Asset Identification for Security Risk Assessment in Web Applications

Hisham M. Haddad [1] and Brunil D. Romero [2]

(1) Computer Science and Information Systems, Kennesaw State University (USA)
E-mail: hhaddad@kennesaw.edu

(2) Processes and Systems Department, Simón Bolívar University (Venezuela)
E-mail: bromero@usb.ve

ABSTRACT

As software applications become more complex they require more security, allowing them to reach an appropriate level of quality to manage information, and therefore achieving business objectives. Web applications represent one segment of software industry where security risk assessment is essential. Web engineering must address new challenges to provide new techniques and tools that guarantee high quality application development. This work focuses asset identification, the initial step in security risk assessment for web applications. Risk assessment helps organizations determine security risks in information management systems. The formal approach to identifying information assets for risk assessment is investigated using the MAGERIT methodology and EBIOS method. This work is carried out at Simón Bolivar University (Venezuela) for its Student Opinion Survey Coordination web-based application. Under this research, a methodological tool for asset identification was developed to help the University achieve security risk assessment. Assets are identified according to their priorities in the organizational environment. This work contributes to Web Engineering in general, and to Information Security Management with emphasis on security risk assessment.

Keywords: Information Security, Methodologies, Asset Identification, Organization Management, Risk Assessment, Tools, Web Applications, Web Engineering.

1- INTRODUCTION

The complexity of web applications requires the consideration of appropriate mechanisms that guarantee their quality. Security is a quality attribute that must be part of web applications. In fact, the Security Risk Management field has evolved and established valuable risk assessment procedures and tools. During the last decade, the development and use of web applications have been more complex, sophisticated, and intensified. The current design and development of web applications manage solutions for different needs inside the organization, such as guarantee of effective handling of information and support for decision making. The development of information systems, through web applications, should accommodate variety of security requirements inside an organization.

Web applications have special characteristics such as immediacy, evolution, and continuous growth that define their development process as incremental and evolutionary [1]. These characteristics make web engineering different than conventional software engineering. In particular, application security and risk assessment. Risk assessment is used as a diagnostic tool to determine security risks in an organization. According to L. Sena and S. Tenzer [2], risk assessment represents the heart of security procedures in an organization seeking to achieve global security management. Independent of the environment, common processes for risk assessment involve determining what assets need protection, of what to be protected, and how to make them protected [3].

Asset identification, the initial step of risk assessment, is of vital importance because it guarantees successful diagnosis of security concerns in web applications. Each asset is affected by a specific security dimension (dependability, integrity, and/or confidentiality). Therefore, there are many complex events to evaluate in order to suggest a systemic approach to manage security risks. In addition to these considerations, this work involved a preliminary study of risk assessment theoretical foundations (framework) and a case study to determine interrelated characteristics that define the complexity of a web application. This work was carried out at Simón Bolivar University (Venezuela) for its Student Opinion Survey Coordination web-based application. The goal of this project is to develop a methodological tool for asset identification, establishing the groundwork for the remaining steps of security risk assessment (asset assessment and testing).

This paper is organized as follows: Section 2 highlights relevant background information; Section 3 describes the research methodology adopted for this project. Section 4 presents the developed asset identification methodological tool. Section 5 is a reflection and concluding remarks.

2- BACKGROUND

This work required a preliminary study of software security vulnerabilities and mitigation strategies that can be adopted for the development of web applications. The goal is to understand what security risk assessment involve when evaluating web applications. The root causes of most security vulnerabilities are within the software and are introduced during development. There are process improvement models, risk management methods, and good practices and supporting tools that have been reported to help reduce vulnerabilities and exploitable defects. However, there is no single practice, process, or methodology offers the solution for software security. Therefore, different solutions have to be adapted for particular vulnerabilities. Developers must be aware of such vulnerabilities and their consequences [4].

2-1 SECURITY VULNERABILITIES

Wide range of software security vulnerabilities have been reported in the literature. A classification of vulnerabilities includes the following categories:

1. Input Vulnerabilities: Denial Of Service (DOS) [5], Social Engineering [6], Input Data Integrity, SQL Injection [7], and Cross-Site Scripting (XSS) [8].
2. Internal Data Vulnerabilities: Buffer Overflow [9], Memory Dump [10], and Malicious Code [11].
3. Algorithmic Vulnerabilities: They results from debugging sessions that can reveal the inner working of algorithms, allowing unauthorized users to test the code under different input conditions to reveal its secrets.
4. Output Vulnerabilities: They result from software outputs and include Redirection, Piggy Backing, and Information Disclosure.
5. Extensibility Vulnerabilities: They result from the integration of Commercial Off The Shelf (COTS) software components and mobile code components. These integrated components can attempt to compromise the security of the overall system [12].

