A Rule-Based Intrusion Alert Correlation System for Integrated Security Management*

Seong-Ho Lee[1], Hyung-Hyo Lee[2], and Bong-Nam Noh[1]

[1] Department of Computer Science, Chonnam National University, Gwangju, Korea 500-757
shlee2@lsrc.jnu.ac.kr, bongnam@jnu.ac.kr
[2] Division of Information and EC, Wonkwang University, Iksan, Korea, 570-749
hlee@wonkwang.ac.kr

Abstract. As traditional host- and network-based IDSs are to detect a single intrusion based on log data or packet information respectively, they inherently generate a huge number of false alerts due to lack of information on interrelated alarms. In order to reduce the number of false alarms and then detect a real intrusion, a new alert analyzing system is needed. In this paper, we propose a rule-based alert correlation system to reduce the number of false alerts, correlate them, and decide which alerts are parts of the real attack. Our alert correlation system consists of an alert manager, an alert preprocessor, an alert correlator. An alert manager takes charge of storing filtered alerts into our alert database. An alert preprocessor reduces stored alerts to facilitate further correlation analysis. An alert correlator reports global attack plans.

1 Introduction

IDSs have evolved significantly over the past two decades since their inception in the early eighties. The simple IDSs of those early days were based on the use of simple rule-based logic to detect very specific patterns of intrusive behavior or relied on historical activity profiles to confirm legitimate behavior. In contrast, we now have IDSs which use data mining and machine learning techniques to automatically discover what constitutes intrusive behavior and quite sophisticated attack specification languages which allow for the identification of more generalized attack patterns[1].

Attackers usually try to intrude a system after collecting and analyzing the vulnerabilities of the victim. As traditional host- and network-based IDSs are to detect a single intrusion based on log data or packet information respectively, they inherently generate a huge number of false alerts due to lack of information on interrelated alarms. To address this problem, a research to correlate the alerts of several IDSs has emerged recently. These researches include CRIM[2,3] and Hyper-alert Correlation Graph[4,5]. Those correlation methods are based on attack specification with pre- and

* This research was supported by University IT Research Center Project.

M. Bubak et al. (Eds.): ICCS 2004, LNCS 3036, pp. 365–372, 2004.

post-condition of an attack. However, those correlation methods have some disadvantages. First, if the specifications are not correct, those correlation methods do not provide useful results. Second, it is difficult to cover all attack specifications in those correlation methods because new attacks are developed continuously.

So, we propose a rule-based alert correlation system to reduce the number of false alerts, correlate them, and decide which alerts are parts of the real attack. Our alert correlation system is composed of an alert manager, an alert preprocessor, and an alert correlator. An alert manager takes charge of storing filtered alerts into our alert database. An alert preprocessor reduces stored alerts to facilitate further correlation analysis. An alert correlator reports global attack plans.

2 Related Work

2.1 CRIM

CRIM is an IDS cooperation module developed within MIRADOR project[2,3]. This project is initiated by French Defense Agency to build a cooperative and adaptive IDS platforms. As figure 1 in the below, CRIM is composed of 5 functions.

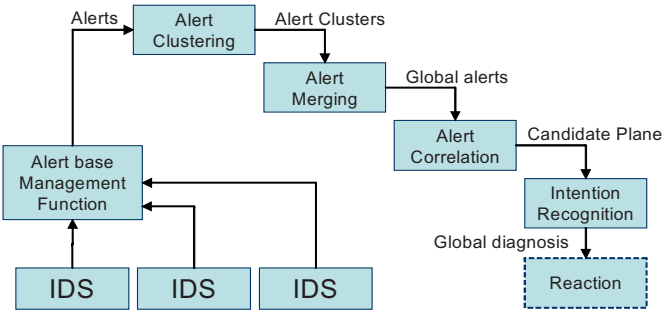

Fig. 1. CRIM architecture

The alert base management function receives the alerts generated by different IDSs and stores them for further analysis by cooperation module. When an attack occurs, the IDS connected to CRIM may generate several alerts for this attack. The clustering function attempts to recognize the alerts that actually correspond to the same occurrence of an attack. These alerts are brought into a cluster. Each cluster is then sent to the alert merging function. For each cluster, this function creates a new alert that is representative of the information contained in the various alerts belonging to this cluster.

Alert correlation function further analyzers the cluster alerts provided as outputs by the merging function. The result of the correlation function is a set of candidate plans that correspond to the intrusion under execution by the intruder. The purpose of the intention recognition function is to extrapolate these candidate plans in order to an-

ticipate the intruder actions. The result of this function is to be used by the reaction function to help the system administrator to choose the best counter measure to be launched to prevent malicious actions performed by the intruder.

