Strategic Management of Security Information through an Entropy-Based Alert Correlator

We present an integrated system to process in real time a huge incoming stream of alerts produced by current intrusion detection systems. A key component of this system includes an unsupervised clustering algorithm that combines a temporal sliding window, entropy tests, and expert rules to track the on-the-fly evolution of alert groups.

G. C. Garriga [1]

HIIT Basic Research Unit
Helsinki University of Technology
gemma.garriga@hut.fi

J. L. Balcázar [2]

LARCA Research Group
Universitat Politècnica de Catalunya
balqui@lsi.upc.edu

J. Ballesteros [3]

TISSAT S.A. Avd. Leonardo Davinci 5, 46980, Paterna, Valencia, Spain
fraballo@gmail.com

J. Megias [4]

GMV S.A. Av. Cortes Valencianas, 39, 46015 Valencia, Spain
jmegias@gmv.com

A. Palomares [5]

TISSAT S.A. Avd. Leonardo Davinci 5, 46980, Paterna, Valencia, Spain
apalomares@tissat.es

Introduction

Managing the security of highly complex networks, composed by thousands of networking systems, is an intensive skill-demanding and time-consuming task. It requires processing, analyzing, correlating and understanding every aspect of the network traffic to successfully detect potentially dangerous events that could pose a threat to any protected systems. Subsystems are thus sought that monitor, through specifically tailored sensors, the main communication events of a system (such as network traffic or user commands that account for human-computer communication events), trying and detecting undesired behaviours. Great advances along this line are described in [1, 3, 6, 7, 8, 9].

Managing the security of highly complex networks is an intensive skill-demanding and time-consuming task.

In fact, the security of this traffic is usually monitored by several security systems, with converging objectives but very different ways to address and identify a potential threat. One common approach is based on a pattern-matching analysis, which raises an alert if a previously known attack, present on the IDS installed rules, is detected. Dozens of new detection rules are added daily for those IDSs, demanding the additional effort for security administrators to update rule sets and tuning them at least weekly. Alternatively, anomaly detection is more capable to detect unknown threats, but very prone to offer lots of false positives.

Therefore, usually the millions of daily alerts detected by those different IDSs are not taken into account by the security managers of those networks, leaving IDS's as a forensics-only tool, useful to understand what happened once a successful attack has compromised one or several systems. Actually, in every attempt so far at using the alerts online for preventive system protection, a major intervention of a human guard is still necessary. The reason lies in that, towards detection of *alarming* behaviour, in most cases we have to be content with detecting *unusual* behaviour; and these notions do not coincide. Thus, a relevant line of research aims at reducing, down to manageable rates, the amounts of information to take care of by humans: First, by organizing all the information, storing information about vulnerabilities as they become exposed by attacks, and tracking their evolution and fix history, so that attacks to a vulnerability already corrected do not distract from more urgent matters; our main aim is the construction of such a system. Second, allowing for simultaneously caring for several alerts at once and organizing them into priority layers; this is where we wish our system to contribute to advancing the state of the art. Inferring that alarms are symptoms of the same malicious behaviour (their so-called **root cause**), may be a way out towards better systems [2, 4, 5, 10, 11]. This needs as much domain information as possible, for instance through ontology-based knowledge. Elicitation of the

necessary knowledge, however, remains often the bottleneck. We explore here a different path: we use a very simple and fast timestamp- and IP-based correlation notion that tends to construct large clusters, thus incurring in the risk of mixing several malicious breaches into a single intrusion alarm, but with the advantage of a reduced human inspection load; and we try to make up for the inconvenience by an entropy-based permanent monitorisation of the clusters themselves, splitting them whenever the system finds it likely that several root causes are being mixed into a single set of correlated alerts. The approach is subjected to a preliminary evaluation using the daily data produced by the different IDS's installed in the network system of Valencia's local government, Generalitat Valenciana.

Background

Generalitat Valenciana (www.gva.es) is the local government of Comunidad Valenciana, the third Spanish most developed region. Being one of the Spanish regions most devoted to providing quality e-government services to its citizens, it has a much evolved security consciousness. Besides, their global network provides many relevant horizontal services (both inbound and outbound) that need a state-of-the-art security, like government's Internet communications, citizen's access to e-government processes, public web services, intranet communications, partner and provider collaboration etc. The security of some of these networks is managed by Tissat, a Spanish IT services company that applies R&D to address and solve day to day problems.

