



Self-adaptive web intrusion detection system

Thomas Guyet, René Quiniou, Wei Wang, Marie-Odile Cordier

► To cite this version:

Thomas Guyet, René Quiniou, Wei Wang, Marie-Odile Cordier. Self-adaptive web intrusion detection system. [Research Report] RR-6989, INRIA. 2009, pp.24. inria-00406450

HAL Id: inria-00406450

<https://inria.hal.science/inria-00406450>

Submitted on 22 Jul 2009

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Self-adaptive web intrusion detection system

Thomas Guyet — René Quiniou — Wei Wang — Marie-Odile Cordier

N° 6989

Juin 2009

A large, light gray, stylized letter 'R' that serves as a background for the text 'Rapport de recherche'.

*Rapport
de recherche*

Self-adaptive web intrusion detection system

Thomas Guyet , René Quiniou , Wei Wang , Marie-Odile Cordier

Thème : Représentation et traitement des données et des connaissances
Équipes-Projets DREAM

Rapport de recherche n° 6989 — Juin 2009 — 24 pages

Abstract: The evolution of the web server contents and the emergence of new kinds of intrusions make necessary the adaptation of the intrusion detection systems (IDS). Nowadays, the adaptation of the IDS requires manual – tedious and unreactive – actions from system administrators. In this paper, we present a self-adaptive intrusion detection system which relies on a set of local model-based diagnosers. The redundancy of diagnoses is exploited, online, by a meta-diagnoser to check the consistency of computed partial diagnoses, and to trigger the adaptation of defective diagnoser models (or signatures) in case of inconsistency. This system is applied to the intrusion detection from a stream of HTTP requests. Our results show that our system 1) detects intrusion occurrences sensitively and precisely, 2) accurately self-adapts diagnoser model, thus improving its detection accuracy.

Key-words: intrusion detection, self-adaptive diagnosis, meta-diagnosis, self-adaptive system, web application intrusion

Système auto-adaptatif de détection d'intrusions Web

Résumé : L'évolution du contenu des serveurs et l'apparition de nouveaux types d'attaques rend nécessaire l'adaptation dynamique des systèmes de détection d'intrusion (IDS). De nos jours, les adaptations des IDS nécessitent des interventions manuelles – non-réactives et rébarbatives – de la part des administrateurs du système. Dans ce papier, nous présentons un système de détection d'intrusions adaptatif qui repose sur un ensemble de diagnostiqueurs locaux. Les redondances entre les diagnostics sont exploitées par un meta-diagnostiqueur qui surveille, en ligne, la consistance des diagnostics locaux, et, lorsqu'une inconsistance est détectée, il déclenche l'adaptation des modèles (des signatures d'intrusion) utilisés par les diagnostiqueurs incriminés. Ce système est appliqué à la détection d'intrusions à partir d'un flot de requêtes HTTP. Les résultats montrent que notre système 1) détecte les intrusions de manière précise et sensible tout au long de la surveillance, et 2) adapte de manière adéquate ses modèles de diagnostic et améliore ainsi ses performances de détection.

Mots-clés : détection d'intrusion, diagnostic auto-adaptation, meta-diagnostic, système auto-adaptatif, intrusion web

Contents

1	Introduction	4
2	Detecting intrusions from HTTP requests	5
2.1	Access logs	5
2.2	Intrusion using HTTP requests	6
2.3	Intrusion Detection Systems analyzing HTTP requests	8
2.4	Advanced IDSs	9
2.4.1	Combining the results of intrusion detectors	9
2.4.2	Adapting an IDS	10
3	System overview	10
4	An adaptive multi-diagnoser system for intrusion detection	11
4.1	Model-based diagnoser to detect intrusion	12
4.1.1	Diagnosis with Dempster-Shafer theory of evidence	12
4.1.2	Model-based diagnosers	13
4.2	Multi-diagnoser architecture	13
4.2.1	Virtual diagnoser agent	14
4.2.2	Multi-diagnoser architecture	14
4.3	Adaptive multi-diagnoser system	14
4.3.1	Integrity constraints	14
4.3.2	Meta-diagnosis	16
4.3.3	Adaptation	16
4.4	Example	17
5	Experiments	17
5.1	Data and experiments	18
5.2	Results	19
6	Conclusion	21

1 Introduction

Computerized systems, personal or professional, public or private, are more and more connected to the world wide web. Since such connections open accesses to data that the user wishes to protect, there is an increasing interest in access security issues. In order to avoid malicious accesses to, copies of or modifications of personal data, intrusion detection systems – IDS – are needed for robustly and precisely detecting intrusions into the protected system.

We are interested in IDSs that can detect intrusions on Web servers. Such servers are computer systems that are mostly encountered on the Web [7]. Since they are accessible by everyone, not surprisingly they are among the most attacked systems. Some of the common techniques used by attackers to compromise a website include exploiting a vulnerable Web application running on the server (by attacking through improperly secured input fields), or exploiting some vulnerability present in the underlying host operating systems. In 2008 alone, Symantec identifies 12.885 site specific web application vulnerabilities. According to this recent study, 63% of vulnerabilities affected Web applications in 2008, an increase from 59% in 2007. Therefore, the development of Web IDSs appears to be an important concern to respond to the need of global network security.

On the long term, the continuous accuracy and robustness of an IDS cannot be ensured in real situations without human assistance. In fact, real situations are not stable, and a major challenge is the conception of IDSs having the capacity to adapt themselves (*i.e.* with as little human assistance as possible) to their evolving environment. The environment can evolve in three ways:

- evolution of the Web Server content: new pages and services are added and removed dynamically. Thus, requests that failed at one time might be successful later.
- evolution of the Web Server usage: clients modify their behavior when using the web server along the time.
- evolution of the intrusions: new kinds of intrusion frequently emerge.

We are particularly motivated by designing a **self-adaptive** system to detect intrusions in Web servers. Considering the evolving environment, the adaptation of the IDS is required. Current adaptation solutions consist in updating or rebuilding some signatures (intrusion signatures or normal behavior signatures). But those solutions are tedious and unreactive because they are mostly manual. Updating a signature base requires an expert who has to perform a boring work to select the signatures that are suitable for his own server configuration and content. Similarly, rebuilding requires a tedious selection of significant trace from which the signatures are rebuilt. In both cases, a human interaction is required and the IDS will not be fully functional until it has been updated or rebuilt. In contrast, we would like to devise a **self-adaptive** Web IDS that could trigger its own adaptation without human assistance.

This situation is known as an on-line adaptation task in presence of “concept drift” [14] and is reputed to be difficult. The design of self-adaptive IDS raises two main issues: (1) the autonomous detection and the diagnosis of the adaptation requirement, and (2) the effective adaptation of the system.

We design a multi-diagnoser architecture, associated to a meta-diagnoser, that uses integrity constraints to decide when adaptation is required and which diagnoser should adapt. In a multi-diagnoser approach, each diagnoser agent constructs its own diagnosis from subsets of features extracted from the observations. As they are ground on different but partially redundant views of the system, these diagnoses are supposed to satisfy integrity constraints. When is not the case, *i.e.* some of the diagnoses are contradictory, the meta-diagnoser, according to a consensus principle, detects that the observed system has evolved.

