Carlos Serrão
Vicente Aguilera Díaz
Fabio Cerullo (Eds.)

Web Application Security

OWASP The Open Web Application Security Project

IBWAS'10

2nd OWASP Ibero-American Web Application Security Conference

16.17 December 2010

ISCTE . Instituto Universitário de Lisboa

LISBOA . PORTUGAL

Carlos Serrão, Vicente Aguillera Díaz,
Fabio Cerullo (Eds.)

Web Application Security

2nd. OWASP Ibero-American Web Application Security Conference
IBWAS 2010
Lisboa, Portugal, December 16-17, 2010
Revised Selected Papers

Volume Editors

Carlos Serrão
ISCTE-IUL Lisbon University Institute
OWASP Portugal
Lisboa, Portugal
E-mail: carlos.serrao@iscte.pt

Vicente Aguilera Díaz
Internet Security Auditors
OWASP Spain
Barcelona, Spain
E-mail: vicente.aguilera@owasp.org

Fabio Cerullo
OWASP Ireland
OWASP Global Education Committee
Dublin, Ireland
E-mail: fcerullo@owasp.org

ISBN-13 **978-1-937087-00-5**

Preface

Following the success of IBWAS'09, the 2nd. OWASP Ibero-American Conference on Web Applications Security (IBWAS'10) joined once more the international web application security academic and industry communities to present and discuss the major aspects of Web applications security.

OWASP IBWAS'10 brought together application security experts, researchers, educators and practitioners from the industry, academia and international communities such as OWASP, in order to discuss open problems and new solutions in the application security field. In the context of this track, academic researchers were able to combine interesting results with the experience of practitioners and software engineers.

The conference was held at the ISCTE-IUL Lisbon University Institute (ISCTE-IUL), in Lisbon, Portugal. During two days, more than 100 attendees enjoyed different types of sessions, organized in different topics. Two renowned keynote speakers, diverse invited speakers and several accepted communications were presented and discussed at the conference. The conference agenda was distributed in two major tracks - technical and research - organized according to the following topics:

- Secure application development
- Security of service oriented architectures
- Threat modelling of web applications
- Cloud computing security
- Web applications vulnerabilities and analysis
- Countermeasures for web application vulnerabilities
- Secure coding techniques
- Platform or language security features that help secure web applications
- Secure database usage in web applications
- Access control in web applications
- Web services security
- Browser security
- Privacy in web applications
- Standards, certifications and security evaluation criteria for web applications
- Attacks and Vulnerability Exploitation.

The OWASP IBWAS'10 organizers, would like to thank the different authors that have submitted their quality papers to the conference, and the members of the Programme Committee for their efforts in reviewing the multiple contributions that we have received. We would also like to thank to the amazing keynote and panel speakers for their collaboration in making IBWAS'10 such a success. Also, a special appreciation to all of our sponsors for making the necessary conditions available for the realization of the event.

Finally, we would like to thank the ISCTE-IUL for hosting the event and all their support.

Carlos Serrão
Vicente Aguilera Diáz
December 2010 Fabio Cerullo

Organization

Programme Committee

Chairs

Aguilera Díaz V., Internet Security Auditors, OWASP Spain, Spain
Cerullo F., OWASP Ireland, Ireland
Serrão C., ISCTE-IUL Instituto Universitário de Lisboa, OWASP Portugal, Portugal

Secretary

Paulo Coimbra, OWASP, Portugal

Members

Agudo I., Universidad de Malaga, Spain
Chiariglione L., Cedeo, Italy
Correia M., Universidade de Lisboa, Portugal
Costa C., Universidade De Aveiro, Portugal
Cruz R., Instituto Superior Técnico, Portugal
Delgado J., Universitat Politecnica De Catalunya, Spain
Dias M., Microsoft, Portugal
Elias W., OWASP Brasil, Brasil
Ferreira J., Universidade de Lisboa, Portugal
Filipe V., Universidade de Trás-os-Montes e Alto Douro, Portugal
Hernández-Goya C., Universidad De La Laguna, Spain
Hernando J., Universitat Politecnica De Catalunya, Spain
Hinojosa K., New York University, USA
Huang T., Pekin University, China
Kudumakis P., Queen Mary University of London, United Kingdom
Lemes L., Unisinos, Brasil
Lopes S., Universidade do Minho, Portugal
Marañón G., Consejo Superior de Investigaciones Científicas, Spain
Marinheiro R., ISCTE-IUL Instituto Universitário de Lisboa, Portugal
Marques J., Instituto Politécnico de Castelo Branco, Portugal
Metrôlho J., Instituto Politécnico de Castelo Branco, Portugal
Muro J., Universidad Politécnica de Madrid, Spain
Neves E., OWASP Brasil, Brasil
Neves N., Universidade de Lisboa, Portugal
Oliveira J., Universidade De Aveiro, Portugal
Raduà F., Universitat Oberta de Catalunya, Spain
Ribeiro C., Instituto Superior Técnico, Portugal
Roman R., Universidad de Málaga, Spain
Saeta J., Barcelona Digital, Spain
Santos O., Instituto Politécnico de Castelo Branco, Portugal
Santos V., Microsoft, Portugal
Sequeira M., ISCTE-IUL Instituto Universitário de Lisboa, Portugal
Sousa P., Universidade de Lisboa, Portugal
Torres V., Universitat Pompeu Fabra, Spain
Vergara J., Universidad Autónoma de Madrid, Spain
Vieira M., Universidade de Coimbra, Portugal
Villagrá V., Universidad Politécnica de Madrid, Spain
Yagüe M., Universidad de Málaga, Spain
Zúquete A., Universidade de Aveiro, Portugal

Table of Contents

Papers

OntoLog: A Security log analyses tool using web semantic and ontology

Clóvis Holanda do Nascimento[1], Rodrigo Elia Assad[1,3], Bernadette Farias Lóscio[2,3], Silvio Romero Lemos Meira [1,3]

[1]Centro de Estudos e Sistemas Avançados do Recife(CESAR) - Recife – PE – Brazil

[2]Departamento de Computação
Universidade Federal do Ceará (UFC) – Fortaleza, CE – Brazil

[3]Centro de Informática
Universidade Federal de Pernambuco (UFPE) – Recife, PE – Brazil

clovishn@gmail.com, assad@cesar.org.br, bernafarias@lia.ufc.br, srlm@cesar.org.br

Abstract. Along with the growth of available information on the internet, grows too the number of attacks to the Web systems. The Web applications became a new target to those invaders, due to the popularization of the Web 2.0 and 3.0, as well as the access to the social networks system's API's, the cloud computing models and SaaS. In this context, the identification of an eventual attack to those applications has become an important challenge to the security specialists. This article presents a proposition which uses Semantic Web concepts to the definition of an ontology for the analysis of the security log archives of WEB applications, with the intent of improving the classification of the attacks occurred and identification of the related events.

Keywords

Keywords: Security, Log Analyses, Ontology.

1. Introduction

Log Analyzes to search for information that can provide data about the process of identifying evidence, events and user profiles related to the system, consists in an ordinary and necessary activity for the teams that administer and manage systems. With the growth and popularization of Web systems (Cannings 2008, LYTRAS 2009), the volume of information generated in logs has grown considerably.

The growth of generated logs made the most common techniques used to analyze them, such as looking for evidence of certain attacks and even compromising those by finding patterns in the logs, not as effective as they were before (Berner- Lee, 2001). This scenario become even more complex when there is the need to identify co-relation between the events that are in the logs, such as identifying which operations a determined user in which systems in the last 3 days?

Alongside the problems described, we are maturing the definition of what can be defined as an attack and how an eventual attacker would use it (OWASP 2008, US-CERT 2009), what allowed to be adopted more sophisticated mechanisms, generating detailed data about the event, but making the log analysis more complicated.

In this context, the use of Semantic Web technologies, specifically the use of ontologies, in the context of security log analysis, showed itself as a possibility of improving the results of the searches in the log files. Generally, is expected that the ontologies can help in the interpretation process of interpretation of the great diversity of information that are present in this kind of archive.

Fundamentally, the role of ontology is to enable the construction of a representation model of a given area through the representation of a terms and relations vocabulary (Studer, Decker et al., 2000). According to Gruber (1996), Ontology is a formal and explicit specification of a shared conceptualization. Thus, as a formal specification, the ontology can be processed by computer software with precision in the analysis, facilitating the search in the log files and thus improving the efficiency of the results analysis.

This article aims to concisely present the proposal for the use of ontologies to analyze and process logs generated by web application firewall [ModSecurity 2009], identifying the generated types of information, its importance and the problems related to the log files .

The remaining sections of this paper are divided as follows: Section 2 presents the difficulties related to log analysis and security. Section 3 presents the use of ontologies for log analysis. Section 4 presents the results of the experiment. Finally, Section 5 presents the conclusions.

2. Log analysis difficulties and Security

The information stored in the logs are invaluable to the security area, for they have the attacks records, input ports, IP numbers, evidence of invasion, typed commands, among others. In addition, logs can be considered as a source in constant growth, due to the use of systems on a daily basis.

Kimball and Merz (Kimball 2000) present some problems found in the log files, such as; multiple file formats, dispersed information, incomplete, inconsistent and irrelevant data, which makes the analysis and extraction of information from these files harder to accomplish.

The security auditor or system administrator has as part of their daily routine duties a rather hard activity, the research and analysis of logs. This task is considered difficult and time consuming, because the log files are created without a semantic description of their format, making the extracting of the meaning of the data impracticable, showing only the words typed in the search, resulting in poor quality results. According to Guarino (1995), this limitation occurs because when the data is generated without the use of ontology, it can present ambiguities and vagueness of elements. This situation becomes more serious when we face major files with gigabytes of information.

3. The use of the Ontology for log analysis

There are various definitions found in literature about what is ontology. Originally the term was born in the field of philosophy, being a word of Greek origin, which deals with the nature of being and its existence. In the field of Computer Science, it was first applied in the artificial intelligence field to computational representation of knowledge, being considered an abstract model of a domain. Below are some of the most used definitions for the term ontology:

- According to Gruber (1996), "Ontology is a formal and explicit specification of a shared conceptualization."
- The W3C consortium (W3C 2010) defines ontology as: "the definition of terms used to describe and represent an area of knowledge."
- According to Noy and McGuinness (2001) there is no one correct way to model a domain, meaning that there is more than one way to develop an ontology.

The basic components of ontology are classes (organized in taxonomy), relations (used to connect the domain concepts), axioms (used to model sentences that are always true), properties (describe characteristics common to the instances of a class or relationships between classes) and instances (used to represent specific data).

Ontologies are used for modeling data from specific domains and also allow inferences to discover implicit knowledge in these. Despite the considerable increase in the use of ontologies, build a complete ontology covering all the expressiveness of the domain continues to be a hard work, making the work of a multidisciplinary team a necessity, in which case it would be an ontology engineer, a security expert, among others, and acting in a participatory and integrated.

More specifically, in this work, we are interested in building an ontology for the representation of data available in security logs of web applications. In this context, ontologies can be useful for improving the classification of the attacks occurred and the identification of related events.

On the next session we presents an overview of ontology, and describe the methodology used for its creation.

3.1 General description of the proposed ontology

The proposed ontology, OntoSeg, has as main objective the representation of data generated by the application firewall log ModSecurity on this work. From a detailed analysis of several samples of the log we identified various classes and their relations. Table 1 presents a brief description of the main classes that compose the ontology for the representation of the security log:

Class	Definition
Audit log header	Represents the log header, and contains the following information: date and time, ID (transaction ID, with a unique value to each transaction log), source IP, source port, destination IP and port of destination.

Request Headers	Represents the request Header, contain the information of the request and the header of the solicitation that was sent by the client.
Request Body	Represents the body of the transaction, contains the contents of the client request.
Intended Response Headers	Represents the status line and headers, this part is reserved for future use, so it is not implemented
Intended Response Body	Represents the body of the response of the transaction, the response body contains the actual case the transaction is not intercepted.
Response Headers	Represents the header of the actual response sent to the the client, contains data and headers
Response Body	Represents the body's effective response to the request, but this part is reserved for future use, so it is not implemented
Audit Log Trailer	Represents additional data, contains additional meta data of the transaction obtained from the web server or ModSecurity
Reduced Multipart Request Body	Represents the body of the request reduced, Part I is designed to reduce the size of the request body with irrelevant content from the standpoint of security, transactions that deal with uploading large files tend to be big
Multipart Files Information	Represents information about the files, the objective of this part is to write the metadata information about the files contained in the request body, this part is reserved for future use, so it is not implemented
Matched Rules Information	Represents the rules, contains a record with all the rules that occurred during the processing of transactions ModSecurity
Audit Log Footer	Represents the end of the log, your goal is just to indicate the end of a log file

Table 1– Main classes of proposed ontology

The Figure 1 represents the relationships proposed in OntoSeg. As can be noted several branches show the complexity of the domain.

The basic relationship between the classes are performed using the property ID (transaction ID, with unique value to each transaction), derived from the log ModSecurity that defines the ModSecurity_log main class, and from this we have the following derived classes:

a) Reponse_headers: contains all the information related to the HTTP header response

b) Request_headers: contains all the information related to the HTTP request header

c) Audit_log_header: contains all the information related to the header of IP and TCP

d) There still is a set of information that derive from ModSecurity_log class that contains information about the HTTP message body
From these subclasses we have the derivation of other subclasses that contains each of the basic elements of ontology OntoSeg.

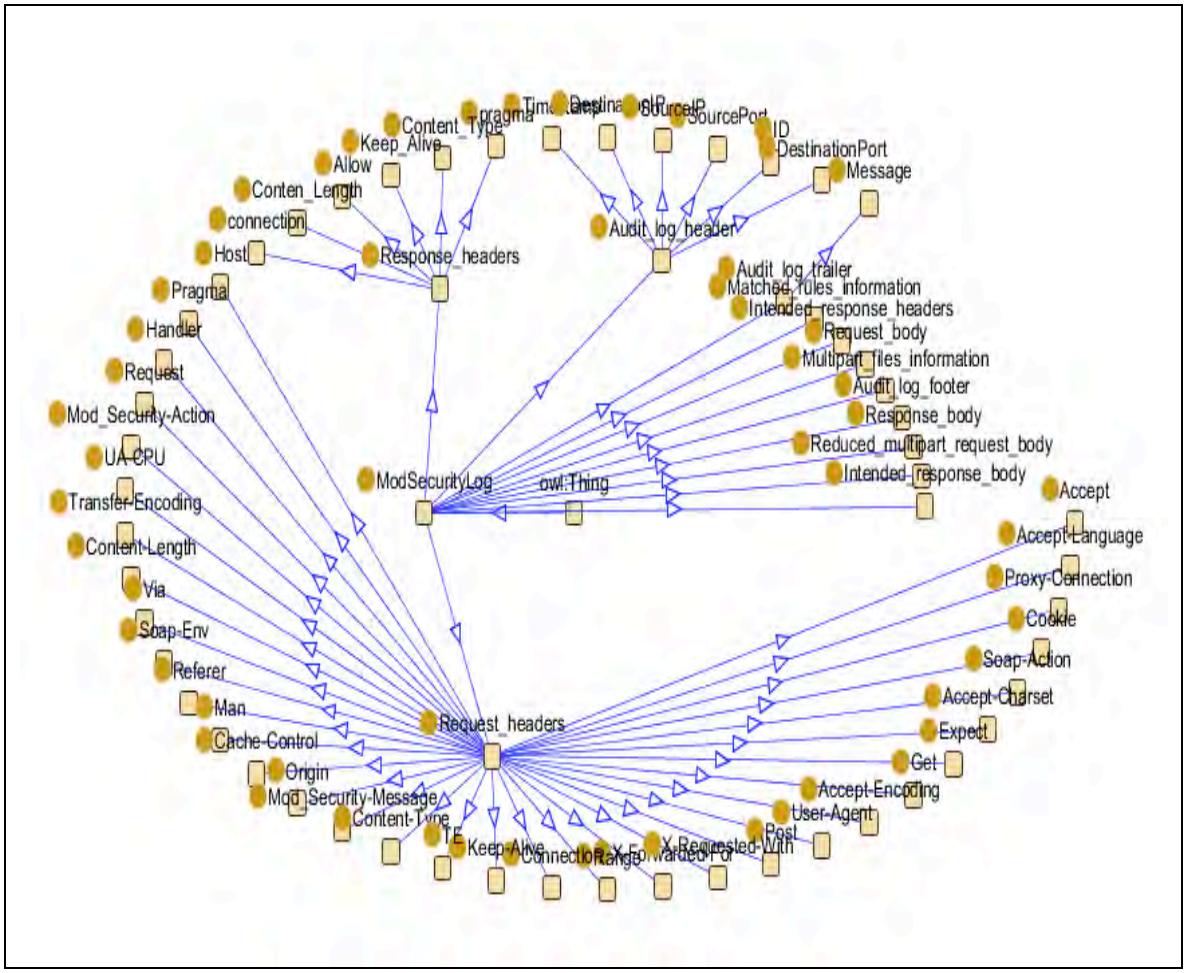


Figure 1. Conceptual model of ontology in protégé - radial view. (Source: Jambalaya plug-in.)

Figure 2 represents the definition of the SourcePort, Get, and SourceIP Content-Type subclasses, with their classes, respectively, Audit_log_header, Request_headers, and Request_headers Audit_log_header:

```
<owl:Class rdf:ID="SourcePort">
  <rdfs:subClassOf>
    <owl:Class rdf:ID="Audit_log_header"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Get">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Request_headers"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Content-Type">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Request_headers"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="SourceIP">
```

```
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Audit_log_header"/>
    </rdfs:subClassOf>
  </owl:Class>
```

Figure 2. Excerpt from the code of the ontology developed in the owl language.

3.2 Creation proccess of the proposed ontology

The experiment with the use of ontologies for log files analysis was developed using an actual log of Web applications developed by the company CESAR (Center for Advanced Studies and Systems of Recife) that contained a series of attacks by these applications. This server has about 212,000 page views per month from 35,900 different addresses. The Web server used is Apache running in the Linux operational system that uses the ModSecurity program as a firewall of web applications. The filter rules include several filters and for most of the known WEB attacks, for security reasons we can not detail how they are configured.

From the analysis of logs generated by the application firewall it was possible to identify the major classes of the ontology and their relationships. The universe of terms that compose the proposed ontology was defined based on the log of ModSecurity, that records all the transaction data from the CESAR WEB systems. The classes were defined according to the division of parts of the ModSecurity log and the subclasses were established based on the configuration of keywords enabled by the administrator of ModSecurity, which were 41 words. This amount could be higher or lower depending on the established configuration, the instances are data that are part of the logs.

Figure 3 shows an excerpt of the ModSecurity log, where you can see that there is no semantic description associated with its data, thus limiting the expressiveness of the concepts.

```
--a2ab5e2b-A--
[13/Apr/2010:13:31:01 --0300] S8SAOMja0G0AAAHXEGgAABeM 2.2.2.2 34571
192.168.1.1 80
--a2ab5e2b-B--
GET /teste.asp?VARIAVEL=I'%20and%201=1%20and%20''='' HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)
Host: www.example.com
Cookie: ASDF=ABC; ASPSESSIONIDQSCRBASD=PADSFAFASDADASdAADA
--a2ab5e2b-F--
HTTP/1.1 200 OK
X-Powered-By: ASP.NET
Content-Length: 88045
Content-Type: text/html; charset=ISO-8859-1
Expires: Tue, 10 Apr 2000 13:31:01 GMT
Vary: Accept-Encoding,User-Agent
--a2ab5e2b-E--

--a2ab5e2b-H--
Message: Warning. Pattern match
"(?:\b(?:(?:s(?:elect\b(?:.{1,100}?\b(?:(?:length|count|top)\b.{1,100}?\bfrom|from\b.{1,
100}?\bwhere)|.*?\b(?:d(?:ump\b.*\bfrom|ata_type)|(?:to_(?:numbe|cha)|inst)r))|p_(?:(
?:addextendedpro|sqlexe)c|(?:oacreat|prepar)e|execute(?:sql)?|makewebtask)|ql_(?
..." at ARGS:VARIAVEL.  [file
"/etc/httpd/conf/extra/mod_security/modsecurity_crs_40_generic_attacks.conf"] [line
"66"] [id "950001"] [msg "SQL Injection Attack"] [data "and 1="] [severity "CRITICAL"]
[tag "WEB_ATTACK/SQL_INJECTION"]
Message: Warning. Pattern match "\b(\d+) ?= ?\1\b|[\"'](\w+)[\"']  ?= ?[\"']\2\b" at
ARGS:VARIAVEL.  [file
"/etc/httpd/conf/extra/mod_security/modsecurity_crs_40_generic_attacks.conf"] [line
"70"] [id "950901"] [msg "SQL Injection Attack"] [data "1=1"] [severity "CRITICAL"]
[tag "WEB_ATTACK/SQL_INJECTION"]
Apache-Handler: proxy-server
Stopwatch: 1271169080460234 798372 (133 856 -)
Response-Body-Transformed: Dechunked
Producer: ModSecurity for Apache/2.5.9 (http://www.modsecurity.org/); core
ruleset/1.6.1.
Server: Apache
WebApp-Info: "www" "-" "-"

--a2ab5e2b-Z--
```

Figure 3. Example of an excerpt from the log of ModSecurity, which represents a real SQL injection attack (some data has been changed to preserve confidentiality of the company).

For the creation of the ontology were chosen: a language for representing ontologies, a specific tool for working with ontologies, and some methodologies for the constructions of ontologies with defined roles. The below summarizes the choices that were made during the creation of ontology:

- Language for definition of the ontology

In order to create an ontology that can be semantically processed by computers, the OWL language was adopted, that has these characteristics, and currently is the language recommended by the W3C consortium.

- Methodologies for building ontologies

In the proposal the following methodologies were used to develop ontologies (SILVA, Daniel Lucas, 2008):

- *101*: Methodology of the simplified and interactive process, which has the following steps: Determine the domain and scope, consider reuse, list relevant terms, define classes and their hierarchy, define classes properties, set property values and creating instances.
- *Uschould and King*: Methodology consists in four distinct stages: Identifying the purpose of the ontology, construction, evaluation and documentation.
- *Methontology*: Methodology that suggests a life cycle of evolutionary model, composed by the following phases: planning, specification, knowledge acquisition, conceptualization, formalization, integration, implementation, evaluation, documentation and maintenance.

• **Tool for the creation of the ontology**

The Protégé tool was chosen, what allows the construction and editing of ontologies through a graphical interface of easy interaction. It also allows the insertion of new capabilities by installing plug-ins. In our case the OWL and Jambalaya plug-ins were installed. Other features of this tool are the importing and exporting of ontologies, open source and be developed in Java. Based on the comparative study of SILVA, Daniel Lucas; ROCHA SOUZA, Renato; ALMEIDA, Mauricio Barcelos (2008), which presents the methodologies for building ontologies, three methods were selected according to the activities involved in the proposal, as described in Table 2.

	METODOLOGIAS		
	Uschould e King	101	Methontology
Determination of the propose of the ontology.	■		
Definition of classes, properties and instances.		■	
Construction of the conceptual model.			■
Implementation.		■	
Verification and validation.			■
Maintenance.			■

Table 2. Methodologies used in each step of the life cycle of the ontology.

4. Results and tests

Among the benefits of using ontologies are: the classification of the terms of the logs, relationships, inferences, formalization, reuse, sharing, among others, we will show some gains in the context of research.

To prove the improvements in research in the generated logs, was used the ontology described in the previous sections to analyze the logs. In addition, the ontology is necessary to use a query language, called SPARQL, which since January 2008 is recommended as the standard by the W3C consortium.

This language allows querying ontologies through clauses, which can combine several classes at the same time and also make filters in the searches. A compiler for SPARQL is already integrated in the Protégé tool.

To strengthen the evidence of the improvements in research involving the use of ontology in comparison with the traditional keyword searches, see below some situations:

1. *A security auditor must analyze the log events that occurred in the 10th day of a month until the 10th day of the following month, that has the ip address range of 192.168.0.13 to 192.168.0.97 with destination ports 3306 and 443, and is on schedule from 21:30 to 24:00 h.*
2. *A security auditor wishes to know what IP adress or group of adresses generated more attacks and in what times.*

Considering the situations above, we have one, between two ways to proceed:

1) Search with the use of Ontology:

It would be enough to use the select command with the filter clause containing the classes mentioned above, and be answered with only the desired data to analyze.

