Hindawi
Security and Communication Networks
Volume 2021, Article ID 3566360, 13 pages
https://doi.org/10.1155/2021/3566360

Research Article

G-CAS: Greedy Algorithm-Based Security Event Correlation System for Critical Infrastructure Network

Peng Lu, Teng Hu, Hao Wang, Ruobin Zhang, and Guo Wu

Institute of Computer Application, China Academy of Engineering Physics, Mianyang, Sichuan 621900, China

Correspondence should be addressed to Teng Hu; mailhuteng@gmail.com

Received 22 July 2021; Revised 18 August 2021; Accepted 31 August 2021; Published 24 September 2021

Academic Editor: Konstantinos Demertzis

The attacks on the critical infrastructure network have increased sharply, and the strict management measures of the critical infrastructure network have caused its correlation analysis technology for security events to be relatively backward; this makes the critical infrastructure network's security situation more severe. Currently, there is no common correlation analysis technology for the critical infrastructure network, and most technologies focus on expanding the dimension of data analysis, but with less attention to the optimization of analysis performance. The analysis performance does not meet the practical environment, and real-time analysis is even more impossible; as a result, the efficiency of security threat detection is greatly declined. To solve this issue, we propose the greedy tree algorithm, a correlation analysis approach based on the greedy algorithm, which optimizes event analysis steps and significantly improves the performance, so the real-time correlation analysis can be realized. We first verify the performance of the algorithm through formalization, and then the G-CAS (Greedy Correlation Analysis System) is implemented based on this algorithm and is applied in a real critical infrastructure network, which outperformed the current mainstream products.

1. Introduction

The critical infrastructure is the lifeblood of a country and region. Once attacked, it will cause irreparable losses. As the critical infrastructures need to communicate with each other for data exchange, it is necessary to form a network to connect all the infrastructures together, and this network is also called the critical infrastructure network.

To improve its security, most of the critical infrastructure network is designed with strict management measures. However, the advanced persistent threat (APT) attacks such as "Stuxnet," "Operation Aurora," "Shady Rat," "Red October," "MiniDuke," and "Colonial Pipeline" have already caused widespread damage to critical infrastructure, severely impacted people's lives, and caused billions of losses and social unrest. After all these attacks, people gradually realized that only relying on strict manage measures can no longer protect the critical infrastructure network [1–3].

Although the critical infrastructure network has multiple security products deployed for its security, these products often fail in the APT attacks' detection [4, 5]. This is because APT attacks usually do not expose their malicious payloads and try to masquerade as benign behavior. However, traditional detection tools such as IDS and vulnerability scanners have difficulty in detecting APT attacks due to their different detection methods and foci. Even if some security event alerts are issued, they are usually regarded as false alarms. As a result, it is difficult for different security products to link up to block APT attacks.

The key of APT detection is to revert the essence of security incidents, so the correlation analysis of all security incidents is required. The Security Information and Event Management (SIEM) products can collect all clues from different data sources, but due to the lack of efficient correlation analysis approaches, SIEM cannot perform correlation analysis well.

An excellent correlation analysis approach can mine the deep relationships between multiple security incidents and identify hidden threats with high efficiency. But there is no common correlation analysis approach for the critical

infrastructure network in current time, and the main reasons for this situation are as follows:

The low efficiency in processing massive data: with the continuous increase of applications' scale and complexity in the critical infrastructure network, the magnitude of data generated by users is also rising drastically. As all the clues of security incidents are submerged in massive data, so how to efficiently identify security risks from massive data is an urgent issue for correlation analysis systems. At the same time, efficient data processing is also a prerequisite for real-time analysis [6, 7].

The lack of universal norms in building the rules for analysis: when we build the rules for the correlation analysis system, the functionality, versatility, scalability, complexity, and matching efficiency of the rules' structure should be taken into account. In addition, how to relieve the decline in analysis performance when the number of rules increases is also an important factor to be considered. However, there are almost no universal norms for building the rules for analysis because different systems have different technical principles and realized methods [8, 9].

The insufficient understanding of critical infrastructure network: security vendors lack a comprehensive understanding of critical infrastructure networks and simply regard the critical infrastructure network as an intranet that is isolated with the Internet. When constructing a security supervision system, they directly transplant the correlation analysis system of the Internet to the critical infrastructure network, but the effect is very limited [10, 11].

Based on the above three reasons, it is difficult to form a common correlation analysis system for the critical infrastructure network.

To guarantee the accuracy and efficiency of the data analysis to improve the security of critical infrastructure networks, we analyze several critical infrastructure network security protection measures. On this basis, we study the current mainstream correlation analysis approach. Based on the greedy algorithm and the tree structure, we integrate the three correlation analysis methods and propose the greedy tree algorithm for correlation analysis. The greedy tree algorithm optimizes the two most time-consuming steps (meta-match and logical-match) of the correlation analysis approach. Under the premise of ensuring the accuracy of data analysis, it can greatly improve the efficiency of data analysis. Based on the greedy tree algorithm, We also design and develop a correlation analysis system, the G-CAS, which has been applied in real critical infrastructure network and has achieved remarkable results.

The main contributions of this paper are summarized as follows:

(1) We propose the greedy tree algorithm and develop a correlation analysis system named G-CAS for critical infrastructure.

(2) We build a general rule system for G-CAS and create 113 general analysis rules which have been applied in the real critical infrastructure network.

The rest of this paper is organized as follows. In Section 2, we introduce the critical infrastructure network and the methods of correlation analysis including similarity-based methods, sequence-based methods, and case-based methods, and we introduce the greedy algorithm and the tree structure. Section 3 describes the greedy tree algorithm. In Section 4, we analyze the performance of the algorithm, and Section 5 describes how we design and develop the G-CAS. In Section 6, we verify the advancement of G-CAS through experiments. In Section 7, we conclude the paper.

2. Related Work

2.1. The Critical Infrastructure Network. The critical infrastructure refers to those essential assets that are considered vital to the continued smooth functioning of the society as an integrated entity. The critical infrastructures are considered "critical" because they are deemed to be essential to the effective functioning of the society, even a minor interruption or destruction of which would have a major impact on health, safety, and the financial well-being of the citizens or impact on the effective functioning of state institutions and public administrations. The US Department of Homeland Security defined 13 infrastructure sectors: agriculture, banking and finance, chemical industry, defense industrial base, emergency services, energy, food, government, information and telecommunications, postal and shipping, public health, transportation, and water [12, 13].

As the critical infrastructures need to communicate with each other for data exchange, it is necessary to form a network to connect all the infrastructures together, and this network is also called the critical infrastructure network. To improve its security, most of the critical infrastructure networks have lots of strict management measures [14], and some are even designed as an intranet that is isolated with the Internet.

Once a critical infrastructure network is attacked, it will cause immeasurable losses to its internal key facilities. However, although the critical infrastructure network is so important, its current security protection is not excellent. The main reasons are as follows.

2.1.1. The Insufficient Knowledge of the Critical Infrastructure Network. Different from Internet, critical infrastructure network's security protection not only cares about malicious network attacks and the serious consequences that are caused but also pays attention to internal users' operations and the files' transfer process. In the critical infrastructure network, the internal users' illegal operations may cause terrible accidents, and the files' transfer process may have some risk of leakage of secrets.

Reference [10] introduced the measures taken by the US government for the security of critical infrastructure, and the author noted that due to the lack of a skilled cybersecurity workforce, the demands of the critical infrastructure

network's security protection were only partially met. It shows that most of the current security practitioners do not have enough knowledge of critical infrastructure networks.

