Virtual Network Security Control Mechanism based on Multi-level Alerting and Linkage Defense

Weiwei Li¹*, Cheng Zhou¹

¹ Global Energy Interconnection Research Institute, Nanjing, Jiangsu, 210000, China
*Corresponding author’s e-mail: liweiwei@geiri.sgcc.com.cn

Abstract. The intrusion detection system can effectively detect network attacks dynamically by analyzing network traffic or data packets. However, for complex multi-step attacks with high concealment and persistence, traditional intrusion detection systems generally have the problems of large redundancy of alert data and poor data readability, which seriously affects the security administrators to quickly identify attack behaviors and intentions. To address the above problems, this paper proposes a multi-level alert aggregation method based on software-defined security and linked correction defense based on the Markov chain model. This method designs the reporting message format between the data plane security component and the control plane security controller, and automatically extends the time window by using the temporal proximity relationship of alarms. Multi-level alert aggregation is performed based on the attack attribute matching method, resulting in the effective aggregation of similar alarms. Further, the control plane uses the Markov chain model to generate alert association graphs to obtain the transfer probability between attack types, which is sent down to the security components for linkage defense. The experiments show that this mechanism can effectively achieve multi-level alarm aggregation and alarm information association linkage.

1. Introduction

At present, with the increasing stealth and persistence of network attacks, the challenge of network attack detection is becoming increasingly severe. An intrusion detection system (IDS) can effectively detect network attacks dynamically by analyzing data such as network traffic or data packets [1-2]. However, for complex multi-step attacks with high concealment and persistence, traditional IDSs generally have the problems of large redundancy of alert data and poor data readability, which seriously affect security administrators to quickly identify the attack behavior and attack intent. With the emergence and application of Software Defined Networking technology, it is necessary to effectively aggregate repeated alarms caused by the same or similar attack behaviors between data plane security components and control plane security controllers to achieve linkage defense [3-4].

Scholars at home and abroad have done a lot of fruitful research work in the field of network alarm aggregation. Using IP address attributes, all alarms with interrelated source or destination IP are clustered into one cluster and alarm aggregation is performed within each cluster, but when dealing with complete attack scenarios such as DDoS, all IPs will be interrelated, resulting in alarms being classified in the same cluster and thus rendering the sub-clustering ineffective [5]. In terms of temporal attribute processing, most of the related research methods focus on alarm aggregation within a sliding time window of a certain size, but if the time window is too large, alerts belonging to multiple attacks are aggregated together, if the time window is too small, all alarms triggered by a single attack cannot be aggregated [6]. To address this problem, the traditional sliding window approach is improved. First
storing the current alert information using a longer time window, and then using streaming processing to aggregate alerts within the sliding time window [7]. However, the size of the current window and the sliding window still needs to be determined subjectively. To solve the alarm aggregation problem with large fluctuations in time intervals, the aggregation method based on a fixed time threshold is improved. Each time a new alarm is generated, the relative mean square error of the time interval of the alarm sequence is calculated, and this is used as the coefficient of variation of the alarm aggregation time window, which can realize the dynamic update of the time threshold [8]. However, the dynamic time window calculated by this method is monotonically increasing. Only when the mean square error of the alarm interval tends to zero, the time window will converge, this condition is difficult to satisfy in practice.

To address the shortcomings of existing methods, this paper proposes a multi-level aggregation and association method for alarms based on self-expanding time windows. In terms of time window setting, the time windows of alarms with similar time intervals are dynamically stretched. In terms of IP address attribute processing, alerts with the same port, IP address, and subnet are aggregated together. Further, the alert correlation is performed by calculating the 1st-order state transfer probability among alerts to provide a basis for attack scenario identification.

2. Self-expanding time window setting method

Since different intrusion detection systems generate alerts in different formats and many of the attributes do not contribute much to alert aggregation, it is important to standardize the alert data types before performing alert aggregation. In this paper, we use the most typical existing alert message format IDMEF to describe the network alerts and perform alert aggregation. The alarm data structure is \( \text{Structure} = \{ \text{Time, SrcIP, DstIP, SrcPort, DstPort, AttackType} \} \). \text{Time} is the start time of the attack represented by the alert, \text{SrcIP} and \text{DstIP} are the sources and destination IPs of the attack, \text{SrcPort} and \text{DstPort} are the source and destination ports of the attack, and \text{AttackType} is the attack type.

**Algorithm 1. Self-expanding time window-based division algorithm.**

**Input:** Alert set, the number of alarms in Alert \( N \), and the preset time interval threshold \( I \).

**Output:** The set of alarms divided by the self-expanding time window, stored in the 3-dimensional array \( setB_j \), each page of which is an alarm within a time window.