In a first section, we present the issue of the diagnosis of an evolving system. The Section 2 is dedicated to the state of the art of the intrusion detection from HTTP requests. Then, in Section 3, we give an self-adaptive diagnosis framework of the system and in Section 4, we present our self-adaptive multi-diagnoser intrusion detection system. Finally, in Section 5, we give some experiments and results.

2 Detecting intrusions from HTTP requests

We are interested in an IDS that can detect intrusions on Web servers. More precisely, we would like to detect intrusions from HTTP requests that are submitted to the Web server. Deployed network IDSs (*e.g.* Snort) have the capability to detect intrusions by analyzing the traffic of TCP/IP packets. As a consequence, intruders attack the upper OSI layers, *e.g.* the application OSI layer. Thus, IDSs dedicated to specific applications seem more relevant to tackle the complexity of these attacks from the TCP/IP packet point of view. For instance, it is easier to detect a breach in a `cgi` script from the request than from the TCP/IP traffic. One of the main challenge is to protect the web server from known intrusions as well as unknown intrusions while avoiding any tedious update by the administrators.

In this section, firstly we introduce the structure of access logs, then we present some known intrusions using HTTP requests and finally we briefly present the general idea of HTTP request intrusion detection.

2.1 Access logs

All the requests received by the server are recorded in the web server access log thus a Web IDS can use the access log to detect intrusions. A log is composed of a list of lines. Each line corresponds to a request submitted to the server and is a rich structured source of information. It has several fields that describe the request and the response made by the server to this request. Figure 1 illustrates the overall structure of a **log line** provided by an Apache server (Combined format). The main fields are:

- **IP**: the IP address of the client (remote host) who has sent the request to the server. The IP address reported here is not necessarily the address of the machine at which the user is sitting. If there exists a proxy server between the user and the server, this address will be the address of the proxy, rather than the address of the source machine.
- **Time**: the time at which the server finished processing the request.

- **Request:** the request line from the client is given in double quotes. The request line contains many useful pieces of information. First, it contains the method used by the client (*e.g.* GET, POST, ...). Second, it contains the requested resources (including the potential scripts parameters), and third, it contains the protocol used by the client (*e.g.* HTTP/1.0).
- **Status code:** the status code that the server sent back to the client. It indicates the kind of response the server made to the request. For example, codes beginning with 2 indicate a successful response, codes beginning with 4 indicate an error caused by the client, ... The full list of possible status codes can be found in the HTTP specification (RFC2616 section 10).
- **Size:** this field gives the size of the object returned to the client.
- **Referrer:** this field gives the site that the client reports having been referred from ("-" if not available).
- **User agent:** The User-Agent is the identifying information that the client browser reports about itself. Especially, this field can be used to identify the robots.

```

123.13.17.2 - - [1/Apr/2008:23:58:34 -0800] "GET /documents/index.html HTTP/1.1" 404 3402 " " "IE 6"
69.12.60.15 - - [1/Apr/2008:23:58:48 -0800] "GET /scripts/access.html HTTP/1.1" 404 435 "http://serveur2/index.html" "Mozilla(5.0)"
256.69.1.23 - - [1/Apr/2008:23:59:37 -0800] "GET /index.html HTTP/1.1" 404 223 " " "Safari"
69.12.60.15 - - [1/Apr/2008:23:59:59 -0800] "GET /scripts/access.pl?user=johndoe HTTP/1.1" 200 22 "http://serveur/scripts/access.html" "Mozilla(5.0)"

```

IP
Time
Request
Status code
Size
Referrer
User agent

Figure 1: Apache log line examples. The session of IP 69.12.60.15 contains 2 requests.

A client **session** can be rebuilt from log lines by collecting the set of requests submitted by a same client (identified by his IP) to the server in a limited time window. The reconstructed session does not necessarily represent the complete client activity [16] as cache mechanisms may hide some requests. But in order to detect intrusions, the most interesting requests are those that are submitted to the server. Nonetheless, a proxy may hide an intruder behind a shared IP. We will use sessions as an alternative point of view on the current request.

2.2 Intrusion using HTTP requests

A Web server can be intruded by an attacker who sends a suitable HTTP request or a suitable succession of HTTP requests. Firstly, we give some simple HTTP request attacks, then we present some attacks using several requests.

Some simple attacks make use of scripts for unauthorized access to protected information such as:

- passwords (*e.g.* `.htaccess` file),
- database (*e.g.* data insertion),
- information about the local area network (*e.g.* request to system commands or access to log files),

```

192.168.0.0 - - [13/Jan/2006:01:07:21 -0200] "GET /awstats/
awstats.pl?configdir=|echo;echo%20YYY;cd%20%2ftmp%3bwget
...;echo%20YYY;echo|HTTP/1.0" 404 291
192.168.0.0 - - [14/Jan/2006:01:01:25 -0200] "GET /cgi-bin/
awstats.pl?configdir=|echo;echo%20YYY;cd%20%2ftmp%3bwget
...;echo%20YYY;echo|HTTP/1.0" 200 291
192.168.0.1 - - [12/Apr/2006:08:05:46 -0300] "GET /rpc/..%35%63
..%35%63..%35%63..%35%63/winnt/system32/cmd.exe?/c+dir+
c:\\+/OG HTTP/1.0" 400 294
192.168.0.1 - - [12/Apr/2006:08:05:47 -0300] "GET /cgi-bin/%2E
%2E%2F%2E%2E%2F%2E%2E%4E%4E%54%2F%73%79%73%74%65%6D%33%32
%2Fping.exe%20127.0.0.1
192.168.0.1 - - [12/Apr/2006:08:05:43 -0300] "GET /cgi-bin/
mrtg.cgi?cfg=../../../../../winnt/win.ini HTTP/1.0"
404 289
192.168.0.2 - - [12/Apr/2006:08:05:43 -0300] "GET /cgi-bin/
mrtg.cgi?cfg=../../../../../etc/passwd HTTP/1.0"
404 289

```

Figure 2: Intrusion examples. Intrusions from IP 192.168.0.0 are based on the awstat script, intrusions from IP 192.168.0.1 try to illegally execute system commands and intrusions from IP 192.168.0.2 try to access to passwords.

- information about the company activities (*e.g.* unauthorized browse of directories).

Another common intrusion is the SQL injection where the intruder attempts to corrupt a SQL database. In such case, the intruder attempts to execute some SQL commands like **select**, **where** or **from** using the HTTP request. The same applies to attempts to execute system commands like **cat**, **grep**, **wget**, **dir**, **ls**, etc. Specific characters like spaces, new lines or null terminators are widely used (and necessary) on most attempts to execute commands. To detect intrusion, it is interesting to look for these specific characters, but intruders often hide those characters using encoded URLs. In an encoded URL, the specific characters are encoded using hexadecimal codes, *e.g.* '%20' encodes a space (*cf.* Figure 2).

Figure 2 illustrates some **awstats** attacks. We see some common system commands, separators and some encoded characters in the URL. Looking at the HTTP result code, we know that one was successful and the other was not (error 404 and 200).

