IT auditing of oracle application server

Zuidweg, N.J.

Award date:
2008

TECHNISCHE UNIVERSITEIT EINDHOVEN
Department of Mathematics and Computer Science

IT Auditing of Oracle Application Server

By
N.J. Zuidweg

Supervisors:

Bart Jacobs (TU/e)
Ton Stevenhagen (EAP)

Eindhoven, February 2008

Acknowledgement

I want to thank my colleagues at EAP for their unremitting support. I especially want to thank Ton Stevenhagen since he has guided me through this project as internal graduation tutor. Also I am grateful to Frank Blom for this practical advices and comments on my work. Last but not least I want to thank my gradation supervisor at the TU/e, Bart Jacobs, who stimulated me to work on an academic level and motivated me to complete this project.

Summary

IT auditing originally was closely related to financial auditing: IT auditors did support financial auditors in evaluating information systems. Nowadays, IT auditors are asked to audit various IT (related) systems and aspects (and to advise about these).

Oracle Application Server (OAS) is a collection of software components (running on hardware components) that makes information (from a database) accessible via the Web. For this purpose, a web server is used to provide either websites or Web Services. The other components facilitate for instance in storing configuration data and administrating users' account information and credentials.

For IT auditors, auditing OAS is quite a difficult job because of its large number of interconnecting components. Normally, IT auditors use audit objectives to which a system should comply. These audit objectives are often derived from general audit objectives like BS7799 and CobiT. To check if a system complies to a set of audit objectives, a so called audit program is used to check all important aspects step by step.

In order to be able to compose concrete audit objectives (and a concrete audit plan) for OAS, it is essential to obtain insight in both the general auditing objectives and in the working of the various components of this product. In this thesis, this analysis is concentrated on 'inherent vulnerabilities' of OAS. These are important since they are opposed to both common security knowledge and what is stated in several general audit objectives. By mitigating these vulnerabilities, a more 'reliable' OAS can be achieved.

Contents

Chapter 1

Introduction

This thesis is part of my final project for the special track Information Security Technology (IST) within the Computer Science and Engineering (CSE) master's program at the Eindhoven University of Technology (TU/e).

The project consists of the development and the implementation of an IT audit of the concept of Oracle Application Server (OAS) for EDP Audit Pool (EAP) of the Dutch Ministry of Finance. IT auditors from EAP are more and more faced with the use of middleware: software that provides program-to-program communication across multiple computers [Lea00]. Traditionally, IT audits have been concentrated on those parts of the software architecture where either user interaction takes place or where data is stored. As a result, middleware systems like OAS usually have been faced as a kind of 'black box'. This also holds for the IT audits that are performed by EAP: when performing a technical IT Audit, the applications running on OASs are examined and also the Oracle Database Systems are assessed. OASs, however, are often not evaluated.

In performing technical IT Audits, 'reliability' (in Dutch: *betrouwbaarheid*) is a key term. It is defined by [vBZ95] as 'the extent to which an organization can rely on an information systems for its information supply' (in Dutch: "*betrouwbaarheid is de mate waarin een organisatie zich kan verlaten op het informatiesysteem voor zijn informatievoorziening.*").

Nowadays, more and more customers of EAP are using OASs. OAS is a powerful tool that interconnects Oracle Databases with web servers and other related applications. Therefore, the 'reliability' of information systems that make use of an OAS is greatly influenced by the 'reliability' of this component. Hence, the auditors from EAP felt the need to investigate the concept of OAS with all its relations in order to be able to perform a technical audit of this IT system.

Because of this, the central research question for this project is: "*In which manner can the reliability of OAS be formulated and measured in practice?*" This research question is addressed from an IT auditors perspective so that the answer can be used by IT auditors to judge a particular instance of OAS. The final result is a set of audit objectives and an audit plan that is based on an examination of the concept of OAS with its relations.

In order to answer the central research question, first the general concept of 'the application server' is discussed in Chapter 2. Chapter 3 addresses the field of IT auditing: several key terms are explained and the EDP Audit Pool is introduced in more detail. Chapter 4 is specifically dedicated to OAS as it explains, per relevant component, the way in which this system works. Chapter 5 discusses several 'general audit objectives' that are used by IT auditors. Chapter 6 uses the information form Chapter 4 and Chapter 5 to identify vulnerabilities that are inherent to the

design and the implementation of OAS. This collection of 'inherent vulnerabilities' is used as a source for a collection of audit objectives (see Appendix A) and an audit program (see Appendix B) for auditing OAS.

It is intended that the results are applicable to any instance of Oracle Application Server 10g. Maybe the main results can even be used for other versions of this product, or even for application servers that have been developed by other vendors.

Chapter 2

Application Servers

The term 'application server' is not easy to define. However, on the Internet, several attempts can be found. An application server for instance is described as "a software platform that delivers content to the Web'[1]'. This is in fact correct, but not very specific. An application server is not only a collection of hardware and software, but it can very well be seen as a concept. To gain insight into the origin and the working of this concept, first a short overview of the development of software architectures is presented in this chapter:

In the early days of computing, all tasks were performed by a central computer: the mainframe. As depicted in Figure 2.1, users could only interact with this mainframe via dumb terminals that displayed the information as prepared by the mainframe.

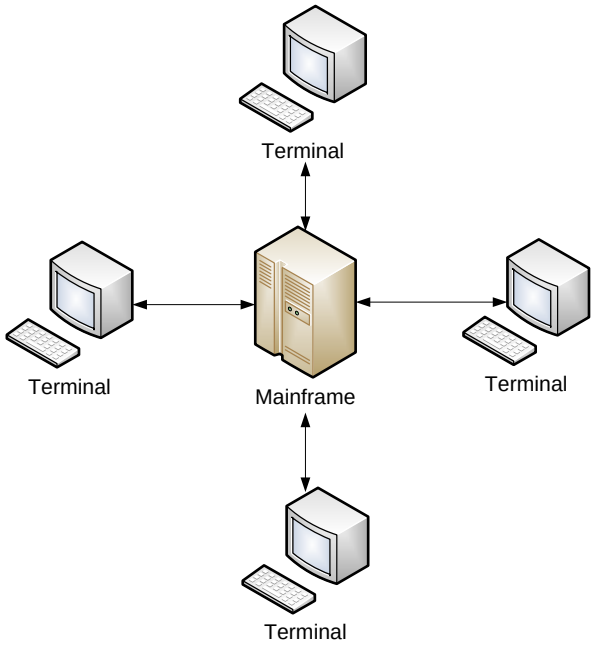

Figure 2.1: One tier: mainframe and terminals

[1] See http://tinyurl.com/2v689j, accessed on 8/2/2008

Once PCs became available, the mainframe often[2] was replaced by a file server. In this way, users could transfer files to their own PC to obtain the information they wanted. By implementing a database management system (DBMS), users could directly request specific information. Using this setting, network traffic could be curtailed since the information that was wanted was extracted by the server. The resulting architecture consisted of two so called 'tiers': the server with a DBSM was one tier, the clients formed the second one (see Figure 2.2).

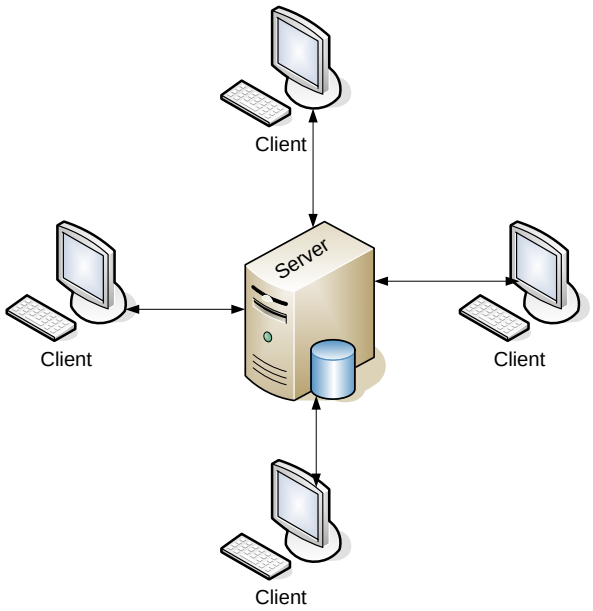

Figure 2.2: Two tiers: server and clients

However, as the number of clients grew, performance began to deteriorate. Furthermore, the system as a whole was bound to a particular DBMS, without having the flexibility of switching to another one easily. Moreover, programmers were faced with problems when they had to guarantee the consistency of the database and the confidentiality of the information in it. An extra 'tier' was needed to overcome these problems. Therefore, the concept of an application server was developed. Acting as an intermediate between clients and one or more database servers, the application server could increase performance by means of caching data. Furthermore, by moving business logic from the clients to the application server, programmers got an easier job in making the system secure. Switching to another DMBS also got easier as a result of having all business logic in one place and using standardized ways to communicate with the DBMS. So, the limitations of the two-tier architecture caused a demand for an extra tier performing various tasks. The resulting three-tier architecture is depicted in Figure 2.3.

ColdFusion, Oracle Application Server, and IBM WebSphere Application Server are well-known examples of commercial application servers. According to Wikipedia[3], at least 30 application servers have been developed. An essential element that all these application servers have in common is the usage of the Web as an access channel. The (Word Wide) Web is a concept that has been proposed by [BLC90]. It is based on HyperText which provides a way to represent textual data in a structured and interlinked manner. In this context the HyperText Transfer Protocol

[2] Mainframes still are used as large data processing systems to perform computationally intensive tasks.

[3] See http://tinyurl.com/2hbxbg, accessed on 27/3/2007.

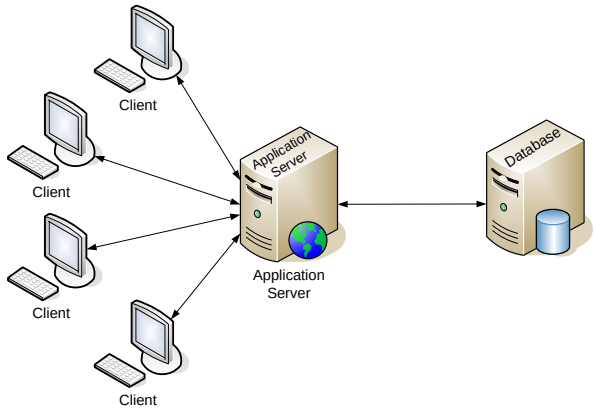

Figure 2.3: Three tiers: client, application server, and database server

(HTTP) plays an essential role as a protocol for transferring HyperText documents. The same holds for the HyperText Markup Language (HTML) that is used to represent HyperText and its markup. HTML is used to compose websites that can be displayed in a browser. HTML is closely related to the eXtensible Markup Language (XML) that is focused on structured content representation only, without describing its layout. The structure of such an XML document can also be described in XML by an so called XML schema. As an alternative to this, it is also possible to use Document Type Definitions (DTDs). These XML schemas or DTDs can be used to automatically check the 'validity' of an XML document. An XML document is 'valid' if it complies with the structure that is prescribed in the DTD or XML schema that is used.

XML documents play an important role in Application Servers to exchange messages. An Application Server is said to offer a Web Service when it allows communication by receiving and sending XML documents. W3C defines a Web Service as "a software application identified by a URI, whose interfaces and binding are capable of being defined, described and discovered by XML artifacts and supports direct interactions with other software applications using XML based messages via Internet-based protocols"[4]. An URI (Uniform Resource Identifier) is a character string used to identify a resource. The XML communication with the Web Service takes places via the Simple Object Access Protocol (SOAP). The name suggests that SOAP is a communication 'protocol', but in fact it is more an 'envelope' (with addressing information) that can contain XML messages. SOAP messages can be transferred via various protocols, but in practice almost always HTTP is used.

The Web Services Description Language (WSDL) specifies the interface of a Web Service. By means of WSDL insight can be gained into the kind of service that is offered. It also shows the manner in which the service can be called (for instance: which parameters are required). The WSDL descriptions of Web Services can be stored as XML document in a directory to enable the discovering of Web Services. For this reason, the concept of a 'Universal Description, Discovery, and Integration' (UDDI) registry has been developed. UDDI provides the opportunity of searching and accessing WSDL documents. The general outline of this whole concept of collaboration of protocols is depicted in Figure 2.4. In practice, web *sites* are generally used to present information to end users (clients) whereas web *services* are used to exchange information between application servers or with dedicated programs that are used by clients (see Figure 2.5).

[4] See http://tinyurl.com/ysx8kw, accessed on 2/5/2007.

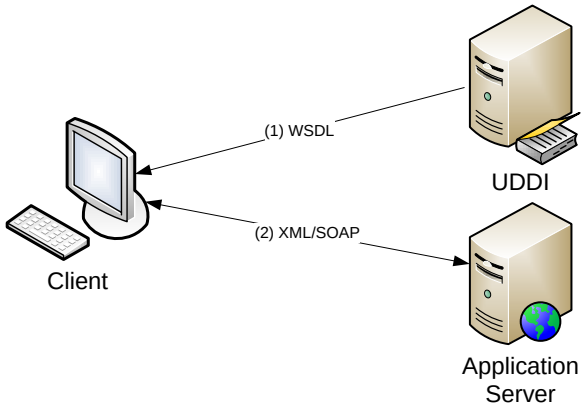

Figure 2.4: Basic Web Service client interaction

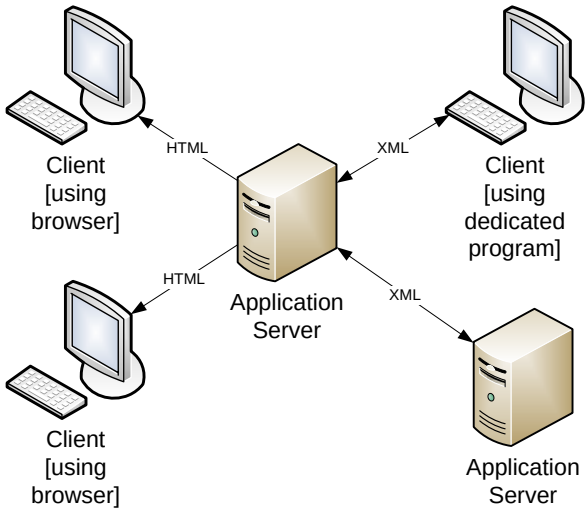

Figure 2.5: An application server offering a website (using HTML) and a Web Service (using XML)

Chapter 3

IT auditing

IT auditing was formerly called EDP auditing, with EDP standing for Electronic Data Processing. As the term EDP passed into disuse, the expression EDP auditing was replaced by the term IT auditing. Initially, doing IT audits was part of the job of a financial auditor[1], a person who is authorized to certify the balance and income sheets. The financial auditor encountered computerized parts of the accounting department when performing a certification. As the complexity of these automated parts grew, it became more and more challenging for the financial auditors to assess these systems. Therefore, some financial auditors began to specialize in computerized financial systems. In 1967 a group of these specialized auditors joined in what they called the Information Systems and Control Association[2] (ISACA).

In the Netherlands, IT auditors joined in 1992 in the '*Nederlandse Orde van Register EDP-auditors*'[3] (NOREA). To become a member of NOREA, one needs to accomplish a postgraduate training at the 'Amsterdam Graduate Business School' (from the *Universiteit van Amsterdam*), the '*Erasmus Universiteit Rotterdam*', the '*Tias Business School*' (from *Tilburg University*) or the *'Vrije Universiteit Amsterdam*'. The exact contents of the training differ per university; however, they all do focus on aspects like business proces management, managerial information supply, risk management, project management, (auditing) standards, and information security. An admission committee decides if candidates are allowed to participate in the training. The training is of part-time nature and its official duration varies between two and three years. Financial auditors and operational auditors can complete the IT auditor's training in a shorter amount of time. In addition to the training, at least three years of experience in the field of IT auditing are required to become a member of NOREA. Only members of NOREA are allowed to use the Dutch title '*Register EDP-auditor*' (RE). ISACA has its own 'Certified Information Systems Auditor' (CISA) title that may be used after successfully completing a CISA exam.

IT Auditors can be employed within an internal audit department of an organization. Such auditors are called internal auditors. External auditors are generally employed at an auditing organization (such as *Deloitte*, *Ernst & Young*, *KPMG*, or *PwC*). Often these external auditors are involved in the annual check of the balance and income sheets; however, it is also possible they are brought in by the internal audit department or by management.