It could also create a search interface using the language SPARQL query language or another, to prevent the auditor to need to know and type the commands queries.

2) Searches without the use of ontologies :

Quite difficult to be generated because that for the auditor to make the analysis he wants, he would first have to read many logs manually separating by keyword and then make de co-relation, with the risk of losing data, since only the search for information, he will be ignoring the other data you want.

Below are the consult solutions in SPARQL to the situations described above:

```
SELECT ?ID ?DestinationIP ?DestinationPort ?SourceIP ?SourcePort ?Timestamp ?Cookie ?mod_security-message
WHERE { ?ID rdf:type :ID . ?DestinatioIP rdf:type :DestinationIP .
?DestinationPort rdf:type :DestinationPort . ?SourceIP rdf:type :SourceIP . ?SourcePort rdf:type :SourcePort . ?Timestamp rdf:type :Timestamp . ?Cookie rdf:type :Cookie . ?mod_security-message rdf:type :mod_security-message
FILTER((?TimeStamp > xsd:date("2010-07-09") && ?TimeStamp < xsd:date("2010-08-11") && (?DestinationIP > 192168012 && ?DestinationIP < 192168098) && (?DestinationPort = 3306 || ?DestinationPort = 443) && (?TimeStamp > xsd:time("21:29:00") && ?TimeStamp < xsd:time("24:00:01"))) }
```

Figure 4 – SPARQL consult in the Ontology, Situation 1.

```
SELECT ?ID ?SourcerIP ?DestinationIP ?Timestamp                                    WHERE {
?ID  rdf:type  :ID . ?SourceIP rdf:type :SourceIP . ?DestinationIP rdf:type :DestinationIP .  ?Timestamp
rdf:type :Timestamp       GROUP BY  ?SourceIP  } ORDER BY ASC(?SourceIP)
```

Figure 5 – SPARQL Consult in the Ontology, situation 2.

In this sense, the implemented ontology fulfilled its role very well, according to what was previously planned, the searches were carried out in a simple way in the ontology producing the most interesting and visible results in comparison with the traditional consultations using only key words, obtaining better results for event logs identification.

It is seen that with the approach proposed in this paper, the activity log analysis was made simple and independent of the complexity of the log and the generated data volumes allowing the realization of co-relations between events more efficiently.

5. Conclusion

This study aimed to take the first steps in using ontologies for analysis of security logs. For that purpose the logs generated by the program ModSecurity were initially used. As a starting point the log that was generated by this tool on a web server of CESAR (Center for Advanced Studies and Systems of Recife) was used. The ontology modeling was accomplished from the understanding of the logs the model the ontology.

The performed tests proved that there was an easier log interpretation and analysis, allowing the performing of more complex consultations and the implementation of co-relation of events very effectively.

Finally, we proved that the demand for log audits that use ontologies is very large, for the tools and current research procedure is very limited, constituting a critical point in analyzing logs. In this context, this work was made to contribute to the attending of this demand.

6. Bibliography

ALLEMANG, D. e HENDLER, J.. (2008) **Semantic Web for the Working Ontologist**, Effective Modeling. in: RDFS and OWL, Cambridge, Morgan Kaufmann, 2008.

ALMEIDA M. B, BAX M. P. **Uma Visão Geral sobre Ontologias:** pesquisa sobre definições, tipos, aplicações, métodos de avaliação e de construção in Cin:Inf.,Brasilia, 2003.

ANTONIOU, Grigoris e HARMELEN, Frank V., ***A Semantic Web Primer***, Second Edition. London, The Mit Press, 2008.

BERNERS-LEE, T.; Hendler, J.; Lassila, O. ***The Semantic Web***, Scientific American, New York, 2001.

BRANDÃO, Ana Rosa Alves Franco; LUCENA, Carlos José Perreira, **Uma Introdução à Engenharia de Ontologias no contexto da Web Semântica**, Rio de Janeiro, 2002.

BREITMAN, Karin K. ***Web Semântica, a Internet do Futuro***. Rio de Janeiro: LTC, 2005.

CANNING S, Rich. Dwivedi, Himanshu. Lackey, Zane. **Hacking Exposed™** WEB 2.0: Web 2.0 Security Secrets And Solutions. New York: Mc Graw Hill, 2008. - DOI: 10.1036/0071494618

CHAVES, Marcirio Silveira. **Um estudo sobre XML, *Ontologias e RDF(S)***, 2001.

GRUBER, T. R. ,**Toward principles for the design of ontologies used for knowledge sharing**. In Formal Ontology in Conceptual Analysis and Knowledge Representation. Kluwer Academic Publishers, 1996.

Guarino, Nicola, Poli, Roberto. **Formal ontology, conceptual analysis and knowledge representation**. International Journal of Human-Computer Studies. 1995.

Kimball, R. and Merz, R. **The Data Webhouse Toolkit**, New York, John Wiley and Sons, Inc, 2000.

Lytras, Miltiadis D. Damiani, Ernesto. Pablos, Patricia O. **WEB 2.0 The Business Model**. New York: Springer, 2009. ISBN-13: 978-0-387-85894-4

Mcguinness, D. L. e F. V. Harmelen. **OWL Web Ontology Language Overview**. W3C World Wide Web Consortium (http://www.w3.org/TR/owl-features/), 2004.

MODSECURITY. Breach, **ModSecurity 2 Data Formats**, 2009, Copyright © 2004-2009 Breach Security, Inc. (http://www.breach.com)

Noy, N.F.;McGuinness, D.L. "**Ontology Development 101**: A Guide to Create Your.

Owasp, 2008, **owasp testing guide** 2008 v3.0. Disponível em: http://www.owasp.org/index.php/category:owasp_testing_project

Prud'hommeaux, Eric e Andy Seaborne. **SPARQL Query Language for RDF**. W3C- World Wide Web Consortium (http://www.w3.org/TR/rdf-sparql-query/). 2008.

Protégé; **Ontology Editor and Knowledge Acquisition System**

(http://protege.stanford.edu/>)

Seaborne, A. "**RDQL - A Query Language for RDF**". W3C, 2004.

Studer, R., S. Decker*, et al.* **Situation and Perspective of Knowledge Engineering**.

Stuttard, Dafydd; Pinto, Marcus. **The Web Application Hacker's HandBook**, Discovering and Exploiting Security Flaws, 2008.

Us-cert - **technical cyber security alerts**, 2009. Disponível em: http://www.us-cert.gov/cas/techalerts/ . Ultimo acesso em: 29/04/2009

W3C, 2010, http://www.w3.org/ ultimo acesso em 04/08/2010.

Security Quality Assurance on Web-based Application Through Security Requirements Tests Based on OWASP Test Document: *elaboration, execution and automation*

Rodrigo Elia Assad[1,2], Tarciana Katter[1], Felipe Ferraz[1,2], Silvio Romero Lemos Meira [1,2]

[1]Centro de Estudos e Sistemas Avançados do Recife(CESAR) - Recife – PE – Brazil

[2]Centro de Informática
Universidade Federal de Pernambuco (UFPE) – Recife, PE – Brazil

assad@cesar.org.br, tarcykatter@gmail.com, fsf@cin.ufpe.br, srlm@cesar.org.br

ABSTRACT

Historically, it is well known that issues related to security of software applications are normally omitted by the development teams owing to a lack of expertise or knowledge in security policies. With the emergence of WEB 2.0, this situation became more serious. Entire systems, complex or not, have outstanding access availability and therefore are highly vulnerable to threats. This work aims to discuss how the security requirements should be elaborated in order to making easier the execution of its tests and consequently improving the quality of the solution developed.

Keywords

Keywords: security, requirements, tests, validation, quality assurance.

1. Introduction

Nowadays, web-based applications offer access to complex systems differently of Web 1.0 systems based exclusively on static texts. Web systems have been evolved by using dynamic interface that enables user interactions, previously only used on desktop applications [CANNINGS 2008], besides incorporating the trends of Web 2.0[LYTRAS 2009]. Consequently, these systems show a outstanding growth on number of access as well as complexity of features offered.

The increasing use of technology to implement Internet Business Interfaces has allowed identifying a growing number of flaws and vulnerabilities in systems. The need to be more careful about system security and information implies the need to have a life cycle of software development activities focused on specific security as a means to achieve greater assurance of integrity, reliability and availability of information.

To define the security needs demanded by an application, it is required experienced and trained stakeholders with abilities in all disciplines with a focus on security [HOWARD 2006]. Throughout the development of a web application, it is important that the activities of elicitation and specification of requirements to be followed by a process of validation. It is essential to track and approve if the security requirements are being satisfied on applications. The traceability of non-functional requirements is naturally a complex task, because in general, they affect the entire system. To carry out successful safety tests on a application, is necessary to identify what types of vulnerabilities could enable attacks on the system. In other words, we must understand where the flaws may be present to learn how to avoid them.

From this point of view, we identified that one of the possible causes to security flaws relies on the low quality of software security requirements and consequently in its implementation, validation and tests phases.

This paper aims to propose a guidance that can be used by software analysts though the process of security requirements elaboration. In addition to that, it is proposed a generic template to the elaboration of software requirements documentation as well as a process that can be applied to software test and validation. On section 2 we present a discussion about safety requirements, on section 3 we show how we create the guide, on section four we present how create a requirements based on OWASP proposal, on section 5 we present how we can make security tests based on the requirements and on section 6 we show the results obtained applying the techniques presented on this paper.

2. Safety requirements

The software development process focusing on business requirements engineering elicitation, systems, applications and components development is far than just document the functional requirements of these. Even though, most analysts understand the necessities to elicit some quality requirements such as interoperability, availability, performance, portability and usability, many of them still sin with regard to addressing issues related to security.

Unfortunately, documenting specific security requirements is difficult. These tend to have a high impact in many functional requirements. Furthermore, security requirements are usually expressed in terms of how to achieve security not as the problem that needs to be resolved [Haleyv 2004].

This problem became more complex because most requirements analysts have no knowledge in Security. Some received training and has a general overview of some security mechanisms such as passwords and encryption rather than meet real requirements in this area [Yoder, 2003 Firesmith , Firesmith RHAS'03]

Safety requirements deal with how the system assets must be protected against any kind of evil attack [Haleyv 2004, Haley 2006]. An asset is something inside the system, tangible or not, that must be protected [ITTF]. An evil or attack is what a system must be protected, is a potential vulnerability that can reach a well. A vulnerability is a weakness of a system that an attack try to exploit. Security requirements are constraints on the functional requirements in order to reduce the scope of vulnerabilities [Haleyv 2004].

Donald Firesmith very simply consolidates the Security requirements, they are [Firesmith RHAS'03]:

Identification Requirements: Define the degree and mechanisms that an organization uses to know, or recognize it as external entities eg.: users, external applications or services before they can interact. Examples of Identification Requirements:

- The application must identify all of your client applications before allowing them access to its facilities.
- The application must identify all users before allowing them access to its facilities.
- The data center staff must identify the work before they have access to facilities.
- Access to the system must be identified.

Authentication Requirements: Define the degree and mechanism for an entity to verify, confirm or deny the identity presented by external users, external applications or services before they can interact. Examples of Authentication requirements:

- The application must verify the identity of all its users before allowing them access to its facilities.
- The application must verify the identity of all its users before they can change any type of a system.
- The application must verify the identity of all users before accepting any type of payment from them.
- The system should generate a confirmation cases, the user identification data are correct.

Authorization Requirements: Define the degree of access and privileges that a particular (s) entity (ies) or profiles will receive after being authenticated and validated by some part of the system. Examples of Authorization Requirements:

- The application must allow each user to access all your personal data.
- The application may not allow users to have access to information from other users.

- The application may not allow third-party services, or third parties have access to information relating to payments from its users.
- Application must associate a user profile to log in and validate the permissions of this profile.

Immunity Requirements: Defines the degree to which an organization protects itself from invasion, infection or changes on the part of malicious software, eg.: viruses, trojans, worms and malicious scripts. Examples of Immunity Requirements:

- The application must be able to disinfect of any file that is detected as harmful, if possible.
- The application should notify the security administrator and user logged in the event of detecting any harmful program, during a scan process.
- The application must be able to protect against possible contamination by programs, scripts, data, malicious.
- The application must contain mechanisms, efficient enough, to not allow certain data associated with users being altered by others.

Integrity Requirements: Defines the degree to which a communication or data, hardware and software, are protected from attempted bribery by third parties. Examples of Integrity Requirements:

- The application must prevent corruption, unauthorized data, messages and values, and any other party sent by the user.
- The application must ensure that the data stored therein are not corrupt.
- The application must prevent corruption, unauthorized data messages and values stored by the user.
- The application must ensure that user profiles are not altered, improperly, and thus be subject to data changes.

Intrusion Detection Requirements: Defines the degree to which an attempt at unauthorized access or modification is detected, recorded and notified by the system. Examples of requirements for intrusion detection:

- The application must detect and record all access attempts that fail in the requirements identification, authorization and authentication.
- The application must notify officials immediately when a profile without the correct permission tries to access a restricted area more than once.
- The application must notify those responsible for the daily safety of all crashes that occurred during the past 24 hours.
- The application should be able to apply the correct penalties in case of repeated failed attempts at access.

Requirements for Non - Repudiation: Defines the degree to which a portion of a given system prevents one party from interactions taken in, or the system, eg an exchange of messages, deny their involvement at any given time. Examples of Requirements Non - Repudiation:

- The application must be able to store data about transactions made by a user in order to be able to accurately identify which user made which transaction.
- The application must be able to store data about the actions made by each user in order to be able to accurately identify which user has what action.
- The application must provide reliable mechanisms to prove the authors of certain actions.

Privacy requirements: Also known as confidentiality. Defines the degree to which critical data and communications are kept private access by individuals or programs. Examples of Privacy Requirements:

- The application must ensure that users can not access sensitive data from other users.

- The application must ensure that any communication made by users can not be understood in case of capture.
- The application should allow users to have access only to information they were intended.
- The application must ensure that the data stored by a user will not be accessed or modified by other users even if they have the same privileges.

Security Audit Requirements: Defines the degree to which the team responsible for security have permission to check the status and use of security mechanisms through the analysis of events related to these. Examples of Security Audit Requirements:

- The application should be able to detect and store all transactions made by you
- The application must provide a mechanism through which the management staff can track the actions of each user within the system.
- The application must be able to organize by: The Action; The User; The date and time; Value (if applicable);

All the events and actions occurring in the system for possible future reference.

Fault Tolerance Requirements: Defines the degree to which an entity continues to execute its mission by providing essential resources to their users even in the presence of some kind of attack. Examples of Tolerance Requirements:

- The application can only present a single point of failure.
- The application can not have its own audit system off the air.
- The application must remain functional even if some of their internal systems are down.
- The application must have the ability to resume her vital modules in an automatic and transparent to the end user.

Physical Protection Requirements: Defines the degree to which an entity is physically protects another entity. Examples of Physical Protection Requirements:

- Data stored in hardware should be protected against physical damage, destruction, theft or unauthorized change.
- Those who handle the data must be protected against damage, death and kidnapping.
- Backup tapes must have a constant vigilance.
- The data server can not suffer physical data between certain hours.

Requirements for Maintenance of Security Systems: Defines the degree to which the system must maintain your security settings even after modifications and upgrades. Examples of Maintenance of Security Systems:

- The application must not violate its security requirements after upgrading of data, hardware or components.
- The application must not violate its security requirements after an exchange of data, hardware or components.
- Once implemented and validated the security requirements of the application, they can not be changed without notice and schedule.
- The application must not violate its security requirements after installation of new versions.

3. Elaboration of requirements from the OWASP testing guide

The of creation of a safety guide for web applications have more objectives then only help engineers understand the requirements and detail the security needs in a more profound and clear way, but also to serve as reference for the other phases of system development. The guide was formulated based on the list of most common mistakes and how to test them in a web application proposed by OWASP [OWASP

2008] and the requirements formulated in accordance with the standards proposed by [withall 2007]. We used the template specification documentation of use cases diverted [Talukdar 2008].

The guide combines the safety tests in accordance with the requirements proposed by [Firesmith RHAS'03]

Requirement	Owasp Test
Identification	OWASP-IG-003, OWASP-IG-004
Authentication	OWASP-AT-001, OWASP-AT-002, OWASP-AT-003, OWASP-AT-004, OWASP-AT-005, OWASP-AT-006, OWASP-AT-007, OWASP-AT-008, OWASP-AT-009, OWASP-AT-0010
Authorization	OWASP-AZ-001, OWASP-AZ-002, OWASP-AZ-003
Imunity	OWASP-IG-005, OWASP-IG-006, OWASP-CM-002, OWASP-CM-003, OWASP-CM-006, OWASP-CM-008, OWASP-DV-001, OWASP-DV-002, OWASP-DV-003, OWASP-DV-004, OWASP-DV-005, OWASP-DV-006, OWASP-DV-007, OWASP-DV-008, OWASP-DV-009, OWASP-DV-0010, OWASP-DV-0011,OWASP-DV-0012,OWASP-DV-0013,OWASP-DV-0014, OWASP-DV-0015,OWASP-WS-002, OWASP-WS-003, OWASP-WS-004, OWASP-WS-005, OWASP-WS-006, OWASP-WS-007, OWASP-AJ-002
Integrity	OWASP-SM-001, OWASP-SM-002, OWASP-SM-003, OWASP-SM-004, OWASP-SM-005
Intrusion Detection	OWASP-CM-005
Não – Repudiation	
Privacity	OWASP-IG-001, OWASP-CM-001
Security Audity	
Fault Tolerance	OWASP-DS-001, OWASP-DS-002, OWASP-DS-003, OWASP-DS-004, OWASP-DS-005, OWASP-DS-006, OWASP-DS-007, OWASP-DS-008
Physical protection	
Maintenance of Security Systems	OWASP-CM-004, OWASP-CM-007

As the guide is based on existing faults observed that it is possible to reuse the information for different types of web systems, because the problems are belonging to well-defined classes and we can write reusable requirements so efficiently. This result was confirmed and will be described in the results section.
Each point of the guide has been designed to support one or several phases of development of specific life cycle of system development.
The guide can be written and updated with the help of the entire team, that is, the idea is that each profile is responsible for updating each section of the guide, thus creating a greater exchange of information and by leveling all. Well as allowing new faults to be incorporated as they arise.

4. Writing the security requirements document

From the moment you have a guide that identifies flaws, the next step is to map them into the application being developed. Thus the following stage of the development cycle of software being developed is the analysis of requirements.

According to [SINDRE 2002], templates are important because they encourage analysts to write clear and simple actions sequences. The template suggested by this work helps to orient analysts to document functional and non-functional requirements based on the security guide.

After analyzing the SIG process to elicit non-functional requirements proposed by [CHUNG 2000], it was possible to identify important points that have to be presented on requirement documents.

Taking into consideration those points and information gathered on the security guide, we have formulated a template with the following points:

- *Security goals:* describe what security goal must be achieved by te system. This item is composed of identifier, title and description.

- *Associated Requirements:* requirements hat belongs to the goal. This information can be obtained from the security guide according to the selected goal. This item is composed of:
    - Identifier – unique requirements identifier
    - Title – requirement title.
    - Description – a detailed description of the functioning or behavior of the requirement, and, if possible, which functional requirements will be impacted for them;
    - Priority – requirement priority to be implemented on the system;
    - Impact – what is the impact if the vulnerabilities occur?
    - Probability – what is the probability of the vulnerabilities occur?;
    - Vulnerabilities – what vulnerabilities are associated to this requirement? ;
    - Mitigation actions – what mitigation actions that are proposed on the security guide were selected to solve the problem?

By using this guide as a reusable source of security requirements, we can start the elicitation need for security for a specific web system, as will be shown on the following sections.

4.1 Building the requirement document

To build a requirement document, we are going to select the security goals and its associated requirements that are essentially important for the systems to be developed. In this case, we have created a checklist according to the guide that can be easier and usual for understating the client needs. The checklist must contain some other pieces of information beyond those cointained in the guide. Some questions were formulated to help on the specification of security requirement process:

1. Is there a security policy on the company?
2. Is there a structured environment to receive the data?
3. What are the features that give access to restricted information?
4. Is there any access control mechanism?
5. Is there any access policy mechanism?
6. Is the information classified according to the security level assigned to it?
7. What language will be used?

After answered all the questions above, it is necessary to understand the system domain to select, within this domain, what are the client security needs. To select these needs, we are going to use the guide according to the security goals. As an example, if the system has restricted data access, we can consider as a security goals data integrity, access control and its decomposed requirements. The checklist list thus is used to select what are the adequate and possible for the system to be developed.

By selecting the security requirements that will compose this work, we propose that:

1. Verify the associated vulnerabilities that are listed and related to the security guide.
2. Define which patterns and solutions proposed on the guide will be used on the system
3. Evaluate the impact f each vulnerability;
4. Evaluate the probability of each vulnerability happening;
5. Perform a risk analysis;
6. Priorize requirements.
7. Relate dependent requirements.

5. Security Tests

In this section, we are going to present a mechanism to map, write, execute and automate security tests.

5.1 Planning tests

The test phase should start at the early stages of any project development. Postponing the development and the validation of test to the last phase may cost more and generate a lot of extra work. To try to minimize this impact, we are going to use the following strategy:

1. Revision of the requirement document – the requirement must be measurable, clear and testable.
2. Revision of the code – taking into consideration the points that needed to be checked described on the security guide;
3. Manual execution of the test cases;
4. Automatic execution of test cases;
5. Execution of penetration tests.

5.2 Mapping security Test Cases

To map security tests with requirements, we are going to use a decision table technique [PEZZÉ 2008], normally used to write functional test. However, this technique will be used for mapping scenarios according to the described vulnerabilities on the requirement and in which part of the system it should be applied.

The first step is to choose a requirement to be tested and then verify the environment and features that this requirement can be applied. After that, we are going to use a cause and effect graph technique with some modifications.

To use the graph technique adapted to security requirement we must:

1. Choose a component to be tested;
2. Define the scenario that Will execute the test;
3. Decompose the features as a list;
4. Make a list of the effects for each cause that will occur on the system according to its environment. The effects can be identified according to is environment to make the vulnerabilities mapping easier.
5. Make a list of the most likely vulnerabilities.
6. Identify within the flow where the vulnerabilities may occur;
7. Write a table and verify the valid tests to be written.

Cause	Effect	Environment	Vulnerability	Impact
C1		Interface	V1	High
C2	E1	Internet	V2	Low

Table 1 – Resume of the cause and effect graph

After all the table is filled, we can select the most important test cases according to its impacts on the system.

5.3 Test Project: specifying test cases

After mapping the vulnerabilities on the scenarios, a selection of test case must be realized. All scenarios and vulnerabilities should be tested, however, if there is not enough time, the selection of test case should be done based on the level of the impact and the probability of occurrence of failure.

The writing of the test case follows a template defined on [ISTQB] where one should consider the following points: identify test case, goals, pre-conditions to the test execution, entrance data, step-by-step execution and what the expected outcomes for each test.

On the other hand, for the security test case writing, other points should be considered, such as:

1. On the goals description, it should be define which attack will be executed;
2. Insert the item of vulnerability that composes the attack;
3. Determine where the security may be vulnerable;
4. Which proposed environment should be described on the pre-condition?
5. Which tools should be used on the attack?
6. Which are the scripts and databases should be used on the attack?

The generic security test case template including those fields above should be written as following:

<<Test case ID>> – <<Description of the attack>>	
Severity: High / Medium / Low	**Execution Type:** Manual / Semi- Automatic /Automatic.
Requirement: Requirement ID	**Vulnerability:** name of the Vulnerability to be tested
Pre-Conditions: the state the system should be to execute the tests.	
Entrance Data: the necessary data to the test	
Steps: Which are the steps to execute the test?	**Expected Outcome:** Which are the expected outcome for each steps?
Tools: Which tools Will be used?	**Script:** What script should be used?

Table 2 – Template for security test cases

In order to make the test case reusable, we propose that it is written at a high level way and with its respective goals described as broad as possible. Thus, it is much easier to find reusable test cases for requirements that have the same vulnerability or even reuse test cases to other systems to be tested. The script tests will be written in a parameterized way to make its adaption to other system more simple.

