td>202.77.162.213-172.16.115.87</td>
<td></td>
</tr>
<tr>
<td>202.77.162.213-172.16.112.100</td>
<td></td>
</tr>
<tr>
<td>202.77.162.213-172.16.113.148</td>
<td></td>
</tr>
</tbody>
</table>

*Binary correlation A>B can be extracted from multiple pairs of hosts.

The largest correlation graph generated based on LLDOS 1.0 data set is shown in Fig. 8. Comparing with the correlation graph in [6], the most difference is that our correlation graph lacks FTP intrusion action node and email intrusion action node. By reviewing IDS alert logs, the FTP-related alerts and email related alerts are not detected by IDS, therefore, they do not appear in the correlation graph. In other aspects, our graph contains fewer nodes and clearer relations because no repeated nodes are allowed in the graph, and a compacted graph provides better performance on searching and counting.

![Correlation graph based on the LLDOS1.0 data set.](image)

The accuracy of the prediction of this framework is also of great interest to us. We mix the LLDOS 1.0 and LLDOS 2.0 as training data (LLDOS 1.0 and LLDOS 2.0 use the same intrusion technology) and use one of them as the test data. After training, the system generates an association graph composed of 9 action nodes. With the increasing of training data, the accumulation of binary relations in the correlation graph will increase, and the accuracy of prediction increased too. The framework calculates the probability of the next step(s) by searching all the branches starting with the input action in the graph. The experimental results show that the prediction accuracy is more than 90%.

**C. EVALUATION ON CICIDS2017**

CICIDS2017 intrusion detection evaluation data set [25] was published by the Canadian Institute for Cybersecurity in 2017 which covers variety of the most up-to-date well-known attacks and it is free to download for studying. More features of the data set are listed below:

1) The data set resembles the true real-world network data (PCAPs), and contains realistic and naturalistic benign background traffic that simulating the everyday interaction behavior of 25 users;
2) The data was continuously captured for five days, from Monday to Friday in July 2017, the size of data captured in each day is around 10GB and is labeled for training;
3) The implemented attacks include Brute Force FTP, Brute Force SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet and DDoS;
4) The dataset was built 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. All the traffic was captured based on the HTTP, HTTPS, FTP, SSH, and email protocols;

To evaluate the IACF framework, we chose an infiltration scenario that occurred between 15:00 and 16:00 on Thursday afternoon according to the CICIDS2017 data set. The data captured in an hour are used to test the framework rather than the whole day data because massive data is not related to the scenario for the test.

The infiltration scenario consists of two steps: 1) The victim download the infected file and open it; 2) The attacker executes the Nmap and portscan on the entire victim-network [26]. There is no more detailed information about the infiltration scenario.

All the preset of system parameters are consistent with the evaluation based on LLDOS data set. Based on the scenario data, snort generated about 3200 raw alerts, and are grouped into 35 sets by source IP and destination IP. After the process of action extraction, 743 intrusion actions of 22 different types are extracted. 35 intrusion sessions are generated, and 24 sessions contain no more than 3 actions after pruning and 11 sessions have more than 4 actions.

As shown in Table VI, 15 types of actions with a symbol to each type are listed in the left column and the alerts
composing the corresponding action in the right. More detailed information is shown in Table VI as below:

<table>
<thead>
<tr>
<th>Action Symbol</th>
<th>Alerts composing the action</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>protocol-snap request tcp</td>
</tr>
<tr>
<td>B</td>
<td>protocol-snap agents/tcp request</td>
</tr>
<tr>
<td>C</td>
<td>echo reply ping</td>
</tr>
<tr>
<td>D</td>
<td>indicator-shellcode x86 inc ehx noop</td>
</tr>
<tr>
<td>E</td>
<td>winnuke attack</td>
</tr>
<tr>
<td>F</td>
<td>tcp packet with urgent flag attempt</td>
</tr>
<tr>
<td>G</td>
<td>realnetworks helix server ntlm authentication heap overflow attempt</td>
</tr>
<tr>
<td>H</td>
<td>destination unreachable port unreachable packet detected server-webapp robots.txt access ftp anonymous login attempt</td>
</tr>
<tr>
<td>I</td>
<td>app-detect failed ftp login attempt</td>
</tr>
<tr>
<td>J</td>
<td>openssh maxstartup threshold connection exhaustion denial of service attempt ssh brute force login attempt</td>
</tr>
<tr>
<td>K</td>
<td>smbv1 protocol detection attempt</td>
</tr>
<tr>
<td>L</td>
<td>openssh maxstartup threshold connection exhaustion denial of service attempt</td>
</tr>
</tbody>
</table>

By observing the action E and F in the Table VI, we can find that E and F are similar except that E has one more alert than F. This is a “containing” relation between the two actions, we can say E contains F or F composes E. We do not know whether they are denoting the same type of attack action but we can improve the extracting policy to merge the two actions or divide the bigger one into smaller one in the future, so as to achieve the more accurate model and prediction.

