FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Oracle Warehouse Management System – Security Enhancements

Marc-Olivier Esteves Gonçalves

FINAL VERSION

Report of Project
Master in Informatics and Computing Engineering

Supervisor: Gabriel de Sousa Torcato David (Ph.D.)

2009, July

Oracle Warehouse Management System – Security Enhancements

Marc-Olivier Esteves Gonçalves

Report of Project
Master in Informatics and Computing Engineering

Approved in oral examination by the Committee:

Chair: José Manuel Magalhães Cruz (Ph.D.)

External Examiner: José Manuel Matos Moreira (Ph.D.)

Internal Examiner: Gabriel de Sousa Torcato David (Ph.D.)

29th July 2009

(This page intentionally left blank)

Abstract

Nowadays, information constitutes the greatest valuable and important business asset, for most, if not all, companies. This is clearly true for retailers and supply chain professionals which use information and technology to improve overall business value.

Therefore, information must be secure in order to retain its value and prevent the problems that could arise from its unauthorized exploitation. To achieve this purpose, the databases that hold such data as well as the applications by which the data are accessed must be secured. Yet, security is not always considered when developing systems.

Oracle Retail Warehouse Management System – ORWMS – is a Oracle multi-tier application which provides several functionalities necessaries to the efficient management and control of warehouses' merchandise and information. However, the security mechanisms provided by ORWMS are very rudimentary and still well behind those offered by the others applications that integrate the Oracle Retail suite. In fact, the security mechanisms integrated into the application – password expiration and access control based on privileges levels – don't ensure essential security tenets, considered as the basis for a good security solution, like the 'defense in depth' and 'least privileges' principles.

The main goal of this project was, therefore, to design a security model which could enhance the overall application security. Due to being a broad field, and considering the fact that the majority of security breaches are perpetrated by insiders, we focused on database and application's security with the objective of providing an internal security model.

The process consisted on the study and analysis of the application to identify existing vulnerabilities, at both database and application levels, and the steps required to fix them.

Then, a set of solutions were identified to enforce the security not only within the database but also at application level. However, since too much security can be self-defeating and because determining how to protected is based on what we are trying to protect, the nature and sensitivity of the stored data, as well as the type of environment usage were taken into consideration for the design of the final model. Performance and usability were also considered.

As final step, all selected solutions had been integrated to form a new security model for the ORWMS. This model is able to fix the vulnerabilities found, thus enhancing the ORWMS security as a whole.

Acknowledgements

First and foremost I would like to thank my family, especially my parents, my brother and my little sister for all their unflagging love and support throughout my life. All the words are not enough to express my gratitude to them.

I wish to express my warm and sincere thanks to my supervisor, Professor Gabriel David. His insights, continuous guidance and feedback were invaluable.

I am also deeply grateful to Dr. Ana Paula Barroso Oliveira, my supervisor at Wipro Retail, for her kindness and all the encouragements and guidance she gave me.

All my internship colleagues, and now my friends, deserve a special thanks for their daily enthusiasm and good mood, and all the funny discussions and moments we shared together.

I would also like to express my gratitude to Eng. Hugo Neto for having placed his confidence in me, by providing me the opportunity of doing this internship.

I would like to acknowledge and extend my heartfelt gratitude to all the Wipro Retail collaborators whose great sympathy and welcome have made this internship an agreeable experience. In particular, a special thanks to Carla Almeida for having followed my integration in the company, and for her constant worry with my well-being.

Moreover, I would also like to thank all the Wipro ORWMS' experts who answered my questionnaire.

I am also highly thankful to my closest friends for the weekly meetings, for all the laughs and all the good times that have made the last five years an enjoyable experience.

Last but not least, I owe my gratitude to all my Professors at FEUP, with whom I had the privilege to learn, for their availability and the immense knowledge they shared.

To all, my most sincere thanks.

Marc-Olivier Esteves Gonçalves.

Contents

List of Figures

List of Tables

Abbreviations and Symbols

3GL	*Third-generation programming language*
4GL	*Fourth-generation programming language*
ASN	*Advanced Ship Notice*
DB	*Database*
DIY	*Do-It-Yourself*
DML	*Data Manipulation Language*
GUI	*Graphical User Interface*
IT	*Information Technologies*
ORMS	*Oracle Retail Merchandising System*
ORWMS	*Oracle Retail Warehouse Management System*
RF	*Radio-Frequency*
TNS	*Transparent Network Substrate*
VA	*Vulnerability Assessment*
WWW	*World Wide Web*

1 Introduction

This chapter introduces and contextualizes the problem that the thesis addresses, to help all interested parts to understand and follow the project and its development. The motivation and project's objectives are also briefly discussed in this section and, at the end, the report's structure is described, giving an overview of each chapter.

1.1 Contextualization

This document aims to describe the work done by the author in its internship which took place at Wipro Retail.

Wipro Retail is an international IT and business services company that delivers measurable value to the world’s leading and best-known retailers, through the supply of Retail consulting services or the deployment of innovative IT-based solutions.

Wipro Retail (in the past with the name Enabler) was created in 1997 through the planned separation of the IS/IT department of Portugal's leading retailer, Modelo Continente Hipermercados, a division of Sonae Group [Wipro09]. Since its creation, Enabler has managed significant and sustainable growth focusing on leading retailers in the major European markets.

In the following years, Enabler reached high levels of performance regarding sales and profitability growth rates, expanding its clients' portfolio beyond Europe. In 2006, Enabler was acquired by Wipro technologies, the global IT Services arm of Wipro Limited. This acquisition enabled Wipro to achieve a high know-how in retailing and thereby reinforces and leverage its solutions portfolio. Thus, Wipro Retail was born.

Figure 1.1: Wipro Retail Logo
[Wipro 09]

Presently, Wipro Retail works for a diverse set of Retail formats (grocery, fashion, DIY[1], department stores) (Figure 1.1), offering to its customers services at system integration, development, implementation and support levels, based on the suite of applications Oracle Retail.

Figure 1.2: Wipro Retail client portfolio

The retail industry is, nowadays, among the most challenging and competitive sectors. This complexity should cause retailers to rethink how they manage the impact of these challenges on their supply chain. Over the next decade, information-driven collaborative supply chains will form the core of retailers' business models. Today, most supply chains are built on inventory. Inventory reserves are quite common to ensure that risks are balanced with "just-in-case" demand requirements. Companies have built-in safety stocks to adjust for inconsistent supply sources, which instils a level of constant "buy-more" trade off to make up for those suppliers who can't ship on time. To meet these challenges, the retail industry is beginning to look at the supply chain as a strategic opportunity within a broader company strategy. Success will be measured by how well supply chain professionals use information and technology to improve overall business value.[OraMSR]

Oracle Corporation, it's the world's largest business software company, with more than 320.000 customers, in more than 145 countries around the world [Oracle09]. Although being best-known due to its Database Management Systems (DBMS), the corporation also builds tools for middle-tier software, Enterprise Resource Planning (ERP), Customer Relationship

1 DIY or Do-It-Yourself - It is used as indicative sign for shops specialized in the supply of construction, repair, decorative or assembly goods mostly used by skilled artisans but made available in convenient quantities for people wishing to do it themselves;

Management (CRM) and Supply Chain Management (SCM)., thus strengthening its position in the retail applications market globally.

Oracle Retail is the result's name of Oracle's strategic acquisition of best-of-breed applications, as well as the realization of its long-term vision for the retail sector.

In the last few years, Wipro Retail's relationship with Oracle Retail has strengthened dramatically and the two companies actively work together to deliver strong business solutions for the world's largest retail organizations. As consequence, Wipro Retail is today a preferred integrator of Oracle Retail solutions and provider of Retail consulting services for global retailers.

1.2 Brief Description of the Project

Supply chain management constitutes the key to having the right product in the right place at the right time. The objective for retailers - maximizing service levels and in-stock positions while minimizing inventory and operational costs - is complicated by the lack of timely information and the flexible execution systems necessary for adapting to rapidly changing business conditions.

Oracle Retail has been building a suite of products to cover major retailers business processes. Oracle Retail Warehouse Management System[2] (ORWMS) is a Oracle Retail application which provides all the necessary tools for efficiently managing and controlling a complex distribution center. Leveraging a process-based application framework, Oracle Retail Warehouse Management provides the functional flexibility and timely, accurate information needed for consistently managing and improving distribution operations. The results are improved service levels, reduced inventory and lead-times, increased productivity, and reduced labour costs. [OraRWM]

However, the security mechanisms of ORWMS are very rudimentary and still well behind those offered by the other applications that integrate the Oracle Retail suite. This may be explained due to the operational environment, the restricted group of people which usually use the application, as well as the nature of the stored information, compared with that involved in other products of the suite.

Nevertheless, with data becoming one of the most valuable assets for today's companies, and the problems that may arise from the misuse of such data, security started to become a legitimate concern. As a consequence, this project arises internally in response to some concerns related to the ORWMS security, that started to be issued by several customers where the product was implemented. Despite several security vulnerabilities, these concerns are mainly related with the implementation of a control access on the ORWMS functionalities.

The main project's goal is, therefore, to design a security model for ORWMS which could enhance the application regarding to this important non-functional requirement that is security.

1.3 Motivation and Objectives

Information is a company's most important business asset. Information drives today's businesses, creating value for organizations and giving them a vital resource needed to enter new markets and offer additional products and services.

Information must be both secure and available in order to retain its value. If it is secure but unavailable, it cannot be used. If it is available but not secure, it is not trustworthy and therefore should not be used. In this way, security within most applications is a critical and increasingly complex subject. Database applications have more stringent security requirements because of the centralized storage of information. Modern multi-tier application architectures present

2 The abbreviation ORWMS will be used to refer to the Oracle Retail Warehouse Management System

additional security challenges because each level and each network connection provides additional opportunities for data to be inappropriately viewed, modified, or corrupted. But the major challenge faced for designing, building, and deploying secure applications running against an Oracle Database is that there are few, if any, best practice documents, technical blueprints (architectures), or other reference guidelines showing how to link together varying technologies to build secure database applications.

This project was born with the aim to meet the security requirements that were recently issued by several customers for whom Wipro Retail has implemented the ORWMS application.

Therefore, this project has two main goals. The first is to explore ORWMS in what concerns to security, identifying existing vulnerabilities within the application and the steps required to mitigate the identified risks. The second is to propose a security model that will be able to fix the vulnerabilities found, thus enhancing the ORWMS security as a whole without affecting its correct functioning.

Due to the limited amount of time available (less than 20 weeks) and the broad scope of the area where the project is inserted, a prototype should be developed, as a secondary objective, with the purpose of serving as a 'proof-of-concept' for solving the ORWMS security concerns. Moreover, if proven as a valid approach, by satisfying the proposed security requirements, this solution may also serve as a starting point for the development of a new commercial product that could be offered by Wipro Retail as an additional package, for those customers who wish to implement ORWMS and improve its basic security mechanisms.

1.4 Project Schedule and Time Line

This project has two end dates. One related to the Master thesis, and the other related to the end of author's internship at Wipro Retail. The first one has a shorter duration (from 2nd March to 29th June) and the project objectives, described in the previous section, have been established accordingly.

Prioritizing objectives in a project is always a critical step. Giving priority to the tasks that are really important to reach the final goal is a an essential and difficult task.

After a careful evaluation of the project, some decisions have been made and the prioritization was done according to the following factors:

1. Importance of vulnerabilities identification phase and design of security model - Before begin implement security countermeasures, it's essential to have a clear understanding of the security design. This is critical because the design decisions determine how effective the security implementation will be once it has been built;

2. Dependence on feedback from the ORWMS pool collaborators – It was established at the beginning, that a meeting with ORWMS experts from Wipro Retail would take place, with the objective to serve as checkpoint and determine the next steps to be taken;

The following figure shows the initial project's planning that resulted from these considerations.

Oracle Warehouse Management System - Security Enhancement

Project Schedule

	Activities	Project Start Date	Project End Date	Duration (days)
		02-03-09	29-06-09	
1	**Structuring the project's page**			
2	**Study of the tool ORWMS**	02-03-09	06-03-09	5
2,1	Installation of the tool ORWMS	02-03-09	04-03-09	3
2,2	Reading of Oracle documentation	04-03-09	06-03-09	3
3	**Induction**	09-03-09	20-03-09	12
3,1	Wipro metodologies and process, Retail, ORMS	09-03-09	13-03-09	5
3,2	PL/SQL, Pro*C	16-03-09	20-03-09	5
4	**Study of security of the ORWMS application**	23-03-09	04-05-09	43
4,1	Analysis of security vulnerabilities in ORWMS	23-03-09	17-04-09	26
4,2	Study and analysis of possible solutions	26-03-09	04-05-09	40
5	**Analysis and Design of the new security model**	02-04-09	29-05-09	58
6	**Development of the new security model**	04-05-09	26-06-09	54
7	**Preparation of the Master's Thesis**	08-06-09	26-06-09	19

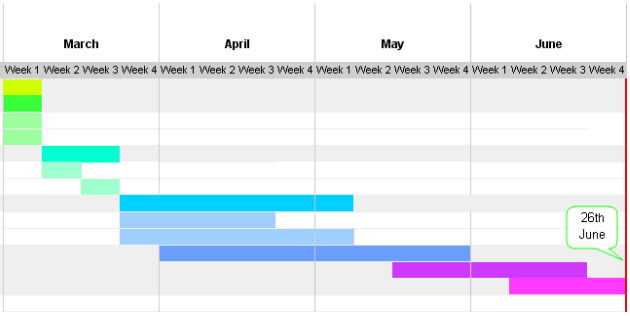

Figure 1.3: Project Schedule

The first phase considered ,“Study of the tool”, took approximately 1 week, and consisted on the installation of the tool and the reading of the official documentation.

This first week was followed by 2 weeks of induction about Wipro methodologies and process, retail, ORMS (Oracle Retail Merchandising System), as well as PL_SQL and Pro*C.

The Study and Analysis of ORWMS security took approximately 7 weeks. It included the identification of existing vulnerabilities in ORWMS, as well as the analysis of potential solutions to be applied. A few days after the beginning of this phase, the Analysis and Design of the new security model was initiated. This phase took nearly 8 weeks but it was interrupted approximately 2 weeks due to the allocation in a different project that needed some task force, and for which the contribution of the author was asked for. Unfortunately, this fact limited the amount of time available for the next phase, as no project’s output was prepared or developed during this period.

Then, as a consequence of tight time constraint, the Development phase didn't had the desirable amount of disposable time. Some difficulties in the configuration of the technology required for this phase, also had an impact on the time. Thus, only 2 weeks was dedicated to this phase.

The remaining available time was used exclusively for the elaboration of this thesis.

1.5 Report Structure

This report is organized into seven chapters. The first and the second are intended to let the reader understand the project by presenting the initial thematic, its objectives and the background that surrounds it.

The third chapter focuses itself in the review of the technologies associated with the project.

The fourth chapter analyses the problem and gives an overview of the solution. It also describes the methodology followed throughout each phase.

The fifth chapter describes the potential solutions to be applied, gives an overview for each of them, and explains the reasons why some of them were left aside. It also presents and describes the global design of the whole solution.

The sixth chapter describes some of the steps of the implementation phase, and the tests conduced in order to prove the adequacy of the solution proposed.

In the last chapter, the results that were obtained from the project are reviewed, some conclusions of the author about the project are drawn. The future work that has to be done is also presented.

2 State of the Art

Despite the rudimentary security mechanisms implemented by Oracle in its Warehouse Management solution, there is no public work related to security enhancements of this Oracle application. Such absence is not a proof that security is not an important concern, but probably a sign that customers security requirements was not been communicated to Oracle yet. As a consequence, IT companies, which implement this Oracle application, see this as an opportunity for them to provide a customized value-added service to their final customers and thereby, an opportunity to differentiate themselves from competitors.

Hence, this chapter presents the current state of the art on the field of ‘Security of Information Systems’, explaining the concepts needed for the project comprehension.

2.1 Security of Information Systems

The term 'information system' usually refers to a computer-based system, one that is designed to support the operations, management, and decision functions of an organization, eliminating tedious tasks and giving workers greater autonomy. In a narrow sense, a specific application software that is used to store data records in a computer and automates some of the information-processing activities of the organization. But, information systems are more than this. Indeed, they are composed by various components: computer hardware and software, databases, telecommunications systems, human resources, and procedures.

Computer software falls into two broad classes: system software and application software. The principal system software is known as the operating system. Application software are programs designed to handle specialized tasks;

A database is a collection of interrelated data (records) organized so that individual records or groups of records that satisfy various criteria can be retrieved.

Telecommunications are used to connect, or network, computer systems and transmit information. Various computer network configurations are possible, depending on the needs of an organization. Local area networks (LANs) join computers at a particular site, such as an office building or an academic campus. Wide area networks (WANs) connect machines located at different sites, and often within different organizations. The Internet is a network of networks, connecting millions of computers located on every continent. Through networking, personal computer users gain access to information resources, such as large databases, and to human resources, such as co-workers and people who share their professional or private interests.

Qualified people are a vital component of any information system. Technical personnel include development and operations managers, systems analysts and designers, computer

programmers, and computer operators. In addition, workers in an organization must be trained to utilize the capabilities of information systems.

Procedures for using, operating, and maintaining an information system are part of its documentation.

Therefore,an information system connects all the organization's components together and provides for better operation and survival in a competitive environment, enabling humans to perform tasks for which their brains are not well suited, such as: handling large amount of information, performing complex calculations, and controlling simultaneous processes. Hence, information systems play a central role in every organization, and it is therefore necessary to preserve their proper functioning. In addition, information systems are vulnerable to a number of threats, which require strict controls as countermeasures and regular audits to ensure that the system remains secure.

Information systems security is responsible for the integrity and safety of system resources and activities. Most organizations are dependent on the secure operation of their information systems. Information systems are at the heart of intensive-care units and air-traffic-control systems. Financial institutions could not survive a total failure of their information systems for longer than a day or two. Electronic funds transfer systems handle immense amounts of money that exist only as electronic signals over telecommunications lines or as magnetized spots on computer disks.[Britann09]

The best definition for 'information security' that we could find is from the Cornell University Law School [CULS07]. It reads:

“The terms 'information security' means protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, or destruction in order to provide integrity, confidentiality and availability.”

In practical terms, security[3] is concerned with the protection of assets. Assets are entities that someone places value upon, such as the content of a file or a server, or the access to a classified facility. Many assets are in the form of information that is stored (data), processed and transmitted by IT products, such as authentication credentials; privileged communication sessions; intellectual property; privacy act information; sensitive but unclassified information which is not approved for public access; or various forms of classified data. Thus, information owners may require that availability, dissemination and modification of any such information is strictly controlled and that the assets are protected from threats by countermeasures. These relationships are illustrated in Figure 3.1.

3 The word 'security' is used throughout this document to refer to information security or the security of information systems

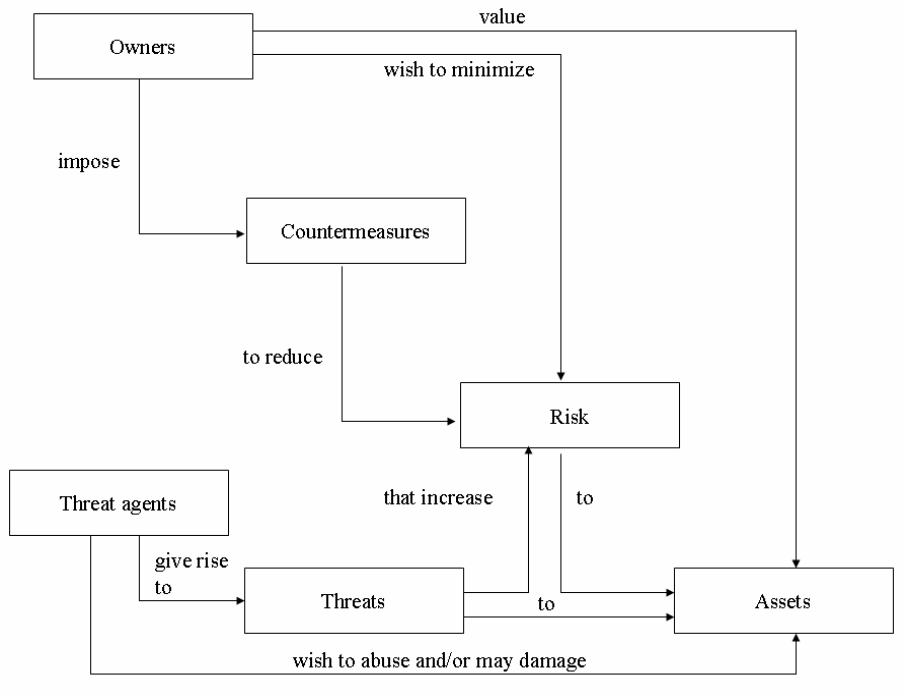

Figure 2.1: Security relationships [CC06]

Threat agents include hackers, malicious users, non-malicious users (who sometimes make errors), computer processes and accidents, and they give rise to threats that are perceived as potential for impairment of the assets such that the value of the assets to the owners would be reduced. Security-specific impairment commonly includes loss of asset confidentiality, asset integrity and asset availability. These threats therefore give rise to risks to the assets, based on the likelihood of a threat being realised and the impact on the assets when that threat is realised. Subsequently countermeasures are imposed to reduce the risks to assets. However, countering a threat does not necessarily mean removing that threat, it can also mean sufficiently diminishing that threat or sufficiently mitigating that threat. [CC06]

2.1.1 Information Systems Security Foundation Data

The first step in creating a secure information system is to identify threats. Once potential problems are known, the second step, establishing controls, can be taken. Finally, the third step consists of audits to discover any breach of security.

For the first step, it's essential to remember that the security of Information System has many aspects, and all must be considered:

- Secrecy and confidentiality – Data should not be disclosed to anyone not authorized to access it. It's about keeping valuable information only in the hands of those people who are intended to see it;
- Accuracy, integrity, and authenticity – Accuracy and integrity mean that data can't be maliciously or accidentally corrupted or modified. Indeed, information only has value if we know that it's correct. Authenticity is a variant on this concept; it provides a way to verify the origin of the data;

- Availability and recoverability – Systems and data can be recovered efficiently and completely (with no loss of accuracy or integrity) in case of loss, and are available and operational when they are needed;

Depending on the specific environment that is likely to be secured and user base, some of these aspects of security may be more important than others. However, security must be tight enough to protect data from both internal and external threats and security must be granular enough to protect privacy by limiting access to specific pieces of data for specific groups. To reach this goal, organizations must implement best practices that manage internal as well as external risk and secure the infrastructure where the data resides.

Good security practices help to create good security implementations. In fact, many of the implementation best practices can be traced back to a few proven security principles. Security principles are collections of desirable properties, behaviours, designs and implementation practices that attempt to reduce the likelihood of threat realization and impact should that threat be realized. By considering each of these principles, we can derive security requirements, make architecture and implementation decisions, and identify possible weaknesses in systems.

