Thesis presented to the Faculty of the Department of Graduate Studies of the Aeronautics Institute of Technology, in partial fulfillment of the requirements for the Degree of Master in Science in the Program of Electronic Engineering and Computer Science, Field Computer Science.

Gabriel Negreira Barbosa

A CMS-BASED TOOL FOR CONTINUOUS AND COLLABORATIVE RISK MANAGEMENT PROCESS

Thesis approved in its final version by the signatories below:

[illegible]

Prof. Dr. Edgar Toshiro Yano

Advisor

Prof. Dr. Celso Massaki Hirata

Head of the Faculty of the Department of Graduate Studies

Campo Montenegro

São José dos Campos, SP - Brazil

2009

Cataloging-in-Publication Data

Documentation and Information Division

Barbosa, Gabriel Negreira
A CMS-Based Tool for Continuous and Collaborative Risk Management Process / Gabriel Negreira Barbosa.
São José dos Campos, 2009.
146f.

Thesis of master in science – Course of Electronic Engineering and Computer Science. Field of Computer Science – Aeronautical Institute of Technology, 2009. Advisor: Prof. Dr. Edgar Toshiro Yano.

1. IT security. 2. IT security risk management process. 3. Collaborative software. I. General Command for Aerospace Technology. Aeronautics Institute of Technology. Division of Computer Science. II. Title

BIBLIOGRAPHIC REFERENCE

BARBOSA, Gabriel Negreira. **A CMS-Based Tool for Continuous and Collaborative Risk Management Process**. 2009. 146f. Thesis of Master of Sciences in Computer Science – Aeronautics Institute of Technology, São José dos Campos.

CESSION OF RIGHTS

Gabriel Negreira Barbosa
Rua Dom Gabriel Paulino Bueno do Couto, 30 - Casa da Frente, Parque Santa Rita
CEP 12227-260 – São José dos Campos–SP

A CMS-BASED TOOL FOR CONTINUOUS AND COLLABORATIVE RISK MANAGEMENT PROCESS

Gabriel Negreira Barbosa

ITA

This thesis is dedicated to the memory of my grandmothers. Thank you very much for everything!

Esta tese é dedicada à memória de minhas avós. Muito obrigado por tudo!

Teresa Navarro Garcia
⋆ 01/03/1927 – † 17/05/2006

Maria de Lourdes Braga Barboza
⋆ 21/07/1925 – † 22/07/2005

Acknowledgments

First of all, I wish to thank God for the inspiration and strength to accomplish this work.

I would like to thank my advisor, Professor Edgar Toshiro Yano, for the confidence, support and encouragement.

I wish to thank my undergraduate professors Paulino Ng and Ítalo Santiago Vega for the confidence and motivation.

Special thanks to my parents, José Luiz Braga Barbosa and Teresa Negreira Navarro Barbosa, for all the support and encouragement. They always motivated me in the studies; without it, this achievement would be impossible.

Special thanks also to my brother Luis Guilherme Negreira Barbosa for the friendship.

Thanks to all my relatives for the encouragement.

I would like to thank all fellows that, in some way, contributed to this work. In special to the following friends: Nelson Alves Pinto and Rodrigo Rubira Branco, that helped me so much since the beginning of the master; Otávio Freitas Ferreira Filho, for the software engineering tips; Marco Antonio Moreira de Carvalho and Mario Tadashi Shimanuki, for the help with LaTeX; Adalberto Dias Costa, that organized my thesis presentation preview, and the other crew, for the valuable tips; and the friends who lived with me (Alberto Messias da Costa Souza, João Carlos Vilela de Castro, Nelson Alves Pinto and Rogério Marinke), for the patience and motivation.

I would also like to thank all the ITA staff for the support.

Many thanks to all of you!!!

"The good news about computers is that they do what you tell them to do. The bad news is that they do what you tell them to do."
— Theodor Holm Nelson

Resumo

Um Processo de Gerenciamento de Riscos de Segurança é um importante elemento para iniciativas de segurança relacionadas a *software*. Para ser efetivo, este processo precisa tratar de aspectos sensíveis, como a análise de diversos riscos e suas contra-medidas, a constante interação entre os profissionais envolvidos, e o nível elevado de criticidade das informações envolvidas que requerem controles de acesso robustos. Nesta pesquisa, foi desenvolvida uma solução para gerenciamento de riscos de segurança, baseada num CMS (*Content Management System*) *open-source*. Tal ferramenta atende requisitos para um gerenciamento de riscos seguro, contínuo e colaborativo, devido à relevância de tais aspectos para atividades de segurança e à carência de soluções maduras nestes aspectos.

Abstract

A Security Risk Management Process is an important part for any approach to software security. To be effective, this process should address some issues, like the analysis of different risks and related safeguards, continuous collaboration among involved professionals, and a robust access control mechanism due to the sensitivity of the involved information. In this research, it was developed a solution for security risk management based on an open-source CMS (Content Management System). This tool attends requirements for a secure, continuous and collaborative risk management, due to the importance and the lack of mature solutions in those aspects.

List of Figures

List of Tables

Contents

1 Introduction

Organizations are increasingly more dependent on software systems to achieve their main business goals. Some systems can be interconnected with other ones providing, for example, services to partners. It follows that an attacked system can compromise a company's core business and put at risk all of its interconnected systems. [2]

Due to the importance of IT (Information Technology) systems to the world, this field has a high development rate. It follows that new technologies are often developed, resulting in a growing complexity of IT environments.

As shown in Figure 1.1, over the past ten years the number of incidents reported to CERT.br [1] has grown significantly.

With an increasing business reliance on software systems, a growing complexity of IT systems, and a constantly evolving security threats, software security has become a critical business function.

An effective IT security risk management process is an important component of a successful approach to software security. The risk management-based approach to security seeks a balance between risk avoidance, risk mitigation and risk acceptance, by prioritizing risks based on their likelihoods and potential impacts. Risks that are both likely to occur and expensive to recover from, are tagged as high-level risks and are candidates either to be mitigated or avoided. Risks that are highly unlikely or extremely cheap to recover from, are classified as low-level risks and are reasonable candidates for risk acceptance. [4]

In spite of the importance of an IT security risk management process for organizations' business, in practice just a few companies implement such effort. Among the probable reasons are the lack of prioritization for security, and the high cost of specialized software, people and services.

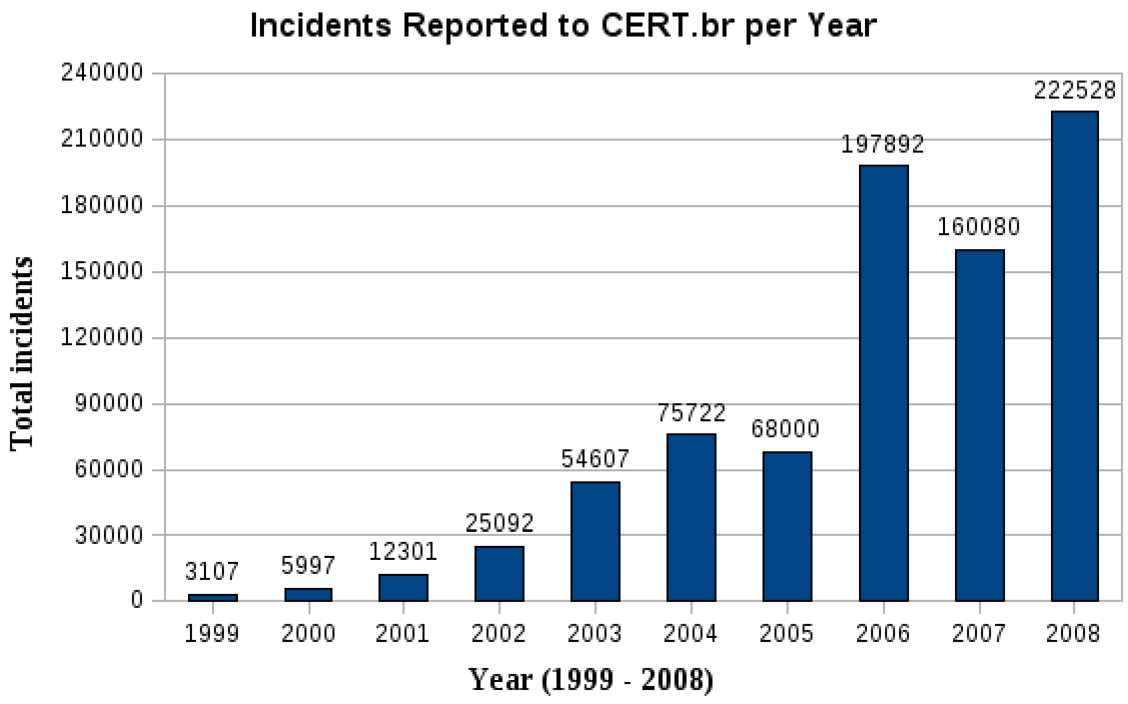


FIGURE 1.1 – Security incidents reported to CERT.br over the last ten years (based on a chart from [1]).

1.1 Problem Definition

An IT security risk management effort should be handled by organizations as a continuous and collaborative work, with a proper access control mechanism for the information involved.

It must be a continuous effort because new vulnerabilities are often discovered, hardware and software technologies are in constant evolution, and the business environment is frequently changing during the SDLC (System Development Life Cycle).

It must be a collaborative work because its activities require collaboration of many professionals, from different parts of the organization, for purposes like monitoring, analyzing, and approving technical or business issues. Nowadays, the collaboration need is often addressed by standard meetings with related professionals.

It requires a proper access control mechanism because the information involved is highly sensitive with details, such as critical vulnerabilities and adopted mitigation actions, that can lead attackers to compromise the core business. The access control mechanism must support a dynamic business environment where participants are frequently changing positions or even going to other companies.

Additionally, due to high amount of information to handle, a specialized software can help an IT security risk management effort.

This research identified many tools to support an IT security risk management process [5, 6, 7, 8], all of them are proprietary systems. Additionally, none of them has all the following characteristics:

- Specific environment for a continuous IT security risk management process;
- Support for an effective collaborative environment for continuous group discussions and artifacts creation, fully integrated with the IT risk management process; and
- Robust access control to protect the information in a dynamic environment, with a high turnover of professionals.

1.2 Objectives

The goal of this research is to develop a secure tool to support an IT security risk management process, in a continuous (ongoing work) and collaborative (professionals freely communicating with each other) way, with a robust access control mechanism (strictly access restrictions compatible with dynamic environments). This tool is intended to be used by organizations to structure their own IT security data, improve business decisions based on security information, and keep the security level always acceptable.

Such a tool would improve the way that the IT security risk management process is conducted, because it provides a risk management environment with features to conduct secure group communications directly linked with the security data, that are kept structured and protected by a robust access control mechanism.

1.3 Work Organization

This work is organized as follows. In Chapter 2, the concepts behind the research are discussed, and the state of the art established through the analysis of some related work. Chapter 3 identifies some requirements for a tool supporting IT security risk management, presents the proposed solution and its conceptual model, and gives an overview of the implementation. Chapter 4 explains the software's implementation, focusing on the adopted software engineering decisions. Chapter 5 evaluates the developed application through a case study. Finally, Chapter 6 presents the final considerations of this research, discussing its compliance with the established objective, analyzing the contributions, and listing recommendations for future work.

2 Review of the Literature

This chapter introduces the concepts behind the developed research. Section 2.1 describes the basic concepts of computer security. Section 2.2 addresses the basic theory of IT security risk management process, and Section 2.3 details the NIST approach [3] for such a process. However, there are other IT security risk management process efforts, and some of them (SEI's OCTAVE Method [9] and Microsoft [10]) are overviewed in Section 2.4. CMS (Content Management System) concepts are briefly discussed in Section 2.5. Some related work is described and compared to the developed research in Section 2.6.

2.1 Basic Concepts of Computer Security

Computer Security is defined by Matt Bishop [11] as an aggregation of three components:

- Confidentiality: This is the concealment of information or resources that for some reason cannot be accessed publicly. Access control mechanisms, such as cryptography and passwords, support the confidentiality component;
- Integrity: This component includes both the correctness and the trustworthiness of data or resources. It includes data integrity and origin integrity (often called

authorization). There are two classes of integrity mechanisms: prevention and detection. The former strives to deny any unauthorized changes or modifications in unauthorized ways. The latter seeks to detect integrity violations and may report their causes. Examples of integrity mechanisms are digital signatures, hash values and access control systems (login/password, for example); and

- Availability: This refers to the ability to use information or resources. Attacks against availability, called Denial of Service (DoS), make information or resources unavailable.

NIST [12] identified eight elements of computer security:

- Computer security should support the mission of the organization;
- Computer security is an integral element of sound management;
- Computer security should be cost-effective;
- Computer security responsibilities and accountability should be made explicit;
- System owners have computer security responsibilities outside their own organizations;
- Computer security requires a comprehensive and integrated approach;
- Computer security should be periodically reassessed; and
- Computer security is constrained by societal factors.

Security analysis is a process to examine a system in order to identify security flaws and propose measures to correct the identified problems. Once flaws have been discovered, a security management process is needed to handle their correction strategy and keep the company's security level acceptable. There is a structured approach for security analysis and management processes, discussed in details in the next section, called IT Security Risk Management Process. However, it always has to be assumed that "computers cannot ever be fully secured" [12].

2.2 IT Security Risk Management Process

With the notable development of technology, companies are adopting computer systems to support and optimize their business processes. As more computer systems are used, more potential points to fail exist. It follows that a computer security analysis process in a company with many computer systems is not a simple task and may result in a high number of flaws. Thus the management of flaws and their mitigations is a complex task. There is a process to address this problem, focused on risks, called IT security risk management (or computer security risk management).

The Longman dictionary [13] defines risk as "the possibility that something bad, unpleasant, or dangerous may happen". NIST [12] defines risk management as "the process of assessing risk, taking steps to reduce risk to an acceptable level and maintaining that level of risk". When the risks arise from an organization's use of IT, the process is called computer security risk management [12] (or IT security risk management).

IT security risk management is a critical process in every organization with security concerns because it is a structured way to achieve and maintain an acceptable security risk level. Due to its high level of importance, many organizations around the world, of the most different natures, have developed their own approaches to handle this demand.

For example, three well known security risk management processes are SEI's OCTAVE Method [9], NIST [3] and Microsoft [10]. In spite of the existence of many models, there are some activities and processes that should be performed [12]. This section describes the basic concepts and aspects that are commonly present behind computer security risk management approaches.

A general process of managing risks has three basic activities according to NIST [12], two primary, risk assessment and risk mitigation, and one underlying, uncertainty analysis. These activities are described in the next three sections, respectively. Section 2.2.4 discusses the importance of applying a risk management process from the early stages of an SDLC.

2.2.1 Risk Assessment

Risk assessment, the process of analyzing and interpreting risk, involves three basic activities: (1) determining the assessment's scope and methodology; (2) collecting and analyzing data; and (3) interpreting the risk analysis results. The first is responsible for identifying the target system, determining the scope and boundary of the analysis, and selecting a methodology that best fits the requirements. The selection of those items can improve the cost-effectiveness of the analysis because the study of most important and critical systems for the company's business can be prioritized and the usefulness of the results can be improved. [12]

The second risk assessment activity is responsible for obtaining and analyzing data. It is possible to get more data than can be analyzed, so there is a need for a process to refine the information. This process is called screening, and it should help the risk management process to focus on the most important and sensitive areas for the organization. [12] The collected data should be structured against some concepts. There is confusion in the literature related to some of these concepts in that there are authors who do not adopt the same meanings [14]. This research adopted some definitions and they are used consistently throughout this work even if the source of information uses others (in this case, the source's definitions are implicitly converted to the adopted ones). The following concepts are the ones adopted by this survey:

- Asset: Anything that has a value to the organization, whose protection is desired; [15]
- Vulnerability: A flaw or weakness that can be exercised (accidentally triggered or intentionally exploited); [3]
- Threat: A potential violation of security, producing damage to one or more assets; [11, 16]
- Threat-source: An entity or event with the potential to harm a system (materialize a threat), accidentally or not, through a vulnerability. It is also known as threat agent; [12, 3, 17]
- Risk: A function of the likelihood of a given threat-source's exercising a particular vulnerability, and the resulting impact of the threat on the organization; [3] and
- Safeguards: Protective measures for risks prescribed to meet the security requirements (i.e., confidentiality, integrity, and availability). Synonymous with security controls and countermeasures. [17, 16, 12] This research also references safeguards as controls.

With the structured data, some analyses are important:

- Likelihood assessment: Assessment of the estimated frequency or chance of a threat happening. The effectiveness of the existent safeguards and the identified vulnerabilities should be used in the classification; [12]
- Consequence assessment: Assessment of the impact of the identified threats for the organization; [12]
- Safeguard analysis: Effectiveness examination of the existing safeguards. New safeguards to be implemented could be identified, but this is normally made later in the risk management process; [12] and
- Risk analysis: Risk evaluation using the defined threat's likelihood and consequence. This analysis' result can be quantitative (with a quantity [16]) or qualitative (on a scale of levels [16]).

The third risk assessment activity, interpreting the risk analysis results, is responsible for analyzing identified risks to detect their importance for the organization. A correct interpretation can be used to prioritize systems to be secured. [12]

2.2.2 Risk Mitigation

The second main activity in a risk management process, risk mitigation, "involves the selection and implementation of security controls to reduce risk to a level acceptable to management, within applicable constraints" [12]. There are three basic functions in this activity: selecting safeguards, accept residual risks, and implementing controls and monitoring effectiveness.

The first is responsible to select cost-effective safeguards to address the identified risks according to the organization's interests. Some factors to be considered in this selection are organizational policy/legislation/regulation, system performance requirements, and life cycle costs of security measures.

The second function, accept residual risks, should decide if the security level is acceptable given the kind and severity of remaining risks. Some risks may have to be accepted because the safeguards can be too expensive, in either monetary or non-monetary factors. This function is related to the accreditation; that is the formal acceptance of the actual security level by the management to make the computer system operational or to remain so.

Finally, the last activity deals with the implementation of the selected safeguards in order to effectively reduce the risks. The risk management process, to be effective, should be an ongoing process, with periodic assessment and improvement of safeguards, and re-analysis of risks. [12] The whole risk management process should be made often, especially when new components have been added to the organizations' technological complex and significant changes have occurred in the existent scenario.

2.2.3 Uncertainty Analysis

The last activity in a risk management process, the underlying one, is called uncertainty analysis. According to NIST [12], a risk management process "often must rely on speculation, best guesses, incomplete data, and many unproven assumptions". The purpose of this activity is to document the uncertainty level of each element of the whole analysis to provide a knowledgeable future analysis of the risk management process.

2.2.4 Risk Management Process into a SDLC

Even though already implanted computer systems can be analyzed, security is more effective when planned and managed throughout a SDLC [12]. It is known that this practice is important and significantly reduces security problems in deployed software [2, 18]. Figure 2.1 shows that "the longer defects persist, the more expensive they are to correct" [2].

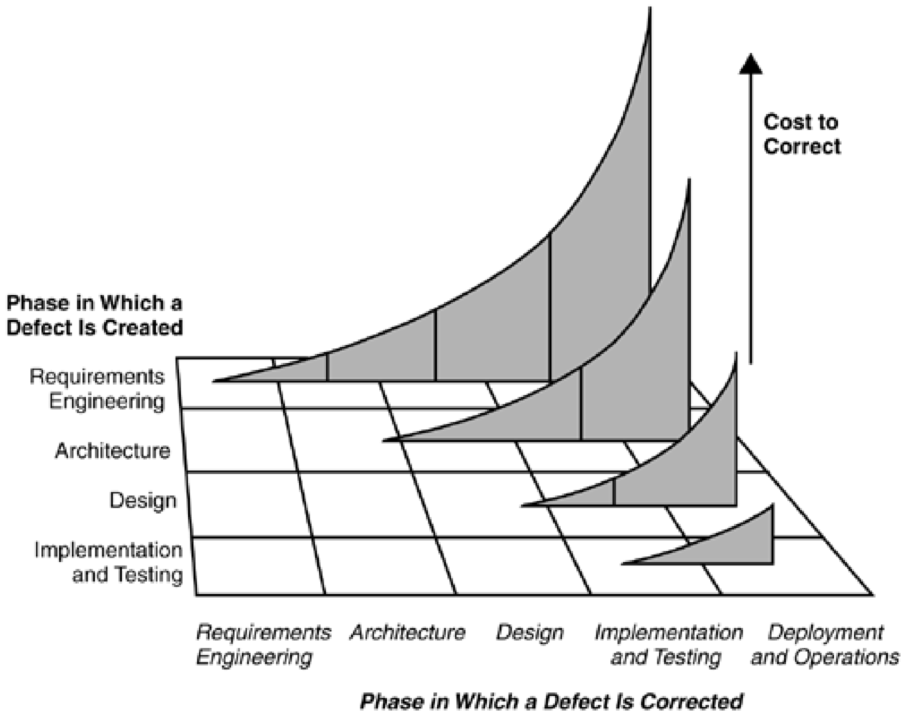


FIGURE 2.1 – Cost of correcting defects by life-cycle phase (obtained from [2]).

A risk management process should be integrated with the SDLC in order to improve its effectiveness [3, 18]. The basic idea of this integration is to analyze and address security risks as they change over the software life cycle [18].

2.3 NIST IT Security Risk Management Process

This section describes NIST's approach for an IT security risk management process according to [3]. Section 2.3.1 describes the integration of this methodology into SDLC. The NIST approach [3] encompasses three processes: risk assessment (Section 2.3.2), risk mitigation (Section 2.3.3), and evaluation and assessment (Section 2.3.4).

2.3.1 Integration into SDLC

To be effective, a risk management process must be totally integrated into the SDLC. According to NIST [3], an SDLC has five phases, and the same IT system may occupy more then one at the same time: initiation, development or acquisition, implementation, operation or maintenance, and disposal. Table 2.1 describes the SDLC phases and how the risk management fits into each of them. Another consideration is that the risk management methodology is the same for every SDLC phase [3].

TABLE 2.1 – Description of SDLC phases and the related NIST risk management process integration (obtained from [3]).

SDLC Phases	Phase Characteristics	Support from Risk Management Activities
Phase 1 - Initiation	The need for an IT system is expressed and the purpose and scope of the IT system is documented.	Identified risks are used to support the development of the system requirements, including security requirements, and a security concept of operations (strategy).
Phase 2 - Development or Acquisition	The IT system is designed, purchased, programmed, developed, or otherwise constructed.	The risks identified during this phase can be used to support the security analyses of the IT system that may lead to architecture and design trade-offs during system development.

Phase 3 - Implementation	The system security features should be configured, enabled, tested, and verified.	The risk management process supports the assessment of the system implementation against its requirements and within its modeled operational environment. Decisions regarding risks identified must be made prior to system operation.
Phase 4 - Operation or Maintenance	The system performs its functions. Typically the system is being modified on an ongoing basis through the addition of hardware and software and by changes to organizational processes, policies, and procedures.	Risk management activities are performed for periodic system reauthorization (or reaccreditation) or whenever major changes are made to an IT system in its operational, production environment (e.g., new system interfaces).
Phase 5 - Disposal	This phase may involve the disposition of information, hardware, and software. Activities may include moving, archiving, discarding, or destroying information and sanitizing the hardware and software.	Risk management activities are performed for system components that will be disposed of or replaced to ensure that the hardware and software are properly disposed of, that residual data is appropriately handled, and that system migration is conducted in a secure and systematic manner.

2.3.2 Risk Assessment

This process is basically responsible for analyzing the target system in order to identify and measure risks, and recommend safeguards. The risk assessment has nine steps, and steps 2, 3, 4, and 6 can be conducted in parallel after the completion of step 1 [3]. Table 2.2 relates the steps with their inputs, outputs, and some collaboration needs (the need for which was identified by this research).

TABLE 2.2 – Risk assessment activities and related inputs, outputs, and collaboration needs (based on [3]).

Risk Assessment Activity	**Input**	**Output**	**Collaboration Needs**
Step 1 – System Characterization	• Hardware • Software • System interfaces • Data and information • People • System mission	• System boundary • System functions • System and data criticality • System and data sensitivity	• Communication with appropriate persons to get information • Validation of gathered information
Step 2 – Threat-Source Identification	• History of system attack • Data from a source such as mass media	• List of threat-sources	• Understand the actors that interact with the target IT system
Step 3 – Vulnerability Identification	• Reports from prior risk assessments • Any audit comments • Security requirements • Security test results	• List of potential vulnerabilities	• Obtain old audit reports • Discover already undocumented known vulnerabilities
Step 4 – Control Analysis	• Current controls • Planned controls	• List of current and planned controls	• Obtain related documentation

Step 5 – Likelihood Determination	• Threat-source motivation and capacity • Nature of vulnerability • Current controls	• Likelihood rating	• Interact with company's personnel for accuracy
Step 6 – Impact Analysis	• Mission impact analysis • Asset criticality assessment • Data criticality • Data sensitivity	• Impact rating	• Communication with related personnel about attacks' impact on the target IT system
Step 7 – Risk Determination	• Likelihood of exploitation • Magnitude of impact • Adequacy of planned or current controls	• Risks and associated risk levels	• Validate identified risks
Step 8 – Control Recommendations		• Recommended controls	• Study organization's operations needs
Step 9 – Results Documentation		• Risk assessment report	• Validate the assessment

2.3.2.1 Step 1: System Characterization

The system characterization step has to define the scope and boundaries of the risk assessment effort. Then, the target system has to be studied in order to get as much information as possible to define risks. The system-related information should be collected first and is usually classified as follows according to NIST [3]:

- Hardware;
- Software;
- System interfaces;
- Data and information;
- Persons who support and use IT system;
- System mission;
- System and data criticality; and
- System and data sensitivity.

Information related to the operational environment also has to be identified, for example [3]:

- Functional requirements;
- Users of the system;
- Security policies;
- Flow of information;

- Environmental security such as humidity control, for example; and
- Personnel security.

In order to facilitate gathering information, some techniques can be used [3]:

- Questionnaire: A list of questions can be developed and distributed to appropriate persons in order to get some information. The questionnaire can be used also during on-site interviews;
- On-site interviews: The security professionals that are gathering information can visit the target organization or department in order to interview selected personnel to get needed information. During the visit, physical, operational and environmental aspects can be observed also;
- Document review: Documents regarding the target system are good sources of information about the used and planned security controls. Examples of documentation are system documents, policies and previous audit reports; and
- Automated scanning tool: Technical methods of collecting information can be used. For example, a network mapping tool can be used to identify services running on hosts.

2.3.2.2 Step 2: Threat-Source Identification

The goal of this step is to develop a list of potential threat-sources to the target system. Common threat-sources according to NIST [3] are:

- Natural threat-sources: Floods, earthquakes, tornadoes, landslides, avalanches, electrical storms, and other such events;

- Human threat-sources: Events enabled or caused by human beings (directly related to the target system, such as insiders, or not, such as external persons), both intentional (e.g., network based attacks, malicious software upload, unauthorized access to confidential information) and unintentional (e.g., inadvertent data entry); and

- Environmental threat-sources: Long-term power failures, pollution, chemicals, liquid leakage.

2.3.2.3 Step 3: Vulnerability Identification

The main objective of the vulnerability identification step is to identify and develop a list of the target system's vulnerabilities that could be exploited by threat-sources. The vulnerabilities can be associated with the related threat-sources and threats to help steps 5, 6, and 7 of the risk assessment process. This association is called vulnerability/threat pair and Table 2.3 illustrates an example.

TABLE 2.3 – NIST's vulnerability/threat pair example (based on [3]).

