IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 30, 2019, accepted December 14, 2019, date of publication December 23, 2019,
date of current version January 3, 2020.

Digital Object Identifier 10.1109/ACCESS.2019.2961517

A Multi-Step Attack Detection Model Based on Alerts of Smart Grid Monitoring System

HUA ZHANG[1], XUEQI JIN[2], YING LI[1], ZHENGWEI JIANG[2], YE LIANG[3], ZHENGPING JIN[1], AND QIAOYAN WEN[1]

[1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]State Gird Zhejiang Electric Power Company, Ltd., Hangzhou 310007, China
[3]Nari Group Corporation, Beijing Kedong Power Control System Company, Ltd., Beijing 100000, China

Corresponding authors: Hua Zhang (zhanghua_288@bupt.edu.cn), Ying Li (liyingemily@bupt.edu.cn), Zhengping Jin (zhpjin@bupt.edu.cn), and Qiaoyan Wen (wqy@bupt.edu.cn)

This work was supported by the Science and Technology Project Funding of State Grid Corporation of China (Research and Application of Network Security Situation Awareness Technology in Power Monitoring System from 2018 to 2019) under Contract 521104180028.

ABSTRACT Smart Grid Monitoring System(SGMS) is an important means to protect the security of smart grid. The high volumes of alerts generated by SGMS often confuse managers. Automatically handling alerts and extracting attack events is a critical issue for smart grid. Most of the existing security event analysis methods are designed for Internet, which will not be directly applicable to the power grid for high reliability and low attack tolerance requirements. In this paper, a multi-step attack detection model based on alerts of SGMS is proposed. In this model, an alert graph is constructed by IP correlation, and then transformed into candidate attack chains after being aggregated. Consequently, the candidate preliminary attack chains are pruned and denoised by negative causal correlation and non-cascading events. Finally, attack chains and visual attack graphs are formed. Our proposal model needs a little of priori knowledge while automatically extracting multi-step attack events and demonstrating the trajectories among IPs. The experimental results show the model performs well on China Grid data and DARPA 2000 data set.

INDEX TERMS Smart grid security, alert correlation, multi-step attack, attack graph.

I. INTRODUCTION

Whis the construction of smart grid and the wide application of big data, cloud computing and Internet of Things technologies, the power grid is facing the vulnerabilities and risks of the Internet [1]–[3], which brings challenges to the original security protection system of power. To approach this challenge, security protection and management systems, such as firewalls, intrusion detection systems(IDS), vertical encryption authentication systems and anti-virus systems, have been studied and applied in power grids [4]–[7]. These systems generate a large number of independent alerts, and the real threat information can easily be flooded with a large number of useless alerts [8]. Therefore, how to handle the alerts and detect major security incidents more accurately has become a major problem in smart grid [9], [10].

With the development of security protection technology, multi-step attacks have become the main form of attack [11], [12]. Multi-step attack detection technology, which is oriented to large-scale network environment, plays an important role in network security monitoring system. It acquires, understands, displays the security factors and future trends in network situation.

In the Smart Grid control center, once an attack occurs, unpredictable serious consequences for the entire grid infrastructure may happen. On December 23, 2015, the Ukrainian power sector was attacked by malicious codes. More than half of the area and part of the Ivano-Frankovsk area were cut off for several hours. Therefore, high reliability attack detection and quick location of attack paths strategies are required for such production environments.

For the smart grid alert logs with the above characteristics, the existing analysis methods face some challenges. Common methods of alert log analysis include similarity-based correlation analysis [13]–[15], attack sequence-based approach [16], [17], and attack sample-based machine learning [8], [18]–[20].

The similarity-based correlation analysis mainly uses the similar characteristics of same kinds of alerts. However, since the logs of the SGMS have been aggregated, the similarity

The associate editor coordinating the review of this manuscript and approving it for publication was Ana Lucila Sandoval Orozco.

```
2018-05-01 06:02:50-2018-05-01 06:02:50 Serial port occupation
2018-05-01 06:02:50-2018-05-01 06:02:50 File/permission changes in key
directories
2018-05-01 06:02:50-2018-05-01 06:02:50 Illegal login, *.*.1.62 multiple use
users A illegal login *.*.1.67
2018-05-01 06:02:51-2018-05-01 06:02:51 Illegal login, *.*.1.62 multiple use
users A illegal login *.*.1.67
2018-05-01 06:02:51-2018-05-01 06:02:51 Illegal login, *.*.1.2 multiple use
users A illegal login *.*.1.47
2018-05-01 06:02:50-2018-05-01 06:02:50 Sql statement has not been submitted
for a long time
```

FIGURE 1. Segments of a class of alerts obtained by similarity-based correlation analysis.

is weakend and the method produce little effect in smart grid alert logs. Fig.1 shows some segments of a class of alerts obtained by similarity-based correlation analysis in reference [15]. These alerts do not belong to the same attack. In addition, aggregation methods cannot describe the attack paths. It just bring the alerts of the same event together.

Attack sequence-based approach associates alert by matching attack rules [21]. It is difficult to cover all attacks [14]. In addition, the SGMS has different and more strict attack detection strategies, which generates different alerts from the internet, like illegal protocol detection and usb insertion detection. It is hard to draw lessons from mature experience.

Attack sample-based machine learning first needs to acquire alert logs or network traffic which are labeled with attack types for learning [22]. Labeling work requires directly obtaining attack logs by constructing virtual scene of simulating attacks or manual verification of attacks. However, because the alert logs in SGMS are already aggregated [4], and manual verification has large workload and difficult to implement in smart grid, the alert logs cannot be simulated.

Contributions: To solve the above problems, this paper proposes a multi-step attack detection model based on the alerts of Smart Grid Monitoring System with only a small amount of expert knowledge. The contributions of this paper are as followings:

1) We extract attack chains according to IP hops, as well as propose a child node aggregation method based on smart grid network alert graph. The attack chains are sequences of suspicious behavior. Because the attack chains are extracted according to IP hops, they also retain the host information of the attack chains. The child node aggregation is carried out before the formation of attack chains, which effectively reduces the redundancy by aggregating child nodes in the alert graph that belong to the same attack.
2) A negative causal correlation method is proposed to eliminate the non-causal alerts. The alerts of SGMS have a variety of granularity and types, and it is difficult to formulate attack rules. However, the method we propose can prune attack chains while minimizing the dependence on expert knowledge.
3) An attack chains denoise method based on non-cascading event is proposed. This method can reduce the redundancy of attack chains by removing the events that are obviously not caused by previous ones.

II. MULTI-STEP ATTACK DETECTION MODEL FRAMEWORK

At first, there are two basic definitions that need to be explained.

Definition 1 (Multi-Step Attack): A well-predefined intrusion process with more than one attack action is called a multi-step attack. The intrusion should include a clear purpose of attack.

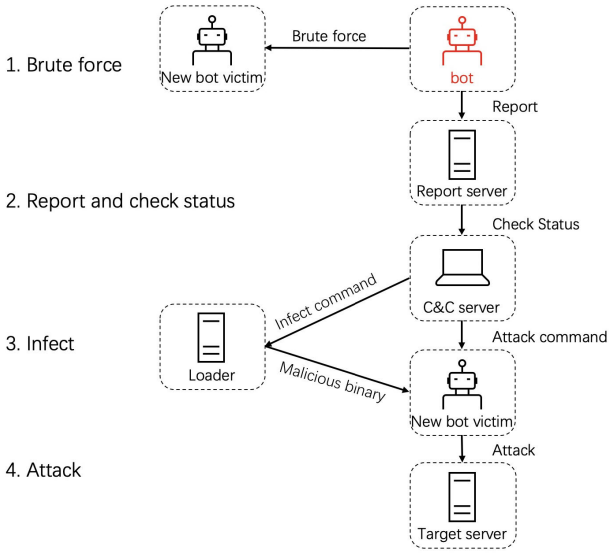

FIGURE 2. Key steps of a DDoS attack using Mirai malware.

Fig.2 is an example of multi-step attack. It shows the key steps of a DDoS attack using Mirai malware [23]. The intrusion process includes

1) default credential discovering through a brute-force attack,
2) shell gaining and forwarding various device characteristics to the report server,
3) new prospective target victims checking,
4) infection command sending to the loader,
5) downloading and executing the corresponding binary version of the malware,
6) instructing all bot instances to commence an attack against a target server.

The purpose of the intruder is using bot instances launching a DDoS attack to the target server. In order to achieve this purpose, the intruder has to prepare well and implement multiple actions.

Definition 2 (Single-Step Attack): Each attack action in multi-step attack is called a single-step attack. A single-step attack may correspond to one alert or multiple duplicated alerts.

A variety of security devices are deployed in the SGMS. The control center aggregates the original alerts into alert log [4]. As shown in Fig. 3, the inputs are the original alerts of the SGMS. After pre-processing, they are formed structureing, noiseless and analyzable alerts. Then they are sent

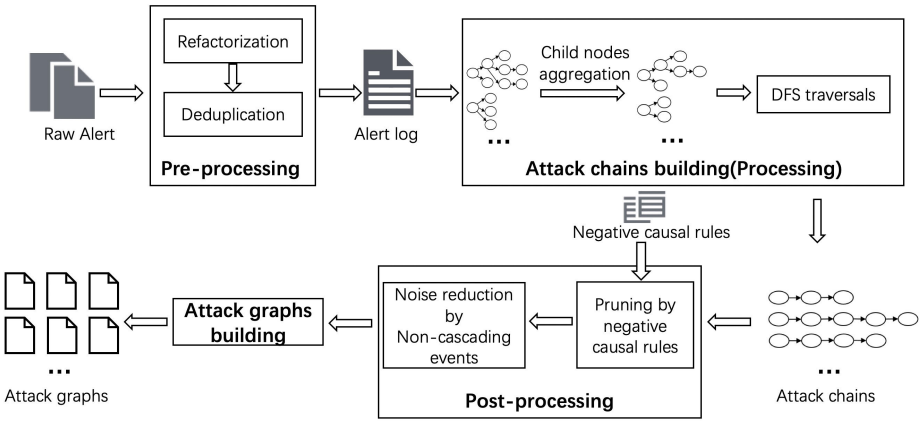

FIGURE 3. Multi-step attack detection model framework.

to the preliminary attack chain building module for aggregating and traversing. After that, preliminary attack chains are formed. Then they will be pruned and denoised by the Post-processing module. Finally, interconnected attack chains are drawn into individual subgraphs.