There are three important tenets that should be apply to all applications and databases, and should be followed to ensure effective security.[Knox04]

a) Security by Design

Security doesn't begins when the user authenticates. It begins when the application is designed. Therefore, the first tenet is this: *Security has to be built into the system, not bolted on afterwards.* A proper security design helps to ensure that an application will be secure.[Knox04]

In the majority of projects, security is considered as a non-functional requirements and frequently it is given less importance that it should. This is due to some partial myths that have cropped up.

- Security causes huge performance problems;
- Security increases system management complexity;
- Security features can complicate the implementation of other common enterprise architecture features, such as high availability or disaster recovery.

This may be true if security is not incorporated early into the design cycle to increase awareness of the constraints imposed by security. Thus, security should be considered as an architectural goal instead of a system property. [Ramac02]

We can find in [Knox04] an interesting analogy with an every-day example. Think about automobile construction. Imagine that automobiles don’t come with door locks because this security measure interferes with usability. We buy an automobile and immediately recognize this security shortcoming. Perhaps our solution would be a big chain and a padlock around the doors or perhaps we think of a way to attach a rope around the door handle. Whatever security solution we think of, it’s probably not elegant. The same applies to information systems when security is not part of the design and in many cases, performance and practical implementation issues may prevent the security “feature” from actually being used at all.

An other reason for adopting this principle is that project's budget constraints limit remediation. At the end of the development lifecycle, most of a project’s budget is exhausted. Often, development teams are unable to secure additional funding for application redesign and must move ahead without remediation. That results in greater risk to the applications and, in turn, to the business information it manages.[Kapur05] On the other hand, as applications move through the development and delivery process, errors can quickly become more complicated and expensive to remedy, than they are in the earlier in lifecycle, where discover and fix vulnerabilities are most cost-effective.

As the following figure shows, fixing a design error after an application is available costs approximately 30 times more than addressing it during design. And these estimates do not even factor in costs such as losses in market share, reputation or customer satisfaction.

Type of error	Design	Coding	Integration	Beta	Deployment
Design	1x	5x	10x	15x	30x
Coding		1x	10x	20x	30x
Integration			1x	10x	20x

Figure 2.2: Relative cost of vulnerability fixes, based on time of discovery [IBM09]

The discipline of Software Architecture has only recently started to integrate security as a design principle into its methodologies, giving it the weight normally accorded to the better-understood principles of performance, portability, scalability, reliability, maintainability, profiling, and testability. In the past, unlike these established principles of software development, security has been presented as an independent property of a system rather than as a fundamental system feature to be specified, designed, and developed. [Ramac02] However, nowadays, the software community recognizes that it is often more costly to resolve software security issues than it is to fix functional bugs, thereby the "secure-by-design" concept is becoming popular in the software development world.[SI07]

b) Defense in Depth

This tenet for effective security says that *security should be a multi-layered composition*. This is commonly known as "defense in depth," and it implies there will be no single point of failure from within the security domain. [Knox04] By creating multiple layers of security, we can generate a system with higher security assurance, placing multiple barriers between an attacker and the system and, thereby, increasing the cost of an attack. If one layer of defense is defeated or fails, another will hold and provide the necessary security to protect the system. This principle is very important because the battle between those trying to protect and those trying to break in is overbalanced. Those trying to protect must get it 100% right. Those trying to get in only need to get it right once. Without the application of this principle, a single hole found in a security system allows an attacker to breach that security system and get to the protected assets.

If we look again to the automobile analogy, we see this principle is applied. In fact, to prevent automobile theft, cars are secured by more than just door locks. For example, some cars come equipped with an alarm system, an ignition-kill switch (also known as immobilizer) to prevent the car from being started, or a pedal locking device. Each layer incrementally adds more security, thus increasing the security of the system as a whole.

Going back to the information systems case, this means we also have to add security to the layer of database security, layer of network, operating system and application. However, implementing a defense-in-depth strategy can runs counter to the "simplicity" principle often practiced in security. Actually, adding new protection functionality adds additional complexity that might bring new risks with it. Therefore, the total risk to the system needs to be weighed.

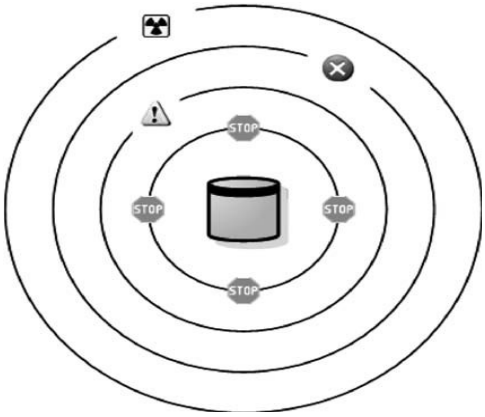

Figure 2.3: Layered protection of assets -
Defense in Depth approach [Natan05]

c) Least Privileges

The last tenet of effective security is to *maintain least privileges*. “Least privileges” means it should be given to people, process and programs only the permissions they need to do their job or task and nothing more: in other words, we give them the least amount of privileges. In reality, one of the easiest ways to compromise a system is to exploit accounts that have been granted too much privilege.[Knox04]

The process of abiding by least privileges is simple and not following this principle can result in serious consequences. Nevertheless, it is not followed regularly and one primary reason to explain this is 'laziness'. This creates a huge risk because the user can do anything if the least privileges principle is not practiced—read any table, update any data, execute any procedure in any schema, and so much more.

Database security is based on privileges. Privilege abuse cannot occur if the privileges haven’t been granted in the first place. While it requires some effort to determine what privileges are needed, it’s essential for maintaining least privileges and least privileges are essential for effective security.[Knox04]

The principles described above provide a foundation upon which a more consistent and structured approach to the design, development, and implementation of security capabilities can be constructed.

[SHF04] proposed a compiled set of additional engineering principles for system security. While the primary focus of these principles is the implementation of technical controls, they also highlight the fact that, to be effective, a system security design should also consider non-technical issues, such as policy, operational procedures, and user education and training.

In fact, good security implementations are based on good security policies. Security begins not with encryption algorithms, firewalls, or advanced authentication techniques; it begins with a clear and well-defined security policy. The policies define the rules by which everyone should abide. In other words, policies allow organizations to set practices and procedures in place that will reduce the likelihood of an attack or an incident and will minimize the damage caused that such an incident can cause, should one occur. There must be specific policies about authentication, access control, and auditing. Confidentiality and privacy concerns have to be addressed as well. However, there is no “one-size-fits-all” security policy. Determining *how* to protect is based on *what* you are trying to protect.[Knox04]

Policies provide the guidance necessary for effective implementations. Without a clearly defined policy, there is nothing to implement, nothing to enforce, and no way of knowing if the system is secure.[Knox04]

Actually, there is a direct correlation between the security policies, the security implementations, and the data. Security policies can vary in level of detail and level of enforcement. These variations are based on the data sensitivity and the application's intended use. From an IT perspective, they initially apply to all applications throughout an enterprise. As applications are developed and fielded, the security policies associated with the applications can become more specific to the applications. For sensitive data, such as proprietary data, the policies are strict and may cover everything from the physical protection to availability and recoverability to the personnel operating the system. For applications that deal with less sensitive data, such as an intraoffice equipment checkout application, the policies are generally less comprehensive and less strict.[Knox04] Moreover, it is important to maintain the correct level of detail and right level of enforcements because too much security can have a bad side effect on usability. To illustrate this point, consider a password policy defined by a company, which indicate that users passwords must be at least 12 characters in length, contain mixed numbers, upper- and lower-case characters, expire every month, and cannot be reused for seven cycles. This effort to increase security has the opposite effect. In fact, people are forced to write the passwords down, as they cannot remember and create passwords with that level of sophistication and so frequently in time. Therefore environment's security is reduced. Thus, effective security has to incorporate practicality and usability issues to ensure that the overall security of a system will be increased.

Security also competes with other requirements besides usability such as performance, cost, and administration. Typically, some trade-offs will be necessary (Figure 2.4), because as security increases, usability and performance decrease while costs and administration increase.

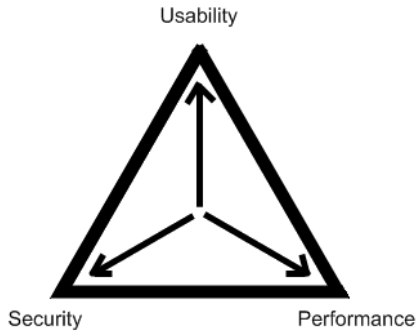

Figure 2.4: Requirements trade-offs

Scalability, performance, and administration ease-of-use often get most of the attention. Security, however, must also be part of the consideration. Obviously, the goal will be to achieve security *and* usability *and* performance, although the last two are arguably more important than security in the mind of the system architect because they directly affect the business goals of the system.

2.1.2 The Importance of Securing Information Systems

A company's greatest and most valuable asset is its data—and the people who control, manage and interact with that data. New data is constantly being created as organizations look to capitalize on these benefits and extract as much value as possible from their information. According to a survey by [Analyst08], approximately 70 percent of organizations indicate a year-on-year increase in the volume of data they generate and attempt to manage. Thus, due to a proliferation of data, corporate information is stored in electronic or digital form rendering the storage, access, and retrieval of data easier. With significant technological development which

helps in harnessing information, the number of databases has risen. Organizations therefore, need to pay attention in order to undertake and maintain proper database security measures.

At the heart of any data security program must be a database security program. As stated by [SI07], most of the world's sensitive data spends more than 95 percent of its time in a database, most commonly an Oracle database. If a company's database is compromised, it could likely have serious percussions on the viability of its business. Therefore, the database has two important roles: First, serve the data - databases are commonly referred to as data servers; Second, protect the data, acting as an impenetrable safe or data vault. That is the reason why it is called "the crown jewels."

In the last several years, there has been a substantial growth in potential vulnerabilities as well as real attacks on applications and databases. Many of these threats are a result of the changing nature of enterprise applications and databases. While, a decade ago, databases were usually kept physically secure in a central data center and accessed mostly by applications within the corporate borders, today, applications and databases may be distributed in business units to meet local needs. In fact, applications and databases are increasingly made available to suppliers, customers and business partners in order to conduct business over the Web. It is a business imperative today that, for example, suppliers can check their customers' production schedules to coordinate just-in-time deliveries of raw materials. However with this increased access comes increased risk. Hence, organizations need to bring the same level of protection to applications and databases as they apply to servers and networks, with solutions that can automatically detect and respond to application-level threats in real time, and that are granular enough to provide access for customers and business partners while keeping attackers out.

2.1.3 Information Systems Security Threats

For an Information System, hardware does not pose much of a security threat. Of course, the equipment might be stolen or might break down, which will require to back up the databases to ensure that data can be recovered if anything bad happens. Similarly, software poses certain threats inasmuch it may contain bugs that may damage data.

Although hardware and software threats are real, the most significant threat to systems and data is presented by the people who use them and the people who would like to. [TH98] Moreover, despite the frequency of external threats, the biggest threats come from inside. According to [Analyst08], 70-80 percent of all database breaches arise internally with no network connection required, and 65 percent of internal threats are undetected. It was also reported that 57 percent of implicated insiders had privileged access to data at the time of breach. In fact, data is often left vulnerable because organizations tend to pay more attention to protecting data from outside agencies, yet overlook that data can be hacked internally by employees. Because insiders are immune to the security provided by firewalls and intrusion prevention systems (IPSs), it is therefore evident that perimeter and network security alone are not enough to stop such breaches.

Besides, according to a survey realized by Absolute Software [ASoft07] to 185 IT professionals, less than one in 100 employees consistently follow company data policies. This is a bad thing because, as we saw, policies secure the 'human element'.

Moreover, it is highly expensive to resolve security issues and often the most costly security attacks are those perpetrated by insiders capable of bypassing traditional security defenses at the network level to exploit vulnerabilities at the application level, such as weak authentication systems or poor methods of auditing application and database accesses. Indeed, more than 95 percent of intrusions that result in significant financial loss are perpetrated by insiders, employees or those with internal access to an organization. [Behn07]

According to [AppSec08], the average cost of managing a security breach for a company exceeds 300$ per breached record. Typically, these costs include credit protection for those affected, increased marketing costs resulting from attempts to recover lost customers and the

legal and public relations costs of managing the breach itself. Speaking in term of costs per user, a breach can cost organizations 90$ per user for investigation fees, communications, clean up and recovery, customer services, lawsuits and increased security audits. In contrast, it costs organizations just 16$ per user to use data security measures to prevent a breach. [TB06]

Despite that, projects have a fixed budget for implementing all security controls, typically 2 percent to 5 percent of the total cost of the current system release.[Ramac02] For this reason, security must be cost effective.

2.1.4 Current Concerns of Information Systems Security

Applications and databases form the core of an organization's information technology infrastructure. Without the business processes they support and the data they hold the business cannot function.

Yet the current state of Application Security[4] reflects the fact that security has been an afterthought. Moreover, applications and databases have been disturbingly neglected compared to the security provided for networks and servers. Through the years, protection of data in transit and storage was the primary concern, and cryptography successfully addressed this problem. However, the threats to applications have evolved beyond those addressable by protocols and cryptography to the software itself. [Cware06] Nowadays, according to [Pavone07] only 25% of the attacks seen today are aimed at the network and host layers, while the largest number of attacks target at the enterprise applications.

Application Security, the protection of an application against security threats, is a dynamic and challenging task as defenses against every imaginable attack should be incorporated.

At its simplest definition, an application is software that runs on hardware and manipulates data. Applications are, in fact, significantly more complicated than this, so they have been abstracted into many layers. From an Application Security perspective, every layer must be considered an application, thereby necessitating the need for security in each of the layers.

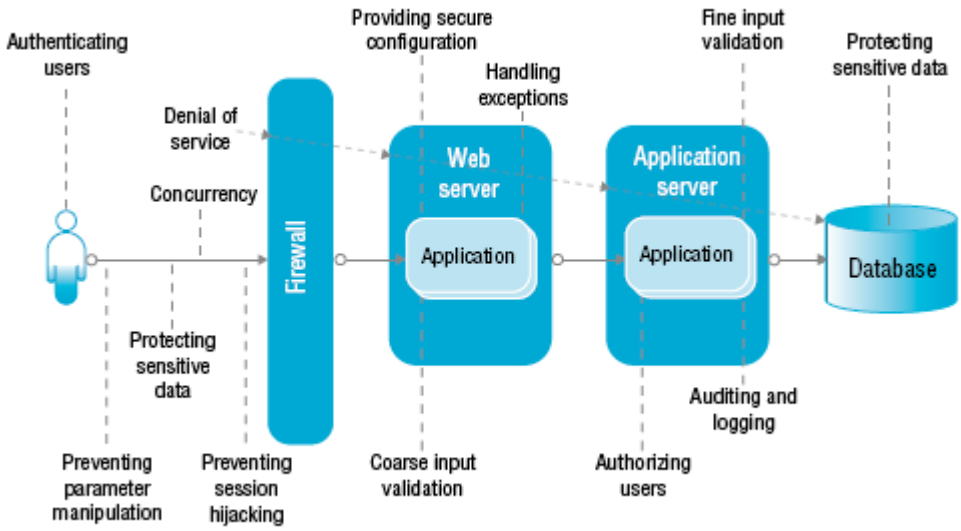

Figure 2.5: Application security concerns [IBM09]

Application Security is comprised of Network Security, Data Security and Software Protection. The following figure shows the points within a system that might require protection.

4 The term "application security" is meant to apply to all facets of applications - not only the user interface but also the databases, and the network.

Network Security addresses external attacks generally against resources inside the firewall providing a service across a network. In other words, Network security consists of keeping the intruders out and has traditionally been addressed using firewalls, intrusion detection systems and virus scanners. Data Security is the protection of data, used locally by an application or transmitted between users and servers, from corruption and undue access. Software Protection is the protection of the software, or services rendered by the software, from attacks, thereby preventing theft of intellectual property and licensed content and ensuring that the software continues to function as intended. Typically these attacks include reverse engineering, tampering, copying, and automated forms of these attacks that can be launched across the network or on a desktop by relatively unsophisticated attackers.[Cware06]

Data protection is the key driver behind Application Security, however, Application Security must extend beyond traditional network and data security to incorporate the need for Software Protection (defense in depth principle). The approach to Application Security must also be driven by a clear and full understanding of the potential threats at each point in the system or network.

Past approaches to Application Security have largely focused on preventing access to databases via perimeter defenses around applications. Perimeter security includes items such as firewalls and intrusion detection equipment. This is also known as the eggshell approach. Then, as tools became more sophisticated and as corporate networks and applications became more interconnected, next step, was to create reactive defenses. However, perimeter and reactive solutions do not address the many threats that an organization faces today. Nowadays, while these are no longer enough, Security experts and corporations realize the need to make applications inherently secure by building in Application Security up front. [Cware06] This means that the application security function should work closely with the application architecture team, at the outset, to improve application security (security by design principle). Instead of treating the symptoms, as is done with reactive approach, working with application architecture eliminates application security issues before they are a problem, therefore being less expensive because there is not the need of retrofitting security measures after the application is deployed.

2.1.5 Information Systems Security Basic Measures

Organizations that understand the importance of their applications and databases recognize the need for proactive, dynamic tools that can find and stop attacks on applications and databases before they cripple the company.

Basically database security can be broken down into the following key points of interest. [Wied09]

- Server Security
- Database Connections
- Table Access Control
- Restricting Database Access

Server Security

Server security is the process of limiting the access to the database server itself. Servers are frequently targeted by attackers because of the value of their data and services, and thereby should be carefully planned. This is one of the most basic and most important components of database security. It is imperative that an organization does not let their database servers be visible to the world. The basic idea is that people cannot access what they cannot see.

If an organization's database server is supplying information to a web server, then it should be configured to allow connections only from that web server. Also, every server should be configured to allow only trusted IP addresses.

The following are examples of common security threats to servers:

- Malicious entities may exploit software bugs in the server or its underlying operating system to gain unauthorized access to the server.
- Denial of Service (DoS) attacks may be directed to the server or its supporting network infrastructure, denying or hindering valid users from making use of its services.
- Sensitive information on the server may be read by unauthorized individuals or changed in an unauthorized manner.
- Sensitive information transmitted unencrypted or weakly encrypted between the server and the client may be intercepted.
- Malicious entities may gain unauthorized access to resources elsewhere in the organization's network via a successful attack on the server.
- Malicious entities may attack other entities after compromising a server. These attacks can be launched directly (e.g., from the compromised host against an external server) or indirectly (e.g., placing malicious content on the compromised server that attempts to exploit vulnerabilities in the clients of users accessing the server).

Database Connections

Web application connects to a database. Users supply the user name and password, establish the connection, and run the query. If users are allowed to make updates to a database via a web page, the system should validate all updates to make sure that they are warranted and safe. For example it should be ensured that any possible SQL code from a user supplied input are removed. If a normal user should never be inputting data, system should not allow the data to ever be submitted.

Table Access Control

Table access control is related to an access control list, which is a table that tells a computer operating system which access rights each user has to a particular system object. Table access control is one of the most overlooked forms of database security because of the inherent difficult in applying it. In order to properly use Table access control, the system administrator and database developer will need to collaborate.

Restricting Database Access

Internet based databases have been the most recent targets of attacks, due to their open access or open ports where they listen to. It is very easy for criminals to conduct a "port scan" to look for ports that are open that popular database systems are using by default. The ports that are used by default can be changed, thus throwing off a criminal looking for default open ports.

Many database attacks go unnoticed by organizations until long after the data has been compromised. Attack methods used to break into databases are very simple , such as exploiting weak passwords and lax configuration, and capitalizing on known vulnerabilities that go unpatched. Vulnerabilities can be classified as follows [AppSec09].

- Vendor bugs
- Poor architecture
- Misconfigurations
- Incorrect usage

Vendor Bugs

Vendor bugs are buffer overflows and other programming errors that result in malformed commands doing things they should not have been allowed to do. Downloading and applying patches usually fix vendor bugs.

Poor Architecture

Poor architecture is the result of not properly factoring security into the design of how an application works. These are typically the hardest to fix because they require a major rework.

Misconfigurations

Misconfigurations are caused by not properly locking down the database. Many of the configurations options of database can be set in a way that compromises security. Some of these parameters are set insecurely by default.

Incorrect Usage

Incorrect usage refers to building programs using developer tools in ways that can be used to break into a system, such as SQL Injection.

So, the first step for protecting a database application is to identify flaws and potential vulnerabilities. There are full of tools that can be used to automate the process of discovering the most prevalent vulnerabilities. These tools are called vulnerability assessment (VA) tools or vulnerability scanners. VA tools scan the database instances and return a report showing what changes should be performed to make the database more secure. The results are presented in the form of a security report where each problem is classified and a recommendation is provided. These checks and recommendations usually cover the items specified in:

- Checks for software vulnerabilities
- Checks for misconfigurations
- Checks for misuse of the database

All of these checks are necessary to check for vulnerabilities in the database.

Vendor bugs, misconfigurations and incorrect usage vulnerabilities can be easily fixed by database administrators. Moreover, if software defects account for 35% of vulnerabilities, configuration errors account for a whopping 65% of vulnerabilities. This means that the biggest cost-to-benefit ratio in terms of avoiding vulnerabilities is an investment in configuration management, assessments of configurations, and repeatable (safe) configurations. [Natan05] Nevertheless, only a small portion of organizations undertakes this effort, while according to zone-h.org, 45% of attacks make use of vulnerabilities rather than brute force. Indeed, typically, data breaches occur for one of five reasons:

- Ineffective management of patches
- No security scanning
- Weak database level security
- SQL Injection
- Lack of real-time security monitoring

Hardening is a process by which systems are made more secure and is sometimes referred to as locking down the system to protect it from unauthorized access. When hardening the system environment, vulnerabilities that result from lax configuration options are removed and vulnerabilities that are caused by vendor bugs can be compensated. The essence of the process involves three main principles. The first involves locking down access to important resources that can be misused—maliciously or by mistake. The second involves disabling functions that are not required for the implementation, which can be misused by their very existence. The third

principle is that of least privileges. The purpose of system hardening is to eliminate as many security risks as possible.

There are two documents that provide very mature guidelines and best practices in terms of hardening system. One is the Database Security Technical Implementation Guide (STIG) developed by the Defense Information Systems Agency (DISA) for the Department of Defense (DoD). The second is the Center for Internet Security (CIS) Benchmark developed by the CIS. Some best practices given in these guideline are: remove all unnecessary material from the hosting servers and only install services that are required for the application/system to function; remove or lock predefined accounts that are not used; change the password for accounts that have a predefined password; remove predefined roles, options, privileges and components in the database software that are not used; install patches as soon as they are released and available and keep aware of the security bulletins that pertain to the database; restrict system operating access; ensure limited file permissions for database configuration files.

Systems are often configured to bring up services and allow connections that are not required. It is usually easier to install a system with its default configuration rather than define precisely what is and is not required. Vendors always prefer to have an all-enabling starting configuration because it avoids problems that can be interpreted as "the system not working." Hardening is a task that requires work but should not be neglected.