2-2 MITIGATION STRATEGIES

Security is enforced using functionalities such as data encryption, authentication, and access control [13]. Data encryption consists of making the information either illegible (in order to ensure data confidentiality) or unalterable (digital signature in order to ensure data integrity), or both. Encryption is used to protect stored and exchanged information against reading or modification. Authentication, on the other hand, ensures the entity's identity. That is, to verify that the entity (e.g. a user or a process) is actually the one that it claims to be. Authentication is considered the fundamental security mechanism. Access control governs operations on system's entities. For instance, in a file system, access control consists of checking whether users are allowed to access files. Access control clearly depends upon authentication, the entity's identity having to be unique and unforgivable. These security functionalities are the basis for mitigation strategies to over come the vulnerabilities discussed in the previous section. These strategies help developers identify engineering practices and development approaches to incorporate security into the design and implementation of the software.

2-2-1 Input Strategies

Denial Of Service: A successful DOS mitigating strategy is to constantly check the source of user input and detect any "excessive" access to the software from that source. Requesting access to the system and attempting to supply the same user credentials from more than one source during a given time period or window should be checked, detected and disallowed. If the system requirements demand access from multiple sources at the same time for the same user, a finite number should be used to reduce the resources the user can claim during these sessions.

Social Engineering: Integrating biometrics [14], two factor challenge response, and other stringent user authentication methods should be employed to minimize a successful social engineering attack on the software. Enforcing rollover or change of authentication credentials supplied by users to the software can also minimize unauthorized access. User credentials with a limited life span can prevent access if the attacker takes too long to supply the credentials they obtained by social engineering means. In the case of brute force attack, enforcing good non-trivial credentials gives attackers a challenging time, encouraging them to move on. Non-trivial credentials include usernames with random letters and numbers and passwords with at least four types of characters.

Input Data Integrity: The developer should check all input data to make sure it meets the requirements and if possible perform checks on the input data and transaction history. For example, in banking applications, if a customer has been depositing or withdrawing same amount for several months, the software should be able to flag if this pattern changes and allow for follow up to test the integrity of the account transactions.

Sql Injection: Validating user input take into account the following [15]: use of parameterized queries and stored procedures; use of limited and offset parameters; use of white-list (principle of least privilege); and ensuring to provide sufficient information to the user when an error occurs.

Cross-Site Scripting (XSS): Prevention calls for good coding practices and verification of input parameters. General approaches to prevent XSS include: encoding outputs based on input parameters; filtering input parameters for special characters; and filtering outputs based on input parameters for special characters. Additionally, it is recommended to validate input parameters using white-list (principle of least privilege) versus black-list approach (disallow) [15].

2-2-2 Internal Data Strategies

Buffer Overflow: The developer should use a modern string classes that does not depend on a character delimiter for the size or length of a string. These string class variables, as found in Java and other programming languages, provide a buffer against users taking advantage of knowledge of how a string or data is stored in the memory.

Memory Dump: Developers should ensure that information is encrypted while being temporarily stored in the memory and only decrypted while computation is being applied. Fast hashing algorithms and processing power is available to allow for data placed in memory to be encrypted and decrypted on the fly. This will substantially reduced the likelihood of a memory dump causing information to be available after the termination of the program code.

Malicious Code: The developer, who is aware of the Input vulnerabilities previously discussed and have implemented mitigating code, will find the output to

their computations not as expected after a malicious program injected its own data. The developer can reduce this form of vulnerability by being effective in validating all input data before use and as previously mention encrypting the data only when decrypting is used. This on the fly decryption can be used to validate the origin of the data. If it was not entered through the normal input interfaces or API, then it would not be decrypted properly and the developer should discard and throw an exception.

2-2-3 Algorithmic Strategies

Debugging and reverse engineering is a common approach for securing algorithms. Developers for their part in securing the software code should ensure that the proprietary algorithm is not easily exposed when the software is released for use. The developer can employ encryption and decryption techniques on the code during execution. Setting the code so it is not easily debugged is also a good way to secure the code. Other mitigation techniques include Fail-safe-design[1]; defining how sensitive data has to be managed to get secured; and auditing and logging options.

2-2-4 Output Strategies

Redirection: Setting up an SSL tunnel or some other trust relationship, using strong encryption should be employed by the developer to secure the information transfer from the software to the target recipient.

Unauthorized Viewing (piggy backing): In some cases, security may involve requesting an authenticated user's credential to verify that the person who requested the data is the person who is about to read it. The display should have a limited lifespan on the screen. This however is context base. A web page with blog entry and information can stay on the screen indefinitely. A customer's account information should stay displayed for a limited time (thirty to sixty seconds or so), requiring a refresh or interaction by the user to ensure that the user is still available to view the information.

Information Disclosure: To prevent information disclosure, it is necessary to take into account the following coding practices: never include sensitive data in code; eliminate code remarks and comments before generating the final version; handle error messages from a lower layer and customize them before showing them to the users; and avoid showing **unnecessary** information, among others.

2-2-5 Extensibility Strategies

There are different mitigating techniques to minimize the security impact of COTS components, including confinement by placing restrictions on the privileges COTS have when executing, if they are compromised they cannot take advantage of the privileges of the executing module.

(1) Event of failure responds in a way that will cause no harm or at least minimize the impact.