It is therefore important to understand that the project had two complementary focuses: First, to enhance Generalitat Valenciana's threat detection capabilities, and hence to be able to protect more thoughtfully its information, and second, to rationalize and ease the work of their security experts, using its expensive skills to do high level work, to understand long-term patterns on security evolution and to be able to promptly respond to any dangerous threat. One of the most important features desired was the capability of grouping similar threat events detected by any IDS/IPS, issue not fully addressed on any commercial or open source tools (it is required that the user to manually configure the rules to correlate and group similar events, causing to share the root problem with IDS/IPS systems: the need of fine tuning the system constantly).

System description

The Security Information Management System used to manage Generalitat Valenciana's common networks has been crafted to be highly modular, with several different evolving services:

- *Vulnerability Management*: Its goal is to automate the management of the vulnerabilities lifecycle, which is composed of the phases of discovery, asset prioritisation, analysis, remediation and verification. At later stages this information will be correlated with threat alerts, therefore making the system able to tell which attacks actually landed on a system vulnerable to them. The base tool to manage vulnerabilities is the formerly open-source system Nessus (www.nessus.org).

- *Threat Management*: Its main aim is to offer an almost-real time console that allows the Security Operations Centre team to view only relevant security events (they are really clusters of alerts) detected by any IDS/IPS, allowing it to react in a timely manner. The base tools used to feed the alert information chosen was SNORT (www.snort.org), an open source IDS with a very actively maintained rule set, used in many kinds of security environments.
- *Malware and Virus Management:* Similar to the Threat Management module, but much more focused on the particular characteristics of Virus, Worms, Trojans and Spyware traffic, providing the ability to detect infection patterns. This system can be fed with any tool that can detect the previously named types of malware, but, in the sake of using open source software, the first one included has been ClamAV (www.clamav.net), an open source virus detection tool.

The underlying system used to support those modules is based on a distributed agent-based environment, with a three tier application that processes the information and offers an interface for users. Those agents control local parsers that import and normalize data (in order to easily add additional tools), opening secure connections to the system's agent-manager module that its responsible for double-checking agent's data, correlating it with the organization's assets inventory and dumping this information to the system's central database. Agents and agent-manager modules are designed to treat millions of events daily.

Data

The data for the system is provided by several IDS's located on different networks, which log millions of events into their local IDS databases. On any network IDS there is a certain set of attributes provided, no matter what tools are used:

- Time of detection: moment when the attack was detected by the systems.
- Source: Where the suspicious event came FROM, always an IP address.
- Target: Where the suspicious event was headed to, always an IP address.
- Event ID: Each tool uses its own proprietary taxonomy and classification.

Time is one of most important attributes for any security correlation attempt, given that it is used to define the relevant time windows. Therefore, it is a must that all IDS's are time-synchronize using NTP (Network Time Protocol). One of the most useful attributes that can characterize an event is the CVE id (Common Vulnerability Exposure, cve.mitre.org): it is a public list of standardized names for vulnerabilities, used both by Vulnerability Assessment Tools (VAS) and IDSs. If this attribute is nonempty, it can provide direct correlation with a specific vulnerability, allowing the system to promptly determine if the attack was successful

(the event CVE id and the vulnerability were the same on the same target). All data logged on into the IDS local database is periodically polled, preprocessing and normalizing information event using a standard event exchange format. This normalization is used to fill in any missed event attributes, to add information about the IDS that sends the event and finally to ease the addition of new IDS technologies: building a parser suffices.

Once the data is preprocessed, it is sent to the system's central database, which consolidates information in order to discard blatantly repeated events. It is only at this point when the adaptive clustering algorithm can be used.

Modelling approach

Formally the traffic of a given network is analyzed by a set of IDSs, namely $\{ids_1, \ldots, ids_n\}$. Attacks against the system manifest themselves as an stream of alerts produced by each IDS, S_i. Each S_i generated by ids_i is an infinite sequence of pairs (a, t), where *a* is an alert and *t* is the occurrence time of such alert, namely the timestamp. Without loss of generality and to simplify the discussion, we consider that all the streams produced by the different IDSs in a network can be collapsed into one single stream of alerts (possibly overlapping in time) $S=\langle\ldots(a,t)(a',t')\ldots\rangle$, $t \leq t'$. The properties of each alert in this stream S identify which IDS detected the event. Given the input stream of alerts S, the output of our algorithm is a stream of alert clusters or alert groups. A cluster or group c is a set of alerts that are close enough according to a certain distance function – e.g. the value of such function is over a certain user specified threshold – and they occur also close enough in time – e.g. one alert in the cluster can reach another in the same cluster within a time window width given by the user. More details about these factors will be provided in the next subsection. The output stream produced by our algorithm will be of the form $O = \langle\ldots(\{c_i\}, t)(\{c_j\}, t')\ldots\rangle$, where $t < t'$, $\{c_i\}$ is the set of clusters of alerts active at time t, and likewise $\{c_j\}$ and time t'.

