published as: Sebastian Roschke, Feng Cheng, Christoph Meinel: A New Correlation Algorithm based on Attack Graph
In Proceedings of the 4th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2011), Torremolinos, Spain, June 2011

A New Alert Correlation Algorithm Based on Attack Graph

Sebastian Roschke, Feng Cheng, and Christoph Meinel

Hasso Plattner Institute (HPI), University of Potsdam,
P.O.Box 900460, 14440, Potsdam, Germany
{sebastian.roschke,feng.cheng,meinel}@hpi.uni-potsdam.de

Abstract. Intrusion Detection Systems (IDS) are widely deployed in computer networks. As modern attacks are getting more sophisticated and the number of sensors and network nodes grows, the problem of false positives and alert analysis becomes more difficult to solve. Alert correlation was proposed to analyze alerts and to decrease false positives. Knowledge about the target system or environment is usually necessary for efficient alert correlation. For representing the environment information as well as potential exploits, the existing vulnerabilities and their Attack Graph (AG) is used. It is useful for networks to generate an AG and to organize certain vulnerabilities in a reasonable way. In this paper, we design a correlation algorithm based on AGs that is capable of detecting multiple attack scenarios for forensic analysis. It can be parameterized to adjust the robustness and accuracy. A formal model of the algorithm is presented and an implementation is tested to analyze the different parameters on a real set of alerts from a local network.

Keywords: Correlation, Attack Graph, IDS.

1 Introduction

Intrusion Detection Systems (IDS) have been proposed for years as an efficient security measure and is nowadays widely deployed for securing critical IT-Infrastructures. The problem of false positive alerts is a well known problem for many IDS implementations [1]. Suboptimal patterns or insufficient thresholds for pattern-based and anomaly-based IDS approaches are the main reasons for a huge number of false-positive alerts. By deploying the IDS sensors in a distributed environment, the number of false positive alerts increases as a single event may be detected and reported multiple times by different involved sensors. The popular solution to address this problem is correlation and clustering of relative or similar alerts[2]. Modern algorithms for alert correlation are using environment information, e.g., attack graphs (AG) [3,4], to improve their performance. The approach proposed in [5] correlates IDS alerts in a memory-efficient way by using an AG and a matching function. The algorithm works with implicit alert correlations, considering only the last alert of a certain type per node in the attack graph. While it is memory-efficient, the approach can not easily be used for forensic analysis. It takes many additional efforts to identify similar attack scenarios having the same anatomy, i.e., using the same exploits on the same hosts at different times. Furthermore, the approach does not consider different mapping functions and aggregation of alerts in detail.

Á. Herrero and E. Corchado (Eds.): CISIS 2011, LNCS 6694, pp. 58–67, 2011.

In this paper, we design an AG based correlation algorithm to overcome the above mentioned drawbacks of the algorithm proposed in [5]. By running the algorithm on hardware[6] with 2 TB of main memory on a single machine, we can make all the correlations between alerts explicit. Therefore, the algorithm is capable of identifying multiple attack scenarios of the same anatomy using an attack graph. The algorithm consists of a mapping of alerts to nodes in the attack graph, an alert aggregation, a building of an alert dependency graph, and a function for finding suspicious alert subsets. Apart from the formal model of the correlation algorithm, we analyze multiple possibilities for AG node matching and aggregation by parameterizing the algorithm. A proof-of-concept implementation is tested using a real data set with alerts from our university network.

The rest of the paper is organized as follows. Section 2 provides an overview on different approaches of alert correlation. In Section 3, the proposed correlation algorithm is described by a formal model. Its capabilities for detecting multiple attack scenarios is discussed. Section 4 presents some experiments and discusses the influence of selected parameters for the algorithm. Section 5 provides a short summary of the contributions and future works.