For extensible and mobile code software, the developer can make sure that only signed and approved modules can be used on their system using various crypto graphical means, verifying all communications among the modules so modules don't interfere with each other's operations. R. Grimm and B. Bershard [16] gave an excellent discussion on facing and ensuring system security in extensible systems.

Table 1 provides a summary of discussed vulnerability classes and their suggested mitigation strategies.

Table 1 Vulnerabilities and mitigation strategies.

Class	Description	Mitigation Strategies
Input	DOS, DDOS, Social Engineering, Input Integrity, Sql injection, and XSS.	Check Input, Minimize resource use per session, trust relationship with user.
Internal data	Buffer Overflow, Memory Dump, Malicious Code.	Encrypt/decrypt data as needed. Verify source data.
Algorithmic	Debugging, Reverse Engineering.	Encrypt/decrypt hashing data and code segments.
Output	Redirection, Unauthorized viewing (piggy backing), information disclosure.	Only show requested data, limit output/sessions.
Extensibility	COTS, extensible, mobile code software.	Validate COTS, trust relation with code, and use only verified signature add-ons and libraries.

2-3 IMPLEMENTATION OF MITIGATION STRATEGY

The CYBSEC Security Systems [15] determined that software vulnerabilities mainly deal with design and implementation faults, these kinds of faults deal with the software development life cycle. Furthermore, recent studies [8,17,18] trying to solve the complexity of mitigation strategies found two aspects that affect the management of vulnerabilities: 1) software theoretical foundations and 2) software business interest. These two aspects are diverging, especially when the software industry deal with cost cutting measure. Both aspects share a common point that should be recognized. That is, software quality involves high cost. One approach to offset the high cost of development and be profitable is to integrate security practices into each phase of the development life cycle. Such practices help mitigate security vulnerabilities in software products.

In addition, there are external aspects related to insecure applications [15], such as exponential growth of software applications, increasing number of vulnerability research (availability of testing tool, black market of vulnerabilities and legitimate vulnerability market), time-lag between vulnerabilities and availability of their solutions, and patch implementation cost. These aspects reflect the complexity of software vulnerability.

Requirements Analysis: In this development phase, asset identification (the first step of security risk assessment [2]), helps developers identify the kind of threats the software is likely to have. The *Input* and *Output* mitigation strategies should be considered for this phase.

Design: In this phase, developers must conduct a threat modeling to describe the possible threats that can occur in a given security environment. It's composed of three high-level steps: 1) understanding the adversary's view, 2) characterizing the security of the system, and 3) determining threats. The *Internal Data* and *Algorithmic* mitigation strategies should be exercised in this phase.

Implementation: Often the choice of the programming language is done tactically for compatibility reasons. However, the language strengths and weaknesses, along with the principles and practices for secure coding, should be the key factors. In addition, implementing cryptography is a viable option. Even so, it's recommended not to develop new cryptography algorithms because it's different from being a (good) security-code developer [18]. Also, the time it takes to develop new algorithms. The *Internal Data*, *Algorithmic* and *Extensibility* mitigation strategies should be exercised in this phase.

Testing: This phase involved activities that take place throughout the life cycle even before there are any code artifacts to test [19]. Functional security testing and risk-based security testing should be exercised. Functional testing ensures correct software behavior; while risk-based testing addresses software risks and probe a specific risk that was previously identified through risk analysis. Automated tools for security testing are available and should be utilized throughout the life cycle. Since some tools can perform simple tasks, the development team should select testing tools appropriate to the application being developed.

Operation and Maintenance: Security aspects should be considered to mitigate vulnerabilities, such as installation design, installation and hardening of the base software, installation process, and operation and maintenance management.

Finally, throughout the development cycle, it is important to learn from mistakes by implementing the following activities: 1) record and explain identified vulnerabilities; 2) share information related to mitigation strategies used with developers; and 3) keep historical record by vulnerability type, responsible people and risk level. Another important recommendation is to establish security metrics to measure quality in: 1) the final product (vulnerabilities amount and security functionality fulfillment) and 2) development process (fulfillment level of security in software development life cycle). In addition, it is significant that security metrics: 1) be measured in a coherent way (objective and repetitive criterions), 2) come from inexpensive data collections, 3) have precise measurement units, and 4) be expressed in numbers (percentage, proportion or coefficient).

3- THEORETICAL FRAMEWORK

For the purpose of this work, this section describes the theoretical foundation for this project. It highlights three essential aspects: risk assessment and its practices, web application assets, and organizational environment.

3-1 RISK ASSESSMENT

Risk is defined as an eventuality that disables the achievement of an objective. In the technological environment, risk is generally outlined alone as threat for what is required to determine the occurrence grade of this eventuality and take the necessary actions to reduce its impact [20]. “Risk is a function of the likelihood of a given threat-source’s exercising a particular potential vulnerability, and the resulting impact of that adverse event on the organization” [17]. Risks in a technological environment (shown in Fig. 1) illustrate direct relationship among the following elements:

1. Threat: Actions that can cause negative consequences in the operative process of an organization.
2. Assets: Assets related to the information system or application to evaluate (data, hardware, software, services, documents, human resources, among others).
3. Impact: The consequences of the threat’s different occurrence.
4. Vulnerability: Certain inherent conditions to the assets or that exist in their environment to facilitate materialization of threats making assets vulnerable.
5. Likelihood: Evaluating all activities with uncertainty of what can be expected.