A. PRE-PROCEEING

The pre-processing module first reconstructs the original alerts, and then performs de-duplication processing to obtain the alert logs. The reconstructed alert contains necessary fields for alert log analysis, which represented as a 10-tuple $Alert(Starttime, Fintime, Content, Type, SrcIP, DstIP, SrcPort, DstPort, Times, Level)$. In a reconstructed alert, *Starttime* represents the start time of the alert event. *Fintime* represents the finish time of the alert event. *Content* represents the alert content. *Type* represents the alert type. *SrcIP* represents the source address. *DstIP* represents the destination address. *SrcPort* represents the source port. *DstPort* represents the destination port. *Times* represents the number of alert repetitions. *Level* represents the alert level.

After reconstruction, the alert logs need to be de-duplicated. A complete attack process involves at least one attack step. For a single-step attack in multi-step attacks, the security device may report multiple redundancy alerts. But not all same alerts belong to the same attack.The deduplication of the alert logs based on the retention of time information can not only reduce the number of alerts, but also retaining the most source data information. The deduplication rule is: if the type, IP and port of an alert are the same as any of the previous two, the latter alert will be removed. The previous alert's *Finishtime* will be updated to the removed one, and *Times* will be accumulated.

B. ATTACK CHAINS BUILDING (PROCESSING)

In the attack chain building module, alert graph is structured by the alert logs. Then the redundant information is removed by aggregation. Finally, the attack chains are obtained after the depth-first traversal. The detailed process is in Section III.

C. POST-PROCESSING

The post-processing module removes the unreasonable attack chains. The negative causal association module prunes the attack chains by negative causal rules, and the non-cascading event noise reduction module denoises the attack chains through the continuity of events. The detailed process is in section IV and section V.

D. ATTACK GRAPH BUILDING

The attack graph building module presents the association between attack chains intuitively. The process is mainly divided into two steps: structureing the attack chains into a directed graph; dividing interconnected parts into independent subgraphs.

III. INITIAL CONSTRUCTION OF IP-BASED ATTACK CHAINS

The construction of the attack chains based on IP is divided into three steps: structureing node information, child node aggregation, and alert graph depth-first traversal.

A. STRUCTUREING NODE INFORMATION

The alert log can be regarded as a network which is connected by source and destination IPs. Nodes of the network are host IPs. Node information structureing process can construct such a network into a graph diagram struct.

The structured multi-step attack is an attack chain. The single-step attacks in an attack chain meet three conditions:

- IP Relevance: The destination IP of a single-step attack is the source IP of the next one.
- Causal Relevance: A single-step attack has a causal relationship with the previous one logically.
- Temporality: The *Starttime* of a single-step attack is later than that of the previous one.

We indicate IP as the IP address of the single-step attack in the attack chain and *Alert* as the alert information (including *Content*, *Starttime*, *Finishtime*, etc.). The attack chain can be represented as $C_1 = [(IP_1, IP_2, Alert_1), (IP_2, IP_3, Alert_2),$

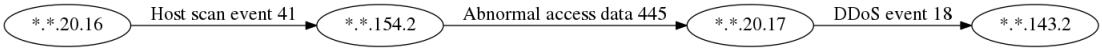

FIGURE 4. An example of an attack chain.

TABLE 1. An instance of attack chain (Log representation).

SrcIP	DstIP	Alert
..20.16	*.*.154.2	2018-5-30:10 2018-5-4:15:47: Host scan events from [*. *. 20.16] to [*. *. 154.2:4808].
..154.2	*.*.20.17	2018-5-4 0:30 2018-5-8 4:32: Abnormal access data from [*. *. 154.2] to [*. *. 20.17:52607].
..20.17	*.*.143.2	2018-5-5 8:51 2018-5-8 17:29: DDoS events from [*. *. 20.17] et al. to [*. *. 143.2:4808].

(IP_3, IP_4, $Alert_3$)]. Fig. 4 shows an example of attack chain, Table 1 lists the corresponding alert log.

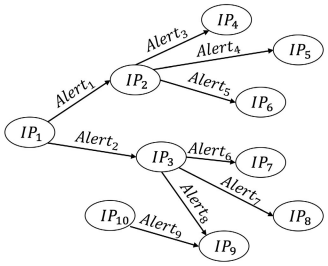

FIGURE 5. An instance of Alert graphs.

TABLE 2. An instance of alert log.

No.	SrcIP	DstIP	Alert Info
1	IP1	IP2	Alert1
2	IP1	IP3	Alert2
3	IP2	IP4	Alert3
4	IP2	IP5	Alert4
5	IP2	IP6	Alert5
6	IP3	IP7	Alert6
7	IP3	IP8	Alert7
8	IP3	IP9	Alert8
9	IP10	IP9	Alert9

In the entire network, IP-based alerts in alert log can be represented as a graph diagram. Table 2 shows an example of alert logs. Fig. 5 shows the alert graph corresponding to Table 2. The graph nodes represent IPs and directed edges represent the alerts from source IPs to destination IPs. The graph diagram of the alerts in the network can facilitate the following child node aggregation and attack chain extraction process.

Each IP node is modeled to a *TNode* class: $TNode = \{SelfIP, ParentsIP, ChildrenInfo\}$. *SelfIP* is the node's own IP, *ParentsIP* is its parents' IP set, and *ChildrenInfo* is its children with their corresponding alert information. Alert graphs are constructed while nodes modeling. Now each IP is represented by a unique node.

B. CHILD NODE AGGREGATION

After structing alert logs, preliminary attack chains can be obtained by depth-first traversal with root nodes. However, there is still a lot of redundancy in alert logs. In order to further aggregate these alerts while retaining the alert information as much as possible, we proposed the Child Node Aggregation Algorithm as Algorithm 1.

Algorithm 1 Child Node Aggregation Algorithm

Require: N: All TNode in alert logs; t: Aggregation interval

```
function CHILD NODE AGGREGATION(N, t)
    Let '==' be a symbol, where alert_A == alert_B means A's Type, Srcip, Dstip is equal to B's
    for each node ∈ N do
        Children[1...k] ← node.ChildrenInfo
        lastChild ← Children[1]

        //First time aggregation
        for child ∈ Children[2...k] do
            if lastChild == child then
                lastChild.Finishtime ← child.Finishtime
                delete child
            else
                lastChild ← child
            end if
        end for

        //Second time aggregation
        Let i = 0
        for child ∈ Children[2...k] do
            i ← i + 1
            for lastChild' ∈ Children[1...i] do
                if child == lastChild' and (child.end − lastChild'.start < t) then
                    lastChild.Finishtime ← child.Finishtime
                    delete child
                    break for
                end if
            end for
        end for
    end for
end function
```

The algorithm traverses the set N, and aggregates the child nodes. This algorithm contains twice aggregations:

TABLE 3. An instance of alert log.

No.	Alert duration	Alert event	relationship
1	2018-05-04 20:58:31-2018-05-09 11:10:08 2018-05-05 00:00:29-2018-05-05 23:44:05	DDoS event: DDoS event from [IP_1] et al. to [IP_2: 7998]. Illegal access: From [IP_2] to [IP_3: 36074].	Negative causal relationship
2	2018-05-03 00:00:47-2018-05-08 14:42:30 2018-05-03 01:52:10-2018-05-03 01:52:10	Abnormal access data: From [IP_4] to [IP_5: 9053]. Illegal access: From [IP_5: 54917] to [IP_6: 5000].	No Negative causal relationship

In the first aggregation, if same events from host A to B have been continuously happening, and no other events triggered by A occured during this period, then the continues events are considered to be the same attack. The algorithm first traverses each node's child node set. Once the alert type, source IP, and destination IP of one child node and the previous child node are the same, update the *Finishtime* of the previous one to the latter one and accumulate the *Times*. Then delete the latter event.

The second aggregation is based on a premise that the same events that occur in a small time window belong to the same attack. The algorithm first traverses each node's child nodes set. Once the alert type, source IP, and destination IP of the two child nodes are the same, while the time range between the two alerts is not bigger than the setting time window, update the *Finishtime* of the previous event to the latter one and accumulate the *Times*. Then delete the latter event.

C. GETTING THE ATTACK CHAIN

After the node modeling and the child node aggregation, we propose the Get Chains Algorithm as algorithm 2. This algorithm is for deeply traversing the alert graphs and obtain attack chains. The head nodes of the chains include two cases.

Algorithm 2 Get Chains Algorithm

Require: N: All TNode in alert logs
Ensure: C: A set of chains

```
function GET CHAINS(N)
    for each node ∈ N do
        if node has no parent then
            C' ← DFS(node)
            C ← C ∪ C'
        end if
    end for
    N' ← CheckNoVisited(N)
    for each node ∈ N' do
        C' ← DFS(node)
        C' ← C ∪ C'
    end for
    return C
end function
```

- The first one is that the root node in the attack graph.
- The second one is that the node has parents, but the *Starttime* of its parent events are later than itself.

The two cases both means the nodes have no former events. The function CheckNoVisited() is corresponding to the second case.

The function *DFS*() represents the depth-first traversal algorithm, and the attack chain set C' obtained by each depth-first traversal is added to the final attack chain set C.