Unfortunately basic system security related with misconfigurations and inappropriate usage of the database by web and/or application servers, as well as network and operating system security are a little out of the scope of this thesis. We do not go into the details of how to secure these layers. Here, we will focus on the software application and database vulnerabilities due to poor architecture and lack of security mechanisms.

2.1.6 Advanced Database and Application Security

Database application security is not a "one size fits all" problem. The risks, size, and complexity of applications differ, the level of security awareness among team members varies, and most importantly the goals of each organization are different. Hence, just as customized applications are necessary to meet business needs, customized security solutions must be engineered to match each application. This customization process includes tightening vendor products, as well as integrating securely designed and implemented custom-written portions of the application. However, although software applications can vary widely in terms of the use for which they are intended, all applications tend to have fairly similar security needs in terms of such things as levels of identification, authentication, authorisation, auditing, integrity and privacy required.

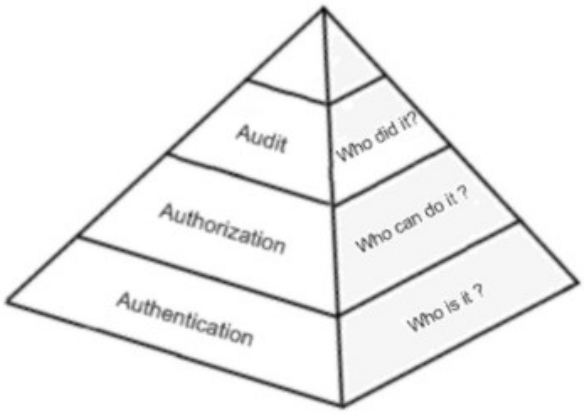

Figure 2.6: Aspects of a good security model

Nevertheless, applications are rich in functionality, and creating a good application security layer is difficult. This is a common theme in securing the core, and both application security as well as database security tend to require more complex technologies and a deeper understanding of environments than those that were developed as part of perimeter security. The other interesting note is that application security overlaps with database security. Application security is first and foremost about securing application data. Although some solutions protect the application server or secure the application from denial-of-service attacks, most of the topics addressed by application security involve the data, which almost always resides in databases. In fact, an application may behave precisely in the manner that the designers intended through normal usage; but through unforeseen manipulation, attackers may use the application to gain access to restricted resources that organizations assume are protected. Application security can act as the best tool to control application usage and prevent malicious users from accessing critical data and resources.

As was stated earlier, a secure application may not only prevents malicious external attacks from succeeding, but it also should detect internal attacks or other threats to the data it processes. The reason is that the majority of security breaches that occur are by those individuals that are within the organization.

Therefore, a key component of any security effort is the prevention of insider attacks. Threats from the inside are difficult to identify and resolve because of the very nature of the action. In many cases, insider attacks are premeditated and deliberate, but organizations must also recognize that non-malicious insiders can inadvertently access and distribute sensitive information. We can group insiders into three categories [AppSec07]:

1. Authorized and Intelligent: These “authorized and intelligent” users are described as privileged internal employees that use IT resources appropriately and in accordance with the defined security policies of an organization.
2. Authorized and Dangerous: These “authorized and dangerous” users are described as privileged internal employees that make unintentional mistakes that appear as malicious or fraudulent and compromise the defined security policies of an organization.
3. Unauthorized and Malicious: These “unauthorized and malicious” users are described as individuals within an organization that mask their identity and their behaviour for the purpose of compromising the security of the database.

Identifying the “unauthorized and malicious” user is very difficult and requires a system designed to effectively monitor privileged activity and not simply named privileged users. In addition to understanding the type of user, it is also important to understand the type of attacks that commonly occur within the categories of intentional and unintentional. The most common attacks for each category consist of: [AppSec07]:

a) Unintentional Authorized User “Attack”

Accidental Deletion: An authorized user inadvertently accesses sensitive data and mistakenly modifies or deletes the information.

Policy Violation: Inadvertent, but non-malicious security policy circumvention, which could lead to unintentional security breach.

b) Intentional “Attacks”

Unauthorized Access: User steals a co-workers user name and password and accesses and steals sensitive information.

Hacking: SQL injection, denial of service, malicious bots, password attacks, penetration attempts, database platform attacks utilizing known vulnerabilities.

The best way to guard against insider breaches is to monitor network access and to monitor the database for unusual activity and set thresholds that represent acceptable use for different classes of users. It is also recommended to do access audits accounts to identify which users have access to what information. A good security control must be capable of rapidly detecting security breaches at the application level, but also prevent application security breaches through strong access control and authentication solutions.

Identification and Authentication

An important aspect of the identification process is identity preservation and propagation. It must be ensured that user identities are available everywhere security requires them. The minute the identity stops, security stops. From a secure database perspective, this is a particularly acute problem. It often occurs that the user authenticates to an application, and the application connects to the database not as the actual user but pseudo-anonymously. When designing database applications, the more information it can be provided about the user's identity, the finer levels of security can be applied. "The more the identity, the better the security." [Knox04] Furthermore, preserving user identity to the database enables the use of database security that is consistent with the defense in depth principle.

User authentication is the first step in many end applications. Authentication is the process of establishing the validity of a claimed identity. There are many authentication schemes: authentication based on something the user knows, like User Ids and passwords; authentication based on something the user knows and something the user owns, such as SecurID tokens and Smartcards; and biometric schemes that authenticate users based on something they know, something they have, and something they are, like fingerprints or facial recognition.

The authentication method should be determined by considering the sensitivity of the data and the privileges and access rights of the user as well as by balancing other competing factors. The first guiding directive is based on what we are protecting. The second factor for determining appropriate authentication mechanism is based on what the user can see and do. In fact, the greater the privileges and the greater the access, the stronger the authentication.

Identification and authentication are the gateway to the database, but for the vast majority of systems, the gatekeeper requires no more than a valid username and password pair to allow anyone to pass. This does not mean that it is necessarily a bad authentication mechanism. Username- and password-based authentication are secure enough for almost all systems, but only when used properly. Here, properly means, among other things, ensure that every user has a strong password.

This is especially important, because 80% of attackers' successful break-ins were due to weak username and password combinations or the use of standard passwords. In fact, weak passwords are a primary target for attackers to gain unauthorized access to databases and other systems. To do that, they use powerful automated attack tools, and, believing [SI07],there are more password crackers out there than any other kind of hacker tool.

Weak passwords are easy to guess. This includes more than the passwords that are easy for a person to guess, but also those that are easy for a computer to guess. A password that meets any of the following criteria is weak [SI07]:

- It appears in the English dictionary;
- It is the name of a well-known city anywhere in the world;
- It is the name of any professional sports team;
- It is a calendar date;
- It is a simple pattern, such as abcdef, 98765, or jjjjjj;
- It is the same as the username;
- It is less than six characters long.

It takes little time to an attacker, with a password cracker tool, to crack a weak password and break into a database. Therefore, where username/password is used for identification and authentication to the database, requiring the use of strong passwords can help prevent simple and more sophisticated methods for guessing at passwords. Thus, best practices recommend that passwords should meet some criterion that defined its strength, such as complexity, reuse, expiration (life time), encryption and history requirements.

If the authentication mechanism is insecure, nothing else matters. To reinforce this idea, here is a good phrase written by [SI07]: “A system with weak passwords and minimal password controls is like a safe with the combination taped to the lock.”

Authorization

Once authenticated, the database knows who the user is and can assign a set of privileges, but this is already outside the scope of authentication and is part of the authorization mechanism.

Authorization is also sometimes referred to as access control. An access control solution must include multiple methods that work together to allow authorized users but also protect, deter, and detect unauthorized attacks. The security solution to be implemented should be set based on the value of the asset to be protected. This assessment should be performed by the data owner and be based on mission criticality and Confidentiality Level. For example, applications that manage or store confidential customer, employee and corporate business data must be a high priority. Therefore, determining the value of the assets being protected and the site-specific constraints are the first steps to consider when selecting access control mechanisms as part of a security solution.

It's also important to refer that this selection process must also be based upon a risk analysis, which carefully identifies and considers the threats, risks, and costs associated with each solution. For each application it is necessary to determine how it can be exploited, what would happen if something were to happen, and how to mitigate such occurrences. Failure to conduct a risk analysis could result in the implementation of ineffective countermeasures to mitigate vulnerabilities. This is very important because sensitive data may hold information that, if discovered or released, could lead to some undesirable outcomes. Then, the golden rule is: “The least possible access to the fewest possible people provides the least opportunity for misuse of the data”. Any user not so authorized that gains access to sensitive data would constitute a compromise of the integrity and confidentiality of the data.

For this reason, access must be based on a comparison between the user’s trust level or clearance and the sensitivity designation of the information (MAC – Mandatory Access Control). Additionally, access control systems must allow the asset owner to specify explicitly the types of access each entity has to the protected asset (DAC – Discretionary Access Control). Overall, access is enforced by both DAC and MAC. The DAC allows users to grant and revoke privileges at their discretion. MAC mediates access to data based on matching the user’s authorizations to the labels attached to the data. The important point is that DAC can’t override MAC.

In computer systems security, Role-Based Access Control (RBAC) is an approach to restricting system access to authorized users. It is a newer alternative approach to traditional MAC and DAC, and it's sufficiently powerful to simulate both. In applications or databases using role-based access control (RBAC), users gain access based on assigned roles. Roles are defined within the system based on job functions within the organization with regards to the authority and responsibilities of the users assigned to each role. This meets both separation of duties and assignment of least privileges according with the application functions users are required to perform.

For example, an accounting application may define user roles based on discrete financial application functions such as General Ledger, Accounts Receivable, Accounts Payable, and the

Financial Manager. General Ledger functions require read/select access to some specific data, insert access to other data, and update or delete access to still other data. The same would be true for Accounts Payable and Accounts Receivable functions. Although all functions may require access to the same data, the same access to the same data may not be required. In this example, roles can be created for General Ledger users, Accounts Payable users, and Accounts Receivable users. The specific access type required to the data is then assigned to these roles. Database user accounts are then assigned the role and thereby, the privileges appropriate to perform their assigned function(s). Security is optimal when each function can be broken down to a level of least privilege and the user assigned all the roles required to perform all of their assigned functions.

Multi-tier applications may utilize a single database account to access the database for application user functions and enable database roles per the requesting user's authorized privileges. The application may also choose to use multiple database accounts based on the job function being performed. The use of multiple accounts may provide a better means to employ the database to assist in auditing functions or other security functions, but the final solution would be dependent on the specific application.

The big challenge here, is to strike the right balance between providing workers with appropriate access on a need-to-know basis and protecting sensitive information as much as possible.

Audit

Monitoring constitutes another important security requirements that must be taken into account. Indeed, monitoring for suspicious activity and system misuse is a cost-effective way to detect issues and react to them.

Monitoring is primarily used for discovery, both of 'how an application is used by real users', and also for 'how it can be misused'. The fundamental value of monitoring is to learn what we do not already know. Therefore, a good monitoring solution should answer to questions like "Who did what, from where, and when?", "What was accessed?", "Did it violate the data permissions policy?". The lack of monitoring, can result in abuse, unauthorized changes, and stolen or misused data.

However, monitoring it's not sufficient. Recording it's also necessary. Auditing combines both activities. Indeed, auditing is the process of monitoring and recording events to provide accountability, track usage, and alert the administrator of potential problems. But, in order to elevate security using auditing, a pragmatic solution must be implemented and the data that is collected through the auditing mechanisms must be able to use. In fact, an auditing solution (and therefore the architecture put in place) has two important parts: the part that collects the information and the part that uses the information. Data is not useful unless it is possible to extract actionable information from the data. In the case of security, this means that the auditing solution must allow to mine the information to expose anomalies, intrusions, mistakes, bad practices, policy violations, and so on. As an example, having a log file that contains 20 million line items every day does not elevate security. In fact, it creates a false sense of security and in doing so makes the environment less secure. If the database environment has no security and auditing provisions, then people are more likely to pay attention to anomalies and various strange events than they would if they think they have some form of security and auditing framework in place.[Natan05]

Auditing can generates a large amount of data everyday. On the other hand, because auditing can cause performance and integration issues, it's necessary to carefully choose the areas in which auditing make sense:

- Specific tables whose data is vitally important to company are excellent potential candidates for auditing.

- Keep track of who is creating specific kinds of objects in the database to help ensure that an unauthorized person is not siphoning off information to which he or she should not have access.
- Track accounts that have had failed connection attempts because someone may be attempting to guess their way into the database.

Additionally, because of the performance impact, log analysis can optionally be performed offline rather than in real time. This procedure might not prevent intrusions but only detect them after the fact.

2.2 Security Impact and Benefits

As we saw, the security of applications has evolved from an option to a requirement. The reality is that application security is typically underfunded, because it's difficult to convince management why an application that appears to be up and running just fine needs additional protection, when the benefits of such security are difficult to quantify. Related to this is the fact that is difficult to evaluate and recognize the adequacy of the security measures. We cannot prove that some security is 100 percent effective; we can only prove that it's not effective. The truth is that systems that are 100 percent secure are not possible to build.

Moreover, change is hard and risky, and while security may reduce costs in the long run, it forces companies to pay costs immediately without knowing the full potential returns. But it's worth making efforts in this direction, to mitigate security problems and avoid data to become compromised. There have been a significant number of cases where information was disclosed maliciously or inadvertently, often because of an innocent or silly mistake. That was the case of the credit card processing company CardSystems Solutions who suffered an attack, in 2005, by an hacker that was able to exploit vulnerabilities in order to gain access to the company's internal network and retrieve detailed data stored in a database. The same happened with ChoicePoint (a data aggregator and information broker), TJX, and more recently Société Generale where a trusted insider, with sinister ambitions, racked up a mountain of fraudulent trades that wound up costing the bank 4.9 billion euros.

All these incidents, gave as examples, are a lesson on the importance of external and especially internal controls, regular security reviews and keeping tabs on what users are up to. So, we are able to identify some security benefits such as cost savings from incident reduction, including incident response and recovery costs; and depending on the private information stored (financial/retail/healthcare): fraud reduction; user protection and reputation that, even not being quantifiable to a financial amount, constitute a major justification for investment in security to protect users from being compromised by their trust in the company.

Some companies try to start a comprehensive system security program only to end up nowhere but frustrated. The most common reason for their failure: establishing a "one size fits all" approach to securing their systems. The database environment within a typical enterprise is extremely diverse, and different systems have different needs. This makes it next to impossible to create one overarching set of standards to which every system must comply. The one size fits all approach can fail in several different ways, but the end result is always the same: no real database security program, and an easy target for any person who cares to attack.

The first step in the security program is to establish a plan and a set of guidelines, which is a common point of failure. Some companies will establish a working group in order to build a set of security and configuration guidelines. Problems often arise when the members of the working group, who often represent different functional business areas, are unable to come to agreement on common standards. In the most severe cases, this leads to the database security effort being abandoned before it really begins.

Bear in mind that the time necessary to secure a system can be anything from an hour to a couple of weeks, depending on the sensitivity of the data, the environment it operates in, and several other factors. The important thing is to take the task step-by-step, eliminating the most critical issues first and making real progress from the beginning. Security is never complete and no system is unbreakable, however, it's possible to build significant defenses so that the effort to break in far outweighs any potential pay-off to the attacker.

3 Technological Overview

This chapter presents in some detail the technology which surrounds the base of the project. Additional technologies that appear as a part of the solutions proposed on the next chapters, will be described when appropriate.

3.1 Oracle Warehouse Management System

A Warehouse Management System (WMS) provides the bridge between corporate level production, scheduling, purchasing, logistics planning and order management systems that permit the dynamic response to order demand essential to supply chain management. With an accurate view of available inventory, equipment and staff, the WMS directs the operations that feed components and raw materials to production in the manufacturing environment and fill customer orders in wholesale and retail distribution. With bar code, voice data entry and radio frequency data communications technology, WMS transform conventional warehouses by improving efficiency and productivity. WMS adds levels of control that permit users to better plan and manage resource and inventory allocation in both conventional and automated operations, while providing management and corporate systems with real-time visibility of actual performance. [Hill03]

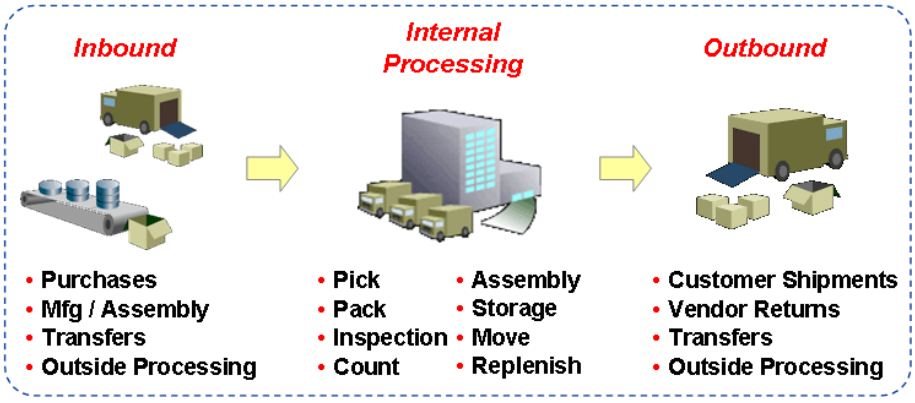

Figure 3.1: Warehouse Management at-a-glance [Mande06]

Oracle Retail Warehouse Management System (ORWMS) is an N-tier, Web-architected Warehouse Management System. ORWMS is the centerpiece of the Oracle Retail Enterprise, a suite of software products that manages and optimizes retail and consumer-direct (catalogue, e-commerce) supply chains, across all types of retailing businesses. ORWMS streamlines the supply chain for multichannel retailers, including store, catalogue, and e-commerce retailers. ORWMS also supports consumer-direct fulfilment capabilities, moving merchandise both to and from the customer faster and at a lower cost.[OraWMc]

ORWMS facilitates the coordinated movement of merchandise and information throughout the distribution process. Using sophisticated, yet flexible configuration and built-in best practices, it ensures the efficient utilization of resources-people, equipment, and space in your distribution process [OraWMc].

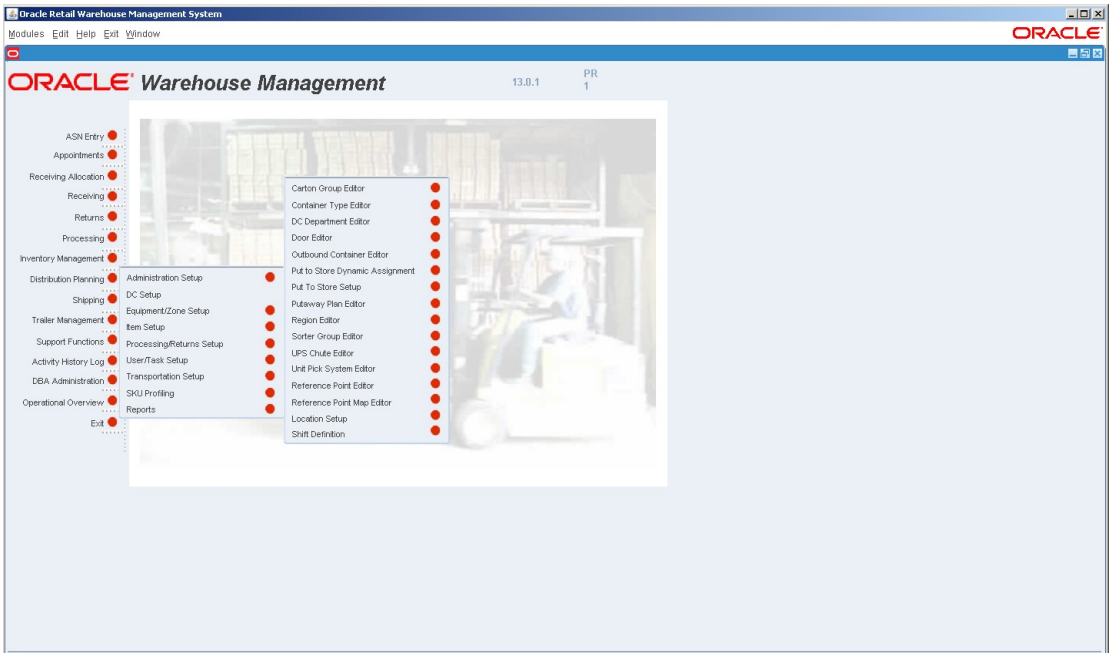

Figure 3.2: ORWMS v13.0.1 Main screen GUI

With Oracle Retail Warehouse Management System retailers can maximize their investment in distribution facilities and equipment, even extending execution capabilities beyond their four walls to increase visibility through trading partner collaboration. [OraWMc].

The list of benefits that follows was taken out from [Oracle09]:

- Accelerate the flow of merchandise through the supply chain, reducing lead times and freeing up working capital.
- Real-time inventory management and best practices provide timely, accurate data, resulting in increased operating efficiencies and improved forecasting, planning, and allocation.
- Built-in best practices, optimization algorithms, and workload monitoring enable increased labour productivity.
- Configurable solution supports all facility types and merchandise flows, including cross-dock[5] and pick-by-line[6].

5 Cross-docking is a practice in logistics that consists in the moving of goods directly from a receiving area to a shipping area, with little or no temporary storage elsewhere.

6 Pick-by-Line is a picking operation, where the exact numbers of cases or items are presented for picking. They may came from the reserve storage area or may be specifically ordered from suppliers for cross-docking. For both cases, the unit load of one product line is picked to waiting customer orders.

- Built-in best practices support all facets of grocery, soft-lines, hard-lines, and consumer direct operations.
- Extend execution capabilities beyond the four walls to trading partners through support of Advanced Ship Notices[7] (ASN), inbound planning, appointment scheduling, and yard management.
- Standard integration to high-speed material handling and sortation equipment like unit, case, and garment sorters as well as pick/put-to-light[8] equipment.

ORWMS manages merchandise in the distribution center via mobile Radio-Frequency (RF) devices. These devices connected via access points to the network are used to perform different operations such as general inventory functions, directed put-away, and movement of inventory, inventory adjustments, returns processing, and cycle counting.[OraWMc]. Some functions are only accessible through these mobile devices.

These devices combined with the use of auto-id techniques (such as barcode), permit actual work performance (e.g. picking an item, counting a location, etc.) to be unified with the recording of that activity in the database server.