Reference [11] contrasted the critical infrastructure cyber security policies between the US, EU, and Turkey, and the author noted that although the US performs better than the EU and Turkey, the current overall understanding of the critical infrastructure networks' protection is lacking.

2.1.2. The Relatively Backward Correlation Analysis Technology. The strict management measures have caused the critical infrastructure network's correlation analysis technology for security events to be relatively backward, and this makes the critical infrastructure network's security situation more severe.

Reference [15] introduced cyber risk management for critical infrastructure and proposed a risk analysis model. However, it has two shortcomings. One is that the model analysis is still in the theoretical stage, and the other is that the model is weak in promoting the security technology of the critical facility network.

Reference [16] introduced the existing approaches for the taxonomy of cyber threats of critical infrastructure facilities, but its focus is on threat classification rather than threat analysis.

2.2. The Correlation Analysis. At present, there have been some studies on correlation analysis, which can be roughly classified into three categories: similarity-based method, sequence-based method, and case-based method [17].

2.2.1. Similarity-Based Method. The similarity-based method makes its analysis based on the alerts' similarity, and it merges the alerts to reduce the number of them, so as to improve the efficiency of manual analysis of the alerts later.

In [18], Faraji Daneshgar and Abbaspour proposed a model that consists of an online-offline module. In [19], Hua et al. converted the nominal features to balance the datasets before clustering.

The similarity-based method's low complexity and simple implementation have proven its efficiency for reducing the number of alerts. But it is unable to find the deep causal link between the alerts and the origin data. The purpose of aggregating alerts based on similarity is to reduce the number of alerts. There is no analysis of multiple alerts, and no new alerts are discovered.

2.2.2. Sequential-Based Method. The sequential-based method makes its analysis based on the alerts' causal links, and it can identify new alerts using the alerts' prerequisites and consequent relationships.

In [20], Ramaki et al. proposed a model which creates an attack tree based on the critical episodes. In [21], Zhang et al. presented a real-time alert correlation approach based on the attack planning graph (APG). In [22], Soleimani and Ghorbani presented a multilayer framework.

Most of the research studies on the sequential-based method are still at the experimental level and most of them are merely filtering alerts. The accuracy of their alerts is mostly not very satisfactory, and all research studies rarely mention online applications and integration with existing knowledge bases.

2.2.3. Case-Based Method. The case-based method lists all the known scenes and make their analysis based on the correlation of the known. When there is a new alert, it can search the known scenes to find out the deep threat.

In [23], Liu et al. proposed an alert correlation system based on finite automata which investigated the scenes in three types of high-level views.

The case-based method is very efficient at correlating the known scenes, but it is impossible to list all the feasible scenes. There still exist some undiscovered scenes. On the other hand, expanding the set of scenes will increase the cost of data search, which challenges the online application of these methods.

2.3. The Greedy Algorithm. As one of the classic algorithms in the computer field, the greedy algorithm enjoys wide applications in terms of optimized design and task deployment of platforms and systems with its greedy concept of "division problems into several subproblems, and find the best solution for each subproblem." [24].

The greedy algorithm usually seeks the best choice in a particular situation when solving a problem [25]. The main idea of the greedy algorithm is shown in Algorithm 1.

The greedy algorithm has been widely adopted in platform and system optimization due to its intuitive strategies, high operating efficiency, low degree of complexity, and other advantages. Currently, most of the greedy algorithms are used for scheduling optimization, but the applications of greedy algorithms for data analysis platform design are rarely mentioned. For example, in [26], the greedy algorithm is applied to the learning graphical models. In [27], the greedy algorithm is applied to the task allocation for multiagent systems.

2.4. The Tree Structure. A tree structure is a way of representing the hierarchical nature of a structure in a graphical form. It is named a "tree structure" because the classic representation resembles a tree, even though the chart is generally upside down compared to a biological tree, with the "root" at the top and the "leaves" at the bottom.

The element of a tree is called "node." A node's "parent" is a node one step higher in the hierarchy (i.e., closer to the root node) and lying on the same trunk, and a node's "child" is a node one step lower in the hierarchy (i.e., farther to the root node) and lying on the same trunk. The node without children is called "lea -node," and every tree structure has a node that has no parent, so this node is called the "root node." The "root node" is the starting node of a tree and is always used to store the basic information of the tree.

Require: problem.
Ensure: solution to the problem.
(1) **if** Get a problem **then**
(2) N subproblems ← Decompose the problem.
(3) **for** subproblem_i ∈ [subproblems] **do**
(4) Compare all the solutions for subproblem_i.
(5) Get the best solution: solutions_i.
(6) Put solution_i into the subsolutions.
(7) Combine all the subsolution from 1 to i.
(8) Adjust and optimize all the subsolutions.
(9) **end for**
(10) Adjust and optimize the whole solution.
(11) **end if**
(12) **return** The best solution.

Algorithm 1: The greedy algorithm.

The tree structure enjoys wide applications in data analysis due to its strong logic, layering, scalability, and native support for recursion. In [28], the tree is used for Any-Time Time-Optimal Path-Constrained Trajectory Planning. In [29], the tree is used to do topic attention for social emotion classification.

3. The Greedy Tree Algorithm

The greedy tree algorithm integrates the above three correlation analysis methods: merge security events based on similarity to reduce the match times, so as to improve the efficiency of data analysis; analyze the correlation relationships between multiple events based on its causality and time series, so as to discover new security threats; and construct the analysis scenes based on the rules, so as to make it possible for the analyst to customize analysis scenes.

The greedy tree algorithm combines the similarity-based method with the sequential-based method, and it maps all the correlation analysis rules to a specific structure named greedy tree. In the algorithm, each greedy tree has several trunks, and according to the trunk and hierarchy level, the relations among all the rules can be divided into independence, same trunk, and inheritance. After a company's network system has been built, the types of data source that generates security events are relatively stable, so the amount of trunks of the greedy tree is relatively fixed.

The greedy tree algorithm uses the Red-Black-Tree to match the security events, and its time complexity is $O(\log n)$ [17]. The main concepts of the greedy tree algorithm are defined as follows:

(1) **DataSource-Classify:** classify all security events according to their data source and divide them into the trunks of the greedy tree.
(2) **Event-Parse:** parse all the fields of security events to the key : value format.
(3) **Meta-Match:** traverse all the security events and match every field with the filters.
(4) **Logical-Match:** match all of the meta-match results with logical operators.
(5) **Frequency-Statistic:** count up numbers of security events during the time window.
(6) **Threshold-Compare:** compare the result of Frequency-Statistic with the threshold value.
(7) **Alert-Formed:** the alert will be formed and packaged after all the conditions are matched.

The description of greedy tree algorithm is shown in Algorithm 2.

4. Performance Analysis

In this section, we will focus on the greedy tree algorithm's analysis performance, mainly from the following two aspects: based on the same data source, how to reduce the match times of data analysis; as the number of the analysis scenes (rules) increases, how to reduce the decline of the analysis efficiency. To analyze more intuitively, for the first case, we verify it by reducing the number of meta-match and logical-match in the single-rule match, and for the second case, we verify it by increasing the common items in the multirule match.

4.1. Single-Rule Match

4.1.1. Traditional Algorithm. In order to analyze more intuitively, we define Tp as the time cost for each meta-match, Tc as the time cost for each logical-match, m as the number of fields to be matched for each rule, and n as the number of logical-match.