2.2 Hyper-alert Correlation Graph

Peng Ning in North Carolina State University analyzes alert correlation visually and constructs attack scenarios using a hyper-alert correlation graph[4,5]. Figure 2 depicts the architecture of an intrusion alert correlator.

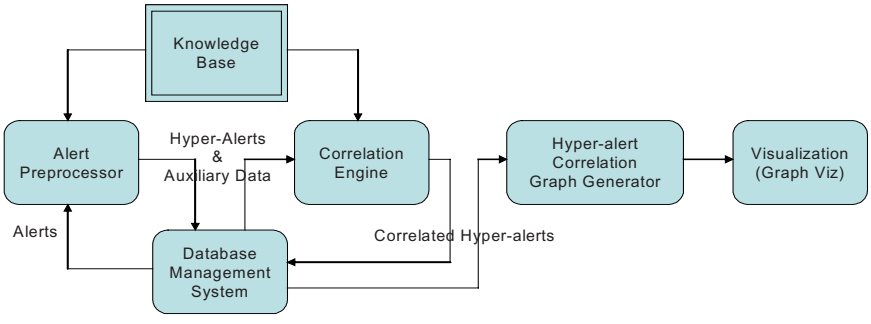

Fig. 2. An architecture of the intrusion alert correlator

It consists of a knowledge base, an alert preprocessor, a correlation engine, a hyper-alert correlation graph generator, and a visualization component. All these components except for the visualization component interact with a DBMS, which provides persistent storage for the intermediate data as well as the correlated alerts.

The knowledge base contains the necessary information about hyper-alert type as well as implication relationships between predicates. In their current implementation, the hyper-alert types and the relationship between predicates are specified in an XML file. When the alert correlator is initialized, it reads the XML file, and then converts and stores the information in the knowledge base.

Their current implementation assumes the alerts provided by IDSs are stored in the database. Using the information in the knowledge base, the alert preprocessor generates hyper-alerts as well as an auxiliary data from the original alerts. The correlation engine then performs the actual correlation task using the hyper-alerts and the auxiliary data. After alert correlation, the hyper-alert correlation graph generator extracts the correlated alerts from the database, and generated the graph files in the format accepted by GraphViz. As the final step of alert correlation, GraphViz is used to visualize the hyper-alert correlation graphs.

3 Rule-Based Intrusion Alert Correlation System

3.1 Alert Manager

The alert manager stores the alerts received from IDSs into alert database. Before sending an alert message, each IDS checks whether an issued alert satisfies filtering rules. In our work, filtering rules are based on the information of protection domain. That is, we select the alerts targeted at interesting systems and store them into alert database. We intend to facilitate further analysis by excluding the alerts targeted at uninteresting systems. We assume that an administrator setups filtering rules in conformance with domain properties. The architecture of an alert manager is shown in figure 3[6]. We use Oracle 9*i* as alert database and Snort 1.8 as IDS.

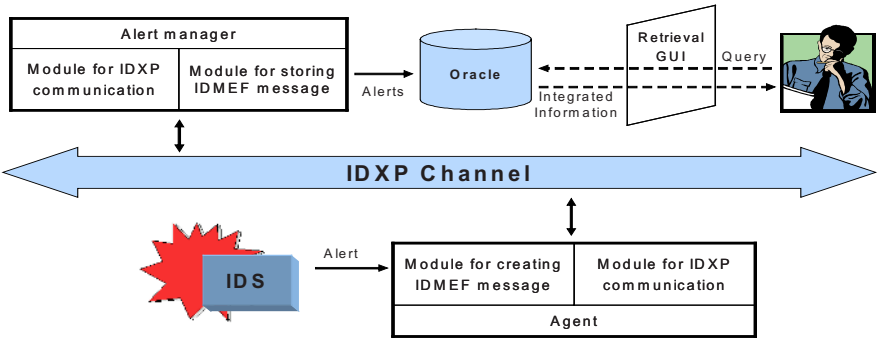

Fig. 3. The architecture of an alert manager

Each IDS sends filtered alerts in an IDMEF format. To facilitate further analysis, we store the information necessary for analysis, not all the information contained in an IDMEF message. This improves the efficiency of storing and retrieval during alert analysis. So, we use only one table for alert storage.