IV. PRUNING BY NEGATIVE CAUSAL RULES

The idea of causal correlation is widely used in the field of alert correlation analysis [16]–[18], [21], [24], [31], [33]. Due to the lack of attack rules in SGMS and the difficulty of network vulnerability scanning, these methods are difficult to use directly. Negative causal correlation is designed in this paper. Using the negative causal correlation method to prune the attack chains not only ensures the causal correlation of the attack event, but also minimizes the dependence of expert knowledge, attack tagging and simulation works.

A. NEGATIVE CAUSAL RULES

In each multi-step attack, the adjacent attack steps are causally related. In order to satisfy the causal relationship in the attack chain, a method based on negative causal correlation is proposed.

Definition 3 (Negative Causal Relationship): If two alerts cannot meet the causal relationship, they meet negative causal relationship.

Definition 4 (Negative Causal Rules): The negative causal rules are represented as a matrix that defines the negative causal relationship between each two alert types. In the matrix, 1 means there is a negative causal relationship between the corresponding two alert types, and 0 means no negative causal relationship. For example, In the alert set $\{Alert_1, Alert_2, \cdots, Alert_n\}$, $(Alert_i, Alert_j)$ in the negative causal rules means the relationship between $Alert_i$ and $Alert_j$. If $(Alert_i, Alert_j)$ is 1, they meet negative causal relationship.

Table 3 shows two examples of attack chains extracted from the alert logs of a SGMS. The events in the attack chains 1 cannot constitute a causal relationship, that is, they meet negative causal relationship. The events in the attack chain 2 don't meet negative causal relationship. In attack chain 1, the first and the second events are "DDoS event" and "Illegal access". Usually, a complete DDoS attack may contain 4 preparations and final flood attacks. The denial of service attack is the last step, with the purpose to make the target machine unresponsive. Therefore, the DDoS attack and "Illegal access" cannot constitute a causal relationship. Unlike the attack chains 1, the events of the attack chain 2 "Abnormal access data" and "Illegal access" may constitute a causal relationship. The alert that "Abnormal access data" can be a virus script. After being infected by the script,

TABLE 4. Alert log examples and non-cascading event description.

Alert No.	Alert duration	Alert event	Relationship
1	2018-03-28 09:47:40-2018-03-28 23:56:34	IP_1 memory usage exceeds the threshold	
2	2018-03-28 20:11:37-2018-03-28 23:56:36	IP_1 memory usage exceeds the threshold	
3	2018-03-29 00:16:37-2018-03-29 23:52:27	Abnormal access data: From [IP_4] to [IP_5: 9053].	Alert 3 and alert 4 is form an attack chain, but belong to non-cascading events
4	2018-03-29 15:52:37-2018-03-29 23:52:26	IP_1 memory usage exceeds the threshold	
5	2018-03-30 00:12:21-2018-03-30 16:57:23	IP_1 memory usage exceeds the threshold	

the host IP becomes a stepping stones and continues to launch further illegal access to IP_6.

B. PRUNING

In an attack chain, if the two adjacent events have negative causal relationship, such as DDoS event and illegal access in Table 3, it is impossible to constitute an attack. To eliminate this connection, pruning by negative causal correlation is needed. As shown in Fig. 6, when $Alert_2$ and $Alert_3$ meet the negative causal rules, the connection between them should be cut off. As a result, this attack chain is broken into two subchains C_1 and C_2, and each part of them meet the negative causal rules.

$$\begin{aligned} C &= [(IP_1, IP_2, Alert_1), (IP_2, IP_3, Alert_2) \\ &\quad IP_3, IP_4, Alert_3), (IP_4, IP_5, Alert_4)] \\ C_1 &= [(IP_1, IP_2, Alert_1), (IP_2, IP_3, Alert_2)] \\ C_2 &= [(IP_3, IP_4, Alert_3), (IP_4, IP_5, Alert_4)]] \end{aligned}$$

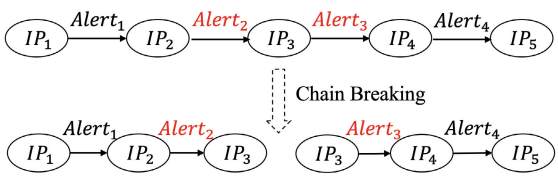

FIGURE 6. Chain breaking.

V. DENOISE BY NON-CASCADING EVENTS

After IP correlation and pruning by negative causal rules, the attack chains are built by meeting three conditions: IP relevance, causal relevance, and temporality. There are still some unreasonable chains. Table 4 gives out some alert log examples. After a series of processing, we obtained an attack chain consisting of alert 3 and alert 4. In this attack chain, the previous event is considered to be the cause of the latter one. But after backtracking the relevant alert records of IP_1, it is found that IP_1 has been reporting the alert "memory usage threshold exceeded" for a period of time before receiving the abnormal data from IP_2. Therefore, it can be inferred that the "memory usage exceeds the threshold" is not directly caused by "sending abnormal data" on a large probability. The above situation is called non-cascading event. Cascading event is defined as following:

Definition 5 (Cascading Event): When the events A, B satisfy the following two conditions, then B is said to be a cascading event of A.

- Temporality. B occurs after A
- B does not occur for a period of time t_n before A occurred.

Non-cascading events should be removed from the attack chains.

Algorithm 3 Noise Reduction Algorithm

Require: C: A set of attack chains
Ensure: G: A set of attack chains without non-cascading events

```
function NOISE REDUCTION(C, t_n)
    Let Q be a queue initially empty
    for each chain[1...k] ∈ C do
        i ← 1
        last ← chain[1]
        t_2 ← chain[1].Starttime
        t_1 ← t_2 − t_n
        for each item ∈ chain[2...k] do
            Alert ← (Type, SrcIP, DstIP, SrcPort, DstPort)
            if FindAlert(t1, t2, Alert)isTrue then
                G ← G ∪ chain[1...i] ∪ NoiseReduction(chain[i + 1...k])
                return G
            else
                i ← i + 1
                t_2 ← item.Starttime
                t_1 ← t_2 − t_n
            end if
            G ← G ∪ chain[1...k]
        end for
    end for
    return G
end function
```

The algorithm for removing non-cascading events is Noise Reduction algorithm. For each attack chain, traverse its two adjacent events. Denote the former one as e_i and the latter one as e_j. The *Starttime* of e_i is recorded as t_2. From t_2 rollback to t_1 for a period of time t_n. e_i and e_j are non-cascading events if there is e_j happened between t_1 and t_2. The subchain between non-cascading events need to be broken. The breaking operation is in section IV.B.

After noise reduction, the final attack chain is formed.

VI. MODEL EVALUATION ON SGMS DATA

In this paper, 2 months alert logs of the SGMS of an area of China Grid are analyzed. We first developed a

TABLE 5. Negative causal rules.

State	Database state abnormality	Database storage abnormality	DoS	Abnormal access data	Memory exceptions	System exceptions	Hardware exceptions	Hardware plug in/out
Database state abnormality	0	0	1	0	1	1	1	1
Database storage abnormality	0	0	1	1	1	1	1	1
DoS	1	1	1	1	0	1	0	1
Abnormal access data	0	0	0	0	0	0	0	1
Memory exceptions	0	0	0	0	0	0	0	1
System exceptions	0	0	0	0	0	0	0	1
Hardware exceptions	0	0	0	0	0	0	0	1
Hardware plug in/out	0	0	0	0	0	0	0	0

negative causal rules according to the characteristics of these data. Then, we analyzed the data of the two time windows and illustrated some of the suspicious attacks. In the global analysis, we describe the processing data of deduplication, aggregation, pruning and denoise, and the formation of the attack graph. Finally, the efficiency analysis is carried out. The balance between resource consumption and the number of attack chains generated can be achieved when $t_w = 3$.

A. NEGATIVE CAUSAL RULES SETTING

We got 73 kinds of alerts from 7,915,376 alert logs. According to the characteristics of alert types, the 73 kinds of alerts were divided into 8 categories: database status abnormality, database storage abnormality, DoS, abnormal access data, memory exceptions, system exceptions, hardware exceptions and hardware plug in/out. Negative causal rules can be learned based on expert experience, as shown in Table 5.

1) DATABASE STATE ABNORMALITY

It represents an abnormal state in the database. This abnormal state may be caused by an ongoing or completed attack. It may generate local exception or subsequently launch attacks after getting information from database.

2) DATABASE STORAGE ABNORMALITY

It indicates that the database storage space is insufficient. It is usually caused by operational non-compliance. It cannot be used as an intermediate step in a multi-step attack. Therefore, it cannot cause other attacks.

It should be noted that the above two classes of alerts are reported by user host rather than database server. It is important to fully understand how alerts are generated for making negative causal rules.

3) DoS

It indicates that the host may be subjected to a denial of service attack. The attack's goal is to cause unavailability to the target. Therefore, we believe the target host will not launch subsequent proliferation attacks.

4) ABNORMAL ACCESS DATA

It indicates that the destination host receives abnormal data. In fact, the abnormal data may be any kind of attack. Therefore, it may cause any kind of attack.

5) MEMORY EXCEPTIONS

It represents the memory abnormal state of the host. This abnormal state may be caused by an ongoing or completed attack. Therefore, it may subsequently launch attacks or generate local exception.

6) SYSTEM EXCEPTIONS

It represents the abnormal state of the system, which may be caused by users or attackers. Any subsequent attacks except hardware plug in/out may occur.

7) HARDWARE EXCEPTIONS

It represents hardware abnormal status, so any subsequent attacks except hardware plug in/out may occur.

8) HARDWARE PLUG IN/OUT

It represents there is a device plugging in or out, which may carry viruses. It may cause any attack.

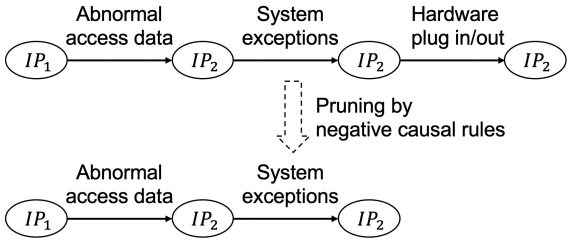

FIGURE 7. An instance of pruning by negative causal rules.