Essentially, every time we will read a new alert (a, t) from S (or equivalently, the set of alerts with the same time stamp t in S) the current state ($\{c_i\}$, t) will be updated. This updating corresponds to either adding to one cluster in the set $\{c_i\}$ the new information given by a, or creating a new cluster with the single alert a if none of the existing clusters is "close" enough; then, possibly, some clusters will become inactive because they have not been updated with any alert in the recent last time stamps. Moreover, it might happen that some clusters will exhibit entropy high enough to be split in several sub-clusters.

Technical details

The following are important technical elements to consider:

Vulnerabilities of the system – The network is commonly scanned in the search of vulnerabilities. Names of vulnerabilities come standardized in a CVE identifier. Unfortunately, running a vulnerability scan is not possible at all times as it consumes network bandwidth and it is usually very costly. Also, the scope of these scans is limited.
Alert – An alert corresponds to an event generated by an IDS to notify a potential breach in the system.
Window – A window sliding over the input stream of alerts will define the notion temporal closeness between events. Formally, at a time t a window of width w is defined as the interval [t − w, t], so that at the current state $O = \langle\ldots(\{c_i\}, t)\ldots\rangle$, the set of clusters $\{c_i\}$ is active, meaning that each cluster contains at least one alert that occurred within the current window. On the other hand, a cluster will get outdated, and thus it will be ready to become inactive and "leave the window", when it has not been updated with any alert within the defined window interval.
Cluster of alerts – This corresponds to a set of alerts which share certain similarity. The centroid of a cluster is updated every time a new alert a is added to the cluster: the centroid will be updated with the most similar field/s shared with the new alert a.

Similarity function for alerts – A similarity function *f* for alerts is necessary to construct clusters. The similarity of an alert a read from S w.r.t. an active cluster c with alert centroid a will be defined by *f*(a,a') in [0..1]. If *f*(a,a') is over a user specified threshold then the alert a will be added to the cluster.

The similarity function is specified for this application according to the expert rules. Similar alerts must occur close enough in time; this will be checked through the window condition: the window width is provided by the user. A second set of criteria analyzes in which cases two sources might be considered similar, and two targets might be considered similar. To compare IP addresses we use cosine similarity:

$$\cos(IP, IP') = \frac{IP \cdot IP'}{\|IP\|_2 \|IP'\|_2}$$

where $IP \cdot IP'$ denotes the dot-product of the two vectors. If two IP addresses are identical then the cosine similarity will be 1. To compare two port numbers we simply perform a ratio between them.
It is necessary to establish priorities when comparing fields between alerts: source port is less important than target port, which is less important than the target IP, which at the same time is less important than the similarity of the source IP. Also, in case two alerts share exactly the same CVE id (if it exists) then similarity should be always 1. The combination of these values is not trivial and decisions have to be taken, e.g. similarity of 1 in IP source would indicate that those two alerts are indeed very similar and represent an intrusion together, but on the other hand, if any of the fields obtains similarity 1, then it is worth trusting in the highest similarity of the four fields. We combine all these rules of experts into a similarity function *f* which is symmetric.

Entropy – In real world conditions or for certain user-specified parameters, clusters might become disperse, i.e. not showing any coherence in the alerts contained in the cluster. For example, if the similarity threshold is not high enough, then many alerts will would grouped together even if they are not really similar; or, if the specified window width is too wide, then clusters may become active during too prolonged periods of time thus favouring the addition of new alerts that increase the disorder; or also, if the expert rules are not sufficient then clusters might not be too concentrated. In such situations, it is useful to split the cluster into a subset of other clusters that exhibit less disorder and less number of alerts.

The Shannon entropy measure (typically used in information theory) will be used here to evaluate the level of disorder that there is in a cluster. Given a cluster c, the entropy will be computed over the set of source IP addresses of the alerts in the cluster, and also over the set of target IP addresses of the alerts in the cluster. Eventually we will assign the maximum of those two values to designate the entropy of the cluster, i.e.:

$$H(c) = \max\{H(IP_{source}), H(IP_{target})\},$$

where

$$H(IP_{source}) = \sum_i -p_i \log_2 p_i ,$$

with index i ranging through the different source IP addresses and p_i being the probability of the address IP_i in the cluster c, computed simply as the ratio of occurrences in the cluster. An entropy score over a user-specified threshold for a cluster c shows the need of splitting the cluster c into as many sub-clusters as different IP addresses (either source or target, depending on the one contributing to H(c) with the maximal entropy).

