Scenario Discovery Using Abstracted Correlation Graph

Safaa O. Al-Mamory
School of Computer Science,
Harbin Institute of technology,
Harbin, China
Safaa_vb@yahoo.com

Hong Li Zhang
School of Computer Science,
Harbin Institute of technology,
Harbin, China
zhl@pact518.hit.edu.cn

Abstract

Intrusion alert correlation techniques correlate alerts into meaningful groups or attack scenarios for the ease to understand by human analysts. These correlation techniques have different strengths and limitations. However, all of them depend heavily on the underlying network intrusion detection systems (NIDSs) and perform poorly when the NIDSs miss critical attacks. In this paper, a system was proposed to represents a set of alerts as subattacks. Then correlates these subattacks and generates abstracted correlation graphs (CGs) which reflect attack scenarios. It also represents attack scenarios by classes of alerts instead of alerts themselves to reduce the rules required and to detect new variations of attacks. The experiments were conducted using Snort as NIDS with different datasets which contain multistep attacks. The resulted CGs imply that our method can correlate related alerts, uncover the attack strategies, and can detect new variations of attacks.

1. Introduction

When the NIDS detects a set of attacks, it will generate many alerts that refer to security breaches. Unfortunately, the NIDS cannot deduce anything from these separated attacks. So, alert correlation is an important solution to link separated attacks, to give alerts another meaning, and to infer attack scenarios.

Alert correlation and analysis is a critical task in security management. Recently, several techniques and approaches have been proposed to correlate and analyze security alerts, most of them focus on the aggregation and analysis of raw security alerts, and build attack scenarios.

An interesting method is the work of Ning *et al*. [1]. They were a proposed alert correlation model based on the observation that most intrusions consist of many stages, with the early stages preparing for the later ones. They were collected alerts from NIDS, correlated off-line, and tried to draw a big picture (through *CG*s) of what happens in the network. However, there are some shortcomings associated with this method:

- The graph explosion problem that occurs in the generated *CG*s makes the resulted graphs complex and hard to understand.
- Huge number of rules used to draw these graphs representing alerts' prerequisites and consequences.
- The affects of the missed attacks by NIDS resulted graphs that yield separated *CG*s.

To address the disadvantages of this method, we have proposed a system that can address these problems. The proposed system contains four components: alert prioritization, alert classification, alert aggregation, and the correlation graph generation. Also Breadth-First search algorithm was used to find the related attacks. The resulted *CG*s show that the proposed system can correlate related alerts, uncover the attack strategies, and can effectively simplify the analysis of large amounts of alerts.

The contribution of this paper is three folds: First, generate compressed and easy to understand *CG*s which reflects attack scenarios. Second, represent the scenarios by alert classes instead of alerts themselves which reduce the required rules. Finally, the ability to pass the missed attacks, by NIDS, that are located in the middle of the scenarios.

The rest of this paper is organized as follows: Section 2 presents related works. Section 3 states the proposed system components in detail. Section 4 presents our experiments and results. The discussion of results is presented in Section 5 and section 6 concludes this paper.

2. Related Work

Many researchers propose systems that aim to build attack scenarios depending on various techniques. Dain

et al. [2] use data mining approach to combine the alerts into scenarios in real time. The probabilistic alert correlation [3] based on the similarities between alerts to correlate them. Measures are defined to evaluate the degree of similarity between two alarms. Qin *et al*. [4] present an alert correlation system combining a Bayesian correlation with a statistical correlation using a time series-based causal analysis algorithm.

The work of Ning *et al*. [1] generates *CG*s depending on pre and post-conditions of individual alerts. They propose an alert correlation model based on the inherent observation that most intrusions consist of many stages, with the early stages preparing for the later ones. The correlation model is built upon two aspects of intrusions that are, *Prerequisites* (the necessary conditions for an intrusion to be successful) and *Consequences* (the possible outcome of an intrusion). With knowledge of prerequisites and consequences, the correlation model can correlate related alerts by finding causal relationships between them. They used *hyper alert* correlation graphs to visually represent the alerts, where each node represents a hyper alert and the edges represents *prepares for* relation.

