A Correlation Approach to Intrusion Detection

Massimo Ficco[1] and Luigi Romano[2]

[1] Dipartimento per le Tecnologie, Universita' degli Studi di Napoli "Parthenope"
Centro Direzionale di Napoli, IT
[2] Laboratorio ITeM, Consorzio Interuniversitario Nazionale per l'Informatica (CINI)
Via Cinthia - Edificio 1, 80126 Napoli, IT
massimo.ficco@consorzio-cini.it,
luigi.romano@uniparthenope.it

Abstract. In this paper we discuss the limitations of current Intrusion Detection System technology, and propose a hierarchical event correlation approach to overcome such limitations. The proposed solution allows to detect attack scenarios by collecting diverse information at several architectural levels, using distributed security probes, which is then used to perform complex event correlation of intrusion symptoms. The escalation process from intrusion symptoms to the identified target and cause of the intrusion is driven by an ontology.

Keywords: detection, fusion, correlation.

1 Introduction

In the last years, Intrusion Detection System (IDS) has emerged as the main technology to protect information systems. The IDSs do not directly detect intrusions, but only the attacks symptoms. Different works [1,2] have observed that IDSs can generate thousands of alerts per days, up to 99% of which are false positives, that make it very difficult to identify the real attacks in progress over the system.

The intrusion detection methods can be divided into two categories: misuse- and anomaly-based [3]. Anomaly detection consists in comparing the observed beaviour with a reference "expected" beaviour. Any deviation between the two beaviours triggers an alarm. Misuse detection solutions consist of comparing the observed beaviour with a reference defining known attack sets. Both types of detection methods are characterized by a number of false positives and false negatives [4]. Misuse detection methods have the advantage of identifying specific attacks, with a few false positives. However, they only enable the detection of symptoms of known attacks forcing the security administrators to regularly update their signature sets. On the other hand, the anomaly-based solutions present the advantage of a great generalization capability, which leads to the ability of detecting new attacks. A major drawback is that no evidence is provided about the root cause of the monitored anomaly (*i.e.*, the reason for which it occurs): (*i*) no diagnosis is performed in order to help the administrator to

P. Chatzimisios et al. (Eds.): MobiLight 2010, LNICST 45, pp. 203–215, 2010.

identify whether an alert is a false positive or not; and (*ii*) in the case of a true positive, no information is provided about the attack that has led to the anomaly [5]. Finally, none of the two methods provides any support to recognize complex attack scenarios, *i.e.*, mechanisms to understand the relationship among sequence of different attacks [6].

In order to overcome the above mentioned limitations, in this paper we propose a solution, which exploits a hierarchical event correlation process based on diversity both in information sources and methods used to detect malicious activities. Different detection methods are adopted to collect streams of information at several architectural levels (*i.e.*, network, operating system, data base, and application), using multiple security probes, which are deployed as a distributed architecture. In order to increase the detection coverage and reduce the time and the cost of managing the large number of false positives generated by probes, a "*clustering*" correlation approach is adopted to aggregate attack symptoms (based on similarity among symptoms attributes [7]), and rearranging the resulting alarms based on their confidence levels [8]. The confidence indicates the likelihood that the correlated alarms are symptomatic of an ongoing attack on the system. It is estimated on the base of "weights" and "thresholds" that provide points of operation that offer the best compromise between the false positives and false negatives. In order to recognize complex attack patterns and enrich the semantics of alerts, a "*causal*" correlation approach is adopted, which captures the causal relationships among the resulting alarms (which can represent intermediate attacks of a more complex attack scenario), by correlating them on the base of temporal and logical constraints [9]. The correlation capability is driven by an ontology (specified in OWL). It is used to recognize attack scenarios and to identify the cause of the symptoms, which are discovered.

Finally, we show that the proposed solution is able to detect both attacks that are characterized by a single malicious activity, called intermediate attacks (*e.g.*, SQL injection), and complex attacks that consist of a specific sequence of malicious activities perpetrated by the attackers in order to discover system's vulnerabilities. We conducted two sets of preliminary experiments on a testbed consisting of web servers running different well known open source content management systems. The former set of experiments exposed the most serious vulnerabilities [10] of the web applications, which are described in the Bugtraq repositories. By using a complex event processing technology, that correlates different alarms on the base of temporal and logical conditions, the latter experiments show the capabilities of the proposed solution to recognize attack scenarios on the fly, as they occur. The experimental tests have shown that our approach results in a better performance of the IDS, in terms of reducing the false positive alarms rate and increasing detection capacity, as well as more accurate identification of the nature of the attack and the specific system component affected by it.