Figure 1 Elements of technology risk.

Risk assessment is a diagnosis tool to establish real exhibition of risks in the organization. It is also known as the heart of all organized performance to achieve security global management. Risk assessment implies determining *What it is needed to protect*, *Of what to be protected*, and *How to protect it*. Risk assessment involves a Risk Management Process illustrated in Fig. 2.

Risk Management refers to evaluating organizational resources to achieve certain security exhibition level [21]. The importance of risk management resides in its ability to allow identifying future impacts of all projects in the organization risk structure. It is a continuous process since it is necessary to periodically evaluate if newly identified risks and the exposure to these risks calculated in previous stages stay effective [2]. In addition, *risk projection* (risk estimation) tries to measure each risk in two ways - the probability that the risk is real and the consequences associated with the risk, if it happened.

Figure 2 Risk Management Process.

3-2 PRACTICES OF RISK ASSESSMENT

The practices of risk assessment are essential part of software development. The primary goal is to identify and eliminate those risks with the greatest potential to occur. This practice involves processes, procedures, and tools to help organizations identify and manage potential risks.

The Ellipse method (proposed by Gómez A. [22]) was considered in this work as it facilitates asset identification. The method consists of three ellipses: concentric (basic processes), intermediate (interaction between sub-processes), and external (extrinsic organizations but have some relationship with the ap-

plication). The method allows visualizing different sub-processes that conform to the entire application. It also allows identifying assets with user and process ownership.

Furthermore, a group of risk assessment methods and tools, including COBRA, CRAMM, EBIOS, MAGERIT, and OCTAVE, were evaluated under this work. The evaluation criteria involved various parameters such as assistance software availability, languages, first release date, conformance to IT standards, and some general aspects that show their global vision. EBIOS and MAGERIT, described below, were selected as they are supported by the most current IT standards.

3-2-1 Methodology for Information Systems Risk Analysis and Management (MAGERIT)

MAGERIT is a public methodology [23] that was elaborated by the Superior Council of Electronic Administration of Spain. Its objective is to study risks relevant to information systems and their associated environments. It is conformed by specific series of techniques for risk assessment: table analysis, algorithmic analysis, attack trees, general techniques, and cost-benefit analysis. MAGERIT´s specific objectives are:

1. to make professionals who are in charge of information systems aware of the existence of risks and the necessity to deal with them on time,
2. to offer a systematic method to analyze such risks,
3. to help discover and formulate an appropriate plan of action to keep risks under control, and
4. to support the organizational process for evaluation, auditing, certification, or accreditation.

MAGERIT is used to introduce security mechanisms into information system core to mitigate the system's weaknesses and to ensure successful development of the system. It is possible to cover various types of information systems, independent of their complexity or importance. MAGERIT phases are shown in the Fig. 3.

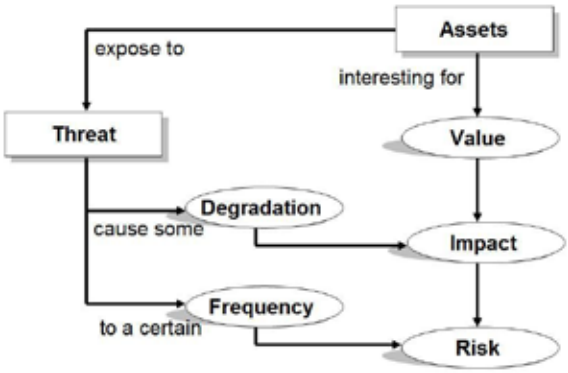

Figure 3 MAGERIT phases.

3-2-2 Expression of Needs and Identification of Security Objectives (EBIOS) Method

The EBIOS method is promoted by Central Information Systems Security Division (France) as international norm. It offers a software tool to help users produce risk analysis and management steps according the five EBIOS method phases. The EBIOS tool is free open source [24]. The method allows appreciating and treating risks relative to Information Systems Security (ISS). As an advantage, the method can be adapted to the context of each organization and be adjusted to its own tools and methodologies, respecting the general philosophy of the procedure. This flexibility is a true toolbox for the ISS actor. In addition, it helps develop a complete global study of some information systems and a detailed study of a particular information system. Fig. 4 shows EBIOS method phases.

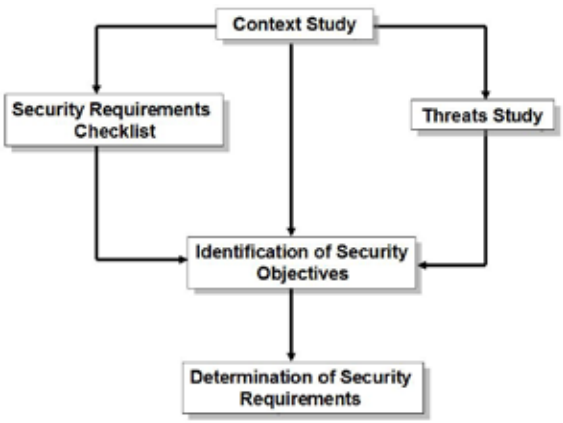

Figure 4 EBIOS method phases.

