Knowledge-Based Model to Represent Security Information and Reason About Multi-stage Attacks

Faeiz M. Alserhani(✉)

Department of Computer Engineering and Networks, College of Computer and Information Sciences, Sakaka Aljouf, Saudi Arabia
fmserhani@ju.edu.sa

Abstract. In an intrusion detection context, none of the main detection approaches (signature-based and anomaly-based) are fully satisfactory. False positives and false negatives are the major limitations of such systems. The generated alerts are elementary and in huge numbers. Hence, alert correlation techniques are used to provide a complementary analysis to link elementary alerts and provide a more global intrusion view. It has been widely recognised that real cyber attacks consist of phases that are temporally ordered and logically connected.

In this paper we present an improved knowledge-based causal alert correlation model. The correlation process is essentially modularized based on an extension of the properties and characteristics of the "*requires/provides"* model. The description of the knowledge base modeling is introduced consisting of attacks classes, vulnerabilities, and alerts generated by security tools. The proposed system is evaluated to detect simulated and real multi-stage attacks and it showes efficient capability to correlate the attacker behavior.

Keywords: Intrusion detection systems · Alert correlation · Multi-stage attack

1 Introduction

Malicious attacks by intruders and hackers exploit flaws and weaknesses in the deployed systems. This is done by several sophisticated techniques and cannot be prevented by traditional measures. Hackers are shifting their focus from looking for fame and advertised attacks to profit-oriented activities. The current trends in cyber attacks are hidden, slow-and-low, and coordinated. NIDS are considered to be important security tools to defend against such threats. The effectiveness of any NIDS depends on its ability to recognize different variations of cyber attacks. The current implementation of intrusion detection systems (commercial and open-source) is employing signature-based detection in addition to a few simple techniques for statistical analysis. The main task of signature-based systems is to inspect the network traffic and perform pattern matching to detect attacks and generate alerts. A huge number of alerts are generated every day stressing the administrator; this may oversight an actual threat. Quality of these alerts is debatable particularly if the majority is false positives. For this reason, high-level and real-time analysis techniques are needed. This can be achieved by discovering the logical connections between the isolated alerts. It has been practically identified that

A. Persson and J. Stirna (Eds.): CAiSE 2015 Workshops, LNBIP 215, pp. 482–494, 2015.
DOI: 10.1007/978-3-319-19243-7_44

most of attacker activities consist of multiple steps (attack scenario) and occur in a certain time (attack window). Identification of such strategy can lead to the recognition of attack intensions and also prediction of unknown attacks.

In this paper we have extended our previous work in [1, 2] to describe the details of the proposed model design. The underlying principle of the model based on *provides/requires* model is defined precisely giving some clarification examples. The discussion has been supported by the evaluation using different metrics. The rest of this paper is organized as follows: Sect. 2 explains the concepts of the proposed model. In Sect. 3, we present a description of the knowledge-based modeling and its related components. Section 4 gives the experimental results and then we conclude in Sect. 5.

2 Model Design

The *requires/provides* model is a general attack model that has been proposed by [3] and is inspired from network management systems to deal with network faults. A cyber attack is described according to two components: (1) capabilities, and (2) concepts. The idea behind this model is that multi-stage intrusions consist of a sequence of steps performed by an attacker, and that the later steps are prepared by the early ones. The target system information collected from scanning or port mapping are advantages acquired and used in order to choose which exploit can be successful. Attacks are modelled in terms of abstract concepts and each concept requires certain capabilities (conditions) to occur and provides others to be used by another concept. Capabilities are defined as general descriptions of the conditions required or provided by each stage of the intrusion i.e. the system state that must be satisfied in order to launch an attack. For instance, a successful Trojan injection requires particular services to be running in the target system and the presence of certain vulnerabilities.

Formally, capabilities are a higher level of intrusion abstraction that specifies the system state after each attack attempt. The attacker uses the capabilities acquired through some of its early actions to generate certain new capabilities. The system state is incorporated in attack scenarios if instances of concepts have matched "*required*" and "*provided*" conditions.

The capability model proposed by [4] is also based on a *requires/provides* model for logical alert correlation, though the authors used different properties of capabilities. An attack model was presented to build blocks of capabilities in a multi-layer fashion and with more expressive definition. References [5, 6] have employed a *requires/provides* model using the concept of predicates, which are similar to capabilities.