2 Alert Correlation

The alerts created by the distributed sensors are usually gathered by a central management system and processed by correlation algorithms. The quality of a correlation depends on accuracy and speed. The speed depicts how many correlations can be found in a certain amount of time. The accuracy depicts how many of the identified correlations represent real existing relations between these alerts. Due to more complex attacks and large scale networks, the amount of alerts increases significantly which yields the requirement for improved quality of the correlation. The correlation accuracy depends on the used correlation algorithm. It is obvious that using environmental information can help to improve the quality of alert correlation. Environmental information can be host addresses, running services, active users, network configurations, existing policies, known vulnerabilities in general, and existing vulnerabilities on hosts. To represent the listed information, AG are usually constructed for further processing and analysis. Attack Graphs have been proposed as a formal way to simplify the modeling of complex attacking scenarios. Based on the interconnection of single attack steps, they describe multi-step attacks. Attack Graphs not only describe one possible attack, but many potential ways for an attacker to reach a goal. In an attack graph, each node represents a single attack step in a sequence of steps. Each step may require a number of previous attack steps before it can be executed, denoted by incoming edges, and on the other hand may lead to several possible next steps, denoted by outgoing edges. With the help of attack graphs most of possible ways for an attacker to reach a goal can be computed. This takes the burden from security experts to evaluate hundreds and thousands of possible options. At the same time, representing attack graphs visually allows security personal a faster understanding of the problematic pieces of a network [7,4].

The alert correlation framework usually consists of several components [8]: *Normalization*, *Aggregation (Clustering)*, *Correlation*, *False Alert Reduction*, *Attack Strategy Analysis*, and *Prioritization*. Over the last years, alert correlation research focused on

new methods and technologies of these components. IDMEF [9] and CVE [10] are important efforts in the field of *Normalization*. Approaches of aggregation are mostly based on similarity of alerts [11] or generalization hierarchies [12]. The correlation algorithms [8] can be classified as: *Scenario-based correlation* [13,14], *Rule-based correlation* [15], *Statistical correlation* [16], and *Temporal correlation* [17,18]. The proposed approach can be classified as *Scenario-based correlation*: attack scenarios are specified by a formal language and alerts are correlated, if they can be combined to one of the known scenarios, i.e., alerts are matched in a specific path of the graph which can be considered as attack scenario (e.g., [5]). False alert reduction can be done by using such techniques as data mining [19] or fuzzy techniques [20]. Attack strategy analysis often depends on reasoning and prediction of attacks missed by the IDS [21]. In terms of Prioritization, the alerts are categorized based on their severity, e.g., using attack ranks [22]. To solve problems of alert correlation, a variety of disciplines are used, e.g., machine learning, data mining [19], or fuzzy techniques [20].

An Attack graph based correlation has been introduced in [5]. This approach maps alerts into the AG for correlation, and provides possibilities for hypothesizing and prediction of alerts. It uses a matching function which maps the alerts by comparing the alert type, the source, and the target address of each alert. Furthermore, this approach distinguishes between implicit and explicit correlation. It significantly reduces the number of explicit correlations by considering only the last alert in a set of similar alerts, i.e., the alert type, the source, as well as the target are identical. The explicitly correlated alerts are stored in a data structure called Queue Graph (QG) which tries to reduce the memory consumption.

3 Towards High-Quality Attack-Graph-Based Correlation

In this paper, a modified AG based correlation algorithm is proposed which only creates explicit correlations. Implicit correlations as described in [5] make it difficult to use the correlated alerts in the graph for forensic analysis of similar attack scenarios. Furthermore, the hardware environment used for the In-Memory databases provides machines with huge amounts of main memory which downgrades the priority of memory efficiency for this work. The algorithms consists of five steps, while each step can be parameterized to fine tune the results: 1) preparation, 2) alert mapping, 3) aggregation of alerts, 4) building of an alert dependency graph, and 5) searching for alert subsets that are related. In the preparation phase, all necessary information is loaded, i.e., the system and network information is gathered, the database with alert classifications is imported, and the AG for the network is loaded. We use the MulVAL [3] tool to generate an AG which describes the corresponding system and network information for the target network. The algorithm is based on a set of basic definitions.