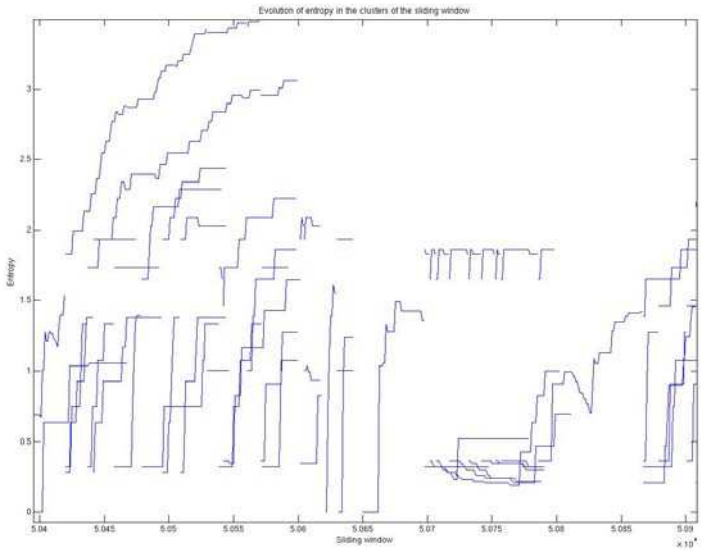

Figure 1: Snapshot of the entropy evolution of active clusters: x-axis represents the time evolution and y-axis is entropy. Each continuous blue line represents the entropy evolution of an active cluster within this time interval snapshot. The minimum entropy threshold was set to 3.5 in this experiment, so when a cluster reaches this limit, then the entropy value immediately drops (meaning that the cluster is split into several others with less entropy). Also note that for some clusters the entropy might decrease at some point of the evolution. This is because the cluster is getting more compact, instead of getting more disordered

Preliminary empirical evaluation

In this section we present a preliminary empirical evaluation of the entropy based alert correlation approach. We will test our approach on the alerts generated during the day 12/27/2006 in the networking system of Generalitat Valenciana. This corresponds to 75,821 alerts, roughly 6Mb of data. Distance between alerts is measured here in seconds. The methodology of these first experiments consists in varying the three input parameters: window width, similarity threshold and entropy threshold. A snapshot of the entropy evolution for the active clusters is depicted in Figure 1.

In next Figures 2 and 3: The first plot is the number of active clusters at time t (y-axis) versus time t (x-axis). The second plot is the total number of alerts in the active clusters at time t (y-axis) versus time t (x-axis). The third plot is the average number of alerts per active cluster at time t (y-axis) versus time (x-axis). The fourth plot is the maximal entropy from all active clusters at time t (x-axis) versus time (y-axis).