3-3 WEB APPLICATION ASSETS

The MAGERIT methodology states that “assets are seen as elements of an information system or an application (or closely related with it) that give value to the organization” [23]. It affirms that dependences among assets define the measure of how a superior asset is affected by an inferior asset security incident. The superior assets depend on other assets such as equipments, communications or human resources. The dependency exists when the superior asset security necessities are reflected in the inferior asset security necessities. In other words, when a threat materializes in the inferior asset, it causes damage in the superior asset. The MAGERIT methodology views asset organization in layers. Layer 1: Environment; Layer 2: Information system or application; Layer 3: Information; Layer 4: Organization functions; and Layer 5: Other assets such as credibility or good image, accumulated know how, criterion independence or performance, among others.

On the other hand, the EBIOS method refers to assets as entities. It states that “the evaluated system is formed of a group of technical and non-technical entities that are convenient to identify and describe” [24]. The entities can be of different types: hardware, software, networks, human resource, establishments, organization, and the systems. These entities require protection since they could have vulnerabilities that some attacks methods can take advantage of. Attempts against functions or data considered as essential or immaterial of the evaluated system. EBIOS uses a matrix to represent the mapping between entities and essential elements of the system. The mapping is used to confront threats and achieve risk assessment goals.

Under this research, it was possible to know that assets have wide range of categories. In general, information and service assets could be identified, but it depends on the analysis point of view. Knowing that the general architecture of a web application includes three layers (data, logic, and presentation), these layers cannot be considered in isolation. The application type and the content being managed determine the setting and the importance of each of these layers. Therefore, the following are essential aspects of web application development: Technical configuration, Software, Hardware, Human resource involved with the application, Information to be managed, Organizational structure, and Network infrastructure. All of these aspects define the structural design of the web application, therefore this structure should be considered because it represents the pattern of asset distribution in this type of application.

3-4 ORGANIZATIONAL ENVIRONMENT

An organization is a system with a formal structure designed to support resources such as financial, humans, technological, information, among others. These resources are integrated in an organized way and regulated by norms or standard practices to help accomplish the organization’s goals. Furthermore, Organizations do not exist in isolation; they work with the overall environment.

The term Organizational Environment refers to the forces that can make an impact. Forces can change over time and made up of opportunities and threats [3].

There are different types of organizations. For instance, Universities are considered an especial type of organizations. They are characterized by their particular configuration, which is called professional bureaucracy. According to Mintzberg H., in this configuration the fundamental coordination mechanism is the normalization of capabilities through the acquired knowledge by professional people in their training phase [25]. Universities are characterized by being coordinated by their own professional people, sometime there are cases in which professors are designated as administrative or direction staff.

Nowadays, Universities are facing administrative (bureaucracy) problems and are affected by external environment factors. This way, only the staff has the opportunity to react to changes instead of producing the changes in their environment. Many authors stated that this practice characterizes decision-making processes to be more complex in Universities than business organizations because of the features mentioned above. These statements indicate the complexity of the organizational environment in Universities.

4- RESEARCH METHODOLOGY

The research methodology selected for this work is base on formal foundations applied in the methodology, given this research specific characteristics.

Methodological Focus: The focus is both qualitative and quantitative due to the complex processes considered in the investigation whose resolution is more appropriate considering both focuses [26,27].

Investigation Method: The method is a case study carried out at the Simón Bolívar University, specifically for its Student Opinion Survey Coordination web-based application. The case study methodology is valid when questions such as “how” or “why” need to be answered, when the researcher has little control over the events, and when the topic is contemporary. All three qualifiers apply to this work [26].

The case study was performed based on Shaw [28], taking into account the following objectives: 1) investigate current situation (case study); 2) analyze (integral way) collected data; 3) conceptualize; and 4) show general conclusions and research implications. Fig. 5 shows how the case study was performed.

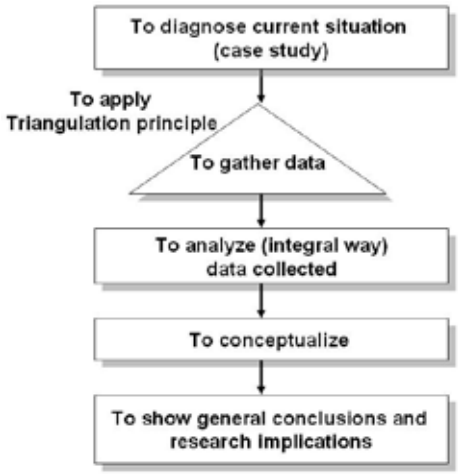

Figure 5 Case study phases.

It is important to note that a gathering data process was needed between first two phases and the triangulation principle was applied. The triangulation principle deals with the process to verify data collected from different sources (direct observation, survey, information resource, among others), allowing to achieve research internal validity [28].