5.4 Automating security tests

After defining which the test cases are, we need to make an analysis of the eligible to be automated [GOUVEIA 2006]. To make this decision, we need to take into considerations some points by answering the checklist below:

1. How many times the test should be executed?
2. Is there a possibility that part of this test be used on other tests?
3. How long is the test implementation?
4. What is the importance of the test case?
5. How many resources are needed to execute the test or which equipment will be used on the execution?
6. How difficult is the manual execution of the test?
7. How reliable is the automated test tool?
8. How portable is the test?
9. How long the test execution takes?

Taking into consideration the questions above, we are going to add some criteria for the test cases automation that will be used on this work:

1. Test cases that can be executed though scripts;
2. Test cases that can be parameterized;
3. Test cases that can be reusable;

The main idea is automating test cases that are portable, language independent, parameterized and reusable. Consequently, we are able to create a reusable script library for automating security test cases.

5.5 Automation Tool

According to the criteria described above, an automation tool will be chosen. As an example, we can cite the OpenScript tool from Oracle. This is an automated test case tool for web application that generates parameterized scripts besides allowing the use of previously generated data file. Besides doing load and stress tests, the tools has the feature of test scheduling and test debug, which allows the tester to see step-by-step where it fails.

In addition to that, the test can be created using the Play/Record functionality and then be edited to meet the needs for security tests. On the next section, we are going to report and analyze security tests.

5.6 Reporting the results and test analysis

The fails report is an extremely important phase for the test software process, because in this moment it is identified systems failures. Once we have the errors reported, we are able to make an analysis of system quality, which is one the main factors do system release delays.

The report of the failures must be efficient, and giving as much as possible information to allow the developer to reproduce the reported failure as well as understand fraction of the code to be corrected. Before reporting an error, it is suggested to follow the instructions below [ASH 2003]:

- Verify if it is really a system failure.
- Verify if it is not failure already known;
- Determine a reliable set of steps to reproduce the failure;
- Document any additional information that will help on the failure reproduction;

After gathering all necessary information to report a security error, some issues should be also reported as shown on the next section.

5.6.1 Template

To report a security error, it is important to fill the following fields [ASH 2003]:

Field	Description
Reporter:	Who reported the error?
Severity:	Which is the error severity?
Description	A brief description of the error.
Environment:	Which environment was used on the test? Example: Browser IE 7.0.5730.13
Build:	Which system baseline was tested?
Test Case	What test case found the error?
Pre-Condition:	Which is the pré-condtiions to run the test? Example: it is necessary to be logged in as XY user
Steps:	What steps were executed to achieve the error?
Expected Results:	What was the expected results?
Actual Result:	What the actual result?
Tool:	What tool were used?
Impact:	What is the error impact?
Mitigation action or solution	What the mitigation action or solution proposed?
Additional Information:	Every information that can help the developer to understand and reproduce the error.

Table 3 – Template for error report

For the error report, is not necessary a special tool to register security errors. One can use its own bug tracking tool to report errors if it contains the mentioned fields described above.

5.7 Validating requirements

After the executions of the tests and reports of its results, it is necessary that one validate if the requirements were tested and if these tests had the adequately coverage. For that reason, it is important to execute a deep coverage analysis.

In order to execute this task, one should verify if test cases were written for all scenarios specified. We can only consider the validation of the requirement was performed correctly if the tests for a given scenario were executed at least once and its result returned a positive outcome.

6. Results

The tasks described by this article were used on the development of some IT projects on C.E.S.A.R (Center of Studies and Advanced Systems of Recife) and UNIMIX. These IT companies are needing to realize a detailed analysis of security issues in some projects with the purpose of making sure that system that are considered critical be tested and validated. As a consequence, this ensures that everything agreed on the contract is respected by the service provider and cannot be questioned by the client.

Due to contracts issues, we are not allowed to give any further information about the context in question. However, some points must be cited, such as:

a) The process of writing requirements has been validated in three projects with companies that act on these sectors: telecommunications and informatics. In these cases, the objective is only write the requirements document and the proposed methodology was employed. As a result, customers noticed an improvement in the problems understanding in early stages of the project.

b) During the risk analysis, the suggested changes in the templates for requirements elicitation indicated a greater understanding of the problems, possible solutions for them and mitigating strategies.

c) Preparation of a repository of reusable security requirements and their test cases based on the recommendations of tests developed by OWASP. This was the case with the requirements of P2P FINEP project, which aims to deploy solutions in environments peer-to-peer. Their requirements and test cases were used for decision making in relation to which requirements and test cases as well as risk analysis for the management solution descktop dreams (HTTP: / / www.dreamsweb.com. br)

d) We have a case study in the development of corporative site of a technology company of Pernambuco. This scenario was run throughout the full proposed cycle in this paper. It was observed that: a) there was significant improvement of the safety requirements of the portal, b) in the testing phase were found about 11 flaws in the site that did not meet the requirements, some of them quite serious, c) Another project in onset may benefit from the basic set of requirements, d) Part of the scripts could also be reused. Unfortunately for security reasons the company was not authorized to divulge more details of the results, as problems are identified security.

e) Observers that the methodology described here is used with extreme efficiency and trends in the proposals brought to the development of systems that must function in an environment of cloud computing. This is because in this environment issues of SaaS, PaaS and IaaS [Cloud] introduce the characteristics of infrastructure as something bringing programmable horizontal scalability for applications. It is undeniable that as we have the scalability of an application being made across the board problems and new security risks arise. These problems not previously considered relevant. Mainly on issues related to security. [SUN] [Cloud]. However, the proposed solutions have a way to specify and reuse them efficiently because the strategies do not vary much scalability.
The main result of this work, we observed an improved understanding of the technical requirements and their implementation by software security engineers and the ability to produce more accurate tests and that met the needs of customers. Thus reducing the need for correction of deficiencies identified in the test phase, which is one of the main mistakes made in building secure software [Meier 2006].

Also as a result it will have a better quality software, on the point of view of ensuring the functionality specified by carrying out a process of validation and elaboration.

Another important result presented in this paper is to provide project managers the ability to quantify risk in relation to the implementation or not a particular requirement before starting the coding phase.

As a consequence of this work, we observed a satisfactory improvement on the comprehension of technical needs and its implementation by software engineers besides the ability to produce test more precise that meet clients need. Consequently, we were able to develop a software with better quality from the point of view of the functionality assurance through the performance of a validation process more elaborated.

7. Conclusion

The software quality cannot be measured only by the assurance of the execution of a process, but by the results of its execution and necessary validations. Within this context, this paper aimed to define tasks, recommendations and process that should be introduced on the cycle of software development with the purpose of guiding the test and validation phase to produce more elaborated and precise results from the point of view of security issues.

The process proposed by this paper is being introduced on the software cycle development s at C.E.S.A.R as specific security needs are required.

The adoption of this process allowed making a more critical analysis of the new features introduction on new projects as well as the test team comprehension at executing these tasks thus improving the software quality observed by the clients.

As a final contribution, we were able to reuse security requirements and its test cases in other projects.

8. Bibliography

Schumacher, Markus (2006), Security Patterns Integrating Security and System Engineering, Willey.

Yoder, Joseph and Barcalow, Jeffrey (1997), Architectural Patterns for Enabling Application Security.

Khomh, Foutse and Guéhéneuc, Yann-Gaël (2008), Do Design Pattern Impact Software Quality Positively.

Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides(1994). Design Patterns – Elements of Reusable Object-Oriented Software. Addison - Wesley, 1st edition.

Longstaff, T. Security of the Internet.

McDaniel, G. Ed. IBM Dictionary of Computing (1994).New York, NY: McGraw-Hill, Inc.

Firesmith, Donald (2003), Engineering Security Requirements, in Journal of Object Technology, vol. 2, no. 1, January-February 2003, pages 53-68. http://www.jot.fm/issues/issue_2003_01/column6

Firesmith, Donald (2004), Analyzing and Specifying Reusable Security Requirements.

Gong,li (1997), Java Security: Present and Near Future.

Haley, Charles, Laney, C. Robin, Nuseibeh, Bashar, Deriving Security Requirements from Crosscutting Threat Descriptions.

Moffett, J. D., & Nuseibeh, B. A (2003) Framework for Security Requirements Engineering, Department of Computer Science, YCS368. University of York, UK.

ISO/IEC (1999). Information Technology - Security Techniques - Evaluation Criteria for IT Security - Part 1: Introduction and General Model, 15408-1. Geneva Switzerland: ISO/IEC, 1 Dec 1999.

Canning S, Rich. Dwivedi, Himanshu. Lackey, Zane. Hacking Exposed™ WEB 2.0: Web 2.0 Security Secrets And Solutions. New York: Mc Graw Hill, 2008. - DOI: 10.1036/0071494618

Lytras, Miltiadis D. Damiani, Ernesto. Pablos, Patricia O. WEB 2.0 The Business Model. New York: Springer, 2009. ISBN-13: 978-0-387-85894-4

Pressman, Roger S. Software Engineering. New York: Mc Graw Hill, 2006. – ISBN: 85-86804-57-6

Howard, michael; lipner, steve . The security development lifecycle: sdl: a process for developing demonstrably more secure software. Redmond, washington: microsoft press,2006. Isbn -10: 0-7356-2214-0

Withall, Stephen. Software requirements patterns. Washington: Microsoft Press, 2007. Isbn-13: 978-0-7356-2398-9

Whittaker, James a.; Thompson, Herbert h. How to break a software security. Boston: Addison Wesley, 2003. Isbn: 0-321-19433-0.

Mcgraw, Gary. Software security: building security in. New York: Addison Wesley, 2006. Isbn: 0-321-35670-5

Application Security Trends Report, 2007, desenvolvido por cenzic security enterprise applications. Avaliable on : http://www.cenzic.com/index.php?id=resources_reg-not-required_trends, last access: 10/01/2009.

Nist, 2008, technical guide to information security testing avaliable on : http://csrc.nist.gov/publications/nistpubs/800-115/sp800-115.pdf last access: 23/11/2008

Mitre – Applaying system engineering and advance technology to critical national problems. Avaliable at: http://www.mitre.org/. Last access: 04/03/2009

Sans - 20 critical security controls - version 2.0, 2009. Avaliable on : http://www.sans.org/cag/guidelines.php . last access em: 29/04/2009

Us-cert - technical cyber security alerts, 2009. Disponível em: http://www.us-cert.gov/cas/techalerts/ . Last access: 29/04/2009

Osvdb - the open source vulnerability database, 2009. Avaliable on : http://osvdb.org/ . Last access: 29/04/2009

Security Tracker, 2009. Disponível em: http://securitytracker.com/ . Last access: 29/04/2009

Chandra, Pravir. Clasp — comprehensive, lightweight application security process. New york: owasp, 2006. Disponível em: http://www.owasp.org/index.php/document_security-relevant_requirements . Acesso em: 12/11/2008

Community-owned model, 2003. Sse-cmm - systems security engineering capability maturity model®. Avaliable at: http://www.sse-cmm.org/docs/ssecmmv3final.pdf, last access: 15/10/2008.

Microsoft , 2009, microsoft security development lifecycle (sdl). Disponível em: http://www.microsoft.com/downloads/details.aspx?familyid=2412c443-27f6-4aac-9883-f55ba5b01814&displaylang=en, last access: 14/03/2009

Microsoft , 2005, security engineering explained - patterns & practices. Disponível em: http://www.microsoft.com/downloads/details.aspx?familyid=75039f39-b33a-4bbd-b041-cf25f7473a0b&displaylang=en, last access: 23/06/2008

Harris, Shon. Cissp – certified information systems security professional. New york: mc graw hill – osborne, 2008. Isbn: 978-0-07149787-9

Talukder, Asoke k.; chaitanya, manish. Architecting secure software systems. Auerbach publications, 2008.isbn-13: 978-1-4200-8784-0

Richard Bender, 2003, requirements based testing process overview. Disponível em: http://benderrbt.com/bender-requirements%20based%20testing%20process%20overview.pdf, last access: 13/10/2008.

Owasp, 2008, owasp testing guide 2008 v3.0. Disponível em:
http://www.owasp.org/index.php/category:owasp_testing_project , last access: 06/05/2009

Sindre, Guttorn; Opdahi, Andreas l. (2002) " eliciting security requirements with misuse cases" – springer- verlag london limited 2004

Chung, Lawrence, Nixon, Brian a., yu, Eric, Mylopoulos, John, 2000. Non-functional requirements in software engineering. Isbn: 0-7923-8666-3

Pezze, Mauro, Young, Michal. Software testing and analysis – process, principles, and techniques. New york: john wiley & sons, 2008. Isbn:13—978-0-471-45593-6

SUN, Introduction to Cloud Computing architecture White Paper 1st Edition, June 2009

Cloud Computing Use Case Discussion Group Version ; Cloud Computing Use Cases A white paper produced by the 2.0 30 October 2009

PMBOK, PROJECT MANAGMENT INSTITUTE (PMI) STANDARDS COMMITTEE A guide to the Project Management Body of Knowledge (PMBOK) Third edition,2008.

C.B. Haley, R.C. Laney, and B. Nuseibeh, "Deriving security requirements from crosscutting threat descriptions," AOSD '04: Proceedings of the 3rd international conference on Aspect-oriented software development, New York, NY, USA: ACM Press, 2004, pp. 112-121.

J. Yoder and J. Barcalow, "Architectural patterns for enabling application security," Urbana, vol. 51, p. 61801.

D. Firesmith, "Engineering security requirements," Journal of Object Technology, vol. 2, 2003, p. 53–68.

D. Firesmith, "Analyzing and Specifying Reusable Security Requirements," Eleventh International IEEE Conference on Requirements Engineering (RE'2003) Requirements for High-Availability Systems (RHAS'03) Workshop, Citeseer, .

C.B. Haley, J.D. Moffett, R. Laney, and B. Nuseibeh, "A framework for security requirements engineering," SESS '06: Proceedings of the 2006 international workshop on Software engineering for secure systems, New York, NY, USA: ACM Press, 2006, pp. 35-42.

Information Technology - Security Techniques - Evaluation Criteria for IT Security, Geneva Switzerland: ISO/IEC Information Technology Task Force (ITTF).

Sommerville I., Software Engineering: Addison- Wesley, 2001.

Meier J. Web application security engineering. IEEE Security & Privacy Magazine. 2006;4(4):16-24.Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1667998

Information Technology - Security Techniques - Evaluation Criteria for IT Security, Geneva Switzerland: ISO/IEC Information Technology Task Force (ITTF)

Neofelis, High-Interaction Honeypot Framework for Mac OS X

João M. Franco and Francisco N. Rente

Universidade de Coimbra - DEI, Coimbra, 3030-290, Portugal
{jmfranco,frente}@dei.uc.pt
http://www.dei.uc.pt

Abstract

A honeypot is a set of computational resources, designed to be swept, attacked and compromised. With a constant monitoring, detailedly record the attacker activities creating means to further understanding of the used approaches and tools. The value obtained from this computational resource is a measure calculated between the captured information and the future use of this data. Neofelis is a framework for high-interaction honeypots on Mac OS X operating system, that allows the system administrator to create a high-interaction honeypot feasible to several different scenarios.
This paper discusses Neofelis design, implementation and pointing out how the framework helps in different areas of the information security, *e.g.* detecting zero-day exploits and capturing informations about the attacks.

Keywords: High-Interaction Honeypot, Framework, Syscall Hooking, Mac OS X, Information Security

1 – Introduction

Information Security is reaching its highest peak since its first appearance, both because of the increasing interest from IT market and the arising criminal financial exploitation.
Nowadays, despite decades of research and experience, it is not easy to deploy secure systems that are entirely foolproof, unknown vulnerability, the so called *zero-day* vulnerabilities, appear more frequently every day. One way to obtain early informations about these new vulnerabilities, is to deploy a computational resource *in the wild*[1] to be attacked and compromised, with the aim of obtaining enough information about the exploited resources and techniques used by the intruder, for a later analysis.
This computational resource is called honeypot and as a conceptual idea it can be deployed in any operating system and/or service.
The main goal of this computational resource is to stay stealth when under attack, ensuring that the intruders do not realize that their actions are being monitored. One of the main goals of honeypots is, recording information regarding to attacker activities, such as keystrokes, file system usage (*e.g.* file uploads or any other kind of

1 - Expression used to describe system deployments in a non-controlled and tend to be non-secure environment, such as the Internet.

modifications), network traffic and used OS resources in a persistent way, represents the creation of the necessary means to a future forensic analysis.
Neofelis is a framework of a high-interaction honeypot for the Mac OS X operating system, which allows the system administrator to deploy distinct and diverse rogue scenarios, with the purpose of being exploited, compromised and later understood. For monitoring and registering the attacker activities were implemented kernel-based extensions which hook syscalls[2] to obtain the required information, such as executed shell commands, network traffic and all file system activity. Furthermore, the referred extensions allow to gather information from executed shell commands regarding its source (from Ring 1 to Ring 3), including crypt flows such as SSH session or by a web shell script over TLS.
In order to dissimulate monitoring activities were used other techniques, such as port-knocking[3] to maintenance services and covert-channel to dissimulate the communications between Neofelis machines. In addition, was also created other kernel extension that allows the hiding of files, directories and processes used by Neofelis itself (to monitor and gather information about the intruder).
Nowadays, Content Management Systems (CMS) are widely used to deploy personal, enterprise or organization web-sites, so its security is vital to prevent unauthorized accesses, otherwise could lead to disclosure of private information or compromise completely the web site integrity. In this study was deployed a HTTP site based on Joomla!, one of the major vendors of CMSes, aiming to test the implemented solution against attack vectors done to the system through web accesses.

2 – Related Work

Literature reveals two different types of honeypots[1][2]: high-interaction defined as a real system with a real operating system and services available to the attacker, who can exploit the existent vulnerabilities and obtain high privileges (until Ring 0) of the machine. The other type is the low-interaction honeypot which consists in a simulation of services with restricted number of available functions, and where the intruder may only obtain low privileges in the machine (Ring 3).
The low-interaction honeypots were specially idealized to capture automated attacks normally made by software that explores known vulnerabilities, such as malware kind known as worms[3]. However, high-interaction honeypots may capture zero-day exploits[4] since the available services and operating system are real and not restricted as in low-interaction.
The different types of honeypots are used according its applicability. The low-interaction honeypots serve the purpose of entities wishing to increase security level to a network, since it is able of warning the system administrator about some security threats (*e.g.* worms spreading in the network), while the high-interaction honeypots are mostly used in research areas to gather information about new attacks, zero-day exploits or detect vulnerabilities not yet understood.
In this project authors decided not to implement a low-interaction honeypot, since it only allows the capture of information about known vulnerabilities and typically are

2 - Technique used to modify the normal execution of a system call, allowing the adulteration of the defined flow.
3 - Technique used to covert a specific service. Using a stealth engine, the service appear offline until a specific sequence of information (*e.g.* network packets) are received.
4 -Software designed to explore a certain vulnerability.

restricted to malware interactions with the deployed machine. Since, the main goal of the Neofelis is to create means to capture information about how the attacks are performed and how unknown vulnerabilities are exploited, it became evident the implementation of a high-interaction honeypot.
Among the available tools to deploy a high-interaction honeypot, Argos[4] and HoneypotX[14] are the most closer to the Neofelis architecture. Firstly, Argos only permits to obtain information about how the attacker exploited a service, through analyzing the binary code that crashed the system and tracing the system memory. This honeypot does not let recording information regarding to performed activities during the attack, such as commands and tools executed by the intruder. Thus, it became necessary to implement a framework for a high-interaction honeypot capable of being totally configurable in terms of possible scenarios, be robust, scalable and at the same time ensuring the integrity of the captured data.
Concerning to HoneypotX, the project is not currently being developed and was built to an older version of .Mac OS X, the 10.1. Moreover, this is a low-interaction honeypot, since it deploys fake network services, such as FTP, SMTP or POP, not allowing any interaction with the service, besides this it checks for remote connections to the system, recording attackers Internet Protocol (IP) address and keeps the connection alive. On the other hand, Neofelis has a real operating system with real services and records not only the information regarding to the IP address, but all the activities performed in the system, such as commands or operations on files.
Mac OS X is the operating system which had the highest grow, in terms of number OS users, in the past years. Which basically means, that in a short term the creation of new attack techniques for this platform will also grow exponentially, since as more users exist, more possible victims the attackers have. This was the main premise behind the choice of using Mac OS X as the base platform for Neofelis. Authors believe that in a short term huge amounts of new threats to Mac OS X will arise, and tools such as Neofelis will have a great importance in the understating and classification of those threats.

3 – Design and Implementation

In this section is presented how the framework was implemented, what techniques were used to dissimulate the performed monitoring functions, as well as how the information captured will be stored and managed in a later forensics analysis.

3.1 System Architecture

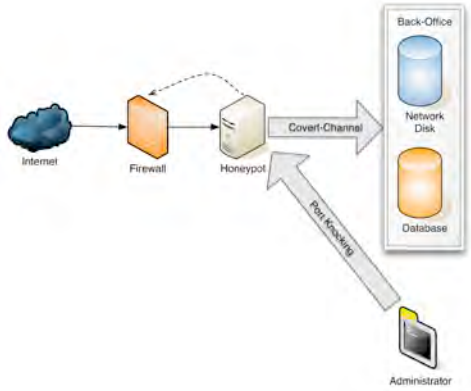
The Figure 1 represents the general architecture of Neofelis, namely the three main components involved in the Neofelis: Firewall, Honeypot and the Back-Office.
The firewall is used to filter the accesses made from the honeypot to the outside, not allowing any communication through ports that aren't associated or related to those used by the available services. In this way it is possible to prevent communications, *e.g.* through Internet Relay Chat (IRC) (normally used to botnet[5] control) or the execution of Denial Of Service (DOS) attacks from the honeypot to external systems.
The back-office will provide support for the honeypot administration process, storing in a persistent way all captured data and publishing results through a interface only

5 - Compromised network, built over a set of internet connected machines which were infected by a malware.

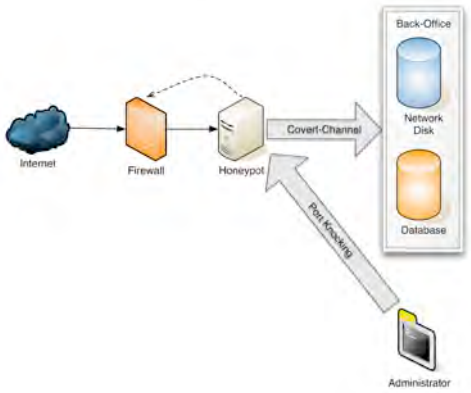
accessible from the management network, during the development and case-study, the CERT-IPN network.
The stored data will be held in a DBMS[6] and a synchronization software. The database will organize the information regarding the executed commands, as well as the network packets exchanged between the outside network and the honeypot. The synchronization software will transfer the persistent information from the honeypot to the back-office disk, recording all modified, created, and renamed files.
The communication between the back-office and the honeypot will be imperceptible to the attacker due to the fact that it was used a covert-channel technique to prevent the attacker from easily detecting and/or subverting the data exchanged in the network during the data collection process.
The maintenance activities will be performed through a tuned SSH Server that runs with a specific configuration that increases stealthiness (*e.g.* different port from the default, do not record any information...) and it is also transported over the covert channel. To increase the covert of the honeypot itself, this SSH service runs with a port-knocking technique that only permits initialize the authentication to the user which send the correct sequence of packets.

3.2 Implementation

The Figure 2 represents the Neofelis internal architecture, as well as the reference to the component responsible to capture and store the activities of the attacker.



6 - Database Management



System

Figure 1: General Architecture Diagram

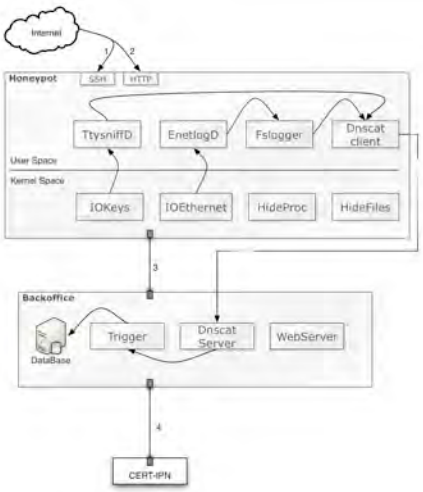


Figure 2: Internal Architecture Diagram

The presented framework permits to create honeypot that can be applied to any network service. However in this paper, authors used a configuration with two different scenarios: a brute-force[7] attack against a SSH server (number 1) and a HTTP server exploitation (number 2).
Numbers 3 and 4 represent, respectively, the physical connection between the machines and the accessible network to perform maintenance activities.
As shown in Figure 2, the honeypot has a separation between the modules that run in user and kernel space. In next sections will be discussed how the information is captured, the monitor activities are dissimulated and the information is stored in a persistent way.