3. The Proposed System

The proposed system in this paper is composed of four parts: Prioritization, Classification, Aggregation, and Correlation Graph Generation. The proposed system as shown in Fig. 1 takes as input the raw alerts that are issued from NIDS then enhances alerts quality using alert prioritization. After that the alert classification is performed using alert abstraction. The third component tries to merge the similar classes produced from the classification process. Finally, the alerts correlation happens to produce *CG*s.

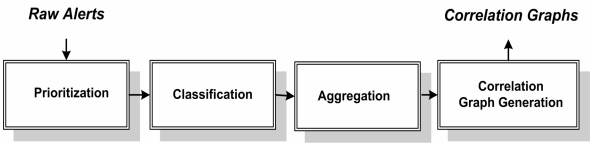

Figure 1. The proposed system architecture

The aim of alert classification and aggregation components is to handle the elementary alerts produced by NIDS due to a certain attack, cluster them to produce a higher-level alert message, called *meta-alert* (*MA*) that summarizes the characteristics of the detected attacks.

A meta-alert is characterized by: alert class, which is the generalized alert type or subattack name, the source IP address, the target IP address, time information, and a reference to the log file of the NIDS so that further investigation on the results can be carried out.

3.1. Prioritization

Alert prioritization is performed to assess the relative importance of alerts generated by the sensors. This method has to take into account the security policy and the security requirements of the site where the correlation system is deployed [5]. Therefore, prioritizing of alerts aids in substantial reduction of alerts volume.

3.2. Classification and Aggregation

In this paper the classification was performed by using alerts abstraction. The alert classification scheme is designed to categorize alerts into groups that most effectively indicate their stage in a multistage attack. An alert can be part of multiple classes. Each class has its name that indicates the general category, in other words any alert can belong to the following classes: enumeration, host probe, service probe, service compromise, user access, root or administrator access, dos, system compromise, sensitive data gathering, active communication remote, or Trojan activity. Alerts descriptions were taken from the Snort signature database [6].

Aggregation component will merge the similar alerts resulted from previous component within specified time window. The clustering we refer to in this paper was performed by the classification followed by aggregation.

3.3. Correlation Graph Generation

This is the last stage of the proposed system that contains alert correlation and *CG* generation. In this paper, we have proposed a technique that builds simple *CG*s using alert clustering and correlation. The correlation depends on the *Relation Matrix* (*RM*) that contains the similarities between every two *MA*s, and few predefined rules. There are three measurements that have been used in this paper listed below. The three measurements are numerical values.

- Msr_1: How much Similar_SourceIP(MA_1,MA_2)? This feature computes the common similar bits of two IP addresses from the left. Then the result divided by 32.

- Msr_2: How much Similar_TargetIP(MA_1,MA_2)? The value of this feature is computed such as the previous one.
- Msr_3: TargetIP(MA_1)=SourceIP(MA_2)? This feature is necessary because sometimes the attacker use one victim as a step stone to compromise another victim.

It is very important to find the strength between any two *MA*s to correlate them together or not. Computation of this strength depends on the similarity of measurements. The correlation strength will be computed for all the *MA*s which assumed to be in time order. We suggest representing the correlation strength of any related *MA*s in a triangle matrix (i.e. *RM*). In this matrix, $V_{(1,3)}$, for example, means the relation between MA_1 and MA_3 and also MA_1 precedes MA_3.

Equation (1) was used to compute the correlation strength value between any two *MA*s in *RM*. The *IsSuccessor(j,i)* variable in (1) is a Boolean variable that determine if MA_j can occur after MA_i. If so, the similarities will be computed otherwise the value is zero. The Msr_k variable in (1) is the k^{th} measurement's value.

$$V_{(i,j)} = \begin{cases} \sum_{k=1}^{3} Msr_k(MA_i, MA_j) & IsSuccessor(j,i) = True \\ 0 & Otherwise \end{cases} \quad (1)$$

Correlation graphs can be represented by nodes (i.e. the subattacks) and arcs (i.e. the relation between two subattacks). The direction of the arcs specifies the temporal relation. The subattack was represented here by *MA*.