Investigation Modality: This research corresponds to feasible project modality because it proposes a methodological tool to carry out asset identification in web applications. Asset identification is required to achieve successful risk assessment.

Investigation Type: This work was supported by two investigation types: documental investigation, which is characterized by information search, document analysis, and theoretical framework revision. The second type is field investigation that allows direct contacts with different groups of participants in the study area, enriches each conducted observation, and strengthens the knowledge of the topic in study. Table 2 shows the investigation specific objectives, the study variables, the variables dimensions, the indicators for those defined dimensions, sub-indicators for added levels of specificity to the investigation, and the questions taking into account in the design of the surveys (number of questions per sub-indicator).

Table 2 Variables System and indicators.

Objectives	Variables	Dimensions	Indicators	Sub-indicators	Question #
1. to diagnose the current situation with regard to the methodologies and norms of good practices related with the Risk Assessment in web applications. 2. to design the assets assessment methodological tool for Risk Assessment in web applications. 3. to evaluate the methodological tool developed through the case study in the Simón Bolívar University	Asset identification for risk assessment in web application	Risk assessment practices Web application assets Organizational environment	Environment	Web application configuration	1
			Software	Web application development software Authentication software	1,2,3,4,5,6,7,8,9
			Hardware	Server Clients	1,2,3,4
			Human resource	Clients Participants	1,2
			Information	Managed data web application Authentication data	1,2,3,4
			Organization	Entities related to web application	1,2,3,4,5,6
			Network	Network infrastructure	1,2

Data Collection Techniques: Direct Observations and a Survey were utilized to gather data from groups of participants. These techniques allowed the investigators to interact with the organization's human resource so that important opinions and feedback are obtained. The survey technique was developed through a group of questionnaires, which included opening questions set according to different type of participants. The questionnaires were elaborated in order to gather relevant information. The question set was designed taking into account the case study questionnaires proposed by the EBIOS method assistance software (version 2.0) [29] and the most relevant aspects contemplated in the theoretical framework. Each question took into consideration each of the identified sub-indicators. The question set was designed to gather information to answer each specific indicator.

Moreover, concurrent type validity was considered in this investigation. It consists of a comparison between the measure of the investigation and another standard measure to which the validity is known. The EBIOS case study questionnaire was the standard measure. In addition, the Ellipse method [22] was utilized to visualize the precision of the different sub-processes involved in the studied web application.

Data Analysis: The data analysis process required cross-case analysis. For this type of analysis, A. Huberman and M. Miles affirm that "it allows a quick analysis to observe the lines crossings and columns to have a general vision that makes it possible to compare the information of all the cases and to identify patterns, topics, similar and divergent aspects" [27]. The cross-case analysis was selected because it is more adaptable for the type of results this research gathers.

Cross-case analysis allows obtaining code matrices to compare the results obtained in each of the applied questionnaires. Each sub-indicator, shown in Table 2, has a code matrix. A code matrix helps compare results between each interviewed group of participants, especially when dealing with complex qualitative results.

An exhaustive search for software tools to support qualitative analysis was carried out. Taking into account the specific characteristics of this research, the problem statement and the investigation objectives, the variables system and indicators, MAXQDA2007 software was selected for this project. The software is specially designed to support qualitative analysis. It is available in multi-languages (Spanish, German, and English), allows creating documents in RTF format, and is useful to analyze textual data to develop theories and proofs.

5- ASSET IDENTIFICATION METHODOLOGICAL TOOL

As the first step, asset identification is an essential phase in risk assessment practices. This phase represents a degree of complexity and is the primary activity in the assessment process. Under this project, a methodological tool (instrument) was developed to help identify assets with security risks in web applications. The overall objective of the developed tool is to generate asset identifications that serve as input into the risk assessment process for web applications. The specific objectives of the tool are to: 1) study the web application environment, 2) build the variables system and indicators for study and analysis, and 3) design the necessary methodological instruments taking into account the study indicators [30].

Application Environment: Studying the web application environment is essential first step for asset identification. It is important that all elements involved in the organization environment be identified to understand the business processes in depth. This step requires different activities such as document revision and gathering organizational information. The outcome of this step is creating a context study. In this project, the Ellipse method was utilized to complement this step, allowing the investigators to visualize different sub-processes involved in the web application.

Variables System and Indicators: For this work, the Student Opinion Survey Coordination informative resource [31] was used. It is based on the CATOWE model (C: Customers, A: Actors, T: Transformation process, O: Owners, W: World view, E: Environment). The variables designed in the previous stage were used

as a guide, but sub-indicators were necessary to identify for this case study, specifically information and organization indicators.

Once the variables are completed and adapted to the case study, it is necessary to develop an evaluation matrix that consists of Interviewed Participants (to identify participating individuals to answer the questionnaires, whom belong to the organization and human resource sub-indicators) and the Organizational Indicators (to show asset categories: (E) Environment, (S) Software, (H) Hardware, (HR) Human Resource, (I) Information, (O) Organization, and (N) Network).

