From Intrusion Detection to Intrusion Detection and Diagnosis: An Ontology-Based Approach

Luigi Coppolino, Salvatore D'Antonio, Ivano Alessandro Elia, and Luigi Romano

Dipartimento per le Tecnologie - Università degli Studi di Napoli "Parthenope"
{luigi.romano,luigi.coppolino, salvatore.dantonio,ivano.elia}@uniparthenope.it
http://www.dit.uniparthenope.it/FITNESS

Abstract. Currently available products only provide some support in terms of Intrusion Prevention and Intrusion Detection, but they very much lack Intrusion Diagnosis features. We discuss the limitations of current Intrusion Detection System (IDS) technology, and propose a novel approach - which we call Intrusion Detection & Diagnosis System (ID^{2}S) technology - to overcome such limitations. The basic idea is to collect information at several architectural levels, using multiple security probes, which are deployed as a distributed architecture, to perform sophisticated correlation analysis of intrusion symptoms. This makes it possible to escalate from intrusion symptoms to the adjudged cause of the intrusion, and to assess the damage in individual system components. The process is driven by ontologies. We also present preliminary experimental results, providing evidence that our approach is effective against stealthy and non-vulnerability attacks.

Keywords: Intrusion Detection and Diagnosis, Information Diversity, Ontologies, Stealthy and non-vulnerability attacks.

1 Rationale and Contribution

By Diagnosis, we mean the capability of: i) clearly identifying the causes of the attacks, and ii) accurately estimating their consequences on individual system components. Currently available products only provide some (indeed limited) support in terms of Intrusion Prevention and Intrusion Detection, but they very much lack Intrusion Diagnosis capabilities. We strongly believe that this technology trend should be subverted, and that more efforts should be put in the development of effective techniques for implementing Intrusion Diagnosis features.

We propose a novel approach, which extends Intrusion Detection System (IDS) technology to what we call Intrusion Detection & Diagnosis System (ID^{2}S) technology, to overcome this limitation. The basic idea is to collect information at several architectural levels (namely: Network, Operating System, Data Base, and Application), using multiple security probes which are deployed as a distributed architecture, and use Complex Event Processing (CEP) technology to perform sophisticated correlation analysis of intrusion symptoms. The idea of collecting information from different sources to gain more insight into attack/intrusion related phenomena is not new. A (far from complete) list of remarkable works is: [3], [4], and [5]. While the presented works exploit the concept of correlation and multilayer analysis, they do not address the issue of diagnosing the kind of anomaly or attack the system is experiencing. In our approach, the escalation process from intrusion symptoms to the adjudged cause of the intrusion and to an assessment of the damage in individual system components is driven by ontologies. More precisely, we have developed two sets of ontologies: the first one allows us, given that a particular symptom has been observed, to identify which are the attacks that have possibly generated that symptom; the second one can be used to infer an estimate of the damage to specific system components from knowledge of the attack. The output of the process can then be used to drive remediation actions, and ultimately replenish system resources. To demonstrate the effectiveness of our approach, we have conducted preliminary experiments on a testbed consisting of a web server running Joomla, the well known open source Content Management Systems (CMSs) written in PHP. The experiments exposed vulnerabilities to SQL injection (SQLi) and Cross Site Scripting (XSS) attacks of the target applications, which are described in the Bugtraq repository. The experimental tests have demonstrated that our approach leads to better detection results, both in terms of improved accuracy of the classification process and of enhanced reliability of the decision-making process. Also importantly, we were able to clearly identify the nature of the attack, as well as the specific system components which are affected by it. We emphasize that the proposed approach is effective against an emerging class of new attacks, which is referred to in the literature [1] as "stealthy". These attacks represent a major threat, since not only they have a dramatic impact in terms of economic losses [2], but (i) they also are invisible to current State-Of-The-Art IDSs, and (ii) current Intrusion Prevention Systems are ineffective against them. Instead, since such attacks have clear symptoms at architectural levels other than the network, by collecting information also at multiple architectural levels using diverse security probes, and performing sophisticated correlation analysis of attack symptoms, we are able to detect them. The rest of the paper is organized as follows. Sect. 2 describes the approach we propose, and how it is currently implemented in the framework of the INTERSECTION and INSPIRE projects. Sect. 3 describes the ontology-based detection and diagnosis process. Sect. 4 provides a description of the case study and presents preliminary experimental results. Finally, Sect. 5 gives some concluding remarks, along with information concerning the directions of our future work.