Both models mentioned above are reasoning models that aim to discover the causal relationships between elementary alerts. Attacker states are abstracted to describe the gained privileges and what level of access is obtained. Moreover, the system states are modelled into a higher level of abstraction to specify the impact of the attack. Relationships between these states are defined to generate rules that determine the dependency between alerts.

The *requires/provides* model has been selected because it fits our purpose to correlate alerts in the same intrusion. It has some advantages over other models:

i. Ability to uncover the causal relationships between alerts and it is not restricted to known attack scenarios.
ii. Ability to characterize complex scenarios or to generalize to unknown attacks.
iii. Attack is represented as a set of capabilities that provides support for the abstract attack concepts.
iv. Flexibility and extensibility as the abstract attack concepts are defined locally.
v. It does not require a priori knowledge of a particular scenario.
vi. Numerous attacks can be described implicitly and an unknown attack can be defined by generalisation.

Our approach is a variation of the *requires/provides* model, but differs in the following aspects:

- Different definitions for capabilities and concepts are employed to overcome the limitations expressed in other approaches. The work in [3] used a very detailed specification language called JIGSAW to describe attack scenarios. A complete satisfaction of "*required*" and "*provided*" conditions is necessary to correlate two alerts, which will fail in case of broken scenarios. However, the authors in [5] have adopted a partial satisfaction technique which is also implemented into our framework. The main concern with their approach is the high rate of false positives, and the possibility of a huge graph being created. We have managed to overcome this limitation by using certain techniques: hierarchical multi-layer capabilities, accumulated aggregation, alert verification and alert maintenance.
- A near real-time processing approach for correlation, aggregation and event generation. The security officer can monitor the attack progress which is displayed as an intrusion graph. An event is triggered once at the minimum of two alerts being correlated, and any additional related alert based on its attributes will join the same event.
- Online and offline graph reduction algorithms during the correlation process in addition to alert aggregation in order to provide a smaller manageable graph.
- We have modelled IDS signatures as abstracted attack concepts instead of defining new concepts locally. In *requires/provides* models, IDS signatures are considered complementary external concepts.
- Separation of the concepts and their capabilities from other dynamic information. Two different types of capabilities have been used: internal and external. The first type denotes abstract attack modeling consisting of IDS signatures and associated capabilities. The second type refers to dynamic details, including system configuration, services and vulnerabilities. This provides more flexibility to the model whilst at the same time allowing utilization of other knowledge resources.
- Capabilities' modeling has been made using a hierarchical methodology based on attack classes and inheritance between these classes.

Our approach is based on the assumption that the attack scenario consists of a sequence of related actions and that early stages can incorporate later ones. The link between these stages is determined using five factors:

i. Temporal relationships (e.g. alert timestamps).
ii. Spatial relationships (e.g. source IP addresses, destination IP addresses and port numbers).
iii. Pre- and post-conditions of each attack.
iv. Vulnerability assessment of the target system.
v. Target system configuration.

Capabilities are formalized in terms of pre- and post-conditions by grouping conditions that share similar characteristics into a broad definition. Knowledge about elementary alerts is mapped to instantiate the attacker and the system states according to their temporal characteristics:

– *Pre-conditions:* are logical capabilities that characterize the system state to be satisfied in order to launch an attack. These capabilities are derived from the attack description. A hierarchical approach is adopted based on an attack classification to provide coarse-grained definitions of different alerts related to the same behaviour.
– *Post-conditions:* are logical capabilities that characterize the system state after the attack succeeds. In other words, specifications of the effects of intrusions on the system, such as the knowledge gained and the access level of the attacker. Moreover, attack classification incorporates the definitions of these capabilities in a hierarchical manner.

To formulize the capability sets as pre- and post-conditions of higher quality, certain requirements must be satisfied:

1- Capabilities must be expressive in order to achieve a true logical relationship.
2- Avoidance of ambiguity in defining capabilities.
3- Use of multi-layers of abstraction to achieve scalability.
4- Reduction of the number of elements in the capability sets without affecting attack coverage.
5- Inference rules should be separated from the capability set.
6- The set should also be constant and independent of variable information such as vulnerability and system-configuration knowledge.