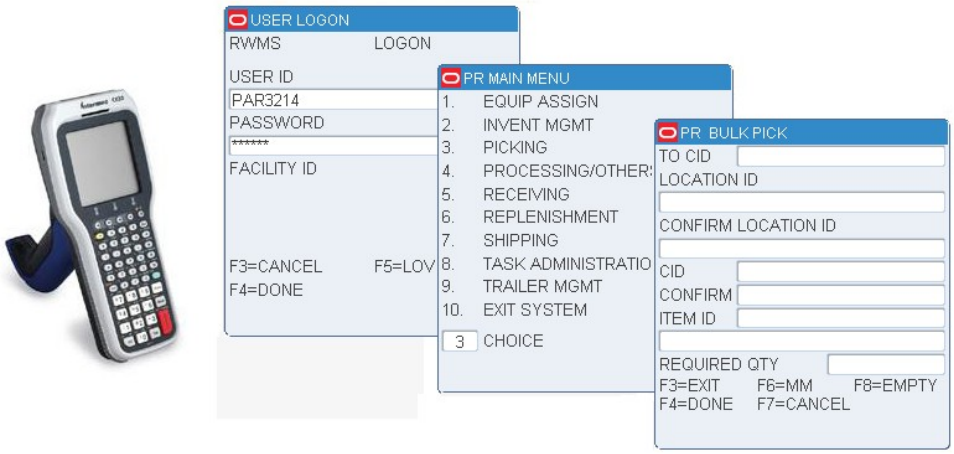

Figure 3.3: ORWMS v13.0.1 RF device screens

In a typical, non-WMS distribution center, materials arrive, and they are then identified and checked in. Receipt data is transcribed for later entry into a host level inventory system. The dock personnel then make a determination as to disposition, and issue or execute move tasks accordingly. The storage locations are maintained in a spreadsheet file and the operators depend upon experience for location selection. Move completions are transcribed for key entry to the host system or manual adjustment of warehouse inventory records and location files. The process is time consuming, paper intensive and error-prone.

In a WMS environment, unless inbound materials have been bar coded by the shipper, the check-in process is essentially the same as that described above. But, this is where the similarity ends. Check-in is executed by operators using WMS-linked radio frequency data terminals. The WMS matches each line item against its records of ASN's or anticipated inbound purchase orders. Exceptions are noted and transmitted to the enterprise system, but the materials are

7 ASN or Advanced Ship Notice is a messages that informs a consignee of the actual dispatch of a shipment and indicates its expected arrival date and other particulars.

8 Pick-to light systems consist of lights and LED displays for each pick location. Software are used to light the next pick and display the quantity to pick.
Put-to-light use technology similar to pick-to-light, however, the light modules are used do direct which container the item is to be picked into, rather than directing which locations to pick from.

accounted for and stored or staged for disposition according to user-defined rules. If goods arrive for which there is no ASN or the purchase order is not on file, the WMS will verify the part number, stage or temporarily store the materials and request disposition instructions from the enterprise level purchasing system.

The WMS then generates a unique bar code tag ("license plate") to be applied to each carton or pallet received and the goods are staged for putaway[9]. This tag number is what the WMS uses to track the specific receipt during its journey through the warehouse. Damaged goods, returns and materials requiring Quality Control are segregated from those to be cross-docked or stored. As check-in and staging for receipts are completed, the WMS immediately tasks operators via radio data terminals to move to the staging area. At staging, an operator scans the bar code "license plate" on a receipt and the WMS determines where it is to be moved.

The WMS maintains a map of each warehouse location by size, capacity and status (empty, full or partially full) and the type(s) of handling equipment required to service it. The system also maintains a "part master" including the dimensions, weights, handling instructions and popularity for each SKU's[10] handling unit of measure (e.g., each, case, pallet). If a receipt is to be moved into a warehouse storage location or forward pick location, the WMS will check the part master for SKU characteristics and then match them against available locations and required handling equipment. The WMS then instructs an operator (with the correct piece of handling equipment) via radio data terminal to move the receipt to the selected location.

Each storage location in the distribution center/warehouse carries a unique bar code identification label. Once at the selected location, the operator scans the bar code to verify that he or she is in position and the WMS responds with an instruction to store the load. If the location is full, the operator will so indicate using the radio data terminal and the WMS will instruct him or her to move the load to a secondary location. In this event, the WMS will also record the anomaly and trigger a location cycle count or send a message to a supervisor.

The process for handling damaged goods, returns and Quality Control samples follows the same procedures with operators tasked to move goods to these areas and verify that they have done so. Items selected for cross-docking or to fill backorders are generally moved to a shipment staging area for consolidation with other order components.

There are multiple variations on the WMS receiving and putaway process described above. What is significant is that every task is time stamped and recorded with the identity of the operator who performed it. The location and status of each piece of inventory is available at all times. Indeed, every time an item moves, WMS records are updated instantly - in receiving, storage, picking and shipping. [Hill03]

9 Putaway - the process of physically placing inventory into storage.

10 SKU - stock-keeping unit. Referring to a specific item in a specific unit of measure Also refers to the identification number assigned to each SKU. Used interchangeably with the terms item and item number.

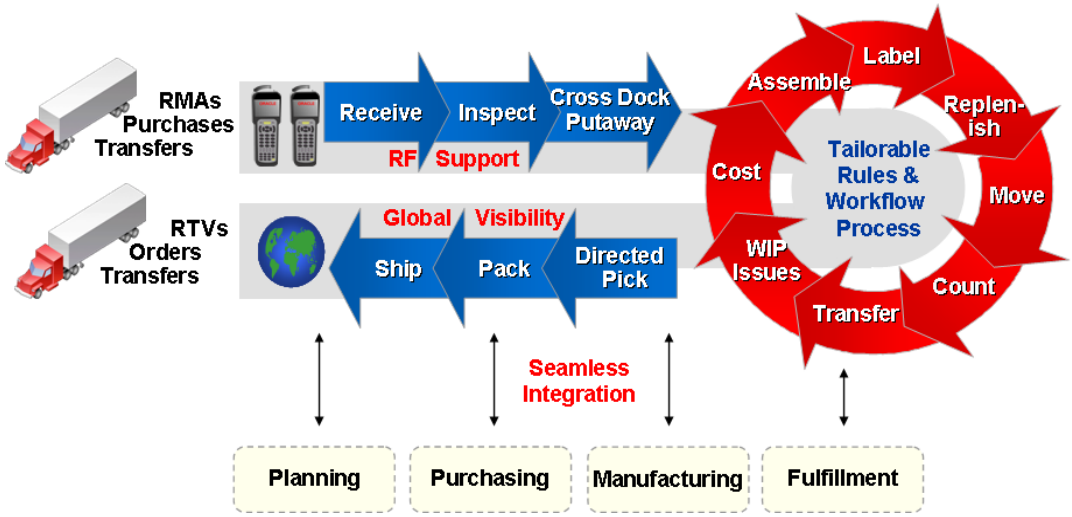

Figure 3.4: Oracle WMS Solution [Mande06]

3.1.1 Oracle WMS Architecture

ORWMS has a flexible and robust true N-tier, Web architecture which utilizes standard Oracle development tools and techniques. Its architecture is based on a Unix/Oracle platform, which interacts with an Oracle database server and an application server. The GUI and the RF terminal's graphical interface are both built using the Oracle Forms technology, and developed with the Oracle Forms Developer tool.

The architecture has the following components:

- A client or initiator process that starts an operation - The user can communicate with the application using thin client support for any standard Web browser (no local software is required), full system access via the corporate intranet or flexible ad-hoc reporting and querying capabilities.
- An application server that performs parts of the operation. It provides access to the data for the client, and performs some query processing, thus removing some of the load from the database server;
- An end server or database server that stores most of the data used in the operation.

The database are kept physically secure in a central data center and is accessed by applications within the corporate borders.

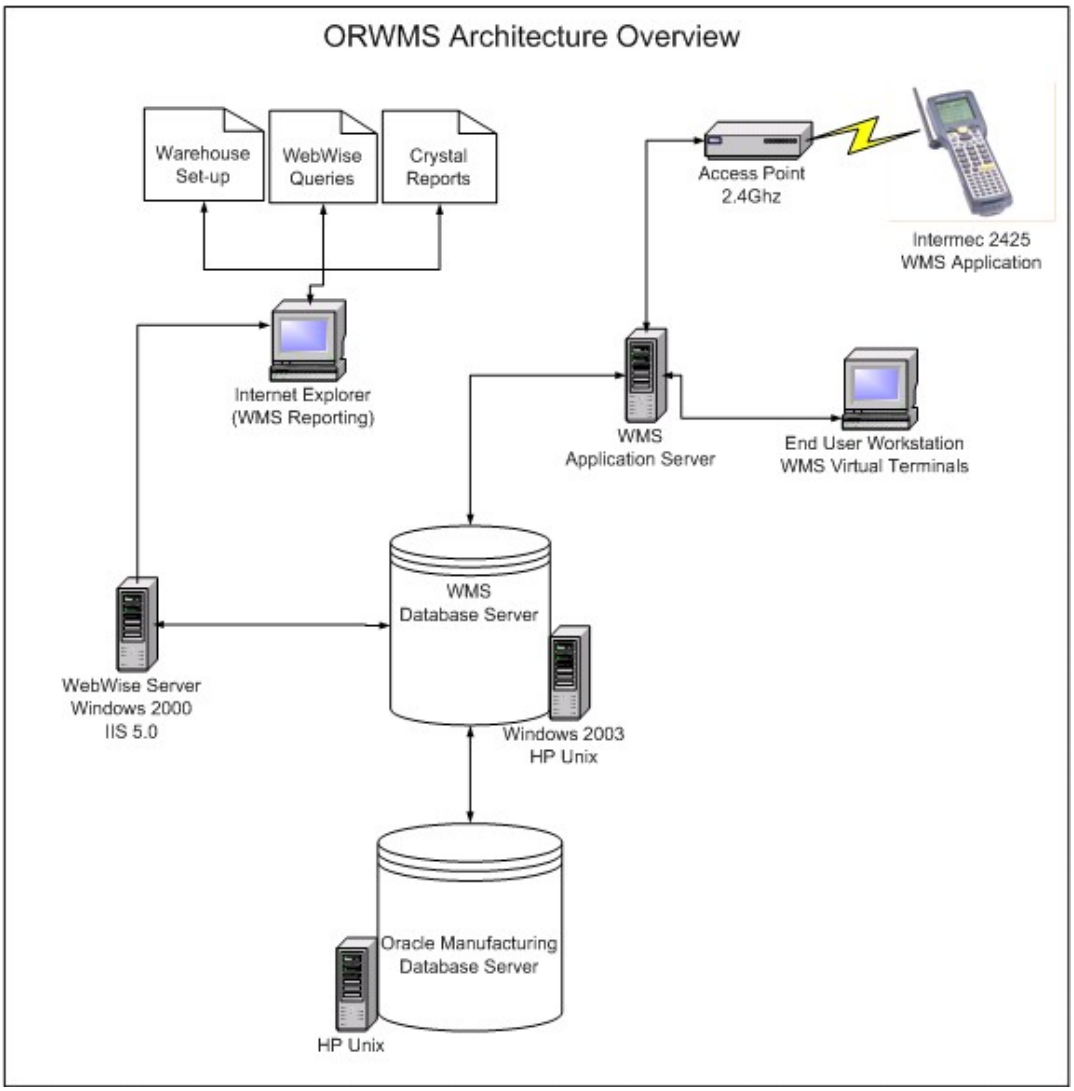

Figure 3.5: ORWMS typical architecture

Oracle WMS is completely integrated with Oracle ERP and Supply Chain Management applications. It is a part of the e-Business suite and therefore provides all the functionality and tools of a regular e-Business installation. A typical Oracle WMS installation consists of the following components [OraWMa]:

- Oracle10g database instance (for the version 13 of ORWMS)
- MWA application server
- Wide Area Network (WAN)
- RF network for mobile devices
- Printers

General Requirements for a database server running ORWMS v13 include [OraWMb]:

Table 3.1: ORWMS general requirements.

Supported on:	Versions Supported:
Database Server OS	OS certified with Oracle Database Server 10g version 10.1.2.2. Options are: • Oracle Enterprise Linux 4.5 x86-64

	• Solaris 10 (SPARC) • HP-UX 11.23 (Itanium) • AIX 5.3
Database Server	Oracle Database 10g Release 2 Enterprise Edition (minimum 10.2.0.3 patchset required) with the following patches and components: Patches: • 5397953 (ORA-07445: [KKPAPITGETALL()+2152] [SIGSEGV] [ADDRESS NOT MAPPED TO OBJECT] [0X34) • 5648872 (SCHEDULER ORA-07445 [OPIDSA() +321] WHEN SETTING UP CHAIN TEST) • 5921386 (WRONG RESULT WITH MERGE JOINT OUTER IN THE EXECUTION PLAN) RAC Only • 5721821 (ORA-7445[KGLOBCL] OCCURED AND INSTANCE WENT DOWN) Components: • Oracle Database 10g • Oracle Partitioning • Oracle Net Services • Oracle Call Interface (OCI) • Oracle Programmer • Oracle XML Development Kit x-Windows interface

3.2 Oracle Forms

Oracle Forms, a component of the Oracle Developer Suite, is a 4GL Rapid Application Development (RAD) environment and a Oracle's long-established technology to design and build enterprise applications, which access, change, or delete data from an Oracle (and other) database, in an efficiently, quickly and tightly-coupled way.

This technology enables the interaction with a database through the use of forms. Oracle Forms accesses the database and generates a default form that presents the data. The source form (*.fmb) is compiled into an "executable" (*.fmx), that is run (interpreted) by the forms runtime module. The form is used to view and edit data in business applications. Various GUI elements, such as buttons, menus, scrollbars, and graphics can be placed on the form. Applications built with this technology have the advantage of being quickly and easily learned and used by users without any particular technical knowledge of database field.

A large amount of the applications that compose the Oracle Retail suite are built with Oracle Forms due to its transparency and facility of interaction with database . This is the case of ORWMS. The version of Oracle Forms used with the ORWMS v13 is Oracle Forms 10g

3.3 PL/SQL

SQL is a declarative language that allows database programmers to write a declaration and hand it to the database for execution. As such, SQL cannot be used to execute procedural code

with conditional, iterative and sequential statements. To overcome this limitation, PL/SQL was created.

PL/SQL is an imperative 3GL and a procedural extension to the Oracle Corporation's proprietary implementation of the SQL database language. It is commonly used to write data-centric programs to manipulate data in an Oracle database, and it supports exactly the same datatypes as SQL.

PL/SQL includes object oriented programming techniques such as encapsulation, function overloading, information hiding (all but inheritance). But, it has one thing that other programming languages don't have: the ability to easily integrate with SQL.

Below are some of the differences between simple SQL and PL/SQL:

- SQL is executed one statement at a time. PL/SQL is executed as a block of code.
- SQL tells the database what to do (declarative), not how to do it. In contrast, PL/SQL tell the database how to do things (procedural).
- SQL is used to code queries, DML and DDL statements. PL/SQL is used to code program blocks, triggers, functions, procedures and packages.
- You can embed SQL in a PL/SQL program, but you cannot embed PL/SQL within a SQL statement.

Server-side PL/SQL is stored and compiled in Oracle Database and runs within the Oracle executable. It automatically inherits the robustness, security, and portability of Oracle Database.

4 Problem Description and Analysis

This chapter describes and analyses the problem that this project intends to address, giving a clear vision of the project itself and its objectives. It also describes the approach methodology used and presents the solution without the implementation details.

4.1 Project Overview

Oracle Retail Warehouse Management System (ORWMS) is the warehouse management system from the Oracle Retail suite. Its function is to facilitate and accelerate the coordinated flow of goods and information throughout the distribution process, still ensuring the efficient use of resources - people, equipment and space - through the use of best practices and flexible configurations tailored to the reality of retail .

As all Oracle applications, ORWMS it's a solid application that offers robustness and reliability in many aspects, but fails especially in terms of internal security mechanisms. In fact, security can be considered the Achilles heel of this Oracle retail application, since this aspect is still far behind the mechanisms implemented on other applications of the suite.

In terms of application database security related, ORWMS only features two security measures: passwords and privilege levels.[OraWMd] These mechanisms are applied to both GUI and radio-frequency (RF) terminals.

Passwords

A user id and a password are required to access ORWMS. There are two parameters that enforce password expiration and renewal. The PASSWORD_OLD parameter informs the users to change their password after a predefined number of days have passed. Then, after an other predefined number of days, the users are prompted to change their password. The PASSWORD_EXPIRE parameter forces users to change their password if it has not been done previously. These two parameters are setted by the system administrator.

A user must change his password sometime between the date of the first prompt and the password expiration date or he can no longer access ORWMS.

Moreover, passwords must be unique and be at least five alphanumeric characters in length. Passwords are case-sensitive.

Example of these password's security parameters:

Password old – 28 days
Password expire – 31 days

Privilege Levels

Each application user is assigned a privilege level, between 1 (Lowest) and 8 (Highest), by the system administrator. Each menu and screen/form has also a privilege level in this range.

ORWMS recognizes each user's privilege level and allows access to screens or menu entries accordingly, based on the rule that a user cannot access menus or screens/forms with a higher privilege level than his own privilege level. Put in another way, a user can only access to all menu options and forms with a privilege level equal or lower than his. The following figure illustrates this architecture with a simple example.

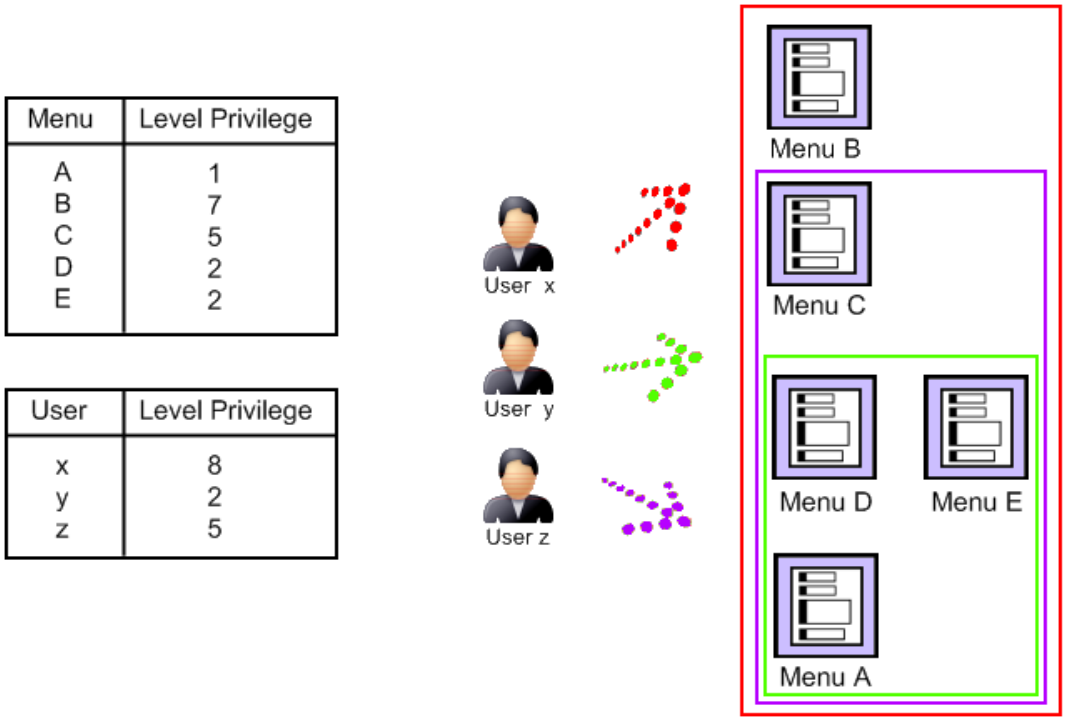

Figure 4.1: ORMWS level privileges architecture example

Note that ORWMS does not allow partial access to the forms (view mode or edit mode), meaning that if a user has access to the form then he is able to perform all the functionalities allowed in the form.

ORWMS application has its own special set of security challenges and requirements. As we will see in the Problem Analysis section, these two mechanisms are not sufficient to provide adequate security as they don't ensure some principles like the 'least privileges' principles or even a control access on a need-to-know basis.

Hence, as it has been already stated, this project aims to propose a security model that could provide a set of solutions that will be able to fix potential threats that arise from vulnerabilities not considered by this two security mechanisms.

4.2 Problem Analysis

Identifying applicable security vulnerabilities on an existing application is a complex task. The flood of security vulnerability sources that are available today further complicates this task.

For this reason, the security related with the network, the operating system (OS) and the web and application's servers will not be considered here. In fact, we assume that the system hardening process was conduced, therefore eliminating as many security risks as possible. This generally involves securing the underlying networks and platforms, restricting database user privileges, removing unneeded functionality, and closing off non-essential modes of access to the system, such as unused default user accounts or network ports, as well as securing the TNS Listener[11] and other configurations and installation files. A system cannot be adequately secured unless all interconnecting components are also secured.

On the other hand, researches has shown that the biggest data security threat organizations face is internal—both negligent and malicious. Thus, we will consider the security of the application and the database from an internal point of view. In other words, we will focus on a Insider Thread Model, where the attacker is internal.

Note that, despite being an internal web application, with database and connecting middle-tier system both located on an internal network, all this hardening process is necessary. In fact, a common assumption is that systems behind the perimeter firewall are not exposed to the same types of attacks as typical web applications, but this assumption has proven disastrous in many cases.

The analysis phase began by reading all the Oracle manuals related to the application. Then, a fresh application was set in a Virtual Machine with the Cent OS Enterprise Edition operating system. The tnsnames.ora and listener.ora files had to be configured, and some data had to be inserted in the database.

Finding and diagnosing vulnerabilities requires a combination of application software expertise, security experience, and knowledge of the business. So, we started to study the application and the existent security mechanisms. The study consisted of understanding all potential threads that could arise, according with the project's scope established.

Once the study and analysis phases have been completed, we identified the major vulnerabilities security-related. Then, in order to identify additional vulnerabilities and prioritize the identified ones, we drawn up a short questionnaire to be answered by the ORWMS pool experts of Wipro Retail.

The questionnaire began with a brief presentation of the project's purpose as well as a description of the vulnerabilities already found. According to their experience in the field, each participant evaluated the vulnerabilities in terms of their priorities, and the impact on the application and database security.

Unfortunately, the questionnaire didn't go the way it was expected, as it was believed to gather more answers. In fact, only 4 people answered to it.

The questionnaire can be consulted in Appendix A.

A letter of the alphabet was assigned to each vulnerability. This will identify unambiguously the vulnerabilities throughout the document. However, note that some of the vulnerabilities identifiers given in the questionnaire differ from the identifiers given here. (From the questionnaire, vulnerabilities B and C were coupled together. Vulnerability D is now C).

4.2.1 Data Security

In most applications, adequate security controls are built into the system to ensure authorized access by user id and password. This is also the case of ORWMS.

These end-user’s credentials constitute sensitive data, since unauthorized modification of this information may result in unauthorized granting and denial of privileges or access to users.

11 The TNS Listener is the component of the database system that listen for network connection attempts, checks to make sure a valid database connection request has been sent, and then passes the communications on to the appropriate database or security identifier.

The same applies to the privilege levels assigned to each application user. It is then obvious that the access to such data must be controlled.

In ORWMS database, these data are stored into a single table (dms_user) which is accessible by everyone with select privilege on that table, by doing a simple SELECT * FROM dms_user; The following figure shows the results of that query.