2 Conceptual Architecture of the ID^{2}S Technology

In order to effectively assess the security status of a networked system, the results of the monitoring activities performed at different observation points need to be correlated. Such observation points are distributed throughout the network

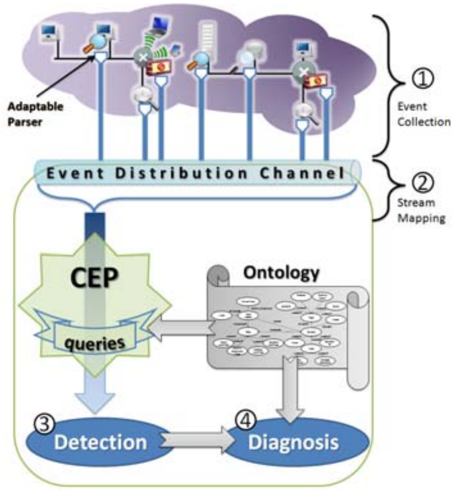

Fig. 1. Conceptual architecture of the ID^{2}S technology

as well as throughout the system to be protected. The more diverse the information sources and the processing methods, the more effective the correlation process. The deployment of probes at different observation points located in the networked system and at different architectural levels (network level, operating system level, application level, etc.), allows to fulfill the requirement of diversity of the information sources. By exploiting information diversity, it is possible to improve the accuracy of the detection process, as well as to implement diagnostic capabilities.

Fig. 1 shows the architecture of the proposed Intrusion Detection and Diagnosis System, which comprises the following functional blocks:

- Event Collection - Collecting security-related events from a wide range of detection probes implies dealing with heterogeneous data sources. To cope with heterogeneity, a promising solution is the use of Adaptable Parsers (APs). APs extract security-related information from multiple data feeds, and convert it to IDMEF (Intrusion Detection Message Exchange Format) standard messages, which are then routed to the Event Distribution Channel. A system component, called Decision Engine (DE), is in charge of processing data available on the Event Distribution Channel, and of correlating them, in order to take a decision on whether the collected symptoms represent an actual attack or not. The current implementation of AP components relies on Java Compiler Compiler (JavaCC) technology. More details on adaptable parsers are available in [8].
- Stream Mapping - This function converts IDMEF messages describing the symptoms of a possible intrusion to streams of tuples, which are then fed to a Complex Event Processing (CEP) engine. CEP technology enables the Decision Engine to perform sophisticated correlations on information gathered

from the multiple security probes. The current implementation of the CEP engine is based on Borealis [10].

– Detection - This phase consists in the extraction of higher level knowledge from situational information. This is done in real-time or near real-time.
– Diagnosis - An ontology-based hierarchical organization of event patterns is used to automate the process of deriving the queries which implement the diagnostic analysis. The knowledge formalized by the ontology is used by the diagnostic process to identify which intrusions/attacks are the possible cause of the observed symptoms, and which part of the system has been affected by the malicious activity.

3 Ontology-Based Detection and Diagnosis

The Decision Engine is in charge of detecting ongoing attacks by analyzing and correlating the security-related events which have been conveyed to the Event Distribution Channel. The queries are formulated based on the information and knowledge contained in the threat ontology.

Fig. 2 presents a high level view of such an ontology. Properties of concepts and sub-concepts (denoted by ellipses) are not shown because they would make the ontology unwieldy.

An event generated by a probe is considered as a potential symptom of an attack against a specific target. Each kind of attack is associated to a set of symptoms. An attack is described by using a number of indicators that assess the trustworthiness of the probe used to detect it.