2 Related Work

In order to improve the attack detection rate, enrich the semantics of alarms, and reduce the overall number of false alarms, different works propose explicit

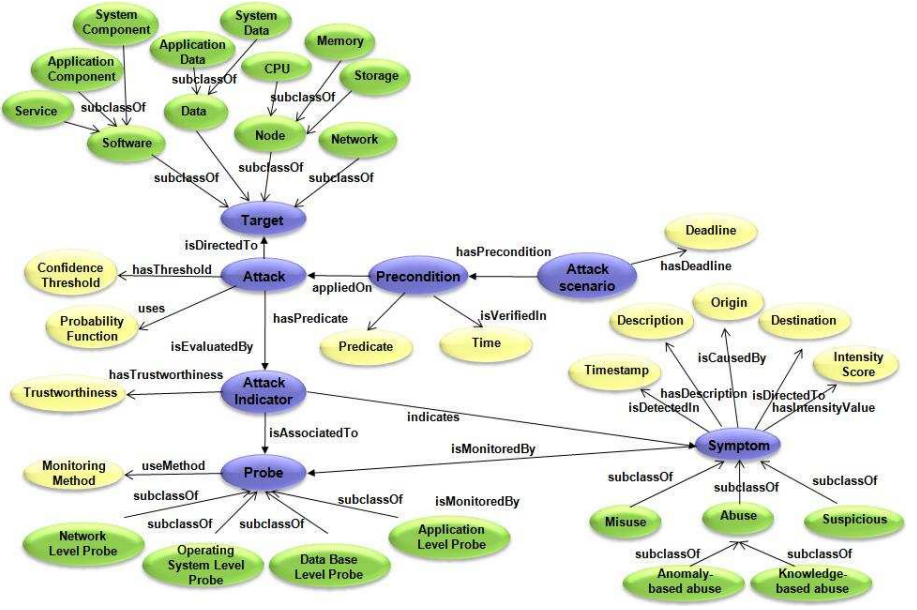

Fig. 1. A simplified view of the proposed ontology

alarm correlation approaches. In particular, Valeur et *al*. [11] proposes a correlation workflow intended to unify the various steps of the correlation process. Our work is strongly inspired by the this framework, but we propose a weight-based correlation approach, in which each monitored symptom is weighted on the base of the trustworthiness of a probe to monitor a specific attack. Majarczyk et *al*. [12] propose an intrusion detection architecture based on COTS diversification applied to Web applications. The authors adopt an approach for anomaly detection using redundancy and diversification techniques. Although, the proposed solution provides a high attacks detection, and a low level of false positives, no diagnostic function is defined in the system. Moreover, they have only restricted experiments to the contents of static http requests/responses. Both Haibin et *al*. [13] and [12] adopt correlation approaches that combine only events that represent the independent detection of the same attack occurrence by different probes. They have not proposed any mechanism for the recognition of attack scenarios. Morin et *al*. [9] proposed a misuse correlation approach based on the chronicles formalism, which is a high level declarative language to describe temporal patterns that represent possible evolutions of attack scenarios. On the other hand, the use of raw security data (attack's symptoms) in the correlation process, implies the need to define a large number of correlation signatures computationally complex to perform. In our work, symptoms are before aggregate in high-level messages, which represent intermediate attacks, and then they are correlated to recognize the attack scenario.