Fig. 7 shows an instance of pruning by negative causal rules. In this candidate attack chain, IP_1 first sent abnormal access data to IP_2. Then some system exceptions occurred on IP_2. Finally, a USB plugs in IP_2. In this chain, there are two alert pairs need to be judged. The first is "abnormal

TABLE 6. Data deduplication and aggregation effect of Experiment 1.

Time range	Data	Alert	Num. of branches	Prop. of original
	original	101101	87721	100%
2018-5-3 2018-5-5	After deduplication	82041		81.15%
	After aggregation		4800	5.47%

TABLE 7. Attack chains processing effect of Experiment 1.

Data	Chains num.	Prop. of original	Graph num.	IP num.	Avg. IP num.	Max. IP num.	Min. IP num.
Initial chains	11237	100%					
Pruning	10027	89.23%					
Denoise	739	6.58%					
Generate graphs			24	137	5.71	34	1

access data'' and ''system exceptions'', according to Table 5, the corresponding number is 0. The second is ''system exceptions'' and ''USB plugged in''. The corresponding number is 0, according to the definition 4, the second alert pair match negative causal relationship and should be cut off.

B. SUSPICIOUS DATA ANALYSIS EXAMPLE

We take two time windows as examples. Window 1 is from May 3th, 2018 to May 5th, 2018. Window 2 is from June 10th, 2018 to June 12th, 2018. The time window t_w is 3 days, and the non-cascading event time interval t_n is set to 2 hours.

Experiment 1: There are 101,101 alert logs in window 1. After reconstructing and deduplicating, we obtained 82,041 alert logs. Then node modeling and child node aggregation were performed. The data deduplication and aggregation effect is shown in Table 6. After deduplication, the data was reduced to 81.15% of the original data. After aggregation, the branches was reduced to 5.47% of the original branches.

As shown in Table 7, after the pruning by negative causal correlation and the noise reduction by non-cascading events, 739 attack chains were finally obtained, which accounted for 6.58% of the original data. Graphviz tool were used to draw the attack graphs.

Attack process can be inferred according to attack graphs. Followings are three instances.

1) **Suspected to be the victim hosts sending back data**
Fig. 8 is centered on *.*.21.3 and *.*.21.5. These two hosts had unauthorized access to other hosts for multiple times. The latter hosts also sent back abnormal data for multiple times. Based on these facts, we infer hosts *.*.21.3 and *.*.21.5 suspicious for attack attempts. The abnormal access data generated by *.*.29.4 and *.*.20.129 may be the sensitive data that needed to be send back after successful invation. In addition, hosts *.*.27.170, *.*.27.33, *.*.25.1, *.*.76.2 also suffered unauthorized access, but did not send abnormal data back. The attack may fail.
2) **Suspected to be the intruder controlling multiple hosts and then launch a DDoS attack**
DDoS attacks are generated by flooding a system from several machines [25]. In some DDoS attacks, the intruders need to detect vulnerabilities and spread malicious applications to the hosts first for making them zombie hosts. In Fig. 9, there were scanning or abnormal data transmission events between *.*.13.31, *.*.20.16, *.*.154.2, and *.*.20.17. These events may be the detection, intrusion and malicious program deployment process before the DDoS attack. Host *.*.21.16 and *.*.20.17 are likely to be the zombie hosts for launching DDoS attack to *.*.159.2, *.*.143.2.
It should be noted that the DDoS events in alerts have been aggregated. Its full alert content is ''DDoS event: DDoS event from [IP_1] et al. to [IP_2]''. As a result, the DDoS event detection of our method is not aimed at flooding data, but the complete process of well-planned attacks including preparation and invasion phase.
3) **Suspected to be the host being compromised, and launching large number of attack attempts**
Fig. 10 shows an attack graph consisting of 18 hosts. In this graph, host *.*.17.1 was first subjected to a large number of scan events and abnormal access data from hosts *.*.105.133, *.*.137.168, *.*.105.134. After that, multiple attacks, including scan events, sending abnormal access data, and DDoS events, were launched to other hosts by *.*.17.1. It can be inferred that *.*.17.1 may be compromised and then launched a wide range of scan and attack attempts.

Experiment 2: There are 631,862 original alert logs in window 2. After reconstruction and deduplication of the alerts, node modeling and child node aggregation were performed. Table 8 shows the result. After deduplication, the alerts was reduced to 66.68% of the original alerts. After aggregation, the branches were reduced to 1.17% of the original.

After the pruning by negative causal correlation and noise reduction by non-cascading events, as shown in Table 9, 165 attack chains were finally obtained, accounting for 2.57% of the original. Followings are three instances of Experiment 2:

1) **Suspected to be the intruder trying to raise local privileges**
Fig. 11 is centered on the host *.*.1.3, which is subjected to a large number of unauthorized accesses

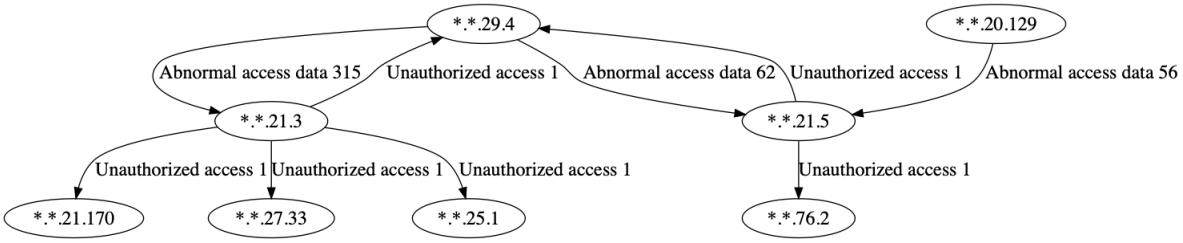

FIGURE 8. Suspected to be hosts *.*.21.3 and *.*.21.5 sending back data in Experiment 1.

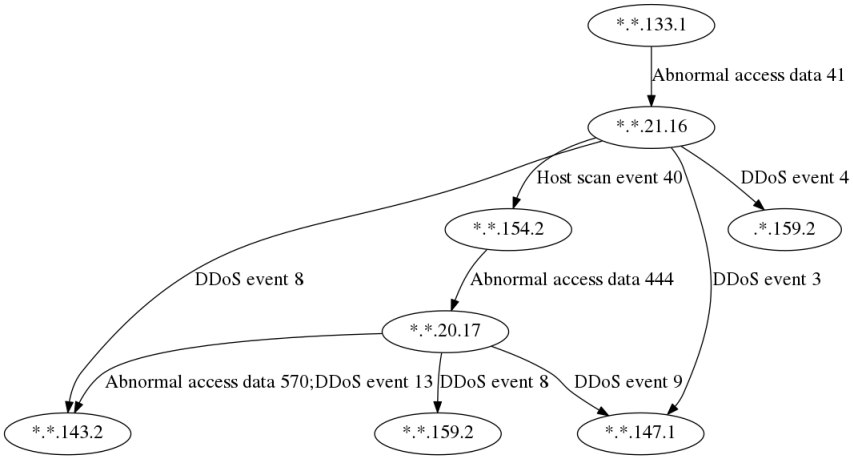

FIGURE 9. Suspected to be the intruder controlling zombie hosts *.*.21.16 and *.*.20.17 and launch a DDoS attack in Experiment 1.

TABLE 8. Data deduplication and aggregation effect of Experiment 2.

Time range	Data	Alert	Num. of branches	Prop. of original
	original	631862	447509	100%
2018-5-3 2018-5-5	After deduplication	421318		66.68%
	After aggregation		240	1.17%

TABLE 9. Attack chains processing effect of Experiment 2.

Data	Chains num.	Prop. of original	Graph num.	IP num.	Avg. IP num.	Max. IP num.	Min. IP num.
Initial chains	6419	100%					
Pruning	5843	91.03%					
Denoise	165	2.57%					
Generate graphs			36	142	3.94	25	1

from *.*.1.2, and several illegal logins from 127.0.0.1. There are also alerts about host *.*.1.3 itself, which are opening illegal ports and attempting to access the root user through SSH. Based on this, we can infer that *.*.1.2 may be the source of the attack. The large number of unauthorized access to *.*.1.3 may be attack attempts. An illegal port of IP 127.0.0.1 may be utilized to hijack a service on the host *.*.1.3 by the intruder. Then the intruder may login the local user through this service. Besides, it is suspicious that host *.*.1.3 tried to login itself through SSH. This may be the intruder's action after getting the root privilege.

2) **Suspected to be a continuous intrusion**
Fig. 12 can be roughly divided into two lines. One is that host *.*.1.30 invaded host *.*.1.32, and then host *.*.1.32 invaded host *.*.1.2. The other is host *.*.1.24 invaded host *.*.1.2. The targets of the two intrusion lines are both host *.*.1.2. Therefore, the host *.*.1.2 can be inferred to be the target of the attack.

Experiment 1 and Experiment 2 successfully extract the attack events. It indicates that the negative causal correlation can ensure the causal relevance of the attack chains and our model is valid.

C. GLOBAL ANALYSIS

This section conducts experiments on the total of 7,915,376 alert logs from a certain area of the China grid from April 16th, 2018 to June 19th, 2018. The time window t_w is 3 days, and the non-cascading event time interval t_n is 2 hours, the time window slides forward every 1 day.