Attack has the *isEvaluatedBy* property, which is defined by the *AttackIndicator* concept. *AttackIndicator* has the following properties: (i) *hasTrustworthinessValue*, which is defined by the *Trustworthiness* concept; (ii) *isAssociatedTo*, which is defined by the *Probe* concept, and (iii) *indicates*, which is defined by the *Symptom* concept. *Symptoms* are classified into *Abuses*, *Misuses* and *Suspicious Acts*.

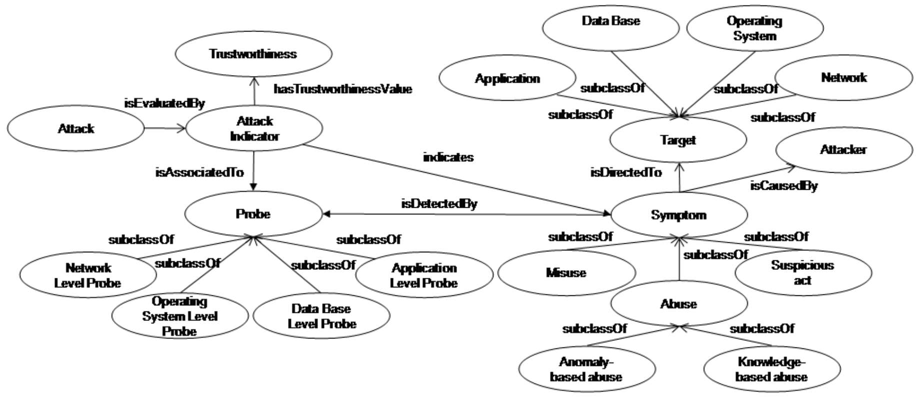

Fig. 2. The Threat Ontology

Abuses are actions which change the state of an asset. They are further divided into *Anomaly-based* and *Knowledge-based Abuses*. The first category of abuses includes anomaly behaviors (*e.g.*, unusual application load, anomalous input requests, etc.), while the second category relies on the recognition of signatures of well-known attacks (*e.g.*, brute force attacks). *Misuses* are out-of-policy behaviors which do not affect the state of the system components (*e.g.*, authentication failures, query failures). *Suspicious Acts* do not violate any policy. They are events of interest to the probes (*e.g.*, execution of commands providing information about the system state). The *Symptom* concept has the following properties: (i) *isDetectedBy*, which is defined by the *Probe* concept; (ii) *isCausedBy*, which is defined by the *Attacker* concept, and (iii) *isDirectedTo*, which is defined by the *Target* concept. Each *Symptom* is characterized by the *hasDetectionTime* property, which specifies the detection time of the symptom, and the *hasIntensityScore* property, which measures the probability of occurrence of the symptom with reference to the specific probe detecting it. *Targets* are classified in 4 categories, *i.e.* Network, Operating System, Data Base and Application.

A software tool, named Query Generator, browses the threat ontology, extracts the properties characterizing an attack, and generates the queries to be executed by the Complex Event Processor.

Since the events feeding the distribution channel encompass information about both the attack symptom and the detection probe, the *Symptom* and the *Probe* concepts of the threat ontology can be used to build the query which drives the correlation process performed by the CEP engine. More precisely, the structure of the threat ontology shows that the *Attack Indicator (AI)* concept *indicates* a symptom and is *AssociatedTo* a probe. Therefore, events on the Event Distribution Channel can be considered as *AI* instances. While generating the queries, the Query Generator uses the *Attack Indicator* properties to define the query parameters which help detect the specific attack.

Fig. 3 shows the main functions that compose the process, namely: Classification, Aggregation, and Filtering.

Classification. The purpose of the classification function is to create a separate stream for each possible attack (*i.e.* every *Attack* described in the ontology), in order to make the subsequent aggregation function more efficient. Every attack is associated to a stream containing the *Attack Indicators*, *i.e.* the attack information, which should be used to detect the attack according to the *IsEvaluatedBy* relationship in the threat ontology.