3.1 Definitions

Let $\mathcal{T}$ be the set of all timestamps, $\mathcal{H}$ be the set of possible hosts, and $\mathcal{C}$ be the set of classifications. $\mathcal{A}$ can be defined as:

$$\mathcal{A} = \mathcal{T} \times \mathcal{H} \times \mathcal{H} \times \mathcal{C} \tag{1}$$

Let a single alert $a \in \mathcal{A}$ be a tuple $a = (t, s, d, c)$ while the following functions are defined:

- $ts(a) = t$ - returns $t \in T$, the timestamp of the alert
- $src(a) = s$ - returns $s \in H$, the source host of the alerts
- $dst(a) = d$ - returns $d \in H$, the destination host of the alert
- $class(a) = c$ - returns $c \in C$, the classification of the alert

Let $\mathcal{I}$ be the set of impacts described by MulVAL [3] and $\mathcal{VR}$ be the set of known vulnerabilities. Let V be a set of vertices defined as:

$$V = \mathcal{I} \times \mathcal{H} \times \mathcal{VR} \tag{2}$$

For each triple $v = (im, h, r), v \in V$, the following functions are defined:

- $imp(v) = im$ - returns $im \in \mathcal{I}$, the impact of the vertex
- $host(v) = h$ - returns $h \in \mathcal{H}$, the host if the vertex
- $ref(v) = r$ - returns $r \in \mathcal{VR}$, the vulnerability reference of the vertex

Let $AG = (V, E)$ be an AG with vertices V and edges E. An edge $e \in E \subseteq V^2$ is an ordered tuple of vertices (v, v') with $v \in V \wedge v' \in V$. PAG defines all the paths in the AG. The path $P \in PAG$ is defined as a set of edges $P = (v, v') \in E$. $ord(P)$ defines the number of edges in the path P. $in(v, P)$ depicts whether a vertex lies in the path:

$$in(v, P) := \exists (v, v') \in P \vee \exists (v', v) \in P \tag{3}$$

3.2 Mapping

The mapping function map_i maps matching alerts to specific nodes in the AG and is defined as:

$$map_i : a \mapsto \{v \in V \mid \Phi_i(a, v)\} \tag{4}$$

There are different kinds of $\Phi_i(a, v)$ defined in (5), (6), (7), (8), and (9) to parameterize the mapping function.

$$\begin{aligned} \Phi_1(a, v) := \; & \exists v' \in V : (src(a) = host(v')) \\ & \wedge (dst(a) = host(v)) \wedge (class(a) = ref(v)) \end{aligned} \tag{5}$$

$$\Phi_2(a, v) := \exists v' \in V : (dst(a) = dst(v)) \wedge (class(a) = ref(v)) \tag{6}$$

$$\Phi_3(a, v) := \exists v' \in V : (class(a) = ref(v)) \tag{7}$$

$$\Phi_4(a, v) := \exists v' \in V : (src(a) = host(v')) \wedge (dst(a) = host(v)) \tag{8}$$

$$\Phi_5(a,v) := \exists v' \in V : (dst(a) = host(v)) \tag{9}$$

We will refer to match modes when using a specific $\Phi_i(a,v)$. The match modes are named as follows:

- $\Phi_1(a,v)$ - match mode *cvesrcdst*
- $\Phi_2(a,v)$ - match mode *cvedst*
- $\Phi_3(a,v)$ - match mode *cve*
- $\Phi_4(a,v)$ - match mode *srcdst*
- $\Phi_5(a,v)$ - match mode *dst*

3.3 Aggregation

Let $A \subset \mathcal{A}$ be the set of alert that is supposed to be aggregated. Let th be a threshold and $x \in A, y \in A$ two alerts, then the relation R_A is defined as:

$$\begin{aligned} R_A = \{(x,y) \in A^2 : \\ (\mid ts(x) - ts(y) \mid < th) \wedge (src(x) = src(y)) \\ \wedge (dst(x) = dst(y)) \wedge (class(x) = class(y))\} \end{aligned} \tag{10}$$

R_A^* defines an equivalence relation on the transitive closure of R_A. The alert aggregation combines alerts that are similar but where created together in a short time, i.e., the difference of the timestamps is below a certain threshold th. It defines a set of equivalence classes $A_{/R_A^*}$ over the equivalence relation R_A^*.