3.2.1 Information Capture.

This process is created through the implementation of kernel extensions that hook syscalls to obtain the intended information, such as executed commands, commands passed as arguments and network packets exchanged between the Internet and the honeypot.
The modules *IOKeys* and *TtysniffD* are responsible for capturing all the executed commands locally or from a remote session (independent of the use of TTY). For example, it can capture as well commands executed from a web shell and those passed by arguments in an opening SSH session. Besides the commands captured, it can record the SSH session information, like the source IP, the used ports, the user that logged in the system and the identification of the opened process. The *IOKeys* is a kernel extension that makes possible to capture information regarding the pressed keys (keystrokes) by hooking the *sysent()* and the *read()* syscalls. This implementation is presented in the Listing 1.1, described by Landon Fuller in [5] and summarized by the Figure 3. Furthermore, *IOKeys* filters the required information and sends it to *TtysniffD* that runs in user space and sends the captured data to the back-office.

7 - Systematically check all possible logins until find the correct one.

```
// Hooking Syscalls
sysent = find_sysent();
// Stores the original pointer of read syscall
org_read = sysent[SYS_read].sy_call;
// Redirect the read syscall to our own function
sysent[SYS_read].sy_call = new_read;
```

Listing 1.1 - Hook of the Read Syscall

```
extern kmod_info_t *kmod;
static void hide_kernel(){
kmod_info_t *k;
k=kmod;
kmod = k->next;
}
```

Listing 1.2 - Method used to hide kernel extensions

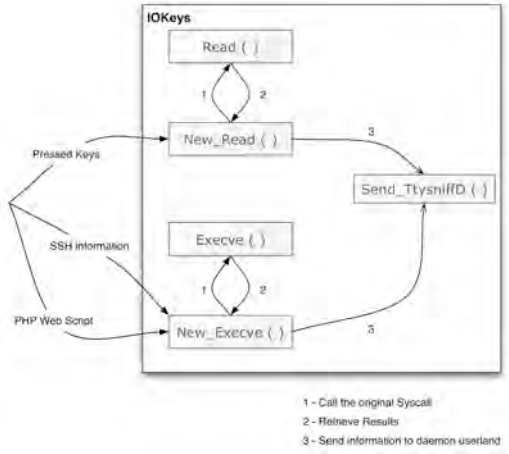


Figure 3: Intercepted syscalls in *IOKeys*

IOEthernet is responsible of capturing network packets, for both incoming or outgoing flow, and sending them to the process running in the user space, the *EnetlogD*. This process, after receiving the network information, stores it in the PCAP format and the *Fslogger*[6] is notified, sending the file to the back-office.
Fslogger is a process that runs in the user space that will be notified by special events in files or folders, such as creation, modification and rename. The *Fslogger* uses the same event notification subscription of the *Spotlight* application[7]. In each notification, it sends to Trigger the name of the process that modified the file or folder and its path. After that, the *Trigger* application synchronizes the file from the honeypot to the back-office.

3.2.2 Dissimulation of Monitoring Activities.

HideProc and *HideFiles* are kernel extensions responsible for dissimulating the system activities, such as hiding processes, files and folders used to store the captured information about the activities of the attacker. These two kernel extensions hook the system calls *getdirentries*() responsible for list the content of a directory (typically used by common shell such as sh or bash), *getdirentries64*() which is a newer version of the previous syscall, *getdirentriesattr*() used for listing files and directories by Mac OS X tools and, at last, the *__sysctl*() syscall used to obtain processes information. This whole procedure is explained by Wowie and Ghalen in [8].
The kernel extensions need to be hide from attackers to dissimulate their activities.

The code provided in the Listing 1.2 was added to the source code of each implemented kernel extension. Its goal is to remove the current extension from the extensions linked list. Thus, when the attacker runs the tool *kextstat* (prints the loaded kernel extensions) Neofelis extensions are not listed.

3.2.3 Storing Captured Information.

Dnscat Client and Server[9] are the processes responsible for the communication between the honeypot and the back-office, by supporting the network covert-channel technique. *Dnscat* Client is responsible for encrypting the information and sends it as DNS traffic to the back-office. *Dnscat* Server receives the information sent by the client and redirects it to the Trigger process.
Trigger is running in the back-office with the goal of receiving the information about the activity of the honeypot and storing it in the database or, in situations related with file system modifications, synchronizing the files which suffered events from the honeypot to the back-office disk.

4 – Results / Evaluation

The results presented in this section were obtained during the period between 25 of June and 11 of July of the present year, 2010. Neofelis was tested against two different scenarios, a brute-force of a SSH server and exploitation of HTTP Server. In both of the scenarios were created some intended vulnerabilities to permit the intruders to attack and compromise the system.

In the HTTP server exploitation scenario was created a front-end site using the Content Management System Joomla! , which contained the vulnerability CVE-2008-3681, that when exploited provides access with administrator permissions to the application. One of the possible attack vectors was the upload of a web shell that allowed the execution of commands, as the apache user, and may run an exploit that grants more privileges to the attacker.
During the period that Neofelis was exposed to Internet, were detected intrusions from Hungary, Belarus, Portugal, Latvia and South Korea. The majority were attempts to access to common sites of databases configurations (e.g. phpmyadmin). However, Neofelis detected two attacks from Portugal and Belarus that took advantage of existent vulnerabilities and culminated in sending files and executing commands on the honeypot, see Table **4**.

In the implementation of the Brute-Force attack against SSH Server scenario was created an SSH server that listened at the default port 22 and two users with weak credentials.

- User - *test* Password - *test*
- User - *admin* Password - *admin*

During the deployment of Neofelis were registered 16 attacks from different provenance countries that opened SSH sessions, executed commands and downloaded files, the results are presented in the Table **4**.

Provenance	Scenario	Activities Performed
Portugal	HTTP	In this attack were uploaded three files, all of them were the same Web Shell, but with different extensions and then abandoned the honeypot. To understand why the attacker had left the machine, the authors tested the transferred Web Shell in other machine and the result was that Web

		Shell needed some other PHP libraries that were not available on the installed Web Server.
Belarus	HTTP	The attacker modified some of the Joomla! configurations with the goal to allow transferring files with the PHP extension. After that, uploaded a file encoded in base 64 easily decrypted by the authors, who realized that it was a reverse Shell. In this way, the attacker had access to the remote system from his own computer and execute commands in the honeypot. Initially the attacker obtained informations about directories listing, system, users and the executing processes. Then he tried to obtain root access, verified the file of the TTY that was using, the users that were logged in and finally moved the file of the Web Shell to the parent folder and renamed it.
France	SSH	The intruder logged in the system through the user test. The first actions were to obtain informations about which users were logged in, then tried to obtain informations about the operating system, through the directory /proc and the tool uname. After realizing which operating system was being used, moved directory and downloaded a compressed file with the name of DarwinBoot from a Russian site. Unfortunately the file was corrupted and so, the attacker could not decompress it.
Spain	SSH	The user admin accessed to the honeypot and verified which users were logged in the system, visualized what operating system was running and changed the password of this user. The last action taken by the intruder was to keep the user admin only for him and is expected in the future the attacker uses this user to compromise the system.

From the results presented in the Table **4** is possible to realize that Neofelis can capture information about the modified and created files, the commands executed and the activities of the attacker. The activities taken against the honeypot did not had a malicious nature, however was demonstrated that Neofelis is a system capable of monitor and capture information of future attacks.
The obtained results reveal the tools and tactics used by the attackers to subvert the system, fulfilling the outlined requirements.

5 – Conclusion and further work

Neofelis is, at the time this paper was written, the first high-interaction honeypot implemented entirely and from scratch for Mac OS X. As said, the main purpose was to create a framework which helps to understand the security incidents of the operating system and associated network services. This kind of architecture enables a panoply of possible study scenarios, creating the necessary requirements to capture and properly analyze exploits and vulnerabilities unknown or not yet well understood. The information captured by Neofelis will be used in a posterior forensic analysis to reveal possible threats to the security of the system and the privacy of the user. This analysis will be useful to inform companies related to information security, operating

system and deployed network tools to create and apply security patches.
It is important to refer that the attacker's capability to detect the presence of Neofelis is very low. However, in order to do it, the attacker will have to compromise the kernel and make a further syscall scan.
In order to enrich the future development, Neofelis was idealized to be modular and scalable, increasing the monitoring and capturing capabilities without losing the ability of being applied to the specified scenario. The use of different dissimulation techniques, such as covert-channel to make the communication between machines imperceptible to the intruders, hooking syscalls to hide processes, files and loaded kernel extensions and the use of port-knocking to hide the maintenance services, makes Neofelis a high-level of stealthiness.
For further work it is planned to implement a filter of network packets through a rule-based system and IDS integration capabilities.

References

[1] L. Spitzner, *Honeypots: Tracking Hackers*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[2] N. Provos and T. Holz, *Virtual honeypots: from botnet tracking to intrusion detection*. Addison-Wesley Professional, 2007.

[3] P. Baecher, M. Koetter, M. Dornseif, and F. Freiling, "The nepenthes platform: An efficient approach to collect malware," in *In Proceedings of the 9 th International Symposium on Recent Advances in Intrusion Detection (RAID*, pp. 165–184, Springer, 2006.

[4] G. Portokalidis, A. Slowinska, and H. Bos, "Argos: an emulator for fingerprinting zero-day attacks," in *Proc. ACM SIGOPS EUROSYS'2006*, (Leuven, Belgium), April 2006.

[5] L. Fuller, "Fixing ptrace(pt_deny_attach, ...) on mac os x 10.5 leopard," visited in February 2010.

[6] A. Singh, "A file system change logger," *Mac OS X Internals*, visited in May 2005. `http://www.osxbook.com/software/fslogger/`.

[7] Apple, "Spotlight," `http://www.apple.com/macosx/what-is-macosx/spotlight.html`.

[8] W. wowie@hack.se and ghalen@hack.se, "Developing macos x kernel rootkits," in *Phrack*, no. 0x10 in 0x42.

[9] R. Bowes, "Dnscat." `http://www.skullsecurity.org/wiki/index.php/Dnscat`, visited in October 2010.

[10] N. Provos, "A virtual honeypot framework," in *In Proceedings of the 13th USENIX Security Symposium*, pp. 1–14, 2004.

[11] SANS, "The top cyber security risks." `http://www.sans.lfullerorg/top-cyber-security-risks/`, visited in September 2009.

[12] H. N. Security, "Top 15 most common security attacks." `http://www.net-security.org/secworld.php?id=8597`, visited in December 2009.

[13] J. Briffaut, J.-F. Lalande, and C. Toinard, "Security and results of a large-scale high-interaction honeypot," *JCP*, vol. 4, no. 5, pp. 395–404, 2009.

[14] Ugmpt, "HoneypotX", http://www.macupdate.com/app/mac/11462/honeypotx, visited in December 2010.

SecureForms: Hypervisor-level Service to Secure Online Sensitive Information from Spyware and Key-loggers

Mordechai Guri

Department of Computer Science
The Hebrew University of Jerusalem
Jerusalem 91904, Israel

Abstract. For the past decade, the threat from malware such as spyware, Trojan horses and rootkits has continued to mar the internet browsing experience. Credit card numbers, passwords and personal identity information can easily be stolen by spying techniques at different levels of the operating system. The identity theft resulting from spyware-collected information totals hundreds of millions of dollars annually. In this paper, we present the architecture, design and implementation of SecureForms a hypervisor-level service for securing online sensitive information from malware. By keeping the untrusted OS in a virtual machine and using a proxy-based component outside it, sensitive input to web pages and forms is handled by the hypervisor-level service, which is completely separate from the untrusted environment. We show that when using SecureForms, sensitive information can be entered without being monitored by spyware, key-loggers or password stealers.

Keywords: SecureForms, spyware, malware, key-loggers, virtualization, hypervisor, personal identity information, sensitive information

1 Introduction

Over the last ten years, internet services have grown rapidly and today, hundreds of millions of people browse the internet on a daily basis [1]. Social networks, web-mail services and online shopping websites are only a few of the internet services that require passwords, credit-card numbers and personal identity information. This rapid growth has also resulted in a big increase in malware infection and sophistication [2], [3], [4]. A home computer can silently be infected by downloading spyware-bundled software, opening untrusted email attachments or by unknowingly installing malicious browser extensions [5], [6]. Leaks of online sensitive information can be extremely harmful, as was seen in 2008 when approximately 10 million Americans were victims of identity fraud. About 11% of all cases are the result of spyware-based identity theft [7].

Currently, the most common end-user solution is buying commercial anti-spyware software. Although anti-spyware programs are considered to be necessary for today's desktop computers, protecting the system against spyware has one main drawback: it

is only part of the endless battle pitting malware against security vendors [8]. Systems must constantly be updated with the latest spyware definitions to guarantee the most effective protection [7]. However, even a well-maintained system with up-to-date security software is not totally immune to spyware. For example, key-loggers are undetectable by most anti-spyware solutions [3]. Advanced malware such as rootkits are very hard to detect and can even disable anti-virus monitoring capabilities [9].

This paper addresses one of the most harmful aspects of spyware in the internet era that causes much more damage than breaches of surfing habits or internet history logs: the theft of sensitive information. To address these threats, we introduce SecureForms, a virtualization-based service that provides the user with a secure environment to input sensitive information over the internet. The core concept is to treat the untrusted system as a virtual machine (VM) and handle sensitive user input outside it. Through the guaranteed resource isolation between the VM and the hypervisor, operations such as entering credit card numbers, user names and passwords are completely transparent to the VM. By definition, spyware is unable to break the virtualization sandbox, intercept the keystrokes or access memory outside its VM context. Additionally, the components installed outside the VM are considered to be trusted, as they cannot be compromised by malware that runs inside the guest OS.

SecureForms provides two forms of protection:

1. Input Aliasing: This method protects sensitive information filled in by users on web-pages and forms. It works with current websites without altering the server or the web-browser, but changes the way the user enters the sensitive input. Specifically, before entering sensitive information in a web-page, a predetermined alias is keyed in instead of the actual information. A trusted proxy-based component intercepts the HTTP requests, locates the aliased input fields and substitutes the original, sensitive value. If a value for the aliased input fields has not been stored, the HTTP proxy initiates a trusted dialog asking for it.

2. User Interface (UI) dispatching: This method protects a wide range of desktop applications beyond forms and web-page input. Here, a secondary trusted channel is initiated between a client at the hypervisor and the server. The server can use the trusted channel to send forms that need to be secure at the user end. While the UI dispatching method does not change the way the user inputs the information, it requires modifications to the server to support its protocol. For example, UI dispatching can be used for desktop messaging applications to protect login and payment dialogs, or send “off-record” text, voice and even video messages.

The next section explains the design goals and assumptions behind this architecture. Section 3 then presents the SecureForms architecture, which is extended to the two protection methods in sections 4 and 5. Possible attack vectors on SecureForms are countered in Section 6, which is followed by the specifics of the implementation and evaluation of SecureForms in section 7. Section 8 describes related works, Section 9 presents possible future work on the architecture and section 10 concludes the paper.

2 Design Goals and Assumptions

2.1 Goals

SecureForms must fulfill the following criteria:

• **Full information isolation:** The protected information must not pass through the guest OS at any time.

• **User experience preservation:** The web-browsing experience should not be fundamentally affected and must be user friendly.

• **No client modification:** Even though components in the guest OS might ease implementation and provide additional flexibility, in-the-box components could potentially be compromised by malware.

• **No server modification:** SecureForms must provide information protection without any alteration of existing web-servers or web-pages.

2.2 Assumptions

• **Trusted hypervisor:** In this paper, we assume a trustworthy hypervisor model. Malware that runs inside the VM may subvert any component inside the guest OS and access its VM resources. However, it cannot break out of the sandboxed environment by accessing resources outside the VM context or compromising the underlying hypervisor. Note that this assumption is the basis for many other hypervisor-based security researches works [10], [11], [12], [13], [14].

• **Hypervisor networking and interfacing capabilities:** We assume a hypervisor with networking and interfacing capabilities (i.e., a TCP/IP stack and user interface (UI) support). The implementation consists of a Type II hypervisor model, where a VMware server runs on top of a host OS. In this case, the TCP/IP capabilities and UI support are naturally provided by the host OS.

3 The SecureForms Architecture

In this section we describe the underlying architecture and components of SecureForms and the interactions between them.

Virtualization plays a major role in the SecureForms architecture. The system configuration consists of the user OS, running as a guest on top of a Virtual Machine

Monitor (VMM)[1]. The guest OS runs the end-user applications (e.g., internet browser, email client, word processor). The guest OS is considered to be untrusted and might already be infected with spyware and key-loggers. As discussed in Section 2, the hypervisor environment is assumed to be trusted.

SecureForms utilizes virtualization to maintain its security by enforcing the following concepts:

• **Trusted UI:** SecureForms handles user input outside the untrusted OS. Dialogs that ask for sensitive user information are initiated from a component that runs at the hypervisor level. While malware can access all resources inside the VM context, it cannot break through the virtualization sandbox and gain access to system resources outside it. Consequently, sensitive keyboard input and screen output are completely transparent to the guest OS and any malware running on it.

• **Trusted HTTP(S) proxy:** SecureForms uses an HTTP proxy to control the HTTP requests and responses between the web-browser and the web-server. By installing the HTTP proxy outside the VM, it cannot be compromised by malware running on the guest OS. HTTP proxies use the man-in-the-middle technique to decrypt SSL based HTTP sessions. The proxy acts as a Certificate Authority (CA) to the web-browser, and manages its own HTTPS connection to the web-server. Since the certificate generated by the http proxy is self-signed, under normal circumstances a web-browser raises a warning message about the use of non-trusted CA. This message can be circumvented by adding the proxy address to the trusted CA list in the browser.

As part of the SecureForms architecture, the HTTP proxy runs outside the guest OS and monitors it's outgoing and incoming http traffic. Crucially, the user must configure the web browser proxy settings to point to the http proxy at the hypervisor. Adjusting the browser settings is the only step required at the level of the guest OS. Possible attacks on proxy settings are discussed in section 6.

The two SecureForms protection schemes are described in the upcoming sections. Note that the first method, input aliasing, requires no changes to the server, whereas the second method, UI dispatching, requires a small change to the server but does not change the user experience in any way.

4 Input Aliasing

The first method to secure sensitive information in SecureForms is Input Aliasing. Input aliasing allows the user to type in simple aliases such as $WEBMAIL_USERNAME$ and $WEBMAIL_PASSWORD$ (the $ sign was chosen arbitrarily) instead of sensitive information. Thus, sensitive information is

[1] In this paper the terms VMM and Hypervisor are used interchangeably, depending on the context.

never input in the untrusted guest OS. This method does not require any server-side modification, and can be used in any website.

The only difference from ordinary browsing is the user's experience. In classical browsing, sensitive information input fields appear inside the web-browser. The user then fills the required information in and submits the form. In the input aliasing scheme, the sensitive information is masked from the guest OS by aliases. When an HTTP request is submitted, the HTTP proxy prompts the user for the real values of the fields, and in turn replaces the aliased input with the actual value.

Input aliasing works transparently with password managers and automatic form-fillers because the real value substitution is carried out at the hypervisor level, after the form is submitted. Besides secure form filling, input aliasing can be used to conceal sensitive information inside web-based emails. For example, a user can provide his or her own security code by aliasing it as SEC_CODE inside the body of the email.

4.1 HTTP(S) Proxy

After the form is submitted, a hypervisor-level HTTP proxy intercepts its request headers and checks whether it includes aliased input fields by filtering the alias pattern. If there are such fields, a trusted dialog is shown to the user asking for the actual values of the aliased fields (Fig. 1). Optionally, the user can save the value of a specified alias to avoid having to enter it the next time it is used. New aliases can be added manually and removed or modified by using a simple alias manager panel in the hypervisor, which also cannot be compromised.

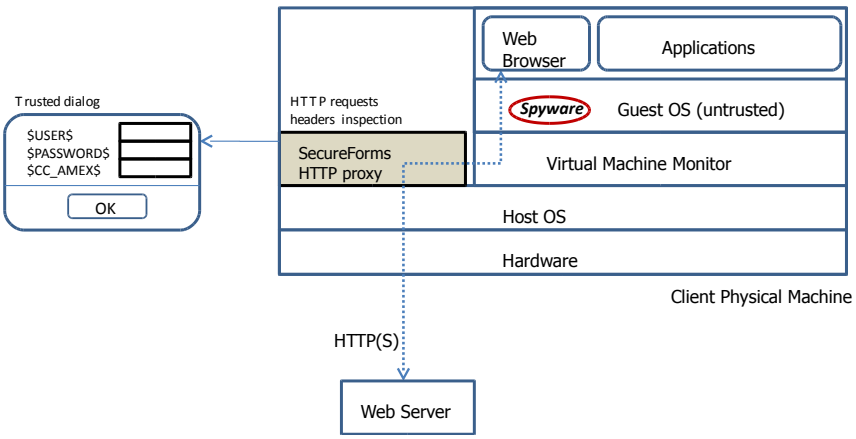


Fig. 1. SecureForms Input Aliasing scheme. The aliased input fields replace the real values by the SecureForms proxy component.

4.2 User Experience

Input aliasing does not affect the browsing experience dramatically. Essentially, it pops up only when submitting forms with aliased inputs. In fact, aliasing may even affect the user experience positively. For example, the aliases $BANK_PWD$ and $AMEX_CC$ are easier to remember and type than a real password or credit card number. It is worth noting that SecureForms also works on public computers (e.g., internet cafés), which are much more vulnerable to personal information theft [15], [16].

5 UI Dispatching

UI dispatching provides a generic solution for securing the user interface from an untrusted OS. Even though this section refers to web forms, UI dispatching can be adapted to many other client/server applications, such as "off-the-record" IM chat messages.

In an ordinary HTTP session, a single connection is used to request web-pages, including those that contain sensitive information. In the UI dispatching scheme, a secondary channel, the secure UI channel, is initiated between the SecureForms client at the hypervisor and the server. Instead of residing in the untrusted guest OS, the sensitive forms are sent using the SecureForms channel.

As illustrated in Fig. 2 the UI dispatching architecture consists of three components: the HTTP proxy, the SecureForms client and the SecureForms server component. While the first two components reside at the hypervisor level, the third component resides at the web server. This is done to identify and separate the two channels according to the demands of the web server. The following subsections describe all three components in detail.

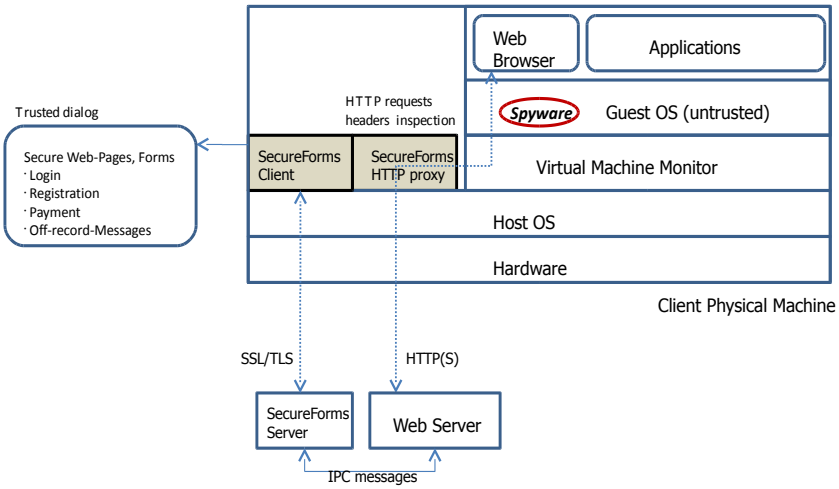


Fig. 2. SecureForms UI dispatching scheme. Two separate channels control the user experience, one for the regular browsing (highlighted in green) and the other for sensitive information.