[1] In The Netherlands: *Register Accountant* (RA), in Great Britain: Chartered Auditor (CA), in the US: Certified Public Accountant (CPA).

[2] See http://www.isaca.org

[3] See http://www.norea.nl

The (Dutch) government also has employed IT auditors: in fact, each ministry within the Dutch government has its own audit office. Additionally, the EDP Audit Pool (see Section 3.4) provides IT audit services to all departments within the Dutch government.

Nowadays the job responsibilities of an IT auditor go far beyond the original task of supporting the financial auditor. The expansion of tasks started when, in addition the the financial auditors, also management began to ask IT auditors to perform risk-based assessments of IT systems. Later IT auditors were also asked to audit other IT related aspects (and to advise about these). [NOR98] distinguishes between six different IT related objects that can be assessed as part of an IT audit:

The information strategy contains the general ideas that an organization has about the way information should be handled. The evaluation of the information strategy of an organization is a good starting point for an IT audit since it can contain useful information about the organization's information standards from which concrete criteria can be deduced.

Information (technology) management is already more concrete as it relates to the general plan of managing information within an organization. This plan contains decisions about what to automate, where to store what and, for instance, how to structure the network.

Information systems do support one or more business processes by facilitating data processing. It is for instance possible to have a dedicated information system to support the purchasing process, but it is also possible to have an integrated information system that supports a number of business processes.

Cybernetic systems also support business processes; not by facilitating data processing, but only by controlling specific devices (for instance robots or machines in a production process).

IT systems, that are also called **technical systems**, are the application components of an information system or process system. OAS is an example of an IT system and so are operating systems, compilers, access control systems and database management systems. An audit of an IT system is called a technical audit.

Operational IT support contains all activities that are needed to manage and support the complete IT infrastructure within an organization. The help desk is for instance part of the operational IT support.

Since this project concerns IT auditing of OAS, it can be identified as the setup and the implementation of a technical audit of the concept of an IT system. Although some issues related to operational support and information systems will be addressed, the main focus will be on OAS. An Oracle Application Server is an *IT system*. For this reason, the rest of this chapter is directed towards IT systems in general and OAS in particular.

3.1 Audit objectives and audit programs

In auditing IT systems, audit objectives (in Dutch: '*normen(kader(s)*')) play a central role. Audit objectives are criteria that are used to evaluate IT systems. These criteria are often derived from standards like the ISO/IEC 17799 'Code of Practice for Information Security Management' (See Chapter 5 for more on these standards.). The audit objectives are usually determined in consultation with the mandator of the IT audit. When a financial auditor asks to perform an IT audit, the

central audit objective often will be that 'the financial systems within an organization will lead to reliable data processing', meaning that the financial information from the computer systems can be used as evidence to support the annual certification of the balance and income sheets. Within the Dutch government, all departments have to comply with the '*Voorschrift Informatiebeveiliging Rijksdienst (VIR)*', a regulation that provides the outline for requirements on information security security measures within the Dutch government. All these audit objectives are formulated on a high level. As a result, more concrete criteria are needed to perform an IT audit in practice.

These concrete criteria are formulated in a so called 'audit program' (in Dutch: *controle (werk)programma*). An audit program is a document in which different requirements are stated. One ore more actions are needed to see if a requirement is met. The requirements that are stated in this program are of the kind that they can be checked off or that 'simple' remarks on them can be made. An audit program can for instance request for the review of log files or demand to ask for handbooks or procedures to see if they are on hand. Also it may for example ask to check certain configuration parameters.

3.2 Judgements

Using the criteria from the audit program, the IT auditor can state a judgement that is based on the audit objectives that were used. Interviews with users are a useful resource in judging an IT system. However, real evidence is needed to substantiate the judgement. Log files and screen dumps can for instance be part of this evidence. In spite of this evidence, it is practically impossible to give full guarantee that a system does comply with all the criteria. Similarly to the situation in which a financial auditor is certifying the balance and income sheets, it is possible that some important details have been overlooked. Moreover, it is in general not cost-effective to review all evidence so the IT auditor has to work with partial observations. Therefore, the judgement is always stated with a particular degree of certainty. In practice, almost all judgements are state with 'reasonable certainty' and in some seldom cases (where too little evidence has been seen) with 'restricted certainty'.

For the possible types of judgements, IT auditors use the same terms as the financial auditors. That means that a judgement can be approving, approving with conditions or denying. A judgement is always bounded in its context: it does only take into account the audit objectives and it only concerns the state of the judged system at the time that it is assessed. When an involved party has reason to believe that an IT auditor passed an erroneous judgement, the IT auditor can be demanded to testify to a disciplinary committee. In the Netherlands, the disciplinary committee of NOREA has heard three cases at the time of writing. All three judgements led to acquittal of the IT auditors who were involved.

3.3 Information criteria

When certifying the balance and income sheets, financial auditors are concerned with the question if these documents show a true impression (in Dutch: *een getrouw beeld*) of the actual financial situation of a particular organization. An IT auditor does need other criteria when passing a judgement. In addition to this, as financial auditing is more directed towards *compliance*, IT Auditing is mainly focussed on *risk-control* (in Dutch: *risicobeheersing*). In this context, it is important to identify a number of criteria that an IT system needs to satisfy in order to keep risk *in control*. These criteria then also can be used in auditing an IT system.

These criteria can be subdivided into a small number of meta-requirements that are called 'information criteria' (In Dutch literature: '*kwaliteitscriteria*' or '*kwaliteitsaspecten*'). These 'information criteria' can be used to express the 'reliability' of an IT system. As explained in Section 5.1, the 'reliability' of an information system is defined as 'the extent to which an organization can rely on an information systems for its information supply' [vBZ95]. EAP uses four different 'information criteria' to determine this 'reliability':

Confidentiality (in Dutch: *exclusiviteit*) "concerns the protection of sensitive information from unauthorized disclosure" [ITG07]. In [NOR98], NOREA provides a definition in the Dutch language which can be translated like this: "Confidentiality is the extent to which[4] only authorized persons or equipment via authorized procedures and under restricted powers make use of IT processes." This definition is not focussed on information, but on IT processes.

As an example, a confidentiality-related criterium from the audit program for OAS could be that no information from the database should leak to unauthorized parties. For IT auditors, evaluating the access control system then is an important step, but also examining the communication channels (is the communication encrypted) and seeking potential backdoors (for instance: is there a way to log on directly to the database for maintenance reasons?) may not be overlooked.

[NOR98] explains that for instance privacy, isolation of processes, identification, and authentication are aspects related to confidentiality.

Integrity (in Dutch: *integriteit*) "relates to the accuracy and completeness of information as well as to its validity in accordance with business values and expectations" [ITG07]. In [NOR98], NOREA provides a Dutch definition which can be translated like this: "Integrity is the extent to which the information contained in the IT system is in accordance with the depicted reality." Both definitions show the relation between IT auditing and financial auditing because financial auditors do have to check if the balance and income sheets do represent the actual (financial) situation of an organization. For a real IT system, the only aspect that really can be checked is if the system does process input, storage, and output of data while preventing corruption and unwanted alteration of it.

As an example of a concrete integrity-related criterium, one could think of the requirement that the data that is cached in the OAS has to be consistent with the data in the database. The IT Auditor first examines the way in which OAS handles this issue, when extra measures are necessary it is checked if they are indeed implemented.

There is some overlap between 'confidentiality' and 'integrity': when an unauthorized person can access confidential data so that changes in this data can be made, then the integrity of this data cannot be guaranteed.

[4] The notion of 'the extent to which' (in Dutch: '*de mate waarin*') in the NOREA-definition is added to express two things. In the first place, not all systems do require authorization of persons or equipment using the system. For instance a public website must be accessible by anyone, without the need for authorization. Secondly, 'confidentiality' relates to concrete criteria that are stated in the audit program. In practice it is (almost) always the case that the system does not fully comply with all these concrete criteria. When performing an IT audit, the auditor writes a report in which per criterium is stated to what extend it is met. Commonly it is also mentioned what the consequences are of not meeting a criterium. In this way, the term 'degree' does not include any explicit quantitative information on the confidentiality of a system, but it only indicates that most likely not all criteria are met. This second motivation for the notion of 'the extent to which' can also be applied to the other three criteria.

Availability (in Dutch: *beschikbaarheid*) "relates to information being available when required by the business process now and in the future. It also concerns the safeguarding of necessary resources and associated capabilities" [ITG07]. In [NOR98], NOREA provides a Dutch definition which can be translated like this: "Availability is the extent to which the system is continuously available and the extent to which the data processing can proceed in an undisturbed way."

As an example of a concrete availability-related criterium one can think of the requirement that the system must continue to work when a harddisk fails.

There is some overlap between 'confidentiality', 'integrity' and 'availability': when an unauthorized person for instance can erase data from a harddisk, then 'confidentiality', 'integrity', and 'availability' are possibly all involved.

Auditability (in Dutch: *controleerbaarheid*) is not exactly defined in [ITG07]. In [NOR98], however, NOREA provides a Dutch definition which can be translated like this: "Auditability is the extent to which it is possible (for an IT auditor) to acquire knowledge about the structure and the working of a system." In practice, checking the auditability of an IT system often boils down to the analysis of logging possibilities and the configuration of those.

These definitions (especially that of 'integrity') show that the focus is not only on security (is a system resistant to attacks and misuse), but also on function correctness (does an IT system do what is intended to do in the right way).

In fact, different text books use different selections of 'information criteria', also the definitions of the criteria are not used consistently and the same holds for the term 'information criteria' itself. For instance Section 5.1 shows that [vBZ95] uses only the first three of the 'information criteria'. Furthermore, also the definitions of the criteria (slightly) deviate from the definitions above.

Within the field of computer science the three first of the criteria as mentioned above are used as 'information security criteria', forming the well known acronym CIA. This is only one acronym and computer scientists do also take into account other aspects like non-repudiation and monitoring (which is related to 'auditability'). The CIA criteria are now defined from computer science perspective to clarify their relation with the IT auditors' definitions:

"Confidentiality involves an obligation to protect some other person's or organizations' secrets if you know them." [And01]. Anderson explains that "secrecy is a technical term that refers to the effect of the mechanisms used to limit the number of principals who can access information, such as cryptography or computer access controls" [And01]. The combination of these two definitions covers the IT auditor's definition of confidentiality quite well.

Integrity is defined by [And01] as a term that is closely related to *authenticity* which is academically defined as *integrity plus freshness*. Freshness ensures that communication is genuine, not a replay of previous messages. Integrity can be applied to messages, demanding that they are not altered. It can also be applied on stored data, demanding that it is not corrupted. With this definition, integrity does for instance not demand checking of input data. The IT auditor's definition is less concrete and (therefore) harder to measure. This computer science definition will suffice since this project is about IT auditing of the concept of OAS, and not of the applications running on OAS.

Availability is not exactly defined in [And01], but from the text it can be concluded that this simply concerns the fact that the system must be up and running. Fault tolerance and failure recovery are important terms related to this aspect. When analysing OAS for availability, the focus will be on the question under which conditions components of OAS will fail, what the consequences for other components can be, what measures are necessary to make the OAS more fault tolerant, and what can be done to improve OAS's recoverability.

A complete chapter in [And01] is dedicated to monitoring systems. Monitoring systems and 'auditability' are closely related since monitoring is an important aspect of auditability. For instance the network traffic or the changes to configuration files can be subjected to monitoring.

3.4 EDP Audit Pool

EAP is a directorate of the Dutch Ministry of Finance, providing IT audit services to all ministries within the Dutch government. In the eighties the pool was founded in 1987 when internal financial auditors of some departments were asking for more expertise to perform audits or IT systems. It then was decided to join the auditors' knowledge of IT auditing into an interdepartmental cooperation that was called EDP Audit Pool. First only a few departments took part in EAP. Nowadays, all departments of the Dutch government do participate in it.

Often EAP is commissioned by the internal auditors of the various departments to perform IT audits. However, sometimes they are asked by management of a department or by the 'The Netherlands Court of Audit' (in Dutch: '*Algemene Rekenkamer*') which audits the balance and income sheets of the Netherlands national government. Nowadays, EAP is often asked to issue 'third party statements'[5], which give information about the 'reliability' of external parties[6] like hosting providers.

Various departments have contributed employees to EAP. Per employee that has been brought in, a department has a claim on a certain amount of working hours from EAP. These hours can be used to assist the department's internal auditors, but also to perform an independent audit, or to advise about one of the IT auditing objects that are described in Section 3. Additionally, all departments can 'buy' working hours from EAP. These working hours are mainly used for supporting departmental auditing offices, but sometimes EAP also give advice (to management). Obviously, these advices only are given when EAP does not has to audit this (part of the) organization (in the future).

[5] At EAP, these third party statements are called '*Third Party Mededelingen*' or 'TPMs' in a mixture of Dutch and English).

[6] This auditing external parties is advocated in the 'Statement on Auditing Standards No. 70' (SAS 70) which is an auditing statement issued by the Auditing Standards Board of the American Institute of Certified Public Accountants (AICPA).

Chapter 4

Oracle Application Server

As Chapter 2 was about application servers in general, this section is dedicated to one particular instance: Oracle Application Server (OAS). First some historical notes have to be made to clarify the background of OAS:

The Oracle corporation was founded in 1977 under the name Software Development Laboratories. In 1979 its relational database technology was made commercially available. In that same year, its name was changed into Relational Software Inc. (RSI). In 1983 the company was renamed to Oracle because most of RSI's activities were concentrated on its database product with this name. Although Oracle was implementing some middleware as part of their database product, the company's first real application server product was not made generally available until 1998. It took until 2001 before a redesigned version of this product was released as OAS 9*i*. The *i* was included to emphasize its ability to operate as an Internet platform component. Version 10*g*, the version that is addressed in this report, was made available in 2003. The *g* (from grid) was added to show that this version of OAS was designed to utilize computer clusters, meaning that the computing power of different systems can be combined to perform certain jobs. OAS is now a central part of Oracle's Fusion Middleware program. This program consists off several different middleware components. Products like 'identity management' and 'business integration' are presented as separate software solutions. However, most of these products are actually part of releases of OAS.

At the time of writing (fall 2007), four different versions of Oracle Application Server are available[1].

Enterprise edition is the version that contains the most components including functionalities for identity management and wireless development.

Standard edition contains less functionality than the enterprise edition. Still, the most commonly used components like 'single sign-on' and OAS Portal are included.

Standard edition one contains the same components as the standard edition, except for the Oracle Internet Directory. According to Oracle, standard edition one enables a 'one-click installation' and it also is said to offer a web based management interface that is easy to use.

[1] More details can be found on a page on the Oracle website (see http://tinyurl.com/ywy66b, accessed on 3/5/2007).

Java edition is the most stripped version of OAS. It does not contain Oracle Web Cache or Oracle Internet Directory. This version is primarily aimed at developers to facilitate a light-weight environment for developing and testing enterprise Java applications.

At EAP, I have been working with the Enterprise edition of OAS 10*g* (version 10.1.2.0.2[2] running on SuSE Linux Enterprise Server 9. On a separate machine, with SuSE Linux Enterprise Server 9, Oracle Database 10*g* (version 10.2.0.1.0[3]) is running. Both OAS and Oracle Database are installed to gain insight into the way in which the components work that are described in this chapter.

The main components of the OAS Enterprise Editions' architecture are depicted in Figure 4.1. Notice that the components are subdivided into two distinct 'blocks'. The *mid-tier* block contains the components that offer the core application server functionalities, the components from the *infrastructure* block provide the more supplementary functionalities. In the following sections, each of these OAS's components is discussed briefly: Oracle has provided a large amount of documentation for OAS and all its components. This chapter gives only very brief summaries with a focus on what is relevant in the present context.