So, the total time cost of both meta-match and logical-match is as follows:

$$T = \sum_{i=1}^{m} Tp_i + \sum_{j=1}^{n} Tc_j. \quad (1)$$

4.1.2. Greedy Tree Algorithm. (1) Meta-Match. Based on the above analysis, it can be concluded that the time cost can be reduced by cutting the matching times. All the rules for analysis have a prerequisite condition, that is, data source

Require: the data of security events.
Ensure: the security alerts with specific conditions.
(1) **if** Received a security event **then**
(2) Greedy-Tree ← Init the rules.
(3) DataSource classify.
(4) Key-value ← Event-Parse.
(5) LogicMatchers ← Generate Logic Matcher.
(6) **for** Key_i ∈ [Keys] **do**
(7) Meta-Match with tree structure.
(8) Optimized in the greedy tree.
(9) **end for**
(10) **for** LogicMatcher_j ∈ [LogicMatchers] **do**
(11) Logical-Match based on the greedy algorithm.
(12) Optimized in the greedy tree.
(13) **end for**
(14) Frequency-Statistic.
(15) Threshold-Compare.
(16) Alert-Formed.
(17) **end if**
(18) **return** Alerts.

Algorithm 2: The greedy tree algorithm.

type, and we call it "Pr condition." For example, when the spreading of virus security event is analyzed, its data source is the antivirus software, so its "Pr condition" is that all the data being analyzed are from antivirus software; in the same way, when analyzing the attack, its "Pr condition" is that all the data being analyzed are from IPS/IDS or firewall. The greedy tree algorithm will first match the "Pr condition." If "Pr condition" is not met, all the subsequent steps can be skipped directly.

Assume that the volume of data source Si accounts for $1/S$ of the total data volume, so the cost of the ratio of the traditional algorithm to the greedy tree algorithm in terms of a single event is expressed as follows:

$$\frac{\sum_{i=1}^{m} Tp_i + \sum_{j=1}^{n} Tc_i}{(1-(1/s)) * Tp_1 + (1/s)^* \sum_{i=1}^{m} Tp_i}. \tag{2}$$

According to the formula, Tp_1 stands for the cost for the matching of the events that do not meet "Pr condition," which is equivalent to $1/m$ of the original value.

(2) Logical-Match. For the traditional algorithm, the logical-match process will first calculate the value of each atomic formula separately and then obtain the matching result through logic operations ("AND" and "OR"). The logical-match process of traditional algorithm is shown in Figure 1.

Supposing the number of logical-match for "AND operator" is α and the number of logical-match for "OR operator" is β, the cost for the matching of traditional algorithm is as follows:

$$\sum_{j=1}^{n} Tc_j = \sum_{j=1}^{\alpha} Tc_j + \sum_{j=1}^{\beta} Tc_j. \tag{3}$$

The greedy tree algorithm uses logical matchers for optimization of logic operations. We divide the logical matchers into two categories:

(1) **AND logical matcher:** if "false" appears in any meta-match result, return "false," and the subsequent matching steps will be skipped.
(2) **OR logical matcher:** if "true" appears in any meta-match result, return "true," and the subsequent matching steps will be skipped.

The logical-match process of greedy tree algorithm is shown in Figure 2.

Supposing the "AND logical matcher" returns "false" after α_1th times meta-match and the "OR logical matcher" returns "true" after β_1th times meta-match, the time cost of greedy tree algorithm is expressed as follows:

$$\sum_{j=1}^{\alpha_1} Tc_j + \sum_{j=1}^{\beta_1} Tc_j = \sum_{j=1}^{\alpha_1+\beta_1} Tc_j. \tag{4}$$

Because $\alpha_1 \le \alpha$, $\beta_1 \le \beta$, the time cost of logical-match is reduced. Based on the conclusions above, the time cost for "meta-match" and "logical-match" is as follows:

$$T = \sum_{j=1}^{\alpha_1+\beta_1} Tc_j + \left(1-\frac{1}{s}\right) * Tp_1 + \frac{1}{s} * \sum_{i=1}^{m} Tp_i. \tag{5}$$

As the result of logical matches returned, the subsequent field matching process will be skipped. Supposing that the number of meta-match after optimization is "m_1" ("m_1"≤"m"), the time cost of the greedy tree algorithm is as follows:

$$T = \sum_{j=1}^{\alpha_1+\beta_1} Tc_j + \left(1-\frac{1}{s}\right) * Tp_1 + \frac{1}{s} * \sum_{i=1}^{m_1} Tp_i. \tag{6}$$

The ratio of time cost for the traditional algorithm and the greedy tree algorithm is expressed as follows:

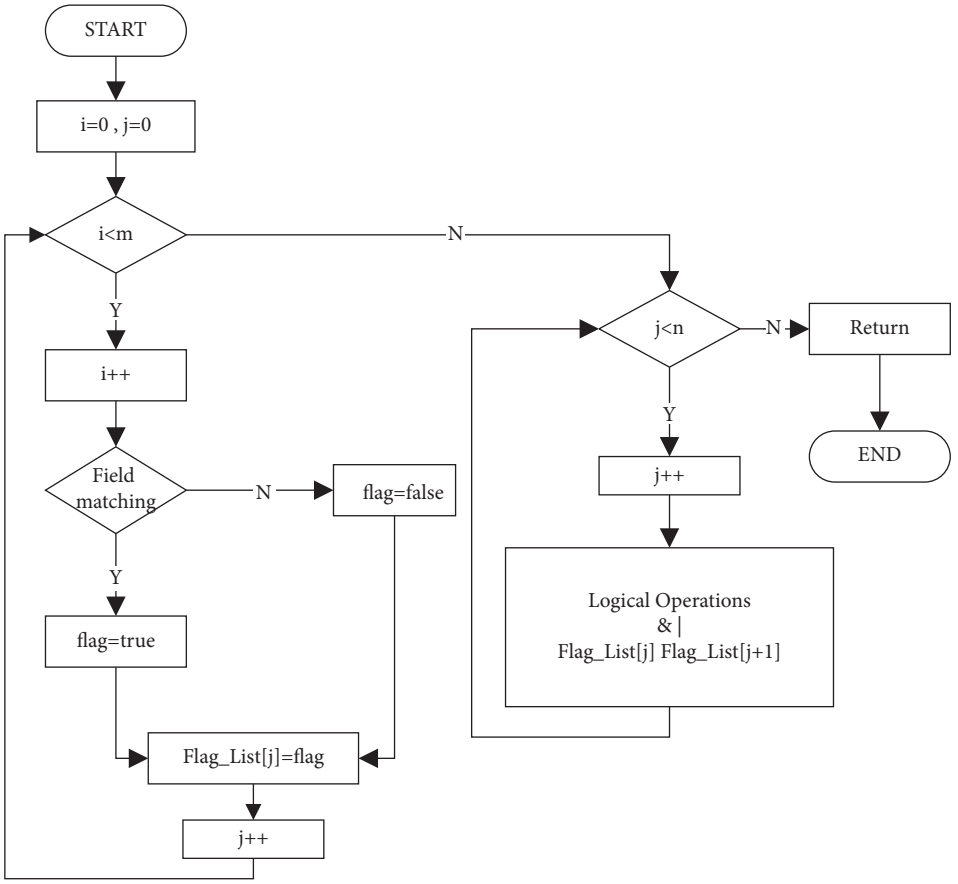

Figure 1: The flowchart of logical-match in the traditional algorithm.

$$\frac{\sum_{i=1}^{m} Tp_i + \sum_{j=1}^{n} Tc_i}{\sum_{j=1}^{\alpha_1+\beta_1} Tc_j + (1-(1/s)) * Tp_1 + (1/s)^* \sum_{i=1}^{m_1} Tp_i}. \quad (7)$$

According to the formula, as "m" and "n" are fixed factors, the factors that affect the time optimization rate are "$\alpha_1 + \beta_1$," "m_1," and "S," and the optimization effect is negatively correlated with the three factors.