Some complex intrusions require several HTTP requests. In their security threat report [9], Symantec noticed increasing of complexity and sophistication of attacks. "while a single high-security flaw can be exploited to fully compromise a user, attackers are now frequently stringing together multiple exploits for medium vulnerabilities to achieve the same goal". For example, a category of attacks consists in using a software security hole to install malicious software that will generate unauthorized traffic on the server. Another example – and the best known attack of web servers – is the DoS attack which consists in overflowing a server with malicious requests or with requests that generate an internal error. A DoS attack provokes a system failure where the server is no

more able to respond to non-intrusive requests. In such cases, the analysis of only one request may not be sufficient to detect an attack and the analysis of a session may be required.

2.3 Intrusion Detection Systems analyzing HTTP requests

The techniques for intrusion detection fall into two major categories: signature-based detection and anomaly-based detection. Signature-based detection (*e.g.* Snort [20], ModSecurity [19]) identifies malicious behavior by matching a behavior profile against pre-defined descriptions of attacks. Anomaly detection [8], on the other hand, defines a profile of a subject's normal behavior and attempts to identify any unacceptable deviation as the result of a potential attack. The first category of techniques has good precision and sensibility for known intrusions but has difficulties to deal with new kinds of intrusions. In fact, a new kind of intrusion will not be detected, since its own signature is not in the intrusion signature base. On the other hand, anomaly-based techniques can easily detect unknown attacks, but their usage generates a lot of false-positive alarms.

Signature-based intrusion detection techniques are widely used for Web server intrusion detection. The request received by the Web server are successively compared with the signatures of malicious requests. Consequently, the intrusion detection problem stands in the proposition of models (signatures) that may have the capability to represent robustly and precisely the various intrusions that may be encountered. The role of a model is, on the one hand, to focus on the request features that are relevant for intrusion detection issues and, on the other hand, to abstract the feature values into the representation.

A wide range of log line models have been already proposed. Tombini *et al.* [25]'s anomaly model is a list of pairs linking the accessed resources and the combination of parameters that were used, if any. Since web sites are organized as trees, the global model can be represented by a simple tree structure. If the requested anomaly detector belongs to the anomaly model, the anomaly detector checks whether the combination of parameters used is allowed or not. Kruegel and Vigna [13] introduced an anomaly-based detector of Web-based attacks. They proposed several intrusion detection models based on request features: attribute length distribution, attribute character distribution, Markov model of the structure of the query attributes, attributes order specifications, . . . Ingham *et al.* [11] use a deterministic automaton to model the sequence of tokens¹. Bolzoni *et al.* [4] recently proposed to use regular and irregular expressions models. Cheng *et al.* [5] propose to prevent attacks by monitoring the user behavior with templates modeled by Markov models. In [22, 27], clustering techniques are used to construct dynamically a model of normal behavior as a set of clusters based on the character distribution feature.

To the best of our knowledge, no session model has been proposed so far. Nonetheless, it must be related to the recent interests in alert correlation [6]. Several techniques, especially Bayesian Networks or causal networks, are used to combine alerts and to recognize intrusion plan [18] or scenarios [17].

¹Tokens are semantic units of the URL separated by specific characters ('', '?', '&')

2.4 Advanced IDSs

In this paragraph, we review some specificities of advanced IDSs : 1) the combination of diagnoses and 2) the adaptation.

2.4.1 Combining the results of intrusion detectors

It is generally admitted that the combination of several intrusion detectors (*i.e.* forming an ensemble of IDSs) can achieve a better performance [2]. The intrusions can be better detected by combining several pieces of information that are known to be complementary. Complementary aspects can be observed along four axes:

- information that comes from distinct IDSs distributed on the local network
- information that is gathered from different kind of logs: different OSI layers (*e.g.* TCP/IP packets, HTTP logs, ...) and/or different applications, sources (system commands, database, web server, ...).
- information that is extracted from the same source but through different information filters (*e.g.* logs attributes),
- information that comes from systems with different “security policies” (*e.g.* Signature-based vs Anomaly-based, or Anomaly detection vs Misuse detection [25]).

The first two axes use several information sources, while the last two axes combine the information obtained from only one source but from different points of view and with the aim to extract as relevant information as possible from this source.

The first two axes enable the proposition of an architecture to detect and prevent attacks in local area networks. The main idea is to centralize the information that comes from several existing tools in order to make the detection more robust. The system of Tsian *et al.* [24] merges alarms that comes from several network-based IDSs and host-based IDSs deployed on the local network. It uses the Dempster-Shafer [21] for data fusion. Gu *et al.* [10] propose a decision-theoretic alert fusion based on a likelihood ratio test (LRT). In a global area network, a collaborative approach [15, 26] to intrusion detection aims at giving a global view of the network attack activity. Augmenting the information obtained at a single site with information gathered from the network can provide a more precise model of an intruder’s behavior. For instance, the *Worminator* [15] is a P2P collaborative approach to the intrusion detection.

The last two axes are widely used in case of rich and structured data such as access logs. Since early work on web attack detection [13], it has been noticed that some access log line attributes are more efficient to detect some attacks than others. Similarly, some attributes generate more false alarms on some normal data. A method based on a single attribute would be unable to detect robustly and accurately all the attacks that can be encountered on web servers, and it may be fairly easily circumvented by new attacks created by malicious clients who can hide their intrusions by avoiding the traces they know to be detectable through some monitored features. To cover a wide range of attacks and to detect most intrusions, the Web IDS must analyze several attributes and combine the results of the analysis. For instance, the system of Kruegel and

Vigna [13] computes an anomaly score using by a weighted sum of anomalous probabilities.

2.4.2 Adapting an IDS

Interaction modes between a client and a web server are highly dynamic. So, the features of normal and abnormal behaviors may change rapidly necessitating the adaptation of the monitoring system. Adapting an IDS aims at 1) progressively improving the detection reliability, and 2) at acquiring the capability to detect new kind of intrusions. We focus our attention on the discovery of new kinds of intrusion.

Practically, a Web IDS (*e.g.* ModSecurity [19]) requires a lot of human actions, mostly tedious, which restrict the IDS reactivity. To detect new kinds of intrusion, administrators must update manually the list of intrusion patterns from signatures elaborated by experts. It appears to be strongly desirable to automatize (a part or the totality of) the discovery of new kinds of intrusion, the construction of signatures and the effective update of the IDS.

HoneyComb [12] is a NIDS that facilitates the discovery of new kinds of intrusion by using HoneyPot. A HoneyPot is a decoy computer resource. Since there are no entry points for users to interact with these systems, activities on HoneyPots is considered suspicious by definition. Activities of entities attacking HoneyPots are logged to identify suspicious behaviors and to automatically extract intrusion signatures.

Another kind of approach aims at adapting the intrusion signatures on-line. Bojanic *et al.* [3] propose to use HMM for intrusion detection in system command sequences. In this method, normal and abnormal (intrusions) behaviors are modeled by HMMs. If a sequence is suspected as being non probable with respect to known sequences, additional analyzes are performed. If these new analyzes tend to show that the sequence does not correspond to an intrusion then the HMMs linked to normal behavior are updated, else HMMs associated to intrusions are modified or a new HMM is created. In [23], Srinoy proposes to use SVM intrusion models associated to a swarm intelligence technique enabling a dynamic adaptation of intrusion models. Wang *et al.* [27] are confronted to the same concept drift issue as us and propose an adaptive Web intrusion detection system based on outlier detection with the affinity propagation clustering algorithm and an outlier reservoir that gathers potential intrusion waiting for further analysis.