Vulnerability	**Threat-Source**	**Threat**
Terminated employees' system identifiers (ID) are not removed from the system.	Terminated employees	Dialing into the company's network and accessing company proprietary data.
Company firewall allows inbound telnet, and guest ID is enable on XYZ server.	Unauthorized users	Using telnet to XYZ server and browsing system files with the guest ID.
Application access the internal server through a plain connection and the network are based on hubs.	Internal malicious users	Sniffing the network and getting passwords.
		Man-in-the-middle attacks.

There are three recommended methods according to [3] for identifying vulnerabilities:

- Vulnerability sources: Information-gathering techniques described in Section 2.3.2.1 can be used to help identifying vulnerabilities. Industry sources (e.g., vendor Web pages with bugs and flaws) may have useful information for preparing interviews and developing questionnaires. Another source of information is the internet to discover flaws on known systems together with correction patches. Documented vulnerability sources should also be considered, for example, previous risk assessment documentation, audit reports, security advisories, and vulnerability databases (e.g., NIST's NVD [19], and OSVDB [20]);

- System security testing: Proactive methods can be used to identify vulnerabilities, such as automated vulnerability scanning tools, security test and evaluation (development and execution of security tests), and penetration testing; and

- Development of security requirements checklist: Development of a security requirements list that can be used to systematically evaluate the security of the target system. It is important to keep the checklist updated against the organization's environment. Examples of sources that can be used to help compiling the list are system security plan, organization's policies, and industry practices.

2.3.2.4 Step 4: Control Analysis

This step strives to identify and analyze the security controls planned or already implemented by the organization regarding the target system. According to NIST [3] there are two categories of security controls:

- Preventive controls: block attempts to violate security policies. Examples are encryption and authentication; and

- Detective controls: detect and report attempts to violate security policies. Examples are audit trails and intrusion detection methods.

The security requirements checklist, described in Section 2.3.2.3, can also be used to analyze controls.

2.3.2.5 Step 5: Likelihood Determination

The goal of the likelihood determination step is to rate the probability of a threat-source exercising a vulnerability. It means rating the likelihood of each tuple of the vulnerability/threat pair described in the vulnerability identification step (Section 2.3.2.3). According to NIST [3], there are some factors to consider:

- Threat-source motivation and capability;
- Nature of the vulnerability; and
- Existence and effectiveness of current controls.

The likelihood rates can be one of the three levels described by Table 2.4.

TABLE 2.4 – NIST's likelihood levels and their descriptions (obtained from [3]).

Likelihood Level	**Likelihood Definition**
High	The threat-source is highly motivated and sufficiently capable, and controls to prevent the vulnerability from being exercised are ineffective.
Medium	The threat-source is motivated and capable, but controls are in place that may impede successful exercise of the vulnerability.
Low	The threat-source lacks motivation or capability, or controls are in place to prevent, or at least significantly impede, the vulnerability from being exercised.

2.3.2.6 Step 6: Impact Analysis

This step is responsible for determining the magnitude of the adverse impact resulting from a threat source successfully exploring a vulnerability. It means rating the impact of each tuple of the vulnerability/threat pair (Section 2.3.2.3) considering a successful attack. To analyze the impact, the following information has to be obtained as discussed in Step 1 (Section 2.3.2.1):

- System mission;
- System and data criticality; and
- System and data sensitivity.

The recommended approach to determine the impact levels is to interview the respective system and information owners. [3]

According to NIST [3], the impact can be described in terms of loss or degradation of the three security components (confidentiality, integrity, and availability) described in Section 2.1. There are two types of impact assessment:

- Quantitative: Based on measurements (specific values) like the cost of repairing the system; and
- Qualitative: Based on a scale of values, like High, Medium, and Low.

Due to the generic aspect of the NIST risk management process methodology, the qualitative approach was adopted in the guide [3]. The three possible values for the impact level are described by Table 2.5.

TABLE 2.5 – NIST's impact levels and their descriptions (obtained from [3]).

Magnitude of Impact	**Impact Definition**
High	Exercise of the vulnerability (1) may result in the highly costly loss of major tangible assets or resources; (2) may significantly violate, harm, or impede an organization's mission, reputation, or interest; or (3) may result in human death or serious injury.
Medium	Exercise of the vulnerability (1) may result in the costly loss of tangible assets or resources; (2) may violate, harm, or impede an organization's mission, reputation, or interest; or (3) may result in human injury.
Low	Exercise of the vulnerability (1) may result in the loss of some tangible assets or resources or (2) may noticeably affect an organization's mission, reputation, or interest.

2.3.2.7 Step 7: Risk Determination

The purpose of this step is to determine the level of risk for each vulnerability/threat pair (Section 2.3.2.3). This level can be expressed as a function of the:

- Likelihood;
- Impact; and
- Adequacy of security controls (implemented or planned).

A risk scale and a risk-level matrix must be developed in order to measure risks. The risk scale describes the possible risk levels, and the risk-level matrix is used to combine the likelihood and impact values to determine the respective risk level. NIST [3] has created both artifacts based on qualitative values (high/medium/low), as expressed by Table 2.6 and Table 2.7, respectively. Other kinds of risk-level matrix can be used by some sites, for example a 4 x 4 or a 5 x 5 matrix. The latter can include "Very Low" and "Very High" values for both likelihood, impact, and thus risk level; in this case the "Very High" risk level may require actions such as shutting down the system. [3]

TABLE 2.6 – NIST's risk scale with respective descriptions and necessary actions (based on [3]).

Risk Level	**Risk Description and Necessary Actions**
High	If an observation or finding is evaluated as a high risk, there is a strong need for corrective measures. An existing system may continue to operate, but a corrective action plan must be put in place as soon as possible.
Medium	If an observation is rated as medium risk, corrective actions are needed and a plan must be developed to incorporate these actions within a reasonable period of time.
Low	If an observation is described as low risk, it must be determined whether corrective actions are still required or the risk can be accepted.

TABLE 2.7 – NIST's risk-level matrix (based on [3]).

Threat Likelihood	*Impact*		
	Low	**Medium**	**High**
High	Low	Medium	High
Medium	Low	Medium	Medium
Low	Low	Low	Low

2.3.2.8 Step 8: Control Recommendations

The main objective of this step is to provide safeguards for the identified risks. Alternative solutions should also be recommended. According to NIST [3], the following factors should be considered:

- Effectiveness of recommended options;
- Legislation and regulation;
- Organizational policy;
- Operational impact; and
- Safety and reliability.

2.3.2.9 Step 9: Results Documentation

The goal of this step is to officially document the risk assessment once it has been done. The result report is a management artifact that should be use to make decisions regarding the security of the target IT system. Unlike audit efforts which seem to detect errors, a risk assessment report should not have accusatory content, just an analytical view to help keep an acceptable level of security. That is why the vulnerability/threat pairs, for some people, are referenced as observations instead of findings. [3] Annex A has a NIST's template [3] for the risk assessment report.

2.3.3 Risk Mitigation

This is the second process of the NIST approach for risk management, and involves prioritizing, evaluating, and implementing the appropriate security controls recommended by the last process, risk assessment (Section 2.3.2). As it is close to impossible to eliminate all risks [3], it is needed to adopt a “least-cost approach and implement the most appropriate controls to decrease mission risk to an acceptable level, with minimal adverse impact on the organization's resources and mission” [3].

The next sections describe the risk mitigation options (Section 2.3.3.1), the risk mitigation strategy (Section 2.3.3.2), a control implementation approach (Section 2.3.3.3), a brief overview of control categories (Section 2.3.3.4), a cost-benefit analysis to implement the recommended controls (Section 2.3.3.5), and residual risk considerations (Section 2.3.3.6).

2.3.3.1 Risk Mitigation Options

According to [3], risk mitigation is a systematic methodology to reduce mission risk that can be achieved through any of the following options:

- Risk assumption: accept the risk or implement controls to reduce it to an acceptable level;
- Risk avoidance: avoid the risk by eliminating its cause and/or consequence;
- Risk limitation: implement controls to minimize the related impact of the risk;
- Risk planning: a risk mitigation plan is developed to prioritize, implement and maintain controls in order to manage the risk;
- Research and acknowledgment: acknowledge the vulnerability and research controls to address it; and
- Risk transference: adopt options to compensate the loss, for example purchasing insurance.

As it may be not practical to address all identified risks [3], safeguards have to be prioritized focusing on the organization's mission.

2.3.3.2 Risk Mitigation Strategy

NIST [3] developed the risk mitigation chart shown in Figure 2.2 in order to help detect when control actions must be taken. The appropriate action points are indicated by the word "YES".

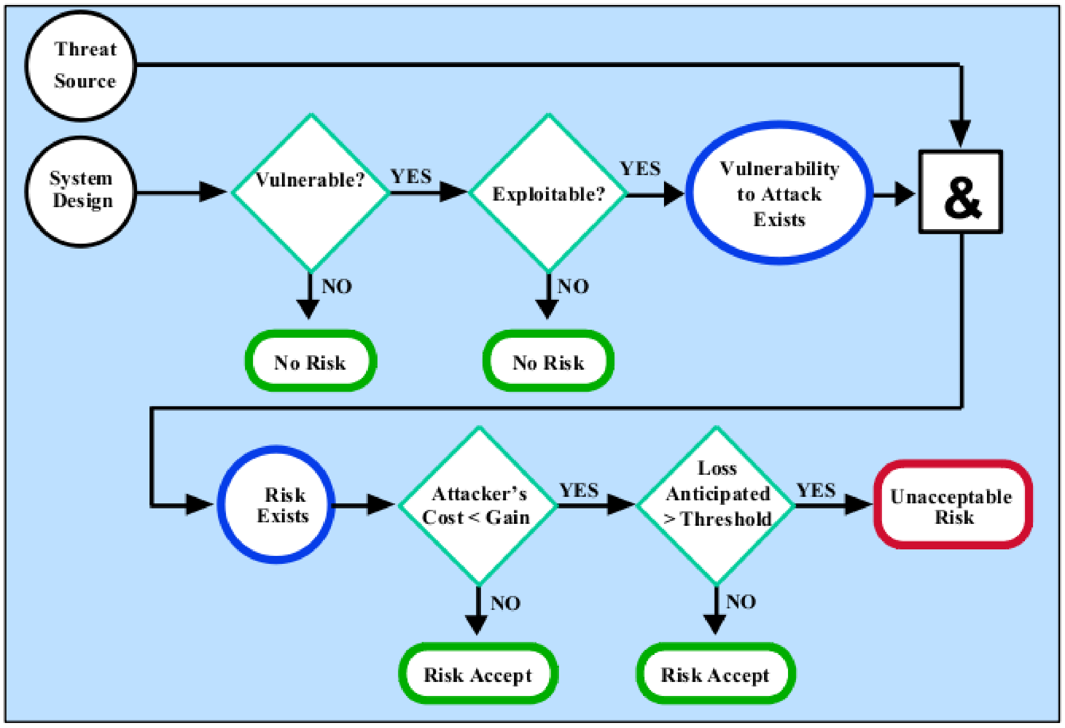


FIGURE 2.2 – NIST's risk mitigation chart (obtained from [3]).

Additionally, NIST [3] developed the following rules to guide mitigation actions for risks from intentional human threat-sources:

1. When vulnerability exists: implement assurance techniques to reduce the likelihood of a vulnerability's being exercised;

2. When a vulnerability can be exercised: apply layered protections, architectural designs, and administrative controls to minimize the risk of or prevent this occurrence;

3. When the attackers' cost is less than the potential gain: apply protections to decrease an attacker's motivation by increasing the attacker's cost (e.g., use of system controls such as limiting what a system user can access and do can significantly reduce an attacker's gain); and

4. When loss is too great: apply design principles, architectural designs, and technical

and nontechnical protections to limit the extent of the attack, thereby reducing the potential for loss.

The rules can also be applied for risks arising from environmental or human unintentional threat-sources, but in this case the third rule has to be ignored, because there is no attacker, motivation, nor gain. [3]

2.3.3.3 Approach for Control Implementation

Once control actions have to be taken, the following rule should be considered: "Address the greater risks and strive for sufficient risk mitigation at the lowest cost, with minimal impact on other mission capabilities" [3].

The NIST [3] risk mitigation methodology to implement security controls is described by the following steps:

- Step 1 - Prioritize actions: The risk levels determined by the first risk management methodology process, risk assessment (Section 2.3.2), should be analyzed in order to prioritize vulnerability/threat pairs to be addressed;

- Step 2 - Evaluate recommended control options: The feasibility and effectiveness of the controls recommended by the risk assessment process (Section 2.3.2) should be evaluated to determine the most appropriate ones;

- Step 3 - Conduct cost-benefit analysis: A cost-benefit analysis should be conducted against the safeguards recommended by the risk assessment process (Section 2.3.2), as described in Section 2.3.3.5;

- Step 4 - Select controls: Based on the cost-benefit analysis from Step 3, the cost-effective controls should be determined. It is recommended to combine technical, operational and management controls, as described in Section 2.3.3.4, to improve the security;

- Step 5 - Assign responsibility: The responsibility to implement the selected controls should be assigned to professionals with appropriate skills;
- Step 6 - Develop a safeguard implementation plan: It has to be developed a safeguard implementation plan, also known as action plan, that prioritizes control implementation actions. Annex B has a sample summary table for the safeguard implementation plan, according to NIST [3], with the minimum fields and their respective explanations; and
- Step 7 - Implement selected control(s): Some implemented controls can lower but not eliminate the risk. These kind of risks are called residual risks and are discussed in Section 2.3.3.6.

Table 2.8 summarizes the methodology steps, together with their inputs, outputs, and collaboration needs. The collaboration necessity was identified by this research.

TABLE 2.8 – Risk mitigation activities and related inputs, outputs and collaboration needs (based on [3]).

Risk Mitigation Activity	**Input**	**Output**	**Collaboration Needs**
Step 1 - Prioritize Actions	• Risk levels from the risk assessment report	• Actions ranking from High to Low	• Determination of correct levels

Step 2 - Evaluate Recommended Control Options	• Risk assessment report	• List of possible controls	• Get IT-related environment characteristics
Step 3 - Conduct Cost-Benefit Analysis		• Cost-benefit analysis	• Discover restrictions
Step 4 - Select Controls		• Selected controls	• Detect implementation restrictions
Step 5 - Assign Responsibility		• List of responsible persons	• Interact with involved departments to select appropriate persons
Step 6 - Develop Safeguard Implementation Plan		• Safeguard implementation plan	• Validation of the plan
Step 7 - Implement Selected Controls		• Residual risk	• Monitoring implementation

2.3.3.4 Control Categories

According to [3], to improve a computer system's security level, an organization should consider implementing a combination of the following three classes of controls:

1. Management controls: “The security controls (i.e., safeguards or countermeasures) for an information system that focus on the management of risk and the management of information system security”; [17]

2. Operational controls: “The security controls (i.e., safeguards or countermeasures) for an information system that primarily are implemented and executed by people (as opposed to systems)”; [17] and

3. Technical controls: “The security controls (i.e., safeguards or countermeasures) for an information system that are primarily implemented and executed by the information system through mechanisms contained in the hardware, software, or firmware components of the system”. [17]

Due to ease of use, each class contains some families, and each family contains security control types [21]. There is a list of the security controls types classified according families and classes, according to [21], summarized in Annex C.

2.3.3.5 Cost-Benefit Analysis

A cost-benefit analysis against the recommended controls should be conducted at Step 3 of the risk mitigation methodology (Section 2.3.3.3) in order to select the cost-effective controls. The goal of this analysis, that can be quantitative or qualitative, is to demonstrate that the costs of implementing selected safeguard justifies the security gains obtained. The cost-benefit analysis encompasses the following aspects according to [3]:

- Determining the impact of implementing the new or enhanced controls;

- Determining the impact of not implementing the new or enhanced controls;

- Estimating the costs of the implementation. These may include, but are not limited to, the following:
  - Hardware and software purchases;
  - Reduced operational effectiveness if system performance or functionality is reduced for increased security;
  - Cost of implementing additional policies and procedures;
  - Cost of hiring additional personnel to implement proposed policies, procedures, or services;
  - Training costs; and
  - Maintenance costs.
- Assessing the implementation costs and benefits against system and data criticality to determine the importance to the organization of implementing the new controls, given their costs and relative impact.

The acceptable level of risk for the mission must be defined to allow the assessment of the impact of control and the safeguard's inclusion or exclusion decision. The following rules, obtained from [3], can be used to determine the use of new controls:

- If control would reduce risk more than needed, then see whether a less expensive alternative exists;
- If control would cost more than the risk reduction provided, then find something else;
- If control does not reduce risk sufficiently, then look for more controls or a different control; and

- If control provides enough risk reduction and is cost-effective, then use it.

2.3.3.6 Residual Risk Considerations

The risk reduction can be measured in terms of likelihood and impact, the two parameters that constitutes the risk level. According to [3], risks can be mitigated by the following:

- Eliminating some of the system's vulnerabilities;
- Adding targeted control to reduce the capacity and motivation of a threat-source; and
- Reducing the magnitude of the impact.

The risk that remains after the implementation of safeguards is called residual risk. According to [3], it is difficult to have a risk free IT system, and not all security controls can definitely eliminate the related risk. The residual risks should be analyzed in order to keep the computer system operational, or halt it. The process of formally authorizing the system to operate is known as accreditation or authorization.

2.3.4 Evaluation and Assessment

Due to the continuous changes of computer systems and personnel in some organizations, risks are constantly changing. Thus, risk management is an ongoing and evolving process. The risk assessment and mitigation processes should both be executed periodically and when needed due to some technological change, for example. [3]

The success of the risk management program, according to NIST [3], relies on:

- Management commitment;
- Full support and participation of the IT team;
- Competence of the risk assessment team;
- Members of the user community, who must follow established procedures and cooperate for the company's security; and
- An ongoing evaluation and assessment of the IT-related mission risks.

2.4 Other Risk Management Approaches

As stated before, there are approaches for risk management process from groups of the most different natures. In spite of their difference in comparison to NIST's approach, all of them strive to detect and structure the security concepts described in Section 2.2.1. The SEI's OCTAVE Method [9] and Microsoft risk management approach [10] are briefly described by Sections 2.4.1 and 2.4.2, respectively.

2.4.1 SEI's OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) Method

According to [9], this methodology was developed for larger organizations (300 employees or more [22]) and encompasses three phases:

- Phase 1 - Build Asset-Based Threat Profiles: The first phase is an organizational evaluation. It determines the most important company's assets (critical assets) and their respective current protection efforts;

- Phase 2 - Identify Infrastructure Vulnerabilities: During this phase, the information infrastructure is evaluated. The analysis process strives to identify technology vulnerabilities which have the potential to compromise critical assets; and

- Phase 3 - Develop Security Strategies and Plans: Risks are identified and a mitigation plan is developed.

Managers should define steps to continuously review and improve the security effort. [9] According to [9], OCTAVE Method [22] is the original version, but there are two other approaches: OCTAVE-S [23], for smaller organizations (100 employees or less), and OCTAVE-Allegro [24], a streamlined variant oriented by information assets.

2.4.2 Microsoft Security Risk Management Approach

According to [10], this risk management approach has four phases:

- Phase 1 - Assessing Risk: This phase is responsible for determining risks. A qualitative approach is used to quickly triage risks. The most important risks are submitted to a quantitative approach;

- Phase 2 - Conducting Decision Support: Safeguards are proposed and evaluated. The best ones are presented to the organization's appropriate personnel;

- Phase 3 - Implementing Controls: During this phase the safeguards are implemented; and

- Phase 4 - Measuring Program Effectiveness: The implemented controls are monitored to validate an appropriate degree of protection. Additionally, changes in the environment are analyzed in search of risk profile changes.

2.5 Content Management System (CMS)

According to [25], CMS “is a software that facilitates the creation, organization, manipulation, and removal of information in the form of images, documents, scripts, plain text (or anything else for that matter)”.

According to [26], there are some essential and desirable features for a CMS. The essentials are:

- Continuity: It is needed to retain information through user interactions and protect it from hijacking. This aspect should be exported in an easy way to be used by extensions;
- User management: It is fundamental to provide mechanisms for user controlling, robust enough to handle many users and different authentication schemes;
- Access control: It is important to limit users’ privileges on the system;
- Extension management: A CMS should be easy to extend. Additionally, the administrator should be able to select what functionalities to enable on the system; and
- Security and error handling: A CMS has to be as secure as possible and should provide a helpful environment for building secure extensions. An error handling mechanism is also required.

The desirable features according to [26] are:

- Efficient and maintainable code handling: Among all code files, it is preferable that just the necessary ones, at some time, should be loaded. The mechanisms used need to handle code files of the extensions. It is important to note that the reference [26] is about development of CMS framework with PHP 5 [27], an interpreted language; that is why there are references for loading code files. So, analogies for other styles of programming languages can be made to turn this feature relevant;

- Database interface: High level functions for database access should be available for both the CMS core and extensions. This abstraction can enable the use of different databases through the same functions;

- Caches: Lightweight and easy to use cache mechanisms are desirable, in order to achieve optimization goals such as speed of operation and processing load;

- Menu: Means of creating and managing menus should be provided, including a standard interface to be used by extensions; and

- Languages: Different languages support should be available. A mechanism for fixed text translation is desirable. Additionally, other internationalization presentation should be available, like dates, times, monetary amounts, and numbers.

2.6 State of the Art and Related Work

The related work identified by this research is described by Sections 2.6.1, 2.6.2, 2.6.3, and 2.6.4, and represent a vision of the current state of the art of approaches to supporting the IT security risk management process. Section 2.6.5 compares the software proposed by this research with the presented related work.

2.6.1 GSTOOL

GSTOOL [5] is a proprietary software that supports IT security tasks that meet the requirements of IT-Grundschutz [28], a set of security practices and artifacts, some of which are in the BSI-Standard [29]. This tool is strongly coupled with their adopted security methodology, and it is not user friendly; according to [30], "the software is a tool for experienced users of the IT-Grundschutz Handbook. To be able to use it properly you must be familiar with the procedure described in the IT-Grundschutz Catalogues". Additionally, it requires a Microsoft Windows operating system, and does not provide a collaborative environment for user interaction.

2.6.2 GRC (Governance, Risk Management and Compliance) Tools

According to [31], GRC is a system of people, processes and technology that enables an organization to:

- Understand and prioritize stakeholder expectations;
- Set business objectives congruent with values and risks;
- Achieve objectives while optimizing risk profile and protecting value;
- Operate within legal, contractual, internal, social and ethical boundaries;
- Provide relevant, reliable and timely information to appropriate stakeholders; and
- Enable the measurement of the performance and effectiveness of the system.

Some GRC tools, with support for IT security risk management tasks, were identified [6, 7, 8]. These GRC systems are proprietary, so the information presented here was based on their official websites and documentations.

The identified GRC systems focused on security provide powerful functionalities for conducting security tasks, but do not rely on a collaborative environment.

The Fortify approach [8] seems to have a group interaction feature (comments), but in a non-structured way. This feature was discovered by some screen shots at [32], and as they are not directly explained in the documentation, it is not possible to analyze its real behavior. But, as shown in the screen shots, the comments are not structured to be considered a group discussion feature; they are just a list of comments.

2.6.3 RiskGuide

RiskGuide [33, 34, 35, 36, 37] is a web-based tool with a knowledge base of risks. It discusses the need for a continuous and collaborative tool to support risk management activities, but in a very different way than this work.

The main goal of RiskGuide is not directly related to security; instead, the main focus is to achieve requirements related to software development processes, like finishing the project on time, and keeping expenses within the budget. So, the risks modeled in such a system are general software engineering risks, not IT security ones.

The collaborative requirement identified by RiskGuide means many professionals individually identifying risks, using the same web-based application, in a way that the manager can analyze and consolidate those identified risks.

The need for a continuous risk management discussed by RiskGuide means that, at any time, any involved professional could input a new risk to the system and it got stored in a database for future analysis.

2.6.4 Protection Poker

Protection Poker [38] is a game, based on Planning Poker [39], that stimulates group discussions about security. During the collaborative practice, risks were identified and the security knowledge of members are shared among the team.

An experiment with Protection Poker was conducted in an undergraduate software engineering course at North Carolina State University, and it resulted in a more effective software security learning than in prior semesters. Additionally, students were observed relating vulnerabilities to business impacts for system studied. [38]

2.6.5 Comparisons with the Proposed Tool

Like GSTOOL, the application proposed by this research is based on a specific IT security risk management process, a NIST-based [3] one, but it is concepts-oriented. It means that users deal directly with concepts (e.g., vulnerability, risk, and safeguard). So, it differs from GSTOOL because there is a weak dependency against the adopted methodology. As it is only needed to handle concepts, that security professionals know and are easy to learn, it achieves the user-friendly characteristic that is lacking on GSTOOL. Additionally, the proposed tool is free and open-source.

GRC approaches provide a powerful environment, with many features that are not present on the current stage of the proposed tool, like charts, many kinds of reports, and compliance with many patterns. These functionalities improve the security effort, but they are not fundamental to an IT security risk management process; as identified by this research, the collaborative environment is a central requirement for any IT security risk management effort. None of the GRC tools relies on a collaborative environment, and they are not free and open-source like the proposed software.

In comparison to the Fortify approach, the proposed tool also provides a comments feature, but in a threaded-structured way to act as a group discussion mechanism. Additionally, the proposed tool has a real collaborative environment, with a forum for general security discussions and a subsystem to collaboratively create documents, important features that are not present in any of the identified GRC tools focused on IT security.

RiskGuide has an interesting knowledge base implementation, but there is no robust access control mechanism, the data are not structured through widely-accepted security concepts, and it is not focused on IT security. Its web-based characteristic achieves the portability requirement. The proposed tool is also web-based, but structures all information in security concepts, has a robust access control mechanism, and is focused in IT security.

Protection Poker is not a software tool, just a methodology with a game aspect, to identify risks, share security knowledge, and discuss security topics. The core of such an approach is to stimulate group discussions, achieving a better security analysis and improving the security knowledge of the persons involved. The proposed tool has mechanisms to conduct group discussions, and through them, security tasks can be improved because of the same benefits of Protection Poker, plus all corporate ones, like avoiding meetings, that are time-consuming practices which are sometimes difficult to schedule because the professionals involved are usually busy.

3 Proposed Solution

This work identified some requirements for an effective IT security risk management process, as discussed in Section 3.1. A solution was developed that fulfills these requirements: a software, based on an open-source CMS. Section 3.2 presents an overview of the proposed solution. Section 3.3 addresses the main concepts that are the core of the proposed IT Security Risk Management Collaborative Software (referenced by this work as IT-SRMCS). In Section 3.4, there is a high-level discussion of the developed software architecture.

3.1 Main Identified Requirements for an Effective IT Security Risk Management Process

According to [40], a requirement is "a condition or capability to which a system must conform". Based on issues discussed in Chapter 1 (Section 1.1), some high-level requirements were identified to improve the efficiency of an IT security risk management process. The following list enumerates the main identified requirements, that are discussed in more detail by the next four sections:

- Knowledge management solution support (Section 3.1.1);

- Continuous approach (Section 3.1.2);
- Collaborative work support (Section 3.1.3); and
- Robust access control mechanism (Section 3.1.4).

3.1.1 Knowledge Management Solution Support

The reports, developed as results of risk management activities, are usually voluminous, with a large number of risks to handle. A knowledge management solution can support information management in a structured format and make it easy to find desired information. For example, it is possible to execute a query for lists of risks with a high level, risk mitigation implementation status, and system components with a large number of identified flaws. This software solution should be secure, because an exploitation could lead attackers to get confidential information.

3.1.2 Continuous Approach

The business environment is very dynamic for the security point of view. For example, new vulnerabilities are often discovered, hardware and software technologies are in constant evolution, and there is a high turnover of professionals. So, to keep the security level of a company always acceptable, it is needed a continuous approach, in a way that as soon as new security information is detected, it must be properly handled.