4.2. Multirule Match

4.2.1. Traditional Algorithm. For traditional algorithm, the relationship between all the rules is independent. Assuming that the number of rules is "k," the time cost is

$$Ts = \sum_{i=1}^{m} Tp_i + \sum_{j=1}^{n} Tc_j,$$
$$T = \sum_{i=1}^{k} Ts_i. \quad (8)$$

According to the formula, as the number of rules increases, the time cost will increase linearly and the performance will become significantly poorer.

4.2.2. Greedy Tree Algorithm. When all the rules in the greedy tree algorithm are independent, the time cost is as follows:

$$Ts = \sum_{j=1}^{\alpha_1+\beta_1} Tc_j + \left(1 - \frac{1}{s}\right) * Tp_1 + \frac{1}{s} * \sum_{i=1}^{m_1} Tp_i,$$
$$T = \sum_{i=1}^{k} Ts_i. \quad (9)$$

According to the formula, when all the rule-trees are independent, the matching time increases linearly with the increase of number of rules. In order to solve this problem, the rules are described as follows.

Create several roots for the rule-trees, and each root represents a type of data source, such as firewall, IPS/IDS, antivirus, and so on. Resolve the rules and put each rule into the root system to classify data source and use the root system filter to complete "Pr condition" match. Extract the public factors of all the rules in each root so that the "Pr condition" match of multiple rules can finish at one time.

Assuming that there are two rules to analyze the number of deny logs of firewall to discover the attack, the two rules are Rule 1: {key 1: source, value 1: firewall; key 2

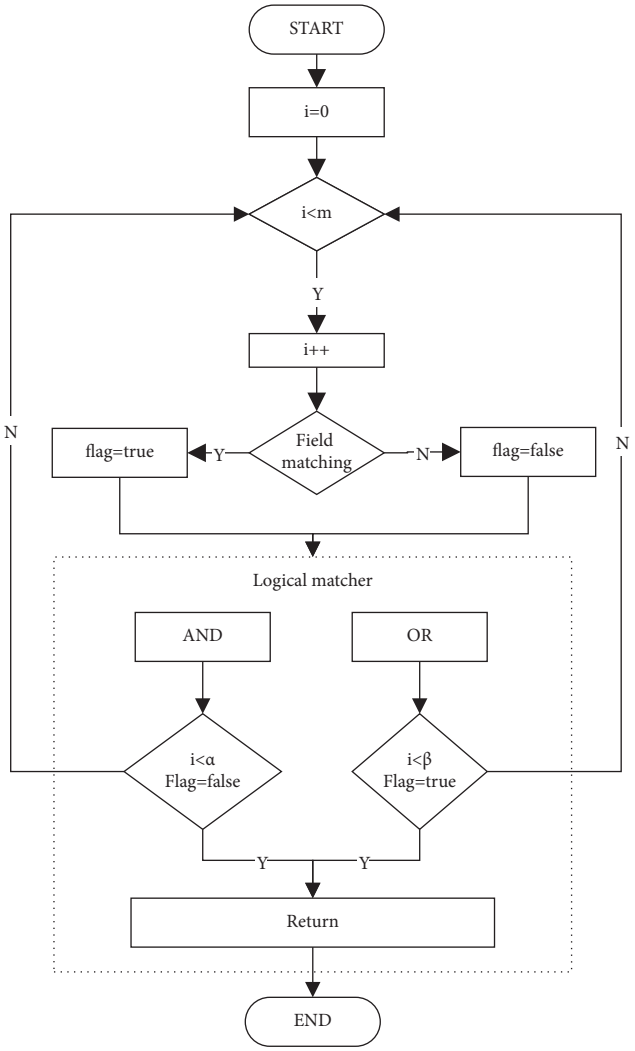

Figure 2: The flowchart for logical-match of greedy tree algorithm.

: operation, value 2; deny} and Rule 2: {key 1: source, value 1 : firewall; key 2 : operation, value 2: deny; key 3: dip, value : 192.168.1.1 or 192.168.1.2}. It can be found that key 1, value 1 and key 2, value 2 in Rule 1 and Rule 2 are exactly the same, and Rule 2 only needs to match key 3 based on the Rule 1's match result; there is no need for Rule 2 to match key 1 and key 2, which can greatly improve the match efficiency. The relationships between multiple rules in the greedy tree are shown in Figure 3.

Based on the figure, Branch 1, Branch 2, and Branch 3 have the same trunk (Trunk 1), and Branch 4's root is Trunk 2 . The relationships among rules are defined as follows:

Independent. All the rules have different trunks and are independent of each other. As shown in Figure 3, the relationships between Branch 4 and Branch 1, Branch 4 and Branch 2, and Branch 4 and Branch 3 are all independent.

Same trunk. The rules have the same trunk. As shown in Figure 3, Branch 1 and Branch 2, Branch 2 and Branch 3, and Branch 1 and Branch 3 all have the same trunk (Trunk 1), the relationships among Branch 1, Branch 2, and Branch 3 are "Same Trunk."

Inheritance. There is a nesting relationship between the rules, such as Branch 2 and Branch 1 in Figure 3 (inheritance can be considered as a special case of the same trunk).

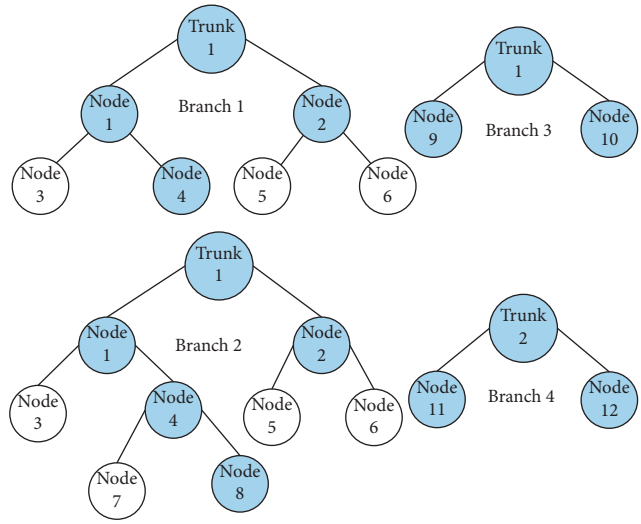

Figure 3: The schematic diagram of tree structure.

In order to analyze more intuitively, we define p as the number of public match factors, q as the matching frequency of each factor, u as the number of public logical matchers, and k as the number of rules.

The time saved by the greedy tree algorithm is as follows.

Independent:

$$\Delta T = 0. \tag{10}$$

Same trunk:

$$\Delta T = (k-1) * Tc_1 + Tp_1. \tag{11}$$

Inheritance:

$$\Delta T = q * Tc_j,$$
$$Tc_i = \sum_{i=1}^{p} \Delta T_i + \sum_{i=1}^{u} Tp_i = \sum_{i=1}^{p} q * Tc_i + \sum_{i=1}^{u} Tp_i. \tag{12}$$

According to the formula, during the analysis, the number of public match fields is inversely proportional to the matching time cost.