3 Ontology-Based Approach

The proposed ontology drives the correlation process in order to recognize more complex attack scenarios, as well as to aggregate symptoms produced by different probes with the final goal of identifying one or more intermediate attacks as likely causes of observed symptoms. Ontology is also used to automate the process of deriving queries for diagnostic analysis. Figure 1 presents a simplified view of our security ontology, which is implemented in Web Ontology Language (OWL). Each kind of attack can be related to a set of potential symptoms. More specifically, an attack is described by using *attack indicator*, that estimates the trustworthiness of the probe to detect the specific symptom. *AttackIndicator* has the properties: (*i*) *hasTrustworthinessValue*, which is defined by the concept of likelihood that the observed feature is a symptom of the considered attack, (*ii*) *isAssociatedTo*, which is defined by the concept *Probe*, and (*iii*) *indicates*, which is defined by the concept *Symptom*. *Symptoms* are classified into Abuses, Misuses and Suspicious Acts. *Abuses* represent actions which change the state of a system's asset. They are further divided into *Anomaly-* and *Knowledge-based*. The former represents anomalous behaviors (*e.g.*, unusual application load, anomalous input requests), whereas the latter is based on the recognition of signatures of previously known attacks (*e.g.*, brute force attacks). *Misuses* represent out-of-policy behaviors in which the state of the components are not affected (*e.g.*, authentication or queries failed). *Suspicious Acts* are not violations of policy, but events of interest to the system administrator (*e.g.*, commands providing information about the system state). Each *Symptom* is characterized by the property *hasIntensityScore*, which reflects the probability of occurrence of the given symptom with regards to the specific monitoring method. *Probes* are characterized by *Monitoring Method* property, which defines the method used to monitor the symptom. Moreover, they are also classified depending on the specific architectural level to which they belong, namely *Network Level Probe*, *Operating System Level Probe*, *Data Base Level Probe* and *Application Level Probe*. A similar classification is adopted for *Target*, that refers to anything that should be protected: *Network*, *Node*, *Data* and *Software*. Finally, *Attack Scenario* identifies the correlation rule for the attack scenario recognition. It has proprieties *hasDeadline* and *hasPrecondition*. The former defines the time window by which monitoring the attack scenario. The latter defines the *Precondition*, which describes the conditions required for including an intermediate attack in the considered scenario. The *Precondition* has the properties: *Time*, which is the temporal condition for attack detection of an intermediate attack, and *Predicate*, which corresponds to a numerical or logical condition over the attack instance to be verified.

4 The Detection and Diagnosis Process

The proposed process produces a diagnostic report that shows what parts of the system are under attack, and what kind of attacks the system is experiencing. In the following, the steps performed by the process are described.

- **Monitoring:** Distributed probes observe different attack symptoms by using specific monitoring methods. For each symptom, the method computes a probability value, named *Intensity Score* (IS), which reflects the likelihood that the observed symptom represents a malicious behavior.
- **Classification:** Symptoms are analyzed, filtered and aggregated in categories. Symptoms are classified on the base of the concepts presented in the proposed ontology (*i.e.*, misuses, anomaly-based, knowledge-based, and suspicious acts).
- **Normalization:** Since each probe can provide security information with different representations or formats, every monitored symptom is coded and normalized in a standardized format, as well as augmented with additional information, such as timestamp, probe identifier, source and target of the monitored anomalous behavior.
- **Fusion:** It receives the classified symptoms and aggregates them by using *clustering-based* correlation rules. They combine symptoms based on the 'similarity' among their attributes. In this work we consider aggregation based on the attack type, the target component, and the temporal proximity, *i.e.*, we combine different symptoms of the same attack occurrences, which are directed to the same target. For each monitored target, the ontology identifies the symptoms to aggregate for each potential attack associated with that target. The temporal proximity is based on a time window Tc, that is specified as a parameter by the administrator. We are aware that a temporal order requires that all clocks at probes be synchronized. We do not address this issue here, since this problem is out of scope of this work. Several approaches can be used, *e.g.*, using a total ordering based on timestamps. At the end of this phase, for each target, an event $E(k) = \{e_{A1}(k), ..., e_{Am}(k)\}$ is generated. For each possible attack A_i, $e_{Ai}(k)$ contains the symptoms correlated during the time window k.
- **Ranking and filtering:** In order to reduce the number of false positives produced during the fusion step, we adopt an approach based on the *confidence* (C) of the events. Assuming that $e_{Ai}(k) = \{s_1, ..., s_z\}$ is the set of correlated symptoms related with the attack of type Ai during the time window k, the confidence is the likelihood that the monitored symptoms represent an underlining attack of such a type. C_{Ai} is computed using Function 1.