	FACILITY_ID	USER_ID	USER_NAME	USER_PRIVILEGE	USER_PASSWORD	USER_LOGDATE
1	PR	PAR3214	RDM Schema Owner	9	PAR3214	16-08-2008 17:51:49
2	PR	RDMUSR	RDM User	5	RDMUSR	06-03-2009 2:50:27
3	PR	PAR1111	RDM User	9	par1111	05-06-2009 5:44:18

Figure 4.2: Query application users result

The same is valid for insert and update operations. So, as we can see, the user passwords are stored in plain text and are therefore likely to be modified. The same applies to user privilege levels data. This fact constitutes a vulnerability that can be enunciated as follow:

Vulnerability A:

Visualization of data within the *user_password* column when accessing the *dms_user* table; Possibility of changing the user's authentication credentials;

Visualization of data within the *user_privilege* column when accessing the *dms_user* table; Possibility of changing the user's privilege level.

Protecting user identities can be just as important as protecting the data the identities get access to. Hence, the solution should control the view and modification of such data.

One might think that application users don't have access to the database without passing through the application, and that only the DBA has direct access. However, a user can circumvents the application, guesses the database account login, or depending on the company policy, be allowed to directly access the database. All possibilities have to be taken into consideration.

Distrust and caution are the parents of security
Ben Franklin (1706-1790), Poor Richard's Alamanack(1733)

Database access control is the fundamental mechanism for data security, therefore account sharing between groups of users is a bad practice. While that's great for performance and scaling, it really rules out a lot of nice database features that would ease the security and auditing needs. Moreover, if the application security fails, or the user circumvents the application, then the security can be compromised.

In the current ORWMS implementation , when a end-user login itself on the application he/she only needs to provide his/her user application credentials. Indeed, ORWMS have a built-in authentication service that obviate the need for the user to provide a database authentication. This increases the usability, but once authenticated in the application, each connection to the database is made using the same DB login. This is known as the *One Big Application User* authentication model.

Furthermore, the application connects all end users to the same database schema. That is not necessarily bad. However, in doing so, the application schema is granted all privileges required by all users (otherwise the application will not be able to perform the operations required by all users). The problem with this design is that it's difficult for the database to separate the security privileges for the different users since they are all connected to the same

schema. Ensuring that only the right privileges are available to the user is left mostly to the application. From an auditing perspective, the user's identity is not natively supported with this design, so their individual actions may be untraceable as well as unregulated.

Summarizing, the use of the following features is compromised by the One Big Application User model:

- Auditing - A basic principle of security is accountability through auditing. If all actions in the database are performed by One Big Application User, then database auditing cannot hold individual users accountable for their actions. The application must implement its own auditing mechanisms to capture individual user actions.
- Roles - Roles are assigned to database users. If application users are not database users, then the usefulness of roles is diminished. The application must then craft its own mechanisms to distinguish between the privileges which various application users need to access data within the application.

Hence, the vulnerability can be written as:

Vulnerability B:

All the application users are sharing the same DB account, and consequently any privileges granted directly to the schema are available to all users that are mapped to that schema. Therefore, the least privileges and defense in depth principles are violated.

Consider the scenario where there are three user groups—one with read-only access, another with read and write access, and an administrator group that can create and drop objects as well as read and write. If all three user groups are mapped to the same database schema, then the application must regulate what privileges to enable and disable based on what it knows about the user. From the database's perspective, all users have the same privileges.

This violates least privilege principle because all users share the same privileges, being not possible for two users with different roles, actually have the different roles when they log in to the DB, according to the tasks needed to do their jobs. Furthermore, this violates defense in depth principle, because all per-user security enforcement cannot be made at the database level, and must be done by the application itself.

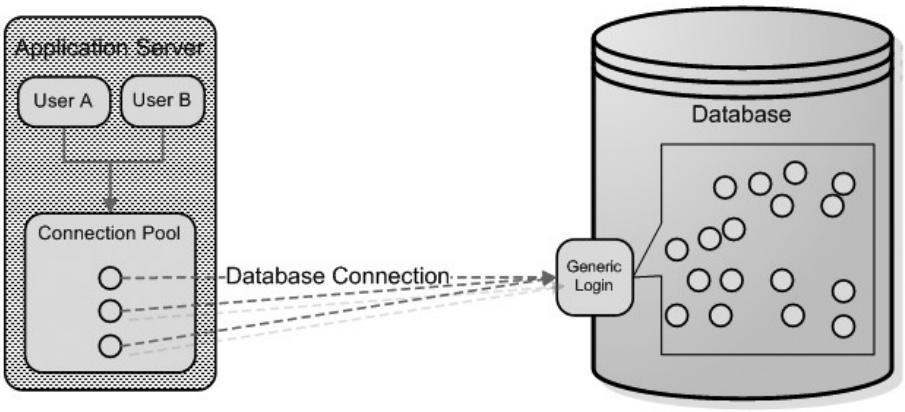

Figure 4.3: User models mismatch [Natan05]

That's true that with multiple user models, as a consequence of n-tier architecture, the application user model and the database user model drift apart. In fact, nowadays, application

logins are no longer commonly associated one-for-one with database logins. Instead, the application server manages a connection pool of database connections. Every time an application thread needs to access the database it requests a connection from the pool, uses it to execute queries and/or procedures, and then surrenders the connection back to the pool. Each connection in the pool is logged into the database using the same database login. Therefore, all of the database authorization mechanisms become trivial and cannot be used effectively or even used at all. This is not properly secure.

As it's completely impractical to go back to a one-to-one relationship between application logins and database logins, due to management and resource limitation problems, a solution that align the user models (user model in the application, and database user model and privileges) has to be found.

4.2.2 Application Security

The best way to ensure the security of a database is to secure the application. Indeed, application security affects database security.

When users are granted access privileges that exceed the requirements of their job function, these privileges may be abused for malicious purpose.

Consider the previous example gave in Figure 4.1, and suppose that the 'User x' is a clerk administrator and the 'User z' is a picker. It's obvious that this two users have different functions and therefore different level privileges needs for doing their job. Suppose that the administrator needs only to accede to Menu C to do its jobs. For the picker, access to Menu D and A is needed. It's easy to verify that this model doesn't gives access on a need-to-know basis. Indeed, both users can access menus and forms they do not need to perform their duties. This clearly violates the least privileges principle and, therefore, could potentially pose a security threat.

Vulnerability C:

The current authorization architecture, based on the 8 privileges levels, allows an operator to visualize/use menus and forms with lower privileges, and whose operations are not necessary for doing its job. Therefore, the least privileges principle is violated.

An effective access controls in place must ensure that users can only access the data they need to get their job done, preventing a user to access a menu entry or a form if it is not needed for the user. Therefore, when application authenticates end users, it should decide what interface to give to whom.

4.2.3 Minor Vulnerabilities and Additional Security

Besides the vulnerabilities presented above, some additional security could be done. As these vulnerabilities don't pose much security threats, they are considered as minor vulnerabilities.

Password Security Enforcement

As seen in Chapter 2, passwords are the most prevalent form of authentication to Oracle Databases. For this reason, they also constitute the first attempt to break-ins. In fact, duping the user authentication mechanism is still the easiest way to break into databases, and nothing makes that easier than sloppy use of default usernames and passwords, weak passwords and

shared passwords. Therefore, having strong passwords is the first and the most cost-efficient mechanisms to prevent attackers to penetrating a database.

The ORWMS doesn't force the use of strong passwords. Indeed, workers can use the password they want to, because the only security enforcement it provides is password expiration date. That is not sufficient and therefore consists on a vulnerability. With the daily contact, it's easy for a malicious worker to intercept the username and password of a colleague, just by "looking over its shoulder", and thus doing bad actions on behalf of his colleague.

The solution must force compliance with best practices to ensure the use of strong passwords.

Effective Audit

ORWMS already includes some log/audit mechanisms. These mechanisms log the labour productivity for each application user, registering the start time and the end time of an activity, the code of the operational activity as well as the total number of operations performed (i.e. number of containers received, picks completed, etc.). There is also a table with all occurred errors, which logs the five-digit error code used to identify the error, a timestamp representing the moment at which it was detected, location where the error occurred, source program where the error occurred, and the identifier of the database user. There is also an activity history log associated with the database user.

Although important, these logs doesn't help to ensure accountability inasmuch the application user are not known.

Auditing is a complementary process of the security cycle, and when done effectively it can act as an invaluable tool. As an a operational system, too much audit mechanisms in ORWMS can be a penalty in terms of performance. Nevertheless, additional audit, especially the monitoring and recording of selected application user actions, could be a good help to identify problems and thereby ensure users are doing only what they are supposed to.

4.3 Methodology Followed

Secure an application database system is a continual process. Indeed, security is not a thing that is applied only once, but instead it has to be constantly on top of concerns' list.

The core tasks in this effort starts with identifying and assessing all vulnerabilities, selecting which fixes come first, implementing them, and then monitoring for intrusions. Once this cycle of work is done, it should be repeated to discover new systems vulnerabilities.

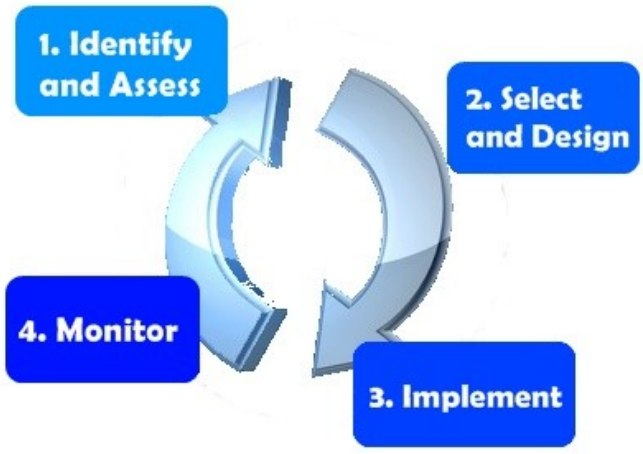

Figure 4.4: Security Lifecycle

For this project, we tried to follow a similar process, taking into account the particularities and constraints of the project itself.

We started to study the application with the objective of: firstly, getting familiarized with it and enlarge the knowledge in this area; and then, understand the potential threats that could emerged from the common application users, beyond its usual and unusual usage. Since security is in many ways a defensive art, it is important to understand potential attackers. Once the initial study was done and the threats were understood, we identified the major vulnerabilities at both database and application levels, considering the assumptions already described in section 4.2 .

In the second phase, we aimed at prioritize the vulnerabilities discovered, in order of their security impact if they were to be exploited. To achieve this goal, a questionnaire was elaborated and then sent to the Wipro Retail ORWMS experts working on the field. Due to feeling the need of security requirements as a consequence of their long experience in ORMWS implementation, we considered they were the right people to help in this task.

The questionnaire had also a second purpose: make known the main vulnerabilities already identified and ask the experts their opinion on certain points. Then, in addition to providing aid in prioritizing vulnerabilities, the answer to questions such as “Does it make sense to strengthen password security mechanisms ?“, “How should the users be allowed to access forms/menu options?” guided toward the identification of the correct security solution and helped to ensure the use of appropriate tools and techniques.

After analysing the answers contained in the questionnaires returned, we carefully identified and reflected on a set of solutions considered able to fix the security threats. Then, all these results and the work done since the beginning were condensed in a document to be presented to several people related to this application area. After all the feedback have been generated, the resulting consensus of emitted thoughts was used to determine the next steps to be taken.

In the third phase, the design of the solution was perfected and finally, a prototype for the top-priority vulnerability countermeasure was initiated with the aim of serving as a proof-of-concept for the solution designed. Then some tests was made to verify the proper functioning of the solution.

4.4 Project Requirements

Security of information systems is a very broad field. Hence, many things could be done and the scope for addressing security could reach huge proportions. Therefore, it had to be reduced according to the manpower and time availability. This already has been explained in the previous sections.

As an internal project, there was no defined set of requirements, but only some wishes and desires emitted by Wipro customers. Unfortunately, it was impossible to contact the main and most interested stakeholders, and therefore proceed to a good requirements specification process. Therefore, the security problem was defined as consisting of threats, and assumptions and the specification was made by studying and analysing the application.

As a final remark, it is important to state that this project had several technologies needs, thus being necessary to carefully balance the value and sensitivity of data with the usability, administration, and costs associated.

4.5 Solution Overview

According with the vulnerabilities discovered, the solution should consider the application and the database as a coupled entity. In this approach, the database plays a central role in application architecture but is allowed to protect itself against application misuse by an user/attacker, in order to limit the exposure to data access attacks that may originate from the

application. Security should not be only handled by the application layer to avoid that any security breach that occurs at the application layer can immediately translate into a serious security incident at the database level.

From a security perspective, the database implementation is much better. It guarantees that the SQL, and thus security, will always be enforced. This helps to provide defense in depth in the case that the application is successfully attacked. In the latter case, the security of the application itself has been compromised, and it is only the database security that will now ensure that an attacker does not gain access to unauthorized data.

Additionally, the approach should be consistent with the other security tenets described in chapter 2: least privileges and security by design. Unfortunately, the latter can not be followed because all the security mechanisms that are part of the solution will be bolt afterwards. Indeed, ideally, security should have been thought during the ORWMS design phase.

We consider that in the ORWMS context, most of internal threats cannot be considered to be malicious but rather accidental. This belief are due to the low criticality and monetary value of the stored data.

Security orders information within the system according to some notion of value. The greater the value, the greater the loss to the business if the information is lost or stolen. Therefore, the solution should also be appropriate to the system use, system environment, information sensitivity and business risk tolerance. Unnecessary mechanisms should not be considered.

Finally, in the design of solution, transversal aspects such as performance and usability should be considered.

5 Solution Design and Specification

As it has been explained in the project objectives, the main goal of this project was to propose a security model that could face the security vulnerabilities present on ORWMS.

This chapter will focus on giving a closer description of the proposed model. The complete solution can be seen as a set of multiple components/solutions, each of them with the purpose of solving one vulnerability.

The solutions will be explained in terms of its requirements, advantages and disadvantages to support the decisions taken, as well as the issues that arose during the process.

5.1 Solution Specification

In last decades, we have witnessed the growth of security standards and technologies. Nowadays, software vendors provide feature-rich security solutions and components at a high level of complexity and maturity. They also usually have a strong background and a better knowledge of security. As a consequence, building our own components is rarely an option, and security architecture work is primarily integration work. [SI07]

Furthermore, an ideal solution should provide comprehensive security mechanisms to proactively protect application and sensitive data without impacting performance or changing system architecture.

Bearing this in mind, the solution specification focused mainly on the study of solutions using Oracle technologies, that could fulfil the purpose sought and be integrated in the current architecture. Indeed, application's specific vulnerabilities should dictate the technology to use.

It's also important to refer that this process was lead by the answers to the following two questions: "why we are protecting the system?" and "what we are protecting it from?". These questions are very important because if they are not understood, we might apply a security solution to a problem that either doesn't exist or can't be solved with the tactic considered.

To get started, it was necessary to identify the business value of the database in order to map on the security controls to suit business needs. Taking into account the competitiveness of the retail industry, information is considered the core of retailers' business models. Thus, the ORWMS database can be assigned to two categories: business impact and business critical. Business impact because the database support business operations. Business critical because the database must be running in order for the business to run, however, it doesn't contain data that if stolen could cause irreparable harm to the business.

With the database classified, it was easier to make smart decisions about which to secure and how far to go in security. This way, the database gets only the security features that it requires, eliminating the excess workload caused by forcing the database to be too much secured. Remember that the goal of implementing a security model is to provide information security and protection, however, it's essential to keep balance between the protection capability and cost, performance, and operational considerations.

The Table 5.1 reminds and summarizes the vulnerabilities found during the analyses phase. Similar to what was done for each of the vulnerabilities, a unique identifier was assigned to each specification.

We will presented each specification in the same order in which the vulnerabilities were presented. For some vulnerabilities, several solutions were considered. Then the reasons why some of them were not suitable for the whole problem is explained.

Table 5.1: Summary of vulnerabilities.

Vulnerability identifier	Vulnerability description
Vulnerability A	Unauthorized view/modification of user's credentials and access level attributes;
Vulnerability B	Account and privileges sharing between all application users - User models mismatch
Vulnerability C	Users can visualize/use application menus/forms with lower privileges than theirs but whose operations are not necessary for their job
Minor Vulnerability – Password enforcement	There is no password's best practices. The only mechanism existent is password expiration.
Minor Vulnerability – Efficient Audit	There is no accountability for the end-user. Besides a labour productivity log, the users' activities are not audited.

5.1.1 Solution Specification - Vulnerability A

It is a good practice to not allow an end user to have access directly to the database. Thus, data is protected. This works great until the user finds a way to connect to the database directly. Indeed, security should not be based on the assumption that a user cannot accede the database if he or her doesn't know the database account credentials.

If a user discovers a valid user ID in the database, then any method of connecting to the database can be used to view the data. Therefore, sensitive data should be protected to prevent such events to lead to potential threats.

ORWMS stores the end-users password without any kind of protection. This data can easily be accessed and therefore must be protected.

Solution A1 – Hashing and database role-based security

When passwords are used for authentication, they shouldn't be stored in plaintext.

Encryption seems to be the natural solution, but it's not. *Encryption*, which is the process of converting plaintext into undecipherable text, implies *decryption*, which is the process of converting the undecipherable text back into plaintext. The golden rule of passwords is that they

are never disclosed to anyone at any time. Encrypting passwords, which allows for potential decryption, could allow this disclosure to occur. [Knox04]

Hashing appears as the solution. *Hashing* takes plaintext and converts it to undecipherable text but, unlike encryption, there is no way to un-hash something. There is no way to take a hashed value and determine what created that value. Because of this property, hashing is called a *one-way function*. Another important property of hashing is that the same input to the hash will always generate the same output. Therefore, hashing is a good solution for securely storing the authentication data.

For the user's privileges level, however, hashing don't seem to be a good solution. Indeed, since privilege level has a limited number of values (a number between 1 and 8), and as hashing the same value provides always the same output, the resulting hashed value can easily be guessed. In spite of that, due to the less sensitive level of this information, hashing still constitutes a lightweight and clean solution.

Looking at the password hash tells nothing about the password. Therefore hashing solves the visualization problem inasmuch it prevents people of acceding to the real value of sensitive data, but it doesn't solve its unauthorized modification. For this problem, it's necessary to make use of the role-based security offered by the database. That includes performing the following steps:

- Create several database's accounts;
- Create the necessary database's roles;
- Create the appropriate views using column-level access, and restricting insert and update operations on the sensitive columns;
- Grant the view to the roles;
- Grant the roles to the users.

Alternatively, the database provides a column level security mechanism by default. For inserts and updates, it's possible to control at the object level whether a user has the ability to affect the values within the column. The ability for the database to support column-level privileges allows the user to insert or update specific columns in a table while simultaneously restricting modifications to other columns.

Therefore, both alternatives ensures that if a user accede data directly via database, he or she will only be authorized to make use of the privileges assigned to that account.

But as we already saw, besides administrator accounts, a typical ORWMS implementation has only a database account, that is used by all end-users to accede to data through the application. Moreover, the process of password renewal or change made by users is done in the application. Thus, this account should be able to update passwords. Here, the solution will be to restrict this privilege only for direct access, and not when used by the application. This can be made using Secure Application Roles technology. Privileges are granted to a role that is disabled by default for the database account, and when the application connects, it would have to explicitly enable the role during runtime, and thus the privileges.

Since this technology also responds to another vulnerability, it will be better described later in this chapter.

Solution A2 -Virtual Private Database (VPD)

This solution uses a technology from Oracle, called Virtual Private Database, which implements fine-grained access control by using row-level security (RLS).

VPD's row-level security allows to restrict access to records based on a security policy implemented in PL/SQL. A *security policy* simply describes the rules governing access to the data rows. This process is done by creating a PL/SQL function that returns a string based on the

user's authorization. The function is then registered against the tables that should be protected by using the DBMS_RLS PL/SQL package. When a query is issued against the protected object, Oracle dynamically and effectively appends the string returned from the function to the original SQL statement, thereby filtering the data records. Furthermore, the security is implemented so that it is transparent and could not be subverted.

VPD is very flexible and very granular. By default, the policy applies to all DML statements, but it's possible to specify which DML operations the policy is to apply. This granularity also allows the database to apply separate policies based on the DML type. For example, the database can easily support a policy to allow all records for SELECT statements; an UPDATE policy to restrict update operations only to user's record; and a INSERT, DELETE policy that restricts DELETE and INSERT operations at all. Multiple policies also can be applied to the same object: the database logically ANDs the policies together.

Moreover, the same query can provide different results regarding how the query was issued or who issued the query.

With this solution, the access and modification of the passwords and privileges columns can be controlled, by ensuring that the original query is modified to incorporate the identifier of the query's author. Thus, the records are filtered to return only the ones related to this user.

Considering our current password and privilege level problem, we could restrict the view and modification of these values to only the user's record. The following figure depicts the situation when each user selects from the dms_user table. The required security condition is automatically enforced.

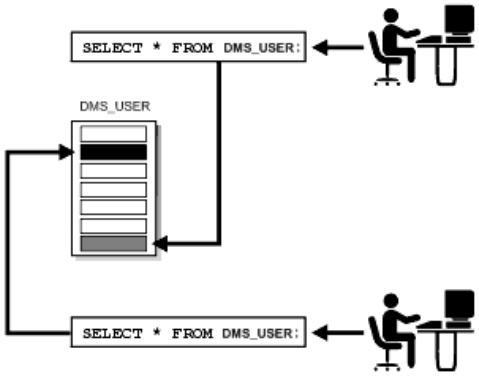

Figure 5.1: Oracle VPD example

Furthermore, it's possible to use the column sensitive option which allows a more selective invocation of the row level security mechanisms. Actually, this option allows to store the sensitive and the non-sensitive information together with the assurance that anytime someone requests the sensitive data, VPD will remove or mask the values. In fact, the security column-sensitive policy will only be invoked when a specific sensitive column is referenced. If it's not referenced, then the policy isn't invoked and the records are not filtered.

One of the strongest arguments for VPD is that the security is tightly fixed to the data it protects—it's consistent, centrally managed, and it can't be bypassed.[PK03] There is no back doors. Moreover, it performs better than Row Level Security built into views, because indexes, bind variables and application contexts, and policy caching can be used.

Unfortunately, having only one database account, ORWMS cannot take advantage of VPD technology. In fact, this solution obliges the creation of several database accounts, in a way every user has its own database account with its own password and own privileges.

Because that is a situation that wants to be avoided due to the need of changing the application architecture, the VPD solution was left aside.

5.1.2 Solution Specification - Vulnerability B

Sharing schemas should only be done when end-user identity can be preserved and the database privileges are identical for all users connected to the same schema. ORWMS uses only one schema and although it can preserve the end-user identity for some audit purpose, the database object privileges needed are not identical for all users.

Oracle WMS uses OCI (Oracle Call Interface) authentication feature. This technology allows a database client to set up, within a single database connection, a number of “lightweight” user sessions. In other words, the application multiplexes database sessions over the same physical connections. These lightweight sessions reduce the network overhead of creating separate network connections from the middle tier to the database. The application can switch between these sessions as required to process transactions on behalf of users. Because physical connections are more costly to establish, this ability is very desirable.