1) Begin
2) \( p = 1; \)
3) \( k = 1; \)
4) for \( (i = 1; i \leq N; i++) \)
5) \( \text{if} (\text{Alert}_{i+1}. \text{Time} - \text{Alert}_i. \text{Time} \leq I) \) then
6) \( setB_j (p, :, k) = \text{Alert}(i, :) \);
7) \( p++; \)
8) end if
9) else then
10) \( p = 1; \)
11) \( k++; \)
12) \( setB_j (p, :, k) = \text{Alert}(i, :) \);
13) end else
14) end for
15) return \( setB_j \);
16) End

Among the various dimensional attributes of alerts, the \text{Time} attribute is crucial for alert aggregation. For repeated alarms generated by the same attack step, or a large number of similar alarms generated by vulnerability scanning and flooding attacks, the \text{Time} attribute interval between two adjacent alarms is
small, while the overall span of the attack presents uncertainty; therefore, a division method based on self-expanding time window is proposed to automatically expand the time window by using the proximity relationship of alerts. The specific steps of its time window setting are as follows.

Step 1: Data preprocessing. The alarm start time $Time$ property is converted to units of s.

Step 2: Arrange the alert sequence in ascending order by the $Time$ property, and set the start time of the first alert to the beginning of the first time window.

Step 3: If the difference between the $Time$ of the next alert and the $Time$ of the previous one is less than the predefined interval threshold $I$, the latter alert will be included in the current time window. Otherwise, the time of the previous alert is set as the end of the current time window, and the time of the next alarm is set as the starting point of the next time window. The specific algorithms are as algorithm 1.

3. Multi-level aggregation method for network alarms

In the alert aggregation process, the traditional attribute similarity-based method calculates the similarity of $AttackType$, $SrcIP$, $DstIP$, $SrcPort$, and $DstPort$ as part of the overall similarity for fusion calculation. This method is limited because: 1) Substituting $AttackType$ into the similarity calculation will aggregate the alarm records generated by unrelated attacks and reduce the accuracy of aggregation. 2) IP addresses are often matched by bit, which can only reflect the probability of IP belonging to the same subnet to a certain extent, but cannot clearly determine whether they belong to the same subnet. 3) The method of substituting IP and port into the similarity calculation, it is difficult to reflect the actual situation of the attack because it ignores the meaning of the port itself. To solve the above problems, a multi-level aggregation method for alarms is proposed. Use $AttackType$ to cluster alarms to avoid mutual aggregation of alarms of different $AttackType$. In terms of IP address and port processing, the subnet mask is used to clearly divide the subnets, and then subnet-level, host-level and service-level aggregation are performed. These 3 levels of aggregation correspond to the alarms generated by 3 different stages of attacks, and the aggregation results have strong practical significance. The specific aggregation process is as follows: (1) Step 1: Use $AttackType$ to divide the first layer. The alerts with the same $AttackType$, are grouped in one cluster set$A_i$. (2) Step 2: Alert classification based on self-expanding time window. Within the cluster set$A_i$ delineated in Step 1, the set of alarms in the same time window is obtained using Algorithm 1, and the set of alarms in the $j$ time window is denoted as set$B_j$. (3) Step 3: Subnet-level aggregation. In the alarm set set$B_j$ delineated in Step 2, the alarms belonging to the same subnet of $SrcIP$ are first divided into a set using the subnet mask, and the set of alarms in the $k$ subnet is recorded as set$C_k$, and then the alarms in the set are aggregated. The reason for using the subnet to which $SrcIP$ belongs is to reflect the attack carried out by attackers using the same router as a springboard. (4) Step 4: Host-level aggregation. Within the alarm set set$C_k$ delineated in Step 3, the alarms with equal $SrcIP$ attributes are divided into a set, and the set with $SrcIP = IP_p$ is called set$D_p$, and then the alarms in this set are aggregated. This aggregation process can reflect the attacks performed by attackers using the same host as a puppet. (5) Step 5: Service level aggregation. Within the alarm set set$D_p$ delineated in Step 4, the alarms with equal $SrcPort$ attributes are divided into a set, and the set with $SrcPort = Port_q$ is denoted as set$E_q$, and the alarms in the set are aggregated. This aggregation process can reflect an attacker using the same service privilege as a springboard to execute an attack. The overall process of network alarm aggregation based on the idea of multi-level division is shown in Figure 1.

4. Network alert correlation diagram generation method

An attack scenario executed by an attacker can be viewed as multiple logical steps, where the current logical step puts the attacker in a specific authority state, and the next possible attack step is only related to the current authority state. Thus, in the logical step, the multi-step attack satisfies the no aftereffect of the first-order Markov model. Therefore, the first-order Markov property can be used to model the multi-step alarm correlation process.
All alert sets

Attack type matching

The Alert set with AttackType$_1$

... The Alert set with AttackType$_m$

Time division based on self-expanding windows

Alert set within Time Window $W_1$

... Alert set within Time Window $W_n$

Use subnet mask for subnet matching

The alert set IP belongs to the first subnet

... The alert set IP belongs to the $r$th subnet

Subnet-level aggregation

IP address matching

Alert set with IP$_1$

... Alert set with IP$_s$

Host-level aggregation

Service port matching

Alert set for Port$_1$

... Alert set for Port$_t$

Service-level aggregation

Figure 1. Integral flow chart of multistage aggregation for network alerts.