5.1 HTTP(S) Proxy

The HTTP proxy detects and initiates the secure UI channel between the SecureForms server component and the client. First, the HTTP proxy notifies the web server that the client supports SecureForms, by modifying the User-Agent field in the HTTP requests as follows:

Mozilla/4.0 (compatible; MSIE 7.0b; Windows NT 6.0; SecureForms)

If SecureForms is not supported by the web server, the field is simply ignored. Otherwise, the web server notifies the client to initiate a secure UI channel with the server at a designated IP and port. The notification is done by adding a predefined tag to the HTTP response headers, which is monitored by the HTTP proxy. Note that the secure UI channel is designed so that it can be initiated at any stage of an HTTP session.

5.2 SecureForms Client

The SecureForms Client resides at the hypervisor-level of the client machine, and provides the trusted UI functionality. After initiating the secure UI channel, it waits for incoming information (e.g. HTML) and shows it to the user. It is also responsible for sending the input back to the SecureForms server component. Due to the nature of the information exchanged over the secure UI channel, an SSL/TLS connection is

used between the SecureForms client and server.

In this paper, we represent the UI as a script-less simple HTML file. HTML provides the strength and flexibility of visualization, with the advantage of being widely used. Protecting the SecureForms client against attacks such as remote code execution or browser exploits is crucial to system security; thus, by using static HTML we eliminate any risk of hypervisor-level external code execution.

5.3 SecureForms Server Components

The SecureForms server component manages the secure UI channels to SecureForms clients. When a web-page with sensitive information needs to be sent to the client, the web server application notifies the SecureForms component to dispatch the UI to the client upon validation of its session ID. The SecureForms component then sends the page to the client mapped to that specific session ID, and returns the sensitive user input back to the web server.

In order to avoid initiation of secure UI channels from false clients, SecureForms tracks the list of existing session IDs by receiving updates from the web server. New clients are accepted only when the given session ID is a valid one.

6 Attack Vectors

Due to information isolation, an untrusted guest OS that runs inside a VM does not receive the sensitive information in any way. Thus, malware and key-loggers installed in the guest OS are not able to intercept the information directly. Potential attack vectors on SecureForms either reside outside the VM or involve tricking the user into typing the sensitive information inside the VM. These attack vectors can be categorized into three types: Spoofing, Denial of Service and Phishing/Pharming. To ensure that the reproduction of your illustrations is of a reasonable quality, we advise against the use of shading. The contrast should be as pronounced as possible.

6.1 Dialog Spoofing

Since SecureForms displays a dialog for the sensitive information, malware can spoof this dialog and mislead the user into thinking the sensitive information is within the hypervisor context, while it is actually typed inside the VM. To make sure the dialog box cannot be spoofed, several techniques are available.

First, a service authenticity certificate is shown in every SecureForms dialog. This certificate can be a string, an image or sound which is selected by the user during the SecureForms installation stage. The fact that the certificate is stored outside the VM means that malware cannot acquire it and hence the certificate functions a ‘signature’ which cannot be spoofed.

Second, VMM usually limit the VM screen output to a rectangular region which does not cover the whole physical machine screen. By showing the service dialog

boxes outside the VM bounded rectangle, the user knows that the dialog that appears on the screen has not been spoofed.

Third, most VMM indicate whether the VM is currently active and is receiving keyboard/mouse input or not. For example, a VM windows caption is highlighted when it is being used. This feature can be verified by the end user to guarantee that the VM is not receiving the typed input.

6.2 Denial of Service (DoS)

To operate SecureForms on a guest OS the only change required is the proxy settings. Malware can potentially attempt to delete or disable the settings, thus allowing direct access to the web-site. The SecureForms architecture prevents this by placing a firewall that blocks all HTTP and HTTPS communications at the hypervisor level that enforces the use of the SecureForm proxy. In addition, such DoS attempts are visible to the user since the aliases do not work and the SecureForms dialog is not displayed when sensitive information is keyed in.

6.3 Phishing and Pharming

Last in SecureForms defensive architecture is the actual server that receives the sensitive information. Both phishing and pharming are ways to obtain sensitive information outside the protected architecture. Therefore, the SecureForms dialog shows both the DNS address (if applicable) of the target website and the resolved IP, so that a suspicious website is immediately detectable. No host routing table or browser address-line modifications affect the lower level of communication; thus the possibility of spoofing the target website is eliminated. Note that SecureForms can be loaded with common phishing website black-lists to provide even more protection.

7 Implementation and Evaluation

A prototype of SecureForms was implemented and used to secure login, registration and payment information, with both the input aliasing and UI dispatching schemes.

In order to simulate the SecureForms architecture with all the assumptions in place, we used a Windows XP (SP3) as the host OS and a VMware Server (2.0.2) as the VMM.

For the HTTP proxy implementation, we used the Fiddler [17] web debugging proxy. Fiddler provides support for a jscript.NET environment to capture, analyze and modify HTTP(S) requests and responses. The OnBeforeRequest handler and the .NET interface extension were employed to implement the trusted dialog and the HTTP header manipulation functionality, which are needed by both schemes, as described in Section 3.

To demonstrate UI dispatching, OpenCart [18] an open source, PHP-based shopping cart system was used. The PHP sources were modified to call up specialized SecureForms functions whenever login, registration and payment forms are requested.

The web-server requests the SecureForms component to transmit the forms to the client using a secure UI channel and waits for the user's response. When the user input is received, the web-server verifies it and continues with the usual flow. We used Apache server (2.2.11), PHP (5.3.0) and MySQL (5.1.36) to manage and run the OpenCart website.

For the UI dispatching scheme, we implemented the SecureForms client as a dialog-based application on the host OS that can display HTML and submit forms (fig. 3). A string-based Service Authenticity Certificate is defined when the client is first launched. The SecureForms server component is implemented as a separate application on the HTTP server. The web-server and SecureForms component communicate using Inter Process Communication (IPC) sockets. Both the SecureForms client and server components are implemented in C#.

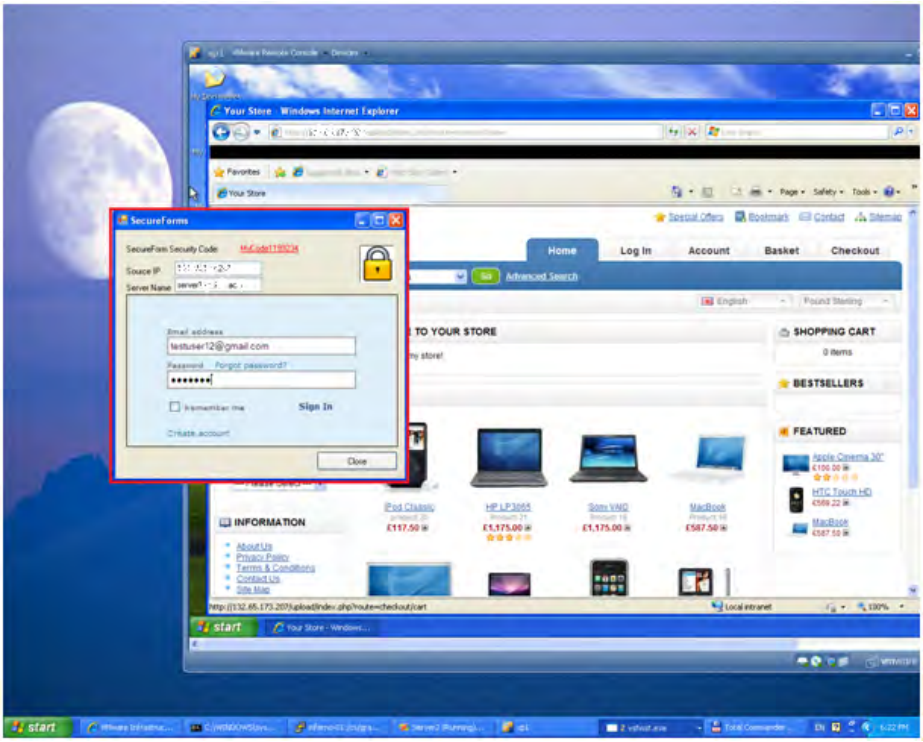


Fig. 3. The UI dispatching scheme secures OpenCart login page. The red square indicates a trusted dialog.

The implementation of SecureForms was tested with the different guest operating systems (Microsoft Windows 7/Vista/XP, Linux) and the three major web browsers (Microsoft Internet Explorer, Mozilla FireFox, Google Chrome). Various key-loggers, screen capture programs and password sniffers were installed on the guest OSes to

demonstrate the untrusted environment.

Then, the UI dispatching scheme was used to secure login, registration and payments pages in the modified OpenCart web-site. The Input aliasing scheme was used to secure a wide range of input fields and to conceal sensitive information in web-based emails.

After the information was submitted, the spyware's log files were examined to find any of the sensitive data that had been entered. As expected, none of the input was captured.

8 Related Works

There are four main approaches to countering sensitive information theft: anti-spyware software, input method alteration, sandboxing, and proxy-based systems. SecureForms can be seen as a combination of the two latter approaches.

Input method alteration is a way to thwart key-loggers. This method changes the way users input information by avoiding actual “keystrokes”. The most common method is an on-screen keyboard [3]. While it theoretically replaces physical keyboard usage, some on-screen keyboards use OS keyboard messages to transmit the virtual keystrokes to the target application. Hence, the virtual keystrokes can still be intercepted by key-loggers.

Another approach to protect the system is sandboxing. By running applications in a virtual sandbox, any change to the host OS is monitored and can be avoided. As a result, malware is unable to install itself to the OS. The two main sandboxing techniques are application-level sandboxing and VM sandboxing. VMware ThinApp[19] and Microsoft App-V [20] are examples of commercial application-level sandboxing systems. VM sandboxing separates applications by running each application in its own VM; examples are NSA’s NetTop [21] and Qubes [22]. A solution that uses VM to provide a secure environment for sensitive user input was presented by Kwan and Durfee [23]. However, their system requires installing a browser extension makes modifications to the server and changes the user experience.

Selected examples in the area of proxy-based systems are KLASSP [24] and URRSA [25]. In this type of solution, a remote proxy server, which is considered safe, resides between the untrusted machine and the server, and is responsible for the actual authentication. Instead of explicitly typing the original password in the untrusted side, various methods are used to authenticate the user in the trusted proxy, which in turn transmits the real password to the server. Examples of methods include one-time-passwords (OTP) [25] and challenges based on a shared secret.

SpyBlock [26], which combines proxy and virtualization techniques to protect user authentication, is the most closely related research to our work. SpyBlock consists of several schemes to protect users from authentication threats such as key-loggers and password sniffers. Although SpyBlock and SecureForms are related, it requires installing an agent in the untrusted OS, and focuses solely on protecting user passwords and authentication. SecureForms, on the other hand, does not require the installation of a component inside the untrusted OS and can protect any user input anywhere as in e-mail body text and off-the-record messages without radically changing the user experience. In addition, SecureForms UI dispatching architecture

allows wide a range of desktop application to use the advantage of trusted dialog at the user end.

9 Discussions and Future Work

To extend the applicability of SecureForms and simplify the installation process, the following improvements would be worthwhile. These improvements are left for future research.

1. The need to modify the proxy server address inside the guest OS can make the installation process cumbersome. Examples include web-browsers that have no proxy settings and the complexity of changing such a setting from the home user's viewpoint. Detection and interception of HTTP/HTTPS packets directly in the hypervisor-level would avoid all of these complications. Such a proxy would use the hypervisor capability to intercept, analyze and manipulate the VM network traffic packets silently without being explicitly pointed by the guest OS.

2. Although UI dispatching provides more flexibility and preserves the user experience (compared to input aliasing), it is not always possible to extend the server-side functionality. This problem can be addressed by transferring the UI page dispatching from the web-server to the HTTP proxy/interceptor, which does not change the server.

10 Conclusions

In this paper, we presented the design and implementation of SecureForms, a hypervisor-level service that protects online sensitive information from spyware and key-loggers. We showed that by handling sensitive input outside the untrusted VM, the information is completely transparent to malware of any type. Based on the SecureForms architecture, we presented two protection methods - Input Aliasing and UI Dispatching. Input aliasing works with current websites without any modification, by altering the way sensitive information is entered. UI dispatching does not change the user experience but requires server-side support. SecureForms was then successfully evaluated against a variety of spyware and key-loggers, which failed to retrieve sensitive information.

References

1. Pew Research Center. Daily internet activities, 2000-2009 http://www.pewinternet.org/Trend-Data/Daily-Internet-Activities-20002009.aspx
2. Symantec. Global internet security threat report: Trends for 2009. http://eval.symantec.com/mktginfo/enterprise/white_papers/

b-whitepaper_internet_security_threat_report_xv_04-2010.en-us.pdf, (2010)
3. S. Sagiroglu and G. Canbek. Keyloggers. Technology and SocietyMagazine, IEEE, 28(3):10 –17 (2009)
4. L. Seltzer. The growth of malware: Up, up, and away. http://blogs.pcmag.com/securitywatch/2009/07/the_growth_of_malware_up_up_an.php (2009)
5. T. Stafford and A. Urbaczewski. Spyware: The ghost in the machine. In AMCIS 2004 Proceedings (2004)
6. W. Ames. Understanding spyware: risk and response. IT Professional, 6(5):25 – 29 (2004)
7. Javelin Strategy and Research. 2009 identity fraud survey report, consumer version. pages 4–7 (2009)
8. M.D. Preda, M. Christodorescu, S. Jha, and S. Debray. A semantics-based approach to malware detection. ACM Trans. Program. Lang. Syst. 30(5):1–54 (2008)
9. SpywareRemove. Trojan rootkits (rootkit.gen) disables antivirus apps and security sites. http://www.spywareremove.com/security/trojan-rootkits-rootkit-gen-disables-antivirus-apps/ (2008)
10. Z. Wang, X. Jiang, W. Cui, and P. Ning. Countering kernel rootkits with lightweight hook protection. In CCS '09: Proceedings of the 16th ACM conference on Computer and communications security, pages 545–554, New York, NY, USA (2009)
11. B.D. Payne, M. Carbone, M. Sharif, and W. Lee. Lares: An architecture for secure active monitoring using virtualization. pages 233 –247 (2008)
12. Z. Wang, X. Jiang, W. Cui, and X. Wang. Countering persistent kernel rootkits through systematic hook discovery. In RAID '08: Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection, pages 21–38, Berlin, Heidelberg (2008)
13. X. Jiang, X. Wang, and D. Xu. Stealthy malware detection and monitoring through vmm-based "out-of-the-box" semantic view reconstruction.ACM Trans. Inf. Syst. Secur., 13(2):1–28 (2010)
14. R. Riley, X. Jiang, and D. Xu. Guest-transparent prevention of kernel rootkits with vmm-based memory shadowing. In RAID '08: Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection, pages 1–20, Berlin, Heidelberg (2008)
15. D. Florencio and C. Herley. Klassp: Entering passwords on a spyware infected machine using a shared-secret proxy. pages 67 –76 (2006)
16. G. Richards. How to improve your security when using a public terminal. http://www.techsupportalert.com/improving-public-terminal-security.htm (2009)
17. Fiddler: Web debugging proxy. http://www.fiddler2.com/fiddler2.
18. Opencart: Open source shopping cart solution. http://www.opencart.com.
19. VMware. Thinapp. http://www.vmware.com/products/thinapp.
20. Microsoft. App-v. http://www.microsoft.com/systemcenter/appv.
21. National Security Agency. Nettop. http://www.nsa.gov/research/tech_transfer/fact_sheets/nettop.shtml (2009)

22. Qubes. http://www.qubes-os.org.
23. P. C. S. Kwan and Glenn Durfee. Practical uses of virtual machines for protection of sensitive user data. In ISPEC'07: Proceedings of the 3rd international conference on Information security practice and experience, pages 145–161, Berlin, Heidelberg (2007)
24. D Florencio and C Herley. Klassp: Entering passwords on a spyware infected machine using a shared-secret proxy. Computer Security Applications Conference, Annual, 0:67–76 (2007)
25. D. Florencio and C. Herley. One-time password access to any server without changing the server. In Tzong-Chen Wu, Chin-Laung Lei, Vincent Rijmen, and Der-Tsai Lee, editors, Information Security, volume 5222 of Lecture Notes in Computer Science, pages 401–420. (2008)
26. C. Jackson, D. Boneh, and J. Mitchell. Transaction generators: root kits for web. In HOTSEC'07: Proceedings of the 2nd USENIX workshop on Hot topics in security, pages 1–4, Berkeley, CA, USA (2007)

Automating Web Applications Security Assessments through Scanners

Nuno Teodoro, Carlos Serrão

ISCTE Lisbon University Institute/SOTA/DCTI, Ed. ISCTE, Av. Forças Armadas, 1649-026 Lisboa, Portugal

nuno.filipe.teodoro@gmail.com, carlos.serrao@iscte.pt

Abstract. Nowadays, some of the most critical services are accessible through specific developed Web applications by millions of users. The pressure put by the management and other stakeholders to reduce the development lifecycle in order to speed up the Web application development and deployment raise security challenges. Web applications security assessment tools are vital in each of the steps of the Web Software Development Life Cycle, including the final stage, before the application deployment. Web scanners are automated tools that can be used to perform black-box vulnerability assessments.
The definition and choice of these web scanners, as well as the comparison between the three major open source Web scanners available today, used in a real case scenario, is the main focus of this paper. These tools are available in large number, making vital the definition of specific decision criteria and results analysis, to select the one that fits better the expected results. The objective of this paper is to provide insight information about some of the best open-source web scanners and methodologies available in order to provide guidelines for a better decision while selecting these tools and how they are used on real case scenarios.

Keywords
Security, web, vulnerabilities, penetration testing, black-box testing, web scanners.

1. Introduction and motivation

Security has become more and more complex. Security can no longer be reduced to the installation of appropriated firewalls or intrusion detection systems on the network. The network level is no longer the main focus of the security and more and more security flaws explore the application layer [1]. Web applications are exposed to the World, accessed by millions of unknown users, which may have bad intentions. Additionally, most services are currently deployed on the World Wide Web (WWW), in particular, critical services like banking transactions, tax payments, health records, details of your personal life, and many more, raising even more these funded security concerns.
The number of attacks documented by some entities help confirm the trend that attackers are targeting more and more the application level. The National Institute of Standards and Technology [NIST] holds a National Vulnerability Database [NVD],

which has more than 40,598 vulnerabilities, identified as of March 13, 2010 [2]. These vulnerabilities include 3590 injection vulnerabilities, 2182 XSS vulnerabilities and 464 authentication vulnerabilities, which are the 3 most common vulnerabilities within web applications, according to OWASP Top 10 2010 [3].

The Web applications security is one of the most difficult issues to deal with because virtually everyone has access to them, but also because it involves a significant number of variables. The Web applications security has a direct relation with the network, Internet, transport and application layers. This is often misunderstood and most security professionals assume that if the lower two or three layers are secure (which in most of the cases are not) then the entire web application environment is also secure. This is often wrong and therefore it is one of the main reasons to prioritize tools and methodologies that can be used to perform application level security assessment. These tools need to be present in every stage of the software development life cycle (SDLC), helping the mitigation of possible errors that can occur in each development phase. Nevertheless, the emphasis of this paper goes to web application scanners, which are mainly used in the end of the SDLC. Also, these tools are helpful not only for the developer but also for the potential customer of a web application product to conduct their own selection assessments based on the product security.
The first important phase is to create the necessary awareness about the need to use such tools. The second major step consists on the selection of the appropriate tools. This is one of the problems that affect the organization development and security teams. Currently the choice is very wide and there are many tools in the market, both commercial and open source, with more or less functionalities. Selecting the one that fits best your requirements can be a headache.
The following section of this paper will provide an overview of what a web scanner is and which are their main characteristics and features. This will provide essential knowledge, which could be of great value when matching this knowledge with the later evaluation methodologies.

Before reading any further, a word of advice. These web scanners are powerful tools in a fast and thorough detection of web application security problems. However, they are not the definitive solution for the web applications security problem. Some of these tools give origin to many false positives and false negatives, which need to be confirmed by hand. Techniques like static code analysis and similar are advisable to conduct an in-depth security analysis of the application.

2. Web Application Scanners

In this section of the paper it will be introduced the concept of a web application scanner and which are its main components and functionalities such as crawling, scanning and reporting functions.

2.1 Definition

Web applications scanners [4] can be defined as automated tools, which perform a black box penetration testing on a web application. Web scanners inspect web applications by crawling through their pages and by parsing their contents while applying internal functions that inspect the retrieved content for a list of known vulnerabilities. This inspection often mimics the attacks performed by malicious users generating inputs and analyzing the web application behavior and response. These malicious inputs are often used in a technique called *fuzzing* [5] which most web scanners are able to perform.
A growing number of web scanners are available today both in the commercial and open source World. This demonstrates the exponential growth that these tools are having in particular by the most common development applications and frameworks, which implement security policies into the SDLC. This relationship between the SDLC security policies implementations and the web scanners is pretty straight forward as these tools most are most often used at the end of the SDLC, in particular at the testing stage, thus providing some extra security before market deployment and production.

It is possible to define three main stages in the web scanners tools: the configuration, crawling and scanning stages. These are the core functionalities within every web scanner, which allow them to be effective, user friendly and important assessment tools [6,7,8].

2.2 Configuration

The configuration stage includes the definition of an URL (or IP) of the target to be tested, as well as the setup of different parameters related to the crawling and scanning processes. This is one important feature of web scanners as it allows to define the crawling depth and the crawling ignore statement. This allows the definition of the scanning granularity and thus reducing the scanning time, which can extend for many long hours.
The scanning settings configuration and the ability to pause and resume scanning is of the most importance since it provides the usability that most of the testers require. These scanning configurations often include the definition of scanning profiles with specific vulnerability tests thus leveraging customization between different scanning processes. Within the scanning feature one important aspect is related with scheduling. Scheduling provides the ability for automatically perform previously defined tests in specific times assisting in maintaining a relatively high security level without the use of a specialized human intervention.
The ability to manually perform attacks within these tools is often ignored but it is a feature professionals want and need. These tools are not able to perform tests in vulnerabilities they were not programmed to detect. This is the motivation for manual testing feature inclusion, thus providing the tester the ability to conciliate an automated tool with his knowledge and skills, achieving a better security assessment performance.

2.3 Crawling

The crawling [9] stage is one of the most important features available on these tools because it is where they become "aware" of the web application as a whole. As a result, the scanner will produce a mapping of the structure of the web application, building the path tree starting from the web application root. This will also allow the discovery of images, folders, HTML files, scripts and so forth. The relevance of this stage is crucial. The failure in the identification of any page/resource, will prevent the capability of performing the scanning stage (testing with the defined audit processes), resulting in a defective security assessment outcome. The crawling process can be described as exemplified in Figure 1.

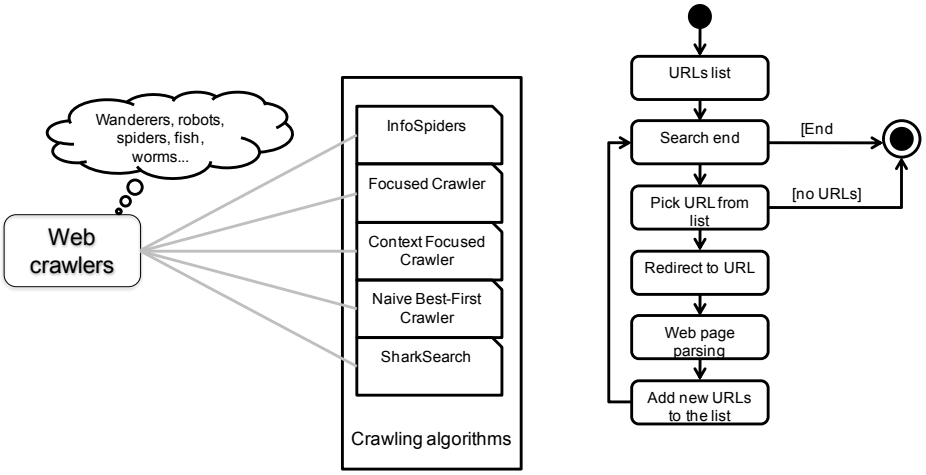


Figure 1 - Crawling process

As demonstrated, the crawler indexes the first link it encounters, examines its source code, and when it finds another link, reproduces the former procedure until no more links can be found. Which is also valid for the different items found.