Through formalization, it can be concluded that the greedy tree algorithm has obvious advantages compared with the traditional algorithm. In the single-rule match case, the greedy tree algorithm reduces the steps of meta-match and logical-match; in the multirule match case, based on the public match keys and values, the greedy tree algorithm enjoys a better effect on the optimization of the rules with the same trunk and inheritance relationships, and when the number of rules grows, the performance of the correlation analysis will not decline linearly.

5. G-CAS

Based on the above verification, this section will introduce the design of the correlation rules and the implementation of G-CAS.

5.1. The Design of the Rules. Based on the functional demands of the greedy tree algorithm, the rule for correlation analysis is defined as a five-tuple: Rule = $< R, I, N, W, L >$ where $R\{R_1, R_2, \ldots, R_n\}$ stands for relationship, and it indicates the relationship among rules, such as independent, same root, and inheritance; $I\{I_1, I_2, \ldots I_n\}$ stands for info, including rule name, type, status, level, and time information (creation time, modification time, on-off time, and so on); $N\{N_1, N_2, \ldots N_n\}$ stands for node, which is the basic structure of meta-match and contains basic information of various field match, such as Key, Value, and operators (equal, not equal, less than, greater than, and fuzzy matching, etc.); $W\{W_1, W_2, \ldots W_n\}$ stands for weight, which is the weight value in each matching node; $L\{L_1, L_2, \ldots L_n\}$ stands for logical operator, which is a logical-match symbol among multiple nodes ("AND" and "OR").

The structure of rules based on the greedy tree algorithm is shown in Figure 4.

As shown in Figure 4, each rule is classified into different trunks based on the data source. The deep color nodes in each rule tree are logical matching nodes, and the light color nodes are meta-match nodes. Each node has a weight value that indicates how important this meta-match is.

We take the detection of attacks from the traffic flow as an example, assuming that the elements that needed to be matched are as follows: whether the data are traffic flow; whether the port changes regularly; whether the data are consistent with the information in threat intelligence; whether the IP or port is focused by network security personnel; and whether there is any scan or exploit behavior. The weight is divided according to the importance of the elements, namely: W_6 (data are captured from network flow traffic), W_5 (exploit behavior), W_4 (consistent with the information in threat intelligence), W_3 (scan behavior), W_2 (focused IP), and W_1 (focused ports). In the correlation analysis, the node with the highest weight will be matched first. As shown in Figure 4, the node with the weight of 6 will be matched first. If the node's match result is false, it will directly return false and the other nodes' match will be skipped; if the node's match result is true, the node with the weight of 5 will be matched. If its match returns true, the node with the weight of 3 will be skipped; otherwise, it continues to match the nodes with a lower weight.

According to function and hierarchical relationships, the rule tree nodes are divided into four categories: Root, Trunk, Branch, and Leaf. The nesting relationships and storage contents of each node are shown in Figure 5.

As shown in Figure 5, Root is the root of the rule tree, it saves the information of the network such as the domain information, and it also saves the information of the whole rule system, such as the data source list, rule list, black/white list, and the relationships of all the rules. One Trunk means a data source, it saves the information of the data source, and it also saves the relationships of all the rules in it. It may have several Branches, and each Branch means a rule, corresponding to R in the five-tuple. Branch can be embedded under the Trunk, and it saves the information of a rule which contains the name, id, and the logical relationship among child nodes, corresponding to L and I in the five-tuple, and Branch can also be nested under Branch. Leaf can be nested under Branch, and the information stored in Leaf is mainly used for meta-match, including key, value, and weight, corresponding to N and W in the five-tuple.

The rules of the greedy tree algorithm are described with JSON array and stored in plain text. The average storage space for a rule is about 1 kB. An example of the rule is shown in Figure 6.

5.2. The Implementation of the G-CAS. In order to verify the effect of greedy tree algorithm, the G-CAS based on it has been designed and implemented. The functional architecture diagram of the G-CAS is shown in Figure 7.

After the G-CAS starts, the initial job will be completed based on the configuration, and then all the rules and resources will be loaded. The G-CAS will generate the greedy tree rule systems, all the rules' relationships and the information of the critical infrastructure network will be saved in the root node, and the data source info will be saved in the trunk node.

The G-CAS maps each rule as a branch node which is belonged to specific trunk, and the information of meta-match and logic-match are saved in the leaf node which belonged to specific branch.

As the security event occurred in the critical infrastructure network, the data will be sent to the event-parse module of the G-CAS, the event will be parsed to the "key: value" format, and then the data are sent to the rule system for analysis. If all the conditions are matched, the alerts of threats will be generated and saved to the database.

Based on data sources, the G-CAS can greatly improve the efficiency of data analysis. The greedy tree rule system can well store the users' behavior records and files' transfer records. So, the G-CAS can more accurately discover the hidden risks and improve critical infrastructure network's security.

6. Experiment

6.1. Preparation. We conduct the experiment in our critical infrastructure network with almost 20,000 computers, 1,000 servers, 200 switches, 100 security devices, and 7000 users.

We compare the G-CAS with Apache Flink [30] (standalone, Version-1.6.3) and Apache Storm [31] (Version-1.1.0), respectively. In order to ensure the fairness of the comparative experiment, we take the following measures:

(1) The three systems were deployed on three machines with the same configurations, and the hardware information is given in Table 1.

(2) All the three machines had the same operating system: CentOS 7.4.

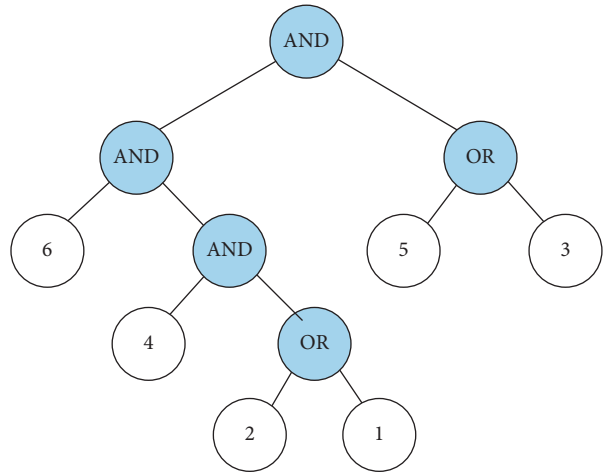

Figure 4: The schematic design of rules.

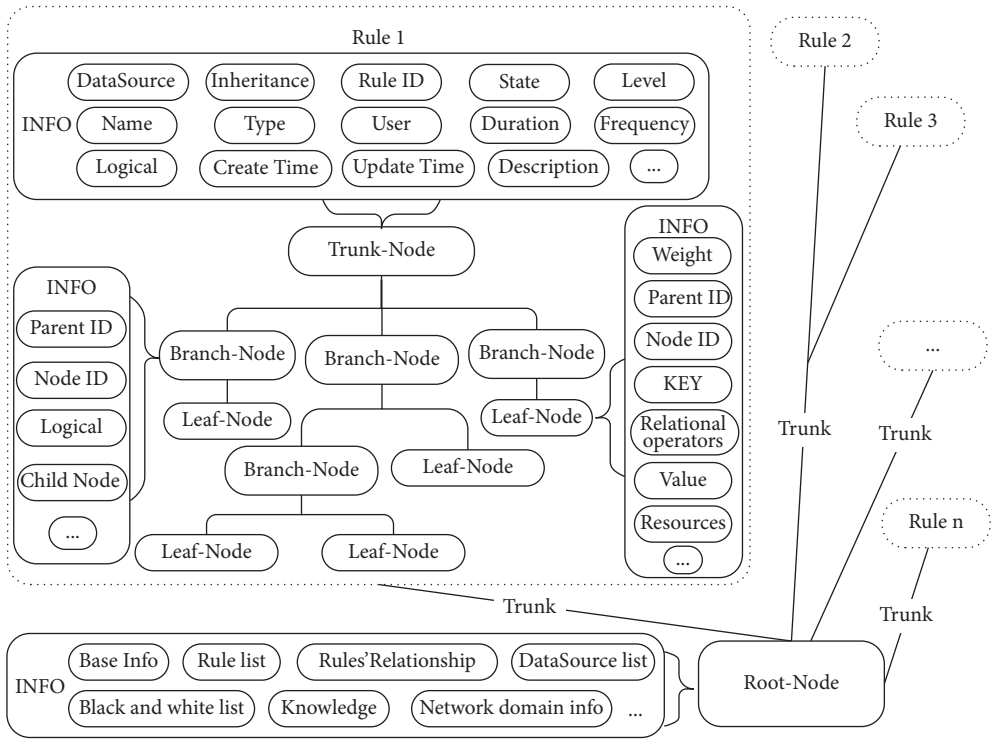

Figure 5: The structure of rules designed for greedy tree algorithm.