The Classifier is implemented as a set of concatenated filter and map queries. A filter query extracts from the event descriptions on the Event Distribution Channel only the attack details that concern specific attack, while a map query converts the selected information to a stream of attack indicators.

Aggregation. Once, for every class of *Attack*, a stream has been created containing only relevant *AIs*, such indicators are aggregated in order to formulate hypothesis about the ongoing attacks. The aggregation is performed by combining subsets of *AIs*. The aggregation process scales with the number and the type

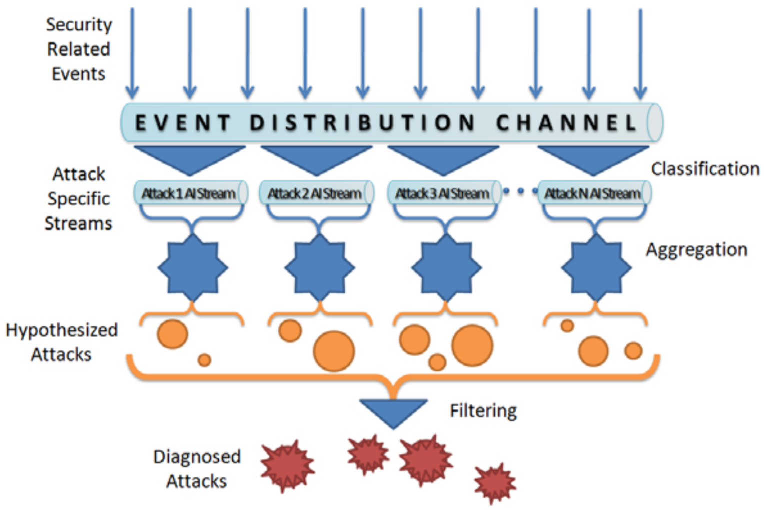

Fig. 3. The query generation process

of matching patterns, thanks to the use of the ontology, which enables to filter out the most relevant information (for the specific domain). The implemented aggregation patterns include temporal proximity and *Source* and/or *Target* matching of the *Symptoms*. Event aggregation is performed by making join queries which generate meta-events containing hypothesis on the possible results of the diagnostic activity. In order to discriminate among such hypothesis, a confidence degree is associated to every meta-event by using the *HasConfidenceLevel* property. The confidence degree is computed through a weighted combination of the *HasIntensityScore* values of the aggregated *Symptoms* where the weight of every *Symptom* is given by the *hasTrustworthinessValue* of the *AI*.

Filtering. The filtering function performs the crucial task of selecting the *Attacks* instances that have a *HasConfidenceLevel* value exceeding a configurable threshold. This threshold-based filtering enables to lower the number of false positives as only the aggregated events showing a robust detection pattern will be considered as an actual diagnosis of an attack and raise an alert.

The diagnostic process aims to extract a higher level of knowledge from the aggregated symptoms of an ongoing Attack. This process is not performed in offline mode on the aggregated *AIs*. Conversely, it is performed during all the phases of the correlation process. The goal of the diagnostic process is to characterize the ongoing attack in terms of:

– Attack Type - The Attack Type indicates the class of the ongoing attack. It is determined during the classification phase, by discriminating *AIs* based on the detectable types of *Attack*. The aggregation performed on a given class of *AIs* will result in the detection of that kind of *Attack*.

- Attack Targets - Information about the attack target(s) is inferred by looking at the *Target* concepts of the aggregated *AIs*. In most cases, not all aggregated *Symptoms* are *DirectedTo* the same *Target*. For example, when detecting an attack against an application server it is required that the behavior of both the application server and the database is monitored, and in case of an ongoing attack the Event Distribution Channel will be fed with different events. Some events are raised by probes monitoring the application server, while others are generated by probes observing the database. The generated events correspond to symptoms having a different target, and therefore the event correlation would result unfeasible. The information allowing the CEP to identify the relationships and dependencies between the different system components is inferred by the *Attack Indicators* concepts. This approach enables the correlation of the different symptoms and the identification of the system components affected by the attack. These components are considered as potentially violated targets, and as such they should not be considered trusted.
- Attack Latency - Basically, the Attack Latency (AL) parameter is the amount of time that an ongoing attack against to a system has been undetected. More precisely, it represents an upper bound estimate of the amount of time during which the system has been manipulated by the attacker. Given $t_1, t_2, .., t_N$ the timestamps of the aggregated *AIs*, and t_D the instant when the attack is detected, (*i.e.* the instant of the generation of the alert), AL can be evaluated as: $AL = t_D - \min(t_1, t_2, .., t_N)$ with N being the number of the available *AIs*.