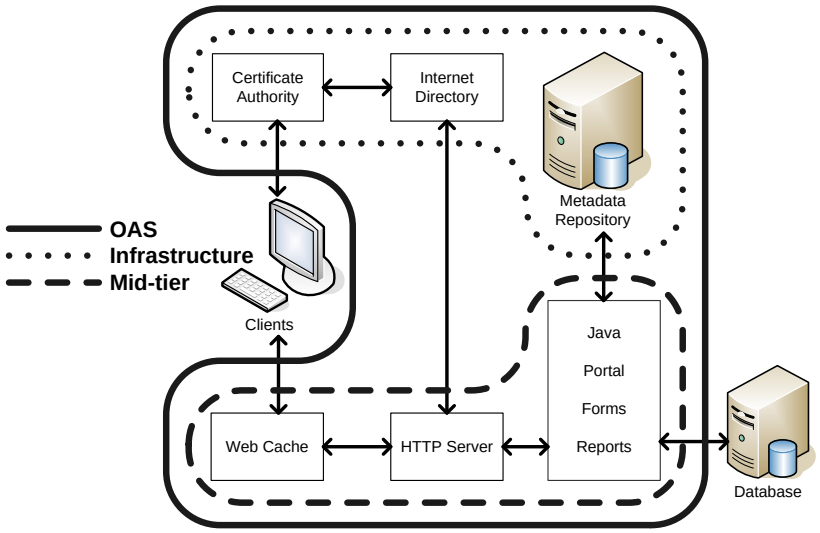

Figure 4.1: Basic OAS architecture.

4.1 OAS Mid-tier

The OAS Enterprise mid-tier can be installed in three different configurations: *J2EE and Web Cache* is the most basic installation and comes with Oracle HTTP Server, OC4J, and Web Cache. It is, however, also possible to do a more sophisticated installation that is called *Portal and Wireless*. The most complete installation is called *Business Intelligence and Forms*. This version is installed on the EAP test machine and contains also OAS Discoverer, OAS Forms Services, OAS Report Services, and OAS Personalization. Nowadays Oracle supplies various kinds of middleware-

[2] At 16/7/2007 the most recent version of OAS was version 10.1.3.2.0.
[3] At 16/7/2007 this is the most recent version for Linux systems.

products containing OAS and various collections of additional components. These products are for instance called 'Oracle SOA Suite 10g' or 'Oracle SOA Suite 10g Companion'[4].

4.1.1 Oracle HTTP Server

Oracle HTTP Server (also called Oracle Web Server) facilitates communication using the HTTP protocol. This product is plays a central role in OAS since all communication with Web Services and websites between OAS and external systems (for instance clients and other application servers) involves the HTTP protocol.

The open source Apache HTTP Server[5] is used as an underlying system for Oracle HTTP Server. This web server can run on both Windows and Unix platforms. It is also possible to use Microsoft's Internet Information Server or Java System Web Server in OAS, but then this third party web server simply forwards all requests to the the Apache HTTP Server that still is used to actually process and answer requests [Sch06].

A request can relate to static content that is directly loaded from the file system, but it can just as well relate to dynamic content that is provided by some executable resource. The functionalities to handle requests for dynamic content are provided by dedicated extensions that are called *mods* or *modules*. In this context, the four most important mods are:

mod_oc4j allows OAS to execute J2EE programs. More details about OC4J can be found in Section 4.1.3.

mod_plsql allows the execution of PL/SQL[6] requests in an Oracle Database.

mod_fastcgi allows the usage of any executable that can be called via CGI[7].

mod_perl allows execution of Perl programs. Unlike mod_fastcgi, this module is specifically optimized to operate Perl programs in an efficient way.

Static content is located on the filesystem of OAS, this also holds for the programs that are used to generate dynamic content. Oracle HTTP server in fact collects all the information and redirects it to external systems over the HTTP protocol.

Besides the HTTP protocol, Oracle HTTP Server also can use the HTTPS protocol. HTTPS is a secured version of the HTTP protocol that uses SSL[8] to encrypt data. More details about security related aspects of Oracle HTTP Server can be found in Section 6.6.1.

4.1.2 Oracle Application Server Web Cache

OAS Web Cache [Ste06] can be placed in front of one or more web servers to cache any content that is transmitted via the HTTP protocol (See Figure 4.2). This means that the OAS Web Cache can offload the Oracle HTTP Server. Static pages can be cached relatively easy. When handling dynamic content, caching is much more of a challenge. It is, however, still possible to achieve caching of the more static parts of the pages.

[4] See http://tinyurl.com/39464k, accessed on 08/06/2007.

[5] See http://httpd.apache.org

[6] Procedural Language/Structured Query Language, a programming language developed by Oracle to communicate with an Oracle database.

[7] Common Gateway Interface, a protocol facilitating communication between a web server and other applications.

[8] SSL stands for 'Secure Socket Layer' and can be used to secure various network based communication protocols. (Also see Section 6.1.)

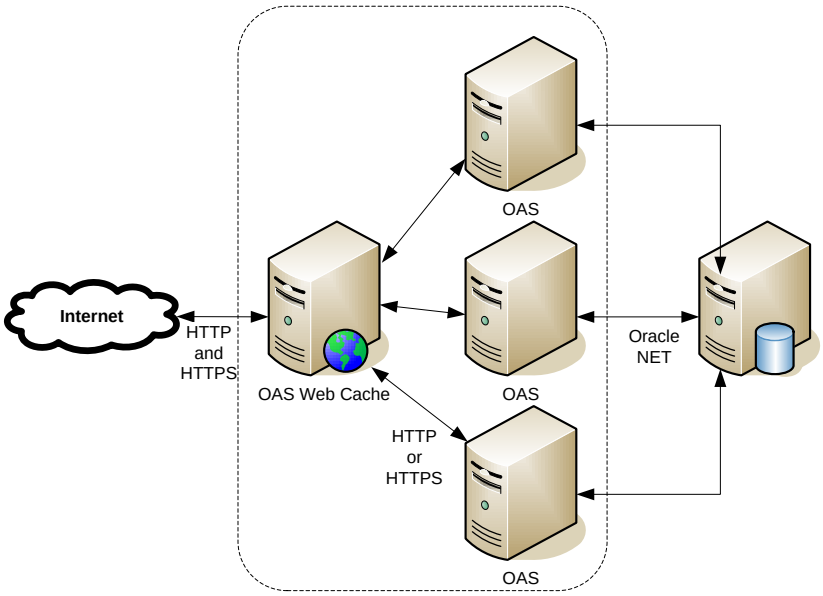

Figure 4.2: OAS Web Cache

When using HTTPS, OAS Web Cache is seen as an endpoint for external systems: the data between the client and the web cache can be encrypted, regardless of whether the communication between the web cache and the HTTP server is encrypted or not. More details about security related aspects of Oracle HTTP Server can be found in Section 6.6.2.

4.1.3 Applications

Various types of applications can be deployed in OAS. The most common application types are described in this subsection.

OAS Forms & Reports

OAS Forms is in fact an adapted version of an existing product called Oracle Forms. Oracle Forms could be used to create client applications that interact with an Oracle database. The client applications were written in PL/SQL to run as applications on a client computer. That made Oracle Forms a typical client-server (two-tier) product. In 1997 Oracle presented a adapted version of its Forms-product that was using the third tier (i.e. An application server). This product was called OAS Forms. The client application now is a Java applet[9] running in the client's browser. This Java applet communicates via HTTP(S) with a server side application called the OAS Forms Listener. The business logic of OAS Forms applications is executed by dedicated server-side PL/SQL programs that are invoked by the OAS Forms Listener. These programs run in the so called Forms Runtime Engine which communicates with the database using the proprietary Oracle NET protocol[10]. This structure is depicted in Figure 4.3.

[9] Needing a special Java runtime environment, so that an extra Oracle plug-in called JInitiator must be installed at the client-side.

[10] This protocol is designed by Oracle for accessing Oracle Databases.

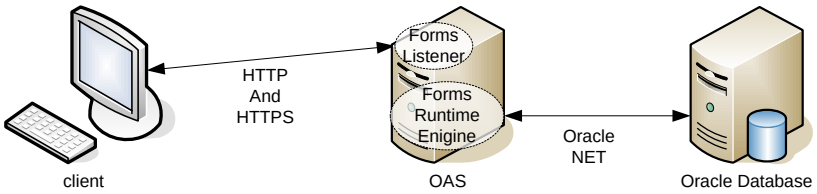

Figure 4.3: OAS Forms architecture

To start the Java applet, users must first connect to a special servlet called OAS Forms Service. This servlet enables the usage of single sign-on[11] identity management using Oracle Internet Directory[12]. It is also possible to stick to a classical log-on system using database accounts. In various departments of the Netherlands' government these three-tier forms are used, however single sign-on and OID are generally still not used.

OAS Reports has many similarities with OAS Forms. Also OAS Reports is an adapted version of an earlier Oracle product, namely Oracle Reports. Oracle Reports can present information from a database (or other sources) in a *report*. This *report* has a static nature and can be of various formats like HTML, PDF, or RTF. OAS Reports has supplied Oracle Reports with the opportunity to request and retrieve reports via a browser. Again, it is possible to restrict requesting reports by using database accounts or Oracle's single sign-on identity management. Like with OAS Forms, the OAS Reports Server is accessed via a dedicated servlet, here called OAS Reports Servlet. For the creation of reports, the OAS Reports Server invokes dedicated programs which actually handle the creating of the reports. For performance and availability reasons, it is possible to cluster various instances of OAS Reports Server. Also the created reports are cached so that they can be (partially) reused when possible. Since OAS Reports can be incorporated in OAS Portal which enables using more fine-grained restrictions by deploying portlets that display information that is extracted from OAS Reports. It is also possible to give access to OAS Reports via Web Services by using Oracle's dedicated *rwwebserverices* servlet.

OAS Portal

A *portal* is an environment that provides server-based access to various information sources. Examples of information sources are web pages, databases, and 'normal' applications. With OAS Portal one can use a step-by-step developing environment to create a portal containing information from various sources. The standard architecture of OAS Portal is depicted in Figure 4.4.

OAS Portal presents information in a hierarchical form. The OAS Portal *site* represents the highest level in this hierarchy. A URL prefix is used to refer to a site (for instance http://site1.eap.nl and http://site2.eap.nl would refer to EAP's OAS portal sites *site1* and *site2* respectively). Each site can contain several *pages* which can contain various *items*. An *item* can either be static content (for instance text, an image, or a URL) or a *portlet*. A *portlet* is a Java or PL/SQL program that, via *portal providers*, can communicate with (remote) data sources to deliver information. These data sources can for instance be a Web Service, a website, or a database. The Parallel Page Engine (PPE) plays a central role in handling requests for pages containing portlets. This PPE is a servlet that creates pages using information that is retrieved from portlets.

[11] SSO is described in Section 4.4.1.
[12] OID is described in Section 4.2.2.

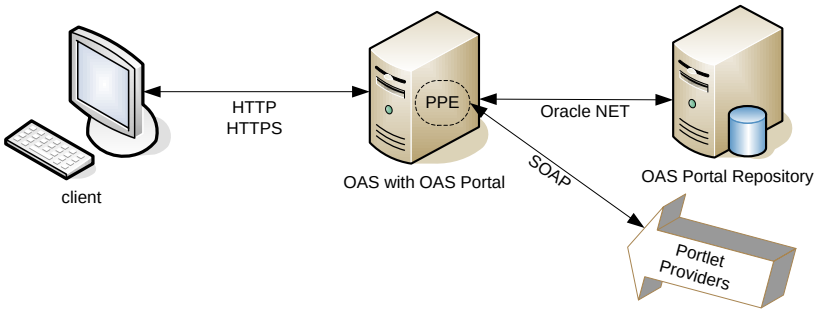

Figure 4.4: OAS Portal basic architecture

The configuration data for OAS Portal is stored in the so called OAS Portal Repository. By default, this repository is located in the same Oracle Database that contains the Metadata Repository. When performing a standard OAS Portal installation, the local database that also contains the OAS Metadata Repository is used for this matter. In practice, often a dedicated Oracle Database is used for the portal's repository. This can be achieved by installing the OAS Portal schemas[13] in this database. Then the OAS Portal's Database Access Descriptor (DAD) has to be modified to point to the external database.

Bringing portlets into action easy since Oracle has provided a HTML-based wizard that enables the instantiation of portlets by only using a browser. For extending the functionalities that are provided by the wizard, it is possible to resort to *OAS Portlet Development Kit* which contains APIs[14] that are needed to create customized portlets.

Using OAS Portal enables the opportunity of caching portlets so that it is not necessary for the portlet provider to contact the information source each time the portlet is invoked. Like other pages, it is of course also possible to cache pages from OAS Portal using OAS Web Cache.

OAS Portal can provide access to large amounts of information. To structure all this information, Oracle has developed a classification framework for OAS Portal using *categories* and *perspectives*. Each item has to be assigned to exactly one category. Content also can be assigned to one or more perspective. Each perspective is related to a role in the organization: for instance a perspective 'management' or 'sales' may exist. Both categories and perspectives can be arranged hierarchically.

For security, OAS Portal uses single sign-on and Oracle Internet Directory as described in Section 4.4. Specific user- and group permissions can be assigned to pages as well as to portlets.

Java

OAS Containers for J2EE (OC4J) provides the Java environment for the OAS. J2EE stands for *Java 2 Enterprise Edition* which is an extension to the standard Java 2 edition of Sun Microsystems. J2EE adds support for so called 'Enterprise Java Beans' to the standard Java environment. OC4J consists of three components:

A servlet container which is an environment for executing servlets. A servlet is a server side Java program. The container loads the appropriate class files, caches servlets when they have

[13] A schema is a collection of database objects (like tables, views, and functions) that are associated with a database user.

[14] API stands for application programming interface and is a library of building blocks for computer programs.

been loaded, and provides access to databases.

An EJB container which is an environment for executing Enterprise Java Beans (EJBs). EJBs can be divided into three types:

Session beans are used for business logic. A *statefull* session bean can store data that can be used in more than one session. A *stateless* session bean is a simple program that only contains business logic.

Entity beans contain the data that is loaded from the database. The container takes care of synchronizing the data between the entity bean and the database when the bean is configured to use *container-managed persistence*. When *bean-managed persistence* is enabled, programming code in the EJBs has to take care of the communication with the database.

Message driven beans use messages to send and receive service requests. Message driven beans are of an asynchronous nature, which means that program execution can continue without having to wait for a response.

A JSP translator builds servlets from Java Server Pages (JSPs). JSPs are compositions of HTML and Java code. For HTML developers JSPs are generally more convenient to write than servlets.

Enterprise Java Beans can also be used to built Web Services. These Web Services can be used by other application servers or by software that is installed at the client-side.

4.2 OAS Infrastructure

The OAS Infrastructure consists of all OAS components that are needed to support the OAS mid-tier. According to Figure 4.1 the core components of the OAS infrastructure are the OAS Metadata Repository (see Section 4.2.1), the OID (see Section 4.2.2), and the Certificate Authority (see Section 4.2.3).

4.2.1 OAS Metadata Repository

The 'OAS Metadata Repository' is a collection of configuration data for OAS that is located in an Oracle Database. Actually, this type of a repository is required for the OAS mid-tier to operate. However, for the *J2EE and Web Cache* installation type[15], a file-based repository is used by default (but also then it is possible to move the file based repository into an Oracle Database) [RWD+04].

The information inside OAS Metadata Repository is used by OAS components. Communication with these components takes place over the proprietary Oracle NET protocol. It is possible to install the OAS Metadata Repository (and the other infrastructure components) on the same computer as the OAS mid-tier, but for 'availability'-reasons is it is better to install the OAS Metadata Repository on a dedicated computer or cluster of computers. The OAS Metadata Repository contains application related metadata (for instance the datasources that are used by J2EE applications) and configuration related metadata (for instance which OASs are using that specific OAS Metadata Repository). One OAS Metadata Repository can be used by more instances of OAS (see Section 4.3.1).

[15] See section 4.1.