2.4 Scanning

The last, and the most important stage in these web application tools, is scanning. At this point, web scanners will perform automated penetration tests against the web application. This involves the simulation of attacks to the web application itself, just like an outsider/insider attacker would do, creating and submitting specific inputs and analyzing the corresponding outputs. In this stage, a huge amount of tests will be performed. Usually these tests are predefined in the web scanning, according to the selected scanning options (established during the configuration stage). The outcome of these tests, requests and responses will be analyzed and documented.

It is important to say that at this stage if new relevant information is discovered (for instance, new links), it will be tested against the previously defined scanning processes.
An important note must be made. Web scanners will not detect all vulnerabilities and this must be taken in consideration, as it will be explained in a later section.
Reporting is one of the many characteristics that need to be present in web scanners. It is actually considered an important one as it may contain useful information like the vulnerabilities found, their descriptions and possible mitigations. It is not unusual for these reports to say which entity better describes what these vulnerabilities are and how to correct them (for instance, references to OWASP documentation or any other organization). This is made by graphical user interfaces (GUIs) which allow an ordered and clean data presentation. Web scanners evolved a lot since they first appeared and they went from command line programs to fully graphical applications capable of interacting with users with less specific knowledge. Also, the capability of generating reports in a variety of formats is very appealing for the user.
Most recent web scanners provide a GUI. This is extremely useful because it can reduce the gap between the need to use these tools and the required knowledge to use them. Additionally, reports and real time results may be presented to the user, giving important information as the scanner crawls through the web application, providing a greater interactivity between the tool and the user. Also, this allows the inclusion of functionalities such as scanning pause and resume, which are of the most relevance for professionals.
Besides presenting the results included in the web scanner GUI, these tools provide mechanisms to produce very professional and well structured reports. As a consequence, this feature helps the quick deployment of the result to top management people, which many times need quick and visually effective responses. Less time is spent preparing reports and more time is spent actually assessing security vulnerabilities.

3. Web Application Scanners Evaluation

Web scanners exist in a large number and with mixed functionalities. This huge offer makes it difficult and laborious to find which tool is the “best” one for the job.

Most people new to this segment quickly try to find out which tool should be used. There is no straightforward response to answer “which is the best Web scanner?” question, as different users have different needs and many factors like time, skills, platform, and others, take part on the answer. As a response, different web scanners evaluation frameworks were developed and here it will be presented the ones which have a bigger acceptance between professionals and better fundaments its choices. Also, another positive point in this approach is that it provides quantitative measures and descriptions about which characteristics should these tools possess and why.

3.1 NIST SAMATE

Before delving into the previously mentioned evaluation criteria framework, it is important to know one of the first evaluation criteria for web scanners, the Web Application Security Scanner Functional Specification.
The National Institute of Standards and Technology (NIST) [11] holds a project called Software Assurance Metrics and Tool Evaluation (SAMATE) [12], which is sponsored by the U.S. Department of Homeland Security (DHS) National Cyber Security Division [13].
Part of the SAMATE project is to identify and measure software security assurance tools, where web scanners are included. This project includes information about what are web scanners as well as some specific functionality that these tools should possess, which are described in the Web Application Security Scanner Functional Specification.

As a result of this functional specification, some key points regarding these tools are defined. Web scanners should be able to detect issues regarding the following:

- Web Applications Issues
- Technical vulnerabilities
- Security Vulnerabilities
- Architectural/Logical Vulnerabilities
- Other vulnerabilities

Although this is an important project, as from 1st January 2010, it is no longer maintained and it now supports the WASSEC project.

3.2 WASSEC

WASSEC is a long document and pretends to cover the main aspects related to Web scanners' evaluation. The Web Application Security Scanner Evaluation Criteria [14] (WASSEC) is produced by WASC [15] and, briefly, is defined by:

1. **Protocol Support**
2. **Authentication**
3. **Session Management**
4. **Crawling**
5. **Parsing**
6. **Testing**
7. **Command and Control**
8. **Reporting**

Additionally, there is an interesting scanning evaluation method. The purpose of that method is to define how scanners should be evaluated against their scanning capabilities.
The rationale of this test suite is to choose a set of vulnerabilities to test and, for each one, implement multiple instances from easily exploitable and detectable to the

unbreakable and virtually undetectable. This can be accomplished by hiding those vulnerabilities behind a series of defense mechanisms.

The way to create these levels is as follows:

1. Select the vulnerability
2. Create the levels based on the available information on how to prevent exploiting that vulnerability
3. Explore Web scanner behavior for each level of security

After applying the first and second evaluation methodologies, the user will have a pretty good idea if the scanner actually serves its purposes or not. This is probably the most complete evaluation it is possible to make as it measures the general and specific web scanner characteristics. These tools should be evaluated according to specific measures, based in user's requirements. Therefore, weights should be used in each step, providing the quantitative measures which will aid the user to choose the best one for him.

4. Web Application Scanners Testing

Testing web scanners can be hard and time consuming. Web scanners scanning and vulnerabilities detection capabilities have to be thoroughly tested. In this stage it is important to build a web application, with known and controlled vulnerabilities, so that it is possible to apply several layers of hiding and concealing vulnerabilities. This, as explained before, is important to assess the scanners detection capabilities.
Another important reason to do this is that in some countries, testing web applications without the proper authorizations can be considered illegal and serious problems can come from that [16].

4.1 Where and what to test

Various manufacturers created their own specific test beds adapted for their own web scanning products. This can be an important tool but are often biased because they are built according to their own scanning detection capabilities (a way to prove to customers that their product actually works). To mitigate this situation, several online test beds are also available and a mix of them should be used. Also, an alternative consists in the development of a personalized one. A list of some known examples is shown below:

- Cenzic (live) [17]
- Watchfire (live) [18]
- WebMaven / Buggy Bank [19]
- Updated HackmeBank [20]
- OWASP WebGoat [21]

- Stanford SecuriBench [22]

Another important tool is the OWASP Site Generator [23]. This is a very important tool because it allows building web applications according to our preferences.

5. Case Study

This study was created from the assessment of 17 different real and deployed web applications, which is part of a project aiming to assess Web application vulnerabilities in Portugal within a master thesis scope.

The groups identified to be part of this project, and included in this assessment, range from governmental, high education and other relevant service providers entities within the Portuguese scope.

5.1 Choosing Web scanners

One of the most important tasks in this case study is the Web scanners selection. This selection was made following some pre-determined steps in the following order:

1. Overall Web scanners discovery on the Open Source community

After searching the most known Open Source Web scanners, the candidates for this case study are:

- Grabber
- Grendel-Scan
- Paros Proxy
- Powerfuzzer
- SecurityQA Toolbar
- Skipfish
- W3AF
- Wapiti
- Watcher
- Websecurify
- Netsparker
- OpenAcunetix
- RatProxy

2. Discard the less accepted Web scanners

At this stage the main concern is to eliminate from the study less accepted Web scanners. As these Web scanners are widely used in the Open Source community, the feedback provided from them is highly important and useful on a first approach. The most used and accepted Web scanners are:

- Grendel-Scan
- Paros Proxy
- skipfish
- w3af
- Websecurify
- RatProxy

3. Apply customized WASSEC

Although all the scanners had a similar score on the WASSEC document, the customized evaluation features were decisive. These customized evaluation points are:

- Recent activity and updates
- New technologies support
- OWASP Top 10 coverage
- Fast bugs solving

Having all previous steps into consideration, the web scanners used to perform these tests were the following:

- w3af
- websecurify
- skipfish

3.1 Applying the scanners to the Web applications

The technologies on these web application range from PHP (8 Web applications), Java (1 Web application) and .NET/ASPX (8 Web applications). This is important as a great range of technologies allows having a better understanding of the scanners overall performance instead of exploiting some weakness related with a specific technology. For privacy purposes, these Web applications will not be identified, and an alias will be provided.

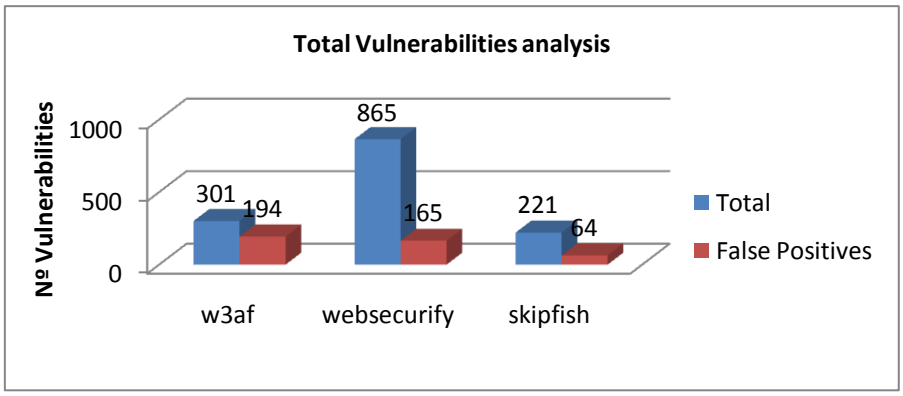


Fig. 1. Total vulnerabilities analysis

For a better understanding regarding the distribution of false positives analysis, bellow there are the several distributions for each Web scanner.

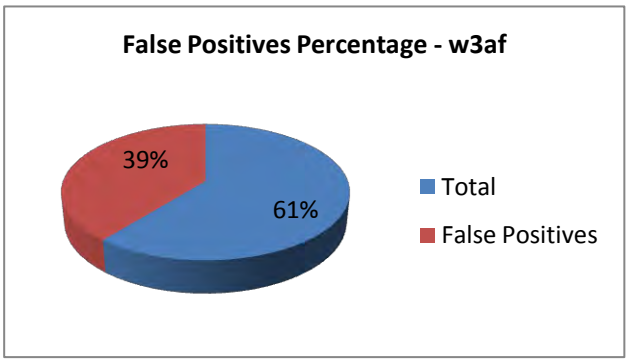


Fig. 2. Vulnerabilities analysis distribution in w3af

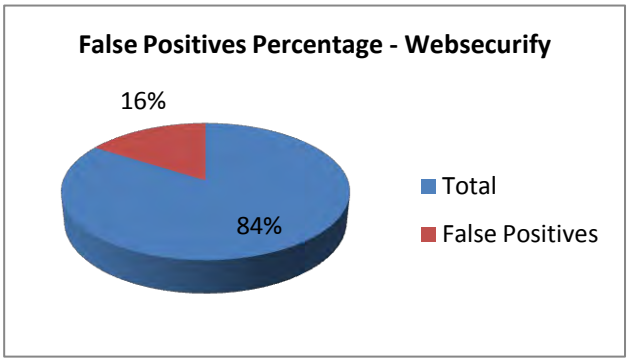


Fig. 3. Vulnerabilities analysis distribution in Websecurify

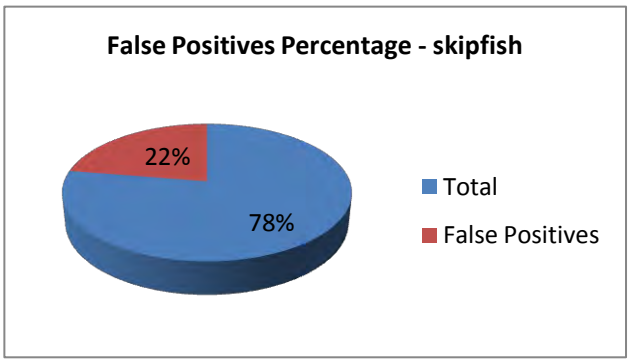


Fig. 4. Vulnerabilities analysis distribution in skipfish

The results show that Web scanners are important and detect a huge number of vulnerabilities although the number of false positives demands that they are used with reservations. The results regarding the 3 Web scanners (Fig. 5) shows that a total of 24% of false positives (in a total of 1161 total vulnerabilities found with the 3 scanners) is still a good number to not discard these tools.

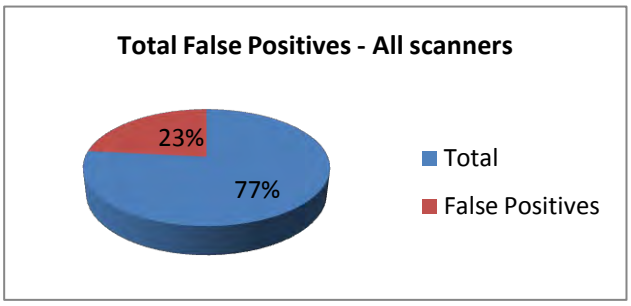


Fig. 5. Total False Positives – All scanners

6. Conclusions

This paper addressed the definition and assessment of web scanners. Web scanners are automatic tools which can perform black-box penetration testing in web applications, and are most often used in the end of the SDLC, and with relative great success. The big problem which such tools is the huge offer the market has, making it difficult to choose the right one. To address this problem, some web scanners' evaluation frameworks were developed, trying to help users with their choice. As it was identified in this paper, the WASSEC is probably the most important method to make this decision as it covers the most important issues and also provides a customized section to add user specific needs in the assessment chart.
Web applications users should therefore chose a few well accepted scanners and use the WASSEC or any suitable method to understand if they fit user's needs. This will be a great cut in evaluation time consumption, providing a faster and efficient way to perform a proper assessment work.

The case study presented in this paper show that Web scanners have flaws and are far from being perfect in detecting all vulnerabilities, but still have good results regarding the number of false positives and the total of vulnerabilities found. This is both important for companies which wish to spend a small amount of money on security, allowing them to achieve a higher security level, and for companies which possess a good security team, backing them up with results derived from tools produced and reviewed by a whole security community.

7. References

1. Now is the time for security at the application level, Gartner Group, http://www.sela.co.il/_Uploads/dbsAttachedFiles/GartnerNowIsTheTimeForSecurity.pdf, visited on May 2010
2. National Institute of Standards and Technology - National Vulnerability Database, http://web.nvd.nist.gov/view/vuln/search-results?cid=2, visited on March 2010.

3. 18. OWASP Top 10, http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project, visited on May 2010
4. Elizabeth Fong; Vadim Okun; , "Web Application Scanners: Definitions and Functions," System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on , vol., no., pp.280b-280b, Jan. 2007
5. Oehlert, P.; , "Violating assumptions with fuzzing," Security & Privacy, IEEE , vol.3, no.2, pp. 58- 62, March-April 2005
6. Ory Segal, Methodologies & Tools for Web Application Security Assessment, watchfire, 2006
7. Elizabeth Fong, Romain Gaucher, Vadim Okun and Paul E. Black, Building a Test Suite for Web Application Scanners, IEEE Computer Society (2008)
8. Larry Suto, Analyzing the Accuracy and Time Costs of Web Application Security Scanners (2010)
9. Gautam Pant, Padmini Srinivasan and Filippo Menczer, Crawling the Web, 2003
10. Wang Ding; Songnian Yu; Qianfeng Wang; Jiaqi Yu; Qiang Guo; , "A Novel Naive Bayesian Text Classifier," Information Processing (ISIP), 2008 International Symposiums on , vol., no., pp.78-82, 23-25 May 2008
11. National Institute of Standards and Technology , http://www.nist.gov/index.html, visited on May 2010
12. Software Assurance Metrics and Tool Evaluation , http://samate.nist.gov/Main_Page.html, visited on May 2010
13. Department of Homeland Security (DHS) National Cyber Security Division, http://www.dhs.gov/xabout/structure/editorial_0839.shtm, visited on May 2010
14. Web Application Security Scanner Evaluation Criteria, http://projects.webappsec.org/Web-Application-Security-Scanner-Evaluation-Criteria, visited on May 2010
15. Web Application Security Consortium, http://www.webappsec.org/, visited on May 2010
16. Taney, F.X.; Costello, T.; , "Securing the whole enterprise: business and legal issues," IT Professional , vol.8, no.1, pp.37-42, Jan.-Feb. 2006
17. Crack Me Bank, http://crackme.cenzic.com/Kelev/view/home.php, visited on May 2010
18. Altoro Mutual, http://demo.testfire.net/, visited on May 201
19. Buggy Bank, http://www.mavensecurity.com/WebMaven/, visited on May 2010
20. Hack me Bank, http://www.foundstone.com/us/resources/proddesc/hacmebank.htm, visited on May 2010
21. Webgoat, http://www.owasp.org/index.php/Category:OWASP_WebGoat_Project, visited on May 2010
22. SecuriBench, http://suif.stanford.edu/~livshits/securibench/, visited on May 2010
23. OWASP Site Generator, http://www.owasp.org/index.php/OWASP_SiteGenerator, visited on October 2010

Weighted Deadline Driven Security Aware Scheduling for Real time Computational Grid

Rekha Kashyap[1], Deo Prakash Vidyarthi[2]

[1] Lal Bahadur Shastri Institute of Management New Delhi,India+91-11-24522474, rekhakashyap@lbsim.ac.in
[2] Jawaharlal Nehru University New Delhi, India +91-11-2670-4738
dpv@mail.jnu.ac.in

Abstract. A new scheduling algorithm Weight balanced EDFMin(WB-EDFMin) for real time grid applications having stringent security requirements is proposed and compared with EDF/ECT, MinMin, MaxMin, SPMinMin and SPMaxMin. The model is based on the observation, that task-set with urgent deadline shows better success rate when scheduled according to EDF/ECT and task-set with relaxed deadline shows better success rate when scheduled according to MinMin. WB-EDFMin prioritizes the task on the weighted sum of its deadline and size priority. Weightage of priorities are computed dynamically on the basis of average deadline of the grid tasks and so takes better of EDF/ECT and MinMin depending on job statistics . For securing the grid job, a security overhead model is formulated which calculates the security overhead for the proposed and compared models on the basis of cipher-suites negotiated. Extensive computer simulation reveals that success rate of WB-EDFMin is best among all the heuristics compared.

Keywords: Real time grid scheduling, Security aware grid scheduling, Cipher-suite, Success rate, Average response time.

1 Introduction

A computational grid is a collection of geographically dispersed heterogeneous computing resources, providing a large single virtual computing system to users [1]. This privately owned heterogeneous, non dedicated network of computers utilizes the idle time of thousands of computing devices across the world producing the power of expensive supercomputers. The dynamic and multi-institutional nature of the grid introduces challenging security threats and therefore new technical approaches towards this problem are to be addressed.

The motivation of grid computing is to aggregate the power of widely distributed resources and provide non-trivial Quality of Services (QoS) to users, where security is considered as an important QoS. In an environment where security is of concern, responsibility is delegated to the scheduler to schedule the task on those resources that meet the security requirement of the task. The Scheduler considers the security constraints exerted by both the job and the grid environment. Such a scheduler is referred as the security aware scheduler [2] [3]. To advocate security as quality of service (QoS), security choices must be presented to the user/application in form of Security Level (SL) and in turn user/application can request a level of security in the

form of Security Demand (SD). The underlying mechanism must be capable of entering into an agreement to deliver the services at the requested level. As of today there are many applications that specify their security demand and the service provider fulfils them. Irvine and Levin in their work have emphasized on the importance of quantifying the security services [4] [5]. Sypropoulou and Agar [6] have worked on quantifying the IPSec Protocol.

Grid computing has extensively supported collaborated projects on the internet and many of them have real time requirements together with stringent security concerns. Thus a real time security-aware grid scheduling algorithm needs to satisfy the security constraints while meeting time deadline and optimizes performance parameters like success rate(number of tasks whose time deadlines are met over total number of task requesting the service), site utilization (percentage of total task running time out of total available time on a given site), makespan (completion time of the entire job set), average response time (average value of all tasks' response time), average slowdown ratio (ratio of the task's response time to its service time) etc. Traditional real time systems are developed to guarantee timing constraints without considering the security concerns of the job, while many security aware schedulers are not for real time applications. In our current work a novel real time scheduler WeightBalancedEDFMin (WB-EDFMin) is suggested which shows better success rate than some of the standard grid schedulers for real time security demanding grid tasks.

Rest of the paper is organized as follows. Some related work has been briefed in next section, Section 3 discusses the proposed work. Section 4, 5 and 6 talk about scheduling, security and deadline strategies respectively. Section 7 shows the working of WB-EDFMin algorithm. Section 8 depicts the results of the simulation experiment. Section 9 infers the conclusion from the work.

2. Related work

Some of the well known grid scheduling algorithm are: DFPLTF (Dynamic Fastest Processor to Largest Task First) which gives highest priority to the largest task [7]. Suffer [8] allocates the processor to the task that would suffer the most if that processor is not assigned to it. Noriyuki Fujimoto and Kenichi Hagihara[9] have proposed *Round Robin* (RR) grid scheduling algorithm for parameter sweep applications which does not require prediction information regarding task length and processor speed. MinMin schedules smallest task at priority to the fastest machine and MaxMin schedules largest task with priority on fastest machine [10]. All these algorithms discussed above are neither real time nor security aware.

SATS, suggested by Xie Tao and Xiao Qin[13] is a security aware scheduler, which takes into account heterogeneities in security and computation. It also provides a means of measuring overhead incurred by security services and quantitatively measuring quality of service (QoS) though it does not assure the desired security rather tries to improve security and minimizes computational overhead. MinMin(Secure)[14] proposed by Song is secured version of MinMin which allocates task to the sites which can provide required level of security. SPMinMin and SPMaxMin[15][16] i.e. Security Prioritized MinMin and Security Prioritized MaxMin

are security aware scheduling algorithms and guarantees security of the job maximizing the makespan. However SATS, SecureMinMin, SPMinMin or SPMaxMin are not designed for real time grid applications as they do not consider the timing constrained imposed by the applications.

One of the well known real time scheduling algorithm is EDF/ECT (Earliest DeadlineFirst/Earliest completion Time). It schedules task with least deadline at priority to the fastest scheduler [11] [12] but it does not considers security.

Some work done for real time secured grid applications are as follows. Secured optimistic concurrency control protocol makes tradeoff between security and time deadline [23]. SAREG is a dynamic real time scheduler whose goal is to maximize the overall quality of security (measured by security value) and guarantee success ratio by dynamically changing each accepted task's security levels [17].

Our work is different from the above as these algorithms aims towards maximizing security for real time applications whereas our work guarantees security for the tasks meeting the deadline and optimizes the success rate.

3 The Proposed Work

A new security aware real time grid scheduling algorithm Weight balanced EDFMin(WB-EDFMin) is proposed and its performance is compared with EDF/ECT, MinMin, MaxMin, SPMinMin and SPMaxMin. As the work proposes security aware scheduling, the computational cost of task takes into account the cost incurred during securing the application. For the comparisons to be fair, the security cost in proposed as well as compared models has been incorporated. While experimenting with existing models it has been noted that EDF/ECT gives best success rate when more tasks in task-set have urgent deadline and MinMin is the best when the schedule has relaxed deadline. The reason for the same is listed below.

1) EDF/ECT schedules tasks with urgent deadline at priority, so when many tasks in the schedule have relaxed deadline, then few task with higher computational size and urgent deadline makes many small task to miss their deadline and degrades the success rate of the EDF/ECT schedule.
2) When many task have urgent deadline its better to schedule them first because it is not wise to successfully schedule few relaxed task at the cost of many urgent tasks. This makes EDF/ECT a better choice for urgent schedule.

WB-EDF takes better of the two schedulers by assigning priority to grid task which is weighted sum of the task deadline priority and size priority. Lesser the deadline more the deadline priority and smaller the computational size of task, more is its size priority. Further the weights associated for each priority is dynamically computed for each schedule. More weight is given to deadline priority if the schedule demands relaxed deadline and in this situation our model will behave more like EDF/ECT and less like MinMin. Similarly more weightage is given to size priority for urgent schedule. Weight computation is shown in section 7 where the algorithm is described in detail.

4 Scheduling Model

Grid is considered to be composed of number of non dedicated processing nodes which in turn may be a single processor or a group of heterogeneous or homogeneous processors. A grid job is comprised of "n" independent tasks with different computational size, security requirements and deadline constraints. A task with a desired security demand can be executed only on a resource providing required or higher security level and is considered scheduled if it is able to execute within its specified deadline. Following is the list of terminologies, used in this paper.

A task T_i is characterized as $T_i = (Sz_i, SD_i, D_i)$, where, Sz_i is the computational size, SD_i is the security level demand and D_i is time deadline of the i^{th} task.