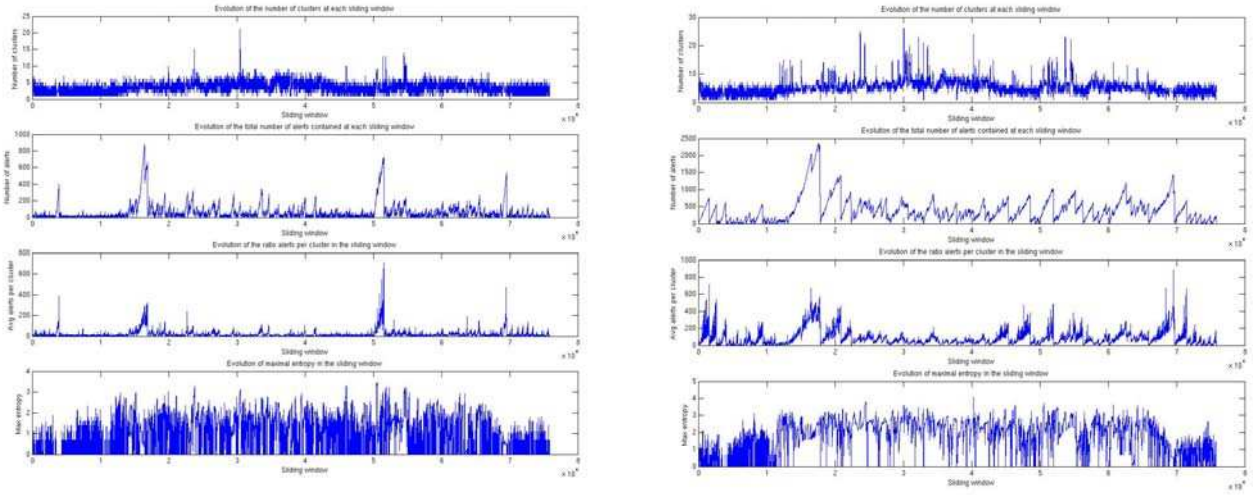

Figure 2: Comparing the state evolution of clusters through time with a tight window width (7 seconds, left) versus a wide window width (20 seconds, right). Rest of parameters: min similarity 1.0 and min entropy threshold 3.0. Observe that with wide window width, the clusters stay active longer thus producing higher accumulations of clusters (peaks of the second plot), and also these are clusters with much more alerts.

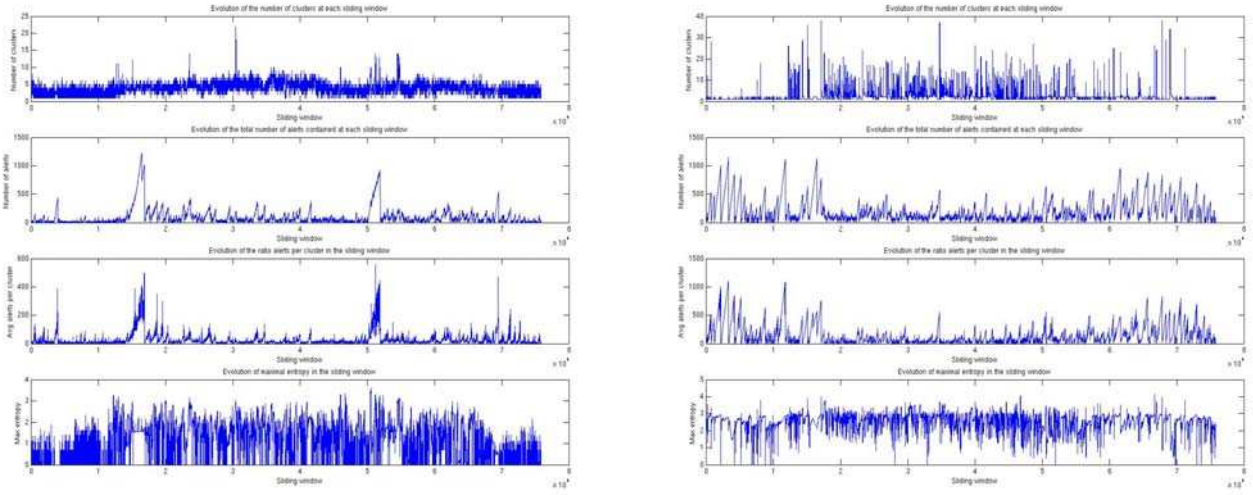

Figure 3: Comparing the state evolution of clusters through time with a strict similarity threshold (1.0, left) versus a loose similarity threshold (0.7, right). Rest of parameters: window width is 9 seconds and minimum entropy threshold is 3.0. Observe that with a loose similarity, the number of alerts in clusters tends to increase. Also the number of clusters per time t is high for similarity 0.7, as alerts updated clusters very easily, and this gives those clusters the chance of being active at least in the next 9 seconds marked by the window width.

CASE STUDY	TYPE	FREQUENT SOURCE PORT	FREQUENT TARGET PORT	CLASS
1	WEB SURFING		80 or 443	AUDIT
2	HTTPD RESPONSE	80 or 443		AUDIT
3	DNS ANY REQUEST		53	AUDIT
4	DNS ANY RESPONSE	53		AUDIT
5	SMTP SENDING		25	AUDIT
6	SSH CONECTION		22	AUDIT
7	TCP PORT SCAN			ATTACK
8	DENIAL OF SERVICE			ATTACK
9	NET SCAN			ATTACK

Table 1: Description of the 9 different types of clusters are classified by the experts during the experiments. For some types of clusters we show as well the frequency source/target port associated to the cluster type. Last column (class) corresponds to the classification of the cluster (either audit or attack) given by the experts.