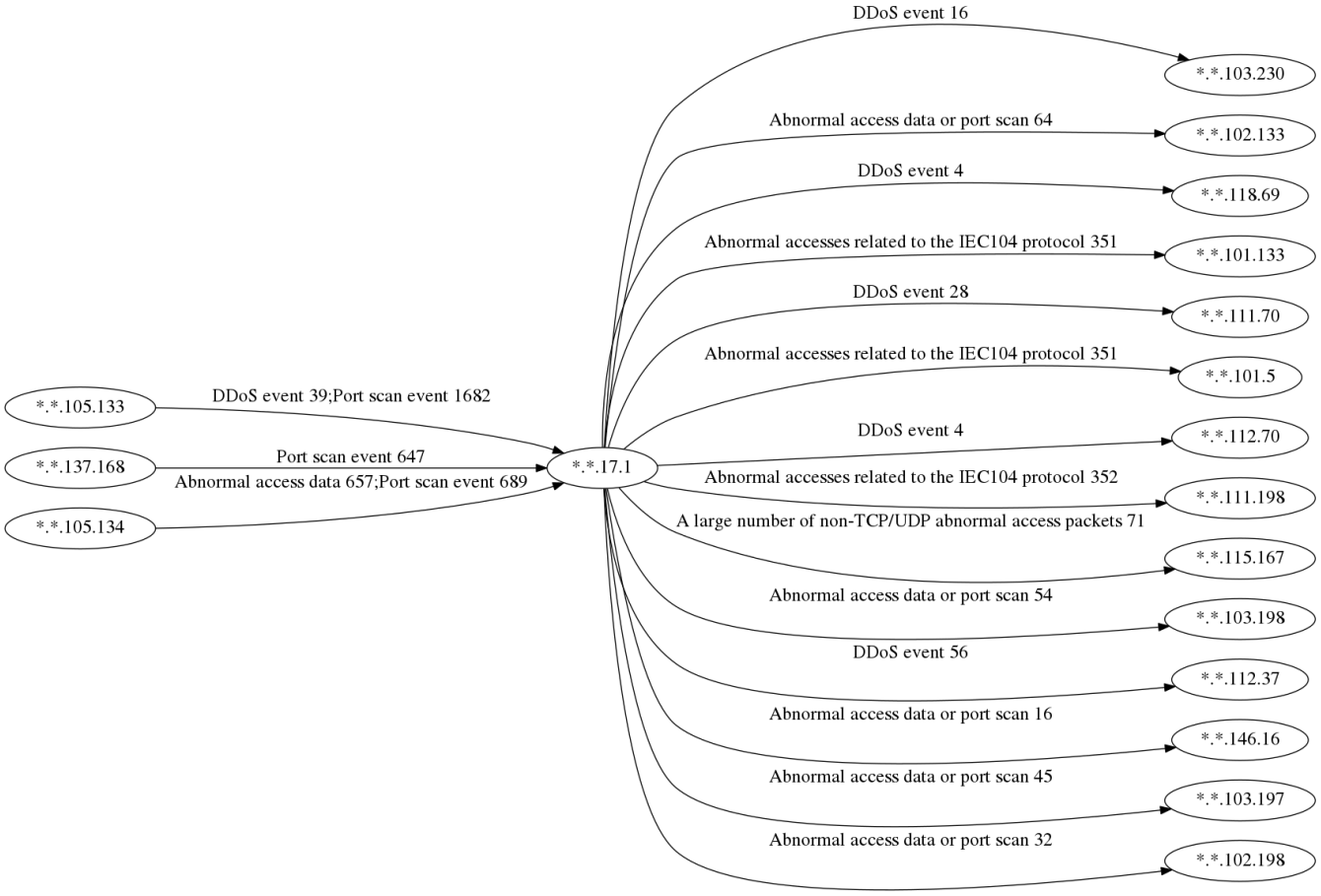

FIGURE 10. Suspected to be *.*.17.1 is compromised after a large number of attack attempts in Experiment 1.

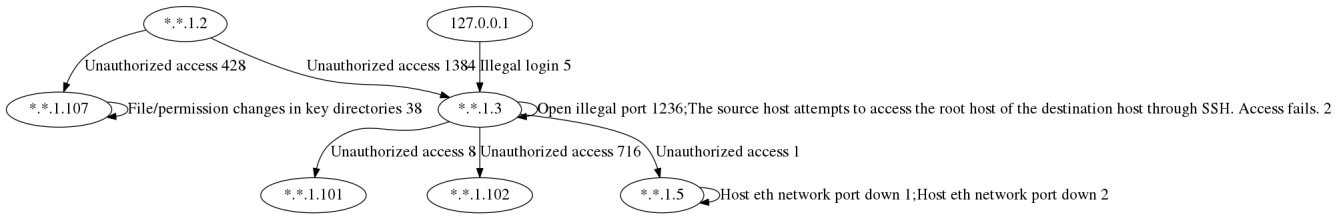

FIGURE 11. Suspected to be the intruder try to raise local privileges after the successful invasion in Experiment 2.

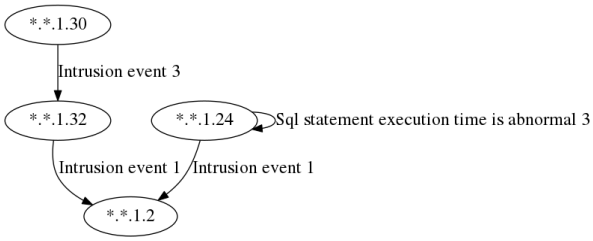

FIGURE 12. Suspected to be a continuous intrusion in Experiment 2.

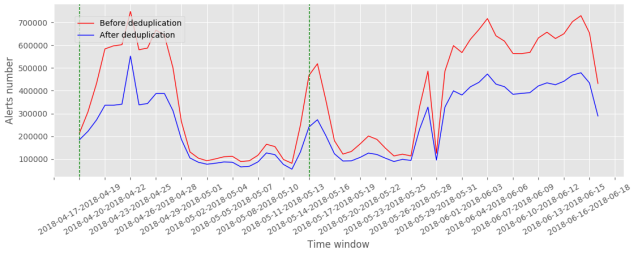

FIGURE 13. Comparison of data volume before and after alert deduplication in a region of China Grid.

As shown in Fig. 13, data preprocessing can effectively remove redundancy. The average deduplication rate is 31.48%, the highest rate is 48.44% and the lowest rate is 13.75%. The denoise effect is better when the amount of data is larger. As shown in Fig. 14, the child node aggregation method can effectively reduce the number of branches of the alert graph. The number of branches is reduced by an average of 90.25%, up to 99.20%, and at least 97.85%.

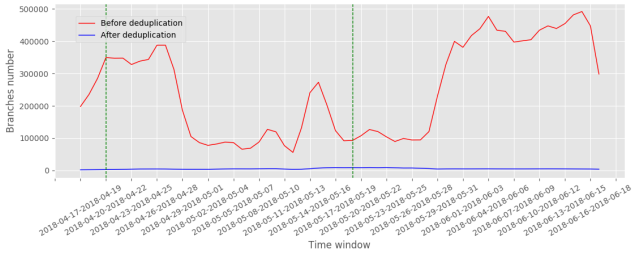

FIGURE 14. Comparison of number of branches before and after child node aggregation in a region of China Grid.

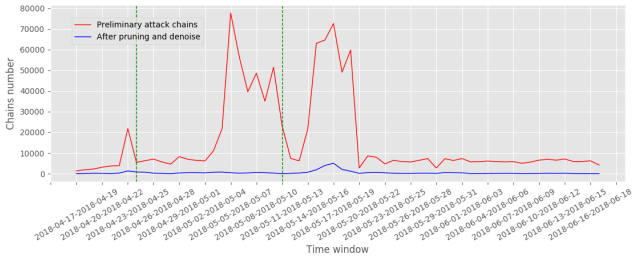

FIGURE 15. Comparison of the number of attack chains before and after the pruning and denoise in a region of China Grid.

As shown in Fig. 15, pruning and noise reduction can greatly reduce the unreasonable noise in the attack chains, and the number of attack chains is basically controlled to 10^2 magnitude. The number of the unreasonable chains is reduced by an average of 95.68%, up to 99.71% and at least 85.12%.

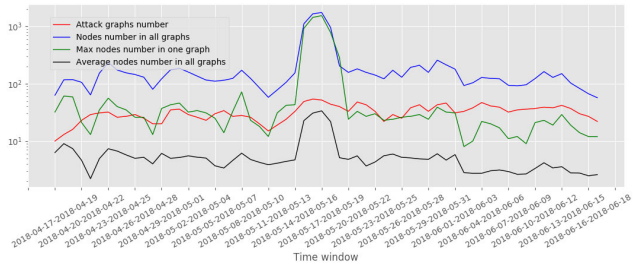

FIGURE 16. Attack graph related data in a region of China Grid, including the number of attack graphs, the total number of nodes in all graphs, the maximum number of nodes in a single graph, and the average number of nodes per graph.

Fig. 16 describes the number of attack graphs, the total number of nodes in all graphs, the maximum number of nodes in a single graph, and the average number of nodes per graph. The data of the attack graphs is relatively stable. The average number of attack graphs is 32.02, with a maximum of 54 and a minimum of 10, The average number of nodes per graph is 6.68. Except for the "unauthorized access" alert which broke out around May 15th, the size of the other attack graphs is reasonable, which is convenient for network administrator to analyze. When the attack broke out around May 15th, the attack graphs could become too large to analyze. In the face of this problem, the information of the attack chains can be used as an auxiliary analysis means to quickly locate the abnormal point.

D. EFFICIENCY ANALYSIS

In this section, we will compare the results and efficiency of the attack chain extraction under different time windows t_w and get a suitble window size. The parameter t_w is set to 1 day, 2 days, 3 days and 4 days, while the time range is 1 week, 2 weeks, and 3 weeks. We recorded the data of the attack chain initial construction, pruning by negative causal correlation, noise reduction by non-cascading events, the resource consumption, CPU usage, and memory usage. The resource consumption is evaluated in terms of program time consumption.

Fig. 17(a) shows that in the initial construction of the attack chain, the program running time increases with t_w and time range. At the same time, Fig. 17(b) shows that the number of attack chains also increases. Fig. 18(a) shows that in the pruning by negative causal correlation phase, the time consumption increases with t_w and time range. At the same time, Fig. 18(b) that the number of attack chains generated also increases.