3 System overview

A Web server receives a stream of HTTP requests. For each new arriving request, the adaptive multi-diagnoser system constructs a diagnosis labelling the request as intrusive or not. If the request is not intrusive, then it will be processed normally by the server. In parallel a meta-diagnoser observes the diagnosis process and can trigger the adaptation of the diagnoser based on the current diagnosed request. Figure 3 illustrates the system architecture.

Our diagnosis approach relies on a multi-diagnoser architecture, *i.e.* several diagnoser agents contribute to the global diagnosis. All the diagnosers diagnose the same (sub-)problem by different methods and from different features

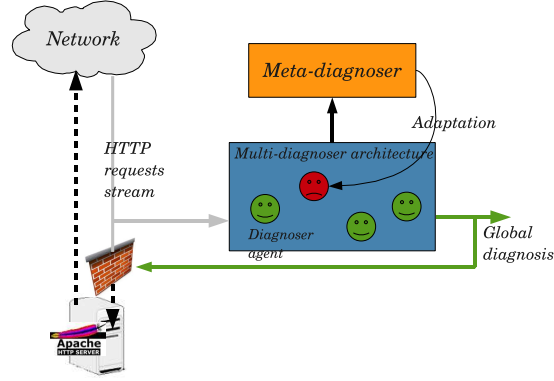


Figure 3: System overview. The multi-diagnoser architecture consists of 4 diagnosers. The decision of the red, “unhappy”, diagnoser is inconsistent with the decisions of other diagnosers. Consequently, the meta-diagnoser may propose to adapt this diagnoser. The global diagnosis is used to block the request or not.

extracted from observations taken at different time points or at different locations. As a consequence, the global diagnosis is elaborated from partial and redundant diagnosis results.

Considering that no absolute reference is available, a diagnosis mistake is detected by identifying the inconsistencies between diagnosers. The redundancy of a set of diagnoses is used by the meta-diagnoser to monitor the need for adaptation. Our idea is to use the several partially overlapping views of the system. Due to the fact that views are partially redundant, they must be *consistent* on the common parts. If not, it can be concluded that some change is occurring in some part of the system. For example in the medical domain, at a particular time the fever severity (**high**, **average** or **low**) of a patient should be the same though it is computed by different means. If the diagnoses are not *consistent* (here, the consistency means fever severity equivalence), it is a good indication that at least one of the models should be adapted so that the diagnosers will have a better behavior in the future.

The meta-diagnoser is in charge of analyzing the output of the diagnoser agents, of detecting some *inconsistencies*, of locating the diagnosers rising problems and of proposing actions to improve them by self-adaptation.

4 An adaptive multi-diagnoser system for intrusion detection

As mention in Section 2.3, models are important to detect intrusions robustly and precisely. Moreover, our aim is to provide automatically the models to administrators and to avoid handmade models. To this end, we focus our attention on models that are learned from datasets of labeled log lines.

Each model focuses the diagnoser attention on a specific feature of the log line. Several features (*e.g.* request length, character or token distribution, status code, etc.) are extracted from observations contained in the log lines at two

abstraction levels, line and session. These features are commonly used for intrusion detection [13]. For instance, character or token distribution may be useful to distinguish malicious requests from normal ones. Some malicious requests use the URL (especially parameters to scripts) to send intrusion instructions to the target server. A perceptible change of character distribution could denote the use of such suspect instructions.

For example, to diagnose the last request of Figure 1, the features would be the following:

- character distribution of the URL in the request : 'a': 1, 'b': 0, 'c': 3, 'd': 1, 'e': 3, ...
- token distribution of the URL in the request : 'scripts': 1, 'access.pl': 1, 'user': 1, 'johndoe': 1
- ratio of the errorful status code in the session : `error ratio (200)`: 0.5
- character distribution of all the URLs in the session : 'a': 2, 'b': 0, 'c': 6, 'd': 1, 'e': 4, ...

A typical character distribution (resp. token distribution) is constructed from examples as the mean of each character (resp. token distribution) occurrences. Note that the feature space dimension is 256 for character distribution, but is infinite for token distribution. Models based on the ratio of errorful status code in sessions make use of a Gaussian distribution model and are represented by the parameters (σ, μ) of the Gaussian.

In this section, we present a proposal for an adaptive web intrusion detection system. In Section 4.1, we introduce the intrusion detection using model-based diagnoser agents. Then, in Section 4.2, we explain how the agents diagnoses are combined. Finally, in Section 4.3, we detail the adaptation layer of the system.

4.1 Model-based diagnoser to detect intrusion

A model-based diagnoser constructs a diagnosis about the current request according to its own model. Our definition of diagnosis is inspired by the Dempster-Shafer (DS) theory of evidence [21]. This choice is justified by the fact that the diagnosis has to take into account the uncertainty coming from the partial views that the different agents have on the observed system. To this end, the quantitative representation of a diagnosis and the explicit management of uncertainty in the DS theory are relevant. Moreover, as a fusion theory, it provides a strong solution to combine diagnoses as required by our multi-diagnoser architecture.

4.1.1 Diagnosis with Dempster-Shafer theory of evidence

A diagnosis expresses the more or less certainty in any of each status that can be associated to the current request: normal (N), intrusive (I), or even unknown (U) *e.g.* when the uncertainty is too high. This notion of diagnosis is formalized in the Dempster-Shafer (DS) theory of evidence.

Definition 1 A *diagnosis* d is a normalized distribution of “masses” on $\Omega = \{N, I, U\}$:

$$d : \Omega \mapsto [0, 1], \quad \sum_{A \in \Omega} d(A) = 1$$

For all $A \in \Omega$, $d(A)$ expresses the belief which supports the claim that the current request is of status A . The preferred candidate in $\Omega - \{U\}$ is called the diagnosis decision and it is unique.

Definition 2 The **diagnosis decision**, dd , is the element of $\{N, I\}$ associated to the diagnosis d having the maximal belief:

$$dd = \arg \max_{A \in \{N, I\}} (d(A))$$

$\max_{A \in \{N, I\}} (d(A))$ gives the belief in the diagnosis decision.

Note that the diagnosis decision cannot be U , but the mass $d(U)$ modifies the belief in the diagnosis decision: the greater $d(U)$, the more uncertain the diagnosis.

4.1.2 Model-based diagnosers

A diagnoser is characterized by its model. The model describes the diagnostic knowledge used by the diagnoser to compute its diagnosis from a subset of the observations. Each diagnoser has its proper and partial point of view on the system.

Definition 3 A **concrete diagnoser agent (CDA)** is characterized by its mode \mathcal{M} where \mathcal{M} contains two parts: the submodel of normal requests \mathcal{M}_N and the submodel of intrusive requests \mathcal{M}_I .