The matrix helped determine how the questionnaires are formed. According to which person or entity function of the organization, and whether one or several indicators are addressed. This way, N number of crossing marks is obtained for each group of participants to indicate which questions will be included in their questionnaire. Table 3 shows the resulted evaluation matrix for this project. It consists of 5 questionnaires and the indicators addressed in each questionnaire.

Table 3 Evaluation matrix.

Interviewed Participants	Organizational Indicators							Questionnaire #
	E	S	H	HR	I	O	N	
Student Opinion Survey Coordination				X	X	X		1
Information Engineering Department	X	X	X	X	X	X		2
Admission and Studies Control Department					X	X		3
Telematics Services Department						X	X	4
Human resource Indicator								
Student Opinion Survey Coordination Council Advisor (members)							X	5

The application of this tool (and its questionnaires) was guided and supervised to facilitate appropriate help to the participants. It was important to receive support and necessary resources to facilitate the interview process and help participants complete the study questionnaires.

Analysis of Results: An analysis procedure was established for this project. In this procedure, establishing inferences was necessary, and it was possible taking into account the relationships among the studied variables, dimensions, indicators, and sub-indicators, to generate conclusions. The first step in the analysis procedure was the cross-case analysis to establish the code matrix to compare the results obtained from each questionnaire. The MAXQDA2007 software was utilized to carry out operations such as text marking, text codification, categories relating, and relationships visualization. The software allowed the investigators to visualize the text code matrix for better interpretation of the results.

As a result of this methodological tool, a summary table of identified assets was compiled (Table 4). The table was presented to the Information Engineering Department. The table shows assets related to the Student Opinion Survey Coordination web application considered in this study for security risk assessment.

With the help of this tool, it was possible to determine the appropriate design for the questionnaires to help identify most critical assets in the web application. Due to the research complexity, a subjectivity grade should be accepted in the results. In addition, the results obtained from this tool indicate that there are restrictions in the University that affect the way web applications are evaluated, including technical, financial, and organizational restrictions. These restrictions are the reasons to evaluate and reflect on strategies to reduce negative impact in the organizational environment. Also, internal and external restrictions were determined and it is necessary that the organization analyzes and reflects on them as well

Table 4 Identified Assets.

Identified Assets	Indicator
1. Web application configuration elements: Application Server, Database Server, Authentication Service.	environment
2. Web application: Administrative module, Query module, Opinion module, and LDAP profiles.	software
3. Equipments: web application auxiliary equipment.	hardware
4. Data managed by the web application: Student auto-evaluation module, professor perform module, USBID, profiles, and subject information.	information
5. Network infrastructure: Queries, reports or graphs of surveys.	Network

6- CONCLUSION

This work contributes to Information Security Management and Web Engineering, specifically to security risk assessment for web applications. Asset identification is crucial phase in risk assessment practices. This work led to the development of a methodological tool for asset identification for web applications. The work is base on the EBIOS method and the MAGERIT methodology. The project was carried out through a case study of the Simón Bolívar University (Venezuela) web application “Student Opinion Survey Coordination”.

To establish the ground work for this project, a preliminary study to investigate software security vulnerabilities and mitigation strategies was conducted. The

goal was to better understand security risks and risk assessment relevant to web applications. In particular, the common classes of vulnerabilities and mitigation strategies to help alleviate such vulnerabilities.

The results obtained through the developed asset identification tool revealed the University restrictions that affect the way web applications are developed and implemented, including technical, financial, and organizational restrictions. These restrictions are the reasons to evaluate and reflect on the University strategies to reduce negative impact in the organizational environment. Internal and external restrictions were also determined.

Interaction problems among organizational levels, such as: strategic, tactical, and operative were found. These problems indicate that studies in this research field are required to guarantee an effective organizational environment or at least strategies to improve it. Another outcome was creating awareness about web application development and security attributes that should be incorporated in application development from a functional perspective. There are security risks and it is impossible to completely eliminate all risks, but it is necessary to take them into account. Finally, it is important to integrate security into the development life cycle as it leads to better quality and secure software.

Future work will address asset assessment, the next step in the risk assessment process. A methodological tool for risk assessment of web Application assets will help organizations achieve appropriate level of security risk assessment. Furthermore, a set of specific tests for each asset, based on standards or certified recommendations, will be established. Such tests help determine the real state of each asset for possible incidents, vulnerabilities, and threats. Future efforts will contribute to the design of strategies to establish effective information security management processes in the University.

REFERENCES

[1] T. Powell, D. Jones, and D. Cutts, “Web Site Engineering: Beyond Web Page Design”, Prentice Hall PTR, pp. 2-24, 1998.

[2] L. Sena, Andy S. Tenzer, “Introducción al riesgo Informático”. Facultad de Ciencias Económicas y de Administración. Universidad de la República de Montevideo, Uruguay, pp.16-17, 2004.

[3] R. Miles, C. Snow, and J. Pfeffer. “Organizational environment: concepts and issues”. Industrial Relations, pp. 244-264,1974.