Hence, capabilities are formulized based on two criteria:
(a) Level of abstraction

i. Generic capabilities which illustrate a broad aspect of a certain attack, such as access, local access and remote access.
ii. Capabilities which illustrate a lower level of attack abstraction, but not a specific one, such as server buffer overflow or client upload file.
iii. Specific capabilities for each single alert in IDSs, such as TFTP Get.

(b) Properties of the system and the attacker state

i. Access level of the attacker (remote, local, user or administrator).
ii. Impact of the intrusion upon the victim machine, such as DOS and implementation of the system commands.
iii. Knowledge gained by the attacker, such as disclosure of host or of service.

The elements in the two criteria above are mutually inclusive; for instance, *disclosure of host* is considered as a generic capability and at the same time is a system state description. In addition, attack classification, which will be presented in the next section, is also involved in defining capabilities.

Examples: generic capabilities are mainly a description of the intrusion's general objective, such as:

- Disclosure of host
- Disclosure of running service
- Disclosure of port number
- Access
- Read or write files

However, a *buffer overflow* attack is a general attack that can target the server, the Web server and the client, and the required and provided conditions are not the same for each category. The capability *client access attempt* is a specific capability for client attacks, because some attacks are client specific, such as ActiveX attacks. Snort [7] documentation contains a description for each signature, including the attack class type, the affected system, and the impact of the attack. This information is valuable in defining attack capabilities if other sources of intrusion analysis are considered.

3 Knowledge-Base Modeling

Two knowledge bases are used, one for attack concepts and the other for vulnerability details. In the attack knowledge base, IDS signatures (e.g. Snort) are modelled to the attack abstractions and their defined capabilities. The knowledge library specifies the relationship between low-level alerts and the attack abstraction. Thus, a knowledge base can be considered a broad template and each element can be instantiated to instances of specific conditions. A generalization mechanism has been used to specify a higher level of specification of attack concepts and capabilities.

The proposed model for the attack knowledge base consists of three sets:

(1) Capability *C*: This specifies a higher level of abstraction of the "*required*" and "*provided*" conditions of the intrusion model. Intrusion attempts are expressed in terms of a set of "*required*" or "*provided*" conditions, and vulnerability constraints of a given alert where:

- Required conditions *R* are a set of pre-conditions specified in the form of capabilities with variable arguments.
- Provided conditions *P* is a set of post-conditions specified in the form of capabilities with variable arguments.
- Vulnerability *V* is a description of the state of the target host or network with variable arguments.

(2) Attack concept *AC* specifies the constructor of a given attack and its related capabilities. "*required*" and "*provided*" conditions for each attack are coded in a language of capabilities.
(3) Arguments $[r_1, r_2, \ldots r_i] \rightarrow r$ are a set of associated attributes such as source IP addresses, destination IP addresses and port numbers.

Definition 1. Attack concept *AC* is an abstraction of elementary alerts generated by the IDS, defined by a set of arguments, required conditions and provided conditions.

Definition 2. *An* attack instance a_i is defined as a set of instances of attack concept *AC* by substituting the associated values in arguments tuple considering the time constraints (start-time and end-time).

Definition 3. Given an attack concept *AC*, the *R(AC), P(AC)* and *V(AC)* sets are the sets of all capabilities C. Given an attack instance *a*, the *R(a), P(a)* and *V(a)* sets are the capabilities by mapping the values to the corresponding arguments in *AC* considering the time constraints.

3.1 Attack Classification

Several attempts have been made to propose a different attack taxonomy or ontology; however, they are diverse and there is no common methodology for the categorization of security intrusions. The majority of the proposed classifications are entirely based on the analysis of published vulnerabilities. For instance, NIDS vendors such as Snort [7] use attack classes that describe the attacker's methods in exploiting these vulnerabilities. We have obtained our classification based on:

- Vulnerability analysis
- Generalized description of the target system (server, client, Web, etc.)