In a bootstrap phase, the submodels are learnt from sets of labelled examples (normal and intrusive requests). The precision p of the model is computed as the ratio of correct diagnoses on the learning sets.

While diagnosing the log stream, each CDA computes its diagnosis d from the distance between the current request (R) and the submodels: $d(N) = \|\mathcal{R} - \mathcal{M}_N\|$, $d(I) = \|\mathcal{R} - \mathcal{M}_I\|$. The uncertainty mass, $d(U)$, is given by $1 - p$, where p is the model precision. Finally, the diagnosis is normalized. The distance $\|\cdot - \cdot\|$ depends on the request feature that is used. For character and token distributions, the model distance to a request is the euclidean distance, and for the ratio of errorful status code, it is the ratio probability given by the Gaussian distribution (*i.e.* $\mathcal{N}_{\sigma, \mu}(ratio)$).

4.2 Multi-diagnoser architecture

A multi-diagnoser architecture can be seen as a multi-agent system in which the agents are diagnosers. In order to combine the diagnoses, we introduce another kind of diagnoser agents, the virtual diagnoser agent (VDA). They aim at merging the diagnosis of several other agents. As a consequence, the agents are organized in a hierarchical structure specifying the fusion scheme from the CDA diagnoses to the global diagnosis of the current request. The agents and their hierarchical structure compose the multi-diagnoser architecture.

4.2.1 Virtual diagnoser agent

Definition 4 A *virtual diagnoser agent* (VDA) D^v is represented by a pair $\langle \mathcal{D}, \oplus \rangle$ where \mathcal{D} is the set of diagnoser agents that provide the input diagnoses to D^v and \oplus is a combination operator used by D^v to compute its diagnosis.

A VDA constructs a diagnosis by combining diagnoses that have been constructed by its related diagnoser agents (concrete or virtual) as defined in \mathcal{D} . It is virtual in the sense that it is not directly related to concrete observation sources. The combination operator \oplus defines how to construct the VDA diagnosis from the diagnoses of the diagnosers of \mathcal{D} . In our context, the Dempster-Shafer combination rule is used. For all subset $A \in \Omega$, the combination of diagnoses d_1 and d_2 is computed by:

$$d(A) = (d_1 \oplus d_2)(A) = \frac{\sum_{B \cap C = A} d_1(B)d_2(C)}{1 - \sum_{B \cap C = \emptyset} d_1(B)d_2(C)}.$$

The Dempster-Shafer combination rule is associative, thus the definition can be easily extended to the combination of more than two diagnoses. Variants of the Dempster-Shafer combination rule exists [28] and could be used as well.

4.2.2 Multi-diagnoser architecture

Definition 5 A *diagnosis combination graph* (DCG) is a directed acyclic graph where nodes represent diagnoser agents and edges specify the communication flow of diagnoses between agents. Nodes with no descendants, called **leaves**, are CDAs and other nodes are VDAs. Among VDAs with no ancestors one is designated as the **root** node and represents the global diagnoser.

Definition 6 A *multi-diagnoser architecture* is represented by a tuple $\langle \mathcal{C}, \mathcal{V}, G, R \rangle$, where \mathcal{C} is a set of CDAs, \mathcal{V} is a set of VDAs, G is a DCG, and R is the virtual diagnoser related to the root of G . The diagnosis computed by R provides the global diagnosis of the system.

The diagnosis is performed recursively through the DCG: the root VDA triggers its children for monitoring. If a triggered child is a VDA, it triggers in turn new diagnosis agents; if a triggered child is a CDA, it computes a new diagnosis based on its model and the current observations. Once its diagnosis is computed, a CDA communicates its diagnosis to its ancestor (a VDA) which will combine all diagnoses sent by its children. Finally, the root VDA combines the diagnoses collected from its children and compute the global diagnosis.

The combination graph we use to detect Web intrusion (*cf.* Figure 4) makes explicit diagnoses based on the session view and diagnoses based on the log line view: the diagnoses of the Request-CDAs (*resp.* the Session-CDAs) are combined by the Request-VDA (*resp.* the Session-VDA) and the Root-VDA combines the diagnoses of the Request-VDA and the Session-VDA.

4.3 Adaptive multi-diagnoser system

4.3.1 Integrity constraints

The meta-model used for meta-diagnosis is represented by a set of integrity constraints that must be satisfied by the diagnoses computed by CDAs and

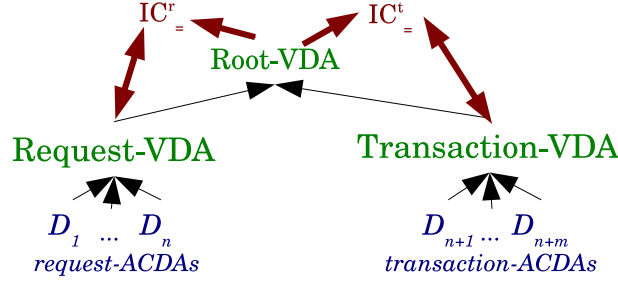


Figure 4: Diagnosis combination graph (DCG) for HTTP intrusion detection. Integrity constraints are illustrated by bold red arrows.

VDAs. Integrity constraints express temporal, spatial or structural properties of the observed system.

Definition 7 An *integrity constraint* IC on CDAs or VDAs $\mathcal{D} = \{D_1, \dots, D_n\}$ is a tuple $\langle \mathcal{D}, c, MDCS_{IC} \rangle$ where

- c is a set of constraints between the diagnoses of the diagnosers in \mathcal{D} ,
- $MDCS_{IC} \subset \mathcal{D}$ is the **meta-detection conflicting set**, i.e. the set of the possible sources of integrity violation.
- $\mathcal{D} \setminus MDCS_{IC}$ is the set of **reference diagnosers**.

The goal is to distinguish intrusive *vs* non intrusive sessions, first, and then to distinguish intrusive *vs* non intrusive requests. The difficulty is then to be able to separate intrusive and non intrusive requests inside an intrusive session. To simplify the problem we assume that every requests in an intrusive session is intrusive. We are conscious that this assumption is too coarse but it could be relaxed later by supposing that a session is intrusive if it contains a high ratio of intrusive requests. Note, however, that this assumption is not directly used for computing the diagnosis but for determining whether a diagnoser should be adapted or not. So, to design the meta-model for intrusion detection, we assume the following property for sessions: every request of a session is of the same *type* as the session it belongs to, *e.g.* if a session is intrusive (I), then all the requests should be intrusive. This assumed property is exploited by the meta-diagnoser: when a session predicts an intrusion whereas a request from the same session predicts a normal behavior an inconsistency should be reported. The converse situation may also occur.

In the context of Web IDS, the assumed property is exploited to define two integrity constraints: $IC^r_{=}$ and $IC^t_{=}$. $IC^r_{=}$ (resp. $IC^t_{=}$) is satisfied if the diagnosis decision of the Request-VDA (resp. Session-VDA) is the same as the diagnosis decision of the Root-VDA. If $IC^r_{=}$ (resp. $IC^t_{=}$) is not satisfied, the MDCS includes all the Request-CDAs (resp. Session-CDAs). Figure 4 represents these ICs. The red bold arrows that link an integrity constraint node to diagnosers node represent the diagnosers involved in the constraint. The red bold arrows with double arrows identifies the diagnosers of the MDCS. Diagnosers that are not in the MDCS are reference diagnosers: their diagnoses will not be contested.