$$C_{Ai}(k) = \sum_{s \in e_{Ai}(k)} \omega_p(s) * IS_s(k) \quad with \;\; \omega_p \leq 1 \tag{1}$$

The confidence C_{Ai} depends on the weights ω_p and the intensity scores ISs of the correlated symptoms. Then the observed features ISs are normalized to zero mean and unit variance. $\omega_p(s)$ is associated with trustworthiness of the probe p to monitor a symptom of the attack Ai. It is assigned on the base of a prior knowledge of the effectiveness of monitoring method being used for the given attack type. Although simple in implementation, choosing proper weights is of critical importance to highlighting the proper features under various attacks. A mapping function normalizes the calculated C_{Ai} values into a value on a scale of 0 to 255. The events ordered on the base of the C_{Ai}

given an indication of most likely cause of detected anomalous behaviors at the step k (intermediate attacks). Finally, the events are subjected to a filtering. If the confidence does not exceed a threshold (specific of each attack) estimated during a validation phase, the event will be discarded (*i.e.*, it is considered as a false positive).

- **Correlation:** The goal of the correlation is to identify complex attack scenarios. In the intrusion detection literature, attack scenario (or attack pattern) is a sequence of explicit attack steps (intermediate attacks), which are logically linked and lead to an objective. In our work, an attack scenario is modeled by a *causal-based* correlation rule. It consists of a set of preconditions, which are logical conditions on the intermediate attack alarms (IAA). Preconditions are linked together by temporal constraints. Each precondition is described by predicates, which are conditions to be verified (*e.g.*, the number of IAAs of type *Ai* must be greater than 5). A deadline is fixed for the recognition of each scenario. The IAAs can be shared by many attack scenarios. The integration of an IAA A_i in an attack pattern recognition depends on the IAA's timestamp t_i, the previously integrated IAAs, and the predicates p_i. An instance of an attack scenario can be described by the following representation:

```
Attack scenario {
    alarm(A1,t1,p1.1,p1.2,p1.3,...);
    alarm(A2,t2,p2.1,p2.2,...);
    alarm(A3,t3,p3.1,...);
    alarm(A4,t4,p4.1,...);
    t1<t2<t3<t4;
    t4-t1 < deadline; }
```

  During the recognition process many partial instances of attack scenarios must be managed. If a predicate is violated, or if a deadline expires, then the instance is dropped, and their constitutive IAA are either correlated to other scenarios or provided to the administrator individually. When a complete match is found, an attack scenario instance is recognized and an alarm is triggered.
- **Diagnosis:** If an attack scenario instance is recognized, a compact report is built and shown to the administrator. This report is hierarchically structured. The leaf nodes are the symptoms monitored by probes, and the root nodes are the higher level alarms (related to an attack scenario). The intermediate nodes represent IAAs. For each IAA, the report contains a summary of the information produced during the previous phases, including the attack type, the confidence, and the target of the attack. Only the hierarchies root are directly shown to the administrator (*i.e.*, the symbolic name of the recognized scenario, and the final target of the attack). If detailed information about the alarms are required, the administrator can browse the alarms hierarchy. Such a recognition process contributes to alarm volume reduction since only one alarm (the recognized scenario) is provided to the administrator instead of each individual alarm (*i.e.*, the intermediate attack). However,

as described in [9], not all attacks can be modeled using scenarios. Firstly, the relevance of an attack scenario is questionable, because many (unpredictable) pattern may lead to a given attack objective. Secondly, it is hard to specify quantitative time constraints in scenarios, because the time gaps among each step may vary a lot, depending on how hurried the attacker is. Moreover, attack can simply consist of a single malicious action (*e.g.*, a single SQL injection attempt). Therefore, alarms that cannot be merged to any scenario, are provided to the administrator individually and rearranged on the base of their confidence levels.

5 Case Study, Experimental Setup and Results

In this section, we present preliminary results obtained by applying the proposed approach in a laboratory experimental setup. In a first set of experiments, we present an example of the detection and diagnosis process of an intermediate attack. It shows that the aggregation of information collected at several architectural levels, using multiple security probes, allows to improve the detection capability and reduce false positives. In a second set, we present an example of attack pattern recognition. The experimental setup consist of the most well-known open source content management systems written in PHP, including Joombla (v.1.5), phpNuke (v.6.0, v.7.1), Mambo (v.4.5), and Drupal (v.4.7). They run on an Apache v1.3.34 web server, and use PHP v4.4.2 and MySQL v4.1.11.