3.1.3 Collaborative Work Support

The risk management process is a collaborative effort where professionals from different business areas and with different expertise must participate to detect and mitigate risks. Tools with features like threaded group discussions can facilitate risk analysis and priority definitions. Every idea and discussion should be stored for future analysis or reuse in a new project. This is an important feature due to the high turnover among IT professionals.

3.1.4 Robust Access Control Mechanism

The information involved in a security risk management process is often sensitive. The adopted solution must handle many users and roles in a way that each user has access only to the authorized information in a given project.

3.2 Overview of the Developed Solution

In order to fulfill the main requirements discussed in Section 3.1, this work implemented an IT-SRMCS extending a popular and widely-used open-source CMS, Drupal [41]. The IT security risk management process used by the proposed tool is based on NIST [3].

The main approach adopted to develop the tool is described in Section 3.2.1. Section 3.2.2 provides a list of software technologies adopted to develop the software. Some functionalities of the developed application are described in Section 3.2.3.

3.2.1 Adopted Approach

The approach adopted by this research to develop an IT-SRMCS relies mainly on five aspects:

1. Security practices;
2. Open-source;
3. Web-based;
4. Implementation over CMS; and
5. Software engineering principles.

The following five sections discuss, respectively, these aspects.

3.2.1.1 Security practices

Good security principles [42] were applied throughout the whole development process. The developed IT-SRMCS was intended to be as secure as possible in order to protect its related data.

3.2.1.2 Open-Source

An IT security risk management process, as stated before, deals with sensitive information about the organization, such as process and software vulnerabilities, that can be used by malicious people to plan attacks with the potential to compromise the core business.

In adopting proprietary software, the organization must rely on the manufacturer: it is possible to audit the binaries themselves, but it is usually not feasible due to the complexity and the dimension of the acquired systems. Additionally, the software updates are released usually by the manufacturer, because the binary patches development is not common by the community. It is also possible to acquire binary patches from specialized security companies, but the software usually loose the support from the manufacturer, an important advantage of proprietary software; additionally, the timeline for the binary patches creation is usually bigger than patches into the code because of the complexity of the process.

When the system is a security-related one, the scenario gets even more sensitive, because information leaking due to a software problem can seriously compromise the company.

The developed IT-SRMCS is open-source and, in being so, the organizations can audit the sources themselves instead of relying on the manufacturer. Additionally, with this approach, the organization can look for bugs and vulnerabilities itself, and correct them. Public reported flaws can also be addressed by the company itself, and the officially released patches can be audited too. It is also possible to acquire non-public patches from specialized security companies.

The developed tool was designed to rely just on open-source components (e.g., MySQL database [43], and GNU/Linux operating system [44, 45, 46]). In this way, the whole environment where the application is deployed can be fully audited, inheriting the described open-source benefits.

3.2.1.3 Web-Based

Nowadays, most of the modern operating systems come with a web browser due to the importance of the Web. Additionally, it is difficult to think of a company that does not use this technology because of the benefits of its applications, such as optimized bank transactions.

The websites can be accessed by most web browsers, running on most operating systems, due to network protocols and standards.

The tool developed by this research adopted a web-based approach in order not to require strong user machines dependencies; just an operating system, with a compatible web browser (the most used graphical web browsers are compatible), is needed. When the system is updated, no client modifications are needed and the end-users can have the operating system that is most convenient for them or for the company. Additionally, there is no specific installation process; a web-browser is required, but most operating systems come with at least one by default, and if another one is needed, the installation process is usually not complex.

Also, most web-browsers (to act as an user interface) and web-servers (to hold the application itself) have implemented security protocols, like HTTPS. So, some application development effort is saved, with security gains, in such a way that these protocols contribute to security, are very mature, and possible changes are transparent for the whole developed application (both client and server components).

3.2.1.4 Implementation over CMS

There are some CMS functionalities that are useful for software that aims to support an IT security risk management process, such as creation of content, user management, access control, and security. Most CMS implementations provide, built-in or through additional extensions, collaborative features like collaborative content creation, forum, blog, and poll.

This research implemented an IT-SRMCS over Drupal [41], a popular and widely used open-source CMS, in order to inherit all of its functionalities, instead of developing them from scratch. This approach has benefits, such as:

- CMS features inheritance: There are many useful features in a CMS implementation, such as forum, poll, and collaborative document creation. Additionally, some CMS implementations, such as Drupal, have new extensions available often, which means that, at any time, a new, and not yet thought of, feature for the IT-SRMCS may become available to make the application even more powerful;
- Fast and better implementation: As many features are inherited, such as content creation and user interaction, the development can focus just on the main area, the risk management process. This can lead to better and faster implementation, because all the development attention can be directed to the core objective of the software; and
- Robustness and security: The responsible for the inherited features is their manufacturers (in this case, Drupal and the module writers). As their attention is focused only on their features, they tend to be more stable, scalable, and secure, than if it was developed from scratch by the IT-SRMCS developers. This happens

because the amount of work is divided, and the developers can focus on their own components. In this way, the security risk management developers can focus just on this specific module, resulting in better implementation than if the same developers also needed to maintain the whole collaborative environment inherited from CMS.

3.2.1.5 Software Engineering Principles

The developed IT-SRMCS was modeled with good software engineering principles [47], using an Object Oriented (OO) approach [47]. The main benefits of this approach are:

- Easy maintenance: "Software is never completely finished" [48]. Even though security risk management basic principles are the same for most approaches, each organization has particular needs. A structured OO model can abstract technical concepts, in a way that the model gets easier for human-beings to understand [49], thus easier to maintain. Additionally, it is possible to achieve a robust architecture resilient to change [49], so the negative impacts of updates can be reduced. As an example, it is possible to change the web GUI (Graphical User Interface) and the database with a few punctual changes, with control of the impacts; and

- IT-SRMCS implementation uncoupled with the CMS: The application proposed by this research is developed as a Drupal module, so the system is highly-integrated with the CMS to inherit the most benefits possible. On the other hand, the proposed OO model makes the application completely apart from that. It follows that, while there is a full integration with Drupal, there is no need to modify the developed system itself to change the CMS implementation, or to eliminate the CMS approach itself to adopt another strategy or technology.

3.2.2 Adopted Software Technologies

The open-source technologies used to develop this research were:

- Operating system: Debian GNU/Linux 5.0 testing “lenny” with some unstable “sid” packages; [44, 45, 46]
- Web server: Apache 2.2.8-3 from Debian repositories; [50, 44]
- Web protocol: HTTPS;
- Database: MySQL 5.0.51a-12 from Debian repositories; [43, 44]
- Programming Language: PHP 5.2.6-5 from Debian repositories; [27, 44]
- CMS: Drupal 6.5; [41]
- UML modeling environment: JUDE Community 5.3 (Model Version: 28) [51]. This tool was used just to create diagrams;
- Text editor: VIM 1:7.1-175+2 from Debian repositories; [52, 44] and
- Web browser: Iceweasel 3.0.6-1 from Debian repositories. This is a Debian’s modification of Mozilla Firefox. [53, 54, 55, 44]

3.2.3 Functionalities of the Developed IT-SRMCS

The main functionalities of the developed tool are:

- IT security risk management support: There is support for a NIST-based IT security risk management process approach. The system is oriented for security concepts (e.g., risk, vulnerability, and safeguard), which means that the user deals directly with them. These concepts were modeled in a generic way, so the users have flexibility to customize the methodology to their needs. For example, the threat-source concept is not required, and neither is the asset one;

- Data analysis: The security effort remains well-structured in the system. So, it is possible to quickly find the desired information and request some classified data, such as current high level risks without protection, and the components of the system with the highest incidence of flaws;

- Support for continuous analysis: Conventional risk management approaches require periodic reassessment. This need is supported by the developed tool, but it is also possible for security professionals to continuously update the security model, whenever they want. This makes the system useful even for large organizations, whose IT environment is very dynamic;

- User notification: When important changes happen in a security analysis, it is possible to notify all authorized users about that. There are automatic notifications, that are generated by the application for the most important events such as the creation of a new security concept, and the manual ones, that are generated by the user through the application's interface. This is useful to keep all authorized users updated with the security efforts that they are involved.

- Robust access control: It is possible to create robust access control rules for users to access security analysis' concepts. Additionally, the access control mechanism is prepared to deal with a dynamic environment that has many and fast changes, and a high turnover of professionals. Changes in security components, other than new risk creation, do not require new access control rules. As an example, it is possible to add safeguards and assets with no access control rules modification;

- Secure virtual meetings on-demand: It is possible, for authorized users, to maintain discussions about each security analysis concept, such as risk and safeguard, in a structured and threaded way. This reduces the need for conventional meetings. Additionally, this interaction does not require considerable user schedule changes, because the communication is done through the system in a threaded discussion style, and does not need the involved users to be online at the same time: users can interact in their free time, with no need to schedule meetings. Additionally, the discussions are kept for future analysis, under a tight access control; and

- General collaborative services: It is possible for the users to interact with each other through forums, blogs, and polls. This can improve the global security knowledge, stimulate security studying, and unite personnel. Additionally, it is possible to create documents collaboratively, with tight access control restrictions. This is useful, for example, in creating security policies, whose development demands interaction among users from many departments.

The collaborative activities were inherited from Drupal.

3.3 IT Security Risk Management Process Concepts in the Proposed Solution

The developed IT-SRMCS handles each security risk management effort, basically, as a set of the concepts represented in Figure 3.1.

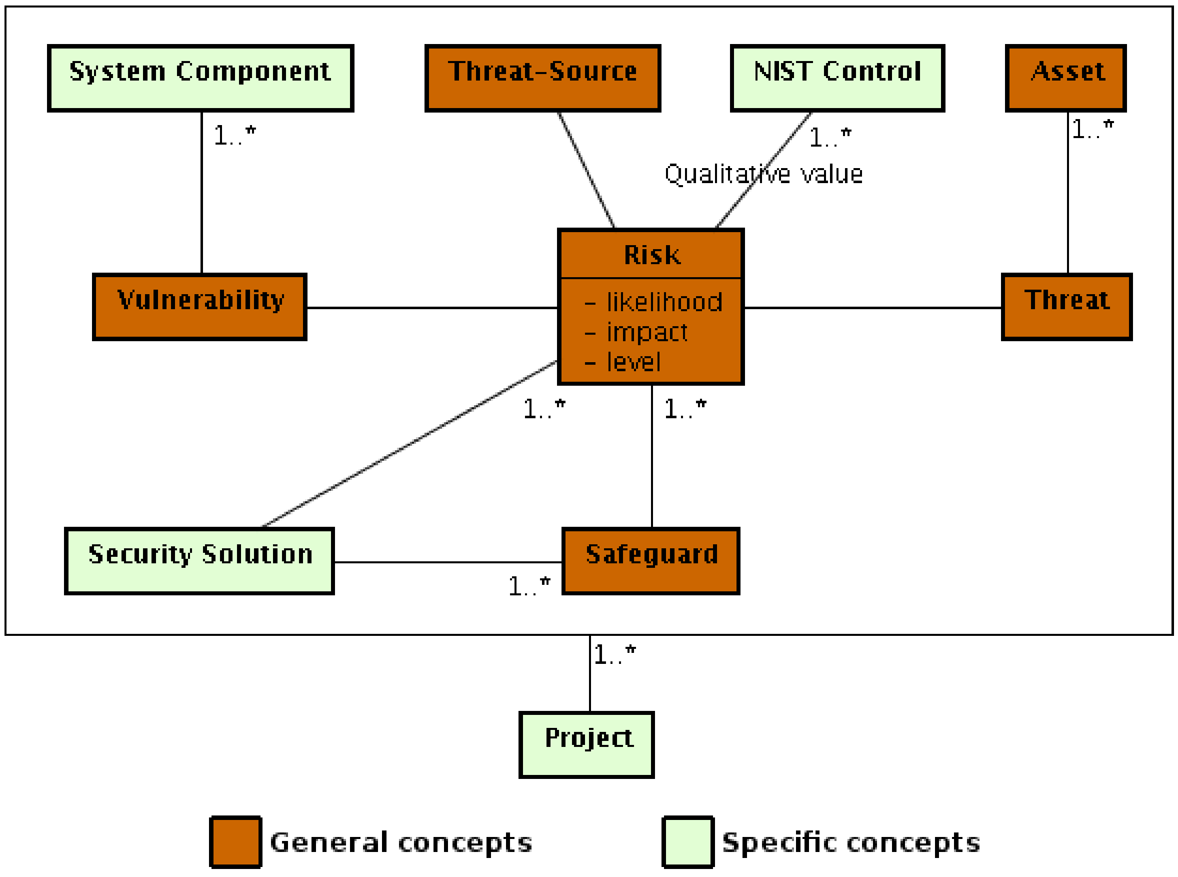


FIGURE 3.1 – Concepts modeling.

The concepts, as shown in Figure 3.1, can be classified in two categories: general and specific concepts. The former represents general security concepts to model security analysis processes, and were discussed in Chapter 2 at Section 2.2.1. The latter, represents specific concepts used in the NIST-based IT security risk management process that is modeled in the developed tool, and have the following meanings:

- System component: It is a component related to the system to be analyzed. It can have a software nature (e.g., confidential files inside a server, operating system, HTTP server, financial management application), hardware nature (e.g., electronic component of a computer, network adapter, router, printer), or something else (e.g., door, printed reports, safety box, human-beings and their habits). A security component does not have the same meaning as an asset: assets are a subset of system components. A system component is anything related to a system, whose protection can be both required (assets) or not (system components that are not assets). This definition does not mean that non-assets system components do not need protection; it means that they are not the core of the protection need. Flaws in system components that are not assets may lead attackers to compromise the assets themselves, and in this case they should be protected; for example, a confidential report inside a room is an asset, and the door is just a system component, but if the door was unlocked (vulnerability), an attacker could get the confidential reports. A vulnerability is related to at least one system component. As an example, an asset could be users' files and a vulnerability could be that the users write their network passwords on papers and keep them on their tables: the vulnerability is related to the user practices (just a system component), not to the asset itself. As another example, an asset could be the financial application, and the vulnerability could be a buffer overflow on this application: in this case, a flaw is related to the financial application itself (an asset that is also a system component);

- NIST control: Types of safeguards recommended by NIST [21]. There are management, operational, and technical controls, as discussed in Chapter 2 at Section 2.3.3.4. Annex C has a list of all NIST controls according to [21]. For each risk, one or more NIST controls are associated to represent the types of safeguards that can address the risk. For each NIST control associated to a risk, a qualitative value is assigned to represent the level that the current implemented safeguards' address the risk in terms of that NIST control;

- Security solution: Security solution is a set of one or more logic-related safeguards. It is also associated with one or more risks, to represent the risks addressed by the security solution; usually, these risks are the union of the safeguards' addressing risks. This concept is important because of the following:

  - It is more user-friendly to organize safeguards through one or more well defined characteristics, for example: operating system memory management safeguards, operating system process management safeguards, network safeguards, and policy safeguards. This provides a better (structured) view of the safeguards, because they get grouped by logical characteristics;
  - In some cases, there are more than one safeguard to address the same risk. But, for some reason, it was decided to adopt only a subset of the safeguards to address the risk, not all of them. A security solution is a way to select the safeguards that should be implemented to address a risk; and
  - There are situations where a single safeguard can address more than one risk. But, for some reason, it was decided to designate that safeguard to address only a subset of their related risks, not all of them. A security solution is a way to select the target risks for a single safeguard implementation.

- Project: The developed IT-SRMCS can be used to make various security risk management efforts. For example, the company's security personnel could analyze the security related to an internal sales process, and the manufacturer of an acquired financial system could conduct an individual analysis for their own system. Each effort is considered a project. For each project is associated one or more of each concept in Figure 3.1. This is a widely used concept, present in many kinds of software, such as programming environments, UML modeling tools, and project management applications.

3.4 High-Level Implementation Overview

This research proposes an IT-SRMCS that relies on a widely-used open-source CMS, Drupal. A Drupal module was developed with IT security risk management process functionalities. As this approach provides a full integration with Drupal, it is possible to inherit all its features. The result of such an approach (Drupal + developed Drupal module) is a robust environment for security-related tasks, and it is referenced by this work as "proposed solution". Figure 3.2 shows a high-level package diagram [56] of the proposed solution, that can be divided into six main components:

- Drupal (represented by package "drupal"): The adopted CMS. Responsible for providing a robust collaborative environment;
- The Drupal module (represented by package "drupalModule"): The developed Drupal module. It is responsible for integrating the developed IT security risk management functionalities with the Drupal environment. Additionally, the view layer is also implemented by the Drupal module;
- Main component (package "main"): Responsible for all functionalities directly related to an IT security risk management process, including the concepts (Section 3.3) definitions and their specific access control mechanism (Section 3.4.3);
- Persistence component (package "DAO"): It is called "DAO" due to the DAO (Data Access Object) pattern [57]. This component is responsible for handling all persistence operations with all data from the developed IT security risk management software. The term "developed IT security risk management software" means all packages developed by this research, with exception of the Drupal module;

- MySQL (represented by package “mysql”): The database adopted by the proposed solution; and
- Knowledge component (package “knowledge”): Responsible for handling the security knowledge base used to infer vulnerabilities and safeguards. Currently, this component is not very mature and was implemented just to validate its benefits for an IT-SRMCS. Additionally, it relies on the MySQL database to store its data.

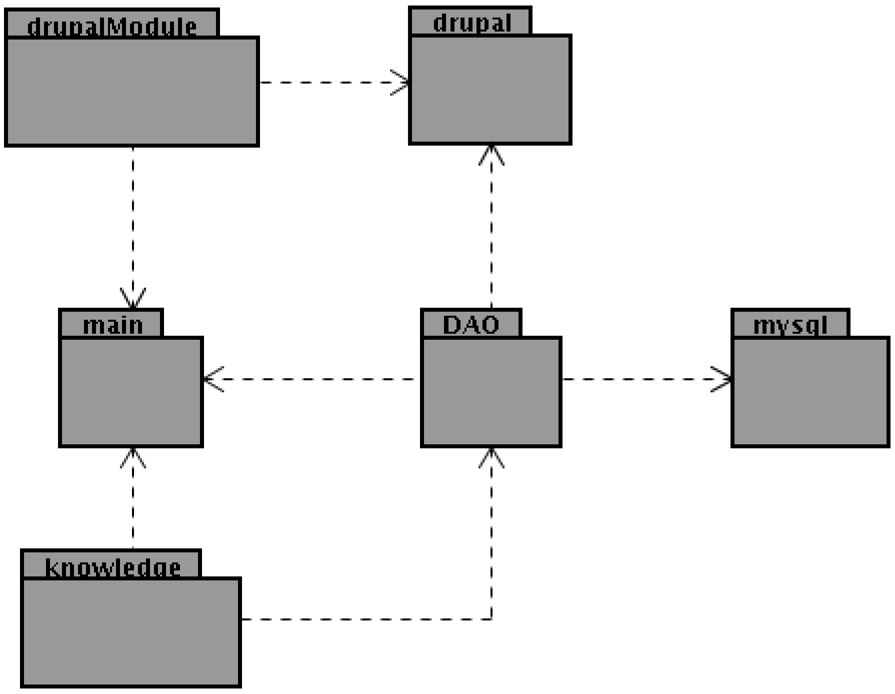


FIGURE 3.2 – Components of the proposed IT-SRMCS.

All packages developed by this research are collectively referenced as “developed software”. In Figure 3.2, only the packages “drupal” and “mysql” were not developed by this research.

A high-level view of the components interactions is described in Section 3.4.1. The coupling among developed IT security risk management software and the CMS is discussed in Section 3.4.2. The concepts, discussed in Section 3.3, have a specific access control mechanism, discussed in Section 3.4.3.

3.4.1 Global Dependencies Overview

Some software engineering principles, such as GRASP's Low Coupling [47], recommend making unstable components (with high update rating and probability), such as technology-related ones like databases, to depend on more stable components (with low update rating and probability). The result is that the most frequent and probable changes will affect a minimum number of components.

Security concepts are all concepts described in Section 3.3 that are directly linked to security tasks; it means all of them, with the exception of Project. There is an important dependency in the system: almost the whole developed software depends on the security concepts, whose definitions are in the main component. It seems like an undesirable dependency, but it is not. These concepts are the core of an IT security risk management process, and are stable definitions that do not change frequently.

Drupal is prepared to handle modules. Each module interacts with Drupal through hook functions. [58] So, there is a dependency from the Drupal Module against Drupal.

As shown in Figure 3.2, the persistence component depends on the main one, the opposite of the most natural logic: the system classes' objects (package "main") calls data access objects (package "DAO") for persistent operations, resulting in a dependency from package "main" against package "DAO". In runtime, this logic is really correct, but in the static model, this dependency represents an undesirable coupling, because an unstable component (persistence, that are directly linked with the database) gets coupled with a more stable one (main component). So, this intuitive dependency is addressed in the static model, as discussed in Chapter 4.

The knowledge component depends on the main one, achieving a robust dependency because an unstable package depends on a more stable one. Additionally, the dependency from package "knowledge" to package "DAO" was not addressed such as in the "persistence-main case", because the "knowledge" package, due to its immature characteristics, is more unstable than the technology one.

3.4.2 CMS Coupling

The Drupal Module component is an intermediate between the developed IT security risk management software and Drupal. In order to execute any security risk management task, the Drupal Module component calls the main one through an object of Coordinator class, its only connection with the developed IT security risk management software. Being so, the main component is uncoupled from the Drupal Module, and the Drupal Module is not coupled with the main component's internals, just with the Coordinator class. As a result, the developed IT security risk management software is low-coupled with Drupal, and at the same time is fully integrated with it. This allows an easy substitution of both, the adopted CMS and the internal logic of the developed IT security risk management software. The package diagram in Figure 3.3 depicts the discussed scenario.

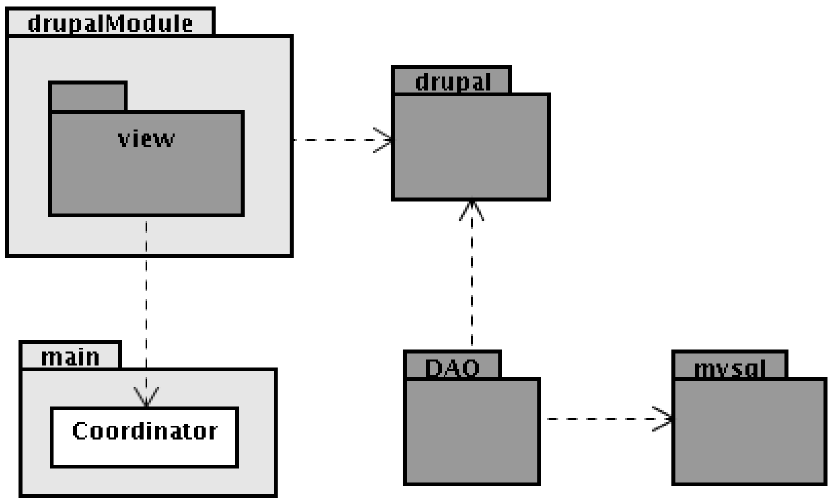


FIGURE 3.3 – CMS coupling.

The view layer is inside the Drupal Module component because all presentation schema were inherited from Drupal. This strong CMS coupling means that, in case of CMS exchange, this layer has to be rewritten. This is acceptable because there is no business logic in the view layer (the business logic is in the developed IT security risk management software), and other developed components are uncoupled with the Drupal Module. Additionally, there are mainly two advantages in inheriting Drupal's interface in the view layer:

- Drupal's theme system: It makes the layout development faster and more robust, with a full integration to Drupal's theme system, where the screens are adjusted to the current theme characteristics automatically; [59] and
- Security: It is possible to prevent some Cross Site Scripting attacks by using Drupal's API (Application Programming Interface) [60] functions to sanitize users' inputs and outputs. [61, 62]

The persistence component ("DAO" package) is not part of the Drupal Module to reduce the dependency on Drupal, but uses functions provided by Drupal API. This is another acceptable CMS coupling, because there is no package that depends on the persistence component's implementation details. Additionally, there are mainly two benefits of using Drupal API functions:

- Database independence: Drupal's has native support for two widely-used and robust open-source databases, MySQL [43] and PostgreSQL [63]. The API functions are independent from both databases, with no need for code changing in case of database exchange; [64, 25] and

- Security: It is possible to prevent SQL injection attacks through the format string parameters used in Drupal's database functions [64]. Additionally, some functions used to sanitize user inputs at the view layer [61] also remove special characters, and this behavior also contributes in preventing SQL injection attacks.

Drupal has a robust access control system that is used by the proposed solution. Additionally, the user management task was also inherited from Drupal. As briefly stated before, the access to information related to the concepts discussed in Section 3.3 (e.g., asset, vulnerability and risk) is protected by a specific access control system, developed by this research. So, the developed access control system should be compatible with Drupal's user management system. Thus, an undesired dependency appears: the main component (where the developed access control system is located) needs to interact with Drupal (where Drupal's user management system is located). This undesired coupling was addressed: the main component deals with an abstraction, "user id" (integer), instead of the user management system itself. So, the main component remains uncoupled from Drupal. Additionally, as many user management systems, from the most different kinds of software, implement this abstraction, the main component is also compatible with them. For the case in which some software, such as another CMS, has an user management system with another abstraction to uniquely identify users, just few changes are needed: the Coordinator class should be updated to transparently convert the values.

3.4.3 Proposed Access Control Mechanism

Drupal, being a CMS, can handle contents. There are many content types available, such as news articles and blog entries. Drupal has a generic abstraction to represent a content, called node. [65] It has also a robust mechanism to restrict access to nodes, called node access system [66]. The proposed IT-SRMCS uses Drupal's nodes, and the node access system, just for view layer purposes.

However, the main concepts (Section 3.3) themselves are not handled by Drupal, just their related view aspects (nodes). So, a robust access control mechanism was developed to protect accesses to concepts (Section 3.3), as shown in Figure 3.4.

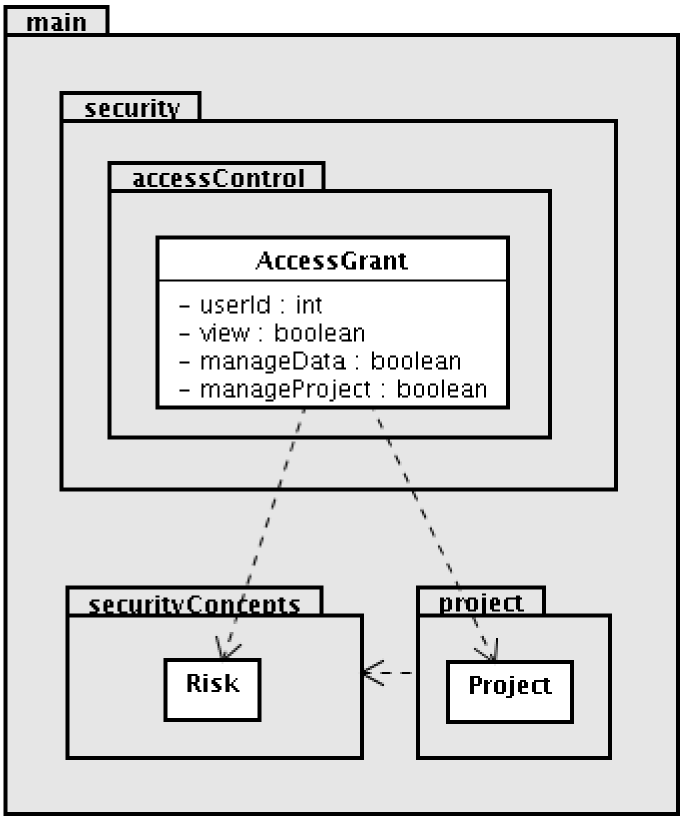


FIGURE 3.4 – Proposed access control mechanism.