3.4 Alert Dependencies

Let $A_m \subset A$ be the set of alerts that have been matched to a node in an AG:

$$A_m = \{[a] \in A_{/R_A^*} \mid map_i(a) \neq \emptyset\} \tag{11}$$

The alert dependencies are represented by a graph $DG = (A_m, E_{m,k})$, with $E_{m,k}$ as defined in (12).

$$E_{m,k} = \{([x],[y]) \in (A_{/R_A^*})^2 \mid \Psi_k([x],[y])\} \tag{12}$$

The set $E_{m,k}$ can be parameterized by the functions Ψ_k as shown in (13), (14), and (15).

$$\begin{aligned} \Psi_1([x],[y]) := (ts([x]) < ts([y])) \\ \wedge (\exists (v,w) \in E : (v \in maps_i(x) \wedge w \in maps_i(y))) \end{aligned} \tag{13}$$

$$\begin{aligned} \Psi_2([x],[y]) := (ts([x]) < ts([y])) \\ \wedge (\exists P \in PAG : (ord(P) = n) \wedge (\exists v,w : \\ (v \in maps_i(x) \wedge w \in maps_i(y) \\ \wedge in(v,P) \wedge in(w,P)))) \end{aligned} \tag{14}$$

$$\begin{aligned}\Psi_3([x],[y]) := (&ts([x]) < ts([y])) \\ &\wedge(\exists P \in PAG : \exists v, w : \\ &(v \in maps_i(x) \wedge w \in maps_i(y) \\ &\wedge in(v,P) \wedge in(w,P))) \end{aligned} \tag{15}$$

The dependency graph DG is defined by the matched and aggregated alerts A_m as vertices and the relations between these alerts as edges $E_{m,k}$. There are three possible ways to define these relations using Psi_k. Psi_1 defines two alerts as related, if they are mapped to neighboring vertices in AG. Psi_2 defines two alerts as related, if they are mapped to two vertices in AG that are connected by the path P with the length of n. Psi_3 defines two alerts as related, if they are mapped to two vertices in AG that are part of the same path P.

3.5 Searching

Each path in the alert dependency graph DG identifies a subset of alerts that might be part of an attack scenario. DG is used in the last step to determine the most interesting subsets of alerts, respectively the most interesting path in the alert dependency graph. The last step of searching alert subsets is done by performing a Floyd Warshall algorithm [24,25] to find all the shortest paths. Furthermore, the diameter dia (i.e. the value of the longest instance of the shortest paths) is determined and each path DP_i that has the length $ord(DP) = dia$ is converted in subsets of alerts. All the subsets S_x are defined as:

$$S_x = \{a \in A \mid in(a, DP_i)\} \tag{16}$$

With a simple optimization, the algorithm allows to identify multiple different attack scenarios of the same anatomy. By sorting the suspicious alert subsets according to the smallest difference between alert a_1 and a_n, the algorithm will identify the alerts that are near each other on the time-line as related to one attack scenario. Let $as_1 = \{a_1, ..., a_n\}$ and $as_2 = \{b_1, ..., b_n\}$ be two alert sets where the only difference between a_i and b_i is the times-tamp $ts(a_i) \neq ts(b_i)$. There are three different combinations how these alerts can be located on a time-line:

1. as_1 and as_2 are not overlapping at all, i.e., $ts(a_n) < ts(b_1)$
2. $as_1 = \{a_1, a_2, ..., a_k, ...a_n\}$ and $as_2 = \{b_1, ..., b_n\}$ are partially overlapping, i.e., $\exists k \in \mathbb{N} \forall a_{i \in \{1,k\}} \mid ts(a_k) > ts(b_1)$
3. as_2 is completely overlapped by as_2, i.e., $(ts(a_1) < ts(b_1)) \wedge (ts(a_n) > ts(b_n))$

The modified algorithm can identify both suspicious alert sets in case 1, which is impossible for the algorithm in [5]. Due to the memory limitation, this algorithm only considers the last matching alert for each node in the AG.