4.3.2 Meta-diagnosis

A meta-detection conflicting set indicates that at least one of its elements is inconsistent with the others. The next step is to localize which diagnoser is responsible of this inconsistency and should consequently be adapted.

Definition 8 The *meta-diagnosis set* ($MDCS_{IC}$) for an integrity constraint IC is the set of agents to adapt if IC is not satisfied.

In our case, we assume firstly that the fusion performed by VDAs cannot be responsible of inconsistencies. Thus, the defective diagnosers must be found among the CDAs. Secondly, we consider that all the leaves that are descendants of the MDCS nodes involved in a violated integrity constraint can be suspected. As a consequence, the meta-diagnosis set for an integrity constraint IC is the set of all the CDAs which are in $MDCS_{IC} = \{D_1^c, \dots, D_n^c\} \cup \{D_1^v, \dots, D_m^v\}$ or which are descendants of at least one $D_k^v \in MDCS_{IC}$.

In case of inconsistency, the current global diagnosis is computed once and labelled as uncertain. This label advises the user to not trust the current diagnosis until a new trustable one, computed with adapted CDAs, will be provided in a near future.

4.3.3 Adaptation

The final step to get a fully self-adaptive multi-diagnoser system is to have means to adapt the defective CDAs from the meta-diagnosis. To this end concrete diagnoser agents are enriched with adaptation functions.

Definition 9 An *adaptive concrete diagnoser agent* (**ACDA**) is represented by a pair $\langle \mathcal{M}, f_A \rangle$ where \mathcal{M} is the model of a CDA and f_A is a model adaptation function.

Definition 10 A *reference diagnosis decision* dd_r is computed by combining the diagnoses provided by the reference diagnosers of an integrity constraint IC .

Once, the meta diagnosis has identify the ACDAs to adapt, a **reference diagnosis decision** dd_r is computed by combining the diagnoses provided by the reference diagnosers of an integrity constraint IC . For example, the integrity constraints noted IC^r , of the Figure 4, the reference diagnoser is simply the global diagnosis (constructed by the root-VDA). In this case, there is only one reference diagnoser, then it is not require to combine several diagnoses.

The reference diagnosis decision dd_r related to an unsatisfied IC is provided to the ACDAs in the $MDCS_{IC}$ for adapting their model. Continuing the previous example, diagnosis decision is provided to the Request-ACDAs (the request-VDA can not be adapted). Each agent uses its own adaptation functions with the current request and dd_r as parameters. Practically, if $dd_r = I$ (resp. $dd_r = N$), then the revised submodel \mathcal{M}_I (resp. \mathcal{M}_N) is computed by a weighted averaging of the observed request feature (character distribution, token distribution, ...) and the old model \mathcal{M}_I (resp. \mathcal{M}_N).

Definition 11 An *adaptive multi-diagnoser system* is a pair $\langle \mathcal{D}, \mathcal{M}_{\mathcal{D}} \rangle$, where \mathcal{D} is a multi-diagnoser architecture whose ACDAs are adaptive and $\mathcal{M}_{\mathcal{D}}$ is a meta-model of \mathcal{D} .

In our case, the meta-model $\mathcal{M}_{\mathcal{D}}$ is a set of integrity constraints.

4.4 Example

Figure 5 illustrates the propagation of diagnoses along the DCG. The process begins by the computing of the concrete diagnosers of the ACDAs, the results of which are given at the bottom of the figure. The diagnosis of the VDAs are computed next by combining the suitable diagnoses applying the Dempster-Shafer rule. In this example, the two ICs are satisfied because the diagnosis decision of the Root-VDA and the Request-VDA or the Session-VDA are the same : they conclude that the diagnosis decision is N and, so, the Root-VDA reports that the request is normal. Note that the diagnoser D_{CD}^r and D_{Token}^r disagree, but there is no IC to conclude on the dysfunction of one of them. The meta-model assumes that it is quite normal to have inconsistent diagnoses at this level and the inconsistency is solved by using the Dempster-Shafer combination rule for the fusion of the contradictory diagnoses.

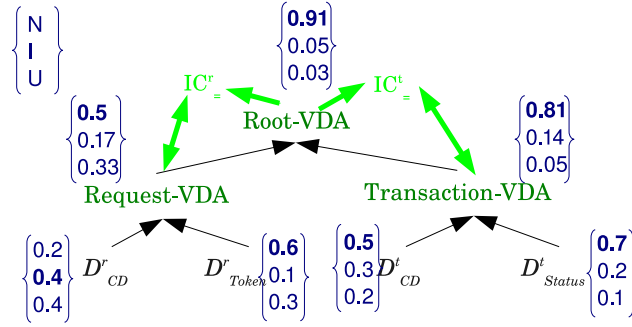


Figure 5: Diagnoses illustration (without adaptation). For each diagnosers, the 3d vector gives the diagnosis (masses distribution of N , I and U). The bold number is the highest belief and its position in the vector indicates the diagnosis decision.

Figure 6 illustrates the case of a diagnosis which leads to an adaptation. We just changed the diagnosis of the ACDA D_{Token}^r . In this case, the Root-VDA diagnosis decision is I but it is not equal to the Session-VDA diagnosis decision (N). Then, the integrity constraint $IC^t_ =$ is not satisfied. The consequence will be the adaptation of the relevant submodels of ACDAs D_{CD}^t and D_{Status}^t .

5 Experiments

The system, called LogAnalyzer², is fully implemented in C++. The main objective of the system evaluations is to show that the multi-diagnoser approach of model adaptation improves the system performances (*i.e.* precision and sensitivity), on the one hand, and enables the discovery and the effective use of new kinds of intrusions, on the other hand.

²see <http://www.irisa.fr/dream/LogAnalyzer/> for more information.

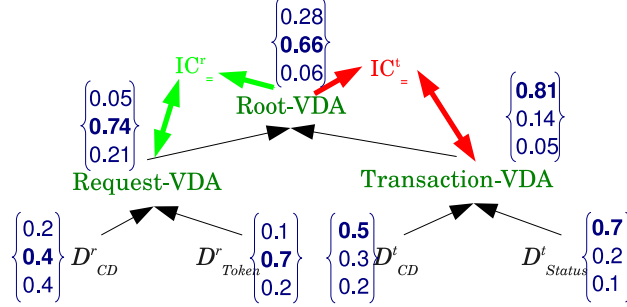


Figure 6: Diagnoses illustration (with adaptation). $IC^t_{=}$ is unsatisfied, D^t_{CD} and D^t_{Status} will be adapted.

5.1 Data and experiments

We collected two large data sets of HTTP access logs on the main Apache server of two research institutes in July 2007 and June 2008 during 1 month (about 10 million of requests). A preprocessing step consists in filtering out bots and known non intrusive requests (*e.g.* requests to static contents: *.html*, *.jpg*, *.pdf*, ...) led to a data reduction. Only 4.66% of the original requests remained in the logs after filtering. The dataset was also checked to verify that it contained no intrusion.