Fig. 19(a) shows that in the noise reduction by non-cascading event phase, the time consumption increases with t_w and time range. At the same time, Fig. 19(b) shows that the number of attack chains generated also increases.

The time consumption of Fig. 17(a)-19(a) shows that as time window t_w increase from 1 day to 4 days, the time consumption also increases. When t_w increases from 3 days to 4 days, the total time consumption is increased by 5535.64 seconds. When t_w is increased from 2 days to 3 days, the total time consumption only increases by 995.96 seconds. When t_w increases from 1 day to 2 days, the total time consumption increases by 1375.91. second.

Fig. 17(b)-19(b) shows that as the time window t_w increases from 1 day to 4 days, the number of attack chains increases. Fig. 18(b) shows the number of final attack chains. The number doesn't increase as the time consumption. When t_w changes from 1 day to 2 days, the number of attack chains increases by 2297. When t_w changes from 2 days to 3 days, it increases by 1851. When t_w changed from 3 days to 4 days, it increases by 579.

Fig. 20 and Fig. 21 show the CPU consumption percentage and memory consumption of the program running under different windows. With the increasing of t_w, the CPU consumption is basically unchanged, and the memory usage increase accordingly. The maximum memory consumption is 3.05GB when $t_w = 4$.

Considering the number of attack chains generated and the consumption of resources by different windows t_w, a good balance is achieved when the value of t_w is 3.

VII. MODEL EVALUATION ON DARPA 2000 DATA SET

To verify the validity of the model, we did another experiment with DARPA 2000, a complete DDoS attack dataset [26].

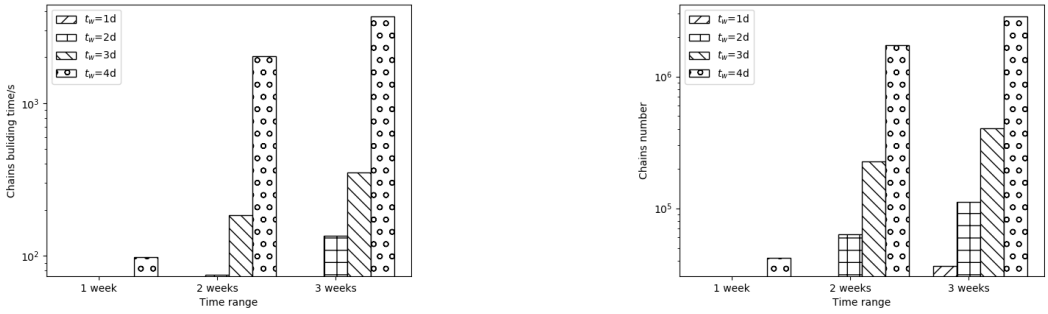

FIGURE 17. (a) Represents the time consumption of chains building under 4 windows sizes. (b) represents the number of preliminary chains under 4 windows.

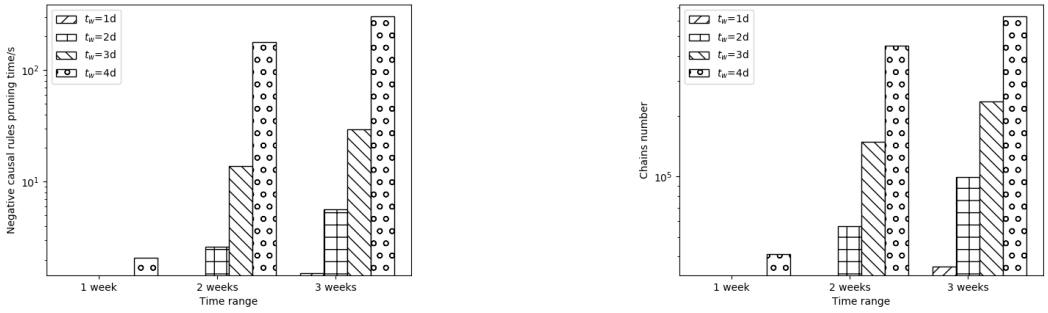

FIGURE 18. (a) Represents the time consumption of pruning under 4 windows. (b) represents the number of chains after pruning under 4 windows.

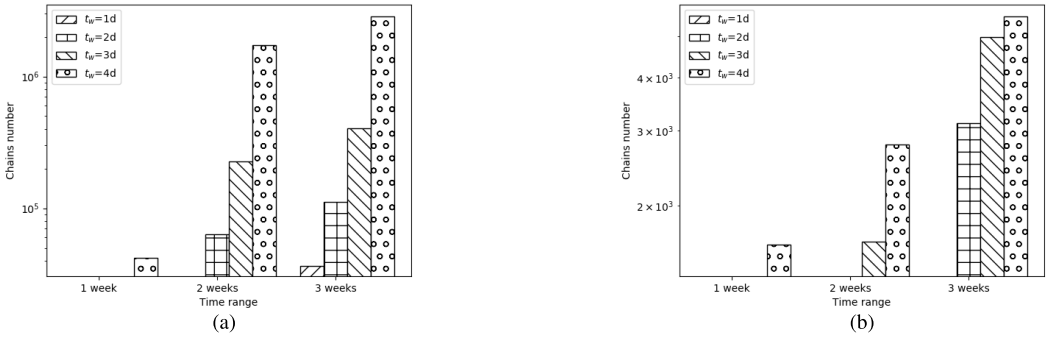

FIGURE 19. (a) Represents the time consumption of denoise under 4 windows. (b) represents the number of chains after denoise under 4 windows.

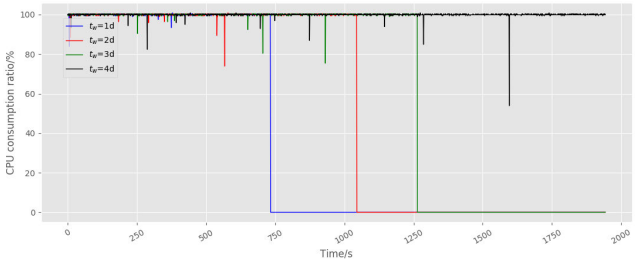

FIGURE 20. CPU consumption percentage under 4 windows.

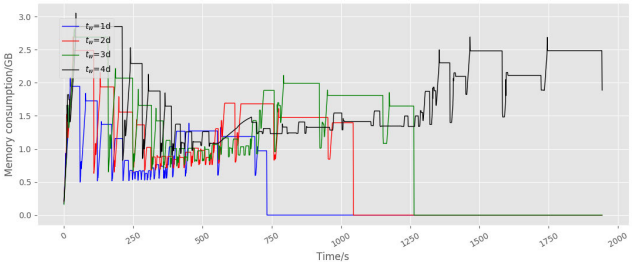

FIGURE 21. Memory consumption under 4 windows.

Result shows the model can successfully detect the attacks and draw the attack graphs.

A. DARPA 2000 LLS_DDOS1.0 DATA SET

LLS_DDOS 1.0 in the DARPA 2000 intrusion scenario data set of MIT Lincoln Laboratory contains a complete real-time distributed denial-of-service attack scenario. The scenario includes four steps as followings:

1) Probing the network to find the hosts who had the Solaris sadmind vulnerabilities.

TABLE 10. Negative causal rules.

State	WEB access response (succeed)	WEB access response (failed)	Login (succeed)	Login (failed)	WEB invalid access	SNMP invalid access	RPC sadmind invalid access	TELNET/RSH access	DoS	MS-Sql overflow attempt	Attack detection	Other abnormal package
WEB access response (succeed)	1	1	0	0	0	0	0	0	0	0	0	0
WEB access response (failed)	1	1	1	1	1	1	1	1	1	1	0	1
Login (succeed)	1	1	0	0	0	0	0	0	0	0	0	0
Login (failed)	1	1	1	1	1	1	1	1	1	1	1	0
WEB invalid access	0	0	0	0	0	0	0	0	0	0	0	0
SNMP invalid access	1	1	0	0	0	0	1	0	1	0	0	0
RPC sadmind invalid access	1	1	0	0	0	0	0	0	0	0	0	0
TELNET/RSH access	0	0	0	0	0	0	0	0	0	0	0	0
DoS	1	1	1	1	1	1	1	1	1	1	1	0
MS-Sql overflow attempt	1	1	0	0	0	0	1	0	0	0	0	0
Attack detection	1	1	0	0	0	1	0	0	0	0	0	0
Other abnormal package	1	1	0	0	0	0	0	0	1	0	0	0

TABLE 11. Negative causal rules of LLDOS1.0.

Categories	Alerts
WEB access response (succeed)	ATTACK-RESPONSES directory listing
WEB access response (failed)	ATTACK-RESPONSES Invalid URL, ATTACK-RESPONSES 403 Forbidden
Login (succeed)	INFO TELNET access
Login (failed)	INFO TELNET Bad Login, INFO TELNET login incorrect, INFO TELNET login failed, INFO FTP Bad login
WEB access	WEB-MISC RBS ISP /newuser access, WEB-MISC /doc/ access, WEB-CGI finger access, WEB-CGI campus access, WEB-CGI db2www access, WEB-CGI wrap access, WEB-IIS %2E-asp access, WEB-IIS register.asp access, WEB-IIS fpcount access, WEB-FRONTPAGE /_vti_bin/ access
SNMP access	SNMP request udp, SNMP public access udp, SNMP AgentX/tcp request
RPC sadmind access	RPC portmap sadmind request UDP, RPC sadmind query with root credentials attempt UDP, RPC sadmind UDP NETMGT_PROC_SERVICE CLIENT_DOMAIN overflow attempt
TELNET/RSH access	RSERVICES rsh root, INFO TELNET access
DoS	DoS
MS-Sql overflow attempt	MS-Sql version overflow attempt
Detection	ICMP Echo Reply, RPC sadmind UDP PING,
Other abnormal package	BAD-TRAFFIC tcp port 0 traffic, NETBIOS NT NULL session

2) Finding valid hosts and using Daemon Buffer Overflow vulnerability in the Solaris operating system to install malicious programs into the three hosts.
3) Using Sadmind Exploit to get root privileges of the hosts.
4) Launching a DDoS attack against the final target.