Elementary alerts generated by NIDS sensors are mapped to generalized descriptions of intrusion in a hierarchical representation. The classification is built in the form of a graph with nodes and edges. The nodes specify the attack class and the edges denote the inheritance relationship between attack classes. The classes are mutually exclusive and each alert belongs to only a single class horizontally, but to different classes vertically based on the inheritance relationship. This structural abstraction mechanism is to minimise redundancy and maximize diversity. Hence, even though some alerts are new and unknown, they can be predicted from the results of situation analyses. If an attack is in progress consisting of certain elementary alerts, these atomic alerts are mapped to a general attack description. For any suspicious or unknown actions not detected by the IDS, the probability of their being related to the detected attack is very high. The level of the abstraction progresses from general to specific in a top-down design of the classification hierarchy. For instance, the buffer overflow class can be classified under server, client or Web classes, as this type of attack can target different types of systems. However, some other classes are only categorized as specific system classes, such as DDoS client activity, which is a client-specific attack. Hence, each alert generated by the IDS will be categorized top-down in a hierarchical manner.

Figure 1 shows three examples of how sub-classes inherit attack features from upper classes and how alerts are classified based on these relationships. In Fig. 1(a), the lower class denotes the exact Snort signature *TFTP Get*, *id:1444*, while this signature is classified as *TFTP buffer overflow*. Similarly, in Fig. 1(b), any IDS signature of type of *ACTIVEX attack* can be classified under this class which is in turn classified as a *client buffer overflow*. Figure 1(c) shows that a *stored procedure attack* is described as a Web PHP injection attack. It should be noted that these are only abstract classes and do not denote instances of actual attacks.

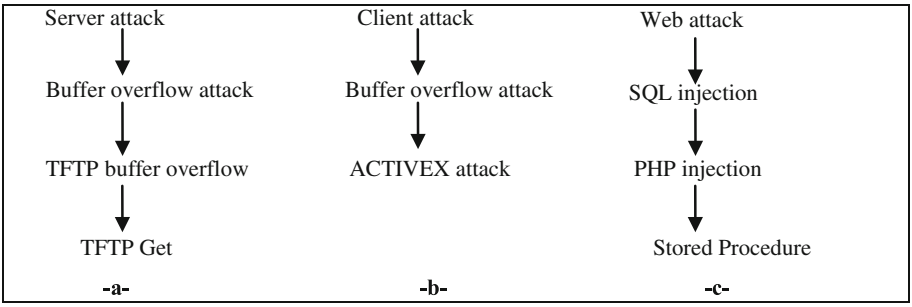

Fig. 1. Examples of attack class inheritance.

3.2 Knowledge-Base Representation

A capability set consists of all the derived elements of capabilities encoded to integer numbers. All alerts are represented in the form of three sections:

1- IDS signature ID to describe the attack by its elementary alert.
2- Pre-conditions set which consists of n capabilities where $n >= 0$.
3- Post-conditions set which consists of n capabilities where $n >= 0$.

The knowledge library of the alerts and their corresponding capabilities are defined into the form shown below:

$$\text{sid:}xxxx;\text{pre:}k_1(n);\text{pre:k}_2(n);\ldots\ldots\text{pre:}k_i(n);\ \text{pos:}l_1(n);\text{pos:}l_2(n);\ldots..\text{pos:}l_j$$

where *xxxx* is the signature ID number, pre denotes pre-conditions, pos denotes post-conditions, k is the capability unique number, and n is a variable argument to specify the attack attributes as follow:

1: source IP address
2: source port
3: destination IP address
4: destination port

3.3 Alert Modeling

IDS alerts are the basic units that represent the occurrence of intrusion as a time series. Essential attack knowledge is derived from signature fields triggered by the IDS in case

of any security violation. It should be noted that the alert generated by the IDS is not necessarily connected to a security attack, as sometimes a legitimate activity can cause some alarms. Moreover, the information in the signature does not contain any sign of whether the attack succeeded or not. However, the abstraction of these alerts to capabilities in respect to temporal and spatial details can give a true view of the security perspective.

Each received alert is mapped to its pre- and post-conditions. It is assumed that the alert is generated because some conditions have to be satisfied and that it will cause some impact on the target system. The relationship between different alerts is identified by matching these conditions. For example, the following alerts (Snort-generated signatures) are obtained from DARPA LLDDOS.1.0 [8] to clarify the correlation concept considering the following Snort signature:

RPC sadmind UDP PING

This signature is generated as result of attempts to test if the *sadmind* demon is running. A *sadmind RPC* service is used to perform administrative activities remotely. The impact of the signature includes disclosure of the running service and system access attempt:

RPC portmap sadmind request UDP

This signature is generated due to the use of a *portmap GETPORT* request to discover the port number of the RPC service, and consequently which port is used by the *sadmind* service.