Alert Correlation Graph: The generated alarm correlation graph is an edge-weighted directed graph $G(V, E)$. Where $V$ is the set of alert type nodes, each node represents an attack-type, denoted as type$_i$; $E$ is the set of directed edges $e_{ij}$, $e_{ij}$ represents the edge from node type$_i$ to type$_j$; the weight $p_{ij}$ of $e_{ij}$ denotes the transfer probability from type$_i$ to type$_j$. The value of $p_{ij}$ is determined by the probability of type$_i$ and type$_j$ occurring in sequence. In the alert association graph, the sum of the edge weights with the same node as the starting point must be 1. The alert correlation graph can quantitatively reflect the attacker's attack path and intention. The main indicators to measure the effect of alarm correlation are the false correlation rate $FR$ and the missed correlation rate $MR$. The calculation formula is as follows:

$$FR = \frac{N_{FCOR}}{N_{TCOR}}, MR = \frac{N_{MCOR}}{N_{TCOR}}$$

Where $N_{FCOR}$, $N_{MCOR}$ and $N_{TCOR}$ are the number of false correlations, the number of missed correlations and the total number of correlations, respectively. In addition, the correlation accuracy index $Accuracy$ can comprehensively evaluate the false correlation rate $FR$ and the missed correlation rate $MR$. The larger the value of $Accuracy$, the better the effect of alarm correlation. $Accuracy$ is:

$$Accuracy = \frac{2 \times (1-MR) \times (1-FR)}{(1-MR)+(1-FR)}$$

5. Experiment and analysis
Before alarm aggregation, the intrusion detection system needs to be used to generate alarm information. This experiment uses data from the LLDoS1.0 attack scenario in DARPA 2000, which is a common dataset for intrusion detection as it is used by the U.S. Department of Defense to simulate attacks on its
This attack scenario is a DDoS attack that runs during multiple networks and multiple audit periods. The format of network traffic is tcpdump, taking the top 10% of traffic from the official website of Lincoln Lab. Configure Snort as the most stringent mode, load Snort's rule table and matching engine. The tcpdump file can output a total of 4,399 alarm data. The results of using attack types to divide alerts are shown in Table 1.

Table 1. Division results based on attack types.

| Name of the attack type of the alert | Attack type | Number of alerts |
|-------------------------------------|-------------|-----------------|
| Sadmind_Ping                         | AttackType1 | 1283            |
| Sadmind_Amslverify_Overflow          | AttackType2 | 276             |
| Admind                              | AttackType3 | 87              |
| Rsh                                 | AttackType4 | 65              |
| Mstream_Zombie                      | AttackType5 | 77              |

The self-expanding time window method is used to divide the alarms of each attack type in Table 1, and the results are shown in Table 2, where \( n_{\text{set}} \) is the number of divided time windows, and \( n_{\text{alert max}} \) is the maximum number of alerts contained in the time window.

Table 2. Division results based on self-expanding time windows.

| Attack type  | \( I = 10s \) | \( I = 30s \) | \( I = 90s \) | \( I = 120s \) |
|--------------|---------------|---------------|---------------|---------------|
|              | \( n_{\text{set}} \) | \( n_{\text{alert max}} \) | \( n_{\text{set}} \) | \( n_{\text{alert max}} \) | \( n_{\text{set}} \) | \( n_{\text{alert max}} \) | \( n_{\text{set}} \) | \( n_{\text{alert max}} \) |
| AttackType1  | 101           | 1151          | 92            | 1151          | 67            | 1152          | 58            | 1165          |
| AttackType2  | 213           | 2             | 176           | 5             | 103           | 9             | 86            | 10            |
| AttackType3  | 50            | 7             | 20            | 11            | 12            | 25            | 9             | 42            |
| AttackType4  | 33            | 5             | 27            | 5             | 23            | 6             | 42            | 6             |
| AttackType5  | 64            | 2             | 56            | 3             | 40            | 4             | 35            | 6             |

In order to optimize the parameters of the time interval threshold \( I \), when \( I \) takes a different value, the alerts are aggregated at three levels, and then the changes in the aggregation effect are observed. The alert aggregation effect is measured by the aggregation rate \( Rate \), and the calculation formula is:

\[
Rate = 1 - \frac{N_{\text{aggregation}}}{N_{\text{origin}}}
\]  

(3)