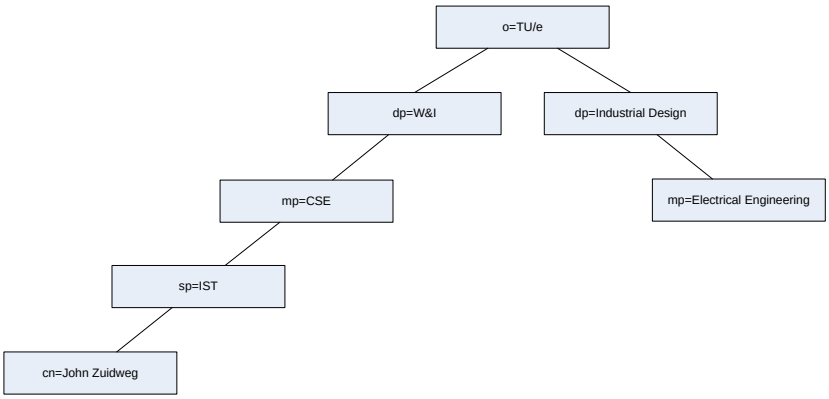

Figure 4.5: Example of a directory information tree (DIT)

4.2.2 Oracle Internet Directory

Oracle Internet Directory (OID) is the centrepiece of Oracle's *Identity Management*. OID is a *directory service*, that is a framework that provides the listing of principals and their specific details. A classical example of a directory service is a telephone book. An online version of such a directory service is in fact a database –that is, a structured collection of data [DS05]. Its interface is, however, different from a 'normal' *relational* database in the sense that a directory service stores *entries* that are associated with a number of *attributes*. Each entry can be uniquely identified by means of a hierarchical structure that is called the *directory information tree* (DIT). Each entry in this DIT can be identified by a so called *distinguished name* (DN). An example of a DN is:

$$\mathrm{cn{=}John\ Zuidweg,\ sp{=}IST,\ mp{=}CSE\ dp{=}W\&I\ o{=}TU/e}$$

Where cn stands for common name, sp for specialism, mp for master's programme dp for department, and o for organization. Figure 4.5 shows an example of a DIT containing this DN.

So, a directory service works with *entries* that are identified by *distinguished names* in a hierarchical structure. A *relational* database, on the contrary, does work with rows and columns that contain a unique *primary key* per row.

An example of an entry is a user, an example of an attribute is a password. So where a relational database can be queried to show, add, update or delete certain (elements from some) rows in a table, a directory server can be asked to show, add, update or delete entries, attributes, or attribute values. The most common operation for directory servers is showing attribute values. Regardless these differences between relational databases and directory servers, OID actually uses a (relational) Oracle Database to store its data. By default, OID uses the same (local) database as the OAS Metadata Repository. All attributes and entries are stored in tables in this database. Communication between this database and the 'Directory Server' takes place over the propriety Oracle NET protocol.

This 'Directory Server' provides an interface (by default listening on port[16] 636 (SSL) and 389 (non SSL)) that makes the directory service available. This interface is compliant with the LDAP-standard. LDAP stands for *Lightweight Directory Access Protocol* and this standard prescribes the way in which directory-servers and directory-clients can communicate with each other. Using this

[16] A (software) port is a logical data connection that is addressed by a number. HTTP servers are for instance listening on port 80 by default.

LDAP protocol, OID can support *Oracle Identity Management* by providing the information that is needed to perform services like authentication and authorization of users. It is important to consider that OID is not a security product, but only a directory service that provides enterprise information including user account information.

OID has an internal system for access control to the information it contains. This system provides a hierarchical structure for managing and delegating privileges (like compare, delete, read, and browse) in OID. Users can access the OID via the web-based *OID Self Service Console* or by means of a Java-tool called *Oracle Directory Manager*.

4.2.3 OAS Certificate Authority

The Certificate Authority (CA) in OAS is a component of Oracle Identity Management that issues, revokes, renews, and publishes X.509v3 certificates. X.509v3 stands for version 3 of the X.509 standard for managing a Public Key Infrastructure (PKI). Participants in a PKI can possess a *key pair* consisting of both a public key and a private key. The private key can be used to provide messages with a digital signature so that data integrity, authenticity, and non-repudiation can be guaranteed. The receiver of the message can verify that signature by using the sender's public key. Both the validity and the authenticity of this public key can be checked by means of a certificate. This certificate contains the public key of the sender, together with some additional information. The most important aspect is that this information has to identify the sender uniquely. Each certificate contains a digital signature of its issuer. In this way, a whole tree-like structure of signed certificates can be build. The process ends when a so called 'root certificate' is reached. Root certificates are either pre-installed or have to be installed manually.

A PKI can not only be used for *signing*, but it can also be used for *secrecy* by encrypting and decrypting messages. In short it works as follows: the sender of a message uses the public key of the receiver to encrypt a message. The receiver then, after getting the message, can decrypt it by using its private key. The sender can look up the signed version of the receiver's public key by using the receiver's certificate.

Besides *signing* and *secrecy*, [Mis05] explains that a PKI can be used to perform *server authentication* as well as *client authentication*. These techniques are used in the SSL-protocol server-authentication, or for authentication of both the server and the client (mutual authentication). Client certificates can be installed in the users' browser so that mutual SSL authentication can be performed. The private key of the user is stored in a so called wallet: an encrypted file on the client's computer that is only accessible after successful authentication. For this authentication, it is possible to use simple passwords, but also physical tokens like smartcards can be used, even in combination with biometric verification.

OAS CA plays a central role in managing certificates. Depending on the CA's configuration, users can request their own certificates –including their private key– either by only using a web interface or by also physically identifying themselves to an administrator. In the first case, users first have to authenticate themselves either by using single sign-on or by performing mutual authentication using an already installed client certificate.

It is also possible to authenticate components in the OAS architecture: for instance a mutually authenticated SSL session can be arranged between the OC4J component of OAS and an Oracle Database. To achieve this, both components need be in possession of a client certificate. In this case, this certificate is not installed in a browser, but in a special application called Oracle Wallet.

One important aspect has not be mentioned until now: Certificate Revocation Lists (CRLs): When a private key gets compromised, it is essential that the corresponding certificate is known

to be not longer valid. To achieve this, a mechanism using these Certificate Revocation Lists has been implemented. This list, which contains identifiers to all revoked certificates, is consulted on regular basis (frequency is configurable) by users of certificates. Additionally, an expatriation date is assigned to each certificate. In principle, the longer the certificates's lifetime, the more needful it gets to consult this CRL.

In this whole context, the OAS Certificate Authority plays a central role. However, OID is still used to lookup and manage user information. The OAS Certificate Authority uses the OID to find *principals* so that certificates can be bound to *principals* in the OID.

4.3 Availability of OAS

Even mission critical computer systems, which need to be available anytime, are subject to planned and unplanned downtime. There are many possible causes of a system being down; maintenance plays a part (as a planned matter) but, in the main, the origin lies in unplanned matters like security breaches (attacks), system failure, disasters or human errors. At least for mission critical systems, measures need to be taken in order to overcome the fact that each computer system will be faced with downtime eventually. Basically, two things can be done to overcome problems with the downtime of computer systems: using redundant systems (see Section 4.3.1) and creating and maintaining backups (see Section 4.3.2).

4.3.1 Scalability of OAS

[GSB04] distinguishes two main types of scalability. *Vertical scaling* can be achieved by running more than one OCJ4 instance per machine. Such a group of OCJ4 instances that share the same configuration is called an 'island'. In this way, it is possible to let each application run in its own OCJ4 environment. This reduces the risk that the deployment of an application affects the implementation of another one.

Vertical scaling can also be achieved by letting an OC4J instance running more than one Java Virtual Machine (JVM). JVM is a program that is needed to interpret and execute compiled Java applications. This type of vertical scaling is particulary advantageous on machines with multiple CPUs since each CPU then can get its own JVM.

It is also possible to use *horizontal scaling* by grouping two or more OASs into a cluster. Before OASs can be grouped into a cluster, they all have to be part of a so called farm. A farm is defined as a group of OASs that shares the same infrastructure. Clustered OASs appear to users and administrators as one OAS, but in the background all OC4J instances and Oracle HTTP Servers within a cluster are working together. This grouping of OC4J instances and Oracle HTTP Servers is achieved by means of a load balancer that forwards requests to a clustered OAS instance. Such a load balancer can be a dedicated hardware device, but it is also possible to perform load balancing either by means of the operation system or the OAS Web Cache.

OASs that are clustered share the same configuration, except for the setting of the number of JVMs per OC4J instance and the numbers of the ports that are used for the communication between OC4J and Oracle HTTP Server.

Scalability is a key issue for application servers. As explained in Chapter 2, increasing systems' performance was an important reason to come up with the 'middle-tier'. Further, especially vertical scaling also can help to overcome the downtime of a computer systems since it offers a form of redundancy: if one instance of OAS fails, the other OASs within the cluster will auto-

matically take over its tasks. As mentioned before, all OASs within a cluster do share the same infrastructure. This means that the failure of an infrastructure component does affect all the OASs within that specific cluster. To make the cluster capable of coping with the failure of an infrastructure component, it is possible to also group two or more infrastructure servers into a cluster.

4.3.2 OAS Backups

To prevent data loss, it is important to create backups regularly. Obviously, this can help to improve the availability of the system. Creating backups of OAS can be automated for the most part using Oracle Secure Backup [AG06]. In order to get useful backups, it is essential to lay down what, when (i.e. the frequency) and how data is backed up. According to [AG06], file system backups are not stored in encrypted form. Database backups, however, are stored in encrypted form by default. The fact that file system backups are not encrypted by default gives reason for extra measures: at least for 'confidentiality'-reasons, tapes containing this un-encrypted data should be stored in a safe environment or they should be encrypted manually.

This far, only 'static' backups were described. There are, however, also more dynamic forms of performing the process of backup and recovery. Vertical scaling of OASs (as described in Section 4.3.1) can be seen as a form of a dynamic backup. This also holds for the undo / redo logs of the Oracle Database (which is an extensive topic that is not discussed in this thesis).

In situations of catastrophic site failures (for instance as a result of an earthquake, a tornado, flood, or fire), all both these static and dynamic backups are most likely not useful as long as they are located on the same site. Therefore, it is a good idea to store backups at a geographically remote location[17]. This also can be performed for dynamic backups. According to [RWD+04], a product called 'OAS Guard' can be used to facilitate this. The basic idea is that two identical OAS architectures are installed: a production architecture at the local site, and a standby architecture (including both the mid-tier and the infrastructure of OAS) the at a geographically remote site. 'OAS Guard' then can be configured to sent data from the production site to the standby site so that in case of a failure at the production site, the standby site can take over its tasks relatively easy. Technically, this can for instance be realized by using a DNS-server[18] that bounds a domain name to the production server by default, and that bounds this domain name to the stand-by server when a failure of the production site is detected. By default, SSL is used to encrypt the data that is transferred between the production site and the stand-by site.

An important note is that backups (whether static or dynamic) should be tested regularly: according to colleagues at EAP this crucial step often is forgotten in practice.

4.4 Identity Management

According to [DS05][19], *identity management* is "the process of managing the complete security life cycle for network entities in an organization". [And01] explains that the term *identity* expresses "the correspondence between the names of two *principals* signifying that they refer to the same person or equipment". In general, a *principal* is an entity that participates in a security system.

[17] Having off-site backups is also required by [ITG07], also see Section 5.3.3.

[18] DNS stands for Domain Name System and is a protocol that is for instance used to translate domain names (like eap.nl) to IP-addresses that uniquely identify nodes on a network.

[19] Towards writing this section, I have read trough several hundred pages of Oracle (related) documentation. Also I have examined various aspects in practice, using the version of Oracle Application Server at EAP's 'test lab'.

Besides a person or a role, this entity can be for instance a piece of equipment, a compound of principals, or a delegation [And01].

In the literature, many publications about identity management are focussed on the process of maintaining persons' digital identifiers and privileges over the Internet for privacy [Seb01]. Also much can be found about *federated* identity management: managing identities and privileges across different companies' networks [PW03]. Furthermore, much can be found on the topic of single sign-on (SSO) [PM03] –that is addressed in Section 4.4.1 in more detail. Oracle Identity Management is more focussed on this latter concept [Lee03], aiming at interlinking users, applications, and computer systems.

The development of this type of identity management was incited by the observation of practical problems: Commonly, within enterprises an extensive collection of software is used. In the worst case, each software application has its own mechanisms for authentication and authorization. As a result, users have to remember various usernames and passwords which may ultimately deteriorate security as users are caused to write down their user names and passwords. This awkwardness does, however, not only bother the users of the applications: Within enterprises, system administrators are faced with *joiners*, *movers*, and *leavers*. While having this extensive collection of independent applications, a lot of work has to be done in order to manage the setup, the maintenance and the dismantling of accounts. Moreover, the complexity of account management structures may result in failures in the provided authorizations. Users might for instance end up with more authorizations then they need, resulting in an increase of fraud possibilities and the violation of the idea of separation of duties. On the other hand supplying users with too few authorizations may also provoke circumventing security measures (for instance by hacking or by exchanging log-in details).

So, Oracle Identity Management is designed to overcome these problems. [Des05] explains that this system consists of six components:

A directory service which offers the functionality of storing and managing information about *principals* and their *attributes*. These attributes give information about the principals. Principals' unique identifiers and privileges are the most important attributes, but commonly also items like contact details are appended as attributes. In OAS, the task of storing and providing this information is generally performed by *Oracle Internet Directory* (OID) (see Section 4.2.2). It is, however, also possible to integrate OAS with other directory services like Microsoft's Active Directory.

A provisioning frame and an integration platform which facilitates the administration of principals and their attributes. In OAS, OID provides basically three different provisioning systems:

- An SSL-secured web-based environment called *OID Self Service Console* whereby users can change their own password and update information like their contact details. Via this same system, delegated administration of entries is possible so that for instance managers can update information regarding their subordinate principals.
- A Java-tool called *Oracle Directory Manager* by which users can manages their (subordinates') entries in OID directly (after successful authentication to OID). Administrators can also use this tool carry out maintenance work.
- A component called *Oracle Directory Integration and Provisioning Platform* that integrates OID with other sources of directory information (for instance Microsoft Active Directory).

A certificate manager which manages *certificates* that can be used to assure that information originates from the claimed entity. Oracle provides the *OAS Certificate Authority* to fulfill the tasks of the certificate manager. Section 4.2.3 explains the characteristics of managing certificates using this *OAS Certificate Authority*.

An authentication model that is designed to perform user authentication. This component is responsible for *Access Control*. Within OAS, the notion of *single sign-on* (see Section 4.4.1) plays an important role regarding this topic. This also holds for the *Java Authentication and Authorization Service* (see Section 4.4.2) from Sun Microsystems.

A delegation system that enables the delegation of privileges. In OAS this can be done by using *Oracle Delegated Administration Services* which is incorporated in the *OID Self Service Console*.

4.4.1 OAS Single Sign-On

As mentioned before, Oracle Identity Management facilitates single sign-on (SSO). Basically this mechanism works as follows: Users only have to authenticate themselves once to the so called *SSO server*[20], after which they are automatically authenticated for all applications that are registered in this server. These applications can be of two types: according to [Wis06], *partner* applications delegate the authentication function to the OAS single sign-on server. Figure 4.6 shows that the process of accessing a partner application basically consists of six steps:

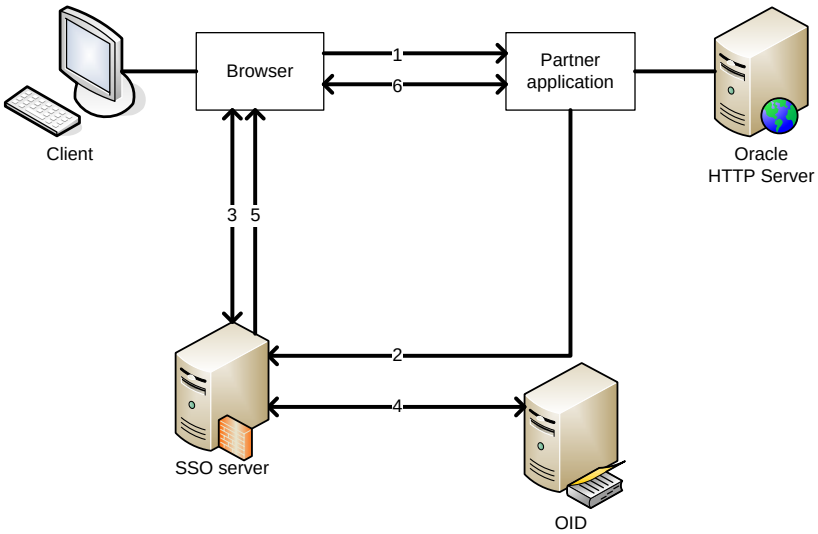

Figure 4.6: OAS SSO for *partner* applications.