Definition1. Given *MA* contains one or more alerts. Let S_{MA} be the set of *MA*s and let *t(MA)* is the earlier time in which *MA* has occurred. Thus it can be said that *Class(MA)* is the class of an *MA* that represents a subattack within a scenario. In a multistep attack, the early step of attack prepares for later ones. So we can build a relation *Prepare-for*(MA_1,MA_2) if *Class*(MA_1) prepare for *Class*(MA_2) in the scenario and $t(MA_1) \leq t(MA_2)$. For any given two *MAs* α and $\beta \in S_{MA}$, α is called a parent of β and β is called a child of α if the relation *Prepare-for*(α, β) is satisfied. It should be noted that any child can have more than one parent. ■

Applying the Breadth-First search algorithm depends on the parent-child relationship that has assumed in definition 1. The pseudo code of the proposed algorithm that builds *CG* is shown in Fig. 2.

Any new *MA* is not always linked to the latest *MA* in the generated scenario. Instead, it is connected to the *MAs* with which it has a high correlation value in *RM*. So, the representation is useful for inference with multiple goals of attackers. The intention of using graph representation for attack scenarios is to give the security analyst an intrinsic view of the network status.

The *Attach_Threshold* is used in the proposed algorithm to control the membership of one *MA* to the scenarios. When this threshold is set to be small, the resulted graphs will be noisy, whereas when its value is set to be high many real attacks do not join to its *CG*s.

The generated *CG*s are concise and abstract, so to show more details about the low-level alerts we can drill-down using the references to the log file that exist in *MA*s.

```
Input:  Stream of meta-alerts in time order
Output: Correlation Graphs
Begin
1:    Initialize Queue Q and Graph G;
2:    Do until All MA∈ RM visited {
3:       Get new unvisited MA and put it in Q and G;
4:       While Q not empty{
5:          ActiveMA←Q.dequeue;
6:          S← set of ActiveMA's children and not visited yet;
7:          Q.enqueue all S elements;
8:          Set ActiveMA as visited;
9:          G←G ∪ ActiveMA (Connect ActiveMA to
            appropriate MAs in G using Attach_Threshold);
10:      }
11:      Output G as detected Correlation graph;
12:   }
End
```

Figure 2. Pseudo-code of graph generation algorithm

4. Experiments and Results

In this section, we report the experiments we performed to evaluate the effectiveness of the proposed method in constructing attack scenarios. The experiments were conducted with the 2000 DARPA datasets [7] and Defcon 8 datasets [8]. Snort (Version 2.6.1) [9] was used in this experiments because it is a freely available NIDS.

4.1. DARPA Dataset Experiment

The 2000 DARPA intrusion detection scenario specific datasets include LLDOS 1.0 and LLDOS 2.0.2 [7]. LLDOS 1.0 contains a series of attacks in which the attacker probes the network, probes the active hosts for Solaris Sadmind, breaks into these hosts with the Solaris Sadmind vulnerability, installs the Msream DDos software on the three compromised hosts, and actually launches a DDos attack against an off-site server.

Each dataset includes the network traffic collected from both the DMZ and the inside part of the evaluation network. We have performed four sets of

experiments, each with either the DMZ or the inside network traffic of one dataset. The *CG*s discovered from the inside network traffic in LLDOS 1.0 were shown in Fig. 3. Each node in Fig. 3 represents a *MA*. The text inside the node is the class of the *MA* followed by the *MA* ID. There are 15 *MA*s in this graph and there are no false alerts with it. Fig. 3 contains three subgraphs from one attacker (the source IP address is 202.77.162.213) to three victims, i.e. destination IP addresses 172.16.112.10, 172.16.115.20, and 172.16.112.50.