Data span is 69 minutes and 44 seconds, which is less than 3 days. So t_w is not set. The non-cascading event time window t_n is set to 10 minutes.

B. NEGATIVE CAUSAL RULES SETTING

Before prepocessing, we used snort as our IDS to transform network traffic to alerts. Since the attacker used the random IP address to attack the target host, MAC address detection for DoS attack was also added to the transformation.There are 32 kinds of alerts, 638 in total.

In order to formulate negative causal rules, we divided the alerts into 12 categories: WEB access response (succeed), WEB access response (failed), Login (succeed), Login (failed), WEB invalid access, SNMP invalid access, RPC sadmind invalid access, TELNET/RSH access, DoS, MS-Sql overflow attempt, Attack detection and Other abnormal package. Table 10 shows the corresponding negative causal rule and Table 11 shows the corresponding alerts for each categories.

C. RESULT ANALYSIS

As shown in Table 12, after deduplication, the amount of data has been reduced to 55.64% of the original. After aggregation, the branches were reduced to 67.83%. After the pruning by negative causal correlation and noise reduction by non-cascading events, as shown in Table 13, 64 attack chains were finally obtained, accounting for 2.57% of the original number 2773.

Fig. 22 shows the generated attack graph, in which the DDoS attack are detected. Attacker with 202.77.162.213 first probed several intranet hosts including 172.16.112.10, 172.16.115.20, 172.16.112.50 et. al with Sadmind Overflow vulnerabilities. After gaining privileges, DDoS attack was launched against target hosts 131.34.1.31.

TABLE 12. Data deduplication and aggregation effect of LLS_DDOS.

Time range	Data	Alert	Num. of branches	Prop. of original
2000-3-7-22:23:10 2000-3-8-00:32:54	original	638	230	100%
	After deduplication	355		55.64%
	After aggregation		156	67.83%

TABLE 13. Attack chains processing effect of LLS_DDOS.

Data	Chains num.	Prop. of original	Graph num.	IP num.	Avg. IP num.	Max. IP num.	Min. IP num.
Initial chains	2773	100%					
Pruning	1389	50.09%					
Denoise	64	2.3%					
Generate graphs			1	14	14	14	14

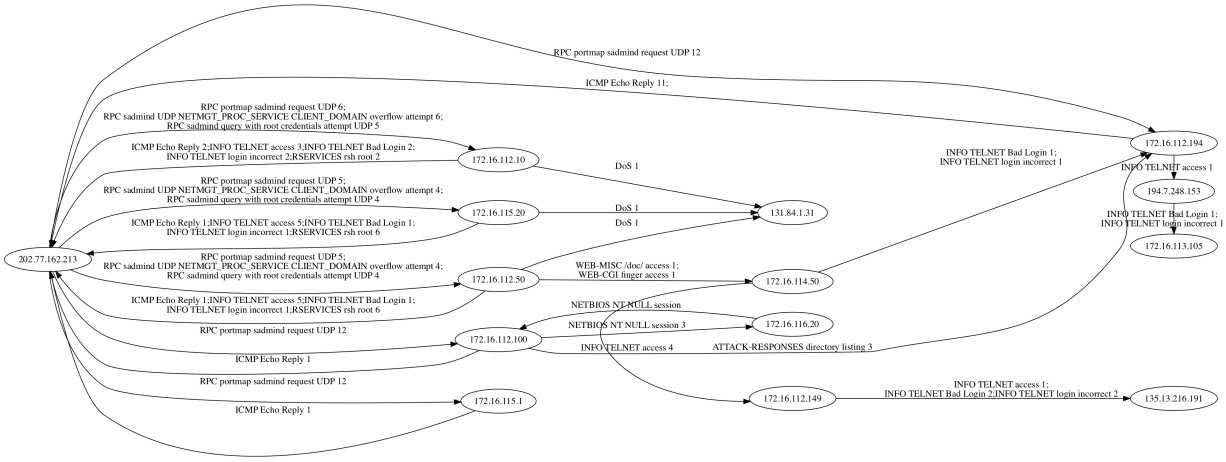

FIGURE 22. Attack graph of LLS_DDOS.

In addition to this truly completed attack, other attacks can also be found in the attack graph. Host 172.16.114.50 used TELNET to communicate with 172.16.112.149 after receiving two suspicious WEB visits from host 172.16.112.50, and then 172.16.112.149 attempted to login 135.13.216.191 for multiple times and failed; Similarly, host 172.16.112.194 used TELNET to communicate with 194.7.248.153 after receiving login attempt and Sadmind request, and then 194.7.248.153 attempted to login 135.13.216.191 for multiple times and failed.

According to the above analysis, the first DDoS attack is the real attack among the three events. The successful extraction of the attack shows the validity of our model. The other two attacks may be fake or failed attacks. It can be seen that although the model of this paper is designed for SGMS, it can also effectively detect the attack events in the internet environment.

In this paper, the correlation time window t_w is set to 3 days for the balance between resource consumption and attack chain generation. However, a smart attacker may attempt to evade detection by slowing down the attack steps. If the time difference between different attack steps is longer than the window we set, it can be a challenge to find them.

From the experiment of DARPA 2000, we can see some attacks detected are fake or uncompleted attacks. In order to reduce the false positives rate, we will try to improve the complexity of negative causal rules on the current basis. In addition, as an openly available data set, DARPA 2000 is classic but outdated. We only use it to verify our method. If the method is applied in current network, the corresponding negative causal rules should be constructed according to the network and alert characteristics.

VIII. RELATED WORK

The current security event analysis methods mainly include alert preprocessing and alert correlation.

A. ALERT PREPROCESSING

It mainly includes two parts: alert aggregation and false positives removal. The purpose of alert aggregation is to reduce the number of alerts. In the process of alert aggregation, either the source or destination IPs with the same alert content are aggregated to form super alerts [17]; an alert event with the same source and destination IP is aggregated [24]; alert events with the proper time order and in an attack life cycle are aggregated [28]. However, aggregation through the alert content or IP will loss some information, thereby hiding the path of the attack along the IP hops. Besides aggregation, common security analysis system will also remove the low-level or false-reported alerts judged by itself since to reduce

the number of alerts [10]. This can apparently improve the accuracy of attack detection.

B. ALERT CORRELATION METHODS

Alert correlation method is a method based on attack intention analysis. In order to achieve their goal without being discovered, attackers usually carefully plan and take multiple actions. The actions are related to each other and effect each other. These methods using alerts' correlation to find attacks are called alert correlation analysis methods. Alert correlation analysis methods are used in Early Warning System (EWS) [29], which can detect attackers' malicious intentions at an early stage. The methods are usually divided into following three ways [30]: similarity-based correlation analysis, attack sequence-based approach and attack sample-based machine learning.

1) *Similarity-based association analysis method*. Yang *et. al.* [31] designed algorithm to calculate the similarity between alerts and classify the alerts with high similarity into one attack scenario. In addition to clustering with alert similarity, alert level, frequency, etc., Humphrey *et. al.* [15] also introduced supporting evidence, such as vulnerability data, logs and network resources to ensure clustering quality. Fredj [8] aggregated the alerts of the same intruder. Actually, in smart grid, it is possible to remove positive alerts or group alerts belonging to different attacks into one category.
2) *Attack sequence-based approach* usually needs an established attack pattern library, and then match the attack behaviors with the patterns. Ramaki *et. al.* [17] use the attack knowledge library to initialize the attack correlation rules. New attack forms are continuously discovered in the following data mining and correlation. Then keep enriching the correlation rules. Wang *et. al.* [32] traversed the vulnerability nodes to generate attack graph according to the collected system vulnerability information, network topology information and attack sequences. After optimizing the attack graph, the best attack route can be then calculated based on the attack cost and benefit.
3) *Attack sample-based machine learning* mainly uses the statistics to learn the attack mode in the sample. Bayesian Model [18], [33], Hidden Markov Model [20] and Deep Learning Model [34] are several common models. Machine learning-based methods have developed rapidly and been widely used for the advantages of high accuracy and the ability to discover the unknown attacks. Compared with rule methods, these methods usually have lower efficiency and higher false positive rate for getting better generalization.

It is difficult to apply the above methods directly in SGMS, since the labeling of attacks and the establishment of attack pattern library are difficult. Therefore, the existing analysis methods need to be adjusted based on the data, environment and requirement of the SGMS. We propose a multi-step attack detection model based on alerts of SGMS, which can automatically analyze the alert data and extract attack events.

In order to make it more intuitive for network administrators to understand the attack situation in the network, we usually visualize the attack events. A common kind of attack graphs take alerts as the nodes. In these graphs, each edge is assigned a weight that corresponds to the number of repetition of the transition from an alert to another [8], [25]. Holgado *et. al.* [20] apply the Hidden Markov Model to analyze the attack events, and use the hidden state obtained to describe the various stages of the attack. Therefore, the nodes of the attack chains are the hidden states of the attack. These two kinds attack graphs can judge the progress of the attack through the alert or state change, but there is no concern about the specific IPs involved. Ramaki *et. al.* [17] also use alerts as the attack graph nodes. But the difference is the source and the destination IPs of the nodes are indicated in the graphs.

IX. CONCLUSION

In this paper, a multi-step attack detection model based on a small amount of priori knowledge is proposed for SGMS. The paper takes the two-month alert logs in an area of China Grid to verify the model and successfully extract attack chains. The model first obtains the alert graphs through IP correlation, and aggregates the child nodes of the alert graphs, reducing the average branch by 90.25%. Then pruning by negative causal correlation and non-cascading events denoise are used, which caused the average removal of 95.68% of the attack chains. After efficiency analysis, we found the model resource consumption and the number of attack chains are balanced when time window t_w is 3 days. During tests, there are on average of 32.02 attack graphs per window and on average of 6.68 nodes per attack graph.