1. A client uses his browser to access a partner application via the Oracle HTTP Server (examples of partner applications include OAS Portal, and Oracle Delegated Administration Services).

[20]This *SSO server* is no dedicated server: it is in fact a special application running Oracle HTTP Server that is invoked when specific URLs are requested.

2. The partner application observes that it not yet has authenticated the client (no valid *application cookie*[21] is present in the client's browser). Therefore, the partner application redirects the user to the SSO server.

3. The SSO server checks if it has already authenticated the client. In Figure 4.6 this is not the case (no valid *SSO cookie* is present in the client's browser), so a login window is presented in the client's browser challenging for the user's SSO username and password (or requesting a client side X.509 certificate, see Section 4.2.3). By default, all communication between the SSO server and the client's browser is secured by SSL.

4. The SSO server retrieves the supplied credentials of the client and checks that with the information in OID. Also the users privileges w.r.t. the requested partner application are checked.

5. The SSO server sends an *SSO cookie* to the client's browser (so that the user can access other applications during this session without authenticating to the SSO server again) and redirects this browser to the partner application using a special URL containing an encrypted *authentication token* that is used to prove that the client indeed is allowed to access this partner applications.

6. The partner application sends an *application cookie* to the client's browser (so that the user can access this specific application during this session without interacting with the SSO server again) and serves up the requested content. (If applicable, the partner application then can then retrieve the client's privileges from OID.)

Obviously, this mechanism is inspired by the famous Kerberos protocol [KN93] (with its ticket-granting-tickets). In fact, OAS SSO can be integrated with Kerberos. Additionally, it can also be integrated with Microsoft's Active Directory so that native network authentication (using the login information for the domain account of Microsoft Windows) can be used to authenticate clients, so that no credentials need to be supplied by the client.

External applications, on the other hand, require a unique user name and password. OAS SSO server can be used to store these credentials so that is can log the user in without requiring the user to authenticate again for this specific external application. In fact, the SSO server works here as a kind of password manager that organizes users' credentials. Figure 4.6 shows that the process of accessing a partner application basically consists of six steps:

1. A client uses his browser to access an external application via an external HTTP server.

2. To access a registered external application using SSO authentication, users have to use a so called virtual URL that actually points to the SSO server instead of pointing directly to the external application.

3. The SSO server checks if it has already authenticated the client. In Figure 4.6 this is not the case (no valid *SSO cookie* is present in the client's browser), so a login window is presented in the client's browser challenging for the user's SSO username and password (or requesting a client side X.509 certificate, see Section 4.2.3). By default, all communication between the SSO server and the client's browser is secured by SSL.

[21] Cookies are small files –for instance containing session information– that are located in the client's browser.

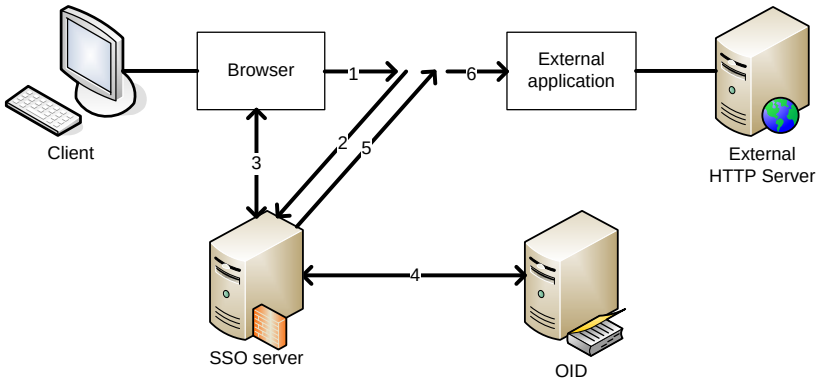

Figure 4.7: OAS SSO for *external* applications.

4. The SSO server retrieves the supplied credentials of the client and checks that with the information in OID. Also the users privileges w.r.t. the requested partner application are checked.

5. The SSO server redirects the user to the login page of the external application.

6. The client's credentials are enclosed in this request, so that the client is logged in automatically (By default, the client's browser is communication directly with the external HTTP server from this point on. It is, however, also possible to use the SSO server as a so called 'proxy server'. This 'proxy server' communicates with the external application so that the user is not longer redirected to the external application, but all communication between the user and the external application takes place via the SSO server.)

Besides single sign-on, OAS SSO also offers a functionality called single sign-off. When a client requests this single sign-off operation, the SSO server tries to remove all *application cookies* and the *SSO cookie* that are present in the client's browser.

4.4.2 Java Authentication and Authorization Service

The OAS Java Authentication and Authorization Service (JAAS) is a set of Java classes that can be used as APIs to enable custom J2EE applications to use SSO, OID and client certificates [Per05]. In fact, OAS JAAS is a modified version of the JAAS from Sun Microsystems[22]. Since JAAS is considered as a 'proven' solution, it is advisable for Java developers to utilize this package in lieu of custom security modules.

4.5 Access Control

From IT auditors' perspective, access control is an important topic. Central questions are 'who has which type of access to what?', 'how are access attempts recorded?', and 'how can access violations be detected?'. This section addresses these topics, focussing on OAS. First, the way in which users can access information using OAS is examined, followed some details about logging.

[22] See http://java.sun.com/products/jaas, accessed on 3/9/2007.

Furthermore, some access control models are described briefly. The section is completed by some remarks on access control related Oracle products.

4.5.1 Access by Clients

When addressing the topic of access control in OAS, it may be clarifying to look into access control in Oracle Database first: Normally, each Oracle Database has a number of so called *schema users*. Each *schema user* represents in fact a user who can authenticated by a login name and a password. This user *owns* a number of database objects (i.e. tables, views, procedures) to which the user has access. The special schema users SYS and SYSTEM have access to all database objects. These schema users also have access to special system privileges such as the right to create new schema users. These system privileges can be delegated to other schema users, it is even possible to delegate the right to delegate permissions.

When using OAS, direct user communication with the database can be avoided. In fact, Oracle advises to use an architecture in which it is impossible for end users to communicate directly with the database: all communication takes place via the application server. In that case, in theory, it suffices to have only one schema user in the database so that OAS can connect with the database using this account.

Within the OAS architecture, the three most important components to protect are: the Oracle database, the OAS Metadata Repository and the OID[23]. These components ultimately need protection since they hold all the information that is used in the OAS architecture. As depicted in Figure 4.1, basically all users' communication with the database, the Metadata Repository, and OID takes place via OAS Portal, custom Java web applications, or Oracle Forms & Reports.

Access to OAS Portal is provided either by means of SSO (see Section 4.4.1) using OID (see Section 4.2.2) or by mutual authentication using SSL (see Section 4.2.3. Of course, it is up to the designer of Portal pages to decide who to provide access to which items.

Custom Java applications can also make use of OID and SSO or mutual SSL authentication by incorporating the JAAS (see Section 4.4.2 API. It is, however, also possible to build J2EE applications that use OID without SSO, or that don't use both functionalities. The way in which access control is implemented in these to latter cases is up to the programmer of the Java applications.

When using Forms & Reports, using SSO is optional: when using SSO the user or administrator has to enter connection details for the database (including a username and a password of a schema user) before access to the information is provided. When SSO is not used, a login screen asking for these database connection details is presented each time a user starts using a form or a report. In both cases, the security of the system depends mainly on the design of the forms or report application, together with the privileges of the database account (that is the schema user) which is used.

So in fact, the most important issues in access control lie in the design of the application; whether it is a portal page, a custom Java application, or a Forms & Reports program. Still, also the configuration of the complete OAS system is relevant. For instance the decision whether or not to enforce SSL (possible with using mutual authentication) for all or some connections can be of great influence. This also holds for the way in which OID is organized (See Section 4.2.2).

[23] As explained in 4.2.2, the Metadata Repository is stored in an Oracle database and the OID data also is located in this database by default.

4.5.2 Logging

Each component in OAS has its own logging mechanisms that record various types of events like errors, access attempts, warning messages and reports about start-ups and shutdowns. Oracle has developed a mechanism to make a number of these logs available for administrators via a standard web interface (see Figure 4.8). This web interface is part of the Oracle Enterprise Manager that is installed as part of each OAS installation by default. Administrators can log on to this web interface using a browser to perform various administrative tasks.

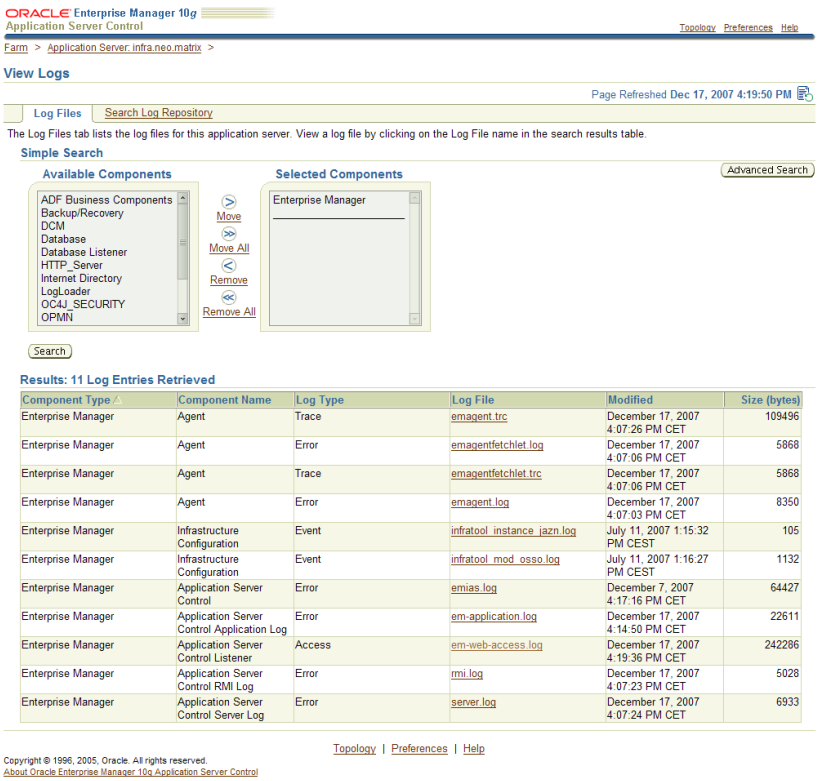

Figure 4.8: Screenshot of web interface for finding and reading logs.

From an auditability perspective, it is a good idea to have such a central log repository. However, it would have been even better if logs were written in a user-centric form so that an audit trail per user could be created. At the time of writing, OAS lacks this functionality.

Notice that the directories where the logs are written should be protected so that users can not modify or delete log files easily. Moreover, log files should be reviewed periodically looking for critical events (for instance find out who has logged on to the Enterprise Manager using the Enterprise Manager log).

In Oracle Database it is possible to log all mutations in the database (of the certain type) (that are performed by a specific schema user). This log, that is called an audit trail, is stored in the database itself. It can also be configured to store identifiers of SSO users instead of database schema users. The latter indeed should be performed when SSO accounts are used, because otherwise auditability will be hard to achieve[24].

[24] Also [BSI99a] recommends that user IDs should be logged so that actions can be related to users.

Oracle Audit Vault is a special Oracle product that can be used to collect audit data in a secure dedicated environment.

4.5.3 Security Models

In the field of access control, Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role Based Access Control (RBAC) are important terms. This subsection briefly examines each of these terms within the context of OAS.

Discretionary Access Control is the type of access control that is most commonly seen in operating systems: subjects (for instance users or groups of users) can own objects (for instance programs or files). These subjects have certain privileges regarding these objects and these privileges can de delegated to other subjects. In this way, it is possible to create an access control list for each object in which is stated which principal has what privileges regarding that object. In fact, this type of access control is seen also seen in both OID and Oracle Databases when leaving everything to standard options.

It is, however, also possible to use Mandatory Access Control in both the Oracle Database and OID. MAC demands that every object is provided with a so called *sensitivity label* and every subject is provided with a so called *clearance label*. This *sensitivity label* expresses the secrecy level of the object (for instance: unclassified, restricted, secret and top secret) as this *clearance label* expresses the authorization level of the subject. A subject that is authorized as 'top secret' can for instance read en write to objects on that level. In both Oracle Database and in OID this type of access control can be enabled by installing *Oracle Label Security* and *Enterprise User Security*. By means of these components, Oracle Database can be integrated with OID. Further they enable the usage of sensitivity labels and clearance labels by providing OID with options to assign labels to subjects and objects (for instance tables in the Oracle Database). In fact, VIR-BI[25] demands the use of MAC for confidential and secret information.

Role Based Access Control (RBAC) is a concept of a different nature. It advocates the usage of roles. A role is different from a group as a group is only a collection of users where a role is a collection of users plus permissions that are bound to that role. RBAC can be used in combination with DAC but also with MAC. In both OID and Oracle Database roles can be used to manage principals' privileges. Creating and assigning roles is not a trivial job: each user should end up with the exact privileges that are needed. Here, also issues like separation of duties play a role. The whole field of role management is quite extensive and is probably worth a thesis on its own. It has to be considered that a 'silver bullet', a solution that solves all difficulties, does not exist. As a matter of fact, it all boils down to creating appropriate roles and assigning them to users carefully.

Oracle Label Security and *Enterprise User Security* were already mentioned as additional components of OAS. Oracle has provided more of these additional components that are related to access control. Talking about these components may sometimes result in confusion of tongues since the marketing department of Oracle has decided to change these components' names every now and then. For instance the *Oracle Access Manager* (formerly known as *Oblix NetPoint* and *Oracle COREid*) provides a web-based management system for administering user and organization information, access control, and integration services. This system is more sophisticated than the management systems that are described in this section.

[25] See Section 5.1.

Chapter 5

General Audit Objectives

As mentioned before, audit objectives are criteria that are used to evaluate IT systems. Concrete criteria that are used to evaluate systems are often based on commonly accepted general audit objectives: a collection of general IT auditing standards and relevant laws. This chapter evaluates a number of these general audit objectives that are used by EAP. Hereby the focus is on those criteria that are applicable to the evaluation of OAS. Section 5.1 explains the implications of the VIR, a regulation regarding governmental information security in the Netherlands. Section 5.2 discusses the most relevant aspects of the international BS 7799 standard that has been adopted as ISO/IEC 17799. Section 5.3 addresses the best practices that are described in CobiT. Section 5.4 examines the contents the ITIL framework and Section 5.5 addresses the Common Criteria. Section 5.6 provides a brief comparison of the described general audit objectives, it also gives some critical remarks on these audit objectives. Then section 5.7 winds up with briefly mentioning other standards and laws that are related to the discussed general audit objectives.

When EAP is asked to perform a technical IT audit, first the specific audit objectives for the object that has to be audited have to be composed. In theory these audit objectives are composed in consultation with the audit's principal (the one who demands the audit), in practice often existing collections of audit objectives are used.

The general audit objectives that are discussed in the next sections were not specifically developed for auditors. All the standards can just as well be used by administrators and system architects when building and managing an information system. This even holds for CobiT, that is created by ISACA. Nevertheless, all these general audit objectives are particularly useful for IT auditors when they want to compose concrete audit objectives and audit programs. In this chapter, all general audit objectives are viewed from the perspective of the IT auditor.

5.1 VIR

As mentioned before (see Section 3.1), governmental organizations in the Netherlands have to comply with the '*Voorschrift Informatiebeveiliging Rijksdienst*' (VIR)[1]. The VIR itself is formulated in a very general way, but the ministry of the Interior and Kingdom Relations has published a book [vBZ95] in which the implications of this regulation are described in a more concrete and technical way.

[1] The VIR passed in 1994, on 20 June 2007 a new version of this regulation was published.