Discussion

Analysis of the outcome of the experiments described (and of further experiments omitted) reveals some interesting considerations. Regarding the time window, we find a rather strict linearity between its length and the number of false positives and false negatives: as expected, but confirmed with a linear dependency on the data obtained, larger windows improve substantially the quality of the detection rate. Of course, larger windows also imply higher resource consumption, so each particular application will need some tuning effort to find the largest window width acceptable for the installation.

Similarity threshold and entropy threshold are harder nuts to crack. No linear correlation is found, but it can be observed that the optimal results are found in specific intervals: [0.7,1] for similarity (which is not surprising) and [2,3] for the entropy: along these intervals, the best ratios with respect to false positives and false negatives were usually found, but little more can be said. These are useful guidelines to tune this algorithm for an actual SIM system.

In our several real-world simulations on the networking systems of Generalitat Valenciana we found that a window width of 300 seconds, together with a similarity of 0.7 and entropy threshold set to 2, were the parameter values preferred by the security experts. The intuition for enlarging so much the window width is the following: we found that for small window widths some of the clusters were removed from the window too early in the process; yet, these clusters would reappear some seconds later containing a very similar bunch of alerts. Indeed this corresponds to observing very big cluster that can never fit into a too small window and thus, is split into several smaller overlapping clusters as the too small window slides through. The set of different type of clusters we found with these parameters can be seen in Table 1.

Even if these parameters are not easy to optimize, one sees there is a very stable behaviour of the algorithm, with a certain ``continuity'' whereby the outcome of slight changes in the parameter settings is reflected in rather stable output. However, it must be noted that the level of abstraction of the algorithm with respect to the network under surveillance is not high, and this results in a high variability of the results of the algorithm with respect to the network on which it has been implemented. Thus, relatively little changes in the network may easily result in a substantial loss of performance and a need of retuning the system.

Future work

This framework offers several opportunities for improvement, given that the strategy of letting clusters grow and inducing them to split under specific entropy considerations indeed gives very good results as of decreasing the load of information to be handled by the security experts in charge.

First, beyond adjustment of parameters, variations in the very similarity function could be explored, since our decision here for starting this work was a rather simplistic one.

One additional major issue is the possibility of using the information derived along the clustering to rank the priority of the groups of alerts, in that evolution of the corresponding cluster may suggest alerts that require immediate attention in much higher degree than others. Data coming from the vulnerabilities database must be treated also, since, for instance, an attack that clearly targets a vulnerability that has been recently fixed is of much less priority than even mild potential attacks to an existing vulnerability.

Acknowledgements

This work was supported by Spanish Government under the MITYC grant PROFIT FIT-360000-2005-46. The main part of this work was done while the first author worked at the Universitat Politècnica de Catalunya, Barcelona, Spain.

References

1. T. Bass. Intrusion detection systems and multisensor data fusion. Commun. ACM, 43(4):99–105, 2000.

2. F. Cuppens. Managing alerts in a multi-intrusion detection environment. In 17th Annual Computer Security Applications Conference, page 22, 2001.

3. H. Debar, M. Dacier, and A. Wespi. Towards a taxonomy of intrusion detection systems. Comput. Networks, 31(9): 805–822, 1999.

4. H. Debar and A. Wespi. Aggregation and correlation of intrusion-detection alerts. In 4th Int Symp on Recent Advances in Intrusion Detection, pages 85–103, 2001.

5. K. Julisch. Clustering intrusion detection alarms to support root cause analysis. ACM Trans. Inf. Syst. Secur., 6(4):443–471, 2003.

6. T. Lane and C.E. Brodley. Sequence matching and learning in anomaly detection for computer security. In AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management, pages 43–49, 1997.

7. W. Lee and S.J. Stolfo. A framework for constructing features and models for intrusion detection systems. ACM Trans. Inf. Syst. Secur., 3(4):227–261, 2000.

8. W. Lee, S.J. Stolfo, and P.K. Chan. Learning patterns from unix process execution traces for intrusion detection. In AAAI-97 Workshop on AI Approaches to Fraud Detection and Risk Management, pages 50–56, 1997.

9. W. Lee, S.J. Stolfo, and K. Mok. A data mining framework for building intrusion detection models. In IEEE Symp. on Security and Privacy, pages 120–132, 1999.

10. A. Valdes and K. Skinner. Probabilistic alert correlation. In 4th Int Symp on Recent Advances in Intrusion Detection, pages 54–68, 2001.

11. F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer. A comprehensive approach to intrusion detection alert correlation. IEEE Trans. Dependable Secur. Comput., 1(3):146–169, 2004.