In the ORWMS, unlike typical OCI use, each of the user sessions is associated with the same database user. Remember that ORWMS only use one database account for all end-users. Because all user sessions are created as the same user, this security model makes it very difficult to the database to achieve data separation for each user. So, this must be done by the application.

ORWMS passes the application user-id to the database for some auditing purpose and for retrieving the privilege level associated with that user, to differentiate what the different end users can see and mediate the actions they can perform within the application. The application user identity is stored into a global variable and it is not used for anything else.

The ideal solution should re-establish the user models alignment to allow to handle different object privileges for different end-users, as shown in Figure 5.2. When the application gets a connection from the connection pool, the first thing it should do is to communicate with the database to let it know the identity of the application user, on behalf of whom all queries that will follow on this connection are made. Then, this additional information must be used to implement granular access control. The least privilege principles should be enforced at database level depending on the identification of the connected user. Each user should only has the privilege needed to perform its duties and nothing more.

The whole idea is illustrated in the following figure.

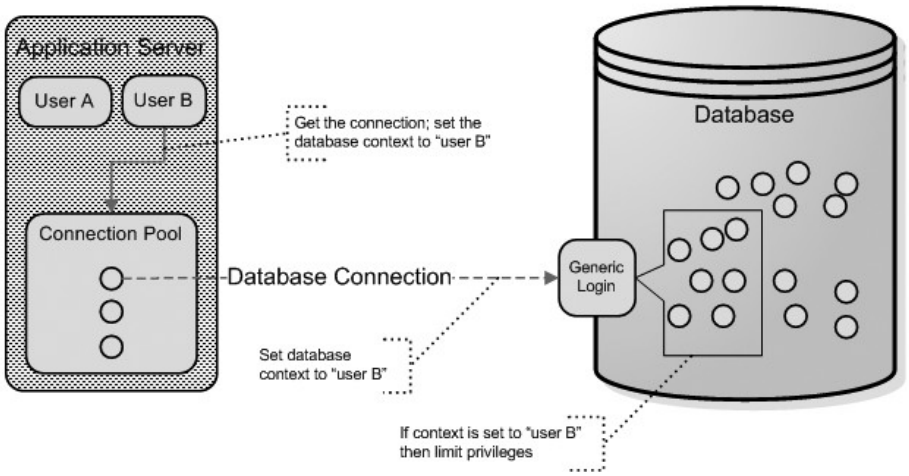

Figure 5.2: User model alignment solution [Natan05]

Solution B1 – Proxy authentication, Global Application Context and Secure Application Roles.

Proxy authentication uses the OCI connection pool. It doesn't require the user's password but in spite of this fact, proxy authentication is still secure because the authentication requires special privileges.

The proxy authentication process is simple. The application first establishes the connection pool to the database via the proxy account. The *proxy account* is the account configured simply to allow the physical database connections (the connection pool) to be established. This account must be protected. It requires only one privilege: the CREATE SESSION privilege. *No other privileges are needed nor should be given to this account.* [Knox04] In fact, the proxy account should only have the ability to connect to the database; it shouldn't have the superset of all privileges of all users. This configuration conforms to the least-privilege principle.

The database needs information about the user for it to provide security. Thereby, next step consists on ensuring the real application user identity preservation through to the database. Once there, the database will utilize this information to restrict user access. Proxy authentication allows the identities and privileges to be preserved, but it has one important requirement: each application user has to be a database user. However, we don't want to have a 1:1 user accounts mapping because it can be very impractical due to management and especially resource consuming when application has several users.

In our context, application users are not known to the database. Indeed, the application simply authenticates users to the middle tier and the middle tier connects to the database as one "big user" and all actions are taken as that user.

We could handle of the application user id by building our own database procedure and passes the user identifier as an argument to the procedure call within the database session. Then it should associate the username it received through the procedure call with the database login that was used to initiate the connection. However, this method requires a few changes at the application level. Furthermore, Oracle already has built-in capabilities for passing such identifiers. For these types of applications the best solution is the use of CLIENT_IDENTIFIER.

`CLIENT_IDENTIFIER` attribute acts like an application user proxy. It's a predefined attribute of the built-in application context namespace, `USERENV`, and it can be used to capture the application user name for use with global application context or it can be used

independently. Global application context stores context information (sets of name-value pairs), in memory, so it can be used by applications that cannot use session-based application context because users authenticate to the application and then it typically connects to the database as a single identity.

When CLIENT_IDENTIFIER is used with global application context, it provides flexibility and high performance for building applications. Instead of each user having his own session set up with individual application contexts, the application could set up global applications contexts for the different type of users. Then, the CLIENT_IDENTIFIER is used to point the session at the correct context in order to retrieve the appropriate type of data. The application needs only to initialize the different global contexts with a particular user or group once, and use the CLIENT_IDENTIFIER to access the correct application context to limit data access. This provides performance benefits through session reuse and through accessing global application contexts set up once, instead of having to initialize application contexts for each session individually.

With this approach, sessions can be reused by multiple users by changing the value of the CLIENT_IDENTIFIER attribute, which is used to capture the name of the real application user. This avoids the overhead of setting up a separate session and separate attributes for each user and enables reuse of sessions by the application.

For example, a user PAR3214 connects to the application. PAR3214 is not a database user, he is an application user. The application accesses the built-in application context namespace and sets PAR3214 as the CLIENT_IDENTIFIER attribute value. At that moment, the privileges for the session are set for the PAR3214 user. PAR3214 completes his job and exits the application. Then PAR1111 connects to the application. Instead of setting up a new session for PAR1111, the application reuses the session that currently exists for PAR3214, by changing the CLIENT_IDENTIFIER to PAR1111 and also the privileges accordingly to the user. This avoids the overhead of setting up a new connection to the database and the overhead of setting up a global application context.

Enabling the database roles and consequently the privileges that must be enabled when the user is connected, should be the responsibility of the application.

One of the problems of assigning roles is that the assigned role becomes the default role of the database user. This means all the privileges assigned to these roles are immediately available to the user. Since only one database user is used and we want to control access depending on the application user, we can't let that happen. Therefore, the default role of the database user should be set to no role. Then, depending on the application user authenticated, the adequate roles are enabled and at the end, disabled again. This is done using Secure application roles and the application context, by setting an attribute with the role the user is supposed to have based on the user's access authorisation. Secure application roles are roles that can only be enabled by authorized PL/SQL packages. This allows specifically to control the role from within an application.

Secure application roles are a great solution when the architecture depends on the selective nature of enabling roles and privileges. As the application connects to the same schema and we want to maintain separate privileges for separate end users, secure application roles can be used to decide when to enable the privileges.

Proxy authentication allows to perform another useful function. Assume that some roles have to be disabled when the user connects via proxy authentication because the user requires no privileges from this role when accessing the database via the application. This can be done. Like this, a user is able to enable different roles and thus different privileges depending on how he or she is connected.

Solution B2 – PL/SQL procedures and Secure Application Roles

This solution is similar to the solution B1 but relies on another technique utilizing PL/SQL package instead of proxy authentication and client_identifiers. This can be done by invoking a PL/SQL procedure that passes user information to the database prior to performing any database-related work within the application. The application has to convey the user's information to the database before any procedures or queries are executed. The package variables are private to the database session. So, additionally, the application has to reset the database state between different and subsequent user requests. All of this extra work is done to ensure that nothing about the previous user's session will leak into the current user's session.

Database object and system privileges should be stored in Secure application roles and selectively enabled and disabled for appropriate users. The application has to manage the process of knowing when to enable and disable roles for the users. The benefit of a Secure application role is that the database ultimately decides whether the role is enabled or not. This is advantageous because it will allow us to modify the security policy without needing to change the deployed applications.

As the secure application role can only be enabled from within a PL/SQL program, this mechanism strengthens security. But there are two security aspects to this. First, the user or application has to have EXECUTE rights on the PL/SQL program to enable the role(s). Second, the PL/SQL program itself will perform a series of verifications and validations to ensure that everything is as it should be *before* setting the role(s). In this way, the PL/SQL program acts as a sentry guarding the role. [Knox04]

The most efficient way is to create a single procedure that performs all the steps—set the user identifier in the PL/SQL package, enable the role(s), do its work, reset the package state, and finally, disable the roles. In fact, one procedure call from the application will be faster than three or four.

From a security perspective, proxy authentication is ideal because database security can be fully exploited with little coding. Moreover, login triggers fire, database privileges and roles are automatically enabled, and auditing can be used to track the end user's actions. Therefore, because proxy authentication offers more simplicity and performance than this solution, the solution B1 was preferred in the detriment of the B2.

5.1.3 Solution Specification - Vulnerability C

When creating a new user in ORWMS it is associated a level of privilege (1 to 8) where the level 8 is the highest. These privilege levels are also associated with ORWMS forms, so when a user tries to access a form, the security process validates if the user can access the form. To access a form a user must have a privilege level equal or higher than the form's privilege.

As we saw, this mechanism is not sufficient to restrict a user to accede only the menus and forms he needs to do its job.

Actually, handing out any more information than is absolutely necessary has never been good for security, regardless of what is being secured.

Organizations need to be able to distinguish between various groups of people, and information that have differing value and differing requirements in terms of security. This is a form of classifying information in terms of its accessibility to people within the organization. But this is no simple task. The solution lies in the definition of security "classification" systems - models by which information resources and people are assigned classification levels which are then used to describe what people will be allowed access to what resource classifications. The ability to control access based on data classification is important for enforcing the "need-to-know"principe.

C1 – Validation by Profile

The idea is to restrict the user's access to menus and forms depending on his profile(s), and one or more profiles can be assigned to a user. The profiles assigned to each user replace his privilege level.

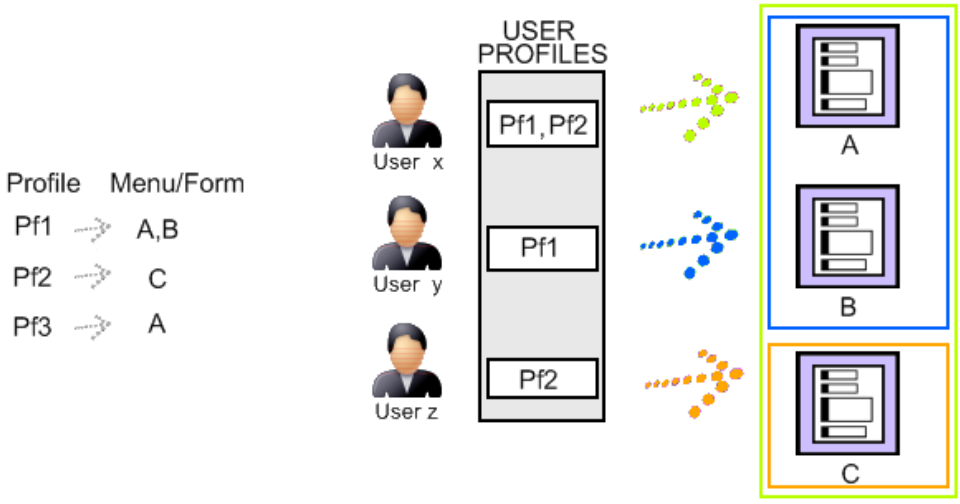

Figure 5.3: Example of Validation by profile solution

Basically, the system will make available only the menu entries and forms that are identified in user profile.

Because it is possible to a have a user with multiple profiles then it is possible to have the same object in different profiles.

When a user deactivates a profile, then only the entries of the profile will be deactivated. The same object for the other profiles will remain active. From the point of view of the system, the user only has to have an active object in one profile to be able to use the object.

The next solution to be presented is very similar but more desirable as it takes advantages of a security proven Oracle technology, being more flexible and requiring less changes to application.

C2 – Oracle Label Security (OLS) and Profile Access Control Architecture

Oracle Label Security (OLS) is a Oracle security technology, based on the work done for U.S. intelligence agencies and the U.S. Department of Defense, and it was designed to meet some of the strongest security requirements ever put forth.[Knox04]

OLS provides a set of PL/SQL APIs, allowing to define a security policy that is implemented by marking the data records with security labels, based on its sensitivity. When the data's sensitivity has been determined, it's marked with a sensitivity designation, also known as a classification. The label markings indicate what rights a person must possess in order to read or write the data. The labels are stored with each record in a special column that is added to the table. The database users are also given labels that indicate their access rights to the data records. When the user accesses the table records, the database's OLS engine compares the user's label with the row's label marking to determine if the user can have access.

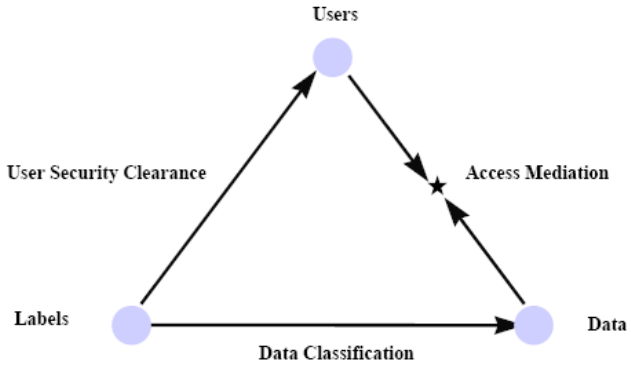

Figure 5.4: Access mediation performed by Oracle Label Security [NW05]

Each label consists of a permutation of three components (at least one level, zero or more compartments, and zero or more groups – like this: *Level:[Compartment1,...,Compartmentn]: [Group1,...,Groupn]*), where numbers are internally used to represent these labels. Only the level component is mandatory but the use of compartments and groups offers more granular access control. OLS first determines what levels the user is authorized for, then determines the groups, and finally, the compartments. When the user's authorizations allow them access to the records, the user's label is said to *dominate* the record's label.

The level is a hierarchical component which denotes the sensitivity of the data. However, there is no requirement to define more than one level. The compartment is non hierarchical component which is used to segregate data. The group component is used to record ownership and can be used hierarchically.

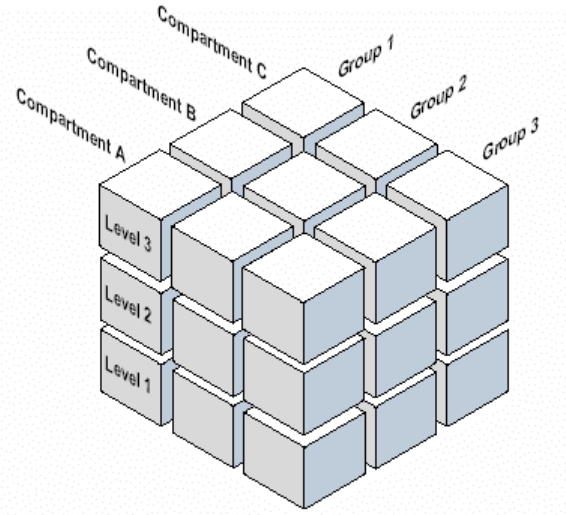

Figure 5.5: OLS data compartmentalization [Oracle02]

OLS also supports multiple ways to classify the data, and allows to create multiple security policies and apply the labels to only selected users and objects.

The idea, here, is to use this technology to build another layer of security on top of the roles. This is done by creating a profile access control architecture with the purpose of separated

the menus options/forms available to individual employees, based on their need-to-know. The user's need-to-know, or what they are authorized to see, is derived from their profile. Here, unlike the above solution, a profile consists on a level and compartments to which the user belongs to. Thus, data is not separated and categorized based on its sensitivity but rather based on the profile that the user must have. Access to a menu/form at a specific classification requires the user to be approved for access, or in other words, to belong to the category of users that have this profile.

For example, remember the scenario imagined in section 4.2.2. in which we had two users, the first one being a clerk administrator, and the second one being a picker. Each user has a different role and therefore must have different need-to-know. Now, consider three menus/forms: one that is only needed by the administrator, another only needed by the picker, and the third shared by both users. With the current ORWMS level privileges architecture, the clerk administrator user would have access to all three menus/forms because its privilege level it's higher than the picker privilege level. This is clearly not satisfactory from an access need-to-know perspective.

The solution proposed here also provides the notion of level. Moreover, if labels use only the level component the solution provides exactly the same result as with the ORWMS level privilege architecture. However, adding compartments and groups, we can achieve an efficient access control, as desired. The decision of using these components should be driven by the granularity of security required.

The solution will use only levels and compartments. Thus, it can be seen as the cross of privileges levels with the category associated to each type of user. The following rules are used, in the sequence listed, to determine a user's read access to a row of data:

1. The user's level must be *greater than or equal to* the level of the data.
2. If label has groups, the user's label must include *at least one of the groups* which belong to the data (or the parent group of one such subgroup).
3. The user's label must include *all the compartments* which belong to the data.

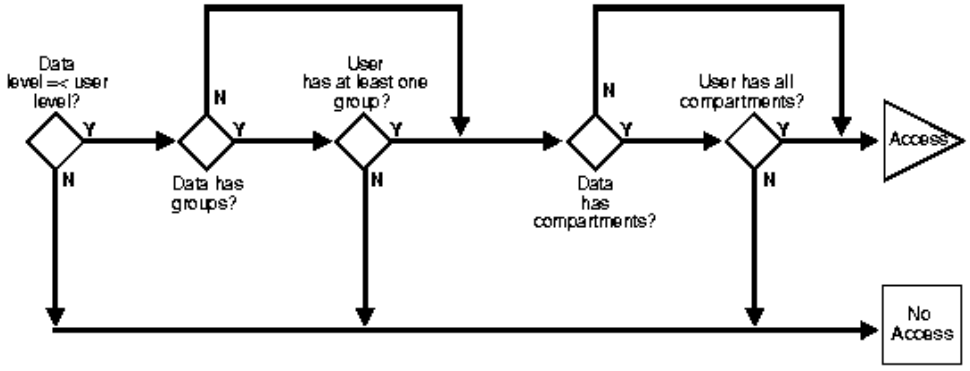

Figure 5.6: Label evaluation process

As a read access request comes in, Oracle Label Security evaluates each row to determine:

1. Is the user's level equal to, or greater than, the level of the data?
2. If so, does the user have access to at least one of the groups present in the data label?
3. If so, does the user have access to all the compartments present in the data label? (That is, are the data's compartments a subset of the user's compartments?)

If the answer is no at any stage in this evaluation process, then Oracle Label Security denies access to the row, and moves on to evaluate the next row of data.

Here is an exemplification. Consider a situation in which three users with picker roles must have access to picking menus/screens. There is a general picker (User A), a unit picker (User B) and a case picker (User C). The first one can do all picking operations. The other two users only can do unit picking and case picking operations respectively. Thus, there will be three compartments. Say:

C1 – pick.
C2 – upick.
C3 – cpick

Now, suppose that Screen 1 should be accessed by all three pickers; Screen 2, which allow unit picking operations, should be accessed only by User A and User B; Screen 3, in which case picking operations are made, should be accessed by User A and User C.

Therefore, screens labels are respectively:

Screen 1 → Level2:C1
Screen 2 → Level2:C1,C2
Screen 3 → Level2:C1,C3

The three screens belong to the same type of operations. Therefore the level of sensibility is the same for all of them. Moreover, each screen has the compartments required to access it.

To allow the access as stated above, the users' profile should be:

(User A) Picker → Level2:C1,C2,C3
(User B) Upicker → Level2:C1,C2
(User C) Cpicker → Level2:C1,C3

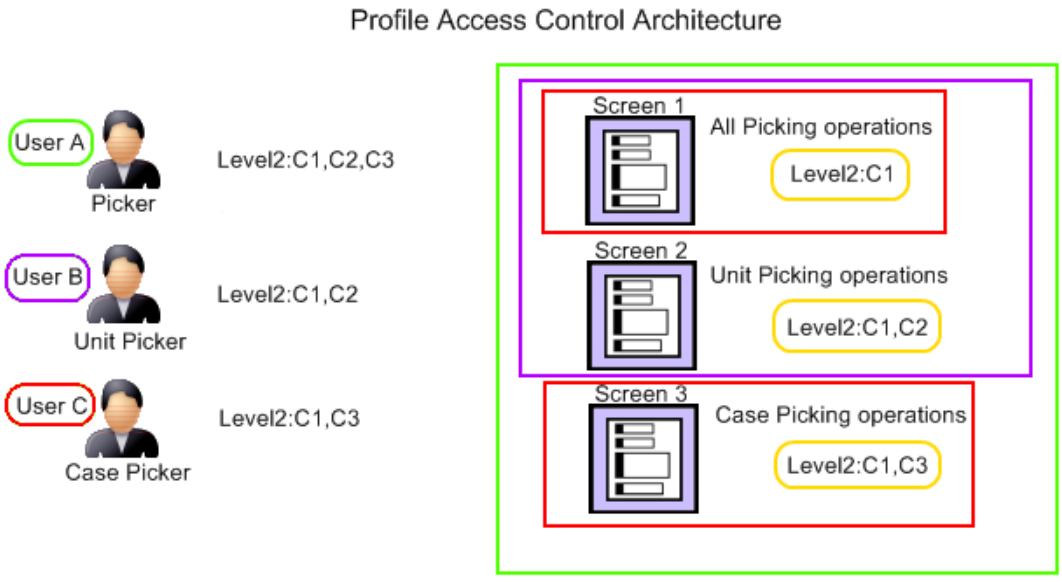

Figure 5.7: Profile access control architecture example

In fact, User A can access to all screens because he dominates the labels. He has all the compartments necessary for acceding to them. Also, User B and User C also dominate the labels associated to the screens they need to do their jobs.

This is commonly referred to as multi-level security (MLS). This solution is therefore suitable because it is able to provide the right information to the right people at the right level of secure data access.

5.1.4 Solution Specification - Minor Vulnerabilities

Password Security Enforcement

For this solution, it's imperative to define a password policy which specifies requirements for strong passwords and enforcement measures. These measures should enforce the choice of good, strong passwords through the use of password complexity routines. Password complexity routines are critical to ensuring that password best practices are obeyed. The complexity routine technically implements the official password policy in the organization.

As the password policy will not be the same for all organizations, we only can define a set of common best practices checks that can be administer within the complexity routine:

- Password isn't the same as the username;
- Password contains at least one digit and a mix of upper and lower case letters;
- Password is greater than some specified length;
- Password isn't the same as the old password and must differ by at least three characters;
- Password isn't an easy to guess word, such as 'manager', 'oracle', or your company's name.

More complexity could be enforced but the context of application use has to be considered, as well as the sensitivity of the data. Additionally, it's important that users can remember the password they choose. If they cannot remember their password, they will end up posting it on a sticky note, in which case password security is back to square one.

These practices reflect the thoughts from the audience of the questionnaire. In fact, the following points were identified from the responses obtained in the questionnaires:

- 75% of people that answered to the questionnaire, strongly agree with the option of forcing the password to not be the same as the username.;
- 50% strongly agree and 50% are neutral about the need of forcing the password length to be greater than some specified length, as well as the mandatory use of a mix between letters and digits;
- 75% agree on a password history, not allowing the reuse of a password previously used .