The general concepts of 'information supply' and 'information systems' are defined in [vBZ95] as follows:

Information supply includes all activities that are necessary to supply an organizational unit with all information that is necessary to fulfill its allocated tasks. (Dutch: "*informatievoorziening is het geheel van activiteiten dat nodig is om een organisatie-onderdeel te voorzien van de informatie die nodig is om de toegewezen taken te vervullen.*")

An information system includes a complete set of data, software, hardware, and people together with the procedures that are followed by these people. (Dutch: "*informatiesysteem = een geheel van gegevensverzamelingen, programmatuur, apparaten en personen met de procedures volgens welke zij werken.*")

Also the notion of 'information security' is defined:

Information security includes a coherent set of measures that are taken and supported to guarantee the reliability of an information systems and, thereby, the information it contains. (Dutch: "*informatiebeveiliging is het treffen en onderhouden van een samenhangend pakket van maatregelen ter waarborging van de betrouwbaarheid van een informatiesysteem en daarmee van de informatie daarin.*")

The term 'reliability' is defined by [vBZ95] as 'the extent to which an organization can rely on an information systems for its information supply' (Dutch: "*betrouwbaarheid is de mate waarin een organisatie zich kan verlaten op het informatiesysteem voor zijn informatievoorziening.*"). [vBZ95] divides 'reliability' into three parts[2]:

confidentiality ('excusiviteit') is defined as the extent to which access to system components is restricted to a group of authorized persons. This definition is in line with the description of this term in Section 3.3.

integrity ('integriteit') is defined as the extent to which system components (and in particular the information and the implementation) are error-free. [vBZ95] divides integrity into six characteristics: *correctness* (in accordance with the reality), *completeness*, *actuality*, *robustness* (process can continue after a change in the system), *consistency*, and *auditability* (the course of the process has to be traceable). This definition of integrity is broader that the description of this term in Section 3.3: EAP discerns 'auditability' as a distinct criterium where [vBZ95] defines it as an aspect of integrity. Furthermore, EAP sees *robustness* more as an availability-related criterium.

availability ('beschikbaarheid') is defined as the extent to which the information system is in operation when that is needed by the organization. [vBZ95] explains that, from this perspective, also 'fixability' and 'replaceability' of system components is important.

Like [Loo97] and [ITG07] in Section 3.3, also [vBZ95] uses the term 'reliability' as a mix of functional correctness and security: especially integrity-related aspects like 'consistency' and 'completeness' are more related to functional correctness than to security.

When ICT-related activities are contracted out, departments have to be able to check that the external parties do comply with the standards that are defined in the department's information

[2] See Chapter 3.3 for a more thorough discussion of these 'information (security) criteria'.

security policy. In this process, ISO-9000 certificates can help. An organization that is ISO-9000 certified can be judged on basis of a 'quality-handbook' that needs to be published by the organization and that is checked by an independent certification bureau. It is also possible to rely on a 'third party statement'[3]. A third way to check the manner in which information security is guaranteed is verifying the output of information systems; one can for instance think of checking all monetary transactions by hand. When auditing OAS, these issues can be relevant since OAS is a product of Oracle, an external party. Moreover, it is possible that the hardware supporting OAS is located at a third party location like a server centre.

[vBZ95] also advocates that an information security policy should be laid down. Such a policy contains strategic decisions about the way in which information security is viewed and handled within a certain department. An important part of such a security policy is the analysis of requirements and threats. This notion is elaborated in the so called dependency and vulnerability analysis (in Dutch: *afhankelijkheid- en kwetsbaarheidanalyse*, abbreviated as A&K-analyse). The dependency analysis is meant to provide insight in the extent to which business processes depend on an information system. This dependency analysis is followed by a vulnerability analysis. [vBZ95] is stating something remarkable here: the term vulnerability analysis is used to express the process of choosing measures to confront vulnerabilities. This can be quite confusing since one would expect a vulnerability analysis to be an investigation of the threats and especially the effects of the manifestation of those threats. In [vBZ95], however, the threat analysis is identified as a separate step that has to be performed prior to the vulnerability analysis. When analyzing threats, the usage of risk-analysis ($\text{threat} = (\text{risk} \times \text{impact})$) is depreciated by [vBZ95] since the impact is never fully quantifiable, the exact risk mostly is not known, and a rational basis is absent for most people when making security decisions. The measure-oriented vulnerability analysis from [vBZ95] identifies four different kinds of measures: preventive, detective, repressive (repress the impact of incidents, for instance installing extinguishing equipment), and corrective (for instance back-ups) measures. In [vAZ07], the dependency and vulnerability analyses is not longer prescribed because –[vAZ07] explains– the focus should not be on a technique, but on the identification of reliability demands and the resultant measures.

According to [vBZ95], it is important to audit the security policy on a regular basis. The mandator of such an audit is the Secretary General[4] of the department from which the security policy needs to be audited. EAP can be asked to perform such audits. Not only the security policy needs to be audited, but is also important to keep track of security incidents. The security policy must contain information about the way in which these incidents are reported and treated. Reporting security incidents is important because it makes clear which kind of measures need to be taken. An important aspect in [vBZ95] is the way in which responsibility of security matters is allocated: each line manager who manages information is responsible for the security of that information. Therefore, each line manager has perform a dependency and vulnerability analysis that shows which measures need to be taken.

In 2004, the VIR was supplemented by the 'Voorschrift Informatiebeveiliging Rijksdienst - Bijzondere Informatie' (VIR-BI)[5]. This regulation prescribes the way in which information security is handled when restricted (in Dutch: '*departementaal vertrouwelijk*'), confidential (in Dutch: '*staatsgeheim confidentieel*'), secret (in Dutch: '*staatsgeheim geheim*') or top secret (in Dutch: '*staatsgeheim zeer geheim*') data is involved. In fact, VIR-BI only is concerned with the confidentiality

[3]This notion is explained in Chapter 3.4.

[4]The Secretary General (Dutch: *Secretaris-Generaal*) of a department is a public servant in the highest rank at that department.

[5]See for instance http://tinyurl.com/2frtnt, accessed on 2/1/2008.

and integrity criteria. [KG95] provides some critical comments on VIR-BI: the issue of backing up secret information is not addressed in VIR-BI, which is an important shortcoming.

[vBZ95] identifies three main sources for the derivation of security requirements. The first source that is mentioned is a risk analysis. Although this technique is depreciated by [vBZ95], it is clear that this method can be of great help in identifying risks and making them explicit. As a second source, [vBZ95] explains that various legal, statutory, regulatory and contractual requirements need to be satisfied by an organization and its trading partners. Laws like the Dutch privacy law need also to be taken into account. Further, extra legal, statutory, regulatory, and contractual requirements may differ per (sub) department or even per system. Last years, an increasingly number of legal requirements has been imposed by the the European Union. An example of such a legal requirement is the 1999/93/EC European directive on electronic signatures. The organization's set of principles, objectives and requirements for information processing is identified as the third source of requirements. For this project, only the first two sources will be taken into account since the approach needs to be general enough to fit all departments.

After all, the VIR is formulated in a rather general and non-technical way. Even using [vBZ95], it is hard to deduce concrete criteria that can by applied to OAS. However, the notion of incident reporting may be useful. Also it is important to consider that evidence is needed for the government to trust external parties (like Oracle or an Internet Service Provider).

5.2 BS 7799 Standard

The BS 7799 standard was published in 1995 by BSI (the British Standard Institute) under the name 'Code of practice for information security management'. In 1999 it was republished as BS 7799-1 and it was complemented by BS 7799-2 'Specification for information security management systems'. In 2000, BS 7799-1 was adopted by ISO as ISO/IEC 17799 and in July 2007 it was renamed to ISO/IEC 27002[6]. BS 7799-3 was published in 2005 as 'Guidelines for information security risk management'. BS 7799-1 and BS 7799-2 were translated by the 'Nederlands Normalisatie-instituut' (NEN) to the Dutch language in 2000 as '*Code voor informatiebeveiliging*' (part 1), and '*Specificatie voor managementsystemen voor informatiebeveiliging*' (part 2).

The BS 7799-1 'Code of practice for information security management' is a collection of best practices in information security. It defines 'information security' as 'the preservation of confidentiality, integrity, and availability'.

A substantial part of [BSI99a] is addressed to organizational matters like security policies, security organization, personnel security, and asset classification and control.

[BSI99a] also stresses the importance of physical security. This is also relevant for this project: the whole collection of technical measures does not make much sense if everybody physically can access the computer system(s) on which OAS or the accompanying database is running.

Section 8.7.6 of [BSI99a] is devoted to the topic of 'publicly available systems'. It is explained that the security of such systems should be carefully controlled since everybody can access them. Here, the most important issues for this project are the protection of sensitive information when it is collected and when it is stored. Also it needs to be guaranteed that access to the system does not lead to unintended access to networks to which it is connected. These two requirements need to be taken into account when auditing OAS.

Regarding access control, [BSI99a] states that rules must be established on the premise 'everything must be generally forbidden unless expressly permitted' rather than on the weaker premise

[6] For organizations, it is possible to acquire an ISO/IEC 27002 certification.

'everything is generally permitted unless expressly forbidden'. This requirement is related to both confidentiality and integrity and can have consequences for the configuration of OAS.

Another confidentiality-related criterium from [BSI99a] is related to passwords. Section 9.2.3 mentions that passwords never should be stored on computer system in an unprotected form. This issue plays a role in both the Oracle Internet Directory and in the way the passwords for managing OAS are stored.

Section 9.7 of [BSI99a] is related to the information criterium 'auditability' as it addresses 'monitoring system access and use'. It is mentioned that logs of security-related events should include user IDs, dates and times for log-on and log-off, and records of system and resource access attempts.

In Section 10.2.1, [BSI99a] explains that data input to application systems should be validated. When auditing OAS in practice, the input validation of applications is one of the most important aspects. According to [BSI99a], checks should be applied to the input of business transactions, standing data, and parameter tables. Dual input or other input checks have to be considered to detect out-of-range values, invalid characters in data fields, missing or incomplete data, exceeding upper and lower data volume limits, and unauthorized or inconsistent control data.

The remark in Section 10.4.1 of [BSI99a] that says that, if possible, operational systems should only hold executable code advocates the usage of servlets instead of JSPs. JSPs are namely stored in OAS in uncompiled form where servlets are only stored as executables. Obviously, the most important reason for this remark is the confidentiality-criterium: having the source code stored on a server increases the risk that the source code is exposed so that hackers may obtain knowledge about confidential details of an application.

[BSI99b] explains how to implement information security management systems. This part of the standard can be used as a basis for a formal certification scheme. However, it partially overlaps with [BSI99a] and it has no extra practical relevance for this project.

5.3 CobiT

CobiT (standing for 'Control objectives for information and related Technology') is a collection of control objectives, management guidelines and maturity models. Control objectives are defined as 'high-level requirements to be considered by management for effective control of each IT process'. Management guidelines are given by showing concrete inputs, outputs, activities, goals, and measures per audit objective. Maturity models are used to benchmark business IT processes against audit objectives. Each (part of a) process can be rated from a maturity level of non-existent (0) to optimized (5). In [ITG07], 34 different IT processes are identified (examples of such processes are 'manage service desk and incidents' and 'assess and manage IT risks').

CobiT was published by ISACA in association with the IT Governance Institute (ITGI). I have access to version 4.1[7] of CobiT [ITG07]. This document was already used as a source for three of the information criteria in Section 3.3.

[ITG07] is subdivided into four domains: Plan and Organize (see Section 5.3.1), Acquire and Implement (see Section 5.3.2), Deliver and Support (see Section 5.3.3), and Monitor and Evaluate (see Section 5.3.4).

[7] Which is the most recent version at the time of writing (1/6/2007).

5.3.1 Plan and Organize

This part of CobiT is subdivided into ten processes. Each of these processes has its dedicated control objectives. Only those processes and control objectives that are relevant for this project are mentioned here:

Defining the Information Architecture is a process that comes with four audit objectives. At least two of these audit objectives are relevant. The first relevant criterium says that an enterprise-wide **classification scheme** should be established that is based on the criticality and sensitivity of enterprise data. The second relevant criterium stresses the importance of **integrity management**, meaning that procedures to ensure the integrity of electronic data should be defined and implemented.

Assessing and Managing IT Risks is needed to be able to make a well-considered decision about how to reduce the expected impact of IT risks. The audit objective **establishment of risk context** has relevance for this project since it is closely related tho the dependency analysis from the VIR (See Section 5.1). Also the identification of threats and their impact, here called **event identification**, is important. Clearly, this audit objective is related to the 'vulnerability analysis' from the VIR.

5.3.2 Acquire and Implement

Acquiring and Maintaining Application Software covers the design and configuration of applications. **Application Control and Auditability** asks for the implementation of business controls so that the 'reliability' of applications can be ensured. Since OAS is an application itself, this also applies on the concept of OAS.

Acquiring and Maintaining Technology Infrastructure comprises **Infrastructure Resource Protection and Availability**, meaning that control, security and auditability measures must be implemented to ensure availability and integrity.

5.3.3 Deliver and Support

Managing Third-party Services includes **Supplier Risk Management**, which is relevant for this project when a case is considered in which (components of) OASs are hosted by external hosting providers.

Ensuring Continuous Service is an important aspect from availability perspective. [ITG07] stresses the importance of having, maintaining, testing, and distributing an **IT Continuity Plan** that describes the way in which preventive and corrective measures are taken to reduce the impact of disruptions on key business functions and processes. Also **Offsite Backup Storage** of critical data is demanded by [ITG07].

Ensuring Systems Security definitely is worth closer examination: [ITG07] mentions that it is necessary to have an **IT Security Plan** in which business, risk and compliance requirements are addressed. Also the relevancy of **Identity Management** and **User Account Management** is explained. Hereby is is stressed how vital is is to implement authentication and authorization while being able to uniquely identify users and their activities on IT systems. In addition to this, [ITG07] notes that **Security Testing, Surveillance and Monitoring** is needed

to ensure the security of IT systems. Two other audit objectives that are relevant for this project are **Network Security** and the **Exchange of Sensitive Data** that can be achieved by techniques like firewalls, intrusion detection and network segmentation.

Manage the Physical Environment is an aspect that not may be overlooked when auditing an IT system. The restriction of **Physical Access** to, and the **Protection Against Environmental Factors** of IT systems are important objectives that touch on all aspects of the 'reliability' of IT systems.

5.3.4 Monitor and Evaluate

Monitoring and Evaluating Internal Control is an important aspect from both the financial and the IT auditor's perspective. Related audit objectives are for example **Assurance of Internal Control**, **Internal Control at Third Parties**, and **Remedial Actions**.

Ensure Compliance With External Requirements is also relevant in this project since governmental organizations have to comply with legal, regulatory and contractual requirements. Therefore, **Identification and Evaluation of Compliance With External Requirements** is important.

5.3.5 Remarks on CobiT

The CobiT framework can be useful for both (IT) managers and IT auditors. By providing a process description, a set of control objectives, management guidelines, and a maturity model per domain, this framework can be of great help by covering the most important aspects of the security of an IT system. However, CobiT does not solve all problems: both (IT) mangers and IT auditors need to adapt the guidelines from CobiT to concrete business situations and they need to consider that CobiT does cover all relevant aspects of the security of the IT systems that are suspect to an IT audit.

5.4 ITIL

ITIL[8] stands for Information Technology Infrastructure Library and is a collection of best practices for IT service management within organizations. This collection can be subdivided into two different categories[9]:

IT Service Management advices about service delivery and service support. *Service delivery* concerns relations between an organization and its service suppliers. For example, some general remarks are made about the Service Level Agreements (SLAs) that contain agreements with suppliers of services. Regarding *service support*, the most well known extension is the *service desk*. ITIL gives some general recommendations about the arrangement and the procedures of such suppliers of support.

Operational Guidance includes ICT infrastructure management, security management, business perspective, application management, and software asset management. In spite of

[8] [KM96] and [BP02] were used as sources for this section.

[9] This holds for version 2 of ITIL, on 30 June 2007 version 3 was released, but I have found too little information on that version to describe it here.

being an *operational* guidance, ITIL is still focussed on non-technical and procedural aspects.

ITIL is aimed at describing the way in which IT management processes can be planned, introduced, and evaluated. Therefore, the scope of ITIL is much broader than information security only. In fact, ITIL mainly is a procedural matter.