In the proposed model, each access grant is related to a user (AccessGrant's userId attribute), and a target project (Project class). There are three types of access grants (determined by the related AccessGrant's attribute value):

- View: This kind of access grant has an additional link with a risk concept (Risk class). The meaning is that a user, in a defined project, can view (just read) that risk's information, and all its related security concepts;

- Manage data: The user has full access to all security concepts in the specified project; and
- Manage project: The user has full access to the determined project. It grants full access to all the project's security concepts such as "manage data" access grant, allows the user to change information related to the Project concept (Section 3.3) itself, and authorizes the user to manage (create, update, delete) access control rules for the project.

A security risk management process is a risk-oriented effort, and the view access grant attends this constraint: all access grants are also risk-oriented.

The proposed access control mechanism does not have a feature to allow a user to change only a specific security concept, such as an asset; either all project-related security concepts can be changed, or only the risk-related security concepts can be read. This decision makes the system more robust, because if a user has only a limited view of the security effort and has permission to make changes, inconsistencies between already existent information, but not accessible by this user, could be created. Additionally, that decision also makes the proposed IT-SRMCS to fit in the dynamic environment of the organizations, because there is no need to write new rules for each new security concept added to the system, such as a new asset, since the security concept was related to an existing risk; access control changes, in this context, are only needed in the case of a new risk creation.

4 Implementation

Chapter 3 (Section 3.4) presented a high-level view of the implementation, discussing the relations among the main components of the proposed IT-SRMCS. This chapter aims to describe the internals of the developed packages that are depicted by the diagram in Figure 4.1 in a black-box perspective.

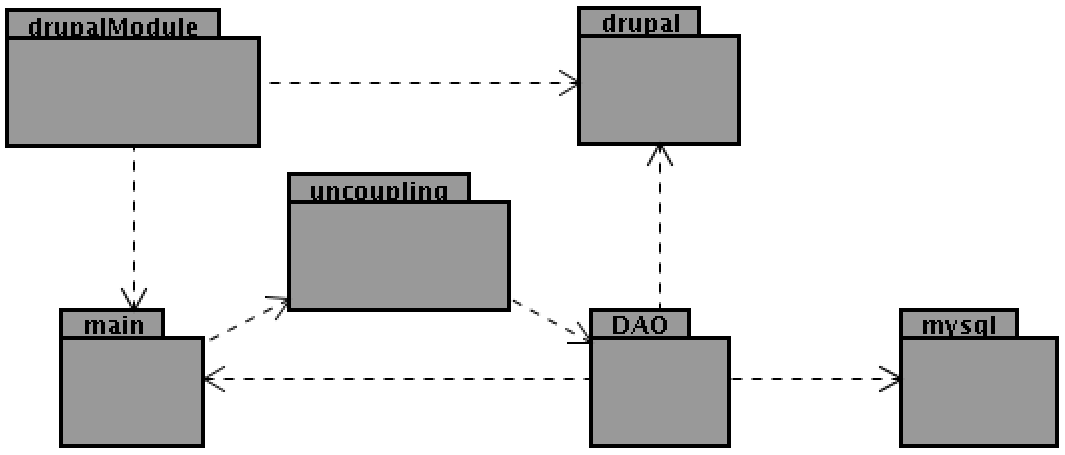


FIGURE 4.1 – Packages of the proposed tool in a black-box perspective.

Packages “drupal” and “mysql” are not developed packages, they only represent adopted technologies for, respectively, CMS and database.

Section 4.1 discusses two packages, “main” and “uncoupling”. Section 4.2 describes the package “DAO”. The developed Drupal module, represented by the package “drupalModule” in Figure 4.1, is described in Section 4.3. Section 4.4 discusses the adopted Drupal modules for the collaborative functionalities of the proposed IT-SRMCS.

4.1 Packages “main” and “uncoupling”

An overview of the package “main” and its main relations is depicted in Figure 4.2.

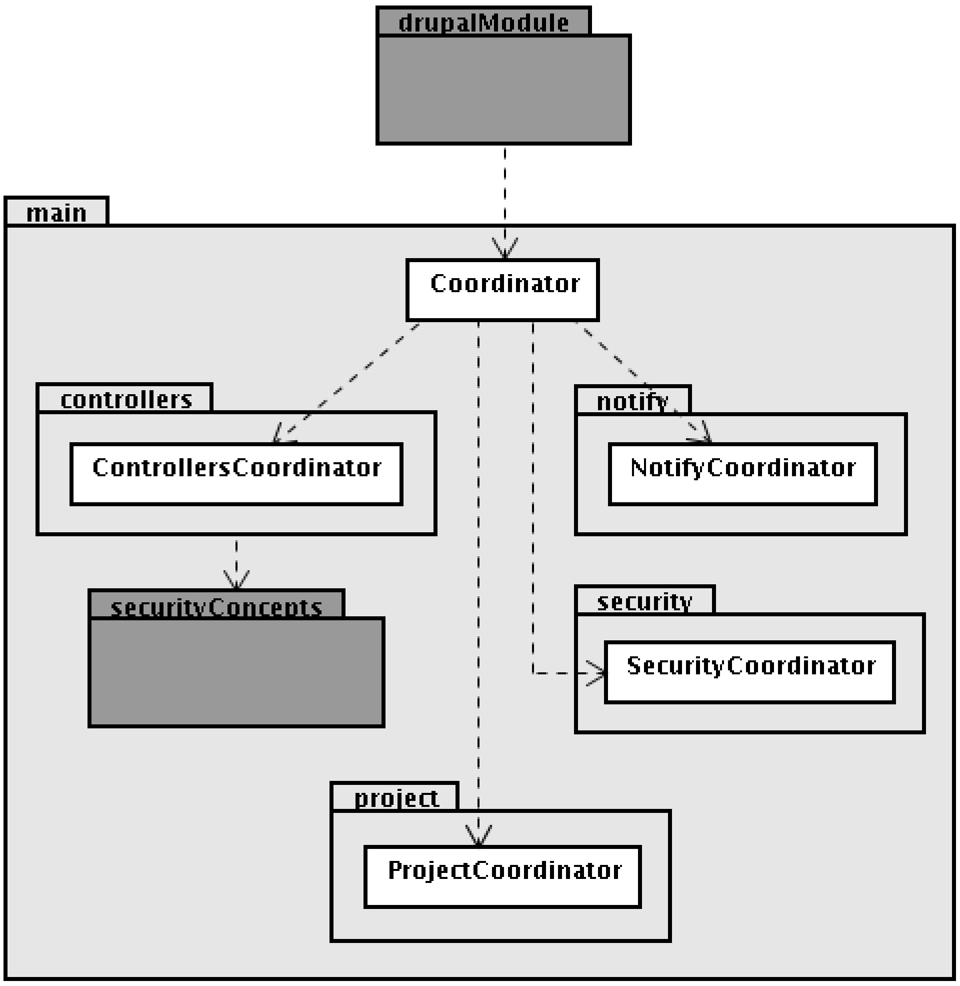


FIGURE 4.2 – Package “main” and its main relations in a gray-box perspective.

As discussed in Chapter 3 (Section 3.4.2), all IT security risk management operations are requested from the developed Drupal module to an object of the Coordinator class. This class represents the GRASP’s Controller (Façade) pattern [47]; it means that an object of the Coordinator class does not do the work, but coordinates the activity delegating the work to other objects. That is why the name Coordinator was assigned.

The sub-packages of “main”, with the exception of “securityConcepts” because all its classes are Jacobson’s Entity [47, 67], have their own coordinator class, that acts as a GRASP's Controller (Façade). This approach reduces the coupling among such sub-packages, because they are contacted exclusively through an object of their coordinator class. It follows that their internal details, with the exception of the coordinator classes, are completely transparent for external packages.

The "main" sub-packages are discussed by the next sections. Section 4.1.1 discusses the "securityConcepts" sub-package. Section 4.1.2 describes the "controllers" and "uncoupling" sub-packages. Section 4.1.3 discusses the "project" sub-package. The "security" sub-package is described in Section 4.1.4. The "notify" package is discussed in Section 4.1.5.

4.1.1 Package "securityConcepts"

The package "securityConcepts", which models the concepts directly related to a security effort, is shown in Figure 4.3.

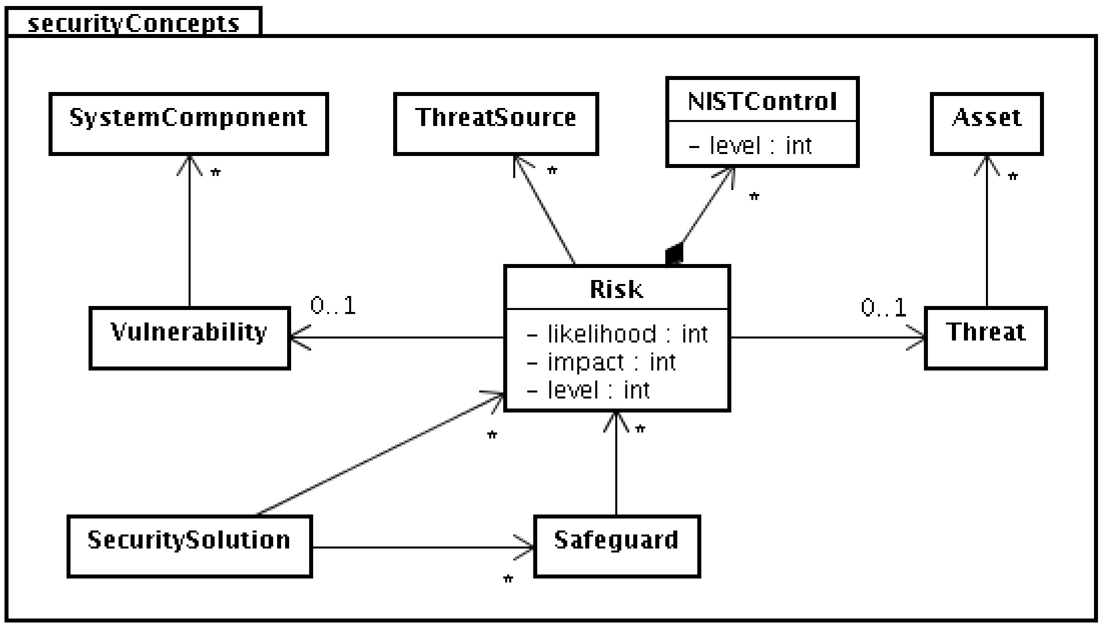


FIGURE 4.3 – Package "securityConcepts".

This package does not depend on any other, but as discussed in Chapter 3 (Section

3.4.1), almost all packages depend on it. This behavior can be explained by the fact that the system is concepts-oriented, so the security concepts are handled all the time, from the user inputs, to the system's outputs. As discussed in Chapter 3 (Section 3.4.1), this is an acceptable coupling.

The concepts modeling illustrated in Figure 4.3 has few deviations from their definition in Chapters 2 (Section 2.2.1) and 3 (Section 3.3). This fact does not affect the concepts' meanings; it is only a project decision to make the system more generic, and to increase the user-friendly characteristic. The following list explains and justifies each deviation from the definitions in Chapters 2 and 3:

- Risk and vulnerability: By the definition, a risk must have one vulnerability. But, according to the proposed model, it is possible for a risk to have no vulnerability. The meaning was not changed, but this characteristic was adopted only for user convenience, in cases that a user identified a mistake in a vulnerability and opted for removing it until the correct one was created. So, a risk without a vulnerability can exist, but is considered incomplete;

- Risk and threat: According to the definition, a risk is associated to one threat. But, as defined by the proposed model, it is possible for a risk to have no threat. This characteristic was adopted to make the system compatible with methodologies that do not handle threats;

- Vulnerability and system component: Through the definition of both concepts, a vulnerability is associated with at least one system component. In the proposed model, it is possible for a vulnerability to be linked with no system components. The involved concepts remain the same, but the possible omission of system components makes the developed system compliant with security approaches that do not handle system components;

- Threat and asset: By the definition of these concepts, a threat is related to at least one asset. In the proposed model, it is possible for a threat not to be linked with any asset. This peculiarity does not affect any concepts' meanings, but improves the developed tool's compliance with security approaches that do not handle assets;

- Risk and threat-source: By the definition, a risk is linked with a threat-source. But, in the proposed model, a risk can have zero or more threat-sources. Due to past author's experiences, in a real world security effort, it is common that many threat-sources can be related to the same risk. The cases when threat-sources are the key to creating risks with distinct content are rare. It means that, if the definition was strictly followed, the analysis, most of the time, would result in many equal risks, with only a threat-source change. This kind of risk presentation is not user-friendly, because there would be many risks to be analyzed with exactly the same information with the exception of the threat-source. So, the proposed model is flexible enough to handle a risk with only one threat-source, like the definition, or with more than one, for user convenience. Additionally, with the proposed model, it is possible for a risk to exist with no threat-sources, which makes the tool more flexible in being used by security approaches that do not handle threat-sources;

- Risk and NIST control: According to the definition, a risk is linked to at least one NIST control. But, in the proposed model, a risk is related to zero or more NIST controls. This deviation does not change any definition, but makes the system more flexible, because the user can decide to use, or not, the NIST control concept in the security effort;

- Safeguard and risk: The proposed model deviates from definition of safeguard, because the safeguards can have no risks. This characteristic was adopted for user convenience, for the case that a safeguard, written for some risk that do not exist anymore, is intended to be used later; and

- Security solution and risk, and security solution and safeguard: It is possible for a security solution to have no risks and/or no safeguards. This characteristic does not change the concepts' definition, and was adopted for user convenience for the case that the user commited a mistake on a risk and/or safeguard and wants do keep the security solution incomplete until the needed corrections were developed.

4.1.2 Packages "controllers" and "uncoupling"

The package "controllers", responsible for handling entities of "securityConcepts", to be completely discussed, demands other packages explanations. Figure 4.4 depicts all packages related to "controllers".

The ControllersCoordinator class is directly connected to all controllers. Each controller class is directly linked to an entity in package "securityConcepts"; for example, AssetController (package "controllers") is related to the Asset (package "securityConcepts"). This dependency happens because the responsibility of controller classes is to handle their related security concepts in a system-wide perspective. Examples of operations are add (that inserts some entity in the system), delete (that removes some entity from the system), and get (that obtains some entity that are currently in the system).

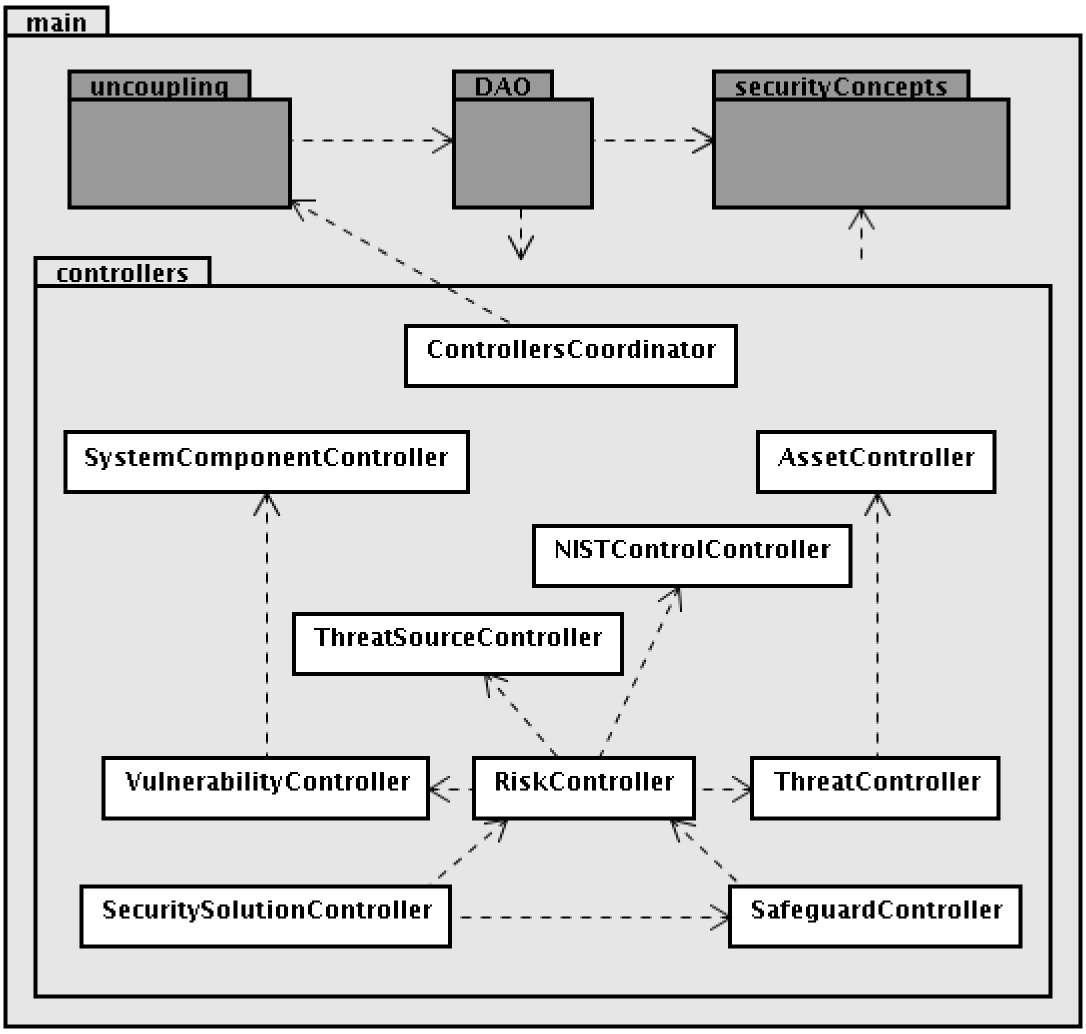


FIGURE 4.4 – Package “controllers” and its main relations.

There are cases where an entity from package “securityConcepts” has dependency against other ones from the same package, such as the Threat that has zero or more Assets. In this case, a ThreatController object talks to an AssetController object in order to manipulate the assets inside threats. This reduces the coupling, because each controller only handles its own entity. In the “Threat-Asset example”, a ThreatController object relies on an AssetController object for all Asset-related operations.

The main operations executed by the controllers need to interact with the persistence layer. So, the DAO pattern [57] was adopted, resulting in the “DAO” package, which contains all database operations. Under good software engineering principles [47], it is possible to invert the default dependency on DAO classes, making “DAO” package to depend on “controllers”, because “DAO” is more unstable once it is linked directly to a database. So, the Adapter design pattern [68] was adopted to invert this dependency according to GRASP Indirection pattern [47], and Inversion of Control and Dependency Injection patterns [69]. It follows that each controller has an interface that should be implemented by the respective DAO class. Dynamically, the dependency remains from “controllers” to “DAO”, but statically the dependency is the opposite. Figure 4.5 depicts the dependency inversion technique for the Asset case.

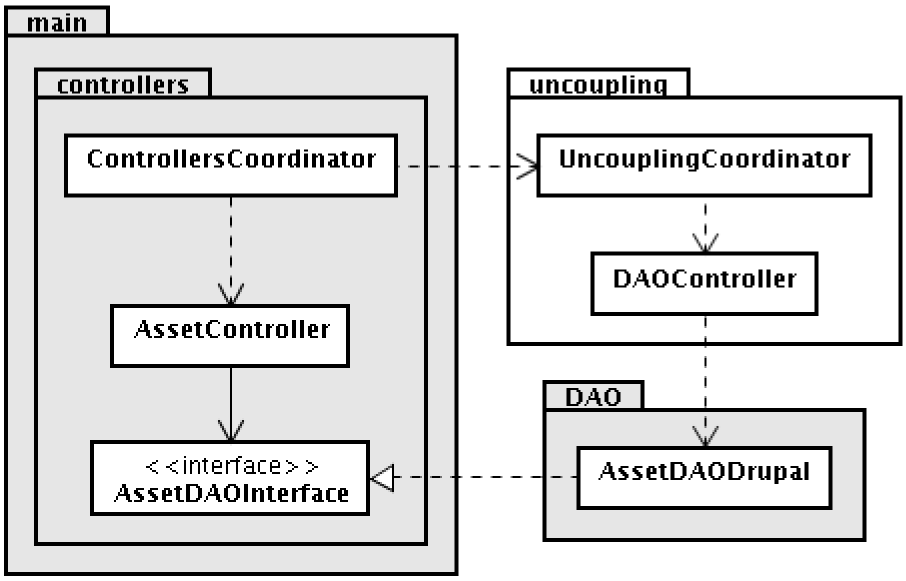


FIGURE 4.5 – Dependency inversion technique among packages “controllers” and “DAO”, illustrated by the asset case.

Business classes are linked with interfaces, so their objects need instances of “DAO” classes. The “uncoupling” package is responsible for instantiating DAO objects and returning them to “controllers” objects. So, a possible database change requires two modifications: coding of a new class to deal with the new database, and the “uncoupling” package changing to modify the instantiated objects. The “uncoupling” package is also communicated only through an object of its coordinator, hiding internal details and achieving a less coupled model.

The inversion of dependency improves a software model that uses the DAO pattern because:

- Distinct database-related implementations can stay together, in a structured way. In the case of database change, if a specific implementation for it already exists, only the "uncoupling" package needs to be changed to instantiate the new DAO objects. In a model with only a DAO approach, the class with database operations must be rewritten, and the old one may be lost;
- Documentation: The model becomes clearer, in such a way that text-based instructions can be avoided in the documentation about the software architecture. As database changes may happen, with only a DAO approach a textual documentation with procedures to exchange the database without losing the actual implementation would be needed, for the case in which the old database was re-adopted. For example, the DAO class could be copied to a safe place, to be restored later. DAO with dependency inversion approach embeds the whole explanation, well-structured, in the model itself; and
- Maintenance effort: As the model is clearer when there is DAO with dependency inversion, mistakes from new software engineers in the development team can be avoided, because the model is self-explained and does not require the reading of long text-based instructions.

4.1.3 Package "project"

Figure 4.6 depicts the package "project" and all other packages needed to discuss its behavior.

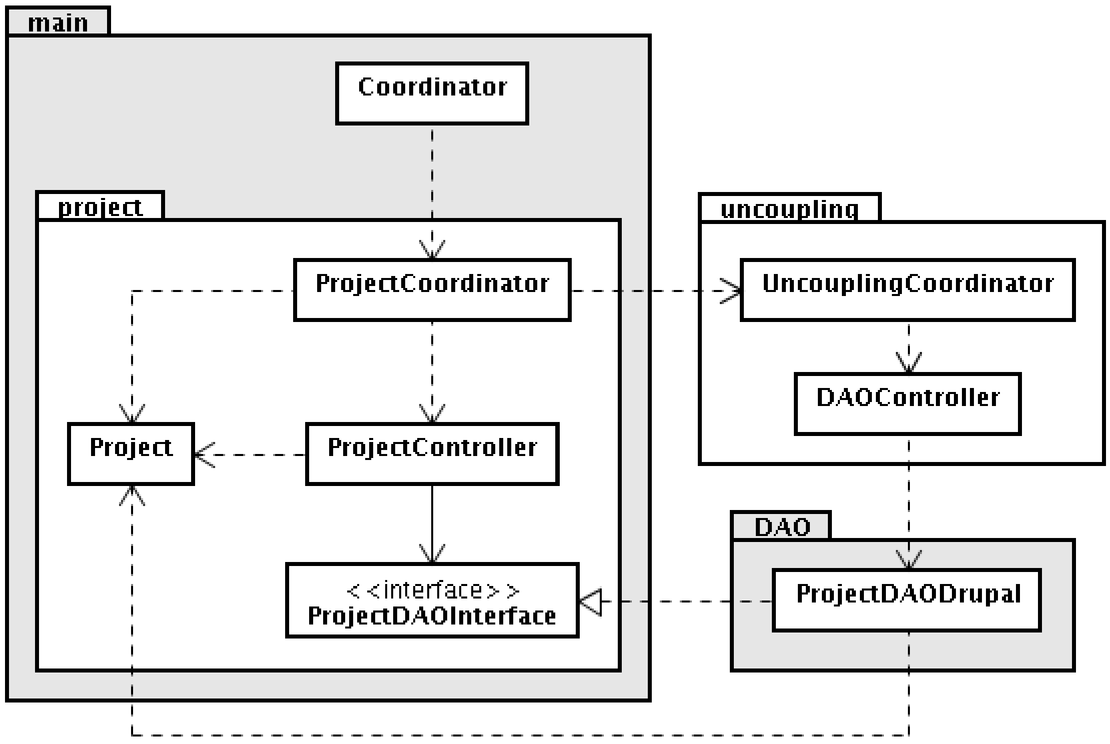


FIGURE 4.6 – Package "project" and its main relations.

The Project class is a Jacobson's Entity related to the project concept discussed in Chapter 3 (Section 3.3). The same already discussed principles of coordinator, controllers, and DAO with dependency inversion, were also applied for the scenario of "project" package. The difference is that all project-related classes are in the same package for simplicity reasons, because there is only one concept involved (project).

Additionally, there is no dependency against "securityConcepts" and "controllers" packages, as expected due to the concept definition (Section 3.3), because the implemented Project entity does not deal directly with security concepts, but with their ids (integer values). So, it is up to the object that is handling a Project instance to call other objects

(ControllersCoordinator or Coordinator) in order to get security concepts' instances. This project decision was adopted to reduce development time, but the proposed model is flexible enough to link the Project directly with the security concepts classes with weak and controlled impacts.

4.1.4 Package "security"

The "security" package is responsible for all security operations regarding the implemented concepts (Section 3.3). In the current stage of the proposed system, there is only an access control mechanism in this package. Figure 4.7 shows the package "security" and all other packages needed to discuss its behavior.

The adopted access control behavior was discussed in Chapter 3 (Section 3.4.3). The entity that represents an access control rule that should be persisted is the AccessGrant class, and it is directly linked to the classes Project ("project" package) and Risk ("securityConcepts" package); it follows that coordinator objects of classes ProjectCoordinator (package "project") and ControllersCoordinator (package "controllers") are called by instances of SecurityCoordinator and AccessControlController classes from "security" package. The same already discussed principles of coordinator, controllers, and DAO with a dependency inversion approach, are also applied for the "security" package scenario.

Additionally, the principles of coordinator, controllers and DAO with dependency investion are also applied in the "security" package.