RPC sadmind UDP NETMGT_PROC_SERVICE CLIENT_DOMAIN overflow attempt

This signature is generated as a result of an attempt to exploit a buffer overflow to obtain a root access.

RPC sadmind query with root credentials attempt UDP

This signature is generated due to the use of root credentials and is an indication of potential arbitrary command executions with root privilege.

RSERVICES rsh root

This signature is generated due to an attempt to login as a root user using *rsh*, and this is an indication of full control of the attacker.

From Table 1, it can be seen that the signatures have some pre- and post-conditions and if a match between these conditions is detected the two alerts are linked as a part of the attack scenario. If two signatures share at least one common capability for instance, *disclosure of running service*, hence they are correlated as a primary stage following by incorporating other factors from the rest of the model components.

3.4 Vulnerability Modeling

Several efforts have been made to correlate IDS signatures with vulnerability information. The aim is to reduce the false positives, which can be a major drawback of such systems. Moreover, these verification mechanisms are incorporated in the IDS to provide a higher quality of alerts, and hence more confidence. The origin of the problem of false positives is that IDSs have no information about the systems they

Table 1. Examples of pre- and post-conditions.

#	Signature	Pre-conditions	Post-conditions
1	RPC sadmind UDP PING	Disclosure of host	Disclosure of running service System access
2	RPC portmap sadmind request UDP	Disclosure of host	Disclosure of port number Disclosure of running service System access Remote Access
3	RPC sadmind UDP NETMGT_PROC_SERVICE CLIENT_DOMAIN overflow attempt	Disclosure of host Disclosure of port number Disclosure of running service	System access Remote access Admin access
4	RPC sadmind query with root credentials attempt UDP	Disclosure of host Disclosure of port number Disclosure of running service System access Remote access	Remote access Admin access

protect. Therefore they are not certain about the success of the attack, simply because the vulnerabilities of the target system are not available. Two trends have emerged in overcoming the false positives issue in IDS performance:

1- Tuning the IDS based on knowledge of the internal policy of the protected environment to operate with a lower number of signatures [9]. Knowledge of network configuration, running services and installed applications is used to disable all the unrelated signatures of the IDS. The advantage of this technique is that the IDS performance is improved significantly. However, some of the information on the activities of the attacker, which may be useful in tracking its behaviour, will be discarded. It should also be noted that real cyber attackers (persistent attackers) try to break into systems using different methods, and these attempts may be not in connection with a particular vulnerability. Moreover, some dangerous attacks in cyber crime do not require any system vulnerability, such as DDoS. In addition, this approach requires intensive and updated vulnerability assessment.

2- The other trend is not suppressing the IDS detection coverage, but instead aggregating, correlating and verifying the generated alerts in a systematic way [6, 10]. Summarized data of occurring events are displayed to the security manager according to their priorities and criticalness. If further details are required to support a specific situation, they can be retrieved by request. A repository of collected information is maintained to support the decision of the IDS management system. Vulnerability scanners are the main candidates to supply this type of data in a periodical manner.

In accordance with the nature of the developed correlation systems, which require full description of any activity in the protected environment, the second mechanism is adopted. The attacks are generalized to obtain a global view of the security situation. This generalization may increase the false positive rate; hence, a suppression technique is needed to reduce the false positive rate without losing any details. This suppression mechanism does not imply any reduction in the IDS coverage, but the consideration of only success attacks.

Snort signatures are supported by two useful fields:

- Vulnerability reference, referring to the major vulnerability standards such as CVE [11], bugtraq [12], and Nessus [13].
- Service to denote a list of the affected services, such as telnet, ftp and MSSQL.

A vulnerability knowledge base is maintained to store the vulnerability situation of each element of the protected network based on the collecting agent (e.g. Nessus). The scanner will also gather the network configuration details such as IP addresses of live hosts and running services, so manual configuration is not considered. In this respect, vulnerability information is considered as external capabilities.

The scope of vulnerability testing is only limited to investigating the presence of the vulnerability and the affected service. An extension can be carried out to consider the target host response; however, there are performance issues (e.g. communication overheads). Nessus is used to extract the following information, which can be used to support the vulnerability component:

– IP addresses of all hosts connected to the target network.
– Operating systems and their versions.
– Open ports and running services.
– Related vulnerability references (e.g. CVE).