For instance: Patching procedures are part of the 'Operational Guidance'. The chapter 'Release Management' of that part describes that five steps need to be taken when a IT system is patched:

1. Build and configure
2. Test and accept
3. Schedule and plan
4. Communicate and prepare
5. Distribute and install

So in this example, when auditors have to audit this process, they need to ask for a handbook in which the patch management procedure is laid down. Then they compare the steps in the handbook with the ITIL steps that are described above.

In fact, this can be of help, but it is more important that patches indeed are applied in a careful way, rather than stating detailed procedures.

5.5 Common Criteria

The ISO/IEC 15408 Common Criteria [ISO06] is an extensive framework for the design and the evaluation of computer security. Development of the standard was mainly done in 1993 and 1994. After that time Common Criteria has been subject to various (minor) updates resulting in different (sub)versions of this standard. Common Criteria is mainly based on ITSEC (Information Technology Security Evaluation Criteria), a set of computer security criteria that was published by the European Commission in 1991. Also Common Criteria is inspired by the Military American TSEC[10] (Trusted Computer System Evaluation Criteria) standard and its Canadian counterpart CTCPEC (Canadian Trusted Computer Product Evaluation Criteria).

In [ISO06] a product under test is called a *Target of Evaluation* (TOE). Key concept in the evaluation process is the *Protection Profile* which is basically a set of security requirements. These requirements have to be of such independent nature that they can be used for different products (of the same class) that are subject to testing. The security requirements are commonly divided into functional requirements, assumptions, threats, objectives and policies. This division is also used for the development of the *Security Target*, a document in which the *Protection Profile* is refined to a particular TOE. The components in both the *Protection Profile* and the *Security Target* are identified in a structured way that can be best explained by providing an example. The 'security target' document [SG05] regarding the certification of Oracle Application Server 10g is used for this purpose.

[10] Also known as 'the Orange Book'.

FAU_GEN.2.1 *"The TSF*[11] *shall be able to associate each auditable event with the identity of the user that caused the event."* Is a functional requirement (**F**) that is related to auditing (**AU**) Data Generation (**GEN**).

A.PHYSICAL *"The processing resources of the TOE and the underlying system are located within controlled access facilities which prevents unauthorized physical access by outsiders, system users and TOE users."* Is an assumption (**A**) regarding physical (**PHYSICAL**) aspects.

T.OPERATE *"Insecure Operation. Compromise of the directory or an application hosted by OC4J may occur because of improper configuration, administration, and/or operation of the composite system."* Is an operational (**OPERATE**) related threat (**T**).

O.I&A.TOE *"The TOE must provide the means of identifying and authenticating users of the TOE."* Is an objective (**O**) that is related to identifying (**I**) and authenticating (**A**) within the context of the **TOE**.

P.ACCESS Is a policy (**P**) (verbatim text omitted here) that states how **ACCESS** to directory objects should be determined.

The mapping between threats and objectives is included as the rationale of the security requirements. In addition to these security requirements and their rationale also an *Evaluation Assurance Level* (EAL) is attributed to each *Protection Profile*. This EAL expresses the profile's strictness and can in theory vary between EAL1 (not strict) and EAL7 (very rigorous). In practice, however, a Protection Profile is almost never given a classification higher than EAL4+.

Common Criteria certifications can only be provided by a small number of selected organizations. The 'Netherlands National Communications Security Agency' (NLNCSA), which is part of the Ministry of the Interior and Kingdom Relations, does represent the Common Criteria in The Netherlands. Brightsight[12], formerly TNO-ITSEF BV, is the only official 'Common Criteria laboratory' in The Netherlands. Official laboratories are registered to evaluate the security claims that are made by the vendor of a product under certification.

The common criteria are rather complex and very extensive. This makes common criteria certifications hard to comprehend. However, the common criteria can for instance be useful in this project to reduce the chance of overlooking relevant threats.

5.6 Comparison and Remarks

So, the previous sections have described five different 'general audit objectives'. **VIR** is mainly written from an organizational point of view: line manageress are made responsible for the security of the information that is under their control. At the same time, the methodology of a so called dependency and vulnerability analysis (which an embedded risk analysis) is a central topic.

CobiT is directed to IT governance and control. It is written form auditor's perspective, but covers much more topics than information security only. In fact, CobiT partially uses the **BS7799** standard, which is a 'best practice' for information security, for addressing this topic. In addition, also elements from **ITIL**, which consists of best practices for IT service management, are used to cover the topics of 'Delivery and Support' and Monitoring. In fact, also **ITIL** and **BS7799** do

[11] TSF standing for TOE Security Functionality, that is the set of all hardware, firmware and software.
[12] See http://tinyurl.com/3342an, accessed on 15/8/2007.

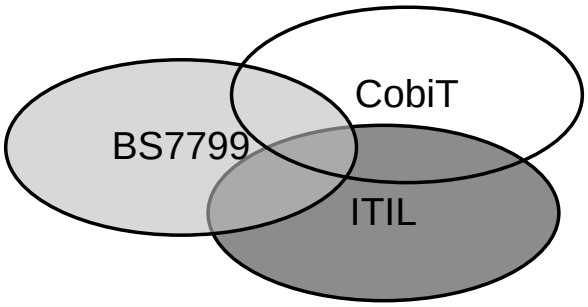

Figure 5.1: Relation of CobiT, BS7799, and ITIL.

overlap each other since ITIL 'Security Management' is based on this standard. The relation between CobiT, BS7799, and ITIL is depicted in Figure 5.1.

Common Criteria is a standard of a different nature, since it is directed to the evaluation of *products* and *systems*. An important drawback of Common Criteria is the fact that suppliers of products and systems can create their own *Security Target* that is used to evaluate their products or systems. Further, Common Criteria certifications are bound to strict limiting conditions: for systems not complying with these conditions, the certification does not hold. So, after all, a Common Criteria certification of a product does not guarantee the security of that specific product.

The standards that are described in the previous sections can be of of help when an IT Auditor has to *evaluate* a system. Evaluation is defined by [And01] as "*our estimate of the likelihood that a system will not fail in some particular way.*". Using for instance Common Criteria can help in carrying out such an estimation by addressing threats that might be overlooked otherwise. On the other hand, by sticking to general audit objectives only, there exist the risk of getting a blinkered attention for relevant aspects outside the scope of these standards.

In this context, it is good to realize that perfect security of a (complex, interconnected) system is not feasible. It might even be the case that a system that is carefully investigated and approved by an IT Auditor today is hacked tomorrow, for instance by an attacker who uses a zero-day exploit[13] that was found this very night.

When a financial auditor passes a favorable judgement on the balance and income sheets, it is still possible that fraud had been perpetrated. Still, the necessity of performing financial audits is generally accepted because of its practical relevance. This also holds for technical IT audits: although perfect information is not available and perfect security is be unfeasible, the evaluation of critical systems is still necessary and useful.

However, these considerations give reason to be careful in making statements about the security of IT systems. From that perspective, the relevance of literally interpreting general audit objectives like Common Criteria may be questionable. Thus it may be advisable to use the objectives more as guidelines than to apply them literally. In order to give a really well-founded judgment of a system, a thorough study of all relevant aspects is needed. Checking off lists can help is this process, but is generally insufficient in *assuring*[14] the security of a system.

[13] A zero-day exploit takes advantage of a vulnerability before a patch is released.

[14] [And01] defines *assurance* as "*the estimate of the likelihood that a system will not fail in some particular way*".

5.7 Other standards and Laws

COSO (Committee of Sponsoring Organizations of the Treadway Commission) is an initiative by enterprises from the United States. In 1994 this committee published report, called "Internal Control - Integrated Framework.". The model presented in this report is commonly known as the COSO-model or the COSO-framework. The COSO-model is addressed to *internal control*, which comprises all activities in an organization to guarantee efficiency of business, reliability of financial reporting, and compliance with relevant laws. By means of this, COSO sets a general framework for internal control. The CobiT standard (see Section 5.3) can be seen as a translation of the general COSO-ideas to an IT environment [Loo97].

The principles that are denoted in the COSO-model and in CobiT grew in attention when the **Sarbanes-Oxley (SOx)** act was signed into law by President Bush on 30 July 2002. This law was intended to discourage misbehave of both management and auditors by making them severally liable for fraud and organizational misbehave. A number of remarkable corporate and accounting scandals (e.g. the Enron scandal[15]) has significantly contributed to the adoption of SOx. In SOx –especially in Section 404–, internal control is seen as an important measure against fraud and companies' misbehavior. Both COSO and CobiT are seen as good starting points of complying with SOx. Since SOx is internationally seen as an important law by both governments and investors, many companies are stimulated or even enforced to comply with this law. As a result, the popularity of COSO and CobiT has also grown since the adoption of SOx.

On 1 September 2002, the Netherlands' **Privacy law** ('*Wet bescherming persoonsgegevens*', or simply Wbp) came into force. Section 13 of this law demands appropriate technical and organizational measures to prevent personal data from being lost or misused. This law touches on all corporate and governmental organizations that store or process personal data.

[15] See for instance http://tinyurl.com/nzdax, accessed on 15/8/2007

Chapter 6

Inherent Vulnerabilities

OAS has some *vulnerabilities* which are inherent to its architecture, and to the various components that it contains[1]. Within the context of this thesis, an *inherent vulnerability* of OAS is defined as *a set of conditions regarding the design and/or the default implementation of OAS that may threat its reliability*. Here, EAP's definition of reliability is used, meaning that the information security criteria –called *information criteria* in this context– are satisfied[2]. The word 'threat' is used in the broadest sense of the word: not only attacks from the in- and outside are examined, but also issues like natural disasters are taken into account. The overview of inherent vulnerabilities and remedies is used as a first step in the development of specific audit objectives and an audit program.

In this chapter, the notion of an *adversary* is used. Such an adversary represents an attacker or a coalition of attackers. Within the context of a specific OAS installation, these attackers try to exploit vulnerabilities in OAS to perform actions that lead to violation of the information criteria. An adversary can be either an outsider or an insider. Outsider adversaries can for instance be persons that try to deface a website or bring a server down just for (their own) fun. Just as well they can be professional hackers that attack a system for fun or for profit (by carrying out spying assignments). Insider attacks can also be a manifestation of espionage or other malicious behavior. On the other hand, attacks can also be performed by employees who are stuck with limited system privileges or functionality as they simply are trying to do their job. The possibility of insider attacks is often ignored, but these attacks constitute a significant part of all attacks [PF04].

6.1 Network Communication

Network communication plays a central role within the OAS architecture. Figure 6.1 shows the basic network communication that takes place within this architecture. All OAS components that are depicted in this picture can namely be installed on different physical computers that are connected to each other via a TCP/IP[3] network. This TCP/IP network can be used to commu-

[1] See Chapter 4 for a description of the architecture and the key components of OAS.

[2] EAP's information criteria are confidentiality, integrity, availability, and auditability, see Section 3.3.

[3] TCP/IP is a collection of network protocols including the Transmission Control Protocol (TCP) and the Internet Protocol (IP). The Internet is for instance based on this TCP/IP protocol stack.

nicate via various *application protocols*. As depicted in Figure 6.1, the LDAP[4] protocol is used to communicate with the Internet Directory whereas the Database and the Metadata Repository are approached via the proprietary Oracle NET protocol [Pol04]. The Web Cache, the HTTP Server, and the Certificate Authority all use the HTTP protocol and the HTTP Server communicates with the (Java) application component using the Apache JServ Protocol[5] (AJP).

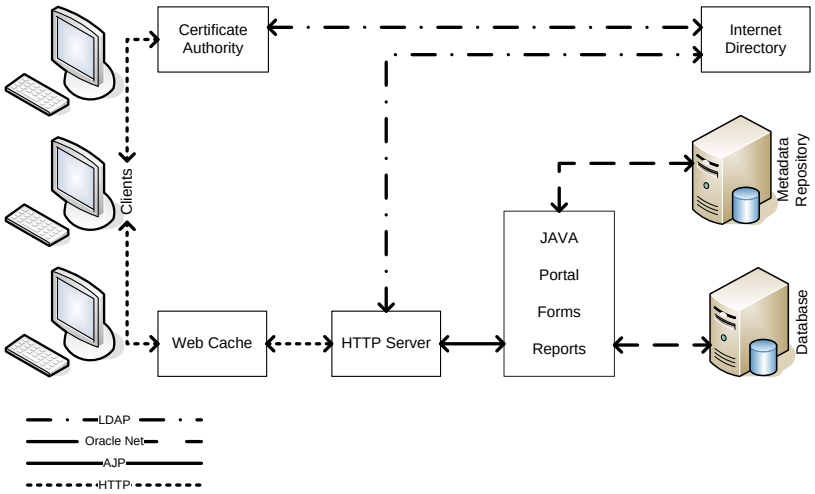

Figure 6.1: Simplified overview of OAS communication

Suppose that the architecture that is depicted in Figure 6.1 is implemented in a computer network of an organization. Recall that internal clients are logged on to the organization's local area network (LAN) and that the external clients connect to computers in this LAN via the Internet. All communication between the clients and the OAS takes place via a web interface that is implemented via either a website or a Web Service (see Chapter 2). So, all communication between the client and the OAS takes place via the HTTP protocol. By itself, the HTTP protocol is not secure: the messages that are transferred using this protocol are not encrypted and even their authenticity cannot be guaranteed. By default, this also holds for the communication between the various OAS components.

This is an inherent vulnerability of the OAS architecture: an adversary who can intercept the traffic that uses these protocols can easily eavesdrop on the communication. Even worse, an adversary can modify communication and can inject messages into it. Therefore, this attack violates the confidentiality criterium in the first place. The vulnerability of having unsecured communication channels can also lead to the violation of the other information criteria since an adversary can modify the network communication. Moreover, he possibly can extract and use login details that are sent over the network.

An adversary that operates from the inside of an organization's network can for instance reroute network traffic via its own computer using a techniques like ARP spoofing or IP spoofing. In fact, the adversary can in this way impersonate an honest user and intercept all network traffic to and from this honest user. For an adversary from the outside it is more difficult to intercept network traffic. In fact, this is only possible when the adversary can control a computer that is located in the internal network. In addition to this, it is possible for an adversary to intercept the

[4] See Section 4.2.2.

[5] See http://tinyurl.com/2jur5q, accessed on 14/12/2007

network communication between OAS and an honest external client when the network traffic of that external client is (re-)routed via a computer (network) that is under control of this adversary.

Basically, two things can be done to mitigate vulnerabilities that are inherent to the network communication with and within OAS. In the first place, all communication can be encrypted. This can be achieved by using SSL (on HTTPS) which enables a secured version of the HTTP, the LDAP, the AJP, and the Oracle NET protocol. The Oracle Certificate Authority plays a central role in the setup of secured communication. More information on this can be found in Section 4.2.3 and Section 6.7.3. Using this secured communication it is in principle still possible for an adversary to re-route the traffic via its own computer, but then it is generally very difficult for him to decrypt the communication. By default SSL is not enabled in OAS because it requires the setup of a certificate structure. Moreover, bringing SSL into action has a negative effect on the performance of the OAS architecture since it involves extra calculations and network traffic. So, in this case, there is a tradeoff between performance and security. As far as systems on the client side are concerned, HTTPS is hardly slower than HTTP (talking about 'normal' text based traffic, not about large file transfers). On the server side, however, implementing SSL can noticeably deteriorate the performance of the system [GBS98]. An HTTP server, for instance, must be able to communicate with a group of users simultaneously. Since the HTTP messages have to be encrypted and decrypted for each user individually, the HTTP server has to perform many more calculations. The more data has to be transferred, the heavier the calculations that need to be performed. This does not only hold for the HTTP server, but also for the other components in the OAS architecture. Obviously, putting fast modern hardware intro action will help to retain performance. Furthermore, OAS's performance can also be retained by clustering hardware components (See Section 4.3.1), or by implementing dedicated hardware devices that facilitate SSL acceleration [PH05].