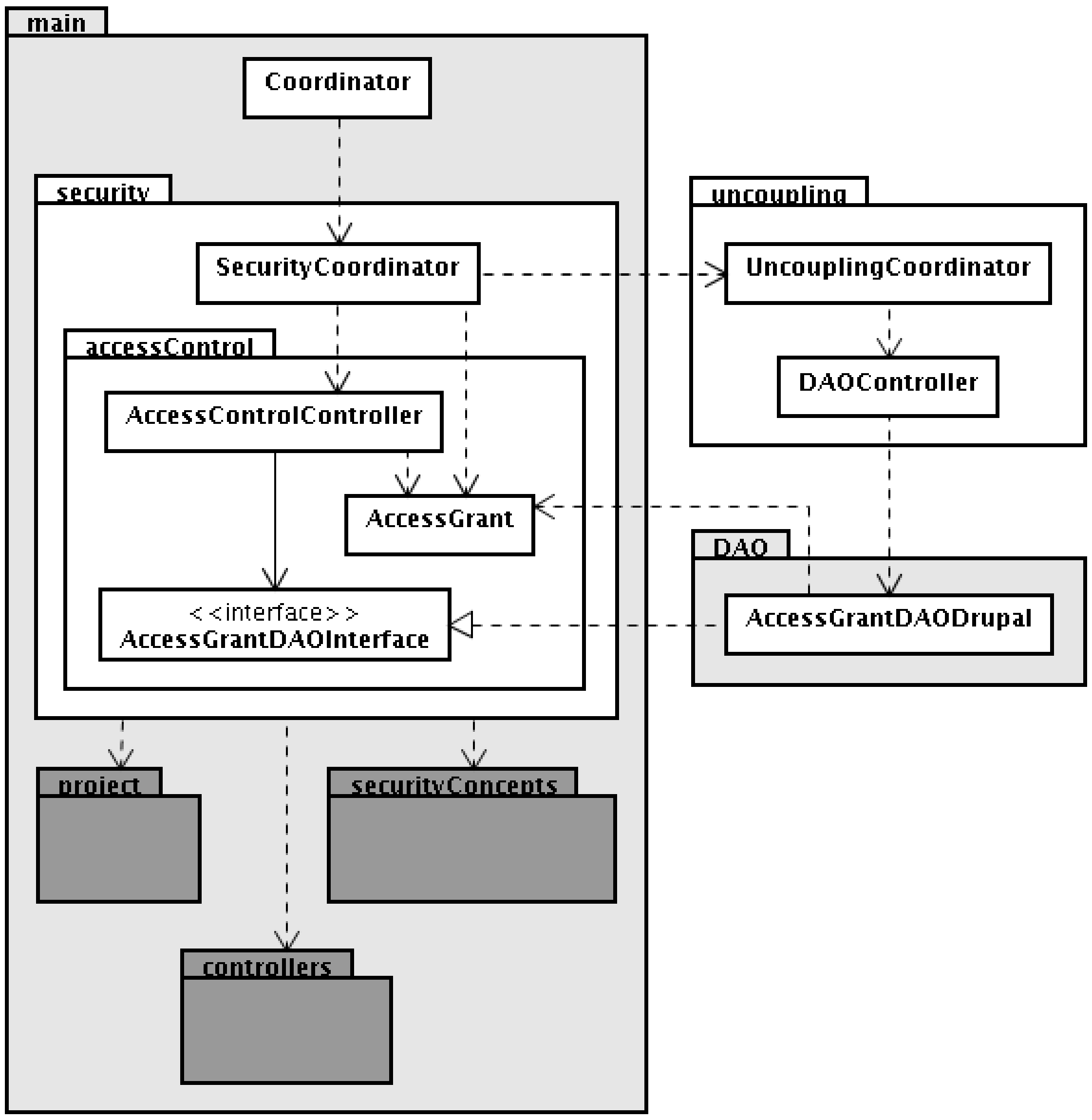


FIGURE 4.7 – Package “security” and its main relations.

4.1.5 Package “notify”

When changes are made in some concept (Section 3.3), it is possible to notify all involved users about that. The package “notify” is responsible for managing all notifications in the system. This package is depicted in Figure 4.8 with its main relations.

The same principles of coordinator, controllers, and DAO with dependency inversion, applied for other packages, are also present in the “notify” package scenario.

The Notification class specifies each requested notification about concepts in the proposed solution. The interpretation of its attributes is that each notification is for a user (userId), in a project (projectId), and is related to a concept (conceptType and conceptId).

Similarly to "project", this package has no dependencies against "securityConcepts", "controllers", and "project" packages, due to the adoption of the concepts' ids (integer values) instead of the classes themselves. But the model is prepared to change this characteristic with weak and controlled impacts.

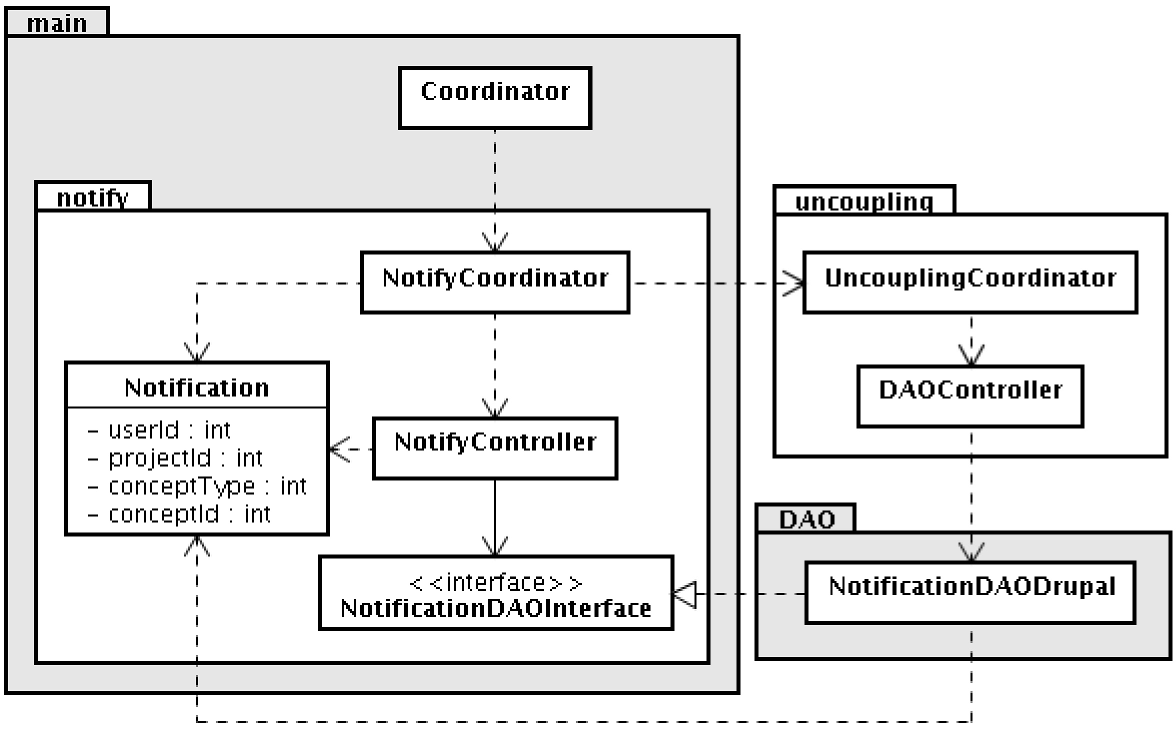


FIGURE 4.8 – Package "notify" and its main relations.

4.2 Package "DAO"

This package is represented by Figure 4.9 together with its dependencies.

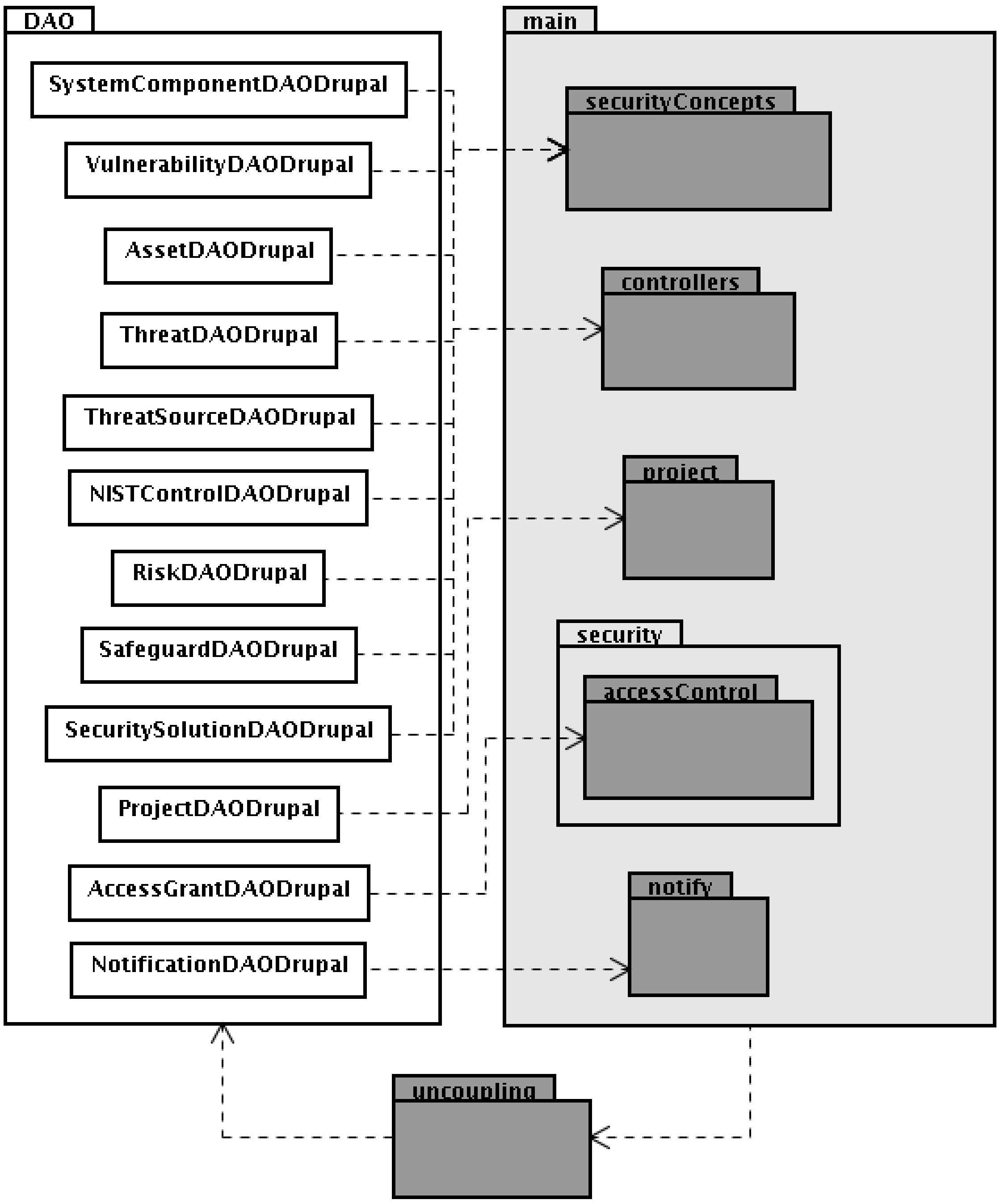


FIGURE 4.9 – Package “DAO” and its main relations.

This package has one DAO class for each persistent entity in the system. In their implementations, Drupal functions are used, as discussed in Chapter 3 (Section 3.4.2). There is an indication that Drupal’s API was used in the name of the DAO classes due to the possibility of having implementations for many databases in the same package, as discussed in Section 4.1.2; instances of “uncoupling” classes choose the target implementation, as discussed in Section 4.1.2.

DAO classes implement an interface in the “main” package due to the dependency inversion technique discussed in Section 4.1.2. Additionally, each DAO class deals with its related concept class, but there are no dependencies among DAO classes.

4.3 Developed Drupal Module

There is no “drupalModule” package in the system; it is just a representation of the developed module for architectural purposes. Similarly, as stated before, packages “drupal” and “mysql” also do not exist; they are just used to represent Drupal and MySQL, respectively.

The developed Drupal module is responsible for providing user interfaces for all operations related to the IT security risk management process. The operations requested by users through the user interface are sent to the developed IT security risk management software, through an instance of Coordinator class (package “main”), to be validated and processed. The response from the coordinator object is formatted and presented to the user by the developed module.

Additionally, this module provides a full integration of the developed IT security risk management software with Drupal.

Drupal handles its modules through hook functions [65]. The following sections discuss the main hooks used in the developed module.

4.3.1 hook_node_info

Drupal is a Content Management System, so it handles contents. Contents are internally known as nodes, and hook_node_info is used to define new node types [70].

Each concept, discussed in Chapter 3 (Section 3.3), was modeled as a specific node type. This project decision makes it possible to use Drupal's functionalities in a specific way for each concept (node type). Additionally, there is also a node type to represent an access grant of the developed access control mechanism discussed in Chapter 3 (Section 3.4.3).

4.3.2 hook_perm

This hook is used to define user permissions [71]. Such permissions are used, by Drupal's administrator, to restrict accesses to module's functionalities [71]. The developed Drupal module creates two permissions, one to restrict access to IT security risk management functionalities, and another to let users, with the first permission, create new IT security risk management projects. So, the Drupal administrator can determine the users that have access to the developed IT security risk management software, and which ones are also allowed to create new projects (Chapter 3, Section 3.3).

4.3.3 hook_access

All user interactions with the system are made through the user interface. Chapter 3 discusses some concepts (Section 3.3) and an access grant class (Section 3.4.3). Such entities are modeled by nodes in the developed module. Interactions with nodes are verified by the respective hook_access implementation to determine if it is allowed or not [72]. It means that all user accesses to those entities' information are filtered by hook_access.

As all information related to the developed IT security risk management software should be protected by the access control mechanism discussed in Chapter 3 (Section 3.4.3), this mechanism is used in the hook_access implementation.

4.3.4 hook_node_grants and hook_node_access_records

As discussed in the previous section (Section 4.3.3), hook_access restricts accesses to nodes. But there are some cases that such hook is not called. As an example, the Tracker core module [73], used to publicly display recently created or updated contents, does not rely on the nodes' hook_access.

It is possible to handle such situations with the implementation of two hooks, hook_node_grants [74] and hook_node_access_records [75]. [66] The developed Drupal module has both functions implemented in a way that all accesses to its nodes, that have not been verified with hook_access, are denied.

4.3.5 Hooks to Handle Users Interactions

The developed IT security risk management software is accessed by the users through web pages generated by Drupal. The following hooks are responsible for handling such web pages:

- hook_form: Its implementation allows the creation of customized forms for node types [76]. It means that each entity described in Chapter 3 (concepts and AccessControl class) has its own form with its specific fields, and through them related information is inputted to the system;

- hook_validate: Its implementation makes it possible to verify if a node is in a valid format to be added to the site [77]. In the context of the developed system, this hook is used to validate some user inputs when some entities (concept or access grant) are created or updated;

- hook_insert: Its implementation defines actions to be taken when a node is inserted [78]. A new node insertion, in the context of the developed software, means the creation of a new object that should be added to the system. Such an object is related to a class that represents a concept discussed in Chapter 3 (Section 3.3), or an access grant also described in Chapter 3 (Section 3.4.3);

- hook_update: Its implementation handles a node update action [79]. In the context of the developed system, it means that an entity (concept or access grant) was changed;

- hook_delete: Its implementation handles a node deletion action [80]. In the context of the developed system, it means that an entity (concept or access grant) was deleted;

- hook_load: Called to load node-type-specific information. [81]. In the context of the developed software, it means the loading of an entity (concept or access grant) information. So, the implementation of such hook, requests, for an instance of the Coordinator class (package "main"), an object relative to the entity that should be presented to the user; and

- hook_view: Its implementation allows the definition of a custom presentation for a node [82]. In the context of the developed software, it means that an entity (concept or access grant) is being presented to the user. So, the implementation of such hook defines the appropriate theme implementation to display such an entity. Themes are registered through hook_theme [83]. In the developed module, each node type has a template to display its related entity.

4.4 Drupal Modules for Collaborative Functionalities

The following modules are responsible for the collaborative functionalities of the proposed IT-SRMCS:

- Comment [84]: This is a core Drupal module, and it provides threaded comments in contents. Its behavior was enhanced with "Re: Comment subjects" 1.4 module [85];
- Content Access 1.0 [86]: This module makes it possible to restrict access control to nodes. It uses ACL module [87] for some functionalities;
- Forum [88]: This is a core Drupal module, and it provides the forum system;
- Blog [89]: This is a core Drupal module, and it provides the blog system; and
- Poll [90]: This is a core Drupal module, and it provides the poll system.

5 Case Study

This chapter aims to evaluate the proposed tool through a case study, described in Section 5.1. A possible security analysis of the scenario is presented in Section 5.2. Section 5.3 shows the procedures to access the developed tool. Section 5.4 discusses the management of different security projects in the proposed system. There are functionalities to support IT security risk management phases, and some of them are described in Sections 5.5 (Risk Assessment), 5.6 (Risk Mitigation), and 5.7 (Evaluation and Assessment). Section 5.8 presents additional collaborative functionalities available in the proposed software. The case study results are analyzed in Section 5.9.

5.1 Case

The case to test the developed tool is a financial organization that provides a service to help customers with stock investments. The main business attraction is a client-server software that allows users to manage their financial investments, access the news on the economy, read economic market analysis by renowned professionals, and interact with economists to support investments. This service is hosted at the financial organization itself.

The technology environment is shown in Figure 5.1.

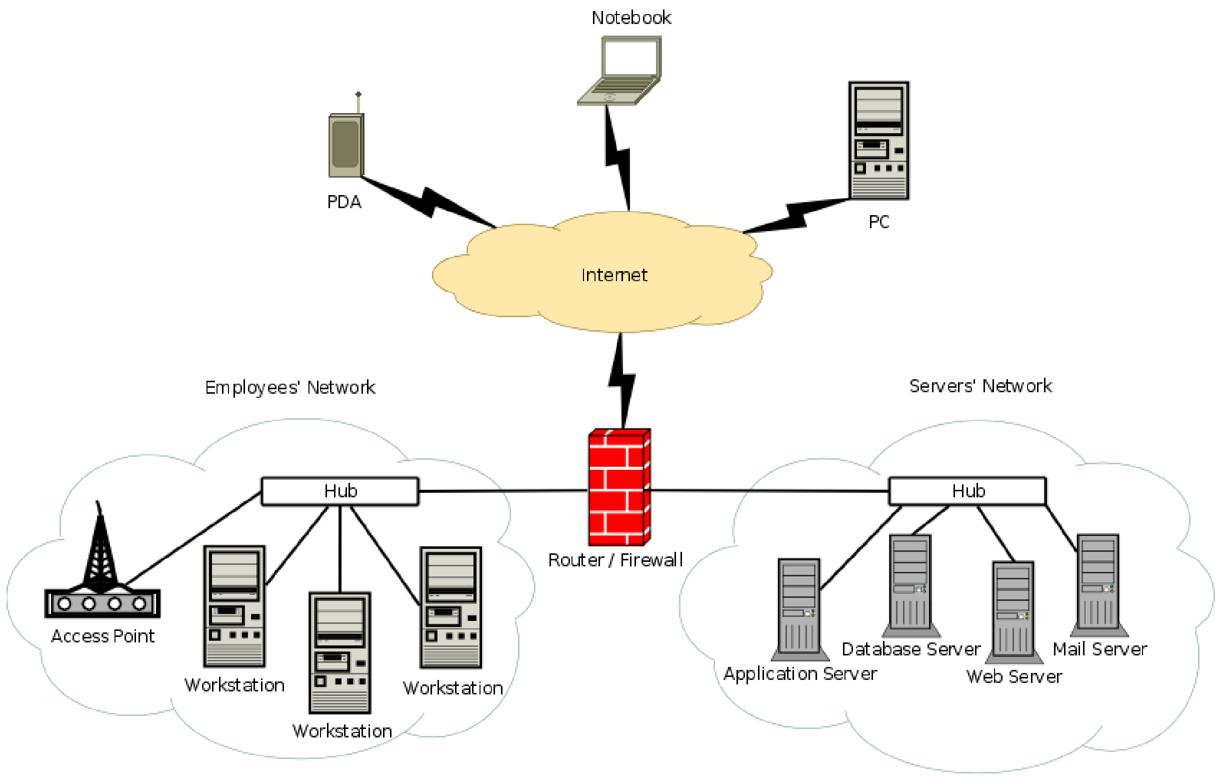


FIGURE 5.1 – Case study.

There are employees working at the financial company, in a specific network (Employees' Network), and all its workstations and printers are linked through a hub. Additionally, there is also a wireless access point linked to this hub to spread the physical working areas inside the company.

There are four servers to deal with the stock system, and all of them are hosted inside the company, in a specific network (Servers' Network). They rely on the Linux operating system with an SSH server enabled for configuration and management purposes. Additionally, they are linked through a hub.

Both internal networks are interconnected through a firewall/router (Firewall/Router). The firewall rules allow all traffic between the two internal networks without restrictions.

The main software, both the client and server sides, is written in C language. The client-side software can execute on PDAs, notebooks and conventional PCs. The server-side tool is deployed in a dedicated server (Application Server). All system-related data are stored in a MySQL database, hosted in another server (Database Server). Additionally, the customers interact with the stock system through the internet.

The communication with the clients about payment and software support are through both, the project's website (hosted at Web Server) and e-mail (hosted at Mail Server). The former server relies on a proprietary webserver service with PHP 4 support; additionally, there are a FTP daemon running to manage web files. The latter server hosts a proprietary e-mail service tool designed to operate with SMTP and POP3 protocols.

The organization has an internet link of 8 Mbps, with a static IP address, connected to the Firewall/Router, that routes the internet to the whole company (for both, servers' and employees' network). The firewall rules ensure that the hosts on the internet can access the webserver service (ports 80 and 443) and its related FTP (port 21), SMTP service (port 25), and the server-side main business application (port 6789).

5.2 Security Analysis of the Scenario

The following tables represent a security analysis of the scenario. Table 5.1 describes the assets. Table 5.2 identifies the system components. The vulnerabilities and their related system components are listed in Table 5.3. The threats and their related assets are listed in Table 5.4. The threat-sources are described in Table 5.5. The risks, represented as a relation between vulnerabilities, threat-sources and threats, are listed in Table 5.6. Finally, the safeguards and their related risks are identified in Table 5.7.

TABLE 5.1 – Case study assets.

Asset	**Description**
Main application	The main application has to be secure. The availability is an important requirement, because the stock market is very dynamic, so the clients must have appropriate support when needed, without delays.
Information in MySQL database	Client's financial information are sensitive data and its confidentiality is guaranteed by contract.
Company's reputation	As personal financial data is held by the system, only serious companies are selected by customers.

TABLE 5.2 – Case study system components.

System Component	**Description**
Router/Firewall operating system	Operating system of the router/firewall.
Router/Firewall configuration	Configuration of router and firewall rules.
Web Server operating system	Operating system of the server with webserver daemon.
Proprietary webserver daemon	Proprietary webserver and its configuration.
PHP application at the Web Server	Application, written in PHP, at the web server, used to communicate with clients.
FTP service at the Web Server	FTP service at the Web Server.
Application Server operating system	Operating system of the server with main application's server-side software.
Main application's server-side software	Server-side application, at the application server.
Mail Server operating system	Operating system of the server with mail service tool.
Mail service software (SMTP and POP3)	Mail service tool and its configuration, at mail server.
Database Server operating system	Operating system of the server with the database.
MySQL database	MySQL database, its data, and its configuration, at Database Server.
SSH service at the four servers	SSH service at all servers.
Servers' hub	Hub used to link the four servers with the Firewall/Router.
Wireless access point	Wireless access point, plugged in the hub used by employees' workstations.
Workstations	Workstations of the employees.
Employee's hub	Hub used to link employees' workstations.

Security policies	Security-related formal policies of the organization.
Personnel practices	Practices of the organization's personnel.

TABLE 5.3 – Case study vulnerabilities and their related system components.

Vulnerability	System Component
SQL Injection at the PHP application.	PHP application at the Web Server
SYN flood attacks can compromise the Web Server.	Router/Firewall configuration
Access point without protection, such as encryption.	Wireless access point
Use of FTP, a clear-text protocol, to manage web files.	FTP service at the Web Server
	Security policies
	Personnel practices
Buffer overflow at the main application server-side software.	Application Server operating system
	Main application's server-side software
Mail service without relay restrictions.	Mail service software (SMTP and POP3)
MySQL accessed with only one user by the main application: root.	MySQL database
	Main application's server-side software
All traffic is allowed between the two local networks (servers and employees).	Router/Firewall configuration
Mail service accessible from external networks.	Router/Firewall configuration
Weak passwords at all servers.	Web Server operating system
	Application Server operating system
	Mail Server operating system
	Database Server operating system
	Router/Firewall operating system
	Security policies
	Personnel practices
There is no formal password security policy.	Security policies

TABLE 5.4 – Case study threats and their related assets.

Threat	Asset
Damage to the organization's reputation.	Company's reputation
Access confidential data from database.	Information in MySQL database
Modify database data.	Information in MySQL database
Stop the communication with clients about payment and software support.	Main application
	Company's reputation
Sniffing network and getting passwords and other sensitive data.	Information in MySQL database
Connecting to the organization's LAN for local attack purposes.	Main application
	Information in MySQL database

Use the mail service to make SPAM through an AP connection.	Company's reputation
Possibility of sniffing FTP-related passwords and change web files.	Company's reputation
DoS at the application.	Main application
	Company's reputation
Malware execution on the Application Server to get passwords and confidential information such as users' investments details.	Main application
	Information in MySQL database
Possibility of sending of SPAM through the mail service.	Company's reputation
Database data can be fully accessed, including configurations.	Information in MySQL database
Lack of accountability in the database.	Information in MySQL database
Internal DoS attacks on the servers.	Main application
	Information in MySQL database
	Company's reputation
All hosts on the internet can use the mail service to send e-mails.	Company's reputation
All hosts on the internet can use the mail service to send SPAM.	Company's reputation
Possibility of cracking passwords to compromise all servers.	Main application
	Information in MySQL database
Passwords cracked once can be used for a long time.	Main application
	Information in MySQL database
Possibility of having many weak passwords inside the company.	Main application
	Information in MySQL database
Possibility of users writing their passwords on papers that are on their tables.	Main application
	Information in MySQL database
Difficulty of punishing employees for password stealing.	Company's reputation

TABLE 5.5 – Case study threat-sources.

Threat-Sources	**Description**
Attacker from the internet	An attacker, from any place, connected to the internet.
Malicious customer	A customer of the financial company that aims to attack it.
Malicious employee	An employee of the financial company that aims to attack it.
Attacker in the AP signal area	An attacker somewhere that can connect to the AP.
Malicious person visiting the organization	A person visiting the financial company that aims to attack it.

TABLE 5.6 – Case study risks (vulnerabilities x threat-sources x threats).

Risk Code	**Vulnerability**	**Threat-Source**	**Threat**
R1	SQL Injection at the PHP application.	Attacker from the internet	Access confidential data from database.
		Malicious customer	
		Malicious employee	
		Attacker in the AP signal area	
R2	SQL Injection at the PHP application.	Attacker from the internet	Modify database data.
		Malicious customer	
		Malicious employee	
		Attacker in the AP signal area	
R3	SYN flood attacks can compromise the Web Server.	Attacker from the internet	Stop the communication with clients about payment and software support.
		Malicious employee	
		Attacker in the AP signal area	
R4	SYN flood attacks can compromise the Web Server.	Attacker from the internet	Damage to the organization's reputation.
		Malicious employee	
		Attacker in the AP signal area	
R5	Access point without protection, such as encryption.	Attacker in the AP signal area	Sniffing network and getting passwords and other sensitive data.
R6	Access point without protection, such as encryption.	Attacker in the AP signal area	Connecting to the organization's LAN for local attack purposes.
R7	Access point without protection, such as encryption.	Attacker in the AP signal area	Use the mail service to make SPAM through an AP connection.
R8	Use of FTP, a clear-text protocol, to manage web files.	Malicious employee	Possibility of sniffing FTP-related passwords and change web files.
		Attacker in the AP signal area	
R9	Buffer overflow at the main application server-side software.	Malicious customer	DoS at the application.
		Malicious employee	
R10	Buffer overflow at the main application server-side software.	Malicious customer	Damage to the organization's reputation.
		Malicious employee	

R11	Buffer overflow at the main application server-side software.	Malicious customer Malicious employee	Malware execution on the Application Server to get passwords and confidential information such as users' investments details.
R12	Mail service without relay restrictions.	Attacker from the internet Malicious employee Attacker in the AP signal area	Possibility of sending of SPAM through the mail service.
R13	MySQL accessed with only one user by the main application: root.	Attacker from the internet Malicious customer Malicious employee Attacker in the AP signal area	Database data can be fully accessed, including configurations.
R14	MySQL accessed with only one user by the main application: root.	Attacker from the internet Malicious customer Malicious employee Attacker in the AP signal area	Lack of accountability in the database.
R15	All traffic is allowed between the two local networks (servers and employees).	Malicious employee Attacker in the AP signal area	Internal DoS attacks on the servers.
R16	Mail service accessible from external networks.	Attacker from the internet	All hosts on the internet can use the mail service to send e-mails.
R17	Mail service accessible from external networks.	Attacker from the internet	All hosts on the internet can use the mail service to send SPAM.
R18	Weak passwords at all servers.	Attacker from the internet Malicious employee Attacker in the AP signal area	Possibility of cracking passwords to compromise all servers.
R19	There is no formal password security policy.	Attacker from the internet Malicious employee Attacker in the AP signal area	Passwords cracked once can be used for a long time.