For each experiment, 1 million requests, corresponding to several days of recording on our server, were extracted from the real HTTP log free of intrusion. Then, some session, 400 on average, containing 20 intrusive requests (on average) were introduced at random positions. The intrusive requests were chosen randomly among 239 known intrusive request examples from the Nikto intrusion database [1]. Among them, a subset of 203 known intrusive request examples were manually selected to be used for learning the initial ACDA models. The other 36 intrusive requests were used for building instances of new kinds of intrusion that could be encountered during monitoring.

For each request, we compared the global diagnosis decision to the known status (intrusion or normal) of the request and it were classified among :

- the false positives (FP) : normal requests that have been diagnosed as intrusive,
- the false negatives (FN) : intrusive requests that have been diagnosed as normal,
- the true negatives (TN) : normal requests that have been diagnosed successfully as normal,
- or, the true positives (TP) : intrusive requests that have been diagnosed successfully as intrusive.

We segmented the log in 100 batches of 10.000 requests. For each batch, we counted the number of FP, TP, FN and TN occurring in the batch, and we computed the following monitoring performance indicators:

	Diagnosis time	Adaptation time
Logline character distribution	31"	2"
Logline tokens distribution	16'11"	6'10"
Session character distribution	42"	ϵ
Session error proportion	21"	ϵ
Global	23'48"	

Table 1: Cumulate time spend by agents (or the system) to diagnose or to adapt the 1 million requests. *epsilon* means less than 1 second.

	FP	FN	TP	TN	DR	FPR	F-Measure
With adaptation	2091	530	2018	997831	0.79	0.002	0.61
Without adaptation	21838	73	2461	978098	0.97	0.022	0.18

Table 2: Performance indicators computed with the diagnoses of the 1 million requests + 2534 intrusions

- Detection rate ($DR = TP/(TP + FN)$), *i.e.* the accurately recognized intrusions.
- False Positive rate ($FPR = FP/(FP + TP + FN + TN)$),
- F-measure ($F - Measure = 2 * DR * P / (DR + P)$ where $P = VP/(VP + FP)$).

In this way, it is possible to observe the evolution of performance indicators over time.

In the experiments, we studied the accuracy of adaptations and the improvement of the detection performances by adaptations. The experiments consisted in comparing diagnosis performances with and without adaptation. Without adaptation, the diagnoses were computed according to the principle of our multi-diagnoser architecture but the models of the diagnoser agents, learned from the training set, do not evolve.

Experiments have been performed using a personal computer (Intel Centrino Duo T7500). It takes less than 25' to process the 1 million request and it requires less than 30 Mo of memory. The table 1 illustrates the computing times we obtained with a personal computer (Intel Centrino Duo T7500).

5.2 Results

Figure 7 shows the evolution of the performance indicators with and without adaptation. The Table 2 shows the performance indicators computed with the entire log. With adaptation, 205565 adaptations occurred. The adaptation accuracy is more than 99%. This means that there are only few cases in which the adaptation is faulty (*e.g.* intrusive models are adapted with a normal request). The main part of the adaptation (202203 occurrences) consists in updating the normal model with a normal request.

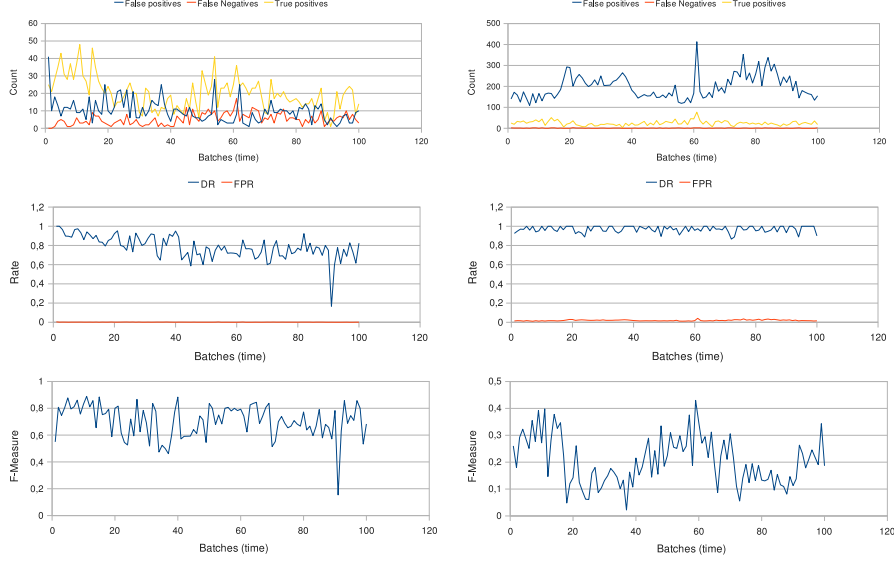


Figure 7: Evolution of the performance indicators over time. Figures in the left column: with adaptation, Figures in the left column: without adaptation. The first two figures (upper), give the evolution of the FP, FN and TP. Figures in the middle, give the evolution of the detection rate and the false positive rate. The last two Figures (lower), give the evolution of the F-Measure.

The first two figures show that the adaptation reduces drastically the number of false positive diagnoses. The number of FP falls down at the very beginning. This means that the adaptation is quickly efficient, *i.e.* the system is reactive. The number of FP does not increase thereafter. On the opposite, the number of FP stays around 200 per batch (without adaptation it is only about 50 FP per batch). Nonetheless, the true positive decreases a little while it is constant without adaptation. We can conclude that the adaptation makes our system more specific to intrusion: it efficiently reduces the false positive rate, but it reduces a little the true positives. The number of false negatives are low in the two cases.

The DR and FPR figures confirm that the detection performances with adaptation slowly decreases over time while it is constant without adaptation. The average of detection rate is 80%. Moreover, we see that the FPR is low, on average: 0.02 without adaptation and 0.002 with adaptation. Despite the only four diagnosers, these performances are quite good.

The F-Measure shows the global performances of a diagnoser, it takes into account both the sensitivity and the precision of the diagnoser to detect the intrusion. With adaptation, the F-Measure varies around $0.70(\pm 0.14)$, while without adaptation, the F-Measure varies around $0.2(\pm 0.09)$. Moreover, we can see that the global performances of the system with adaptation is relatively constant despite the decrease of the detection rate. In fact, the number of intrusion is very low, then the detection rate has only little influence on the global performances.

We explain the FPR difference with and without adaptation by the fact that our evaluation set of intrusions holds some intrusions that are not in the training set. With adaptation, the system discovers these initially unknown intrusions and the signatures have been enriched by this knowledge. On the opposite, without adaptation, the initially unknown intrusions stay unknown for the system and are not detected (false negatives).

The decrease of the true positive rate can be explained by an overlearning of normal models. In fact, a lot of adaptation of the normal model are performed and the models become overfitted. Consequently, some normal requests are less recognized over the time. In fact, diagnosis is a normalized distribution, if the normal mass is lower than before while the intrusion mass stays the same, the intrusion mass may become the biggest.

Based on these experiments, we can conclude that the adaptation improves reactively the global performances of the multi-diagnoser system and also maintains them at a high level. Moreover, our system discovers and detects dynamically new kinds of intrusion.