A processing node N_j is characterized as $N_j = (SP_j, SL_j, BT_j)$ where, SP_j is the speed of node, SL_j is the maximum security level offered by the processing node, and BT_j is the begin time of the node (time to execute the tasks already assigned to the node).

$NS_{qualified,i}$ is the list of all the processing nodes on which the i^{th} task can be executed i.e. list of the nodes qualifying the security demand of the i^{th} task. $ND_{qualified,i}$ is the list of all the processing meeting the deadline of the i^{th} task.

A schedule of the job is the set of n quadruplet $< T_i, P_j, BT_j, CT_{ij}>$ where, T_i is i^{th} task, P_j is j^{th} processing node, BT_j is Begin Time of j^{th} processing node and CT_{ij} is the completion time of i^{th} task on j^{th} processing node. CT_{ij} is calculated using Equation 1.

$$CT_{ij} = BT_j + ET_{ij} + SO_{ij} \quad (1)$$

Where, ET_{ij} is Execution Time of i^{th} task on j^{th} processing node and SO_{ij} is the Security Overhead time of i^{th} task on j^{th} node.

5 Security Model

One of the key factors behind the growing interest in grid computing is the evolution of standards such as TCP/IP and Ethernet in networking. For the TCP networking model IPsec, TLS/SSL and SSH are the most popularly used security protocols operating on its network, transport and application layer respectively[18][19] [20][21]. These protocols, offer security to any grid application by providing the common security services of key exchange, authentication, confidentiality and integrity. Key exchange algorithms are used to securely exchange a shared secret value between two computers over an unsecured network connection. Confidentiality is necessary to protect the information that is being sent between the communicating parties with a strong encryption algorithm. Integrity is important to verify that the received information is the same as it was sent and is incorporated through digital signatures and hash functions. Authentication is necessary to verify that the information has come from intended sender, and that it is received by who is supposed to receive it, i.e. mutual authentication. Each protocol is further configured to match differing security requirements through cipher suites negotiations where Cipher Suite, is a named combination of key exchange, authentication, encryption, and integrity algorithms used to negotiate the security settings for a network connection. During

an SSL handshake, for example, the two nodes negotiate to see which cipher suite they will use when transmitting messages back and forth. The selection of the security protocol and the cipher suite inside that protocol decides the security provided for the task. When a client (that is, a Web browser) establishes a SSL session, the client sends a Client Hello message to the server. Amongst other things, this message contains all the supported cipher suites the client can handle. A single cipher suite is a combination of an encryption protocol (that is, DES, RC4, AES), the encryption key length (40, 56, or 128) and a hash algorithm (SHA or MD5) that is used for integrity checking. When the server receives the Client message, the server controls and decides which offered cipher suite it will use to secure this particular session.

For realizing security as a quality of service parameter the choice of security need to be given to the client task and grid resource must be capable of entering into an agreement to deliver the services at the requested level. Instead of presenting all possible security protocol support and combinations of security mechanisms through cipher-suites for the variant services, a simplified abstraction of security in the form of security level range can be offered to the users. The security level also provides a mechanism for calculating the security overhead expenses. In the present work, we are using the security range from 1 to 10 and have abbreviated requested security service level of the task as security demand (SD) and the offered security service level by the grid site as security level (SL). SL for each cipher-suite can be computed as the weighted sum of security services involved in the cipher-suite. Cipher-suite having security algorithms with longer keys and more secured algorithms will provide more security at the cost of computational cost and should be assigned a high security level.

For the sake of demonstrating our idea of security level assignment for the security protocols, we have listed a subset of cipher suites supported by SSL V3 protocol in Table 1. The third line SSLCipherSpec SSL_RSA_WITH_DES_CBC_SHA indicates use of DES with 56-bit encryption. The fourth line SSLCipherSpec SSL_RSA_WITH_3DES_EDE_CBC_SHA indicates use of 3DES with 168-bit encryption. Among the six cipher suites mentioned in the Table 1 the first one provides the weakest security and the sixth one provides the strongest security. Thus the first cipher suite can be assigned the security level 1 and the sixth can be assigned the security level 6. For more realistic security level assignment for cipher suites the exact security overhead for each cipher suite needs to be calculated, which is the sum total of exact security expense of every algorithm used in the particular cipher suite. As this is a non trival task we are proposing it for our future work.

Table 1. The Subset of cipher suites supported by SSL V3 protocol.

SSLCipherSpec	SSL_RSA_WITH_RC4_128_MD5	Security Level 1
SSLCipherSpec	SSL_RSA_WITH_RC4_128_SHA	Security Level 2
SSLCipherSpec	SSL_RSA_WITH_DES_CBC_SHA	Security Level 3
SSLCipherSpec	SSL_RSA_WITH_3DES_EDE_CBC_SHA	Security Level 4
SSLCipherSpec	SSL_RSA_EXPORT_WITH_RC4_40_MD5	Security Level 5
SSLCipherSpec	SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	Security Level 6

5.1 Security Overhead Computation

Security overhead is calculated as suggested by Tao Xie and Xiao Qin [22] (Equation 2), where, SL_i is in the range [1, 2, 3…..R] and 1 and R are the lowest and highest level of the security. For calculation of security overhead, quality of security service need to be quantified in the form of security levels. For the experiments security level value is obtained on the basis of the cipher suite negotiated between the task and grid node. A task can specify its key exchange, authentication, encryption, and integrity requirements together with the protocol it wants to use in the form of security Demand which in turn decides the cipher suite selection. Every cipher suite should have a security level attached with it.

The rationale behind the security overhead model is based on the observation that security overhead of a particular application tends to be proportional to the execution time of the application. In other words security overhead depends upon the amount of data to be secured thus it is product of execution time(which depends upon data size) and relative security required as shown in Equation 2. Total computation time considering security overhead is shown in Equation 3. Calculating the exact security overhead is a non trivial task and is beyond the scope of this work. As the current work involves comparison of various algorithms thus security expense calculation error arising due to simple security overhead is there in all the algorithms and thus the results of comparison are not much affected.

$$SO_{ij} = ET_{ij} (SL_i /R) \quad (2)$$

$$CT_{ij} = BT_{j+} ET_{ij} (1+ SL_i /R) \quad (3)$$

6 Deadline Model

Real-time workloads have deadlines associated with the tasks but for synthetic workloads deadline assignment is done using Equation 4. For each job, we compute the time to completion on the unloaded node of average specification which is further scaled up by a delay factor (DF) to generate a mixed of urgent and delayed deadline tasks. Higher delay factor gives greater value of deadline and thus is relaxed in nature.

$$D_i = (ET_{i,avg} + SO_{i,avg})* DF \quad (4)$$

Where $ET_{i,avg}$ is the execution time of the i^{th} task on a average unloaded computer. $SO_{i,avg}$ is security overhead time of the i^{th} task on average unloaded computer and DF is the delay factor. For the experimental simulated environment the delay factor value ranges between 1 and 4.

7 WB-EDFMin algorithm

WB-EDFMin schedules jobs based on the priority assigned to the task using Equation 5. Task with highest WB-EDFMin calculated priority is scheduled first on fastest available machine.

$$P_i = W_{deadline} * PD_i + W_{size} * PSz_i \quad (5)$$

Where P_i is the overall priority of the i^{th} task , PD_i is priority based on deadline modeled by Equation 6, PSz_i is priority based on computational size modeled by Equation 7, $W_{deadline}$ is weight assigned to deadline priority and W_{size} is weight assigned to priority based on computational size

$$PD_i = 1 - (D_i / D_{max}) \quad (6)$$

$$PSz_i = 1 - (Sz_i / Sz_{max}) \quad (7)$$

Sz_i is the computational size of the i^{th} task. Smaller task have higher priority based on size. Sz_{max} is the maximum value of computational size in the entire task set. D_i is the deadline of the i^{th} task. Tasks having lesser deadline value have higher deadline priority. D_{max} is the maximum value of deadline in the entire task set.

Weights for the priorities are further computed based on average delay factor (DF_{avg}) for the schedule using Equations 8, 9 and 10.

$$DF_{avg} = \sum_{i=1}^{n} DF_i \ / \ n \quad (8)$$

$$W_{deadline} = DF_{avg} / DF_{max} \quad (9)$$

$$W_{size} = 1 - W_{deadline} \quad (10)$$

The average delay factor of the schedule decides the overall urgency of all tasks within a schedule. Higher value means relaxed schedule and lower value means tight schedule. It has been proved in our experiments that EDF/ECT works better than MinMin when more jobs have relaxed deadline and MinMin outperforms EDF/ECT when more jobs in the schedule have urgent deadline. Thus we have given more weightage to deadline priority when schedule have relaxed average deadline and more weightage is given to smaller task when schedule has urgent average deadline.

For the real workloads delay factor is not specified but the deadline is given. To calculate weightage of the deadline and size priorities average delay factor is required. Delay factor is calculated as shown in Equation 11.

$$DF_i = D_i / (ET_{i,avg} + SO_{i,avg}) \quad (11)$$

Where $ET_{i,avg}$ is the execution time of the i^{th} task on a average unloaded computer. $SO_{i,avg}$ is security overhead time of the i^{th} task on average unloaded computer and Di is the time deadline specified in the task.

8 Experimental Evaluation and Observations

To validate and evaluate the performance of the proposed WB-EDFMin over existing algorithms i.e. EDF/ECT, MinMin, MaxMin, SPMinMin and SPMaxMin, a simulator in java is designed and the experiments are carried out. The simulator is composed of Grid-Generator and Task-Generator. Component Grid-Generator is responsible for the simulation of the grid environment. It can randomly generate different properties of heterogeneous grid nodes; Component Task-Generator can randomly generate different tasks over the range as listed in Table 2. The experimental study considers the complete heterogeneous environment e.g. deadline of the task, security requirement of the tasks, security offered by the nodes, speed of the nodes and size of the task. Deadline is synthetically generated from random selection of delay factor. The experimentations compares the performance of the algorithm on the performance metrics success rate and average response time by varying (a) Deadline value(varying delay factor) (b) Number of grid tasks and (c) Number of Grid Nodes.

EDF/ECT, MinMin and MaxMin are made secured by imposing security constraints and overhead in their implementations and are then compared with WB-EDFMin, SPMinMin and SPMaxMin for comparisons to be fair. To make the scheduling real time, in implementations of all algorithms, tasks are considered scheduled on the node where its overall completion time is less than the deadline specified in the task. Fig. 1 shows the working of WB-EDFMin algorithm.

Performance Metrics

Comparison of the four algorithms is done on following metrics:

- **Success rate**= Number of tasks that meet the deadline/ total number of tasks.
- **Average response time** = $\sum_{i=1}^{n} (CT_i - ARV_i) / n$. (The response time of the task is the time period between the task arrival and its completion time).

```
do until (T_Complete != NULL)    // There are more tasks to schedule
  {
  Compute Pi for each task T_i  in T_complete
      P_i =  W_deadline* PD_i + W_size * PSz_i
      Select T_p=max(P_i)     // Task with highest priority
     Create N_qualified,p        // Nodes satisfying the security  requirement of T_p
  for each node j from node list N_qualified,p
        {
        Compute CT_ij = ET_ij  + BT_j + SO_ij
        }
  Create ND_qualified,i    // Nodes meeting the deadline  of the i^th task.( D_i > CT_ij)
  N_m= Node with min(CT) from nodeset ND_qualified,i
                          // Node meeting deadline and   offering minimum completion time
  Schedule task T_p on node N_m
  Delete task T_p from  T_complete
  Modify BT_m = BT_m + CT_km   // Begin time for the node N_m is modified
 }
```

Fig. 1. The WB-EDFMin Scheduling Algorithm.

Table 2. Input parameters for the simulation experiments

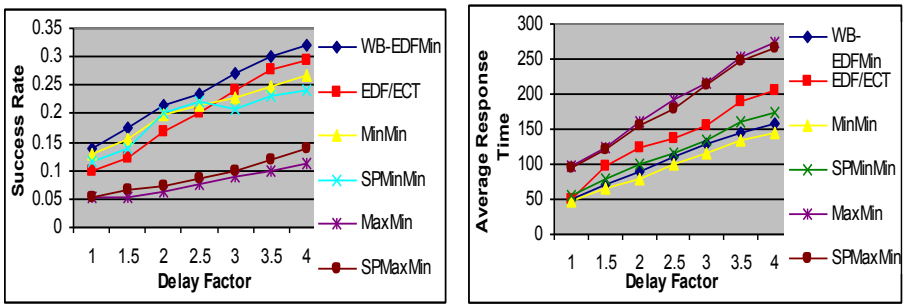
Parameter	Value Range
No of nodes.	1 to 60 (fixed 30).
Speed of the processing node (SP).	1, 2, 5, 10 (MIPS).
Security level of the processing node (SL).	1 to 15.
No. of tasks.	100 to 3000 (fixed 500).
Size of tasks.	10 to 100 (MB).
Security level demand of the grid task (SD).	1 to 10.
Delay factor of the task.	1 to 4.

8.1 Performance Impact of Deadline

In first set of experiment, the proposed algorithm WB-EDF is compared with some existing algorithms by varying the deadline. Different deadlines are generated by varying the delay factor of the task in the range 1 to 4 and fixing node size to 30 and task size to 500. The result of this simulation is shown in Fig 2(a) and 2(b). It has been observed that EDF/ECT gives a better success rate for larger deadline or greater delay factor and MinMin shows better success rate when delay factor is less or deadline is urgent. WB-EDFMin is the better of the two as it dynamically adjusts the priorities of tasks as per the average delay factor of the schedule. Its performance for success rate is best of all the algorithms over the entire range of deadline. MaxMin and SPMaxMin shows the worst success rate because assigning heavier task at priority will allow very few task to meet their deadline. Average response time measures the overall wait time for the entire task set. Since MinMin gives priority to smaller task it shows the best response time and WB-EDFMin is the second best among all the heuristics compared as far as response time metric is concerned.

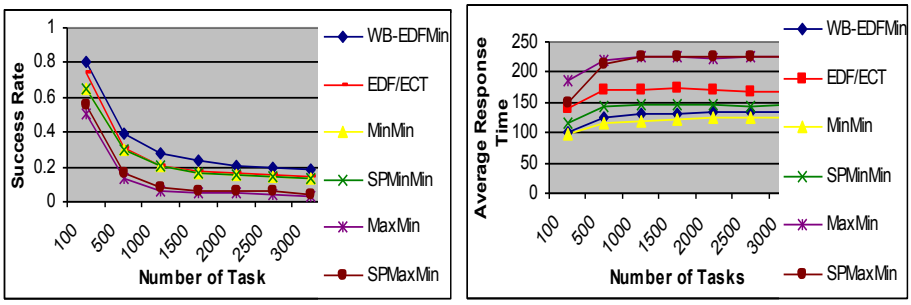
8.2 Performance Impact varying Number of Tasks and Number of Nodes

In the next set of experimentation, we measure the success rate and average response time performance by varying number of tasks to be scheduled from 100 to 3000 fixing node size to 30 and delay factor tot 3. WB-EDFMin shows best success rate than the compared schedules and improvement over other scheduling model is almost constant with the increasing value of number of tasks and is depicted in Fig. 4(b). In the last set of experimentation we have scaled the nos of nodes from 10 to 60. Fig. 5(a) shows that WB-EDFMin is showing better success rate than the compared heuristics. Average response time of MinMin is best and WB-EDFMin is second best when tasks and nodes are scaled (Fig 4(b) and 5(b)).



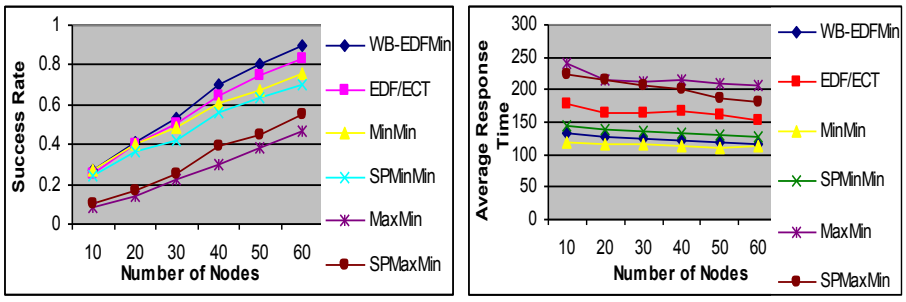
(a) Success Rate (b) Average Response time

Fig. 2. Performance comparison varying the Delay Factor (deadline).



(a) Success Rate (b) Average Response time

Fig. 3. Performance comparison varying the Number of Tasks.



(a) Success Rate (b) Average Response time

Fig. 4. Performance comparison varying the Number of Nodes.

9 Conclusion

In this paper we have suggested a new scheduling algorithm for grid applications, which is secured and real time in nature. For securing the grid task a security overhead model is suggested which assigns security level to cipher suite negotiated. For more realistic security level assignment for cipher suites the exact security overhead for each cipher suite needs to be calculated, which is the sum total of exact security expense of every algorithm used in the particular cipher suite. As this is a non trivial task we are proposing it for our future work.

Success rate and response time are the top most design goal for any real time application. Extensive experiments with various test configurations show that WB-EDFMin gives superior success rate than the compared MinMin, EDF/ECT, MaxMin, SPMinMin and SPMaxMin and its average response time is second best after MinMin.

References

1. Foster, I.: What is Grid? A three point checklist, http:// www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf. Cited (2004)
2. Humphrey, M., Thompson, M.R.: Security Implications of Typical Grid Computing Usage Scenarios. HPDC, 95-103 (2001)
3. Tonellotto, N., Yahyapour, R., Wieder P. H.: A Proposal for a Generic Grid Scheduling Architecture Core GRID TR-00 (2006)
4. Irvine, C. E., Levin, T. E.: Toward a Taxonomy and Costing Method for Security Services: Proceedings of the 15th Computer Security Application Conference, Phoenix, AZ. (1999)
5 Levin, T. E., Irvine, C. E., Spyropoulou, E.: Quality of Security Service: Adaptive Security, The Handbook of Information Security. John Wiley & Sons, Inc (December 2005)
6. Sypropoulou, E., Agar, C., Levin, T. E., Irvine, C. E.: IPsec Modulation for Quality of Security Service", Proceedings of the International System Security Engineering Association Conference, Orlando Florida. (2002)
7. Paranhos, D., Cirne, W., and Brasileiro, F.: Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. In International Conference on Parallel and Distributed Computing (Euro-Par), Lecture Notes in Computer Science, volume 2790, 169–180 (2003)
8. Casanova, H., Legrand, A., Zagorodnov, D., Berman, F.: Heuristics for scheduling parameter sweep applications in grid environments. In 9th Heterogeneous Computing Workshop (HCW), pp. 349–363 (2000)
9. Fujimoto, N., Hagihara, K.: Near-optimal dynamic task scheduling of independent coarse-grained tasks onto a computational grid. In 32nd Annual International Conference on Parallel Processing (ICPP-03), pp. 391–398 (2003)
10. Freund, R. F., Gherrity, R. M., Ambrosius, S., Campbell, M., Halderman, D., Hensgen, E. Keith., T. Kidd., M. Kussow., Lima, J. D., Mirabile, F. L., Moore, L., Rust, B., Siegel, H. J.: Scheduling resources in multi-user, heterogeneous, computing environments with smartnet, In the 7th IEEE Heterogeneous Computing Workshop (HCW'98),pp. 184–199 (1998)

11. W, Zhao., K, Ramamritham., J,A, Stankovic,: Scheduling Tasks with Resource Requirements in Hard Real Time Systems, IEEE Trans. Software Eng., vol. 12, no. 3, pp. 360-369, May (1990)
12. Stankovic, J., Spuri, M., Ramamritham, K., Buttazzo, G. C.: Deadline Scheduling for Real-Time Systems – EDF and Related Algorithms*," Kluwer Academic Publishers*, (1998)
13. Xie, T., & Qin, X.: Performance Evaluation of a New Scheduling Algorithm for Distributed Systems with Security Heterogeneity" J. Parallel Distributed. Computing. 67, pp. 1067– 1081 (2007)
14. Song, S., Kwok, Y., Hwang, K.: Trusted Job Scheduling in Open Computational Grids: Security-Driven Heuristics and A Fast Genetic Algorithms," Proc. Int'l Symp. Parallel and Distributed Processing .(2005)
15. Kashyap, R., Vidhyarthi, D. P.: Security Prioritized Computational Grid Scheduling Model: An Analysis. International Journal of Grid and High Performance Computing, 1(3), pp. 73-84 (2009)
16. Kashyap, R., Vidhyarthi, D. P.: A Security Prioritized Scheduling Model for Computational Grid. In International conference at HPC Asia. pp 416-424 (2009)
17. Xie, T., Qin, X.: Enhancing Security of Real-Time Applications on Grids through Dynamic Scheduling,. Proceedings of the 11th Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP'05), PP.146-158, Cambridge, MA (2005)
18. Salter, M., Rescorla, E., Housley, R.: RFC 5430 Suit B Profile for Transport layer Security and TLS version 1.2, http://tools.ietf.org/html/rfc5430. Cited (2009).
19. Dierks T., Rescorla, E.: "RFC 4346, The Transport layer Security (TLS) Protocol Version 1.1" Available at http://tools.ietf.org/pdf/rfc4346.pdf cited (2006)
20. Hoffman,P.: RFC 4308, Cryptographic suites for IPSec, http://www.ietf.org/ rfc/rfc4308.txt. Cited (2005)
21. Microsoft TechNet, Description of the IPSec Algorithm and methods, http:// technet.microsoft.com/en-us/library/dd125356. Cited (2009)
22. Xie, T., Andrew, S., Qin, X.: Dynamic Task Scheduling with Security Awareness in Real-Time Systems. Proceedings of the 19th International Parallel and Distributed Processing Symposium (IPDPS'05) (2005)
23 Ahmed, Q., Vrbsky, S.: "Maintaining security in firm real-time database systems," Proc. 14th Ann. Computer Security Application Conf.(1998)

From Risk Awareness to Security Controls: Benefits of Honeypots to Companies *

Sérgio Nunes[1] Miguel Correia[2]

[1] Novabase/Universidade Atlântica [2] Universidade de Lisboa

Abstract. Many companies are deploying their business on the Internet using web applications while the question of what is the risk to business operations of cyber-attacks remains unanswered. Risk awareness allows to identify and act upon the security risk of these applications. This paper analyzes different security frameworks commonly used by companies in order to evaluate the benefits of honeypots in responding to each framework's requirements and, consequently, mitigating the risk.

1 Introduction

Many companies are currently deploying their business on the Internet using *web applications*. From online shops to business-to-business applications, from web sites that allow making inquiries to applications that allow doing complex operations, more and more companies are getting in the Web 2.0 wagon. The data accessed through web applications is becoming more and more critical, containing private information that enables financial transactions in multiple online businesses. This vicious cycle is growing and organizations are unable to do risk awareness to be able to analyze these new web threats.

This new massification of web technologies poses multiple questions regarding information security: What is the role of security with this significant change? Is there a decrease in the confidentiality, integrity and availability of information with this new situation? Are there any new security threats that put information at risk?

By exposing web applications in a *honeypot*, attacks can be captured and investigated, as also can the tools and actions of the attacker after the intrusion [13,2,1,16]. The careful analysis of the gathered attack data and the know-how gained by managing honeypots provide an insight about the modus operandi and motives of the attacker, classifying him according to a pre-established profile. Having the attacker profile defined, the threat model can be specified in order to develop the necessary *risk awareness* and *risk mitigation controls*.

Risk mitigation is accomplished in organizations by employing a variety of information security, compliance and *risk frameworks* that address multiple domains across the wide information technology environment. This paper considers three frameworks: ISO/IEC 27001 [6], COBIT [4] and PCI-DSS [9]. These frameworks present a major focus in security guidelines by providing specific control

* This work was partially supported by the FCT through the CMU-Portugal partnership and the Large-Scale Informatic Systems Laboratory (LaSIGE).

requirements and objectives to mitigate risk in organizations integrating people, processes and technology as a whole. These frameworks present most of the time general guidelines that do not descend to specific security technologies, so it is important to evaluate how common security technology concepts adapt to these frameworks. Honeypots can bring added value to such frameworks by satisfying multiple enumerated control requirements.