5.1 Detection and Diagnosis of Intermediate Attacks

In order to validate the proposed process, we consider two classes of attacks: SQL Injection Attacks (SQLIA) and Cross Site Scripting (XSS). They are the most frequent (and most serious) classes of attacks for web applications [10]. For the considered applications, such vulnerabilities are described in the SecurityFocus advisories with Bugtraq IDs 10749, 10741, 15421, 9879, 3609, 13966, 26735, as well as in the Secunia Advisories Bugtraq IDs SA30461, SA31126, SA30922, SA21859, SA10413.

Monitoring and fusion. In order to monitor the attack symptoms we use different probes, which use either anomaly detection methods (AD) or misuse detection methods (MD). In AD methods a value (*Intensity Score*) is assigned to the generated events, which reflects the intensity of the given anomaly with regard to an established profile. For each observed feature, the AD approach can perform in one of three phases: training, validation, and testing. In the training phase data sets are used to parameterize the monitoring method (necessary to determine the characteristics of 'expected' beaviour). The suspicious requests are manually extracted in order to guarantee that the data sets are (as much as possible) attacks free. These requests are used during the testing phase to evaluate the used method. The validation phase aims at validating the detection models and to establish thresholds, that are used to distinguish between regular and anomalous behaviors during the testing phase. In the testing phase the AD

method is used to monitor anomaly behaviors with respect to the desidered profile computed during the training phase. The choice of a proper threshold value is the main problem in this process, since there is a trade-off between the number of false positives and the expected detection accuracy: a low threshold can result in many false positives, whereas a high threshold can result in many false negatives. Once the profiles and thresholds have been derived, the testing phase is operated. If the computed intensity score exceeds the fixed threshold a message is triggered. As for *MD* methods, since they also produce false positives, the intensity score of the generated messages is fixed to a value from 0 to 1. It is estimated during a validation phase, and represents the likelihood to monitor correctly an attack symptom. Table 1 shows the adopted probes, the relevant monitored features during the attacks, the related information sources, and the models/methods used to monitor symptoms.

Table 1. Features evaluated

Probe	Observed featurs	Source	Method/Model
AD-CD	Character distribution	HTTP traffic	AD: Chi-Square test
AD-QL	Query request length	HTTP traffic	AD: Chebyshev inequality
MD-QF	Query failed	Data base log	MD: Pattern recognition
AD-WR	Web response length	Web server log	AD: Chebyshev inequality
MD-SM	Signature of the HTTP request	HTTP traffic	MD: Signature matching
MD-TM	Content of the request	HTTP traffic	MD: Tag matching
AD-QS	Query syntax	Data base log	AD: Model-based

We briefly describe the considered features and the models adopted to perform the monitoring methods. The first four methods are *AD* methods detail described in our previous work [15].

- *Character Distribution (AD-CD)*: It is a method used to capture the concept of 'normal' query's attributes contained in the GET and POST HTTP request sections (*i.e.*, identifies anomalous character distribution). It is based on the *chi-square* function.
- *Query Length (AD-QL)*: It adopts a *Chebyshev Inequality* function to estimate an approximation of the actual distribution of the query's attributes length and detects suspicious inputs that significantly deviate from the observed normal behavior.
- *Queries Failed (MD-QF)*: During a time slicing window, it estimates abnormal rate of queries failed with respect to the normal operational model. In the validation phase, for each database table is fixed a threshold failures rate.
- *Web Response (AD-WR)*: It is based on a model similar to the one used for AD-QL monitoring in order to detect anomalous size of the page generated against the same request.
- *Signature Matching (MD-SM)*: It examines HTTP requests and determines whether a signature of previously know attacks is matched. It is based on Scalp IDS [16].