(3) All the data of the experiment include the normal business data of the critical infrastructure network and the data of the internal users' illegal operations and file transfer exception, and the data are processed into the format of "key:value" by the same event processing module.

(4) The data are sent to Apache Kafka [32] (standalone, Version-1.7.0) for cache, and the same API is used to read and write.

6.2. Experiment. In order to make the experimental results fairer, we made 10 independent comparative experiments and took the average value for comparison. Moreover, all the data for each experiment were independent. The experiments were designed from the following two aspects.

6.2.1. Single Rule. A single rule with five matched fields was built for three systems, and the processing performance and detection rate were detected for the data of firewall and

```
Rule :
{
    Trunk:IPS,
    Name:ips_alarms,
    ID:1234567,
    Parent_ID:null,
    Logic:AND,
    Nodes:{
[ Nodes :
  { Name : Nodes_1 , Logic : AND
    [
      Node :
      {
          Name:node_1,
          Key:src_ip,
          Value:192.168.1.1,
          Operation:equal,
          Weight:5
      },
      Node :
      {
          Name:node_2,
          Key:dst_ip,
          Value:192.168.1.2,
          Operation:equal,
          Weight:5
      }
    ]
  },
  Nodes :
    { Name : Nodes_2 , Logic : OR
      [
        Node :
        {
            Name:node_3,
            Key:Action,
            Value:Accept,
            Operation:equal,
            Weight:5
        },
        Node :
        {
            Name:node_4,
            Key:Action,
            Value:Deny,
            Operation:equal,
            Weight:5
        }
      ]
    }
]
}
```

Figure 6: Example of a greedy tree rule.

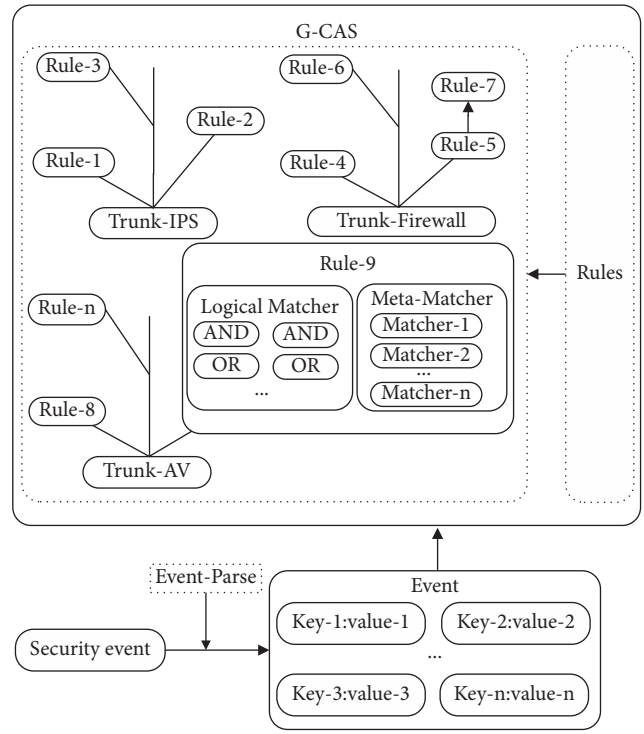

Figure 7: The design of the G-CAS.

Table 1: Hardware of the machines.

Hardware	Details of config
CPU	Xeon E5 with 20 cores and 40 threads
RAM	DDR4, 256G (32G ∗ 8)
System disk	2 ∗ 960G, solid-state disk, raid 1
Data disk	8 ∗ 8T, raid 0

antivirus (among all the data, the firewall data share about 85% and the antivirus data share about 1%). The experimental data are shown in Figures 8 and 9.

6.2.2. Multiple Rules. There are 13 types of data sources in the experimental network environment. Therefore, 13 independent rules were configured for all three systems, with the same data source, same matched items, and same logic operation times. The processing performance and detection rate are compared in Figures 10 and 11.

Based on 13 independent rules, the number of rules increases by 10 each time. As the number of rules increases, the processing performance is compared as shown in Figure 12. The detection rate in the internal users' illegal operations and files' transfer exception is compared as shown in Figures 13 and 14.

6.3. Result Analysis

6.3.1. Single Rule. Figures 8 and 9 clearly show that when there is only a single rule, all the three systems have almost the same detection rate. For the firewall data, the

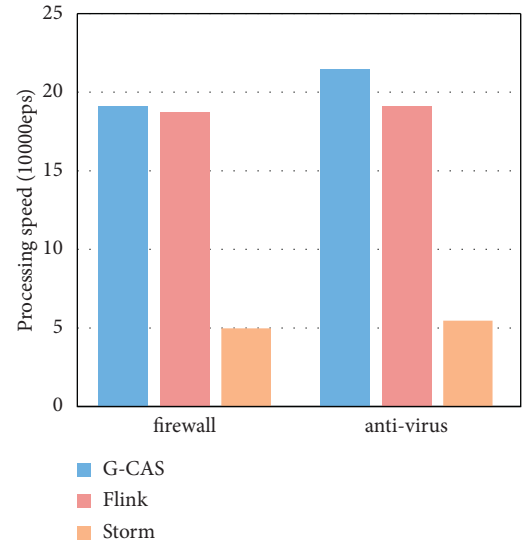

Figure 8: The processing speed of single rule.

G-CAS almost has no difference from Flink in processing speed and has a greater advantage than Storm. For the antivirus data, the G-CAS has a little better performance

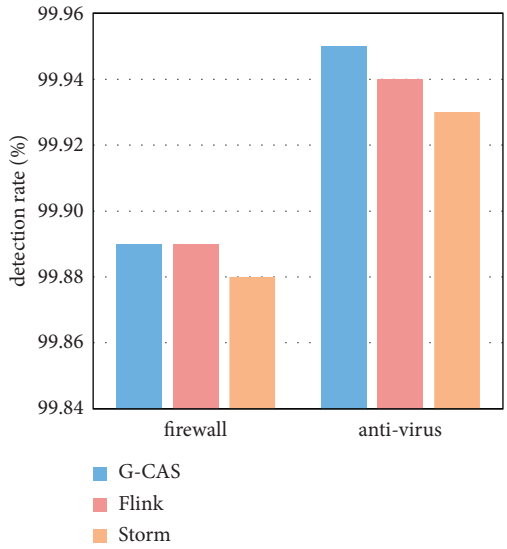

Figure 9: The detection rate of single rule.