4 Experiment and Discussion

The algorithm is implemented on a modularized correlation platform which is designed based on in-memory techniques and a multi-core hardware [6]. For testing the correlation algorithm, we created a data set of IDMEF alerts by running a Snort[23] sensor in our university network. The sensor gathered 43485 alerts in 6 days of runtime. Meanwhile, we had a Snort sensor running in several vulnerable subnets of our university network which include several vulnerable hosts. The AG for this subnet is constructed accordingly. We performed a multi-step attack covering multiple hosts in the existing subnets. This provides us with a set of alerts we consider as attack trace in the following work. By injecting this attack trace into the clean data set of the whole university network, we can simulate alert sets without running the real attacks over the production network of our university. The attack is done by compromising 5 hosts in the vulnerable network spread over 4 subnets. We conducted multiple experiments using the data set and the attack trace.

For this set of experiments, we injected one attack trace into the clean data set to test if it can be found in a real set of alerts. Furthermore, we analyzed selected parameters of the algorithm and their influence on the actual result, such as the match mode, the aggregation threshold, the dependency build mode, and the searching algorithm for alert subsets. We fixed the parameters for defining dependencies between alerts and searching for suspicious subsets of alerts. A dependency between two alerts is defined by $E_{a,1}$ using Ψ_1, i.e., alerts that are mapped to adjacent nodes fulfilling the timing constraint are dependent. The dependency graph $DG = (D_a, E_{a,1})$ is used to find all the shortest paths by performing a Floyd-Warshall algorithm and defining the diameter. As shown in Table 1, the match modes using the CVE[10] as criteria work precisely and can identify attacks that follow the AG.

With the alert set from our experiment, this match mode filters about 99.98% of the alerts. The match modes that ignore CVE and are based on the source and destination address are less accurate in terms of the vulnerability used. In our experiments it still showed a filtering rate of 95.58%. The advantage of this match mode is that an attacker can use different vulnerabilities to compromise a host covered in the AG and can still be recognized by the correlation. In the CVE based matching, there is only one suspicious alert set found, basically the one which is supposed to be found. The matching without CVE identifies much more suspicious alert sets. By looking at these alert sets, we recognized that most of the alert sets are using similar alerts and that the high number of the alert sets is due to the fact that there are two hosts which seem to be frequently used and that the Snort sensor produces more false positives for them. To improve it, we can

Table 1. Experiment results - match modes with fixed aggregation threshold of 2s

	Matched alerts	% filtered	Suspicious alert sets
cvesrcdst	5	99.99	1
cvedst	5	99.99	1
cve	8	99.98	1
srcdst	1836	95.78	1010
dst	1923	95.58	1010

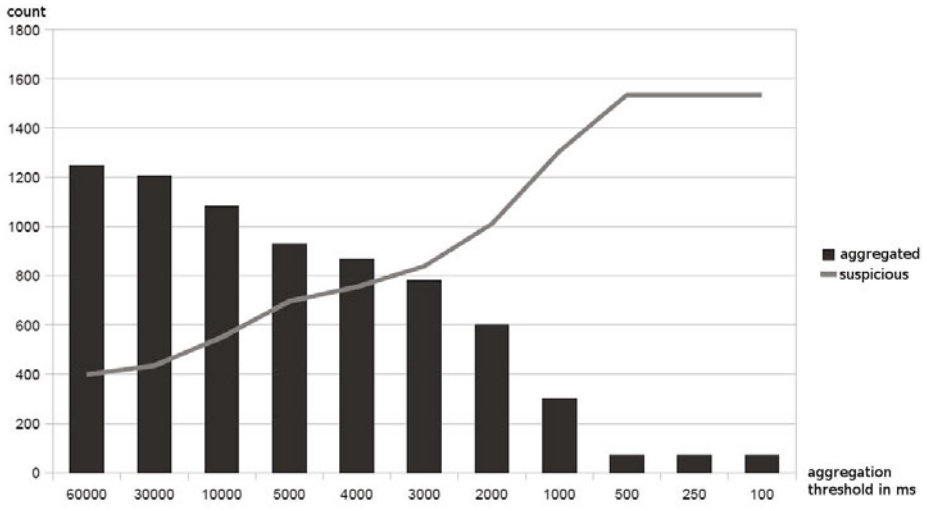

Fig. 1. Experiment results - aggregated alerts and suspicious alert sets for different aggregation thresholds with match mode *srcdst*