- *Tag Matching (MD-TM)*: It simply looks for trivial script tags present in the GET and POST HTTP request sections, such as "<script>", "<img scr=", "<b>", "<u>". Moreover, it monitors specific meta-characters, and injected string such as 'union', 'insert', 1'or'1'='1 and their hexadecimal equivalents.
- *Query Syntax (AD-QS)*: It performs a syntax-aware evaluation of the query string before it is executed, similar to that proposed by [10]. It is a model-based approach used to identify anomalous queries that might be associated with an attack. The query string is parsed to break the string into a sequence of tokens that correspond to SQL keywords, operators, and literals. Then an iteration process checks whether tokens contain trusted data. If all tokens do not pass this check, the query is considered unsafe.

The evaluation of the proposed approach was performed using different data sets from production servers at the university in which the considered applications run. During the training phase, we considered the legitimate requests of three months of real traffic. A validation set is used to validate the models, and to estimate thresholds. For each application, during the testing phase we created a synthetic data set in which a number of attacks (collected on the web), their obfuscated variations (using hexadecimal characters) and normal requests (the traffic of the last month) were introduced. For each application, Table 5.1 shows the total number of normal HTTP query requests and the illegitimate requests (injected during the testing phase by Apache Jmeter [17].

Table 2. Requests for considerate data sets

Applications	Legitimate Requests	Illegitimate Requests SQLIA	XSS
Joomla	372.732	45	24
phpNuke	201.318	36	16
Mambo	129.786	27	21
Drupal	545.760	41	14
Total	**1.249.596**	**149**	**75**

Experimental evaluation. For each possible attack to the monitored target, the fusion step aggregates symptoms on the base of the schema defined by the ontology and their temporal proximity. Figure 2 shows the monitoring and fusion results of the XSS and SQLIA attacks. For each method, the percentage of detected attacks (DAs), and the number of requests that have been classified as false positives (FPs) are provided. These results represent the capacity of each method to monitor potentially malicious behaviors compared with the fusion process results. Results show that each method does not achieve a detection rate of 100%, and presents many false alarms. For example, the AD-WR method presents the highest number of FPs, whereas AD-CD presents the highest rate of detected attacks. MD-TM produces the lower rate of false positives.

As one might expect, the use of multiple probes in the fusion process increases the number of FPs (about 18% compared to AD-WR). On the other hand, the percentage of DAs is increased (100% for SQLIA and 98% for XSS).

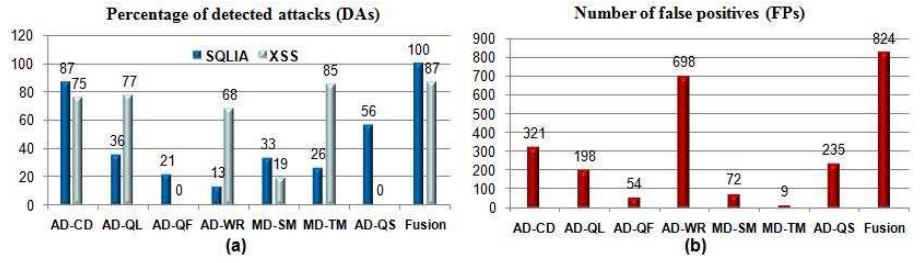

Fig. 2. Fusin process results

Ranking and diagnosis. After fusion, the resulting alarms are ordered based on their confidence C_{Ai}. If the estimated confidence value exceeds a fixed threshold C_T, an IAA alarm is triggered. C_{Ai} is computed using Function 1 with a time window equal to two minutes and $n = 1$ (for each method). For each attack, Table 3 shows the levels of trustworthiness ω_p of the probes inferred during the validation phase. We considered five levels: NULL (0.0), VERY LOW (0.1), LOW (0.3), MEDIUM (0.5), HIGH (0.7) and VERY HIGH (0.9). Figure 3 reports the number of DAs and FPs with respect to the confidence values. It can be noted that by choosing a confidence threshold value C_T of 75, it is possible to reduce the FPs rate of about 98% (*e.g.*, events that are not correlated with other events, or present low intensity scores), while leaving the number of DAs unaffected (only 2% reduction). In order to evaluate the ability of our approach to detect and classify alarms (diagnosis capability), two traditional indexes are considered: recall and precision. *Recall* represents the likelihood that a specific

Table 3. Probes trustworthiness

Probes	Trustworthiness	
	SQLIA	XSS
AD-CD	HIGH	MEDIUM
AD-QL	LOW	HIGH
MD-QF	HIGH	NULL
AD-WR	VERY LOW	MEDIUM
MD-SM	HIGH	MEDIUM
MD-TM	LOW	VERY HIGH
AD-QS	HIGH	NULL