After the modeling phase, one correlation graph is built by IACF, which is a larger and more complex graph than Fig. 8. More details are shown in Fig. 10:

One of the reasons for the complexity of the graph in Fig. 10 may be the false alerts generated by snort which involves unrelated actions to the graph even though the SPA algorithm can minimize the impact of the false positives.

V. CONCLUSION

In this paper, an efficient alert correlation analysis framework IACF is proposed to detect multistep attack scenarios. Firstly, alerts are grouped according to the source IP address and the destination IP address, and intrusion actions are extracted from each group of alerts based on an important feature of intrusion alerts: intrinsic correlation. A method for matching and aggregating the extracted actions is proposed which relies on the concept of “alert set.” Secondly, IACF can split a long sequence of actions into different sessions by calculating the temporal boundary of each subsequence, and prune sessions with the SPA algorithm to remove the redundant action modes to expose the real correlations between actions. Finally, it divides sessions into binary correlations and adds them to the correlation graphs. This paper also proposes a method for predicting future intrusion actions based on correlation graphs. The experimental data shows that the framework has a better effect on detection and prediction.

In the future, a lot of works should be done in the following aspects:

1) Although IACF is efficiency in discovering intrusion scenarios, the generated graphs are not very intuitive for human experts to recognize the whole intrusion action sequence easily, because sessions are decomposed into binary correlations during graph construction and redundant relations especially the relations can form a cycle in graph are removed, however, IACF is a good model for attack prediction due to the accumulating of binary correlations during the training phase, and the pruning mechanism for improving its self in real-time. Enhancing the correlation graph’s intuitiveness will be a research direction. 2) SPA algorithm can greatly reduce the impact of false positives, but a few single-step attack actions which are a part of intrusion scenario are also filtered out because no binary correlations exist in them. So, improvements to the strategies of correlation and analysis of intrusion actions are needed in the future. 3) Deepen the research on intrusion prediction to improve the accuracy of IACF.

REFERENCES


KAI ZHANG was born in 1986. He received the B.Sc. degree in computer science and technology from Shenyang Ligong University in 2006, and the M.Sc. degree in computer science and technology from Beijing University of Posts and Telecommunications in 2012. He is currently pursuing the Ph.D. degree with the School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing, China. His research interests include network security and machine learning.

FEI ZHAO was born in 1988. He received the B.Sc. degree in electronic information science and technology from Shangqiu University in 2011, and the M.Sc. degree in signal and information processing from Yunnan Minzu University in 2014. He is currently pursuing the Ph.D. degree with the School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing, China. His research interests include network security and machine learning.

SHOUSHAN LUO received the B.Sc. degree in mathematics from Beijing Normal University in 1985, and the M.Sc. degree in applied mathematics and the Ph.D. degree in signal and information processing from the Beijing University of Posts and Telecommunications (BUP) in 1994 and 2001, respectively.

He is currently a Professor with the School of Cyberspace Security, BUP, Beijing, China. His research interests include cryptography and information security.

YANG XIN was born in 1977. He received the B.Sc. degree in signal and information system and the M.Sc. degree in circuits and systems from Shandong University in 1999 and 2002, respectively, and the Ph.D. degree in signal and information processing from the Beijing University of Posts and Telecommunications (BUP) in 2005.

He is currently an Associate Professor with the School of Cyberspace Security, BUP, Beijing, China. His research interests include big data security, cloud computing security, and network security.
HONGLIANG ZHU was born in 1982. He received the Ph.D. degree from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China. He is currently a Vice Director of Beijing Engineering Lab for Cloud Security Technology and Information Security Center of BUPT, where he is also a Lecturer and a Master Supervisor. His current research interests include big data security, cloud computing security, and network security.

He is currently an Associate Professor with the School of Cyberspace Security, BUPT, Beijing, China. His research interests include big data security, cloud computing security, and network security.