R20	There is no formal password security policy.	Attacker from the internet	Possibility of having many weak passwords inside the company.
		Malicious employee	
		Attacker in the AP signal area	
R21	There is no formal password security policy.	Malicious employee	It is possible for users to write their passwords on papers that are on their tables.
		Malicious person visiting the organization	
R22	There is no formal password security policy.	Malicious employee	Difficulty of punishing employees for password stealing.

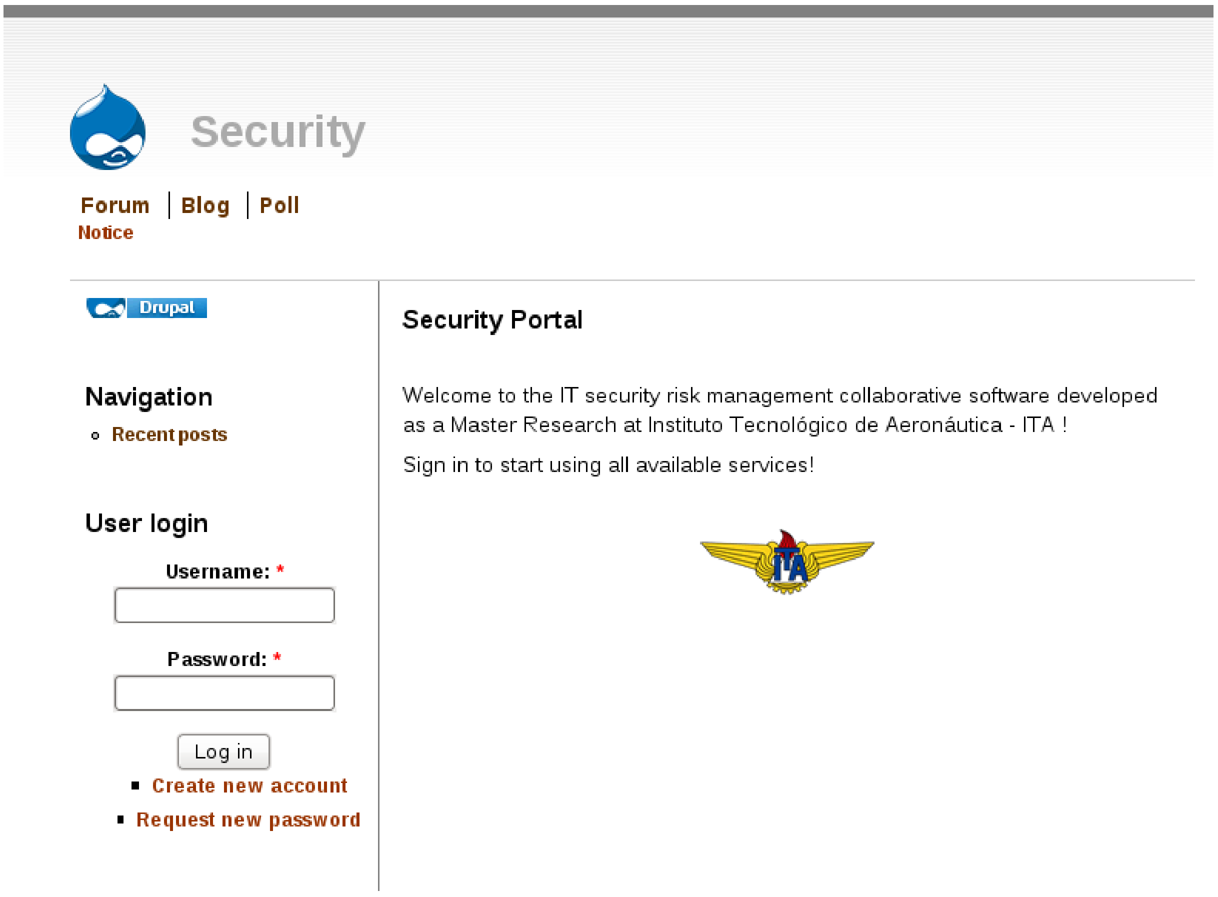
TABLE 5.7 – Case study safeguards and their related risks.

Safeguard	**Risk**
Sanitize user inputs of the PHP application at the Web Server.	R1, and R2
Configure the Firewall/Router to limit the rate of packages to the Web Server.	R3, and R4
Configure the Access Point to use secure standards with secure cryptography algorithms, such as WPA2 with TKIP or AES.	R5
Restrict accesses to the Access Point. For example, use RADIUS, and limit computers through MAC addresses.	R6, and R7
Adopt a secure protocol, such as SCP, to update web files at the Web Server.	R8
Exchange hubs by switches.	R8
Sanitize user inputs at the main application server-side software.	R9, R10, R11, R13, and R14
Fortify the security of the Application Server software environment. Examples of solutions are grsecurity/PaX, SELinux, and GCC's Stack Smashing Protector.	R9, R10, R11
Restrict relay at the proprietary mail service at Mail Server.	R12
Adopt a non-privileged MySQL user for the main application's operations.	R13, R14
Restrict rules at the Firewall/Router to filter traffic between the internal networks (employees and servers).	R15
Restrict rules at the Firewall/Router to block external accesses to the mail service.	R16, and R17

Adopt only strong passwords for the servers, and change them frequently.	R18
Develop a password security policy.	R18, R19, R20, R21, and R22

5.3 Accessing the Tool

The initial screen of the proposed solution is depicted by Figure 5.2.


Security

Forum | Blog | Poll
Notice

Drupal

Navigation
- Recent posts

User login
Username: *
Password: *
Log in
- Create new account
- Request new password

Security Portal

Welcome to the IT security risk management collaborative software developed as a Master Research at Instituto Tecnológico de Aeronáutica - ITA !

Sign in to start using all available services!

FIGURE 5.2 – Initial screen.

From this screen it is possible to access all available features, but some of them, such as the developed IT security risk management software, require a logged-in user. Figure 5.3 shows a login procedure through the login area of the main screen.

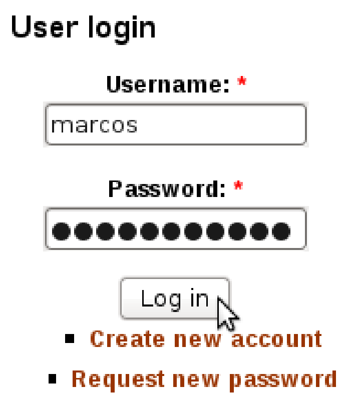


FIGURE 5.3 – User login.

Once the user has logged-in, the link to access the developed software appears. Figure 5.4 depicts the first screen of the developed IT security risk management software, that lists the available projects for the user.

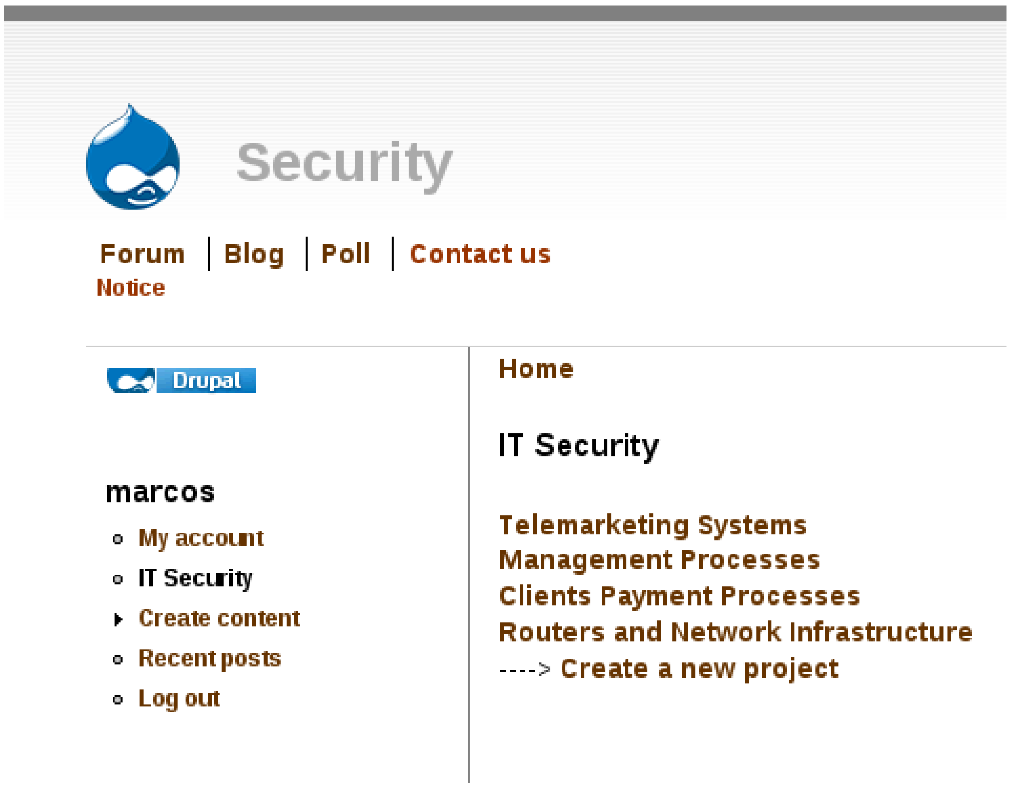
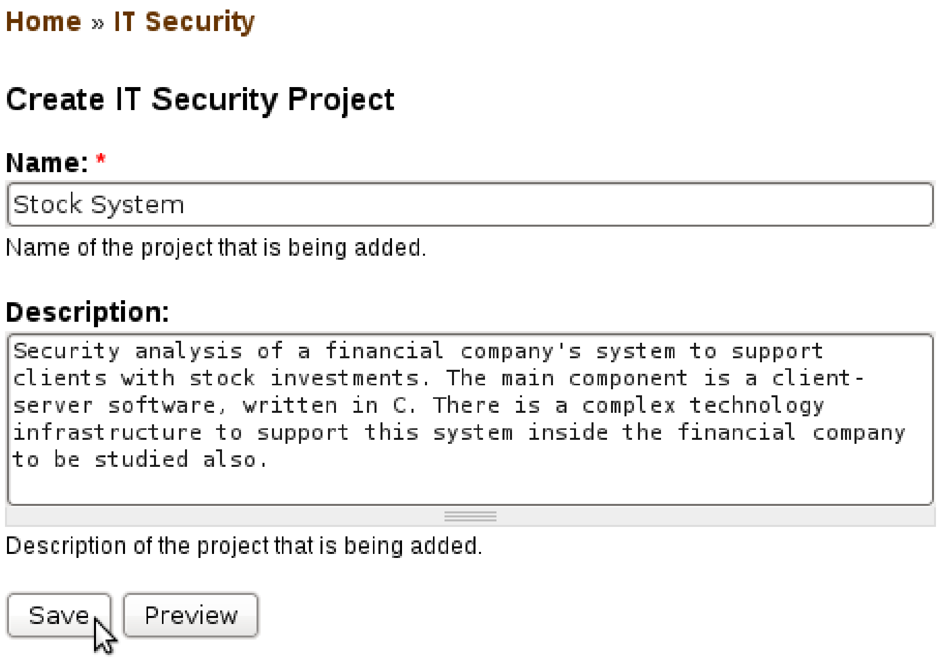


FIGURE 5.4 – Screen listing the projects available to the user.

5.4 Project Management

In order to handle the scenario described in Section 5.1, a specific project for this purpose must be created, as shown in Figure 5.5.


Home » IT Security

Create IT Security Project

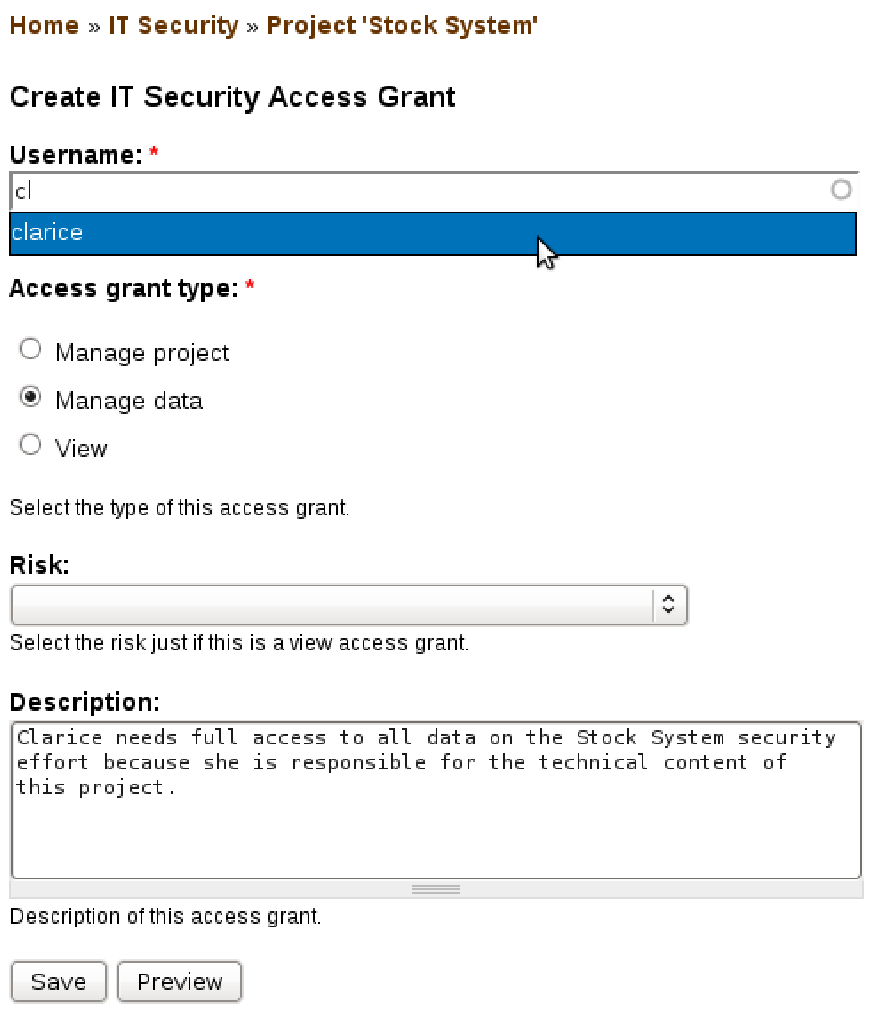
Name: *
Stock System
Name of the project that is being added.

Description:
Security analysis of a financial company's system to support clients with stock investments. The main component is a client-server software, written in C. There is a complex technology infrastructure to support this system inside the financial company to be studied also.
Description of the project that is being added.

FIGURE 5.5 – Project creation screen.

The desired project (in this case, the newly created one) must be selected to start working on it.

The user that creates a project is automatically marked with the “manage project” grant. The next step is to add the users to the project with their respective permissions. Figure 5.6 shows an example when the project owner adds an user with “manage data” grant.


Home » IT Security » Project 'Stock System'

Create IT Security Access Grant

Username: *
cl
clarice

Access grant type: *

Manage project
[✓ Manage data]
View

Select the type of this access grant.

Risk:

Select the risk just if this is a view access grant.

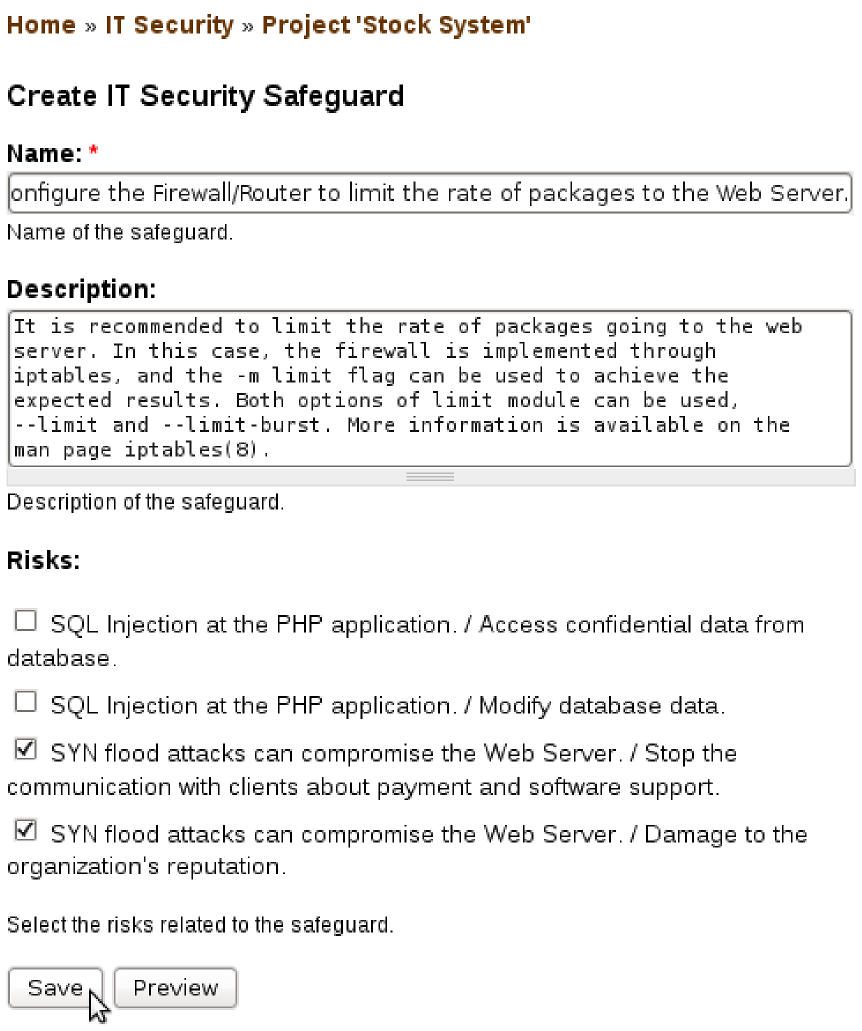
Description:
Clarice needs full access to all data on the Stock System security effort because she is responsible for the technical content of this project.

Description of this access grant.

FIGURE 5.6 – Access grant creation screen.

5.5 Risk Assessment Phase

This phase is responsible for the security analysis of the stock system. So, the security model presented in Section 5.1 is developed and documented with support of the proposed tool. All operations are similarly in all different concepts. Figure 5.7 depicts the procedure of creating a safeguard when the information of Section 5.1 was partially inserted in the system.



Home » IT Security » Project 'Stock System'

Create IT Security Safeguard

Name: *

onfigure the Firewall/Router to limit the rate of packages to the Web Server.

Name of the safeguard.

Description:

```
It is recommended to limit the rate of packages going to the web
server. In this case, the firewall is implemented through
iptables, and the -m limit flag can be used to achieve the
expected results. Both options of limit module can be used,
--limit and --limit-burst. More information is available on the
man page iptables(8).
```

Description of the safeguard.

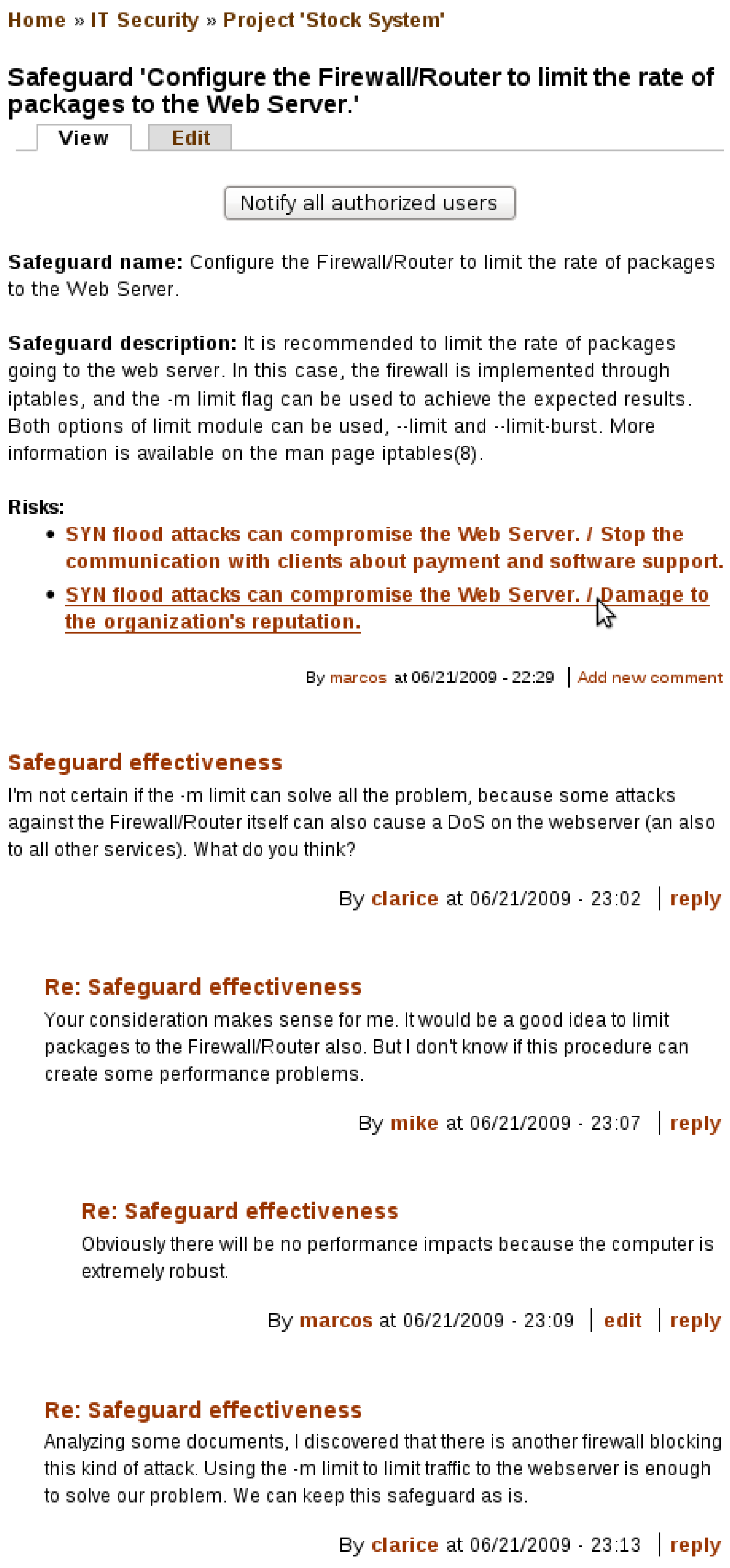
Risks:

- ☐ SQL Injection at the PHP application. / Access confidential data from database.
- ☐ SQL Injection at the PHP application. / Modify database data.
- ☑ SYN flood attacks can compromise the Web Server. / Stop the communication with clients about payment and software support.
- ☑ SYN flood attacks can compromise the Web Server. / Damage to the organization's reputation.

Select the risks related to the safeguard.

FIGURE 5.7 – Safeguard creation screen.

Display screens have all concepts' information, with links to other directly related concepts. For example, a safeguard screen is linked to all its addressed risks. At the risk assessment stage, there is a strong need for discussions related to each of the identified items of information (concepts), to validate and refine the analysis. The screens that display concepts are also used for group discussions. Additionally, there is a button used to notify all authorized users that some new important information in that concept is available, such as a new comment or sensitive data change. Figure 5.8 depicts a concept display screen, using the example of the safeguard created in Figure 5.7.



Home » IT Security » Project 'Stock System'

Safeguard 'Configure the Firewall/Router to limit the rate of packages to the Web Server.'

View Edit

Notify all authorized users

Safeguard name: Configure the Firewall/Router to limit the rate of packages to the Web Server.

Safeguard description: It is recommended to limit the rate of packages going to the web server. In this case, the firewall is implemented through iptables, and the -m limit flag can be used to achieve the expected results. Both options of limit module can be used, --limit and --limit-burst. More information is available on the man page iptables(8).

Risks:
- SYN flood attacks can compromise the Web Server. / Stop the communication with clients about payment and software support.
- SYN flood attacks can compromise the Web Server. / Damage to the organization's reputation.

By marcos at 06/21/2009 - 22:29 | Add new comment

Safeguard effectiveness

I'm not certain if the -m limit can solve all the problem, because some attacks against the Firewall/Router itself can also cause a DoS on the webserver (an also to all other services). What do you think?

By clarice at 06/21/2009 - 23:02 | reply

Re: Safeguard effectiveness

Your consideration makes sense for me. It would be a good idea to limit packages to the Firewall/Router also. But I don't know if this procedure can create some performance problems.

By mike at 06/21/2009 - 23:07 | reply

Re: Safeguard effectiveness

Obviously there will be no performance impacts because the computer is extremely robust.

By marcos at 06/21/2009 - 23:09 | edit | reply

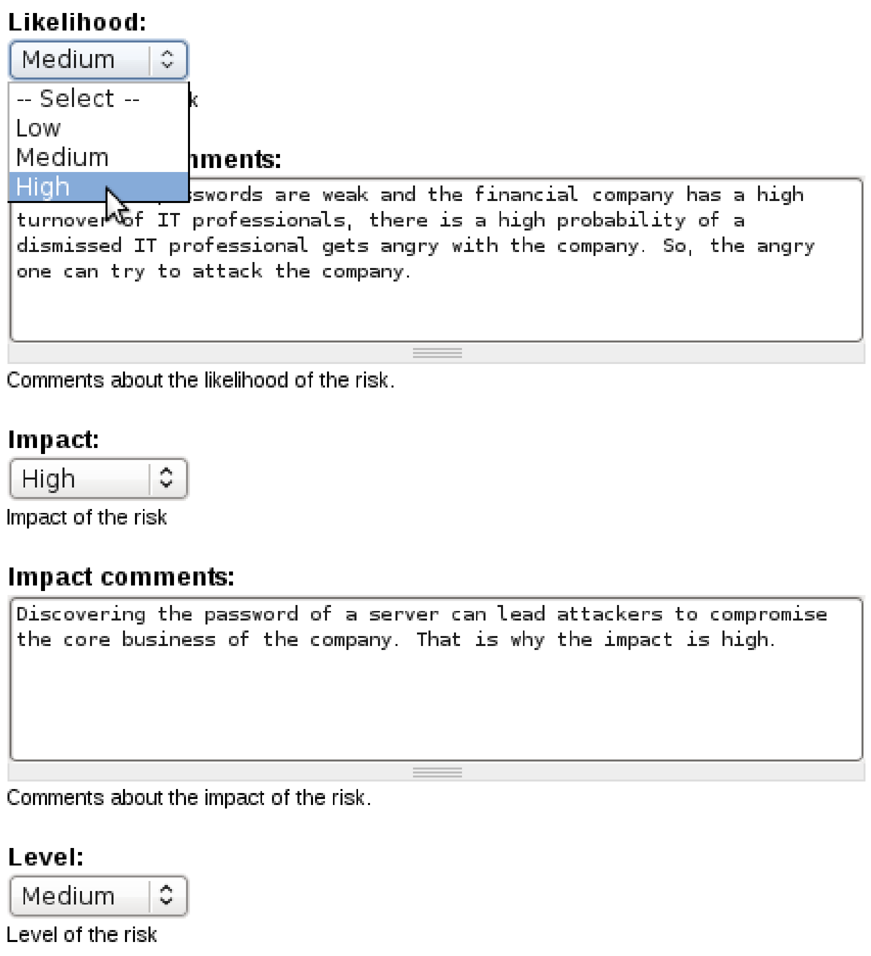
Re: Safeguard effectiveness

Analyzing some documents, I discovered that there is another firewall blocking this kind of attack. Using the -m limit to limit traffic to the webserver is enough to solve our problem. We can keep this safeguard as is.