Effective Audit

According to the answers obtained via the questionnaires, it will be a good thing to have audit mechanisms on the tables related with the users and forms' privileges. The SCP table, which stores the system control parameters, should also be audited. The tables related with activities like inventory adjustments and stock movements are also good candidates to the audit process. The objective should be to identify the "who-what-when-where" tuple.

The appropriate controls, audit trails and activity logs must be designed into the application and therefore be independent from the database. Like this, it will be harder to compromise and will be not sensitive to the vulnerabilities that the database may have. That solution is aligned with the defense-in-depth strategy.

To gain on efficiency, solution must use fine-grained auditing. This allows to be very specific in the details of what actions on an object will be recorded. The audit records are limited based on columns accessed or even the values of those columns. This has two benefits. It makes it easier to find important information in the audit trail because there is less unimportant "noise" to sift through. It also helps limit the size of the logs.

However, due to the targeted tables, especially the ones related with stock operations, the audit process will be space consuming and might not log all of the activities required. Moreover it will slow down the application performance. For these reasons, care must be taken when setting up these audits.

5.2 Putting It All Together

We analysed the potential security solutions and security enhancements to create the new security model.

In the new model, the application uses proxy authentication and client identifiers (application user proxy) for identity preservation, global application context and connection pools for performance, hashing for protecting sensitive data and, label security and secure application roles for granular access control, and strict privilege separation and management.

All these solutions must be putted together and obviously this should be done without breaking the proper application operation.

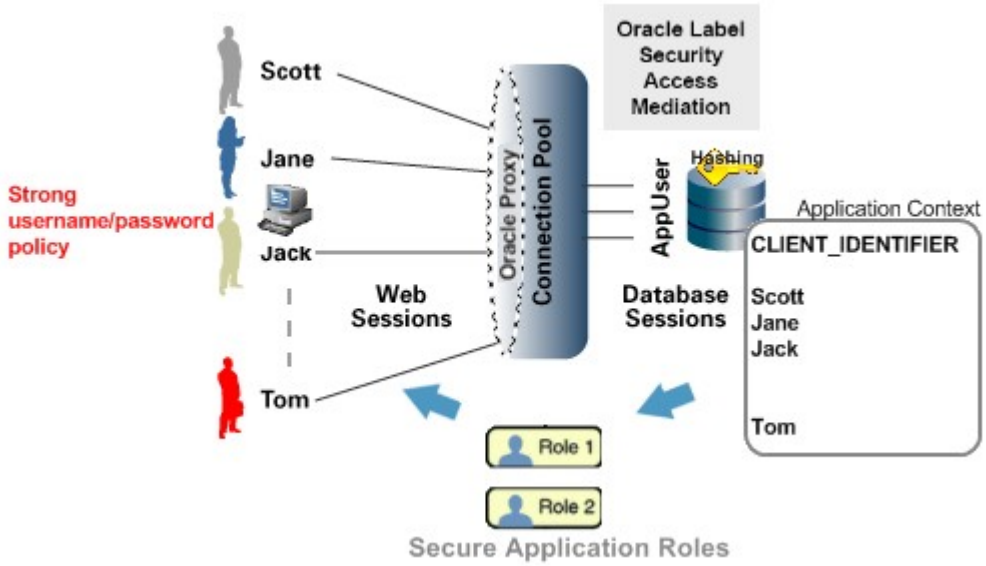

Figure 5.8: New security model architecture

The design decisions determine how effective the security implementation will be. Some of the decisions taken were due to the feedback obtained during the presentation made to the ORWMS key collaborators of Wipro Retail. Unfortunately, due to the overload of their agendas, the audience only counted 3 experts.

The presentation to collaborators started by giving an overview of the project status followed by an overview about security. Then the results of questionnaires were presented, as well as the vulnerabilities and possible solutions to be applied. At the end, an open conversation with the audience about the main points discussed was conduced, to see if there was any important issue forgotten, and to get feedback regarding some points.

From this discussion, it was issued the preference for solutions that did not need the creation of additional database accounts. Also, it has been decided that the password enforcement enhancement would not be well received due to the reality of warehouses and distribution centres. Indeed, the operational nature of these environments requires rapidity on all activities. According to Wipro collaborators' experience, warehouses' workers use a 4 characters password that is written on a card they bring with them. Hence the use of strong passwords and

password management best practices could increase activities' time and consequently lower the workers' productivity. Furthermore, the RF devices are not very practical for the enforcement of mix case password, therefore the usability would be affected. Moreover, due to the low sensibility of the stored data, password enforcement is not a top priority.

In the same way, the performance concern led to questioning the real needs of efficient auditing. Although most of the answers to questionnaires agreed on some additional audit, the feedback obtained during the presentation showed itself more reticent and cautious about such enhancement because of the performance impact. Again, due to the specific environment, the balance between performance and security seems to weigh in the favour of the first.

ORWMS is an execution system and, as such, its primary mission is the management of facility resources, work and material flow to maximize efficiency and productivity. Burdening the system with myriad, and non-critical mechanisms can negatively impact response times and affect timely task execution. For these reasons, these two enhancements were not considered for the model design.

We consider however, that omit password security enforcement is a bad choice and some authentication-related enhancements should be considered. Indeed, all of the security mechanisms in the world will not help an application that performs poor user authentication.

In the next section we will describe the security model's design and architecture.

5.3 Security Design and Architecture

This security design involves many technologies and security features. While this all sounds great, with all these features come increased complexity. Therein lies the problem. Complexity is bad for security. The more features and options we have, the more potential for misconfigurations and new vulnerabilities. Thereby, in order to make a secure design since the beginning, all the new features and components added must be integrated ensuring they will be made secured.

The first step in the security design starts by creating a new schema responsible for the security. This reflects the fact that the data tables should reside in one schema while the security should be enforced from a different schema.

5.3.1 Proxy Authentication Database Setup

In order to get a proxy connection, the proxy user has to be created. This account has to be protected either by strong password or some form of strong authentication. Even though the proxy user has the least amount of privileges possible (the CREATE SESSION privilege), it's not desirable that people can be able to easily guess the password.

Then the proxy user must be allowed to connect as the end users that will be using the application. This is made by altering the user to allow someone to proxy to him.

```
-- create the proxy account

CREATE USER app_user IDENTIFIED BY qej4k9ld;

-- protect the proxy account

GRANT CREATE SESSION TO app_user;
```

```
-- alter the user to allow to proxy to him

ALTER USER wms13 GRANT CONNECT THROUGH app_user;
```

The proxy call simply passes the name of the user to which we want to connect. No other credentials are required. The benefit of this mode is in its simplicity. The application merely needs to determine what user to proxy to, and the proxy connection will occur. From the security perspective, the database privileges have to exist for this to succeed, meaning that there is no significant security risk in doing this even though no other credentials are being passed to the database.

5.3.2 Getting the User Identity

After the user has been authenticated, the real end user identity must be propagated to the database. The approach used is called application user proxy, and it uses the Client Identifier attribute.

Setting and retrieving the Client Identifier is simple. A string is passed to the SET_IDENTIFIER procedure to associate the database session with a particular user. Then, the CLIENT_IDENTIFIER is an attribute of the session and can be viewed in session information. To our purpose, the string has to be the username of the authenticated application user.

The application is responsible for setting the Client Identifier before any database work, as well as resetting the value between requests. To clear the identifier value, the CLEAR_IDENTIFIER procedure in the DBMS_SESSION package must be invoked.

One particular challenge of Client Identifier is that it can be set by anyone to anything, because it uses the DBMS_SESSION PL/SQL package, and the privilege to execute this package was granted directly to PUBLIC. One solution could be to revoke the execute privilege on DBMS_SESSION to PUBLIC and grant it only to the security schema.

However, the revocation of the execute privileges on DBMS_SESSION may break the existing application. Moreover, altering default grants and privileges is generally considered improperly with the Oracle internal mechanisms, and may invalidate the support for Oracle applications.[Knox04]

Hence, the solution has to use a PL/SQL security package or an application context in conjunction with the Client Identifier. We opted for the application context as it provides performance through session reuse by changing the value of the CLIENT_IDENTIFIER attribute. Moreover, application contexts are secure because they are private for the session, they are fast because the values are stored in memory, and they are flexible because they can be set to any user defined string.

When doing this, we are checking to ensure the user hasn’t modified the publicly available Client Identifier. We set the user information in both the Client Identifier and the application context.

First the application context should be created. Then, the application context can only be manipulated by a single PL/SQL program which is specified when the application context is created.

```
-- create the context

create context USR_CTX_ID using secuser.ctx.mgr;
```

Access to the PL/SQL program provides the first level of security. The second level of security comes from the implementation code itself which should performs checks and validations prior to setting the application context. The following interfaces in `DBMS_SESSION` enable to manage application context in client sessions:

- SET_CONTEXT
- CLEAR_CONTEXT
- SET_IDENTIFIER
- CLEAR_IDENTIFIER

An API (PL/SQL procedures) that the application can call to provide all instrumentation information should be implemented. This API will take care of setting the proper values in application context for a specific client identifier. Then, when assigning a database connection to process the client request, the application needs to issue a SET_IDENTIFIER to denote the client identifier of the application session. From then on, every time the client invokes SYS_CONTEXT, only the context that was associated with the set identifier is returned.

To enforce security, procedures would then verify that these values are congruent before doing any action.

The main point to note is how CLIENT_IDENTIFIER and application context interact. The DBMS_SESSION interface for managing application context has a client identifier for each application context. When CLIENT_IDENTIFIER is set, only sessions with the same identifier can share the context data. In essence, the global context data becomes "private" to a given identifier. In this way, application context can be managed globally, yet each client sees only his or her assigned application context.

Since these procedures are called to set the context, they can be called by the function that authenticates the user's password. This function is called every time an application user is authenticated, so it is a perfect place to call these context setting procedures.

The API also should provide a procedure to detach the user from the context by clearing the CLIENT_IDENTIFIER for the session, and a procedure to clear the context.

5.3.3 Securing the Roles

Next it's necessary to restrict what roles the user can enable. Thus, all database roles must be carefully defined with the adequate and correct privileges, assigned to the database user and finally be disabled by default.

Moreover, assuming the application needs only the ROLE_A, we should prevent the application from enabling other database roles.

```
-- create database role
CREATE ROLE role_a;

-- grant role to user
GRANT  role_a TO wms13;

-- disable role by default
ALTER USER wms13 DEFAULT ROLE ALL EXCEPT role_a;

-- grant proxy privileges along with
-- ability to enable the role_a
ALTER  USER wms13 GRANT CONNECT THROUGH app_user
WITH ROLE role_a;
```

To enforce a good security design, the proxy authentication grants should me made to allow only the roles needed by the application. In the example, the configuration restricts all application users to the ROLE_A when proxied from the APP_USER account. All other roles can’t be enabled. However, default roles are automatically enabled upon logon.

Alternatively, we can restricts users to specific roles. Assume WMS13 user, by default, is granted the ROLE_B role. It's a default role that are automatically enabled upon logon. Then, if the user requires no privileges from this role when accessing the database via the application, it should be disabled for proxy authentication.

```
ALTER USER wms13
GRANT CONNECT THROUGH app_user
WITH ROLE ALL EXCEPT role_b;
```

All other roles granted to the user, except ROLE_B, can be enabled.

The user is able to enable different roles and thus different privileges depending on how he is connected. The database supports a distinct set of privileges for the proxy authentication.

That works fine for enabling and disabling the roles shared by all end users that connect using the same database account. Since we want to control access depending on the application user, enabling the database roles and consequently the privileges that must be enabled when the user is connected, should be the responsibility of the application. Therefore, no proxy authentication grants should be made and the profile default role of the database user should be set to no role. Indeed, the solution uses Secure application roles and an additional attribute stored in the application context (app_user_role). Thus, roles have to be created has follows, with the clause 'identified using':

```
-- create a secure application role

CREATE ROLE role_a identified using secuser.secure_user_role;
```

To set the role, a procedure called set_user_role and belonging to the secure_user_role package, should be defined under the security schema, with the purpose of activating the role. The reason is that the package secure_user_role has been used to authenticate the roles, thereby the roles can't be enabled directly or by any means other than calling a procedure from this package. This makes set_user_role a trusted procedure.

When a user accesses the database via the application, the application should know it has to enable the role and it has to do it transparently for the user. Thus, in order to determine which application user has which role, the table dms_user has to be modified, adding a column called user_role. Like this, assuming that the application user PAR3214 logs in, although the database user is WMS13 proxied from the APP_USER account, the client identifier should be set to PAR3214 and the app_user_role context attribute should be set to the role corresponding to the PAR3214 user in the dms_user table.

Note that the name of the role stored in the user_role column should be equal to a existing user database role.

Roles should be enable just after the user has been authenticated. So, a good place to call the role setting procedure, would be in the function that authenticates the user's password. Thus, this function should be modified to retrieve the user_role from the dms_user table in addition to the password, then call the context setting procedures to set the client identifier and the app_user_role attribute. Finally, it should call the procedure to set the role. Unfortunately, it is not valid to call a procedure that sets a role inside a procedure or function owned by another user. In fact, when the context changes to APPUSER (the proxy user), the roles would be unset, since that would create a new session. Therefore, we can't use the set_user_role() procedure inside the password checking function. Thus, the procedure that sets the roles should be called after the login, using the role already available in the application context, under the attribute app_user_role. To do that, an AFTER LOGON TRIGGER should be made.

The role is now properly set. Moreover, the user can't change the context, can't change the role, and can't modify any of the authorized attributes already defined. The model is secure.

As a final remark, all procedures created should check for errors and traps PL/SQL exceptions at very stage.

5.3.4 Protecting Sensitive Data

For the problem of the unauthorized access/modification of user passwords, it's necessary to implement a password verifier. Since the new security model substitutes the privilege level architecture by a profile access control architecture, we only focus on the password column.

The creation of a new user or the update of an existing user should only be authorized by invoking a PL/SQL stored procedure responsible for inserting/updating the user. This procedure should hashes the password provided and store the resulting hashed value. Indeed, direct insert and update privileges in the dms_user table should be revoked for the single DB account existing.

To gain on performance the hash keys should be “stored" in a calculated column that is then indexed. As such, the index contains the hash keys but the table itself does not. The database engine searches the index on the hashed column.

At login, when the user enters a password to authenticate, a database logon trigger should fire immediately and invoke the password verifier procedure. The authentication process is performed by computing a hash of the plaintext user-supplied authentication value (his password) and comparing the resulting value with the one hashed and stored in the database. If they match, the user has supplied the same password and is authenticated.

There are several algorithms available. One of the most secure is to add a long unique random salt value to the hashes to make them unique:

```
Hash = md5('salt' + password)
```

A salt comprises random bits that are used as one of the inputs. The other input is the password. The point of a salt is to make each password unique and long enough that brute force attacks are a waste of time. So, the user's password, instead of being stored as the hash of "m1PAssW", ends up being stored as the hash of 128 characters of random Unicode string + "m1PAssW".

We also want to restrict the privilege of updating the password column on the dms_user when the user access the database directly, but enable such role when acceding via application. This can be achieve by using an application secure role. This allows to prevent the role to be enable outside the application by doing a simple set_role.

5.3.5 Control the Access to Menus

The first step is to create a security policy that will contains the labels, user authorizations, and the protected objects, using the OLS PL/SQL API. When defining an OLS policy, a column name must be provided to store the data classification label. The additional column can be appended as a *hidden* column, thus enabling existing SQL statements to continue working without any changes.

Note that when an Oracle Label Security policy is created, a new database role policyname_DBA is also created. Authorizations to manage policy label components and label authorizations must be granted to this new database role. Once granted, this role can execute the package and create label components, user labels, data labels and administer policies.

An important step is to identify which objects will be controlled. For our solution, the protected objects are the menus entries and forms screens located on the table dms_menu. The next step is to evaluate the data contained in the table. In fact, to control the levels of access different users will have in the system, the specific privileges for each object should be identified. This task should be done by the organization itself.

All users should also be classified. The users' authorizations depends on how the users are classified. A user's classification is called a *Clearance Level* and is used to determine what data a user may have access to. In general, access is only allowed when the clearance is the same level or higher than the classification of the data being accessed . This is what is already done with the current ORWMS control access architecture. Since we want to control the access on a need-to-know basis, each user should be classified accordingly with its roles or compartments which he or she belongs to.

The need-to-know principle places information in compartments. Compartments may extend across security levels, and information and users may belong (have access) to a number of compartments. Remembering what was stated in the specification of the solution C2, we will only use levels and compartments for defining data and user labels, because we do not need more fine grained access control provided by incorporating the group component.

Once the classification system has been finalized, a simple access matrix can then be drawn up, to help in identifying the design:

Table 5.2: Example of an access control matrix.

Access Control Matrix				
USER Profile	MENU OBJECTS			
	rf_invent_menu	hh_begin_unit_picking_s	hh_initiate_unload_s	hh_move_trailer_s
Picker	Allowed	Allowed	Denied	Denied
Transporter	Denied	Denied	Allowed	Allowed
Admin Supervisor	Allowed	Denied	Denied	Denied
Superuser	Allowed	Denied	Denied	Denied

Restricting access based on data classification requires a firm understanding of the various roles and functions that exist among application users accessing data. The first step that needs to be performed is a comparison between the defined data labels and user security clearances. The reason this step is important is to make sure data is accessible to users who should have access based on their job responsibility. In other words, the information required to perform a specific job responsibility might be out of reach to the application user based on their security clearance. In the worst case, data might be assigned a data label that no user can access, effectively hiding the data.

As we want to make a customizable solution, to be adaptable to the specific requirements of organizations, this development implies the creation of some new forms and the customization of some others in the ORWMS GUI interface. With these new forms, an organization will be able to define profiles and assign them to the application's users in accordance with the jobs they have to perform.

However, we recommend the creation of some baseline levels, compartments and labels, general to most organizations, to be assigned to forms and menu entries via the elaboration of a creation script, and then assign the labels using update statements against the base table. These

will be useful because once the policy was applied, no data will be visible until existing data has been assigned a valid data label. This is due to the fact that the policy label column is NULL.

A common way this problem is addressed, is to grant the security administrator responsible for labelling the initial data, the Label Security specific authorization *FULL*. The FULL authorization turns off the access mediation check at the individual row level. This will allow the administrator to see all rows regardless of the data label and ensure that all legacy data rows are property labelled.

The next step is to create the form by which levels and compartments will be created. Two new table should be created to store each new level and compartment defined. The tables should store the policy_name, long_name, short_name, and level_num or comp_num attributes associated with each level or compartment. As the number level_num assigned to the level determines its ranking, it should be asked to the user.

Then, another form should allow to create data labels for the policy, by combining the levels and compartments previously defined. To accomplish that, a procedure that can be executed from within the application should be created, to compute the data labels. Moreover, the labels as well as their attributes should be stored in a table.

After the labels have been created, they should be assigned to each menu and form object. These operations should be handled by a third new form.

Next, the ORWMS form user_table_editor, responsible by user management operations, should be modified in order to remove the privilege level and add the possibility of assigning a profile to a user. The form should allow to choose the objects needed for a user. Then, a procedure should take care of handling the creation of the security profile (create the label) that allow to access the chosen objects.

A profile is composed by a level and the compartments that a user is supposed to have in order to do its job in a need-to-know basis. We design this by first: adding a new column called user_access_profile in the dms_user table; this column will store a string identifying the profile of a user (such as 'picker', 'loader', 'checker'); second: adding the security profile created for a user into the labels table with a flag indicating that the label is a clearance label, and therefore specific to users. This will allow to keep the relation of the security profiles with the corresponding labels.

It's important to refer that all of the new forms created to manage the labels should be accessible only to the administrator, due to obvious reasons. Otherwise, any user could granting himself/herself rights and then performing the action permitted.

The new tables created should also be protected against unauthorized modification, for the same reasons. This could be achieved using Secure application roles.

When a user logs into the application, the application context should determine the appropriate user profile to set the correct user label for the session.

As, we already have the identity of the actual end-user in the application context USERENV, it's easy to retrieve the user_access_profile for that user. Then, the procedure created to set the label for that session would only has to do an appropriate call to the function sa_session.set_access_profile provided by the OLS API.

```
execute sa_session.set_access_profile
(OLSPolicyName, V_user_profile);
```

From then on, the user clearance label is set for the session, logged in by the user.

With the profile development, the old procedure that would validate user privilege against menu/form privilege will be replaced by a new validation. Indeed, the OLS verification motor will use the user' session clearance to allow or deny the access for the user. The access to data is mediated based on four factors:

- Label of the row;

- Label of the user session;
- Policy privileges of the session;
- Policy enforcement options for the table.

Thus, in order to pass a security check, the user's label authorization must have a level equal to or greater than the level associated with the object *and* all the compartments associated with the data label. In other words, the compartments the user has must be a superset of the compartments associated with a sensitivity label.

Moreover, a variety of access control decisions can be made using the OLS built-in PL/SQL functions. These functions can be embedded inside new or existing program units.

When creating a new user, a labelling function to set user profile should be called using an internal trigger within the database. The profile assigned by default will be the more restrictive profile already created. Then, the form responsible by user management operation should be called to force the default profile to be updated.

6 Implementation Details

This chapter will focus on giving additional details for the implementation of the security model, as well as the issues that arose during the process, referring back to the issues discussed in the previous design section when pertinent.

6.1 Implementation

After the presentation which took place with experts from the ORWMS pool, it was decided to implement a prototype that would serve as a proof of concept for the solution proposed. Due to the limited time available, it was also decided to focus on the Vulnerability C, as it is considered to be the one with the greatest impact on the application's security, according to the questionnaires returned. Consequently, it is preferable to be the first one to be fixed.

Implement security in an application already developed might have a cost: it might create conflicts, causing working interfaces to fail; might degrade performance requirements, down-time requirements; might disrupt the operations, administration, and management of the system.

In some cases, a security control modification can have a negative impact on a product’s functionality and usability, or on other products or security controls.

The approach to the proof of concept was to create the solution using all the design decisions and, this way, prove that the design that was created really works in a real implementation.

As this solution has already been specified in detail in the previous chapter, special emphasis will be placed on major decisions taken.

6.1.1 Implementing Label Security

As stated in the previous chapter, there are five steps necessary to implement OLS, which will support the profile access control architecture proposed.

1. *Create the OLS policy.* The policy is the container for the labels, user authorizations, and protected database objects.

2. *Define the OLS label components.* The solution proposed uses levels and compartments to manage access to data on a "need to know" basis.

3. *Create the actual OLS labels to be used.* Valid labels for OLS are created, based on the application's security policy, taking the components defined in step 2.

4. *Apply the OLS security policy (labels) to the table(s).* This adds a label column to the table and adds the infrastructure required to support the row-level security based on the labels. This step also defines the security enforcement behaviour of the policy.

5. *Assign the user clearance label authorizations to be used by the users or applications.* This step determines who will ultimately gain access to what. Labels are assigned to "users," which may be a single user or a group of users. OLS mediates access to data by comparing the user's label with the label on the data record(s).