When an alert is received from the IDS, its message contains the vulnerability reference and the affected system. Therefore, a logical formula is obtained by searching the vulnerability knowledge to find any matches, as follow:

– If the reference is found and the associated service is running, then the vulnerability is true with high priority.
– If the reference is found and the associated service is not running, then the vulnerability is true with low priority.
– If the reference is not found, then the vulnerability is unknown.

The complete algorithm of alert verification using vulnerability knowledge is shown in Fig. 2.

4 Experimental Results

DARPA 2000 datasets, including LLDDOS 1.0 and LLDDOS 2.0 [14], are often used to evaluate IDSs and alert correlation systems. They consist of two multi-stage attack scenarios to launch Distributed Denial of Service attacks (DDoS). The evaluation goal is to test the effectiveness of our approach to recognize attack scenarios, to correctly

```
Algorithm :Alert verification
Input: elementary alerts generated by IDS  A(IP,SV,VR)
       Host vulnerability information generated by scanner VN(IP,OS,SV,VR)
Output: Vulnerable host VH(IP,V,P)
Methods:
       // IP: IP address,  SV: service, VR: vulnerability, OS operating system
       for i←0 to length[VN]
         do
           if A.IP = VN[i].IP get VN(IP,OS,SV,VR)
           in case of
                  A.VR=VN.VR and A.SV=VN.SV then VH.V←true , VH.P←high
                  A.VR=VN.VR and A.SV≠VN.SV then VH.V←true ,VH.P←low
                  A.VR≠VN.VR then VH.V←false , VH.P←unknown
```

Fig. 2. Alert verification algorithm.

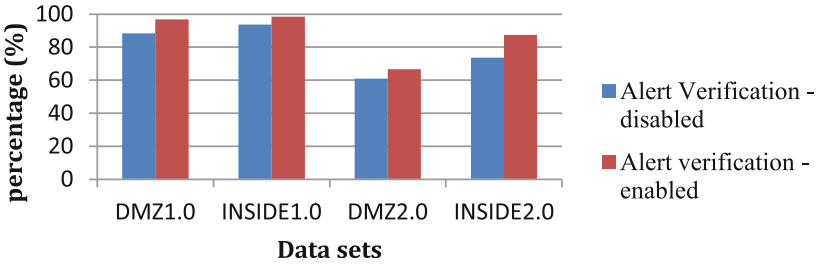

Fig. 3. Recall rate (%) of the DARPA dataset

correlate the alerts, and to minimize the false positives. This experiment is carried out mainly for functional testing to see how the system reconstructs attack stages. A reduction test is also studied in this respect; however, the background traffic in this dataset is limited. We have used these datasets for their available ground truth to assess our correlation approach and to compare our results with those of other researchers. These datasets do not contain the actual alerts from the IDS sensors, and hence we have generated them using a Snort sensor. The detected events evolve over time instead of by batch analysis.

Accuracy metrics are calculated to determine *recall, precision,* and *reduction rate.* Figures 3 to 5 illustrate the key results obtained from different scenarios. Our proposed system has achieved high levels of accuracy among the datasets in LLDDOS1.0, and acceptable levels in LLDDOS2.0. The only low accuracy rate recorded is from the analysis of the DMZ2.0 dataset, and of which we are aware because the actual attack was performed inside the network. The vulnerability model to verify the importance of alerts is also showing a considerable improvement. This is apparent from the number of