[4] B.D. Romero and H.M. Haddad, “Security Vulnerabilities and Mitigation Strategies for Application Development.” Proceedings of the IEEE International Conference on Information Technology: New Generations (ITNG 2009), Las Vegas, Nevada, pp. 235-240.

[5] J. Mirkovic, “A taxonomy of DDoS attack and DDoS defense mechanisms”. ACM SIGCOMM Computer Communication Review, pp. 39-53, 2004.

[6] A. Van Der Merwe, M. Loock, and M. Dabrowski, “Characteristics and responsibilities involved in a Phishing attack”. Proceedings of the 4th international symposium on Information and communication technologies, pp. 249-254, 2005.

[7] M. Bishop, “Computer Security: Art and Science”, Addison-Wesley, section 18.2, 2003.

[8] CGISecurity. “The Cross Site Scripting FAQ”. www.cgisecurity.com/articles/xss-faq.shtml, 2008.

[9] J. Xu, Z. Kalbarczyk, and R.K. Lyer. “Transparent runtime randomization for security”, Proceedings 22nd International Symposium on Reliable Distributed Systems, pp. 260-269, 2003.

[10] G. Hoglund, and G. McGraw. “Exploiting Software: How to Break Code”. Pearson Higher Education, 2004.

[11] M. Howard, and D. LeBlanc, “Writing Secure Code”, Microsoft Press Redmond, Wash, 2003.

[12] J. Tollerson and H.M. Haddad, “Conceptual Model for Integration of COTS Components.” Proceedings of the International Conference on Software Engineering Research and Practice (SERP 2006), Las Vegas, Nevada, pp. 613-619.

[13] M. Gasser. “Building a secure computer system”. Van Nostrand Reinhold, pp. 3-6, 1998.

[14] N. Ratha, J. Connell, and R. Bolle, “Enhancing security and privacy in biometrics-based authentication systems”. IBM Systems Journal, pp. 614-634, 2001.

[15] CYBSEC Security Systems, Seminario de Seguridad en el Desarrollo del software, Buenos Aires, Argentina, Agosto 2007.

[16] R, Grimm, and B. Bershad, “Security for extensible systems”, The 6th

Workshop on Hot Topics in Operating Systems (HotOS-VI), pp. 62-66, 1997.

[17] G. Stoneburner, A. Goguen, and A. Feringa, “Computer Security”, Risk Management Guide for Information Technology Systems. National Institute of Standards and Technology Systems. Falls Church, VA, USA, pp. 8-26, 2002.

[18] A. Apvrille, and M. Pourzandi, “Software Development, Secure Software - Development by Example”, IEEE Computer Society - IEEE Security & Privacy. p.10, 2005.

[19] P. Gutmann, and I. Grigg, “Security Usability”, CryptoCorner, IEEE Security and Privacy, 2005.

[20] R. Menéndez and A. Barzanallana, “Gestión de Riesgos en Ingeniería de Software” Universidad de Murcia, 2005.

[21] J. Ramió, “Libro Electrónico de Seguridad Informática y Criptografía” Versión 4.0, Universidad Politécnica de Madrid, http://www.criptored.upm.es/guiateoria/gt_m001a.htm, 2005.

[22] A. Gómez, “Análisis y Evaluación del Riesgo de Información: Un Caso en la Banca”, Centro de Negocios Pontificia Universidad Católica del Perú (CENTRUM), pp. 20-21, 2006.

[23] Ministerio de Administraciones Públicas de España ”Libro I: Método. MAGERIT. Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información”, versión 2, http://www.csi.map.es/csi/pdf/magerit_v2/metodo_v11_final.pdf 07.15.2007, 2006.

[24] DCSSI Dirección Central de Seguridad de Sistemas de Información. “Compendio – EBIOS”, http://www.ssi.gouv.fr/es/confianza/documents/methods/ebiosv2-memento-2004-02-04_es.pdf 06.11.2007, 2007.

[25] H. Mintzberg, “La estructura de las organizaciones”. Editorial Ariel, 1979.

[26] R. Yin, “Case Study Research: Design and Methods”. Sage Publications, Thousand Oaks, CA, 1994.

[27] A. Huberman, and M. Miles, “Qualitative Data Analysis”. 2nd edition, Thousand Oaks, CA: Sage Publications, pp. 35-38, 1994.

[28] E. Shaw, “A guide to the Qualitative Research Process: Evidence from a Small Firm Study”. Qualitative Market Research: An International Journal,

pp. 59-70, 1999.

[29] Open Source Software. EBIOS. Installation for Windows (ver. 2.0). http://www.ssi.gouv.fr/IMG/zip/ebios-v2-win32-2005-06-07.zip, 2007.

[30] B.D. Romero, H.M. Haddad, and J.E. Molero, “A Methodological Tool for Asset Identification in Web Applications: Security Risk Assessment.” Proceedings of the IEEE International Conference on Software Engineering Advances (ICSEA 2009), Porto, Portugal, pp. 413-418.

[31] Coordinación Encuesta de Opinión Estudiantil (EOE), “Reseña de la Coordinación EOE”. Material Informativo Universidad Simón Bolívar, 2007.