try to filter more alerts in the matching by using a matching mode that is based on alert categories, e.g., we do not need to match alerts that are related to an attacks categorized as Denial-of-Service (DoS).

As shown in Figure 1, the aggregation threshold influences the number of suspicious subsets, as a lot of matched alerts can be filtered. The diagram shows the amount of aggregated/filtered alerts as blocks and the amount of identified suspicious alert subsets as line. The overall positive effect of aggregation seen in the diagram is due to the high similarity of consecutive alerts in our data set. We realized multiple alerts in a row that are pretty similar and most likely belong to the same communication. These alert clusters can be aggregated without loosing accuracy of correlation result. An aggregation threshold of $60s$ or larger filters 1246 out of 1836 alerts, i.e., 67.86% of the matched alerts can be filtered in the best case. The algorithm determined 398 suspicious alert sets for this case, which is a significant improvement over the 1010 suspicious alert sets for the threshold of 2 seconds. Thresholds smaller than 2 seconds do not show reasonable effect.

5 Conclusion

In this paper, an AG based correlation algorithm is proposed that overcomes the drawbacks of the algorithm described in [5]. It creates only explicit correlations and enables the identification of multiple attack scenarios of the same anatomy. The algorithm consists of a mapping of alerts to AG nodes, the alert aggregation function, a function for building an alert dependency graph, and a function for finding suspicious subsets using the Floyd-Warshall algorithm and the diameter value. In addition to the formal model of the correlation algorithm, we analyzed multiple possibilities for the node matching

and aggregation function in detail to parameterize the algorithm. Finally, we tested the capabilities and analyzed the influence of the parameters by using a real data set of alerts generated from our university network.

The algorithm is implemented and tested based on real data. As this data set is still small, we want to conduct more experiments using larger data sets. Running multiple attacks against the secured network that (partially) cover the AG is also useful to analyze the efficiency and performance of the algorithm. Although the *srcdst* based matching filters about 95% of the alerts, it might still be possible to improve this matching by using attack categories or ontologies. This can improve the filtering without getting inaccurate in the results. Furthermore, the algorithm needs some computing power to consume, especially the Floyd-Warschall algorithm. This can be improved by providing a multi-core-compliant version of the algorithm. Apart from IDMEF alerts, there are additional data sources (e.g., log files) that can be used for AG-based correlation. It should be supported by a more flexible mapping function.