4 Case Study and Experimental Results

In this section we present preliminary results attained by applying the proposed approach in a laboratory experimental testbed. The testbed includes a mySQL database and an Apache web server running an open-source Content Management System (CMS) written in PHP, namely Joomla (v.1.5). We carried out experimental tests in order to assess the capability of the proposed system to detect and diagnose two common attacks, namely SQL injection (SQLi) and Cross Site Scripting (XSS). Attacks were modeled and injected in the normal traffic profile. The following probes have been deployed throughout the networked system: (i)Apache Scalp[11], a host-level signature-based analyzer of Apache web server access logs, which uses a set of rules to spot malicious requests; (ii) ACDM (Anomalous Character Distribution Monitor)[6], a host-level anomaly-based probe which analyzes character distribution in HTTP requests and gives a score to every anomalous request; (iii) AQFM (Anomalous Query Failures Monitor), a database-level probe that monitors the rate of failed queries in a SQL database.

4.1 SQL Injection: Assumptions and Experimental Setup

Assuming that the system administrators apply security patches to the web application components in order to prevent attacks exploiting well-known vulnerabilities, any attacker will have to proceed by trial and error in order to find a way to exploit new and unknown vulnerabilities. For this reason the SQLi attack was modeled as composed of a set of unsuccessful attempts possibly followed by a successful exploitation of the vulnerability. The preliminary attempts performed by the attacker will leave a trace in the web server access log. These requests will look increasingly similar to the successful one as the attacker learns more about the internal mechanisms of the application and increasingly different from the requests in the "normal" traffic. Moreover these blind attempts will at first result in the injection of syntactically wrong queries leaving more traces of the ongoing attack as normal traffic usually does not generate errored queries.

Given this attack model, both the application server and the database are monitored in order to detect SQLi attacks. The application server is monitored by using two complementary probes: Scalp, which uses a signature based approach, and ACDM, that uses an anomaly-based approach. The database is monitored by using the AQFM probe. From an ontology point of view the SQLi *Attack isEvaluatedBy* three *AIs*: (i) SQLi attack detection performed by Scalp; (ii) Anomalous Character Distribution detected by ACDM; (iii) Anomalous Query Failures detected by the AQFM. The proposed Intrusion Detection and Diagnosis system collects such *AIs* and merges them in a separate event stream. Afterward, the *AIs* are aggregated in order to group the traces of the same ongoing attack. Since both Scalp and ACDM analyze the Apache access logs events raised by the two probes are aggregated on the basis of the access log entry they refer to. Subsequent attack attempts are correlated by using time proximity and source matching patterns. Once aggregation has been performed a set of meta-events is produced. Each meta-event contains a global, multi-level evidence of an attack and can be considered as a preliminary attack diagnosis. The diagnostic process provides further information which is deduced by looking at the aggregated *AIs*. For example, the starting and ending time of the attack are obtained by evaluating the lowest and the highest timestamp values.

4.2 Cross Site Scripting: Assumptions and Experimental Setup

Strategies for the detection of Cross Site Scripting (XSS) attacks and SQLi attacks are very similar since both approaches rely on the analysis of HTTP requests performed by the attacker in order to find attack traces. Cross Site Scripting is modeled as composed of a sequence of attempts which reflect the fact that initially the attacker has no information about how to inject the malicious code. The difference between SQLi attacks and XSS attacks is that no query failure is detected by monitoring the database in case the XSS code injection is not exploited to trigger the injection of SQL commands. This happens as the attacker usually injects client side code (like JavaScript) in web pages accessed by other users so as to steal sensitive information (*i.e.* cookies or session tokens) from their browsers without modifying the server side code of the application. This kind of *Attack isEvaluatedBy* Scalp XSS and ACDM probes. Thanks to the use of the ontology attack type, target and latency are correctly diagnosed.