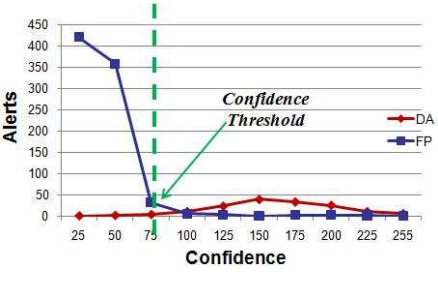

Fig. 3. Monitoring and ranking results

Table 4. The considered complex attack scenario

Attack/ Symptom	Description
Port scanning	A scanning procedure is adopted to trace the open ports on the considered server (*e.g.*, the port 80, 21, 22, 8080 and 3120 are discovered).
Telnet	The discovered port 80 is used to determine if a web server is listening on that port (*e.g.*, by a telnet request)
Directory traversal	By sending to the Apache server a custom request, consisting of a long path name created artificially by using numerous *dot-dot-slash*, the attacker gains a listing of the web server directory contents. During this step, a hidden web page (no public link to that page is published on the web) is discovered.
Policy violation & Buffer overflow	The attacker enters strings (*e.g.*, login data) with variable length as GET parameters. With very long strings an error message is displayed within the Web page. The page contains the login query performed in the data base.
SQL injection	Known the query structure (in terms of tables in which are located login information), the attacker performs attacks by POST requests.

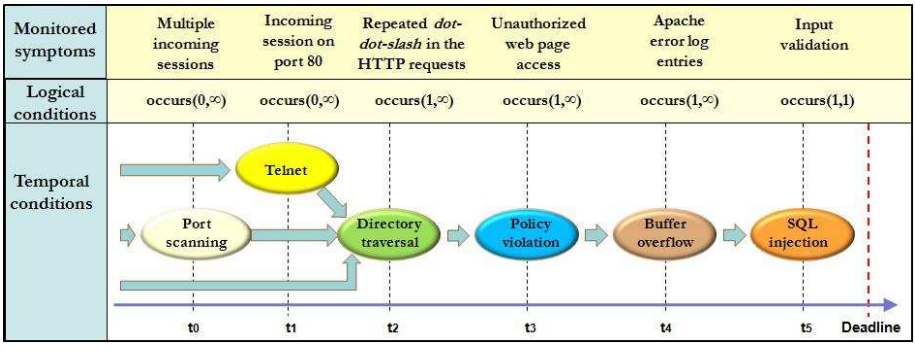

Fig. 4. The attack preconditions set

attack is correctly diagnosed (*i.e.*, the number of correctly diagnosed attacks *Ai* over the number of total injected attacks of such a type), while *precision* represents the likelihood that an alarm tagged as a detected attack is really an attack. As we presented in Table 3, different monitored features can be considered symptoms of both XSS and SQLIA attacks, but with different trustworthiness values. Thus, the experimental results show that, considering only alarms with a level of confidence greater then 75, the percentage of SQLIAs correctly diagnosed was about 93% of injected attacks (5% attacks are marked as XSS), whereas the percentage of XSS was about 88% (9% attacks are marked as SQLIA). Moreover, considering the same confidence threshold, our approach presents a precision equal to 94%.

5.2 Attack Scenario Recognition

During an attack scenario, it is more likely that the attacker first performs the knowledge gathering steps, which consist of a set of commands that enable him/her to gain knowledge about the target system, and then performs the intrusion. For example, in Table 1, we describe an attack instance, which consists

of a sequence of steps (intermediate attacks) followed by an attacker to discover and exploit a SQLIA vulnerability.

As graphically shown in Figure 4, the considered attack scenario can be modeled by a set of preconditions. They are specified by logical and temporal conditions (predicates), which are combined by using logical connectives (conjunctions and negations) among them. The presented set of preconditions can be coded in a complex query to be processed by a Complex Event Processor (CEP), such as [18,19]. In the current implementation we used Borealis [20].