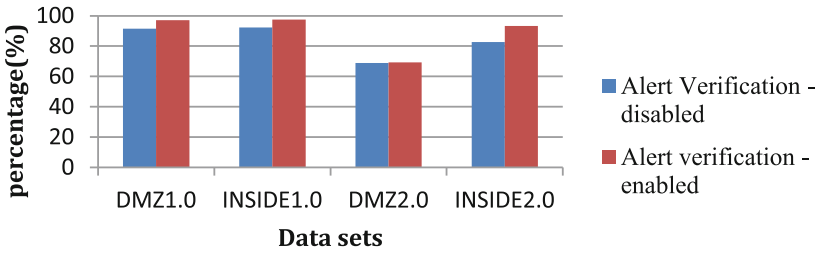

Fig. 4. Precision rate (%) of the DARPA dataset

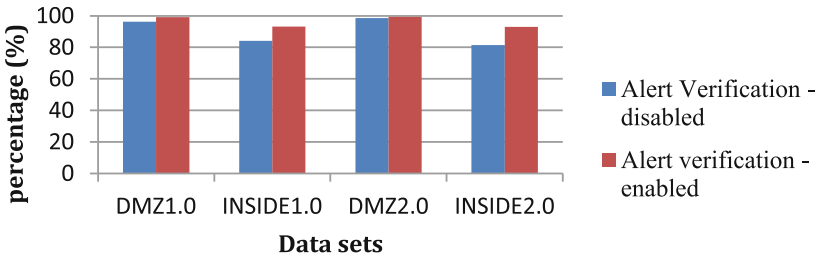

Fig. 5. Alert reduction rate (%) of DARPA dataset

detected events in each dataset. For instance, in DMZ1.0, the number of events has been reduced from 25 events to only 3 related events. The rates are higher if alert verification is used and satisfactory for other tests. In addition, the volume of alert information has been significantly reduced, achieving more than a 90 % reduction rate in most test cases.

5 Conclusion

We have presented the core concept of our knowledge-based reasoning model for alert correlation to address the problem of detection of coordinated attacks. A combined analysis of IDS's alerts and description of attack classes are used to derive the pre- and post- conditions of each received alert. A scheme to represent our knowledge base has been described using a hierarchal and a multilayer classification. Vulnerability modeling is used to support alert verification in order to reduce the generated attack graph.

The evaluation process is based on different metrics to identify the functionality, the reduction and the accuracy rates. An experimental platform has been developed to perform different tests. The obtained results have showed that the proposed system is capable to detect all attack instances with lesser false positive rates. We have confidence that our system has achieved an improvement in relation to identification of attack plans and reduction in graph complexity. False positives have been reduced comparing with other approaches using vulnerability knowledge base. In the next research stage, we will incorporate a statistical model to detect hidden relationships between different attack scenarios.

References

1. Alserhani, F., Akhlaq, M., et al.: MARS: multi-stage attack recognition system. In: Proceedings of the International Conference on Advanced Information Networking and Applications (AINA), pp. 753-759, Perth (2010)
2. Alserhnai, F., Akhlaq, M., et al.: Event-based correlation systems to detect SQLI activities. In: Proceedings Of the International Conference on Advanced Information Networking and Applications (AINA), Bioplis, Singapore (2011)
3. Templeton, S.J., Levitt, K.: A requires/provides model for computer attacks. In: Proceedings of the 2000 workshop on New security paradigms ACM (2000)
4. Zhou, J., Heckman, M., Reynolds, B., Carlson, A., Bishop, M.: Modeling network intrusion detection alerts for correlation. ACM Trans. Inf. Syst. Secur. (TISSEC) **10**(1), 1–31 (2007)
5. Ning, P., Cui, Y., Reeves, D.S., Xu, D.: Techniques and tools for analyzing intrusion alerts. ACM Trans. Inf. Syst. Secur. (TISSEC) **7**(2), 274–318 (2004)
6. Cuppens, F., Miege, A.: Alert correlation in a cooperative intrusion detection framework. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy, 2002. pp. 202-215 (2002)
7. Snort; http://www.snort.org/
8. Haines, J.W., Lippmann, R.P., Fried, D.J., Tran, E., Boswell, S., Zissman, M.A.: DARPA intrusion detection system evaluation: Design and procedures, Technical report, Lincoln Laboratory, Massachusetts Institute of Technology (2000)
9. Valeur, F., Mutz, D., Vigna, G.: A learning-based approach to the detection of SQL attacks. In: Julisch, K., Kruegel, C. (eds.) DIMVA 2005. LNCS, vol. 3548, pp. 123–140. Springer, Heidelberg (2005)
10. Qin, X.: A probabilistic-based framework for infosec alert correlation," Ph.D., Georgia Institute of Technology (2005)
11. Common Vulnerabilities and Exposures (CVE). http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2010-0188
12. Security Focus - BugTraq. http://www.securityfocus.com
13. Nessus: Security Scanner. http://www.nessus.org
14. MIT Lincoln Laboratory; http://www.ll.mit.edu/