By clarice at 06/21/2009 - 23:13 | reply

FIGURE 5.8 – Safeguard visualization screen.

At a group discussion related to a risk, between security analysts and financial company's professionals, a possible mistake in a likelihood value definition was identified. As a result, a consensus about the need for an update was arrived at. The "Edit" tab in a concept visualization, such as in Figure 5.8, appears only for authorized users, allowing them to update some information about a concept. Figure 5.9 illustrates the risk likelihood value change procedure.



Likelihood:

Medium

-- Select --
Low
Medium
High

k

nments:

```
swords are weak and the financial company has a high
turnover of IT professionals, there is a high probability of a
dismissed IT professional gets angry with the company. So, the angry
one can try to attack the company.
```

Comments about the likelihood of the risk.

Impact:

High

Impact of the risk

Impact comments:

```
Discovering the password of a server can lead attackers to compromise
the core business of the company. That is why the impact is high.
```

Comments about the impact of the risk.

Level:

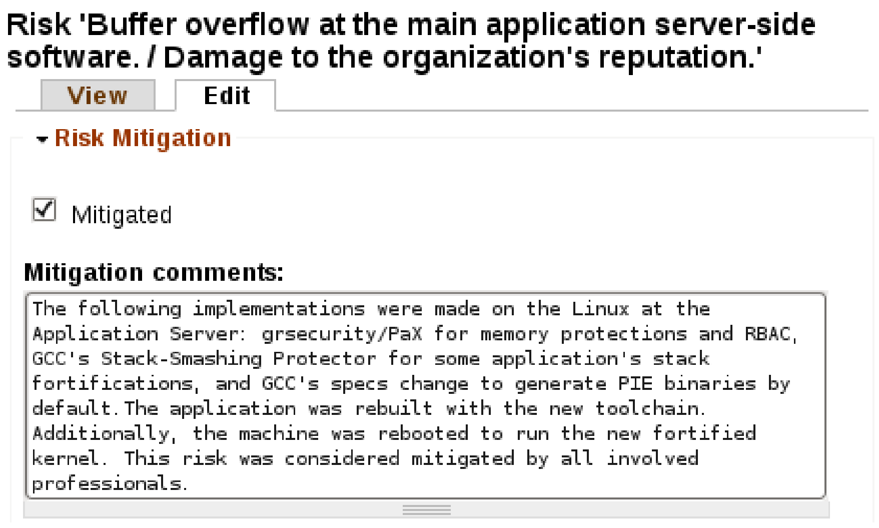
Medium

Level of the risk

FIGURE 5.9 – Likelihood update.

5.6 Risk Mitigation Phase

Through the group discussion mechanism it is possible, for the involved professionals, to manage the implementation of a safeguard. There was consensus, in a risk discussion, that appropriate safeguards were correctly implemented, and the risk was mitigated. The mitigation can be done, by authorized users, through a risk's edit screen. Figure 5.10 shows the mitigation of a risk.


Risk 'Buffer overflow at the main application server-side software. / Damage to the organization's reputation.'

☑ Mitigated

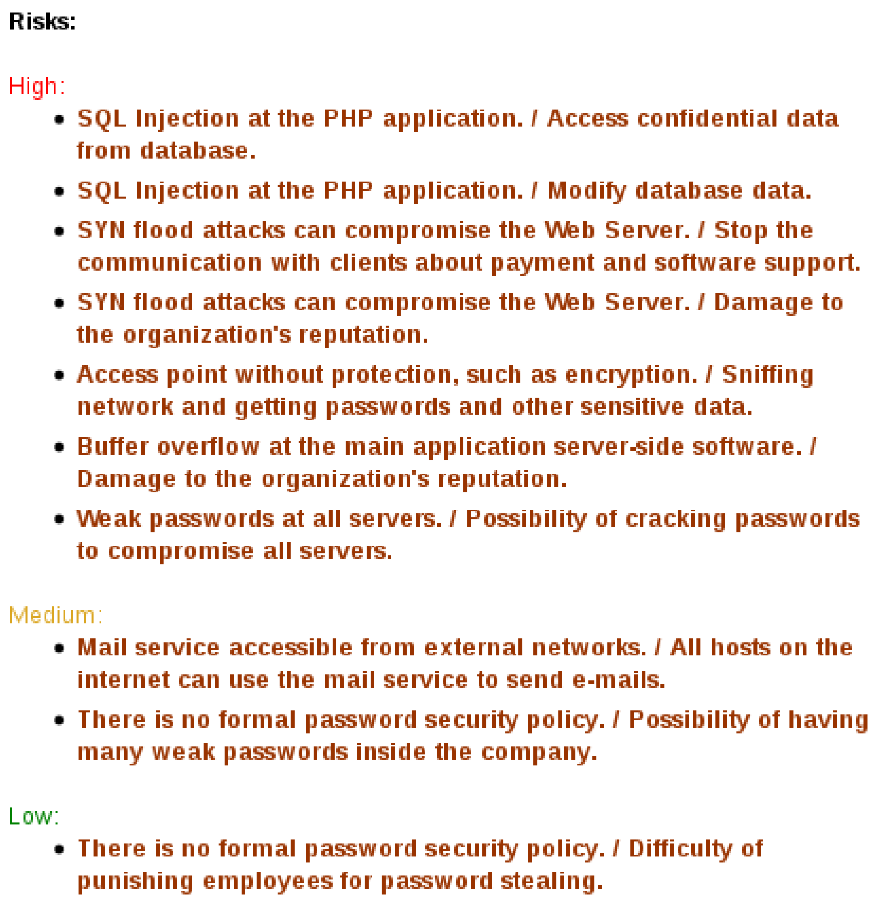
Mitigation comments:

The following implementations were made on the Linux at the Application Server: grsecurity/PaX for memory protections and RBAC, GCC's Stack-Smashing Protector for some application's stack fortifications, and GCC's specs change to generate PIE binaries by default. The application was rebuilt with the new toolchain. Additionally, the machine was rebooted to run the new fortified kernel. This risk was considered mitigated by all involved professionals.

FIGURE 5.10 – Risk mitigation.

5.7 Evaluation and Assessment Phase

During the two main phases of a security risk management process, the proposed IT-SRMCS can be used to analyze the current status of the security effort. The data filtering feature is accessible through a link at the main project's screen. There are some options, such as risks classification according their levels, and determination of the system components with major incidence of vulnerabilities. Figures 5.11 and 5.12, respectively, depict both data filtering options when the analysis of Section 5.2 was partially modeled in the software.



Risks:

High:

- SQL Injection at the PHP application. / Access confidential data from database.
- SQL Injection at the PHP application. / Modify database data.
- SYN flood attacks can compromise the Web Server. / Stop the communication with clients about payment and software support.
- SYN flood attacks can compromise the Web Server. / Damage to the organization's reputation.
- Access point without protection, such as encryption. / Sniffing network and getting passwords and other sensitive data.
- Buffer overflow at the main application server-side software. / Damage to the organization's reputation.
- Weak passwords at all servers. / Possibility of cracking passwords to compromise all servers.

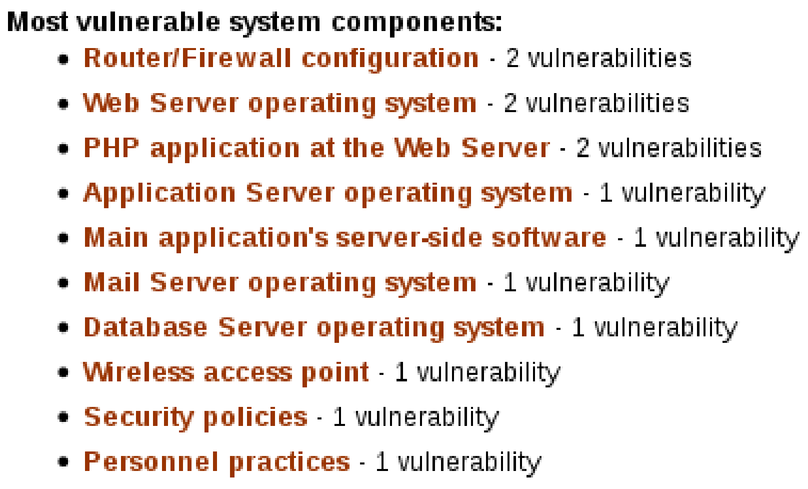
Medium:

- Mail service accessible from external networks. / All hosts on the internet can use the mail service to send e-mails.
- There is no formal password security policy. / Possibility of having many weak passwords inside the company.

Low:

- There is no formal password security policy. / Difficulty of punishing employees for password stealing.

FIGURE 5.11 – Risks classified according their levels.



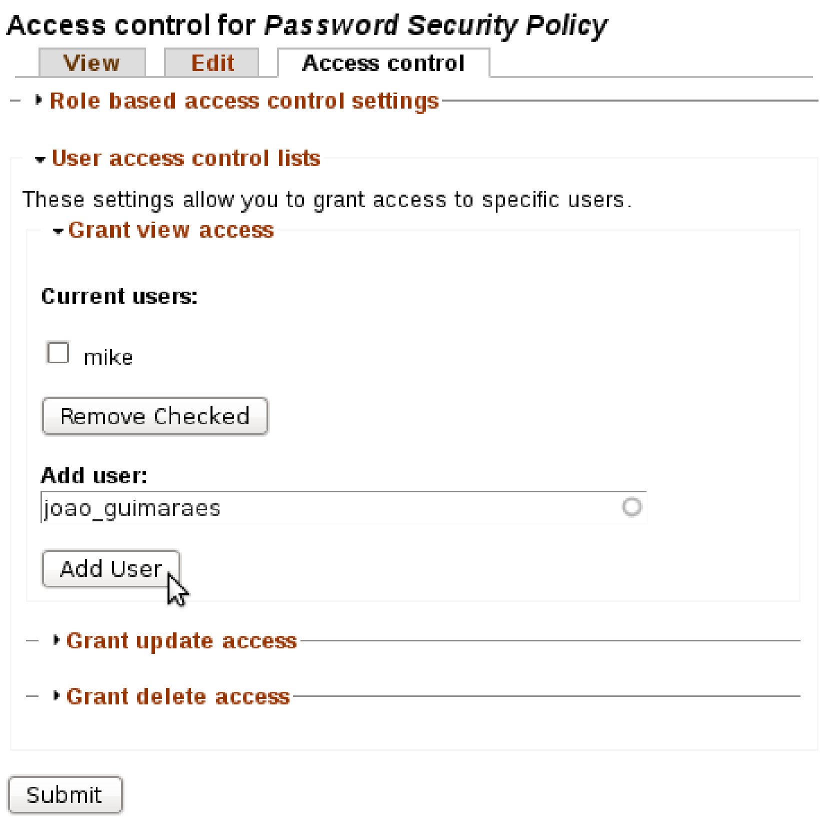
Most vulnerable system components:

- Router/Firewall configuration - 2 vulnerabilities
- Web Server operating system - 2 vulnerabilities
- PHP application at the Web Server - 2 vulnerabilities
- Application Server operating system - 1 vulnerability
- Main application's server-side software - 1 vulnerability
- Mail Server operating system - 1 vulnerability
- Database Server operating system - 1 vulnerability
- Wireless access point - 1 vulnerability
- Security policies - 1 vulnerability
- Personnel practices - 1 vulnerability

FIGURE 5.12 – Most vulnerable system components.

5.8 Additional Collaborative Features

It is possible to create documents collaboratively, under a tight access control policy. Figure 5.13 shows that, when creating a new document, it is possible to refine the users and their respective access grants to it.


Access control for *Password Security Policy*

View | Edit | Access control

▸ Role based access control settings

▾ User access control lists

These settings allow you to grant access to specific users.

▾ Grant view access

Current users:

☐ mike

Remove Checked

Add user:

joao_guimaraes

Add User

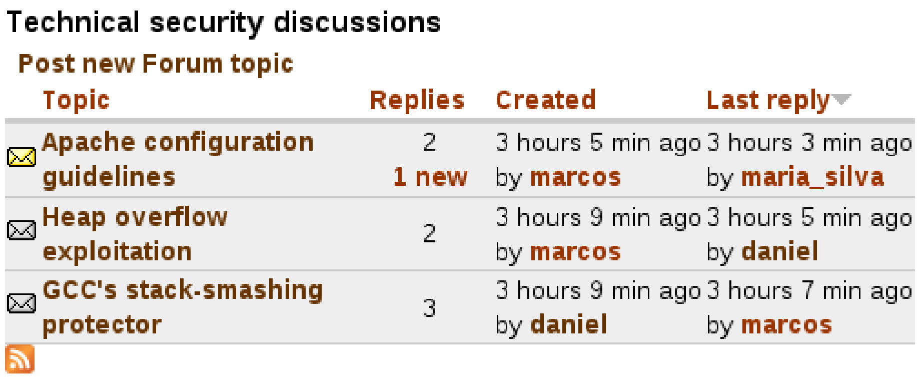
▸ Grant update access

▸ Grant delete access

Submit

FIGURE 5.13 – Collaborative content creation access control.

There is a forum, for general security discussions, as shown in Figure 5.14. Each user has their own blog, as depicted by Figure 5.15. Another example of collaboration is the poll functionality, as shown in Figure 5.16.

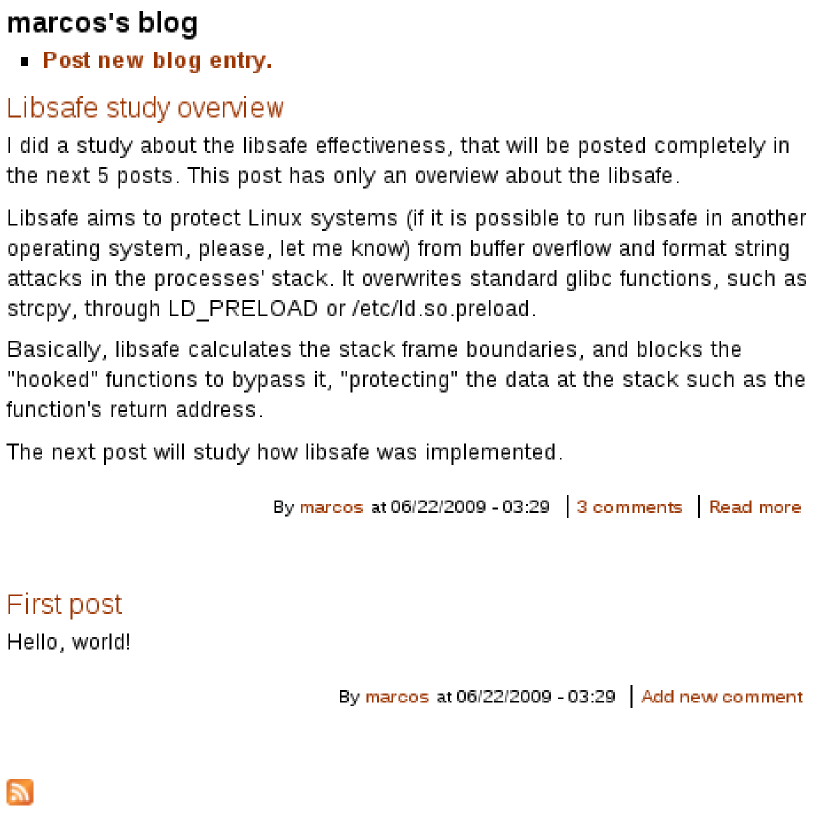


Technical security discussions

Post new Forum topic

	Topic	Replies	Created	Last reply
✉	Apache configuration guidelines	2 1 new	3 hours 5 min ago by marcos	3 hours 3 min ago by maria_silva
✉	Heap overflow exploitation	2	3 hours 9 min ago by marcos	3 hours 5 min ago by daniel
✉	GCC's stack-smashing protector	3	3 hours 9 min ago by daniel	3 hours 7 min ago by marcos

FIGURE 5.14 – Forum.



marcos's blog

- Post new blog entry.

Libsafe study overview

I did a study about the libsafe effectiveness, that will be posted completely in the next 5 posts. This post has only an overview about the libsafe.

Libsafe aims to protect Linux systems (if it is possible to run libsafe in another operating system, please, let me know) from buffer overflow and format string attacks in the processes' stack. It overwrites standard glibc functions, such as strcpy, through LD_PRELOAD or /etc/ld.so.preload.

Basically, libsafe calculates the stack frame boundaries, and blocks the "hooked" functions to bypass it, "protecting" the data at the stack such as the function's return address.

The next post will study how libsafe was implemented.

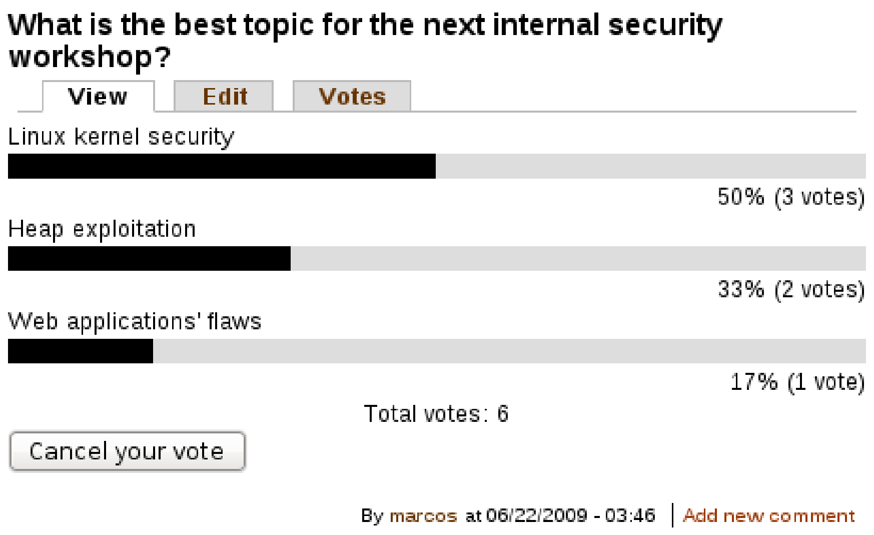
By marcos at 06/22/2009 - 03:29 | 3 comments | Read more

First post

Hello, world!

By marcos at 06/22/2009 - 03:29 | Add new comment

FIGURE 5.15 – Blog.


What is the best topic for the next internal security workshop?

Linux kernel security
50% (3 votes)

Heap exploitation
33% (2 votes)

Web applications' flaws
17% (1 vote)

Total votes: 6

By marcos at 06/22/2009 - 03:46 | Add new comment

FIGURE 5.16 – Poll.

5.9 Analysis of the Results

The IT-SRMCS proposed by this research provides a user-friendly interface that makes it easy to create a security model of a company. Additionally, it is simple to analyze the available data, such as higher non-addressed risks and most vulnerable system components, in order to prioritize security actions and keep the organization with an acceptable security level.

The proposed software is multi-user, and the security model can be continuously changed. The information can be kept updated even in the most dynamic environments. Behind each operation in the system, there is a robust access control mechanism filtering the information according to its rules. The rules schema is also compatible with dynamic environments. Regarding information updates in a dynamic environment, manage data and manage project access grants are generic, and do not require rule modifications; the view access grant only requires changes when a new risk is added, because it is risk-oriented and all internal risk modifications are transparent for the access control mechanism. Regarding the high turnover of professionals in organizations, new rules are only needed when new professionals have been assigned to a security effort.

An important contribution of this research is the group discussions: there is no need for scheduling meetings, optimizing the expected results. The users can interact with each other about the concepts to validate information, improve the security model, and manage security actions. Additionally, all discussions are kept in the system for future analysis and also to act as documentation about procedures and ideas.

The proposed software is and relies only on open-source components, allowing the organization to audit the whole system where their critical security information is stored. Another important characteristic is the web-based approach, that makes the system very portable in order to be used by the most diverse kinds of organizations.

It is possible to write articles collaboratively, with a tight access control mechanism. This is useful in creating and maintaining security policies, specifying security configuration guidelines, and running brainstorming sessions about new safeguards or hardening needs.

The available forum is a place for general security discussions, such as security principles, attacking techniques and related countermeasures, and hardening practices. The benefits for the organization are to improve the global security knowledge, stimulate security studying, unite personnel, and spread the importance of security for the organization's business.

There are many other collaborative features, such as a blog, when each user can write their own articles and receive comments about them, and a poll system, to get statistics on the users perceptions.

Due to its high variety of functionalities, the system may be used as the organization's main security repository. This repository can provide secure storage for IT security related documents and an environment to support security workers' interactions.

6 Conclusions

IT security risk management is a collaborative task, and its content is constantly changing due to the dynamic IT environment. The process is improved if supported by a specialized collaborative tool.

A software was developed, extending a widely-used CMS (Drupal), to securely support a NIST-based IT security risk management process, in a continuous and collaborative way, with a robust access control mechanism due to the sensitivity level of the involved information. This tool enables an effective risk management in all stages of a SDLC through a continuous collaboration among involved professionals. The proposed tool meets the objectives of this research.

6.1 Contributions

The proposed tool provides a secure environment for group communication, in such a way that conventional meetings can be avoided in many cases, and all discussions content is kept documented for future analysis. Additionally, the IT risk management effort is handled in a continuous way, with a robust access control mechanism compatible with the most dynamic environments.

This research has also identified that there are benefits in adopting a CMS as a base platform for a risk management support tool, because of:

- CMS features inheritance: A CMS has many features that can be inherited to improve the system's features, such as group discussion, users interaction, and collaborative artifacts development environment; and
- Tool development efficiency: The development effort can be concentrated just on activities related specifically to the security risk management process. Aspects such as collaborative environment, user management, and presentation layer, are very mature on a robust CMS, and can be inherited.

Additionally, the developed tool is and only relies on open-source components, making it possible for the organization to audit the whole software that their confidential security data will be stored on. Due to the sensitive data manipulated by the proposed software, security practices were also applied in the development, such as user inputs treatment to difficult exploitations.

The developed tool can also be used by other researches as a base system. For example, it is possible for a security knowledge base survey to implement the developed concepts over the proposed tool, achieving clearer results and improving the base system (the proposed tool).

6.2 Recommendations for Future Works

The developed software, due to its software engineering modeling, can be easily extended with change impacts control.

Some recommendations for future work are:

- Match security incidents with risks, in order to validate the risks coverage and the security solutions' effectivenes;
- Incident handling features with integration with applications such as OSSIM [91], in a way that detected attacks could be automatically matched against the existent risks. The result could be new risks undiscovered at that time, security solutions change, or accuracy validation;
- Implementation of a robust knowledge base to help security professionals work inferring vulnerabilities and safeguards. Available public vulnerability databases, such as OSVDB [20], could also be used to infer vulnerabilities;
- Implementation of a knowledge base of best practices and standards. The idea is to add support for governance and compliance practices;
- Integration with other security methods, such as threat tree models, in order to structure even more the security analysis;
- Integration with security metrics to improve even more the risk analysis; and
- Use the proposed tool as a base system for other security-related researches. Examples of possible research areas are collaborative models and ontology.