6 Conclusions

In this work, we have discussed the limitations of current IDS technology, and proposed a hierarchical event correlation to overcome such limitations. We have conducted preliminary experiments on a testbed consisting of web servers running well known open source applications. The experimental tests have demonstrated that the use of a hierarchical correlation of different information: *i*) strengthens the diagnosis, *ii*) reduces the overall number of alarms, *iii*) improves the semantic content of the alarms, and *iv*) allows to detect attack scenarios. In future work, we will aim to define mechanisms for unknown attack patterns identification.

Acknowledgment

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement no. 225553 (INSPIRE Project).

References

1. Axelsson, S.: The base-rate fallacy and the difficulty of intrusion detection. ACM Trans. on Information and System Security 3(3), 186–205 (2000)
2. Manganaris, S., Christensen, M., Hermiz, K.: A data mining analysis of RTID alarms. Computer Networks 34(4), 571–577 (2000)
3. Hervé, D., Dacier, M.: Towards a taxonomy of intrusion-detection systems. The Journal of Computer and Telecommunications Networking 9, 805–822 (1999)
4. Kemmerer, R., Vigna, G.: Intrusion detection: a brief history and overview. IEEE Computer 35(4), 27–30 (2002)
5. Majorczyk, F., Totel, E., Mé, L.: Anomaly Detection with Diagnosis in Diversified Systems using Information Flow Graphs. In: IFIP International Federation for Information Processing. LNCS, vol. 278, pp. 301–315. Springer, Boston (2008)
6. Ning, P., Cui, Y., Xu, D.: Techniques and tools for analyzing intrusion alerts. ACM Trans. on Information and System Security 7(2), 274–318 (2004)
7. Julisch, K.: Clustering intrusion detection alarms to support root cause analysis. ACM Trans. on Information and System Security 6(4), 443–471 (2003)
8. Yu, D., Frincke, D.: Alert Confidence Fusion in Intrusion Detection Systems with Extended Dempster-Shafer Theory. In: Proc. of the 43rd ACM Southeast Regional Conference, vol. 2, pp. 142–147 (May 2005)

9. Morin, B., Debar, H.: Correlation of intrusion symptoms: An application of chronicles. In: Vigna, G., Krügel, C., Jonsson, E. (eds.) RAID 2003. LNCS, vol. 2820, pp. 94–112. Springer, Heidelberg (2003)
10. The OWASP Top 10 Web attacks (December 2009), http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
11. Valeur, F., Vigna, G., Kruegel, C.: A Comprehensive Approach to Intrusion Detection Alert Correlation. IEEE Transactions on Dependable and Secure Computing 1(3), 146–169 (2004)
12. Totel, E., Majorczyk, F., Mé, L.: COTS Diversity Based Intrusion Detection and Application to Web Servers. In: Valdes, A., Zamboni, D. (eds.) RAID 2005. LNCS, vol. 3858, pp. 43–62. Springer, Heidelberg (2006)
13. Haibin, M., Jian, G.: Intrusion Alert Correlation based on D-S Evidence Theory. In: Proc. of the 2th Int. Conf. on Communications and Networking (CHINACOM 2007), pp. 377–381. IEEE CS Press, Los Alamitos (August 2007)
14. Bondavalli, A., Ceccarelli, A., Falai, L.: Assuring Resilient Time Synchronization. In: Proc. of the IEEE Symposium on Reliable Distributed Systems (SRDS 2008), pp. 3–12. IEEE CS Press, Los Alamitos (October 2008)
15. Ficco, M., Coppolino, L., Romano, L.: A Weight-Based Symptom Correlation Approach to SQL Injection Attacks. In: Proc. of the 4th Latin-American Symposium on Dependable Computing (LADC 2009). IEEE CS Press, Los Alamitos (September 2009)
16. Scalp: Apache log analyzer, http://code.google.com/p/apache-scalp/ (last update September 2009)
17. JMeter: Java application designed to load test web applications, http://javaboutique.internet.com/tutorials/JMeter/
18. Coral8 Engine,, at http://www.aleri.com/sites/default/files/assets/product_literature/Coral8%20Engine.pdf (last access October 2009)
19. Oracle CEP, http://www.watersonline.com/public/showPage.html?page=800767 (last access December 2009)
20. The Borealis project, http://www.cs.brown.edu/research/borealis/public/ (last access February 2010)