2 Honeypots

A *Honeypot* was defined by Spitzner as an information system resource whose value lies in unauthorized or illicit use of that resource [13]. The value of this security mechanism relies on monitoring the real steps and tools of a real attack and learning where the unknown vulnerabilities lie and how to protect the critical information assets. These monitoring and decoy capabilities aid the security professional in developing the required know-how of the modus operandi of the attacker and infer the security situational awareness of his network to plan for the adequate safeguards and effective incident responses [15]. Detecting what is unknown via monitoring and providing information for the analysis of the attack is the main factor that differentiates this tool from the rest of the security toolset.

Another concept used in the terminology of honeypots is *honeytokens*. They serve as digital entities that reveal unauthorized access when used [7]. They follow the same principle of not being used for legitimate purposes and can be for example a fake credit card number, a supposed secret document, a false email or a bogus login that is carefully placed among legitimate information.

Honeypots can be classified as research or production [3]. The research honeypots are used by the academic community to study the attacker and gather information about his tools and actions. The production honeypots help an organization mitigating the attack risk by focusing the attacker's attention in useless decoy assets, while the critical assets are safeguarded. This deception enables timely and adequate security incident responses.

Honeypots can emulate vulnerable network services to gather attack information without being exposed to a real intrusion. This type of honeypots is called *low interaction* because of the limitation of malicious activities due to basic service emulation. The deployment of a real operating system with the vulnerable service is know as *high interaction* honeypot and is able to gather the real evidence of the intrusion and to follow the additional steps performed by the attacker after gaining control of the system. Some literature also presents the definition of *mid-interaction* honeypots as the attacker's ability to fully act against an integrated honeypot daemon service, but not being able to compromise the operating system below [10,14].

Honeypots are called physical when there is a real machine connected to the network and virtual when the machine is a guest system residing in a virtualization environment [11,12]. Honeypots can be static and stay implemented without change from the initial deployed architecture or be dynamic and adapt automatically to respond to the attacker's behaviour. The honeypot technology finds place in diverse fields of use, especially where awareness is necessary combined with a

proactive security posture. The most common fields of use are intrusion detection systems, malware, botnets, spam, phishing, wireless and web [2,1,16].

To better understand the benefits of honeypots for web application risk awareness we made an experiment with a high interaction honeynet, which was reported elsewhere [8]. We executed 10 virtual honeypots with different web applications during 3 months. The main results of the experiment were:

- We observed 8858 attacks during that period of time, which shows the risk to which web applications are exposed.
- The most targeted web applications were: PhpMyadmin (81% of the attacks), TomcatManager (8%), Zencart (6%) and Roundcube (3%).
- The most common attacks were: URL bruteforce (73%), command execution (22%), authentication bruteforce (2%).
- The source of attacks were diverse as it can be seen in Figure 1 with the United States (35%) and China (16%) as the major attackers.

Fig. 1. Worldwide attack origin distribution

3 ISO/IEC 27001

3.1 Description

The ISO/IEC 27001 is an international standard that provides a model for establishing an Information Security Management System (ISMS) as a strategic organization decision [6]. The word system does not imply a real asset, but a defined and monitored methodology or security program. The ISMS is formed by tools, processes, templates, documents and best practices. The ISMS can be defined as an overall management system from a business risk perspective that has to be established, implemented, operated, monitored, and maintained. It mandates that the organization systematically examines its risks taking into account

threats and vulnerabilities, implements control procedures to deal with those risks and adopts a continuous improvement information security management process that continuously responds to business security needs.

The ISO/IEC 27001 is used in conjunction with ISO/IEC 27002, formerly known as ISO/IEC 17799 [5], that establishes the code of practice for information security management. This code of practice contains specific controls for dealing with most requirements of ISO/IEC 27001 including technical security, but ISO/IEC 27001 expects that these measures are already taken care of and focuses on the mandatory requirements of an Information Security Management System. ISO 27001 focuses on these control objectives from ISO/IEC 27002 in annex A.

3.2 Benefit Analysis

In the general requirements of the ISO/IEC 27001 standard (Section 4.2) it is stated that it is necessary to assess the realistic likelihood of a security failure occurring in the light of prevailing threats and vulnerabilities and impacts associated with the assets and the controls currently implemented. This realistic likelihood can be measured effectively using real honeypots with the same vulnerabilities as production systems. This approach mimics the production systems behaviour exposing the decoys to the same threat level. The ISO/IEC 27001 standard mandates that is crucial to monitor and review the ISMS to identify attempted and successful security breaches and incidents. The honeypots could bring to this requirement increased added value when compared to traditional intrusion detection systems, because of the detailed information gathered about an attack, which enables gaining real know-how and situational awareness of the risk that the asset faces.

In ISO/IEC 27002, the code of practice for ISO/IEC 27001, there are some controls that can be adapted to the added value of honeypots. The control for protection against malicious code (27001 Annex A.10.4.1) can be complemented with a honeypot by performing evaluation of malicious code using client honeypots and by having a honeypot infrastructure capable of monitoring malicious code spreading mechanisms. The use of multiple different malware analysis is suggested in the standard as a vector to improve the effectiveness of malicious code protection.

The ISO/IEC 27002 standard states that is necessary to reduce risks from exploitation of technical vulnerabilities (27001 Annex A.12.6). The control defines that timely information about technical vulnerabilities of information systems being used should be obtained, the organization's exposure to such vulnerabilities evaluated and appropriate measures taken to address the associated risk. This is the main focus of the honeypot technology and by adequate use of honeypots it is possible to accomplish this goal of establishing an effective management process for technical vulnerabilities that responds to the requirements.

The ISO/IEC 27002 standard details the need to ensure a consistent and effective approach to the management of information security incidents (27001 Annex A.13.2.2). It suggests defining the responsibilities and procedures to deal with the incidents collecting forensic evidence for internal problem analysis. This

collection of evidence can be gathered using honeypots or honeypot data gathering mechanisms. Maintaining the chain of custody has multiple requirements, so training how to collect and preserve the evidence should be an exercise first performed on decoy systems such as honeypots, to prepare for a real incident on production systems. The ISO/IEC 27002 standard states that there should be a learning experience from information security incidents allowing the incidents to be monitored and quantified. The information gained from the evaluation of information security incidents should be used to identify recurring or high impact incidents. This learning can be developed with the risk and threat awareness delivered with the continuous use and analysis of honeypots. Honeypots were founded as a unique possibility of learning about the modus operandi of attackers developing situational awareness about the security status of the infrastructure, allowing investigators to develop the know-how to recognize and treat these incidents with appropriate procedures when they happen in the real critical systems.

In the ISO/IEC 27002 standard there is a section concerning the correct processing in applications (27001 Annex A.12.2) detailing components such as input and output data validation that are the cause of multiple web attacks. Although the honeypots are no direct defense against those attacks, they provide the necessary learning and research capabilities necessary for secure programming and correct evaluation of the risk that results with the lack of validation in applications. The attacked decoy web applications can measure the threat level and serve as case studies for future applications developed.

The protection of organizational records is also a subject detailed in the ISO/IEC 27002 standard regarding its loss, destruction or manipulation (27001 Annex A.12.5.4). Organization information disclosure attacks happen frequently in an enterprise and they are difficult to prevent or even to detect. The concept of honeytokens can help in the detection of disclosure of critical data by placing careful bogus monitored records in such datastores and track those records while they travel through the network serving as a warning that the data is being disclosed. These detection mechanisms can be complemented with intrusion prevention solutions that limit data losses by identifying the bogus records and block further data travel or tear down the connection.

The summary of the honeypots concepts and their relation to the ISO/IEC 27701 standard can be found in Table 1.

Honeypot Concept	**ISO/IEC 27001**
Risk Awareness	4.2 Establishing and managing the ISMS
Secure Coding	A.12.2 Correct processing in applications
Malicious Code Detection	A.10.4.1 Controls against malicious code
Information Disclosure Detection	A.12.5.4 Information leakage
Vulnerability Management	A.12.6 Technical vulnerability management
Incident Response	A.13.2.2 Learning from information security incidents

Table 1. Honeypot benefits to ISO/IEC 27001

4 COBIT

4.1 Description

Nowadays information technology (IT) processes are key activities of any organization and the dependence of the business operations from IT becomes impossible to dissociate. This close dependence can have drastic consequences if not carefully controlled and measured, as business requirements tend not to be shared with IT. It is crucial to understand that the business drives the investment in IT resources and those resources are used by IT processes to deliver the information necessary back to business. The Information Systems Audit and Control Association (ISACA) published the Control Objectives for Information and Related Technology (COBIT) to help information technology governance professionals to align technology, business requirements and risk management [4]. The Committee of Sponsoring Organizations (COSO) is an internal control framework for organizations to deal with financial, operational and compliance-related internal controls and COBIT provides those controls for information technologies.

COBIT may be positioned at the higher business management level dealing with a broad range of IT activities and focuses on how to achieve effective management, governance and control. Being maintained by a non-profit independent group with continuous research improvement, it integrates seamlessly with other standard and best practices as it forms a set of principles that can be adapted to business needs. It covers five areas of IT Governance: Strategic Alignment, Value Delivery, Resource Management, Risk Management, Performance Measurement. COBIT deals with effectiveness, efficiency, confidentiality, integrity, availability, compliance and reliability as business requirements and applications, information, infrastructure and people as resources while adopting the necessary processes for supporting activities. COBIT is illustrated by a process model with 34 processes distributed among four distinct domains:

- Plan and Organise (PO): This domain covers strategy and tactics, and concerns the identification of the way IT can best contribute to the achievement of the business objectives. Furthermore, the realisation of the strategic vision needs to be planned, communicated and managed for different perspectives. Finally, a proper organization as well as technological infrastructure must be put in place.
- Acquire and Implement (AI): To realise the IT strategy, IT solutions need to be identified, developed or acquired, as well as implemented and integrated into the business process. In addition changes in existing systems and their maintenance are covered by this domain to make sure that the systems life cycle is continued.
- Delivery and Support (DS): This domain is concerned with the actual delivery of required services, which range from traditional operations over security and continuity aspects to training. In order to deliver services, the necessary support processes must be set up. This domain includes the actual processing of data by application systems, often classified under application controls.
- Monitor and Evaluate (ME): All IT processes need to be regularly assessed over time for their quality and compliance with control requirements. This

domain addresses management's oversight of the organization's control process and independent assurance provided by internal and external audit or obtained from alternative sources.

4.2 Benefit Analysis

The COBIT plan and organise domain has a process that deals specifically with assessing and managing IT risks (Control PO9). Among its control objectives there is the requirement of risk event identification, where an event is an important realistic threat that exploits a significant applicable vulnerability causing a negative impact to business. These events deal with multiple aspects: business, regulatory, legal, technology, trading partner, human resources and operational. Under the technology events, honeypots can play the role of accessing the threats to the assets, determining the severity of the impact when dealing with vulnerabilities and developing risk awareness in the organization. This enables to determine the nature of the impact and adds value to the risk registry by detailing real relevant risks that might pass unnoticed without using decoy systems. Another control objective inside this process is conducting risk assessments on a recurrent basis being able to determine the likelihood and impact of all identified risks, using qualitative and quantitative methods, where honeypots can gather the necessary information to qualify and quantify the risk impact on technological assets. Honeypots can also contribute to the risk response control objective being able to prepare the personnel for real responses due to training gathered from honeypot detailed attack information analysis.

The COBIT acquire and implement domain has one process that deals with acquiring and maintaining application software (Control AI2) where honeypots also present an adequate measure when regarding risk awareness. Honeypots bring benefits to the application security and availability control objective by feeding the know-how regarding application attacks and how to code adequate security safeguards. Being honeypots most of the time compromised to install distributed denial of service zombies, they create the necessary awareness regarding threats against availability and show how common denial of service proliferates. Regarding the development of application software control objective, honeypots promote secure coding by showing developers how the attacks work and how they can be suppressed. This is performed with detailed information gathered from decoy systems enabling the research without harming critical production systems. The acquire and maintain technology infrastructure process has the control objective of infrastructure maintenance that mandates that there should be a strategy and plan for the infrastructure maintenance that includes periodic reviews against business needs, patch management, upgrade strategies, risks, vulnerability assessment and security requirements. Honeypots contribute to risk awareness that help to identify and quantify risks, they also show new methods and tools to exploit known vulnerabilities, uncover unknown vulnerabilities and provide information to respond to the security requirements that mitigate the risk caused by them.

The deliver and support domain has a specific process that deals with ensuring systems security. The security testing, surveillance and monitoring control objective (Control DS5.5) has the task of ensuring that the enterprise security baseline

is maintained by testing and monitoring the IT implementation in a proactive way. It states that a logging and monitoring function will enable the early prevention and detection and subsequent timely reporting of unusual or abnormal activities that may need to be addressed. The honeypot technology promotes the proactivity by learning from attacked decoys and gathering the latest malicious tools used by attackers. It monitors the environment detecting unusual or abnormal activities while deviating the attention of the attacker from the critical systems. It tests the security baseline of enterprise systems by evaluating the robustness of the honeypots deployed using that security baseline.

Another control objective inside the systems security process (Control DS5.6) is to define security incidents communicating its characteristics so they are properly classified and treated by the problem management process. The classification of security incidents needs specific training and awareness of threats and security risks. Although multiple courses exist on that matter, it is crucial to have live training on the organization's infrastructure to adapt to real incidents and this is where a research honeypot testbed will help. The honeypot testbed trains the personnel by dealing with attacks in research decoy systems that do not interfere with business critical systems and develop a risk awareness mindset that allows them to recognize characteristics of security incidents when they really happen in production systems.

Another control objective inside this process (Control DS5.9), where honeypots play a vital role, is malicious software prevention, detection and correction. Honeypots have measures to deal with malicious software such as viruses, worms, spyware and spam. Worms are detected, monitored and researched by compromising honeypot decoys, spyware is evaluated using client-side honeypots and spam is detected and mitigated using email honeypots in conjunction with honeytokens.

The delivery and support domain has a process that deals with data management and has one control objective of defining and implementing the policies and procedures (Control DS11.6) to identify and apply security requirements applicable to the receipt, processing and storage and output of data to meet business objectives. The requirement of data confidentiality must be maintained in order to preserve business secrets and assure the privacy of clients' records, so information disclosure should be detected. Honeytokens can be used to limit information disclosure by ensuring the detection of careful placed bogus records maintaining data security. The summary of the honeypots concepts and their relation to the COBIT Framework can be found in Table 2.

Honeypot Concept	**COBIT**
Risk Awareness	PO9 Assess and manage IT risks
Secure Coding	AI2 Acquire and maintain application software
Malicious Code Detection	DS5.9 Malicious software prevention, detection and correction
Information Disclosure Detection	DS11.6 Security requirements for data management
Vulnerability Management	DS5.5 Security testing, surveillance and monitoring
Incident Response	DS5.6 Security incident definition

Table 2. Honeypot benefits to COBIT

5 PCI-DSS

5.1 Description

The payment card industry data security standard (PCI-DSS) was developed to assure cardholder data security and unify consistent data security measures globally [9]. It was created by American Express, Discover Financial Services, JCB, MasterCard Worldwide and Visa International to establish requirements for the security of the payment card industry affecting everyone that stores card payment data, including common online commercial transactions. It is guided by a continuous process to ensure adequate monitoring and improvement of requisites by assessing, remediating and reporting procedures. It has six control objectives and establishes twelve requirements for compliance.

5.2 Benefit Analysis

The Payment Card Industry Data Security Standard was build with the main requirement of protecting the cardholder data when dealing with online transactions. It has requirements dealing with storage of limited card information explaining what shouldn't be kept in the database and mandates that encryption is used during travel and storage to protect cardholder data from disclosure (Requirement 3.1). Here the honeytoken principle can be used to detect the disclosure of data by using decoy invalid cards and deviating the attention during an attack from the real encrypted cards.

The PCI-DSS mandates that a process must be established to identify newly discovered vulnerabilities responding to the requirement of developing and maintaining a secure system or application (Requirement 6.2). This requires a constant vulnerability awareness program that can be complemented by deploying decoy honeypot systems monitored to find new vulnerabilities. Only reading security disclosure information from outside sources might not be enough to catch new threats as the vulnerability disclosure time window is decreasing throughout the years, leaving no time for security administrators to act accordingly protecting critical systems.

This standard has detailed requirements of coding guidelines against specific web vulnerabilities such as Cross-site scripting (XSS), Injection flaws, Malicious file execution,Insecure direct object references, Cross-site request forgery, Information leakage and improper error handling, Broken authentication and session management, Insecure cryptographic storage, Insecure communications and Failure to restrict URL access. Secure coding best practices (Requirement 6.5) against these vulnerabilities can be achieved by learning from the attacks that web applications face everyday. Although attacks can be detected using intrusion detection systems, there is no detailed information available that can serve as a learning experience, which is why honeypots are better to learn how vulnerabilities work and how they are exploited by attackers in order to promote secure coding from the lessons learned. This vulnerability awareness and know-how becomes crucial in understanding the issues detected by vulnerability assessments as there is another requirement that states that is necessary to conduct them on an ongoing

basis. The know-how gained from honeypot analysis allows developers to address these issues with secure programming to safeguard for similar situations.

The standard mandates that anti-virus software is installed on all systems commonly affected by malicious software and that it should detect, remove and protect against those threats (Requirement 5.1.1). It should be updated, running and generating audit logs. The honeypot architecture can complement anti-virus software by testing unknown suspicious malware in a controlled environment that the anti-virus has not classified yet. It is capable of detecting web malware using client honeypots and monitors the consequent behaviour and provides implicit detection against worms by detecting its propagation to decoy systems. The concept of having available a honeypot test environment provides the necessary know-how to detect, remove and treat unknown malware threats.

PCI-DSS also requires that an incident response plan (Requirement 12.9) is documented along with providing the appropriate training to staff with security breach response responsibilities. This continuous training can be achieved with periodical external courses, but practical onsite training is also fundamental to get familiar with the infrastructure and issues encountered. Honeypots can perform this role providing staff with onsite non critical system training to develop the necessary incident awareness to respond to real situations. Humans learn better by practicing and making mistakes, so honeypots provide such a research infrastructure without affecting the critical assets.

The PCI-DSS explains that the information security policy should reference that there should be an annual process that identifies threats and vulnerabilities resulting in a formal risk assessment (Requirement 12.1.2). This formal risk assessment promotes the risk awareness capabilities of the organization annually, but this awareness should be maintained with continuous improvements to ease annual evaluation and there honeypots can play a vital part. Honeypots contribute to the identification of threats to business with decoy infrastructures, monitoring exploited vulnerabilities, gathering detailed information about intrusions and malicious actions performed by attackers.

The summary of the honeypots concepts and their relation to the PCI-DSS can be found in Table 3.

Honeypot Concept	**PCI-DSS**
Risk Awareness	12.1.2 Identify threats and vulnerabilities,conduct risk assessment
Secure Coding	6.5 Develop all web applications with secure coding guidelines
Malicious Code Detection	5.1.1 Detect, remove and protect against malicious software
Information Disclosure Detection	3.1 Keep cardholder data storage to a minimum
Vulnerability Management	6.2 Identify newly discovered security vulnerabilities
Incident Response	12.9 Implement an incident response plan

Table 3. Honeypot benefits to PCI-DSS

6 Discussion

It can be observed from the previous individual risk framework analysis that the honeypots can bring benefits to multiple requirements in each framework. The honeypot contribute is not constrained to benefit measures specific to each framework, because they all deal with the same basis requirements under different names and aggregated in different groups (Table 4). More generically, the major benefits of using honeypot concepts when dealing with risk frameworks are:

- The creation of a risk awareness culture being able to correctly identify the threats to IT and evaluate the impact to business of attacks;
- The promotion of secure coding by learning from the application attacks suffered, evaluating the coding vulnerabilities that were explored and developing the safeguards necessary to correct them;
- The detection of malicious code due to monitorization of propagation attempts and unusual activity, along with the testing of suspicious webpages and binaries in a test decoy environment;
- The detection of disclosure of information with the monitorization of decoy bogus items (honeytokens);
- The creation of an accurate and timely vulnerability management framework being able to identify, analyze and patch with a minimum time delay recently disclosed exploits and malicious tools used by attackers;
- The creation of an incident management and response system capable of identifying, classifying and addressing security problems;

Honeypot Concept	ISO/IEC 27001	COBIT	PCI-DSS	Benefit Impact
Risk Awareness	4.2	PO9	12.1.2	High
Secure Coding	A.12.2	AI2	6.5	Low
Malicious Code Detection	A.10.4.1	DS5.9	5.1.1	High
Information Disclosure Detection	A.12.5.4	DS11.6	3.1	Medium
Vulnerability Management	A.12.6	DS5.5	6.2	High
Incident Response	A.13.2.2	DS5.6	12.9	Medium

Table 4. Summary of the honeypot benefits to three frameworks studied

7 Conclusion

In this paper an analysis of the basic concepts of honeypots is performed in comparison with the control requirements of frameworks in order to infer the added value that this security mechanism can bring to enterprises. This research confirmed our previous belief that honeypots are useful for companies but underestimated by them, probably mainly because of a lack of knowledge regarding this technology, its uses and benefits.

The fear of challenging the attacker and being unable to control the consequences of the intrusion is also a deterrence factor in the use of honeypots by

companies. These issues are never balanced with the possibility of developing the necessary risk awareness within the company using these decoy systems to be able to defend the critical assets when a real attack emergency happens.

Companies have multiple risk and security frameworks already in place to be able to respond to compliance requirements. In these risk frameworks the demand of developing a risk awareness program is detailed with the deployment of multiple controls. In this research some of these frameworks were analysed and it can be concluded that the honeypot technology plays a vital part in responding to those needs by applying some of its basis concepts.

References

1. Anagnostakis, K.G., Sidiroglou, S., Akritidis, P., Xinidis, K., Markatos, E., Keromytis, A.D.: Detecting targeted attacks using shadow honeypots. In: Proceedings of the 14th USENIX Security Symposium. pp. 129–144 (August 2005)
2. Baecher, P., Koetter, M., Dornseif, M., Freiling, F.: The Nepenthes platform: An efficient approach to collect malware. In: Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection. vol. 4219, pp. 165–184. Springer (2006)
3. Baumann, R., Plattner, C.: Honeypots. White Paper, Swiss Federal Institute of Technology (2002)
4. ISACA: Cobit framework 4.1. http://www.isaca.org (2007)
5. ISO/IEC 17799: Information technology - security techniques - code of practice for information security management. http://www.iso.org (June 2005)
6. ISO/IEC 27001: Information technology - security techniques - information security management systems - requirements. http://www.iso.org (October 2005)
7. McRae, C.M., Vaughn, R.B.: Phighting the phisher: Using web bugs and honeytokens to investigate the source of phishing attacks. In: Proceedings of the 40th Annual Hawaii International Conference on System Sciences (January 2007)
8. Nunes, S., Correia, M.: Web application risk awareness with high interaction honeypots. In: Actas do INForum Simposio de Informatica (September 2010)
9. PCI-DSS: Payment card industry data security standard version 1.2. http://www.pcisecuritystandards.org (October 2008)
10. Pouget, F., Dacier, M., Debar, H.: White paper: honeypot, honeynet, honeytoken: terminological issues. Research Report RR-03-081, Institut Eurecom, France (2003)
11. Provos, N.: A virtual honeypot framework. In: Proceedings of the 13th USENIX Security Symposium. pp. 1–14 (August 2004)
12. Provos, N., Holz, T.: Virtual honeypots: from botnet tracking to intrusion detection. Addison-Wesley, Boston, MA, USA (2007)
13. Spitzner, L.: Honeypots: Tracking Hackers. Addison-Wesley, Boston, MA, USA (2002)
14. Wicherski, G.: Medium interaction honeypots. German Honeynet Project (April 2006)
15. Yegneswaran, V., Barford, P., Paxson, V.: Using honeynets for internet situational awareness. In: Proceedings of the 4th ACM/USENIX Workshop on Hot Topics in Networks (November 2005)
16. Zou, C.C., Gong, W., Towsley, D., Gao, L.: The monitoring and early detection of internet worms. IEEE/ACM Transactions on Networking 13(5), 961–974 (2005)

OWASP The Open Web Application Security Project

IBWAS'10

2nd OWASP Ibero-American Web Application Security Conference

16.17 December 2010

ISCTE . Instituto Universitário de Lisboa

LISBOA . PORTUGAL

ISBN 978-1-937087-00-5