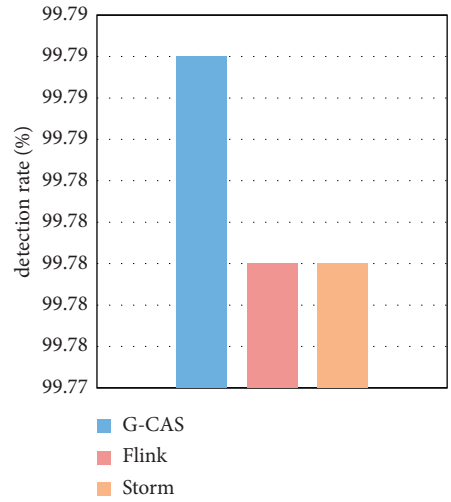

Figure 11: The detection rate with 13 independent rules.

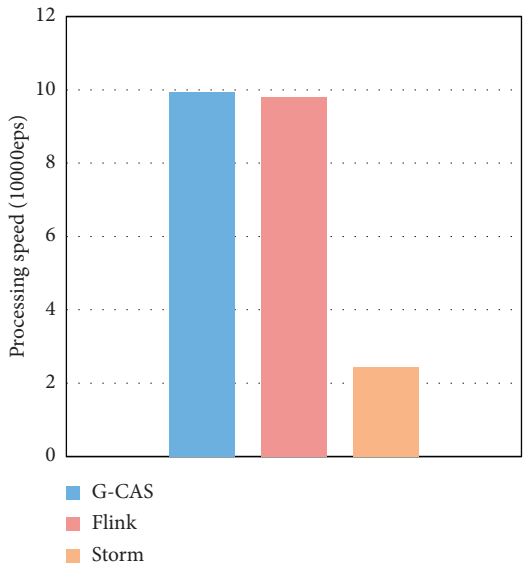

Figure 10: The processing speed with 13 independent rules.

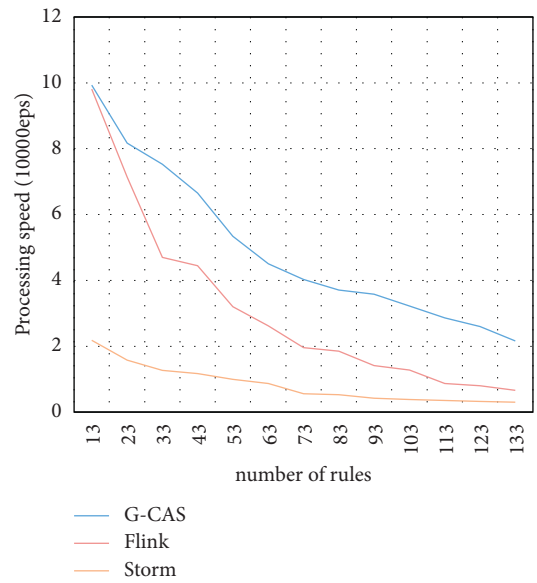

Figure 12: The processing speed with the number of rules increased.

in processing speed than Flink and has a greater advantage than Storm.

Through experimental comparison, it is found that when there is only a single rule in the system, both the G-CAS and Flink have a greater advantage than Storm in detection rate and process performance in firewall data analysis. For antivirus data, all three systems almost have no difference in detection rate, and the G-CAS has a little better process performance than the other two systems.

6.3.2. Multiple Rules. Figures 10 and 11 clearly show that for multiple independent rules, the G-CAS has a slighter advantage than Flink in processing performance and has a greater advantage than Storm. All three systems have almost the same detection rate for attack detection. Figure 12 clearly shows that for the situation where the number of rules increases linearly, the G-CAS enjoys a better processing performance than Flink and Storm. Figures 13 and 14 clearly show that the G-CAS has a better performance in detecting the internal users' illegal operations and file transfer exceptions.

Through experimental comparison, it is found that for multiple independent rules, the G-CAS has a slighter advantage than Flink in processing performance and has a greater advantage than Storm. In the detection of internal users' illegal operations and file transfer exceptions, G-CAS has a better performance.

6.3.3. Summary. By comparing with Flink and Storm, it can be found that the G-CAS has a greater advantage in

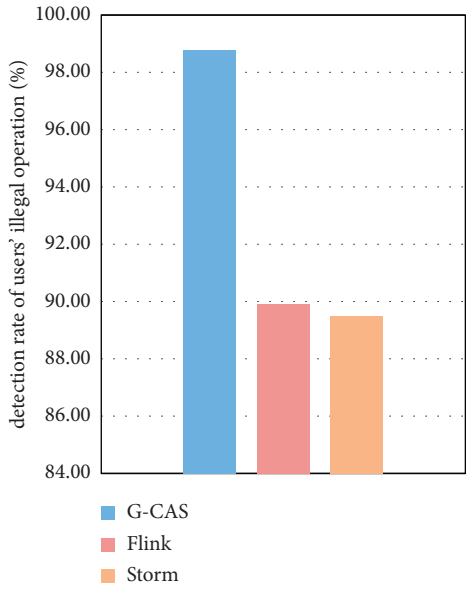

Figure 13: The detection rate of users' illegal operation with 113 rules.

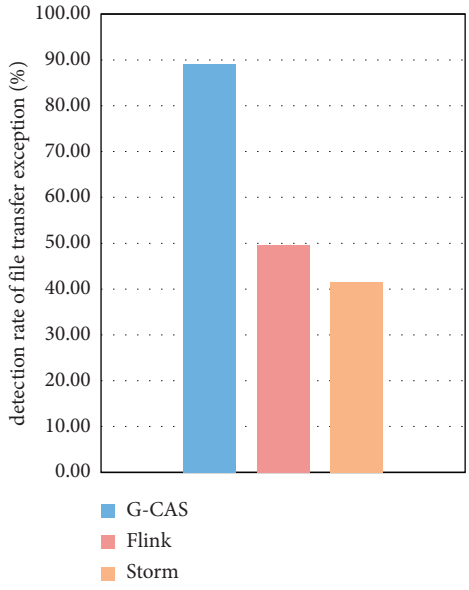

Figure 14: The detection rate of files' transfer exception with 113 rules.

processing speed and can be fully competent for real-time analysis tasks. For the critical infrastructure network, the G-CAS has a better performance in detecting the internal users' illegal operations and file transfer exceptions.

Nevertheless, there are still some limitations to our work: the detection rate of the G-CAS in file transfer exception detecting is relatively low because the file transfer exception is difficult to define and the paths of the file transfer are very complex. The greedy tree algorithm and the G-CAS lack a solution to the problem of cluster deployment and load balancing.

7. Conclusions

This paper proposes a greedy tree algorithm for the critical infrastructure network's correlation analysis and proves its efficacy through formalization and experimental verification. The G-CAS based on the algorithm has been designed and applied in the real critical infrastructure networks. Based on the G-CAS, this paper has summarized 113 general analysis rules, which have been applied and promoted in real critical infrastructure networks.

There are still some works to do in the future: we will do more research on the file transfer exception and try to improve the detection rate of the G-CAS in file transfer exception detection. We will try to explore a solution for the G-CAS to solve the problem of cluster deployment and load balance, in order to break the limitation of computer hardware resources.

Data Availability

The critical infrastructure network's data used to support the findings of this study are currently under embargo while the research findings are commercialized. Requests for data, 12 months after publication of this article, will be considered by the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported by the Defense Industrial Technology Development Program (fund no. JCKY2019602B013), CAEP Foundation (fund nos. CX2019040 and CX20210011), Institute of Computer Application, China Academy of Engineering Physics (fund no. SJ2021A03), China Mobile Information Communication Technology Co. Ltd. (Chengdu) 2020 UAV Operation Management Platform Phase II (Package 2: Safety Subsystem) (fund no. CMCMII-202001245).