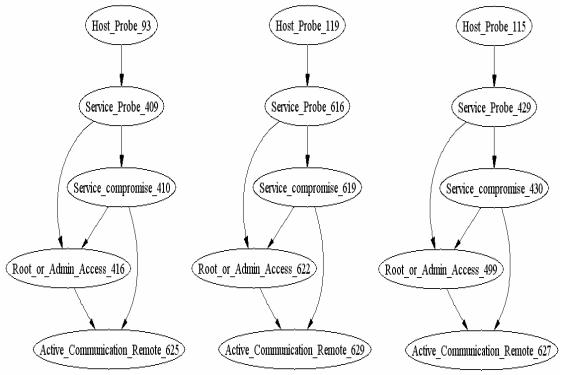

Figure 3. The CGs discovered in the LLDOS 1.0 inside zone

As shown in Fig. 3, we got three disjoint correlation graphs due to Snort's fails to report some parts of the scenario, i.e. communication of the DDoS Trojans on the compromised hosts and also DDoS attack.

To test the effectiveness of the proposed system, we have used the measures of completeness and soundness defined in [1]. The soundness measurement (*Rs*) evaluates the rate of true alerts that appear in *CG*. The completeness measure (*Rc*) looks for missing true alerts from *CG*. The values of these measures can be computed from (2). The results of two measures are shown in Table 1.

$$Rc = \frac{\#\,correctly\ correlated\ alerts}{\#\,related\ alerts}, Rs = \frac{\#\,correctly\ correlated\ alerts}{\#\,correlated\ alerts} \quad (2)$$

It should be noted that the missed alerts by NIDS degrade the system effectiveness and this was the situation in our experiment, where Snort has missed many real alerts that affect the completeness measure results as shown in Table 1. Also the experiments were produced accepted values for the soundness measure except LLDOS 2.0.2 inside zone because there are ten incorrectly correlated alerts. This occurred because the *Attach_Threshold* value has reduced to catch all the real alerts.

Table 1 Correlation effectiveness of the proposed system

	LLDOS 1.0		LLDOS 2.0.2	
	DMZ	**Inside**	**DMZ**	**Inside**
# Correlated Alerts	*122*	*60*	*6*	*24*
# Correctly Correlated Alerts	*122*	*60*	*6*	*14*
# Incorrectly Correlated Alerts	*0*	*0*	*0*	*10*
# Related Alerts	*136*	*72*	*6*	*16*
# Missed Alerts By Snort	*14*	*12*	*0*	*2*
Completeness Measure Rc	*89.7%*	*83.3%*	*100%*	*87.5%*
Soundness Measure Rs	*100%*	*100%*	*100%*	*58.3%*

4.2. Def Con 8 Dataset Experiment

As another case study, we applied our method on the Def Con 8 Capture The Flag (CTF) datasets [8]. Unfortunately, due to the nature of the Def Con 8 CTF datasets, we did not have any information about its scenarios. Thus, we only analyze the resulted *CG*s and discuss some of its scenarios.

The resulted alerts from snort (after prioritization) were 1,847,745 raw alerts. Scanning related alerts divided into two groups: host probe and service probe. Host probe alerts account for 1,255,881 alerts (67.9 %) whereas service probe alerts account for 425,398 alerts (23%). Other alerts include service compromise, DDos, Dos, etc, account for 166,466 alerts (9.1%). The remaining *MA*s after clustering are 170,404.

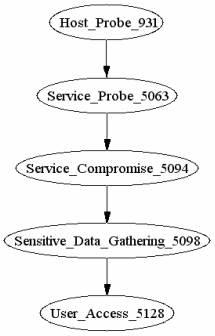

Figure 4. The CG generated from Def Con 8 dataset

There are many scenarios in this dataset, Fig. 4 shows one of the generated *CG*s that contains one scenarios. The attacker in this *CG* is 10.20.11.191 and the victim is 10.20.1.8. In this scenario, the attacker scans the host to see if it is a live and this is appeared as host probe. After that he/she scans victim's ports to gather the active services and this is appeared as service probe. Then he/she exploits the web service by buffer overflow attack (i.e. *CGI applications*) this is abstracted in this figure as service compromise. After that some user account information were stolen, represented in the figure as sensitive data gathering.

Finally the attacker used this information to log as a normal user.