References

1. Northcutt, S., Novak, J.: Network Intrusion Detection: An Analyst's Handbook. New Riders Publishing, Thousand Oaks (2002)
2. Kruegel, C., Valuer, F., Vigna, G.: Intrusion Detection and Correlation: Challenges and Solutions. AIS, vol. 14. Springer, Heidelberg (2005)
3. Ou, X., Govindavajhala, S., Appel, A.: MulVAL: A Logic-based Network Security Analyzer. In: Proceedings of 14th USENIX Security Symposium, p. 8. USENIX Association, Baltimore (2005)
4. Noel, S., Jajodia, S.: Managing attack graph complexity through visual hierarchical aggregation. In: Proceedings of Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC 2004), pp. 109–118. ACM, Washington DC (2004)
5. Wang, L., Liu, A., Jajodia, S.: Using attack graphs for correlation, hypothesizing, and predicting intrusion alerts. Journal of Computer Communications 29(15), 2917–2933 (2006)
6. Roschke, S., Cheng, F., Meinel, C.: A Flexible and Efficient Alert Correlation Platform for Distributed IDS. In: Proceedings of the 4th International Conference on Network and System Security (NSS 2010), pp. 24–31. IEEE Press, Melbourne (2010)
7. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated Generation and Analysis of Attack Graphs. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy (S&P 2002), pp. 273–284. IEEE Press, Washington, DC (2002)
8. Sadoddin, R., Ghorbani, A.: Alert Correlation Survey: Framework and Techniques. In: Proceedings of the International Conference on Privacy, Security and Trust (PST 2006), pp. 1–10. ACM Press, Markham (2006)
9. Debar, H., Curry, D., Feinstein, B.: The Intrusion Detection Message Exchange Format, Internet Draft. Technical Report, IETF Intrusion Detection Exchange Format Working Group (July 2004)
10. Mitre Corporation: Common vulnerabilities and exposures CVE Website, `http://cve.mitre.org/` (accessed March 2009)
11. Valdes, A., Skinner, K.: Probabilistic alert correlation. In: Lee, W., Mé, L., Wespi, A. (eds.) RAID 2001. LNCS, vol. 2212, pp. 54–68. Springer, Heidelberg (2001)
12. Julisch, K.: Clustering intrusion detection alarms to support root cause analysis. ACM Transactions on Information and System Security 6(4), 443–471 (2003)

13. Debar, H., Wespi, A.: Aggregation and correlation of intrusion-detection alerts. In: Lee, W., Mé, L., Wespi, A. (eds.) RAID 2001. LNCS, vol. 2212, pp. 85–103. Springer, Heidelberg (2001)
14. Al-Mamory, S.O., Zhang, H.: IDS alerts correlation using grammar-based approach. Journal of Computer Virology 5(4), 271–282 (2009)
15. Ning, P., Cui, Y., Reeves, D.: Constructing attack scenarios through correlation of intrusion alerts. In: Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002), pp. 245–254. ACM Press, Washington, DC (2002)
16. Qin, X.: A Probabilistic-Based Framework for INFOSEC Alert Correlation, PhD thesis, Georgia Institute of Technology (2005)
17. Qin, X.: Statistical causality analysis of INFOSEC alert data. In: Vigna, G., Krügel, C., Jonsson, E. (eds.) RAID 2003. LNCS, vol. 2820, pp. 73–93. Springer, Heidelberg (2003)
18. Oliner, A.J., Kulkarni, A.V., Aiken, A.: Community epidemic detection using time-correlated anomalies. In: Jha, S., Sommer, R., Kreibich, C. (eds.) RAID 2010. LNCS, vol. 6307, pp. 360–381. Springer, Heidelberg (2010)
19. Manganaris, S., Christensen, M., Zerkle, D., Hermiz, K.: A data mining analysis of rtid alarms. Computer Networks 34(4), 571–577 (2000)
20. Siraj, A., Vaughn, R.B.: A cognitive model for alert correlation in a distributed environment. In: Kantor, P., Muresan, G., Roberts, F., Zeng, D.D., Wang, F.-Y., Chen, H., Merkle, R.C. (eds.) ISI 2005. LNCS, vol. 3495, pp. 218–230. Springer, Heidelberg (2005)
21. Ning, P., Xu, D., Healey, C.G., Amant, R.S.: Building attack scenarios through integration of complementary alert correlation method. In: Proceedings of the Network and Distributed System Security Symposium (NDSS 2004). The Internet Society, San Diego (2004)
22. Porras, P.A., Fong, M.W., Valdes, A.: A mission-impact-based approach to INFOSEC alarm correlation. In: Wespi, A., Vigna, G., Deri, L. (eds.) RAID 2002. LNCS, vol. 2516, pp. 95–114. Springer, Heidelberg (2002)
23. Snort IDS: WEBSITE, `http://www.snort.org/` (accessed November 2009)
24. Floyd, R.: Algorithm 97 (SHORTEST PATH). Communications of the ACM 5(6), 345 (1962)
25. Warshall, S.: A Theorem on Boolean Matrices. Journal of the ACM 9(1), 11–12 (1962)