3.2 Alert Preprocessor

3.2.1 Deleting Duplicate Alerts

Duplicate alerts are generated because several identical type of IDSs issue them on seeing the suspicious log data or data packets. So, if the identical type of IDSs are installed on a network, the IDSs generate the alerts with the same source address and port number, the same target address and port number, and the same classification name. We identify these duplicate alerts and delete them except one to facilitate further analysis.

3.2.2 Alert Merging

Several similar alerts can be generated over an identical event. If these alerts are merged into one with minimizing information loss, the next step, correlation analysis will be performed more easily. An algorithm for merging similar alerts is figure 4.

```
mergeSimilarAlerts( ) {
  for each alert
    flag = 0;
    for each Queue in memory
      if (Creation time of this alert - time of Queue > 30 seconds) then
        Create new tuple;
        Store this new tuple into DB;
        Free the Queue;
      endIf
    endFor
    for each Queue in memory
      if (the attributes of this alert and this Queue are equal) then
        Put this alert into this Queue;
        Set Creation time of this alert to the time of this Queue;
        flag = 1;
        break;
      endIf
    endFor
    if flag == 0 then
      Create new Queue;
      Set the attributes of this alert to this Queue;
    endIf
  endFor
}
```

Fig. 4. An algorithm for merging alerts

3.3 Alert Correlator

In an alert correlator, the process for finding correlations among different kinds of alerts such as vulnerability gathering alerts and U2R attack alerts, is conducted. For this, we extract correlation data by the similarity of alert attributes and verify the usefulness of the extracted correlation data through inference rules.

3.3.1 Extraction of Correlation Data

There are two cases in correlation. One is a consecutive attack to the same target and the other is an attack by an identical intruder. To do such a correlation analysis, we extract correlation data by 'Target address' and 'Source address' respectively. These extracted data are chained in time order.

3.3.2 Verification of Extracted Correlation Data

In this section, we evaluate if the extracted correlation data is useful. Since all correlation data are not useful, we identify meaningful correlation data. To achieve this, we apply inference rules to correlation data. Inference rules are described like Prolog predicates. Figure 5 shows an example of inference rules.

```
sameTargetAddress(A,B).
classificationName(A,'WEB-MISC backup access').
classificationName(B,'WEB-CGI scriptalias access').
correlate(A,B) :- sameTargetAddress(A,B),
                  classificationName(A,'WEB-MISC backup access'),
                  classificationName(B,'WEB-CGI scriptalias access').
```

Fig. 5. An example of inference rules

4 Experiments and Discussion

We use ’99 DARPA dataset as experiment data in our work to ensure certified experimental results. We consider the hosts labeled with “Victim” in Simulation Network ’99 as protection domain. In our work, Snort reads tcpdump files in the dataset and issues alerts. Then, the alerts are stored in Oracle database. Table 1 shows an example of duplicate alerts. This example shows the alert of “WEB-IIS *.idc attempt” dated March 8th among the alerts generated from ’99 DARPA dataset.

Table 1. An example of duplicate alerts

No.	Alert ID	Creation time	Source address	Source port	Target address	Target port	Analyzer
1	10	1999-03-08 13:01:59	206.48.44.18	1058	172.16.112.100	80	Inside IDS
2	176632	1999-03-08 13:01:13					Outside IDS

Table 2 shows an example of alerts to be merged. This example is an alert of “WEB-CGI scriptalias access” dated March 8th among the alerts generated from ’99 DARPA dataset. These 241 alerts are caused by ‘Back’ attack, which is a DoS(Denial of Service) attack over an Apache web server. All the attributes except ‘Creation time’ and ‘Source port’ number are equal.

Table 2. An example of alerts to be merged

No.	Alert ID	Creation time	Source address	Source port	Target address	Target port	Analyzer
1	532	1999-03-08 14:39:12	199.174.194.16	1028	172.16.114.50	80	Inside IDS
			…				
241	772	1999-03-08 14:40:11	199.174.194.16	1379	172.16.114.50	80	Inside IDS

Table 3 shows an example of merging the aforementioned alerts into one. When alerts are merged into one, we should decide how we can merge several different ‘Creation time’ into one. We minimize the information loss of Creation time by introducing

range from the first creation time to the last. 'Alert ID' is newly given during merging process. A new 'Alert ID' is the form of 'M + the first Alert ID'. Source port is processed as '#' if several Source port number exist. 'Analyzer' is dropped as it is not utilized at the next step.