Bibliography

[1] CENTRO DE ESTUDOS, RESPOSTA E TRATAMENTO DE INCIDENTES DE SEGURANÇA NO BRASIL. *Estatísticas dos incidentes reportados ao CERT.br*. 2009. Available at: <http://www.cert.br/stats/incidentes/>. Accessed on: 24 Apr. 2009. ix, 17, 18

[2] ALLEN, J. H. et al. *Software security engineering*: a guide for project managers. [S.l.]: Addison Wesley Professional, 2008. ISBN 978-0-321-50917-8. ix, 17, 28

[3] STONEBURNER, G.; GOGUEN, A.; FERINGA, A. *SP 800-30*: risk management guide for information technology systems. 2002. Available at: <http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf>. Accessed on: 01 July 2009. ix, xi, xvi, 21, 24, 25, 28, 29, 30, 31, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 56, 60, 132, 133, 134, 135, 136, 137

[4] BAUER, M. Paranoid penguin: taking a risk-based approach to linux security. *Linux Journal*, n. 129, p. 14, 2005. 18

[5] FEDERAL OFFICE FOR INFORMATION SECURITY. *GSTOOL*: grundschutz tool. Available at: <http://www.bsi.de/english/gstool/index.htm>. Accessed on: 01 July 2009. 19, 54

[6] MODULO. Solutions for GRC. *Modulo risk manager*. Montclair, 2009. Available at: <http://www.modulo.com/products/modulo-risk-manager-overview.jsp>. Accessed on: 01 July 2009. 19, 55

[7] FORTIFY SOFTWARE. *Fortify 360*. San Mateo, 2009. Available at: <http://www.fortify.com/products/fortify-360>. Accessed on: 01 July 2009. 19, 55

[8] FORTIFY SOFTWARE. *SSA Governance*. San Mateo, 2009. Available at: <http://www.fortify.com/products/fortify-360/governance.jsp>. Accessed on: 01 July 2009. 19, 55

[9] CARNEGIE MELLON UNIVERSITY. CERT Coordination Center. *OCTAVE: Operationally Critical Threat, Asset, and Vulnerability Evaluation*. Pittsburgh, 2009a. Available at: <http://www.cert.org/octave>. Accessed on: 24 Apr. 2009. 21, 24, 50, 51

[10] MICROSOFT CORPORATION. *Security risk management guide v1.2*. 2006. Available at: <http://go.microsoft.com/fwlink/?linkid=32050>. Accessed on: 01 July 2009. 21, 24, 50, 51

[11] BISHOP, M. *Computer security*: art and science. [S.l.]: Addison Wesley, 2002. ISBN 0-201-44099-7. 21, 25

[12] UNITED STATES. Department of Commerce. National Institute of Standards and Technology. An introduction to computer security: the NIST handbook. Gaithesburg, 1995. (S.P.800-12). Available at: <http://csrc.nist.gov/publications/nistpubs/800-12/handbook.pdf>. Accessed on: 01 July 2009. 22, 23, 24, 25, 26, 27, 28

[13] LONGMAN dictionary of contemporary english. Essex:Pearson/Longman, 2003. ISBN 0-582-50664-6. 23

[14] JONES, A.; ASHENDEN, D. *Risk management for computer security*: protecting your network and information assets. [S.l.]: Elsevier, 2005. ISBN 0-7506-7795-3. 25

[15] INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. *ISO/IEC 13335-1:2004*: information technology: security techniques: management of information and communications technology security: part 1: concepts and models for information and communications technology security management. Geneva, 2004. 25

[16] CONSEJO SUPERIOR DE ADMINISTRACIÓN ELECTRÓNICA. Higher Council For Electronic Government. *MAGERIT version 2*: methodology for information systems risk analysis and management. Madrid, 2006. Available at: <http://www.csi.map.es/csi/pdf/magerit_v2/magerit_methode_en_v11.pdf>. Accessed on: 01 July 2009. 25, 26

[17] UNITED STATES. Department of Commerce. National Institute of Standards and Technology. *FIPS 200: minimum security requirements for Federal Information and information systems*. Gaithesburg, Mar. 2006. Available at: <http://csrc.nist.gov/publications/fips/fips200/FIPS-200-final-march.pdf>. Accessed on: 01 July 2009. 25, 47

[18] MCGRAW, G. *Software security*: building security in. [S.l.]: Addison Wesley Professional, 2006. ISBN 0-321-35670-5. 28

[19] UNITED STATES. Department of Commerce. National Institute of Standards and Technology. *National vulnerability database*. Gaithesburg, 2009. Available at: <http://nvd.nist.gov>. Accessed on: 24 Apr. 2009. 36

[20] OPEN SOURCE VULNERABILITY DATABASE. *Open Source vulnerability database*. 2008. Available at: <http://osvdb.org>. Accessed on: 24 Apr. 2009. 36, 124

[21] UNITED STATES. Department of Commerce. National Institute of Standards and Technology. *Recommended security controls for federal information systems*. Gaithesburg, 2007. (SP 800-53 Rev. 2). Available at: <http://csrc.nist.gov/publications/nistpubs/800-53-Rev2/sp800-53-rev2-final.pdf>. Accessed on: 01 July 2009. 47, 70, 138, 140, 144, 147

[22] CARNEGIE MELLON UNIVERSITY. CERT Coordination Center. *OCTAVE method*. Pittsburgh, 2009b. Available at: <http://www.cert.org/octave/octavemethod.html>. Accessed on: 24 Apr. 2009. 50, 51

[23] CARNEGIE MELLON UNIVERSITY. CERT Coordination Center. *OCTAVE-S*. Pittsburgh, 2009c. Available at: <http://www.cert.org/octave/octaves.html>. Accessed on: 24 Apr. 2009. 51

[24] CARNEGIE MELLON UNIVERSITY. CERT Coordination Center. *OCTAVE allegro*. Pittsburgh, 2009d. Available at: <http://www.cert.org/octave/allegro.html>. Accessed on: 24 Apr. 2009. 51

[25] MERCER, D. *Building powerful and robust websites with Drupal 6*. [S.l.]: Packt Publ., 2008. ISBN 978-1-847192-97-4. 52, 76

[26] BRAMPTON, M. *PHP5 CMS framework development*. [S.l.]: Packt Publ., 2008. ISBN 978-1-847193-57-5. 52, 53

[27] PHP GROUP. *PHP*: hypertext processor. 2009. Available at: <http://www.php.net>. Accessed on: 01 July 2009. 53, 66

[28] FEDERAL OFFICE FOR INFORMATION SECURITY (BSI). *IT-Grundschutz*. Bonn. Available at: <http://www.bsi.de/english/gshb/index.htm>. Accessed on: 01 July 2009. 54

[29] FEDERAL OFFICE FOR INFORMATION SECURITY (BSI). *BSI-Standards*. Bonn. Available at: <http://www.bsi.de/english/publications/bsi_standards/index.htm>. Accessed on: 01 July 2009. 54

[30] FEDERAL OFFICE FOR INFORMATION SECURITY (BSI). *IT-Grundschutz Tool - GSTOOL - Download*. Bonn. Available at: <http://www.bsi.de/english/gstool/download.htm>. Accessed on: 01 July 2009. 54

[31] MITCHELL, S. L.; SWITZER, C. S. *GRC capability model*: red book: version 2.0: Basic Member Edition. 2009. Available at: <http://www.oceg.org/download/rb2-limited>. Accessed on: 01 July 2009. 54

[32] FORTIFY SOFTWARE. *Fortify SSA governance module*. San Mateo, 2009. Available at: <http://www.fortify.com/servlet/downloads/public/Fortify_SSA_Governance_Module.pdf>. Accessed on: 01 July 2009. 55

[33] GÓRSKI, J.; MILER, J. *Riskguide project*. Gdansk, 2006. Available at: <http://iag.pg.gda.pl/RiskGuide>. Accessed on: 01 July 2009. 55

[34] MILER, J.; GÓRSKI, J. *Software support for collaborative risk management*. Gdansk, 2001. Available at: <http://iag.pg.gda.pl/RiskGuide/papers/Miler-Gt_for_collaborative_RM.pdf>. Accessed on: 01 July 2009. 55

[35] _____. *Implementing risk management in software projects*. Gdansk, 2001. Available at: <http://iag.pg.gda.pl/RiskGuide/papers/Miler-Gsk_management.pdf>. Accessed on: 01 July 2009. 55

[36] ______. *Towards an integrated environment for risk management in distributed software projects*. Gdansk, 2002. Available at: <http://iag.pg.gda.pl/RiskGuide/papers/Gegrated_environment.pdf>. Accessed on: 01 July 2009. 55

[37] ______. *Supporting team risk management in software procurement and development projects*. Gdansk, 2002. Available at: <http://iag.pg.gda.pl/RiskGuide/papers/Miler-G_risk_management.pdf>. Accessed on: 01 July 2009. 55

[38] WILLIAMS, L.; GEGICK, M.; MENEELY, A. Protection poker: structuring software security risk assessment and knowledge transfer. In: *ENGINEERING secure software and sytems*. Springer: Berlin, 2008. (Lecture Notes in Computer Science, 5429). P. 122-134. 56

[39] GRENNING, J. *Planning poker or how to avoid analysis paralysis while release planning*. Hawthorn Woods, 2002. Available at: <http://www.renaissancesoftware.net/papers/44-planning-poker.html>. Accessed on: 01 July 2009. 56

[40] JACOBSON, I.; BOOCH, G.; RUMBAUGH, J. *The unified software development process*. [S.l.]: Addison-Wesley Professional, 1999. ISBN 0201571692. 58

[41] DRUPAL. *Drupal*. 2009. Available at: <http://www.drupal.org>. Accessed on: 01 July 2009. 60, 64, 66

[42] VIEGA, J.; MCGRAW, G. *Building secure software*. [S.l.]: Addison-Wesley, 2002. ISBN 0-201-72152-X. 61

[43] SUN MICROSYSTEMS. *MySQL database*. 2009. Available at: <http://www.mysql.com>. Accessed on: 01 July 2009. 62, 66, 76

[44] DEBIAN. Software in the Public Interest. *Debian GNU/Linux*. 2009. Available at: <http://www.debian.org>. Accessed on: 01 July 2009. 62, 66

[45] FREE SOFTWARE FOUNDATION. *GNU*. 2009. Available at: <http://www.gnu.org>. Accessed on: 01 July 2009. 62, 66

[46] LINUX KERNEL ORGANIZATION. *Linux Kernel*. 2007. Available at: <http://www.kernel.org>. Accessed on: 01 July 2009. 62, 66

[47] LARMAN, C. *Applying UML and patterns*: an introduction to object-oriented analysis and design and the unified process. [S.l.]: Prentice Hall PTR, 2002. ISBN 0130925691. 65, 74, 81, 86, 87

[48] OESTEREICH, B. *Developing software with UML*: object-oriented analysis and design in practice. [S.l.]: Pearson Education, 2002. ISBN 0-201-75603-X. 65

[49] BOOCH, G. et al. *Object-oriented analysis and design with applications*. [S.l.]: Addison-Wesley, 2007. ISBN 0-201-89551-X. 65

[50] APACHE SOFTWARE FOUNDATION. *Apache HTTP server project*. 2009. Available at: <http://www.gnu.org>. Accessed on: 01 July 2009. 66

[51] CHANGE VISION. *JUDE/community*. 2009. Available at: <http://jude.change-vision.com/jude-web/product/community.html>. Accessed on: 01 July 2009. 66

[52] VIM. *Vim*. Available at: <http://www.vim.org>. Accessed on: 01 July 2009. 66

[53] DORLAND, Eric. *Iceweasel Copyright*. DEBIAN, 05. June. 2003. Available at: <http://packages.debian.org/changelogs/pool/main/i/iceweasel/current/copyright>. Accessed on: 07 June 2009. 66

[54] DORLAND, Eric. *Iceweasel*. DEBIAN, 2009. Available at: <http://packages.qa.debian.org/i/iceweasel.html>. Accessed on: 07 June 2009. 66

[55] MOZILLA. *Firefox*. Available at: <http://www.mozilla.com>. Accessed on: 07 June 2009. 66

[56] FOWLER, M. *UML distilled*: a brief guide to the standard object modeling language. [S.l.]: Addison-Wesley Professional, 2003. ISBN 0321193687. 72

[57]
SUN MICROSYSTEMS. *Core J2EE patterns*: Data Access Object. 2002. Available at: <http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html>. Accessed on: 01 July 2009. 72, 86

[58] DRUPAL. *Drupal API*: hooks. 2009. Available at: <http://api.drupal.org/api/group/hooks/6>. Accessed on: 25 June 2009. 74

[59] ______. *Drupal API*: default theme implementations. 2009. Available at: <http://api.drupal.org/api/group/themeable/6>. Accessed on: 01 July 2009. 76

[60] ______. *Drupal API*. 2009. Available at: <http://api.drupal.org/>. Accessed on: 01 July 2009. 76

[61] ______. *Drupal API*: check_plain. 2009. Available at: <http://api.drupal.org/api/function/check_plain/6>. Accessed on: 01 July 2009. 76, 77

[62] ______. *Drupal API*: l. 2009. Available at: <http://api.drupal.org/api/function/l/6>. Accessed on: 25 June 2009. 76

[63] POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL*. 2009. Available at: <http://www.postgresql.org>. Accessed on: 01 July 2009. 76

[64] DRUPAL. *Database abstraction layer*. Available at: <http://api.drupal.org/api/group/database/6>. Accessed on: 01 July 2009. 76, 77

[65] BUTCHER, M. *Learning Drupal 6 module development*: a practical tutorial for creating your first Drupal 6 modules with PHP. [S.l.]: Packt Publ., 2008. ISBN 978-1-847194-44-2. 78, 94

[66] DRUPAL. *Node access rights*. Available at: <http://api.drupal.org/api/group/node_access/6>. Accessed on: 15 June 2009. 78, 96

[67] JACOBSON, I. *Object oriented software engineering*: a use case driven approach. [S.l.]: Addison-Wesley Professional, 1992. ISBN 0201544350. 81

[68] GAMMA, E. et al. *Design patterns*: elements of reusable object-oriented software. [S.l.]: Addison-Wesley Professional, 1994. ISBN 0201633612. 87

[69] FOWLER, M. *Inversion of control containers and the dependency injection pattern*. 2004. Available at: <http://martinfowler.com/articles/injection.html>. Accessed on: 14 June 2009. 87

[70] DRUPAL. *Drupal API*: hook_node_info. 2009. Available at: <http://api.drupal.org/api/function/hook_node_info/6>. Accessed on: 28 June 2009. 95

[71] _____. *Drupal API*: hook_perm. 2009. Available at: <http://api.drupal.org/api/function/hook_perm/6>. Accessed on: 28 June 2009. 95

[72] _____. *Drupal API*: hook_access. 2009. Available at: <http://api.drupal.org/api/function/hook_access/6>. Accessed on: 28 June 2009. 96

[73] _____. *Tracker*: viewing new and updated content. 2009. Available at: <http://drupal.org/handbook/modules/tracker>. Accessed on: 28 June 2009. 96

[74] _____. *Drupal API*: hook_node_grants. 2009. Available at: <http://api.drupal.org/api/function/hook_node_grants/6>. Accessed on: 28 June 2009. 96

[75] _____. *Drupal API*: hook_node_access_records. 2009. Available at: <http://api.drupal.org/api/function/hook_node_access_records/6>. Accessed on: 28 June 2009. 96

[76] _____. *Drupal API*: hook_form. 2009. Available at: <http://api.drupal.org/api/function/hook_form/6>. Accessed on: 28 June 2009. 96

[77] _____. *Drupal API*: hook_validate. 2009. Available at: <http://api.drupal.org/api/function/hook_validate/6>. Accessed on: 28 June 2009. 97

[78] _____. *Drupal API*: hook_insert. 2009. Available at: <http://api.drupal.org/api/function/hook_insert/6>. Accessed on: 28 June 2009. 97

[79] _____. *Drupal API*: hook_update. 2009. Available at: <http://api.drupal.org/api/function/hook_update/6>. Accessed on: 28 June 2009. 97

[80] _____. *Drupal API*: hook_delete. 2009. Available at: <http://api.drupal.org/api/function/hook_delete/6>. Accessed on: 28 June 2009. 97

[81] _____. *Drupal API*: hook_load. 2009. Available at: <http://api.drupal.org/api/function/hook_load/6>. Accessed on: 28 June 2009. 97

[82] _____. *Drupal API*: hook_view. 2009. Available at: <http://api.drupal.org/api/function/hook_view/6>. Accessed on: 28 June 2009. 97

[83] ______. *Drupal API*: hook_theme. 2009. Available at: <http://api.drupal.org/api/function/hook_theme/6>. Accessed on: 28 June 2009. 98

[84] ______. *Comment*: allow comments on content. 2009. Available at: <http://drupal.org/handbook/modules/comment>. Accessed on: 28 June 2009. 98

[85] HOEBEN, A. *Re: comment subjects*. 2009. Available at: <http://drupal.org/project/comment_subject>. Accessed on: 28 June 2009. 98

[86] ZIEGLER, W. *Content access*. May. 2007. Available at: <http://drupal.org/project/content_access>. Accessed on: 28 June 2009. 98

[87] SALVISBERG, H. *ACL*. Oct. 2006. Available at: <http://drupal.org/project/acl>. Accessed on: 28 June 2009. 98

[88] DRUPAL. *Forum*: create threaded discussions. 2009. Available at: <http://drupal.org/handbook/modules/forum>. Accessed on: 28 June 2009. 98

[89] ______. *Blog*: a blog for every user. 2009. Available at: <http://drupal.org/handbook/modules/blog>. Accessed on: 28 June 2009. 98

[90] ______. *Poll*: community voting. 2009. Available at: <http://drupal.org/handbook/modules/poll>. Accessed on: 28 June 2009. 98

[91] OPEN SOURCE SOFTWARE IMAGE MAP. OSSIM. Melbourne, 2009. Available at: <http://www.ossim.net>. Accessed on: 01 July 2009. 124

Annex A - Sample Risk Assessment Report Outline (Based on [3])

EXECUTIVE SUMMARY

I. Introduction

- Purpose
- Scope of the risk assessment

Describe the system components, elements, users, field site locations (if any), and any other details about the system to be considered in the assessment.

II. Risk Assessment Approach

Briefly describe the approach used to conduct the risk assessment, such as:

- The participants (e.g., risk assessment team members);
- The technique used to gather information (e.g., the use of tools, questionnaires); and

- The development and description of risk scale (e.g., a 3 x 3, 4 x 4 , or 5 x 5 risk-level matrix).

III. System Characterization

Characterize the system, including hardware (e.g., server, router, switch), software (e.g., application, operating system, protocol), system interfaces (e.g., communication link), data, and users. Provide connectivity diagram or system input and output flowchart to delineate the scope of the risk assessment effort.

IV. Threat Statement

Compile and list the potential threat-sources and associated threats applicable to the system assessed.

V. Risk Assessment Results

List the observations (vulnerability/threat pairs). Each observation must include:

- Observation number and brief description of observation (e.g., Observation 1: User system passwords can be guessed or cracked);
- A discussion of the threat-source and vulnerability pair;
- Identification of existing mitigating security controls;
- Likelihood discussion and evaluation (e.g., High, Medium, or Low likelihood);

- Impact analysis discussion and evaluation (e.g., High, Medium, or Low impact);
- Risk rating based on the risk-level matrix (e.g., High, Medium, or Low risk level); and
- Recommended controls or alternative options for reducing the risk.

VI. Summary

Total the number of observations. Summarize the observations, the associated risk levels, the recommendations, and any comments in a table format to facilitate the implementation of recommended controls during the risk mitigation process.

Annex B - Sample Safeguard Implementation Plan Summary Table (Based on [3])

(1) Risk (Vulnerability/ Threat Pair)	**(2) Risk Level**	**(3) Recommended Controls**	**(4) Action Priority**	**(5) Selected Planned Controls**	**(6) Required Resources**	**(7) Responsible Team/Persons**	**(8) Start Date/ End Date**	**(9) Maintenance Requirement/ Comments**
Unauthorized users can telnet to XYZ server and browse sensitive company files with the guest ID.	High	• Disallow inbound telnet • Disallow “world” access to sensitive company files • Disable the guest ID or assign difficult-to-guess password to the guest ID	High	• Disallow inbound telnet • Disallow “world” access to sensitive company files • Disable the guest ID	10 hours to reconfigure and test the system	John Doe, XYZ server system administrator; Jim Smith, company firewall administrator	9-1-2009 to 9-2-2009	• Perform periodic system security review and testing to ensure adequate security is provided for the XYZ server

(1) The risks (vulnerability/threat pairs) are output from the risk assessment process

(2) The associated risk level of each identified risk (vulnerability/threat pair) is the output from the risk assessment process

(3) Recommended controls are output from the risk assessment process

(4) Action priority is determined based on the risk levels and available resources (e.g., funds, people, technology)

(5) Planned controls selected from the recommended controls for implementation

(6) Resources required for implementing the selected planned controls

(7) List of team(s) and persons who will be responsible for implementing the new or enhanced controls

(8) Start date and projected end date for implementing the new or enhanced controls

(9) Maintenance requirement for the new or enhanced controls after implementation

Annex C - NIST Security Controls

To achieve an ease of use in a control selection and specification process, control types are classified in classes and families. There are three classes: management, operational and technical. Each of the seventeen families pertains to some class and has a two character identifier associated for identification purposes. The security control types are associated to some family and have a number identifier. [21]

According to [21], the management controls class has the following families and security control types:

- CA - Certification, Accreditation, and Security Assessments
  - CA-1: Certification, Accreditation, and Security Assessment Policies and Procedures
  - CA-2: Security Assessments
  - CA-3: Information System Connections
  - CA-4: Security Certification
  - CA-5: Plan of Action and Milestones
  - CA-6: Security Accreditation
  - CA-7: Continuous Monitoring

- PL - Planning
  - PL-1: Security Planning Policy and Procedures
  - PL-2: System Security Plan
  - PL-3: System Security Plan Update
  - PL-4: Rules of Behavior
  - PL-5: Privacy Impact Assessment
  - PL-6: Security-Related Activity Planning
- RA - Risk Assessment
  - RA-1: Risk Assessment Policy and Procedures
  - RA-2: Security Categorization
  - RA-3: Risk Assessment
  - RA-4: Risk Assessment Update
  - RA-5: Vulnerability Scanning
- SA - System and Services Acquisition
  - SA-1: System and Services Acquisition Policy and Procedures
  - SA-2: Allocation of Resources
  - SA-3: Life Cycle Support
  - SA-4: Acquisitions
  - SA-5: Information System Documentation
  - SA-6: Software Usage Restrictions
  - SA-7: User Installed Software

  - SA-8: Security Engineering Principles
  - SA-9: External Information System Services
  - SA-10: Developer Configuration Management
  - SA-11: Developer Security Testing

The families and security control types related to operational controls class, according to [21], are the following:

- AT - Awareness and Training
  - AT-1: Security Awareness and Training Policy and Procedures
  - AT-2: Security Awareness
  - AT-3: Security Training
  - AT-4: Security Training Records
  - AT-5: Contacts with Security Groups and Associations
- CM - Configuration Management
  - CM-1: Configuration Management Policy and Procedures
  - CM-2: Baseline Configuration
  - CM-3: Configuration Change Control
  - CM-4: Monitoring Configuration Changes
  - CM-5: Access Restrictions for Change
  - CM-6: Configuration Settings
  - CM-7: Least Functionality

  - CM-8: Information System Component Inventory

- CP - Contingency Planning
  - CP-1: Contingency Planning Policy and Procedures
  - CP-2: Contingency Plan
  - CP-3: Contingency Training
  - CP-4: Contingency Plan Testing and Exercises
  - CP-5: Contingency Plan Update
  - CP-6: Alternate Storage Site
  - CP-7: Alternate Processing Site
  - CP-8: Telecommunications Services
  - CP-9: Information System Backup
  - CP-10: Information System Recovery and Reconstitution
- IR - Incident Response
  - IR-1: Incident Response Policy and Procedures
  - IR-2: Incident Response Training
  - IR-3: Incident Response Testing and Exercises
  - IR-4: Incident Handling
  - IR-5: Incident Monitoring
  - IR-6: Incident Reporting
  - IR-7: Incident Response Assistance
- MA - Maintenance

  - MA-1: System Maintenance Policy and Procedures
  - MA-2: Controlled Maintenance
  - MA-3: Maintenance Tools
  - MA-4: Remote Maintenance
  - MA-5: Maintenance Personnel
  - MA-6: Timely Maintenance

- MP - Media Protection
  - MP-1: Media Protection Policy and Procedures
  - MP-2: Media Access
  - MP-3: Media Labeling
  - MP-4: Media Storage
  - MP-5: Media Transport
  - MP-6: Media Sanitization and Disposal

- PE - Physical and Environmental Protection
  - PE-1: Physical and Environmental Protection Policy and Procedures
  - PE-2: Physical Access Authorizations
  - PE-3: Physical Access Control
  - PE-4: Access Control for Transmission Medium
  - PE-5: Access Control for Display Medium
  - PE-6: Monitoring Physical Access
  - PE-7: Visitor Control

  - PE-8: Access Records
  - PE-9: Power Equipment and Power Cabling
  - PE-10: Emergency Shutoff
  - PE-11: Emergency Power
  - PE-12: Emergency Lighting
  - PE-13: Fire Protection
  - PE-14: Temperature and Humidity Controls
  - PE-15: Water Damage Protection
  - PE-16: Delivery and Removal
  - PE-17: Alternate Work Site
  - PE-18: Location of Information System Components
  - PE-19: Information Leakage

- PS - Personnel Security
  - PS-1: Personnel Security Policy and Procedures
  - PS-2: Position Categorization
  - PS-3: Personnel Screening
  - PS-4: Personnel Termination
  - PS-5: Personnel Transfer
  - PS-6: Access Agreements
  - PS-7: Third-Party Personnel Security
  - PS-8: Personnel Sanctions

- SI - System and Information Integrity
  - SI-1: System and Information Integrity Policy and Procedures
  - SI-2: Flaw Remediation
  - SI-3: Malicious Code Protection
  - SI-4: Information System Monitoring Tools and Techniques
  - SI-5: Security Alerts and Advisories
  - SI-6: Security Functionality Verification
  - SI-7: Software and Information Integrity
  - SI-8: Spam Protection
  - SI-9: Information Input Restrictions
  - SI-10: Information Accuracy, Completeness, Validity, and Authenticity
  - SI-11: Error Handling
  - SI-12: Information Output Handling and Retention

The class of technical controls has the following families and control types according to [21]:

- AC - Access Control
  - AC-1: Access Control Policy and Procedures
  - AC-2: Account Management
  - AC-3: Access Enforcement
  - AC-4: Information Flow Enforcement

  - AC-5: Separation of Duties
  - AC-6: Least Privilege
  - AC-7: Unsuccessful Login Attempts
  - AC-8: System Use Notification
  - AC-9: Previous Logon Notification
  - AC-10: Concurrent Session Control
  - AC-11: Session Lock
  - AC-12: Session Termination
  - AC-13: Supervision and Review—Access Control
  - AC-14: Permitted Actions without Identification or Authentication
  - AC-15: Automated Marking
  - AC-16: Automated Labeling
  - AC-17: Remote Access
  - AC-18: Wireless Access Restrictions
  - AC-19: Access Control for Portable and Mobile Devices
  - AC-20: Use of External Information Systems

- AU - Audit and Accountability
  - AU-1: Audit and Accountability Policy and Procedures
  - AU-2: Auditable Events
  - AU-3: Content of Audit Records
  - AU-4: Audit Storage Capacity

  - AU-5: Response to Audit Processing Failures
  - AU-6: Audit Monitoring, Analysis, and Reporting
  - AU-7: Audit Reduction and Report Generation
  - AU-8: Time Stamps
  - AU-9: Protection of Audit Information
  - AU-10: Non-repudiation
  - AU-11: Audit Record Retention

- IA - Identification and Authentication
  - IA-1: Identification and Authentication Policy and Procedures
  - IA-2: User Identification and Authentication
  - IA-3: Device Identification and Authentication
  - IA-4: Identifier Management
  - IA-5: Authenticator Management
  - IA-6: Authenticator Feedback
  - IA-7: Cryptographic Module Authentication

- SC - System and Communications Protection
  - SC-1: System and Communications Protection Policy and Procedures
  - SC-2: Application Partitioning
  - SC-3: Security Function Isolation
  - SC-4: Information Remnance
  - SC-5: Denial of Service Protection

- SC-6: Resource Priority
- SC-7: Boundary Protection
- SC-8: Transmission Integrity
- SC-9: Transmission Confidentiality
- SC-10: Network Disconnect
- SC-11: Trusted Path
- SC-12: Cryptographic Key Establishment and Management
- SC-13: Use of Cryptography
- SC-14: Public Access Protections
- SC-15: Collaborative Computing
- SC-16: Transmission of Security Parameters
- SC-17: Public Key Infrastructure Certificates
- SC-18: Mobile Code
- SC-19: Voice Over Internet Protocol
- SC-20: Secure Name/Address Resolution Service (Authoritative Source)
- SC-21: Secure Name/Address Resolution Service (Recursive or Caching Resolver)
- SC-22: Architecture and Provisioning for Name/Address Resolution Service
- SC-23: Session Authenticity

More details about the NIST security controls (security control types) and their classification can be obtained in [21].

FOLHA DE REGISTRO DO DOCUMENTO

1. CLASSIFICAÇÃO/TIPO DM	2. DATA 04 de fevereiro de 2010	3. REGISTRO Nº CTA/ITA/DM-129/2009	4. Nº DE PÁGINAS 147

5. TÍTULO E SUBTÍTULO:
A CMS-Based Tool for Continuous and Collaborative Risk Management Process

6. AUTOR(ES):
Gabriel Negreira Barbosa

7. INSTITUIÇÃO(ÕES)/ÓRGÃO(S) INTERNO(S)/DIVISÃO(ÕES):
Instituto Tecnológico de Aeronáutica - ITA

8. PALAVRAS-CHAVE SUGERIDAS PELO AUTOR:
IT security; IT security risk management process; Collaborative software

9. PALAVRAS-CHAVE RESULTANTES DE INDEXAÇÃO:
Tecnologia da informação; Segurança de dados; Gerenciamento de riscos; Trabalho em grupo (computadores); Gestão de conteúdo; Ferramentas de desenvolvimento de software; Engenharia de software

10. APRESENTAÇÃO: **X Nacional** **Internacional**
ITA, São José dos Campos. Curso de Mestrado. Programa de Pós-Graduação em Engenharia Eletrônica e Computação. Área de Informática. Orientador: Prof. Dr. Edgar Toshiro Yano. Defesa em 13/08/2009. Publicada em 2009.

11. RESUMO:
A Security Risk Management Process is an important part for any approach to software security. To be effective, this process should address some issues, like the analysis of different risks and related safeguards, continuous collaboration among involved professionals, and a robust access control mechanism due to the sensitivity of the involved information. In this research, it was developed a solution for security risk management based on an open-source CMS (Content Management System). This tool attends requirements for a secure, continuous and collaborative risk management, due to the importance and the lack of mature solutions in those aspects.

12. GRAU DE SIGILO:
(X) OSTENSIVO **( ) RESERVADO** **( ) CONFIDENCIAL** **( ) SECRETO**