4.3 SQLi and XSS: Experimental Results

In this section, we discuss the results of the experimental campaign, with respect to SQLi and XSS attacks.

Fig. 4a shows the performance of the detection process when a single *AI* is used. With respect to detection, experiments show that:

– Scalp has good performance when detecting SQLi attacks (72%), while the detection rate of XSS attacks is lower (63%);
– ACDM provides a very high detection rate (94%), at the cost of a rather high false positive rate (36% of the normal traffic is erroneously perceived as malicious);
– AQFM never fails to report database level traces of SQLi attacks (100%), but normal traffic and XSS attacks may generate failing queries, too.

As to diagnosis, it is worth noting that:

– Due to their intrinsic nature, the anomaly-based probes, namely ACDM and AQFM, have the drawback of being only capable of raising alarms upon detection of anomalous requests, but they are not able to provide any information about the specific threat which has originated the alarm neither they are able to provide further diagnostic info, e.g. detection latency, parts of the system under attack;
– Since SQLi and XSS attacks share similar patterns, they can be easily confused. Even Scalp, although being a signature-based probe, marks many SQLi requests as XSS (44% of the SQLi attacks, with 20% mapped to both kinds of attack).

Fig. 4b shows the performance of the detection process when complex correlation rules are applied to symptoms detected by probes which monitor different architectural levels, and use diverse detection approaches. Results show that the detection rate increases significantly, and the accuracy of the diagnosis is improved. In particular, the wrong marking performed by Scalp is eliminated. Furthermore, the false positives produced by the ACDM probes are drastically

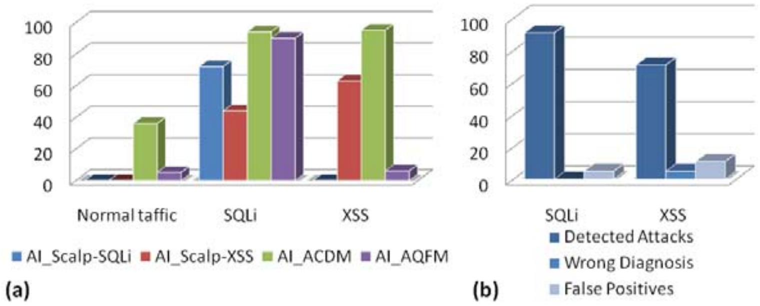

Fig. 4. (a) Detection performances using a single *AI*; (b) Detection performance correlating multiple *AIs*

reduced. Basically, even though Scalp does not detect all the malicious requests and, even worse, sometimes it gives wrong hints, the system almost always detects the SQLi attacks by adding to the Scalp detections those obtained via correlation with symptoms detected by ACDM and AQFM. In this way, the rate of correctly diagnosed SQLi attacks rises from the 73% achieved by Scalp to the 91% of our ID^{2}S. When application-level symptoms are aggregated with ACDM and AQFM symptoms, the successfull diagnosis of XSS attacks rises from 63% achieved by Scalp to 71% of the correlation-based approach.

The cases in which our ID^{2}S fails can be associated to two main scenarios: (i) wrong diagnoses are performed when query failures are generated by an XSS attack which leads to an incorrect SQLi diagnosis (4%); (ii) False Positives are raised by normal traffic that triggers a false detection by the ACDM (36%). When this detection is incorrectly aggregated with query failures generated by an overlapping SQLi attack, it produces a false SQLi diagnosis (5%), otherwise it can produce a false XSS diagnosis (11%). It should be emphasized that the false positives generated by ACDM are halved (16%) by means of the correlation with other symptoms.