6 Conclusion

We have presented a system for self-adaptive intrusion detection from a stream of HTTP requests. Our proposition associates a multi-diagnoser system and a meta-diagnosis process. Each diagnoser agent constructs its diagnosis according to its own point of view of the system. Considering that the views are partially redundant, integrity constraints can be expressed on diagnoses. The meta-diagnosis process consists in using unsatisfied integrity constraints to trigger the adaptation of a subset of the diagnoser models.

The results of our experiments concluded that the multi-diagnoser architecture has good computing and performance results. The adaptation improves reactively the global performances of the multi-diagnoser system and also maintains them at a high level. Moreover, our system discovers and detects dynamically new kinds of intrusion. Nonetheless, we noticed that the number of true positives slowly decreases over time.

It is clear that more sophisticated methods can be used to locate the defective agents as for instance expert rules or any information on the source of the detected problem. Other heuristic or informed methods as well as model-based diagnosis methods (hitting-sets, prime implicants, etc.) could also be adapted to achieve this task.

We presented a general framework for adaptive intrusion detection system and its first application in order to prove the validity of our proposal. A first perspective will be to propose new diagnosers and new hypothesis to construct alternative diagnosis graph. For instance, it may be based on models learnt on the long term *vs* models learnt recently, it may compare the diagnosis from a two different web servers (with their own signatures) may be compared, etc. We only used four diagnosers and the computing performances show that several diagnosers may be added without analyzing time constraints.

A second perspective is to design a more complete IDS solution. We presented a completely autonomous system: not any manual action is required.

Nonetheless, in real situations, it is strongly recommended to use the administrator expertise. For example, administrator may be helpful to correct (not necessarily frequently) the overlearning of some diagnosers that decreases slowly the performances of true positives diagnosis. In such a case, our system may be used 1) as a tool adapting itself reactively to short term evolutions of the web server environment and 2) as a tool supporting the administrator to adapt the signatures base on the long term. To support the administrators, our framework based on learnable signatures or models opens an new research direction in which the system would adapt their models from both their self-adaptation and interactive suggestion or correction from the administrator.

References

- [1] Nikto2. <http://www.cirt.net/>, 2009.
- [2] T. Bass. Intrusion detection systems and multisensor data fusion. *Communication of the ACM*, 43(4):99–105, 2000.
- [3] I. Bojanic. On-line adaptive IDS scheme for detecting unknown network attacks using HMM models. Master’s thesis, University of Maryland, 2005.
- [4] D. Bolzoni and S. Etalle. Boosting web intrusion detection systems by inferring positive signatures. Technical Report TR-CTIT-08-43, Enschede, June 2008.
- [5] Y.-C. Cheng, C.-S. Lai, G.-H. Lai, C.-M. Chen, and T. Chen. Defending on-line web application security with user-behavior surveillance. In *Proceedings of the Third International Conference on Availability, Reliability and Security (ARES 08)*, pages 410–415, 2008.
- [6] F. Cuppens and A. Miège. Alert correlation in a cooperative intrusion detection framework. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 202, 2002.
- [7] CVE. Vulnerability type distributions in CVE (2001-2006). <http://cve.mitre.org/>, 2006.
- [8] D. Denning. An intrusion-detection model. *IEEE Transactions on Software Engineering*, 13:222–232, 1987.
- [9] M. F. et al. Symantec internet security threat report – trends for 2008. Technical report, symantec, Avril 2009.
- [10] G. Gu, A. A. Cárdenas, and W. Lee. Principled reasoning and practical applications of alert fusion in intrusion detection systems. In *Proceedings of ASIACCS’08*, pages 136–147, 2008.
- [11] K. L. Ingham, A. Somayaji, J. Burge, and S. Forrest. Learning DFA representations of HTTP for protecting web applications. *Computer Networks*, 51:1239–1255, 2007.
- [12] C. Kreibich and J. Crowcroft. HoneyComb: Creating intrusion detection signatures using honeypots. *SIGCOMM Computer Communication Review*, 34(1):51–56, 2004.

- [13] C. Kruegel and G. Vigna. Anomaly detection of web-based attacks. In *Proceedings of the 10th Conference on Computer and Communications Security (CCS '03)*, pages 251–261, 2003.
- [14] T. Lane and C. E. Brodley. Approaches to online learning and concept drift for user identification in computer security. In *KDD*, pages 259–263, 1998.
- [15] M. E. Locasto, J. Parekh, A. D. Keromytis, and S. J. Stolfo. Towards collaborative security and P2P intrusion detection. In *Proceedings of the 6th Annual IEEE SMC Information Assurance Workshop (IAW)*, pages 333–339, 2005.
- [16] T. Murgue. *Extraction de données et apprentissage automatique pour les sites web adaptatifs*. PhD thesis, Université Jean Monnet de Saint-Etienne, 2006.
- [17] P. Ning, Y. Cui, and D. S. Reeves. Constructing attack scenarios through correlation of intrusion alerts. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 245–254, 2002.
- [18] X. Qin and W. Lee. Attack plan recognition and prediction using causal networks. In *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC '04)*, pages 370–379, 2004.
- [19] I. Ristić. ModSecurity, the open source web application firewall, 2008. www.breach.com.
- [20] M. Roesch. Snort: Lightweight intrusion detection for networks. In *Proc. of the 13th Conference on Systems Administration*, pages 229–238, 1999.
- [21] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.
- [22] G. Singh, F. Masseglia, C. Fiot, A. Marascu, and P. Poncelet. Data mining for intrusion detection: from outliers to true intrusions. In *Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'09)*, 2009.
- [23] S. Srinoy. An adaptive IDS model based on swarm intelligence and support vector machine. In *Proceedings of the International Symposium on Communications and Information Technologies (ISCIT '06)*, pages 584–589, 18 2006-Sept. 20 2006.
- [24] J. Tian, W. Zhao, R. Du, and Z. Zhang. D-S evidence theory and its data fusion application in intrusion detection. In *Proceedings of the 6th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT'05)*, 2005.
- [25] E. Tombini, H. Debar, L. Mé, and M. Ducassé. A serial combination of anomaly and misuse IDS applied to HTTP traffic. In *Proceedings of the 20th annual computer security application conference (ACSAC '04)*, pages 428–437, 2004.

- [26] N. Verma, F. Trouset, P. Poncelet, and F. Massegia. Détection d'intrusions dans un environnement collaboratif sécurisé. In J.-G. Ganascia and P. Gançarski, editors, *Actes de la conférence Extraction et Gestion des Connaissances (EGC 2009)*, pages 301–312, 2009.
- [27] W. Wang, F. Massegia, T. Guyet, R. Quiniou, and M.-O. Cordier. A general framework for adaptive and online detection of web attacks. In *Proceedings of the 18th international World Wide Web conference (WWW 2009)*, page To appear, 2009.
- [28] K. Yamada. A new combinaison of evidence based on compromise. *Fuzzy sets and systems*, 159:1689–1708, 2007.



Centre de recherche INRIA Rennes – Bretagne Atlantique
IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
<http://www.inria.fr>
ISSN 0249-6399