Decision 1: Label components

The label components are used to create data labels as well as to assign security clearances to database or application type users. The choice of labels' components is driven by the fine grained access control needed, and should therefore be adapted to the application's requirements. For the purpose desired, we choose to not use groups, as compartments are sufficient to enforce specifics users 'need-to-know'. To achieve this, the compartments defined should correspond to the authorizations required for a specific job or function.

For the organization, the first step consist of analysing the user population. It requires separating the users into one or more designated user types. This process may require the assistance of managers and security administrators. The components definition should use the information gathered during analysis. Thus, examples of compartments are 'Loader', 'Stock Control', 'Unit Picking'.

Decision 2: Hiding the OLS policy column

Most, if not all, applications weren't designed to work with sensitivity labels or row level security. While incorporating Oracle Label Security during design phase of an application is easiest, the implementation of such functionality in an application that is already in production can break its correct functioning.

When the policy is applied to an application table, a column with the name specified during policy creation is appended to the application table. Oracle Label Security has an option to hide the label column to provide transparency for existing applications. We opt to use that option to allow SQL statements which do not specify the Oracle Label Security column, or no column names at all, to continue functioning even though a new column has been added to the application table specified in the SQL statement.

Decision 3: Indexes

Considering that warehouses have several users performing the same jobs, and that various screens are needed to perform them, the percentage of the unique labels compared to the number of data rows protected will almost always be extremely low. Therefore, we opted by create a bitmap index on the label security column.

6.1.2 Implementing the Procedural Logic

Because Oracle stores the labels into the data dictionary, we chose to store the compartments, levels and labels made, into three separated table created with this purpose. This

was decided to made the management easier and to allow the visual presentation of these elements to the users. As some attributes, like the label tag used internally to mediate access or the levels numbers which determine the ranking of a label, have to be unique, this was enforced in the tables.

	POLICY_NAME	LABEL_NAME	LABEL_TAG	LABEL VALUE	USER_PROFILE
1	WMSCTRL	Voice Agency Picker	11	VOICE:AGPIKER	0
2	WMSCTRL	RF User Driver	21	RFUSER:FLTDRV	1
3	WMSCTRL	RF User Loader	22	RFUSER:LOADER	1

Figure 6.1: Labels table

The creation of labels is handled by a labelling function which contains procedural logic to compute the labels and then, insert them into the labels table. The labelling function have return type of the LBACSYS.LBAC_LABEL datatype. The LBACSYS role must have EXECUTE privilege on the labelling function.

Another function handles the task of inserting the labels to each row of the protected dms_menu table. This is made using simple update statements, and the char_to_label function provided with the OLS API.

	OPTION_TITLE	MENU_TITLE	OPTION_COMMAND	DOS_COMMAND	OLSLabel
8	BUILD TOTE PALLET	INVENT MGMT MENU	hh_build_tote_pallet_s	hh_build_tote_pallet_s	ORAACCESS
9	INVENT MGMT	MAIN MENU	RF_INVENT_MENU	RF_INVENT_MENU	VOICE:AGPIKER
10	MORE OPTIONS	INVENT MGMT MENU	RF_INVENT_MORE_MENU	RF_INVENT_MORE_MENU	VOICE:AGPIKER
11	CONFIRM RETURN	INVENTORY(MORE)	hh_confirm_return_s	hh_confirm_return_s	RFSPVR
12	FPL MGMT	INVENTORY(MORE)	hh_move_fpl_inv_s	hh_move_fpl_inv_s	INVMGMT
13	INV CONT TROUBLE	INVENTORY(MORE)	hh_inv_cont_trouble_s	hh_inv_cont_trouble_s	VOICE:AGPIKER
14	MANUAL REPLEN	INVENTORY(MORE)	hh_move_unit_inv_s	hh_move_unit_inv_s	VOICE:AGPIKER

Figure 6.2: dms_menu table with the OLS label colum

The user's authorization profiles are also stored into the labels table, but they cannot be used to label the rows of the dms_menu table. They are specific to the users. This is indicated by the value of the column user_profile that can be either 0 or 1.(Figure 6.1) The value 1 indicates that the label is a user clearance. The value 0 indicates that the label is a data label. Thus, the relation between the profiles names with the appropriate authorization label are kept.

When assigning a user profile to a user, the security profile name is stored for the corresponding user into a new column on the dms_user table. This column is a foreign key to the labels table, ensuring that the data is consistent.

	USER_ID	USER_ACCESS_PROFILE
1	PAR3214	RF User Driver
2	RDMUSR	
3	PAR1111	RF User Loader

Figure 6.3: columns user_id and
user_access_profile from the dms_user table

Then the application enables the appropriate authorization to the appropriate database sessions, according to the end-user identifier stored into the application context.

When creating a new user, we chose to embedded a call in INSERT trigger to the function that compute a restrict default label, thus forcing the posterior update of the user profile to be in adequacy with its need-to-know.

6.1.3 Implementing the Forms

The development of forms had to be done in the virtual machine setted up with the ORWMS. Unfortunately, due to time constraints and some difficulties found with the forms technology, this part of the implementation could not be completed.

In fact, some of the time was spent in the correct installation and configuration of Oracle Label Security, which is not installed by default with the Oracle Enterprise Edition.

About the difficulties found, a major problem was the strange behaviour of the ORWMS forms' compilation. Indeed, some forms compiled properly using the Oracle Form Builder application on Windows, but they compiled with errors when compiling on Unix environment.

The forms development implies the creation of three new forms and one menu entry, and the modification of some other existent forms.

The 'naut_gui_library' and 'dc_view_lib' libraries are referenced by almost all forms and are responsible by several pertinent objects related with security. Therefore some changes has to be made in some of their specifics program units. The major changes will be related with the inclusion of the dominates function from the OLS API. This function enables a program module to compare two labels and determine whether one label dominates another label. Thus, the program module should determine whether a user can access a form or menu entry by comparing a user's active session label with the menu/form fixed label. The function returns 1 if the first label provided dominates the second one.

6.2 Testing Process

Tests were made not only for the data classification, but also for the functions responsible for getting the correct end user identifier, as well as the formation of labels.

To prove the correct functioning of the solution, we first started by create a security policy with some data labels. The tests were made according with a general access control matrix provided by a Wipro Retail collaborator, and based on the reality of a major retailer. The compartments were defined accordingly with what was stated earlier.

Next, the policy was applied on the dms_menu and some policy labels was added to a few table rows. We then created some user authorization labels to be associated with the two application users existent in the database. We took care to define labels able to allow users to access to some rows of the table, and leaving other rows out of reach.

Then, simulating the authentication of each user and the process of setting the correspondent user profile, we did a SELECT operation on the dms_menu table. We could then verify that the rows returned were filtered according to the user profile.

6.3 Performance Considerations

Performance is important to all applications. Adding new functionality to existing applications requires due diligence up front to minimize the performance impact. Oracle Label Security provides row level security, basically turning on a security check at each row prior to allowing access. OLS will add a delay during login authentication to initialize additional security contexts in Oracle memory. Moreover, the same delay will be encountered when calling the set_access_profile function. The amount of delay will vary depending on the number of label components defined. Runtime performance overhead will depend on a variety of factors including:

- Number of tables protected by Label Security;
- Label Security enforcement options used;
- Complexity of existing application SQL logic.

With only one table protected by label security, no specifics enforcement options, and only SELECT operations performed against the protected table, the runtime performance of ORWMS isn't considerably affected.

7 Conclusions and Future Work

This chapter serves as a conclusion to the overall subject covered throughout this report and delineates the expected further developments to this project. It also presents an evaluation of the results achieved, and some personal considerations from the author.

7.1 Conclusions

Nowadays, data constitutes the most valuable assets of organizations. Therefore the unauthorized exploitation of this data can lead to several problems. For this reason, the databases that hold such data as well as the applications by which the data are accessed must be secured. Yet, security is not always considered when developing systems.

In this project, we have explored how to enhance the security of the Oracle Retail Warehouse Management System application. The Oracle Retail Warehouse Management System is an information system used in warehouses and distribution center, to maximize the movement of merchandise and information throughout the distribution process. As an Oracle product, this application offers several functionalities along with robustness and performance, but fails in terms of security from an internal point of view. Indeed, the security mechanisms implemented in the application are well behind those offered by the other applications of the Oracle Retail suite.

In fact, we analysed the aspects of a good security solution, and we noticed that the security mechanisms provided – password expiration and access control based on privileges levels – don't ensure essential security tenets like the 'defense in depth' and 'least privileges' principles.

Moreover, in order to develop trustworthy information systems, security aspects should be considered from the early project stages. This is particularly true for authorization and access control services, which decide which users can access which parts of the system and in what ways. Security cannot be addressed in a consistent way when being integrated into the system after the system has been built. Despite of that, we took this challenge.

The main goal of this project was, therefore, to design a security model which could enhance the overall application security but still continuing to function as intended. Given the scope of the coverage area, we assumed some system components, such as the network and the operating system, as being secured, and we focused on database and application's security with the objective of providing an internal security model. In this process, we first started to study

and analyse the application to identify the major vulnerabilities, at both database and application levels, that could emerged from its correct or incorrect use.

Several solutions for each vulnerability were analysed to enforce security not only within the database but also at the application level. Since it is impossible to design the perfect security solution at each layer, vulnerabilities are mitigated by the strengths at other layers.

Furthermore, because determining how to protected is based on what we are trying to protect, the operational environment as well as the nature and sensitivity of the stored data were taken into consideration for the design of the final solution. Performance and usability were also considered.

All solutions had been put together to create a multi-layer architecture, thus being in line with the defense in depth principle and the least privileges principle by the design of an effective access control based on a 'need-to-know' basis.

7.2 Evaluation of Results

The main goal of this project was to identify the ORWMS security-related vulnerabilities and propose a security model for this application. Both activities were successfully achieved: we detected and analysed the major vulnerabilities, and designed a set of solutions subsequently integrated into a security model. Nevertheless, as security represents a complex field and no system is unbreakable, it's hard to evaluate results. We can affirm, however, that the proposed model enhances the security system as a whole. In fact, the new model includes several layers and mechanisms able to provide effective countermeasures to the vulnerabilities identified. Therefore, “if all security objectives are achieved then the security problem is solved”.

As a second goal, the implementation of a prototype for one of the components of the model would serve as a proof-of-concept for the solution. The logic part of the prototype was implemented and tested, proving its adequacy to the problem. However, due to some difficulties and time constraints, the prototype was not fully completed. Even so, as the main objective was fulfilled, we consider that the whole project was a success.

7.3 Future Work

As regards to future work in this project, and considering that the internship continues almost two more months, the implementation of the security model should be continued.

Furthermore, the inclusion of a real stakeholder in the project, in order to get more requirements and useful information, would be a good thing.

Moreover, the audit and the security password enforcement enhancements, not included in the final model, may be further analysed to judge their real usefulness and eventual inclusion in the model.

7.4 Personal Considerations

This project proved to be very challenging and enriching, allowing to gain knowledge about retail, and specially about database and application security. In this context, this project enabled a familiarization with the Warehouse Management System module of the Oracle Retail suite, but from a security point of view, and therefore as not usually seen by typical users.

The internship at Wipro Retail proved to be an extremely enriching experience. The only regret was to not having been integrated on a team working on a external project, requested by a customer, which would allow to obtain more experience on the “field”.

Being an internal project, the major stakeholders didn't participated in the project, contrary to what is recommended by the best software engineer practices. The contact with the main stakeholders would been beneficial for the project, as well as for the personal experience of the author.

To finish, Oracle offers some of the best developed products and its commitment to security has remained constant. For this reason, we believe that Oracle will continue to refine its Warehouse Management solution to include new security technology and capabilities, making the product security easier to use and eliminate product vulnerabilities in future product releases.

Bibliography

[Analyst08] Analyst Perspectives. Database Security: An Area of Growing Concern. Consensus Report. Books24x7, October 2008.

[AppSec05] Application Security, Inc. Protecting the Crown Jewels - An Enterprise-Class Approach to Application-Level Security. White paper. 2005

[AppSec07] Application Security, Inc. Adressing the insider threat. White paper. 2007.

[AppSec08] Application Security, Inc. Secure Your World: Defending Against New and Emerging Databse Threats for 2008. Security note. 2008.

[AppSec09] Application Security, Inc. Protecting Oracle Databases. White paper. 2009.

[ASoft07] Absolute Software. Research Concepts Survey: IT Departments and Data Breach Prevention. 2007

[Behn07] Susan Behn. Application Security – What are my options?. Solution Beacon, LLC, 2007.

[CC06] Common Criteria. Common Criteria for Information Technology Security Evaluation – Part I: Introduction and general model. Version 3.1. September 2006.

[Cware06] CloakWare, Inc. Introduction to Application Security. White paper. 2006

[Britann09] Encyclopædia Britannica, Inc. Information System. Encyclopedia Britannica Online. 2009 http://www.britannica.com/EBchecked/topic/287895/information-system (last accessed on 12th May 2009)

[CULS07] Cornell University Law School. Title 44 of the United States Code. January 2007.

[DISA07] DISA. Database – Security Technical Implementation Guide. Version 8, Release 1. September 2007.

[GSDKS05] Rick Greenwald, Robert Stackowiak, Gary Dodge, David Klein, Ben Shapiro, Christopher G. Chelliah. Professional Oracle Programming. Pages 102-127. Wiley Publishing, Inc, 2005.

[Hill03] John M. Hill. Logistics Execution Systems Perspective. White paper. ESYNC.

[IBM09] IBM Corporation. Understanding Web application security challenges. White paper. January 2008.

[Kapur05] Samir Kapuria. Secure Application Development. White paper: Enterprise security. Symantec, 2005

[Knox04] David Knox. Effective Oracle Database 10g Security by Design. Oracle Press, 2004.

[Levin05] Matthew Levine. The Importance of Application Security. White paper: Enterprise security. Symantec, 2005

[Mande06] Mark Mandeville. Oracle Warehouse Management Overview. May, 2006. http://atloaug.org/presentations/OAUGOracleWMSMarkMandeville200605.ppt

[Natan05] Ron Ben Natan. Implementing Database Security and Auditing. Elsevier Digital Press, 2005

[Natan09] Ron Ben Natan. How To Secure and Audit Oracle 10g and 11g. CRC Press, 2009.

[Oracle02] Oracle Corporation. Oracle Label Security Administrator's Guide, Release 2. 2002. http://download.oracle.com/docs/cd/B10500_01/network.920/a96578/toc.htm (last accessed on 27th June 2009).

[NW05] Paul Needham, Peter Wahl. Data Classification With Oracle Label Security. An Oracle Database 10g Release 2 White Paper, June 2005.

[Oracle09] Oracle Corporation. Oracle Website - Oracle Warehouse Management System. http://www.oracle.com/applications/retail/sc/warehouse.html (last accessed on 29th May 2009).

[OraMSR] Oracle Corporation brochures, Managing at the Speed of Reality. http://www.oracle.com/applications/retail/library/brochures/managing-at-the-speed-of-reality.pdf (last accessed on 28th May 2009).

[OraRWM] Oracle Corporation Data Sheet, Oracle Retail Warehouse Management. http://www.oracle.com/applications/retail/library/data-sheets/warehouse-mgt.pdf (last accessed on 28th May 2009).

[OraSSA] Oracle Corporation. John Heimann. Oracle Software Security Assurance. An Oracle White Paper. January 2007.

[OraWMa] Oracle Corporation. Oracle Warehouse Management Implementation Guide, Release 11i., 2002

[OraWMb] Oracle Corporation. Oracle Retail Warehouse Management System Installation Guide, Release 13.0.1, August 2008

[OraWMc] Oracle Corporation. Oracle Retail Warehouse Management System User's Guide – User Interface, Release 13.0.1, June 2008

[OraWMd] Oracle Corporation. Oracle Retail Warehouse Management Radio Frequency User Guide, Release 13.0.1, June 2008

[OWASP] The Open Web Application Security Project. The OWAPS Home Page. http://www.owasp.org/index.php/Main_Page (last accessed on 26th May 2009).

[Pavone07] John Pavone. White paper: Application Architecture as a Catalyst to Securing Applications. Aspect Security, Inc, October 2007.

[PK03] Matt Piermarini, David C. Knox. Leveraging Oracle Database Security with J2EE Container Managed Persistence. An Oracle Whitepaper. October 2003.

[Peltier04] Thomas R. Peltier. Information Security Fundamentals. New York: Auerbach Publications, 2004.

[Ramac02] Jay Ramachandran. Designing Security Architecture Solutions. Wiley Computer Publishing, 2002

[Sentrig07] Sentrigo, Inc. Practical Guide to Database Security & Compliance. Whitepaper, version 1.1. November 2007.

[SHF04] Gary Stoneburner, Clark Hayden, Alexis Feringa. Engineering Principles for Information Technology Security (A Baseline for Achieving Security). NIST Special Publication 800-27 Rev A. June 2004.

[SI07] Josh Shaul and Aaron Ingram. Practical Oracle Security, your unauthorized guide to relational database security. Elsevier Digital Press, 2007.

[TB06] Third Brigade, Inc. Web Application Security: The Overlooked Vulnerabilities - White paper. 2006.

[TH98] Marlene Theriault, William Heney. Oracle Security. O'Reilly, First Ed., 1998.

[Walt01] Charl van der Walt. Introduction to Security Policies. http://www.securityfocus.com/infocus/1473 (last accessed on 26th May 2009)

[Wied09] Blake Wiedman. Database Security – Common-sense Principles. 2009.

[Wipro09] Wipro Retail. Wipro Retail corporation Website. http://www.wipro.com/retail/retail/ (last accessed on 11st June 2009)

[WRUS08] Wipro Retail US-presentation. Internal Document, Wipro, 2008.

Appendix A: Questionnaire

Within a project that aims to achieve improvements in the security mechanisms of ORWMS, some deficiencies, for which we want to design and implement a set of solutions, were identified and are described below.

In order to identify additional vulnerabilities, we would like to obtain your cooperation to answer a short questionnaire. We appreciate any further comments you wish to make regarding the vulnerabilities already identified.

Your opinion will be a great help. Thank you.

Data security

► User's credentials and access level attributes are sensitive data, since the unauthorized modification of this information may result in modification or denial of user's privileges and access levels.

Problems identified:

Visualization of data within the *user_password* column when accessing the *dms_user* table; Possibility of changing the user's authentication credentials

Visualization of data within the *user_privilege* column when accessing the *dms_user* table; Possibility of changing the user's access levels.

► Database access control is the fundamental mechanism for data security, therefore account sharing between groups of users is a bad practice.

Problems identified:

All the application users are sharing the same DB account, and consequently the same privileges.

► The shared schema design should allow to maintain different objects and system privileges.

Problems identified:

Any privileges granted directly to the schema are available to all users that are mapped to that schema.

It should be possible for two users with different roles, actually have the different roles when they log in to the DB, according to the tasks needed to do their jobs.

Application security

► One tenet of efficient security is the "Least privileges" principle. The application should display to the operators the menus they need to do their job and nothing more.

Problems identified:

The current authorization architecture, based on the 8 privileges levels, allows an operator to visualize/use menus with lower privileges and whose operations are not necessary for its job. Therefore, the least privileges principle is violated

► The user is connected to the application, and the application is connected to the database. Therefore, the application should connect each application user to a distinct database account.

Problems identified:

This design could require the application to know the user's database password. For security reasons, the application should not be allowed to acquire the user's password.

Wipro Portugal S.A.
Rua Engº Frederico Ulrich, 2650 Edifício Wipro 4470-605 Moreira Maia Portugal Phone: + 351 226 077 500 Fax: 351 22 607 74 01
Regd. Office: **Wipro Limited**, DoddaKannelli, Sarjapur Road, Bangalore-560 035, India. Tel: 91-80-28440011, Fax: 91-80-28440054

QUESTIONNAIRE

1. Considering the set of vulnerabilities described above, please classify each of them depending on the degree of importance and impact regarding security.
Choose (highlight) the most adequate option (1- Minor important/impact, 4- Major important/impact)

A. Unauthorized view/modification of user's credentials and access level attributes	**1**	**2**	**3**	**4**
B. Account and privileges sharing between application users	**1**	**2**	**3**	**4**
C. Privileges granted directly to the schema are available to all users that are mapped to that schema.	**1**	**2**	**3**	**4**
D. Users can visualize/use application menus with lower privileges than theirs but whose operations are not necessary for their job	**1**	**2**	**3**	**4**

2. Consider now a possible solution to protect user authentication and access level data.
Taking as example: SELECT user_id, user_name, user_password FROM dms_user; do you think it is more appropriate to:

___ A. Restrict access to (hide) records containing information not related with the author's query.
___ B. Masking the user_password column;
___ C. Do password Hashing + encryptation;

3. Does it make sense to strengthen password security mechanisms, when considering the enterprises' policies normally used, especially as regards the following points:
Choose (highlight) the most adequate option (1-Disagree, 2-Neutral, 3-Agree, 4-Strongly Agree)

a). Do not allow the reuse of a password previously used (password history)?	**1**	**2**	**3**	**4**
b). Force the password length to be greater than some specified length, as well as the mandatory use of a mixed of letters and digits?	**1**	**2**	**3**	**4**
c). Force the password to not be the same as the username?	**1**	**2**	**3**	**4**

Wipro Portugal S.A.
Rua Engº Frederico Ulrich, 2650 Edifício Wipro 4470-605 Moreira Maia Portugal Phone: + 351 226 077 500 Fax: 351 22 607 74 01
Regd. Office: **Wipro Limited**, DoddaKannelli, Sarjapur Road, Bangalore-560 035, India. Tel: 91-80-28440011, Fax: 91-80-28440054

4. Application users should have access to menus on a 'need-to-know' basis. The 8 levels authorization architecture doesn't resolve this security need. So, how should the users be allowed to access menu options?

Choose (highlight) the most adequate option (1- Not adequate, 2-Poorly adequate, 3-Adequate, 4-Strongly adequate)

A. Depending on the sensitivity level of information that can be accessed through the menu; **1 2 3 4**

B. Depending on the access category **1 2 3 4**

C. Depending on User groups **1 2 3 4**

D. A mix of the 3 proposals presented above **1 2 3 4**

5. Although users' categories vary in each warehouse (receiver, picker, manager, etc), the definition of profiles for accessing information can be done by means of hierarchical categories and/or function area. What are the most common categories and user roles you have seen in warehouses?

6. Do you think is important to filter the access to information depending on the category versus merchandise hierarchy? For instance some receivers can only look for items within a department/class or subclass?

7. Is it justified to disable certain privileges depending on how the user access the data (through the application or directly from the BD)?

8. Do you think it is necessary to audit the various operations made on the DB? If so, what kind and for what tables?

Please, feel free to write additional comments:

Wipro Portugal S.A.
Rua Engº Frederico Ulrich, 2650 Edifício Wipro 4470-605 Moreira Maia Portugal Phone: + 351 226 077 500 Fax: 351 22 607 74 01
Regd. Office: **Wipro Limited**, DoddaKannelli, Sarjapur Road, Bangalore-560 035, India. Tel: 91-80-28440011, Fax: 91-80-28440054