REFERENCES

[1] M. D. Smith and M. E. Paté-Cornell, "Cyber risk analysis for a smart grid: How smart is smart enough? A multiarmed bandit approach to cyber security investment," *IEEE Trans. Eng. Manag.*, vol. 65, no. 3, pp. 434–447, Feb. 2018.
[2] K. Hamedani, L. Liu, A. Rachad, J. Wu, and Y. Yi, "Reservoir computing meets smart grids: Attack detection using delayed feedback networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 2, pp. 734–743, Feb. 2018.
[3] P.-Y. Kong, "Optimal configuration of interdependence between communication network and power grid," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 4054–4065, Jul. 2019.
[4] M. Chen, W. Sun, Y. Liang, B. Li, and F. Gu, "Research and implementation of power system secondary network security monitoring," *Jiangsu Elect. Eng.*, vol. 33, no. 3, pp. 31–34, 2014.
[5] P. Radoglou-Grammatikis, P. Sarigiannidis, T. Liatifis, T. Apostolakos, and S. Oikonomou, "An overview of the firewall systems in the smart grid paradigm," in *Proc. IEEE Global Inf. Infrastruct. Netw. Symp. (GIIS)*, Oct. 2018, pp. 1–4.
[6] B. B. Gupta and T. Akhtar, "A survey on smart power grid: Frameworks, tools, security issues, and solutions," *Ann. Telecommun.*, vol. 72, nos. 9–10, pp. 517–549, Oct. 2017.
[7] Z. Ni and S. Paul, "A multistage game in smart grid security: A reinforcement learning solution," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2684–2695, Sep. 2019.
[8] O. B. Fredj, "A realistic graph-based alert correlation system," *Secur. Commun. Netw.*, vol. 8, no. 15, pp. 2477–2493, 2015.
[9] J. Wu, K. Ota, M. Dong, J. Li, and H. Wang, "Big data analysis-based security situational awareness for smart grid," *IEEE Trans. Big Data*, vol. 4, no. 3, pp. 408–417, Sep. 2016.

[10] S. Salah, G. Maciá-Fernández, and J. E. Díaz-Verdejo, "A model-based survey of alert correlation techniques," *Comput. Netw.*, vol. 57, no. 5, pp. 1289–1317, 2013.
[11] I. Friedberg, F. Skopik, G. Settanni, and R. Fiedler, "Combating advanced persistent threats: From network event correlation to incident detection," *Comput. Secur.*, vol. 48, pp. 35–57, Feb. 2015.
[12] F. Quader, V. Janeja, and J. Stauffer, "Persistent threat pattern discovery," in *Proc. IEEE Int. Conf. Intell. Secur. Inf. (ISI)*, May 2015, pp. 179–181.
[13] A. Sapegin, D. Jaeger, F. Cheng, and C. Meinel, "Towards a system for complex analysis of security events in large-scale networks," *Comput. Secur.*, vol. 67, pp. 16–34, Jun. 2017.
[14] M. GhasemiGol and A. Ghaemi-Bafghi, "E-correlator: An entropy-based alert correlation system," *Secur. Commun. Netw.*, vol. 8, no. 5, pp. 822–836, 2015.
[15] H. W. Njogu and L. Jiawei, "Using alert cluster to reduce IDS alerts," in *Proc. IEEE 3rd Int. Conf. Comput. Sci. Inf. Technol.*, vol. 5, Jul. 2010, pp. 467–471.
[16] S. H. Ahmadinejad, S. Jalili, and M. Abadi, "A hybrid model for correlating alerts of known and unknown attack scenarios and updating attack graphs," *Comput. Netw.*, vol. 55, no. 9, pp. 2221–2240, 2011.
[17] A. A. Ramaki, M. Amini, and R. E. Atani, "RTECA: Real time episode correlation algorithm for multi-step attack scenarios detection," *Comput. Secur.*, vol. 49, pp. 206–219, Mar. 2014.
[18] N. Forti, G. Battistelli, L. Chisci, and B. Sinopoli, "A Bayesian approach to joint attack detection and resilient state estimation," in *Proc. IEEE 55th Conf. Decis. Control (CDC)*, Dec. 2016, pp. 1192–1198.
[19] M. R. Kabir, A. R. Onik, and T. Samad, "A network intrusion detection framework based on Bayesian network using wrapper approach," *Int. J. Comput. Appl.*, vol. 166, no. 4, pp. 13–17, 2017.
[20] P. Holgado, V. A. Villagra, and L. Vazquez, "Real-time multistep attack prediction based on hidden Markov models," *IEEE Trans. Dependable Secure Comput.*, to be published.
[21] A. A. Ramaki, A. Rasoolzadegan, and A. G. Bafghi, "A systematic mapping study on intrusion alert analysis in intrusion detection systems," *ACM Comput. Surv.*, vol. 51, no. 3, p. 55, 2018.
[22] M. Ozay, I. Esnaola, F. T. Y. Vural, S. R. Kulkarni, and H. V. Poor, "Machine learning methods for attack detection in the smart grid," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 8, pp. 1773–1786, Aug. 2016.
[23] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other Botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.
[24] I. Ghafir, M. Hammoudeh, V. Prenosil, L. Han, R. Hegarty, K. Rabie, and F. J. Aparicio-Navarro, "Detection of advanced persistent threat using machine-learning correlation analysis," *Future Gener. Comput. Syst.*, vol. 89, pp. 349–359, Dec. 2018.
[25] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, "Taxonomy and survey of collaborative intrusion detection," *ACM Comput. Surv.*, vol. 47, no. 4, p. 55, 2015.
[26] K. Kalkan and F. Alagöz, "A distributed filtering mechanism against DDoS attacks: ScoreForCore," *Comput. Netw.*, vol. 108, pp. 199–209, Oct. 2016.
[27] MIT Lincoln Laboratory. (2000). *DARPA Intrusion Detection Scenario Specific Data Sets*. [Online]. Available: http://ll.mit.edu/IST/ideval/data/2000/2000_data_index.html
[28] S. C. de Alvarenga, S. Barbon, Jr., R. S. Miani, M. Cukier, and B. B. Zarpelão, "Process mining and hierarchical clustering to help intrusion alert visualization," *Comput. Secur.*, vol. 73, pp. 474–491, Mar. 2018.
[29] S. Chen and S. Ranka, "An Internet-worm early warning system," in *Proc. IEEE Global Telecommun. Conf.*, Nov./Dec. 2004, pp. 2261–2265.
[30] J. Garcia, F. Autrel, J. Borrell, S. Castillo, F. Cuppens, and G. Navarro, "Decentralized publish-subscribe system to prevent coordinated attacks via alert correlation," in *Proc. Int. Conf. Inf. Commun. Secur.* Berlin, Germany: Springer, 2004, pp. 223–235.
[31] H. Yang, H. Qiu, and K. Wang, "Network security situation evaluation method for multi-step attack," *J. Commun.*, vol. 38, no. 1, pp. 187–198, 2017.
[32] H. Wang, Z. Chen, J. Zhao, X. Di, and D. Liu, "A vulnerability assessment method in industrial Internet of Things based on attack graph and maximum flow," *IEEE Access*, vol. 6, no. 1, pp. 8599–8609, 2018.
[33] F. Kavousi and B. Akbari, "A Bayesian network-based approach for learning attack strategies from intrusion alerts," *Secur. Commun. Netw.*, vol. 7, no. 5, pp. 833–853, 2014.
[34] M. Labonne, A. Olivereau, B. Polvé, and D. Zeghlache, "A cascade-structured meta-specialists approach for neural network-based intrusion detection," in *Proc. 16th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2019, pp. 1–6.

HUA ZHANG received the B.S. degree in communication engineering and the M.S. degree in cryptology from Xidian University, in 2002 and 2005, respectively, and the Ph.D. degree in cryptology from the Beijing University of Posts and Telecommunications (BUPT), in 2008. She is currently an Associate Professor with the Institute of Network Technology, BUPT. Her research interests include cryptography and information security.

XUEQI JIN received the M.S. degree in electric power system and automation from North China Electric Power University, in 2009. He is currently a Senior Engineer with the Dispatching Automation Department, State Gird Zhejiang Electric Power Company, Ltd. His research interests include power dispatching automation, information security, and network security.

YING LI is currently pursuing the master's degree with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. Her research interests include network security situational awareness and data analysis.

ZHENGWEI JIANG received the B.S. degree in electric power system and automation from North China Electric Power University, in 2001. He is currently a Senior Engineer with the Dispatching Automation Department, State Gird Zhejiang Electric Power Company, Ltd. His research interest includes power dispatching automation.

YE LIANG received the B.S. degree in computer science and technology from Liaoning University, in 2003. He is currently a Senior Engineer with Beijing Kedong Electric Power Control System Company, Ltd. His research interests include electrical power system network communication and information security.

ZHENGPING JIN received the Ph.D. degree in cryptography from the Beijing University of Posts and Telecommunications, in 2010. He is currently an Associate Professor with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. His research interests include design and analysis of cryptographic protocols, security in cloud computing, and industrial Internet.

QIAOYAN WEN received the B.S. and M.S. degrees in mathematics from Shanxi Normal University, Xi'an, in 1981 and 1984, respectively, and the Ph.D. degree in cryptography from Xidian University, Xi'an, in 1997. She is currently a Professor with the Beijing University of Posts and Telecommunications, Beijing, China, and the Leader of the Network Security Center, State Key Laboratory of Networking and Switching Technology, Beijing. Her current research interests include cryptography, information security, Internet security, and applied mathematics. She is also a Senior Member of the Chinese Association for Cryptologic Research.