5 Conclusions and Future Work

In this work, we have discussed the limitations of current Intrusion Detection Systems (IDS) technology, and proposed a novel approach, which we call Intrusion Detection & Diagnosis System (ID^{2}S) technology, to overcome such limitations. The basic idea is to collect information at several architectural levels (namely: Network, Operating System, Data Base, and Application), using multiple security probes which are deployed as a distributed architecture, and use Complex Event Processing (CEP) technology to perform sophisticated correlation analysis of intrusion symptoms. The escalation process from intrusion symptoms to the adjudged cause of the intrusion and to an assessment of the damage in individual system components is driven by ontologies.

We have conducted preliminary experiments on a testbed consisting of web servers running a well-known open-source Content Management System (CMS) written in PHP, namely Joomla. The experimental tests have demonstrated that our approach is effective against an emerging class of new attacks, which is referred to in the literature [1] as "stealthy". These attacks represent a major threat, since not only they have a dramatic impact in terms of economic losses [2], but (i) they also are invisible to current State-Of-The-Art IDSs, and (ii) current Intrusion Prevention Systems are ineffective against them.

Experiments conducted so far indicate that the proposed approach has three important advantages:

– It improves the performance of the detection process;
– It provides diagnostic features;
– It reduces the false positive rates.

Future work will follow two main directions. The first objective will be to conduct a thorough experimental analysis, in order to collect more evidence of the effectiveness of the approach. The second objective will be the implementation of more sophisticated correlation approaches, and the development of more detailed ontologies, so to allow finer grain estimation of the consequences of attacks on individual system components.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under Grant Agreement no. 216585 (INTERSECTION Project) and Grant Agreement no. 225553 (INSPIRE Project).

References

1. Jakobsson, M., XiaoFeng, W., Wetzel, S.: Stealth attacks in vehicular technologies. In: Proc. of The Vehicular Technology IEEE Conference, September 26-29, vol. 2, pp. 1218–1222 (2004)
2. IDC, Worldwide threat Management Security Appliances 2007-2011 Forecast and 2006 Vendor Shares: Still Stacking the Racks, Doc # 209303 (November 2007)
3. Repp, N., Berbner, R., Heckmann, O., Steinmetz, R.: A Cross-Layer Approach to Performance Monitoring of Web Services. In: Proc. of the Workshop on Emerging Web Services Technology, CEUR-WS (December 2006)
4. Yu-Sung, W., Bagchi, S., Garg, S., Singh, N.: SCIDIVE: a stateful and cross protocol intrusion detection architecture for voice-over-IP environments. In: Proc. of Dependable Systems and Networks Conference, June 28, pp. 433–442 (2004)
5. Vigna, G., Robertson, W., Vishal, K., Kemmerer, R.A.: A stateful intrusion detection system for World-Wide Web servers. In: Proc. of 19th Annual Computer Security Applications Conference, December 8-12, pp. 34–43 (2003)
6. Kruegel, C., Vigna, G.: Anomaly detection of web based attacks. In: Proc. of the 10th ACM conference on Computer and Communication Security (CCS 2003), pp. 251–261. ACM Press, New York (2003)
7. Majorczyk, F., Totel, E., Mé, L., Saïdane, A.: Anomaly Detection with Diagnosis in Diversified Systems using Information Flow Graphs. In: Proc. of The Ifip Tc 11 23rd International Information Security Conference, July 17, pp. 301–315 (2008)
8. Campanile, F., Cilardo, A., Coppolino, L., Romano, L.: Adaptable Parsing of Real-Time Data Streams. In: 15th EUROMICRO International Conference on Parallel, Distributed and Network-Based Processing, PDP 2007, February 7-9, pp. 412–418 (2007)
9. Fisher, Gruber, R.: PADS: a domain-specific language for processing ad hoc data. In: Proceedings of the ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation (2005)
10. The Borealis project, http://www.cs.brown.edu/research/borealis/public/
11. apache-scalp, Apache log analyzer for security, http://code.google.com/p/apache-scalp/