Table 3. An example of merging alerts

Alert ID	First Creation time	Last Creation time	Source address	Source port	Target address	Target port
M532	1999-03-08 14:39:12	1999-03-08 14:40:11	199.174.194.16	#	172.16.114.50	80

Table 4 shows an example of correlation by the same target address and the same target port number. An alert of Alert ID 471 is 'WEB-MISC backup access'. This is regarded as an alert by an attack for information gathering about a web server in 172.16.114.50. So, we guess that an attacker investigates if 172.16.114.50 provides a web service and initiates 'Back' attack.

Table 4. An example of correlation by target address and port number

Alert ID	First Creation time	Last Creation time	Source address	Source port	Target address	Target port
471	1999-03-08 14:26:54	#	197.182.91.233	6266	172.16.114.50	80
M532	1999-03-08 14:39:12	1999-03-08 14:40:11	199.174.194.16	#	172.16.114.50	80

In our correlation experiments, we find out various global attack plans such as the scan of an entire network and a large-scale attack. However, our correlation technique should be refined elaborately. To achieve this, we will consider the logs of Secure OS and HIDS(Host-based IDS).

The advantages of our research are as follows. First, we facilitated correlation analysis by reducing alerts to analyze minimizing information loss. Second, we decreased the possibility of false correlation analysis by wrong attack specification. We use the classification name of an alert and if IDSs provide the correct classification name of a single attack, we can get more reliable correlation results. In the contrary, our research has some shortcomings. As our correlation technique is based on the classification name of an alert, it has low efficiency. In addition to them, our correlation technique should be generalized to be applied to IDSs other than Snort.

5 Conclusion and Future Work

In this paper, we presented a rule-based alert correlation system. Our alert correlation system consists of an alert manager, an alert preprocessor, an alert correlator. An alert manager takes charge of storing filtered alerts into our alert database. Before alert

correlation analysis, stored alerts go through preprocessing. An alert preprocessor includes a module of deleting duplicate alerts and an alert merging module. This preprocessing reduces stored alerts to facilitate further correlation analysis. We found out that filtering and preprocessing resulted in reducing a number of alerts through experiments. After that, we did correlation analysis over preprocessed alerts. An alert correlator includes an module for extracting correlation data and an module for verifying the extracted data. In the result, the alert correlator reported global attack plans such as the scan of the entire network and a large-scale attack.

In the future, we will refine our correlation technique. And, we will correlate the logs of HIDS and secure OS and get more reliable experiment results. In addition, we would like to establish a reaction strategy by maintaining an attacker list and a main victim list.

References

1. N. Carey, A. Clark, G. Mohay, "IDS Interoperability and Correlation Using IDMEF and Commodity Systems," ICICS 2002, LNCS 2513, pp. 252-264, 2002
2. F. Cuppens, "Managing Alerts in a Multi-Intrusion Detection Environment," In Proc. Of Annual Computer Security Applications Conference (ACSAC 2001), Dec. 10-14, 2001, New Orleans, Louisiana
3. F. Cuppens, A. Miege, "Alert Correlation in a Cooperative Intrusion Detection Framework," In Proc. Of the 2002 IEEE Symposium on Security and Privacy, May 2002
4. P. Ning, Y. Cui, D. S. Reeves, "Constructing Attack Scenarios through Correlation of Intrusion Alerts," 9th ACM conference on computer and communications security, pp. 245-254, Nov 18-22, 2002
5. P. Ning, Y. Cui, D. S. Reeves, "Analyzing Intensive Intrusion Alerts via Correlation," In Proc. Of the 5th Int'l Symposium on Recent Advances in Intrusion Detection (RAID 2002), Oct 2002
6. S. H. Lee, Y. C. Park, H. H. Lee, B. N. Noh, "The Construction of the Testbed for the Integrated Intrusion Detection Management System," In Proc. Of 19th KIPS Spring Conference, Vol. 10, No. 1, pp. 1969-1972, May 16-17, 2003
7. H. Debar, M. Dacier, A. Wespi, "Research Report: A Revised Taxonomy for Intrusion Detection Systems," Annales des telecommunications, 55(7-8), pp. 361-378, Jul-Aug 1997
8. T. Buchheim, M. Erlinger, B. Feinsteing, G. Matthews, R. Pollock, J. Bester, A. Walther, "Implementing the Intrusion Detection Exchange Protocol," In Proc. Of 17th Annual Computer Security Applications Conference (ACSAC 2001), New Orleans, Louisiana
9. H. Debar, A. Wespi, "Aggregation and Correlation of Intrusion Detection Alerts," In Proc. Of the 4th Int'l Symposium on Recent Advances in Intrusion Detection (RAID 2001), LNCS 2212, pp. 85-103, 2001