5. Discussion

From the literature, Ning *et al*. [1] have proposed a correlation method to extract attack strategies from intrusion alerts, which is similar to this work. The experimental results on the DARPA 2000 dataset and Def Con 8 show that both approaches produce similar attack strategies. However, our approach is different from theirs in three folds. First, they have used pre-/post-conditions to correlate alerts whereas we use scenario based approach. Second, we use few rules to produce these results whereas they have defined a large number of rules in order to correlate the alerts. And finally, we have represented the alerts by classes which reduce the required rules. By using alert's classes to represent scenario rules, there is ability to detect new variations of attacks. In other words, our system is more adaptive to the emerging of new attacks because we focus on alerts classes instead of alerts themselves.

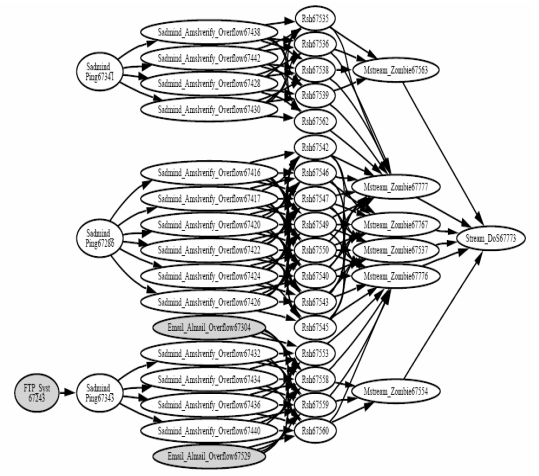

Figure 5. Ning et al. CG for LLDOS 1.0 inside zone

The proposed method provides a high-level representation of correlated alerts that reveals the causal relationships between them. As we have seen in Section 4, *CG*s generated by our implementation clearly show the strategies behind these attacks. One advantage of our method is the compressing of the resulted *CG*s. The *CG* shown in Fig. 5 discovers the attacks in DARPA LLDOS1.0 (inside zone) drawn by Ning *et al*. [1]. They use Real Secure as NIDS which did not miss the last two stages of the scenario like Snort. It can be noted in Fig. 3 (comparing with Fig. 5) the simplicity and compression of CG.

6. Conclusion

This paper presented a systematic method for constructing attack scenarios (or *CG*s) through alert correlation, using predefined attack scenarios. The proposed system is composed of four components which filter out the unnecessary alerts, cluster the alerts as subattacks, and then generate *CG*s using a small set of rules and Breadth-First search algorithm. The generated *CG*s by correlation engine correctly reflect the multistage attacks in the dataset.

7. References

[1] Ning P., Cui Y., Reeves D. S. and Xu D., "Techniques and tools for analyzing intrusion alerts", ACM Transactions on Information and System Security, 7(2) , 2004, pp. 1-44,.

[2] Dain O.M. and Cunningham R. K, "Fusing a heterogeneous alert stream into scenarios", Proceedings: the 2001 ACM Workshop on Data Mining for Security Applications, 2001, pp. 1-13.

[3] Valdes A. and Skinner K., "Probabilistic alert correlation", Proceedings: Recent Advances in Intrusion Detection, LNCS 2212, 2001, pp. 54-68.

[4] Qin X. and Lee W., "Statistical causality analysis of INFOSEC alert data", Proceedings: the 6th International Symposium on Recent Advances in Intrusion Detection (RAID 2003), Pittsburgh, PA, Sep 2003.

[5] Valeur F., Vigna G., Kruegel C. and Kemmerer R. A., "Comprehensive approach to intrusion detection alert correlation," IEEE Transactions on Dependable and Secure Computing 1 (3), 2004, pp. 146-169.

[6] Snort signature database, http://www.snort.org/pub-bin/sigs.cgi.

[7] MIT Lincoln Lab., 2000 DARPA intrusion detection scenario specific datasets. http://www.ll.mit.edu/IST/ideval/-data/2000/2000_data _index.html.

[8] Def Con captures the flag (ctf) contest, http://cctf.shmoo.com/data/cctf-defcon8/, 2000.

[9] SNORT, http://www.snort.org/, 2005.