Figure 2. Final aggregation results of attack types. \( N_{\text{aggregation}} \) is the number of alerts generated by aggregation, and \( N_{\text{origin}} \) is the original number of alerts.
Set the subnet mask to 255. 255. 255.0, and the final alert aggregation rate obtained after three-level aggregation is shown in Figure 2. It can be seen from Figure 2 that as the time interval threshold $I$ increases, the final aggregation rate continues to rise. This is because the increase in $I$ will increase the self-expanding time window, which in turn leads to the final division result in each time window. The number of alerts increases, so the aggregation rate increases. However, the increase of the time window will also cause some irrelevant alerts to be aggregated, which will reduce the accuracy of aggregation. Therefore, the inflection point of the connection slope of the aggregation rate is significantly reduced to determine the value of $I$. The $I$ values of 90, 90, 30, 60, and 30s were chosen for the aggregation process of the attack types $AttackType_1$ to $AttackType_5$. After the time interval threshold $I$ is optimized, the aggregation results of each attack type alert are shown in Table 3.

Table 3. Three -stages aggregation results after the optimization of time internal threshold $I$

| Attack type   | Number of alerts generated by aggregation |
|---------------|-------------------------------------------|
|               | Subnet level | Host level | Service level |
| $AttackType_1$| 67           | 78         | 87           |
| $AttackType_2$| 103          | 139        | 152          |
| $AttackType_3$| 20           | 22         | 29           |
| $AttackType_4$| 25           | 27         | 27           |
| $AttackType_5$| 26           | 57         | 57           |

While getting simplified alerts, the aggregation results obtained in this experiment can also reflect the possibility of IP scanning and port scanning in the network to a certain extent. The higher the aggregation rate of host-level aggregation, the higher the possibility of IP scanning in the network; the higher the aggregation rate of port-level aggregation, the higher the possibility of port scanning in the network.

![Figure 3](image1.png)

Figure 3. Correlation graph obtained by the aggregation method in [8].

![Figure 4](image2.png)

Figure 4. Correlation graph obtained by the proposed aggregation method.

To verify the accuracy of the alert aggregation and correlation method proposed in this paper, the aggregation results obtained by the method in this paper and the method in [8] are used to perform alarm correlation, respectively. The results are shown in Figure 3 and Figure 4. The Accuracy of the correlation
graphs shown in Figure 3 is 95.65%, which is higher than the algorithm in [8], 97.94%. Because the alert aggregation method proposed has a finer granularity and a more reasonable time window, so it can effectively aggregate similar alarms and reduce the proportion of false alerts.

6. Conclusion
In this paper, we propose a multi-level alert aggregation method based on software-defined security and linkage defense based on the Markov chain model. A self-expanding strategy is adopted in the setting of the time window, and multi-level alarm aggregation is performed using information such as IP and port. In addition, multi-step alarm correlation analysis is performed on the basis of aggregated alarms to achieve linked defense. The experiments prove that this method has a good alarm aggregation effect and high correlation accuracy.

Acknowledgments
This paper was partly supported by the science and technology project of State Grid Corporation of China: "Research on The Security Protection Technology for Internal and External Boundary of State Grid information network Based on Software Defined Security" (No. 5700-202058191A-0-0-00).

References
[1] Deng F, Song Y, Hu A, et al. Abnormal traffic detection of IoT terminals based on Bloom filter. Proceedings of the ACM Turing Celebration Conference-China. 2019: 1-7.
[2] Gao S, Peng Z, Xiao B, et al. Detection and Mitigation of DoS Attacks in Software Defined Networks. IEEE/ACM Transactions on Networking, 2020, 28(3): 1419-1433.
[3] Yang L, Song Y, Gao S, et al. Griffin: An Ensemble of AutoEncoders for Anomaly Traffic Detection in SDN. GLOBECOM 2020-2020 IEEE Global Communications Conference. IEEE, 2020: 1-6.
[4] Yubo S, Huiwen Y, Wei W U, et al. Joint DDoS detection system based on software-defined networking. Journal of Tsinghua University (Science and Technology), 2019, 59(1): 28-35.
[5] Feng X , Wang D , Huang M , et al. "A Mining Approach for Causal Knowledge in Alert Correlating Based on the Markov Property." Journal of Computer Research and Development, vol. 51, no. 11, pp. 2493-2504, 2014.
[6] Ghasemigol, M., and A. Ghaemi-Bafghi . "E-correlator: an entropy-based alert correlation system." Security & Communication Networks, vol. 8, no. 5, pp. 822–836, 2015.
[7] Qiu Hui, Wang Kun, Yang Haopu. "Network alerts depth information fusion method based on time confrontation." Journal of Computer Applications, vol. 36, no. 2, pp. 499–504, 2016.
[8] Smys S, Basar A, Wang H. Hybrid Intrusion Detection System for Internet of Things (IoT). Journal of ISMAC, 2020, 2(04): 190-199.