References

[1] D. Pretyman, T. Greaves, and A. Millerick, "The role of natural gas utilities and pipeline operators in a decarbonized economy," *Climate and Energy*, vol. 38, no. 1, pp. 1–10, 2021.

[2] D. Wei, H. Ning, F. Shi et al., "Dataflow management in the internet of things: sensing, control, and security," *Tsinghua Science and Technology*, vol. 26, no. 6, pp. 918–930, 2021.

[3] S. Rass, S. König, and S. Schauer, "Defending against advanced persistent threats using game-theory," *PLoS One*, vol. 12, no. 1, Article ID e0168675, 2017.

[4] L. Huang and Q. Zhu, "Adaptive strategic cyber defense for advanced persistent threats in critical infrastructure networks," *ACM SIGMETRICS-Performance Evaluation Review*, vol. 46, no. 2, pp. 52–56, 2019.

[5] I. Friedberg, F. Skopik, G. Settanni, and R. Fiedler, "Combating advanced persistent threats: From network event correlation to incident detection," *Computers & Security*, vol. 48, pp. 35–57, 2015.

[6] S. Salloum, J. Z. Huang, Y. He, and X. Chen, "An asymptotic ensemble learning framework for big data analysis," *IEEE Access*, vol. 7, pp. 3675–3693, 2018.

[7] Y. Zhang, "Association analysis of user location, social behavior and browsing behavior based on large-scale network traffic," Master's thesis, Beijing University of Posts and Telecommunications, Beijing, China, 2019.

[8] Y. Chen, S.-Q. Shan, L. Liu, and Y. Li, "Minimum-redundant and lossless association rule-set representation," *Acta Automatica Sinica*, vol. 34, no. 12, pp. 1490–1496, 2009.

[9] X. Hua, *Research on application performance optimization methods for big data processing*, Ph.D. thesis, Zhejiang University, Hangzhou, China, 2019.

[10] P. Wagner, "Critical infrastructure security," *SSRN Electronic Journal*, vol. 12, 2021.

[11] E. Düveroğlu, *A comparative analysis of critical infrastructure cyber security policies: best practices from the US, EU and Turkey*, Ph.D. thesis, Bilkent University, Ankara, Turkey, 2020.

[12] G. Brown, M. Carlyle, J. Salmerón, and K. Wood, "Defending critical infrastructure," *Interfaces*, vol. 36, no. 6, pp. 530–544, 2006.

[13] Z. Tong, F. Ye, M. Yan, H. Liu, and S. Basodi, "A survey on algorithms for intelligent computing and smart city applications," *Big Data Mining and Analytics*, vol. 4, no. 3, pp. 155–172, 2021.

[14] A. Kalashnikov and E. Sakrutina, "The model of evaluating the risk potential for critical infrastructure plants of nuclear power plants," in *Proceedings of the 2018 Eleventh International Conference Management of large-scale system development*, pp. 1–4, IEEE, Moscow, Russia, 3 October 2018.

[15] M.-E. Paté-Cornell, M. Kuypers, M. Smith, and P. Keller, "Cyber risk management for critical infrastructure: a risk analysis model and three case studies," *Risk Analysis*, vol. 38, no. 2, pp. 226–241, 2018.

[16] M. Komarov, A. Davydiuk, A. Onyskova, V. Tkachenko, and S. Honchar, "Requirements for a taxonomy of cyber threats of critical infrastructure facilities and an analysis of existing approaches, systems, decision and control in energy II," in *Studies in Systems, Decision and Control*, pp. 189–205, Springer, Cham, Switzerland, 2021.

[17] E. Mahdavi, A. Fanian, and F. Amini, "A real-time alert correlation method based on code-books for intrusion detection systems," *Computers & Security*, vol. 89, Article ID 101661, 2020.

[18] F. Faraji Daneshgar and M. Abbaspour, "Extracting fuzzy attack patterns using an online fuzzy adaptive alert correlation framework," *Security and Communication Networks*, vol. 9, no. 14, pp. 2245–2260, 2016.

[19] H. H. W. Hua, M. M. Siraj, and M. M. Din, "Integration of pso and k-means clustering algorithm for structural-based alert correlation model," *International Journal of Integrated Care*, vol. 7, no. 2, pp. 34–39, 2017.

[20] A. A. Ramaki, M. Amini, and R. Ebrahimi Atani, "Rteca: Real time episode correlation algorithm for multi-step attack scenarios detection," *Computers & Security*, vol. 49, pp. 206–219, 2015.

[21] J. Zhang, X. Li, and H. Wang, "Real-time alert correlation approach based on attack planning graph," *Journal of Computer Applications*, vol. 36, no. 6, pp. 1538–1543, 2016.

[22] M. Soleimani and A. A. Ghorbani, "Multi-layer episode filtering for the multi-step attack detection," *Computer Communications*, vol. 35, no. 11, pp. 1368–1379, 2012.

[23] L. Liu, K. F. Zheng, and Y. X. Yang, "An intrusion alert correlation approach based on finite automata," in *Proceedings of the 2010 International Conference on Communications and Intelligence Information Security*, pp. 80–83, IEEE, Xi'an, China, October 2010.

[24] Y. Cao, T. Ohtsuki, and X.-Q. Jiang, "Precoding aided generalized spatial modulation with an iterative greedy algorithm," *IEEE Access*, vol. 6, pp. 72449–72457, 2018.

[25] T. Zhang, "Adaptive forward-backward greedy algorithm for learning sparse representations," *IEEE Transactions on Information Theory*, vol. 57, no. 7, pp. 4689–4708, 2011.

[26] A. Ray, S. Sanghavi, and S. Shakkottai, "Improved greedy algorithms for learning graphical models," *IEEE Transactions on Information Theory*, vol. 61, no. 6, pp. 3457–3468, 2015.

[27] J. Zhou, X. Zhao, X. Zhang, D. Zhao, and H. Li, "Task allocation for multi-agent systems based on distributed many-objective evolutionary algorithm and greedy algorithm," *IEEE Access*, vol. 8, pp. 19306–19318, 2020.

[28] P. Shen, X. Zhang, and Y. Fang, "Tree-search-based any-time time-optimal path-constrained trajectory planning with inadmissible island constraints," *IEEE Access*, vol. 7, pp. 1040–1051, 2018.

[29] C. Wang, B. Wang, and M. Xu, "Tree-structured neural networks with topic attention for social emotion classification," *IEEE Access*, vol. 7, pp. 95505–95515, 2019.

[30] D. García-Gil, S. Ramírez-Gallego, S. García, and F. Herrera, "A comparison on scalability for batch big data processing on apache spark and apache flink," *Big Data Analytics*, vol. 2, no. 1, pp. 1–11, 2017.

[31] S. Chintapalli, D. Dagit, B. Evans et al., "Benchmarking streaming computation engines: Storm, flink and spark streaming," in *Proceedings of the 2016 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*, pp. 1789–1792, IEEE, Chicago, IL, USA, May 2016.

[32] B. R. Hiraman, C. M. Viresh, and K. C. Abhijeet, "A study of apache kafka in big data stream processing," in *Proceedings of the 2018 International Conference on Information, Communication, Engineering and Technology (ICICET)*, pp. 1–3, IEEE, Pune, India, August 2018.