Segmentation the network is the second thing that can be done to suppress the vulnerabilities that are inherent to the network communication architecture. This measure is advised by [PH05] as a 'best practice' in securing the OAS architecture. Basically, network segmentation is achieved by implementing firewalls. A firewall is device that blocks all network traffic except for predefined permitted communication methods. In this way, a firewall can for instance be implemented to restrict all network traffic to OAS that comes from the Internet, except for HTTP traffic that is addressed to the OAS Web Cache. This example is a good starting point for setting up a more secure architecture. Firewalls, however, do only block certain types of network traffic. So, the content of allowed network traffic is not examined by the firewall. This means that an adversary still may succeed in breaking in to a computer by only using legitimate network traffic. An adversary could for instance hack the web cache via HTTP by sending a special URL that causes a buffer overflow[6]. Therefore, it is necessary to make all the components as secure as possible, besides the segmentation of the network. The impact of such attacks can also be limited by implementing extra firewalls so that the network gets even more segmented. As a starting point, according to [PH05], it is advisable to putt the Metadata Repository, the J2EE server[7], and the Database with business data behind an extra firewall. The resulting architecture is depicted in Figure 6.2.

A firewall can be either a physical network device or a software package that is installed on a computer. Generally, hardware based firewalls offer better performance and greater security.

[6] See section 6.6.2 for more details about these kinds of attacks.

[7] In the simplified representation in Figure 6.2, the component with Java, Portal, Forms, and Reports can be seen as the J2EE server.

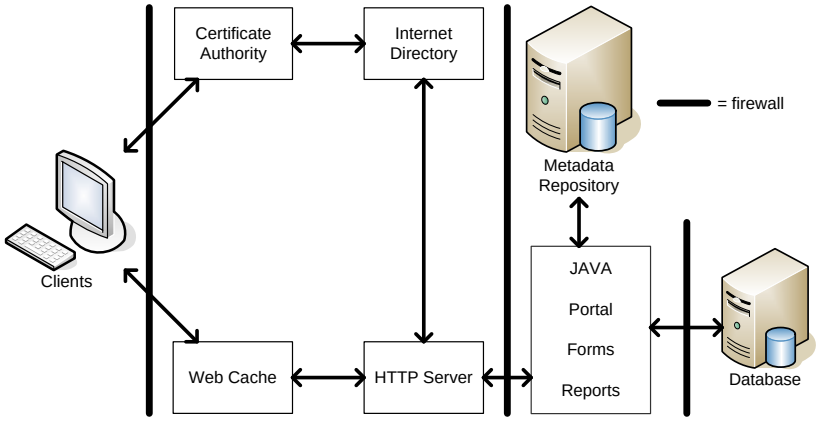

Figure 6.2: Basic architecture of OAS with firewalls

Especially when using software based firewalls that are installed on a computer that also runs OAS components, a performance penalty can be expected. However, a firewall in its basic form blocks certain types of network traffic by only allowing traffic to certain ports while blocking traffic to all other ports. Since the contents of the data packages are not examined, these firewalls hardly deteriorate the performance of the architecture. As a drawback, the implementation of a firewall in fact also introduces an extra vulnerability since also firewalls can get compromised.

So, the basic implementation of a firewall does only restrict certain types of network traffic. It is, however, possible to implement products that actually examine the contents of network traffic. Basically, two types of these products can be distinguished: an *Intrusion Detection System* (IDS) only monitors the content of network traffic and detects suspicious messages. An *Intrusion Prevention System* (IPS) is more of a reactive nature since it actually responds to suspicious content, for instance by blocking traffic. In this chapter, IPSs are mentioned several times. An IPS can basically be of two types: a host based IPS (HIPS) is resident on the computer that needs to be protected, whereas a network based IPS (NIPS) is implemented on dedicated hardware that acts as a proxy for network traffic. An important advantage of a NIPS is that multiple computers can be protected by one IPS. However, in contrast with a HISP, a NISP can not inspect (SSL) encrypted traffic. The implementation of either a NISPs or a HISPs can deteriorate the performance of the OAS architecture since package inspection is a relatively time-consuming task.

Additionally, if ports are not blocked by a firewall, then changing default port numbers can help to mitigate the risk of successful 'fingerprinting' operations. These operations are used by attackers in order to determine OSs and software packages that are installed on target computers.

In Figure 6.2, the distinction between internal and external clients is not made. The reason for this is that it is also advisable for internal users to use the architecture as depicted in this figure: also people within the organization should be restricted in having access to the various components because adversaries may be found within the organization and the risk of finding an exploitable vulnerability in a component can be reduced by prohibiting unrestricted access to that component.

It is a good idea to use separate OASs for *internal* access (via the intranet) and *external* access (via the Internet). The reason is that external users normally need less functionality than internal users, so the OAS for external access can be more hardened[8]. Moreover, this separation idea is

[8]Hardening of a server means that it is made more robust and secure by disabling (unused) elements and func-

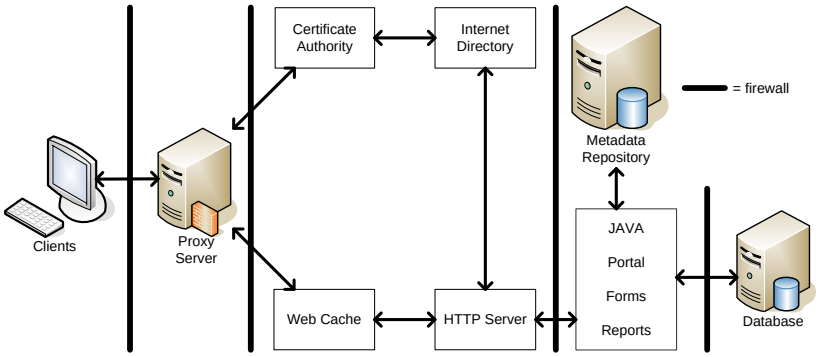

Figure 6.3: Basic architecture of OAS with firewalls and a proxy server

also advisable for availability reasons so that the 'internal' OAS may still work after the 'external' one went down (and vice versa).

A conversation with a database administrator of one of the departments of the Netherlands government made clear that the need of having a dedicated OAS architecture for external access is understood in practice. The administrator told that they wanted to restrict access via the Internet to a certain group of users. To implement this, they decided to install a so called *proxy server* that requires users to log on even before they can connect to the OAS Web Cache (see Figure 6.3). Other ideas of restricting external access include the usage of Virtual Private Networks[9] or Server Based Computing[10]. Obviously, these implementations undermine the idea of single sign-on. In fact, in the specific case at the above-mentioned department, both single sign-on and OID itself were not used at all.

Inherent vulnerabilities of the network communication within the OAS architecture	**Touches on**	**Remedies to mitigate these vulnerabilities**
Interceptable, readable network traffic that can be modified or spoofed	C I A M	Use SSL when applicable
Unrestricted network access to vulnerable parts of OAS components	C I A M	• Segment the network • Make components as secure as possible • Implement an IDS and/or an IPS • Use SBC, a VPN, and/or a proxy to restrict access

Table 6.1: Summary of inherent OAS network architecture vulnerabilities, the information criteria Confidentiality (C), Integrity (I), Availability (A), and auditability (M) vulnerabilities where these vulnerabilities touch on, and the remedies to mitigate these vulnerabilities.

Implementing SSL (with client side certificates) is an important step in securing the network.

tionalities.

[9] A VPN enables encrypted communication with the intranet for authenticated users

[10] The most well know example of SBC is the Windows Terminal Server which provides authenticated (remote) users with encrypted access to a virtual remote desktop environment on the intranet.

Also hardening components and operating systems is an essential step. In fact, these two steps should be enough, but also network segmentation is absolutely necessary because of the large amounts of (revealed) security flaws in network protocols, operating systems, OAS components and other applications. This typically is a strategy of 'defense in depth' since a collection of measures is used to delay and impede attacks, rather than preventing them by a single and strong line of defense.

6.2 Physical Matters

This subsection briefly discusses some physical matters regarding the reliability of OAS. Basically two types of physical matters can be distinguished: physical attacks, and environmental calamities.

Physical attacks can take place when an adversary has physical access to a component that is part of the OAS architecture, or that is related to it. An adversary who has physical access to a computer on which components of OAS are installed could for instance shut this computer down, disconnect it from the network, add a hardware key logger, or even steal a hard disk. Other important issues in this context are power supply and network infrastructure: cutting of the power or bringing down the network will have big consequences for the OAS's availability. Plugging into the internal organization's network furtively (whether wired or wireless) can also be considered as a dangerous physical attack. The same holds for destroying physical components e.g. by arson.

The most important remedy to mitigate these physical attacks is the implementation of a stringent physical access management which goes beyond the scope of this thesis. For suppressing the threat of unauthorized network access it is a good starting point to completely disable wireless network access. Additionally, it is necessary to compose a security plan that contains reactive measures. This means that preparations are made for the case in which components of OAS are destroyed. For critical systems that need high availability, it is necessary to replicate the system on a physically different location so that this back up system can take over the original system if that is brought down.

Another important issue that is related to physical matters is the issue of security awareness. This has to do with the way in which users and administrators of OAS behave: even a system that is perfectly secure from a technical perspective can be vulnerable when users and administrators are not security aware. Choosing easy passwords is an example of such non-security aware behavior. This can be mitigated by setting password policies. However, preventing users from writing down passwords on Post-it notes, or making them aware of *social engineering attacks*[11] cannot be achieved by technical rules and measures. Therefore, training and monitoring of users is needed to supplement the available technical measures.

Environmental calamities are physical matters of a different nature. All physical computers are in principle not resistant to fire, flood, and power failure. This also hold for computers on which components of OAS are installed. Creating and securing backups[12] and taking care of the remote replication[13] of components are the most important remedies to these vulnerabilities.

[11] An example of a social engineering attack is a attacker who impersonates an administrator and contacts users with the purpose of revealing sensitive information like usernames and passwords.

[12] See Section 4.3.2.

[13] See Section 4.3.1.

Inherent physical vulnerabilities of the OAS architecture	Touches on	Remedies to mitigate these vulnerabilities
OAS can be brought down by attacks or calamities that cause damage on computers on which components of OAS are installed. The architecture is also vulnerable for disconnection from power or from the network	A	Implement tight physical access management, create backups, and replicate vital components
Sensitive information can be revealed by stealing hardware or for instance by adding hardware key loggers	C	Implement tight physical access management
Sensitive information can be revealed after gaining unauthorized network access	C I A M	Implement tight physical access management and disable wireless network access
Sensitive information can be revealed after performing social engineering attacks or by thoughtless actions of users and administrators	C I A M	Train users and administrators to increase their security awareness

Table 6.2: Summary of inherent physical OAS architecture vulnerabilities, the information criteria Confidentiality (C), Integrity (I), Availability (A), and auditability (M) vulnerabilities where these vulnerabilities touch on, and the remedies to mitigate these vulnerabilities.

6.3 Operating Systems

All components in OAS's architecture need to be installed on top of an Operating System (OS). According to Oracle[14], OAS's components are certified to be installed on various versions of Sun Solaris, Microsoft Windows, IBM AIX, HP-UX and Linux (including Red Hat, SuSE Enterprise, and Oracle Enterprise Linux). Each of these operating system has its own inherent vulnerabilities. Therefore, evaluating the security of the used OSs is needed when auditing OAS. The topic of OSs' security is far too broad to discuss in this thesis. Obviously, an adversary who can control on OS can also control all applications that run on top of that OS. Control to an OS can not only be obtained via vulnerabilities in (the configuration of) that OS, but also via vulnerabilities in applications that are installed on this OS. Therefore, it is advisable to harden OSs by removing all unneeded features and programs. Further, good patch management is essential: patches should be evaluated and (if necessary) installed as soon as possible after their release. Obviously, OS security is also related to network security. Therefore, the implementation of firewalls, IDSs, and IPSs can mitigate the threats to OSs.

6.4 Patch Management

Section 6.3 already pointed to the issue of patching. This topic is relevant for (almost) any software product, thus also for the components of OAS. A patch is a piece of software that is designed to modify an already installed software product. Generally, patches for a software product are released by the supplier that created that specific software product. Patches can serve many purposes, the most important ones are fixing bugs and solving vulnerabilities.

For an organization it is essential to devise a well-considered patch management plan. Patches are namely essential in fixing bugs and vulnerabilities: a missing patch can result in the violation

[14] See http://tinyurl.com/3bna2x, accessed on 23/10/2007.

of one or more information (security) criteria. On the other hand, installing patches can also have unwanted side effects: for instance applying a patch to OAS can result in J2EE applications that no longer work.

As a starting point for such a programme for patch management, it is important that the organization has documented the details of the complete IT infrastructure including all software (including operating systems, middleware and applications). By means of these details, administrators can decide which patches need to be installed. In the case of OAS for instance, a certain patch may only be relevant for Oracle Enterprise Manager on Windows. So if the Enterprise Manager is not installed, or it is installed but not on the Windows Platform, then the installation of this patch is not required. This is important since the installation of a patch can result in system downtime. This downtime can be of temporarily nature as the result of a mandatory reboot of the system, but it can also be more problematic as certain applications do no longer work (correctly) after the implementation of a patch[15]. In practice, this is a big issue. When auditing IT systems, auditors from EAP sometimes found out that critical patches are missing. When confronting administrators with this finding they often respond that applying applying the patches will result in a breakdown of an application.

Exactly for this reason, it is not only important to be careful in choosing which patches to install, but also in applying the patches that are required. According to what is advised (for change management procedures) in [ITG07], [KM96], and [BSI99a], organizations should examine patches in a test environment. Therefore, organizations should have both a test and a production environment.

Oracle brings out its patches on a quarterly basis. Each Tuesday closest the to 15th day of January, April, July, and October, a bundle of patches is released. This bundle is called a Critical Patch Update (CPU). Notice that these CPUs only address Oracle products. Each CPU is accompanied by risk matrices in which the found and fixed vulnerabilities per product are depicted. The severity of each fixed vulnerability is assessed and quantified using the Common Vulnerability Scoring System (CVSS). This framework can be used to decompose the properties of a vulnerability in three categories: *base* (intrinsic and constant), *temporal* (changes over time but not among user environments), and *environmental* (unique to specific environment and design) [MSR06]. This is quite a change to the earlier policy of Oracle since they used to keep all information about security issues and patches confidential. When that was the case, administrators could not decide which patches they needed to install and which could be left aside.

6.5 Database

Auditing Oracle Databases is a separate issue. It will be discussed only briefly.

The Oracle Database has several (inherent) vulnerabilities. Publications by security experts like David Litchfield[16], Pete Finnigan[17], Aaron Newman[18], and Alex Kornbrust[19] have made that perfectly clear.

The Listener service is known to be a vulnerable part of the Oracle Database system. This service acts as a proxy by which a client can connect to the database. In earlier versions of Or-

[15] A database administrator of one of the departments of the Netherlands government explained that they experienced this exact problem after implementing a patch which even turned out to be unnecessary.

[16] See http://www.ngssoftware.com

[17] See http://www.petefinnigan.com

[18] See http://www.appsecinc.com

[19] See http://www.red-database-security.com

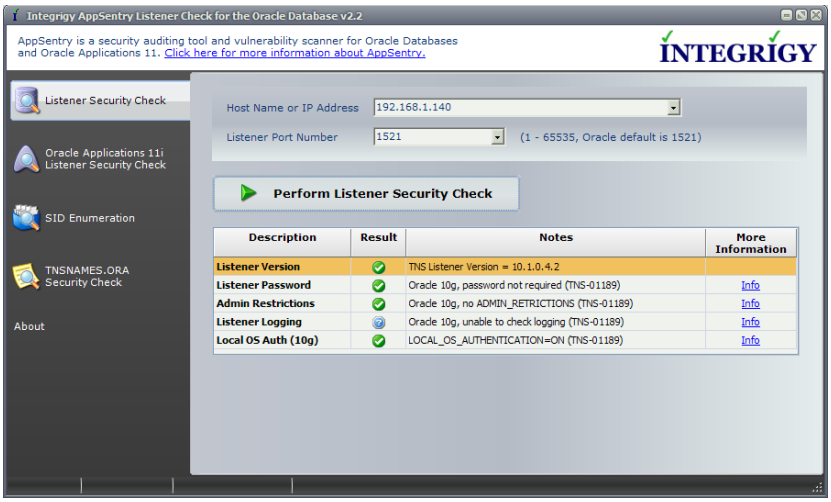

Figure 6.4: Screenshot of a tool that checks the Listener for vulnerabilities

acle Database, the listener was not protected by default. As a result, it could easily be tampered with remotely: by changing the value of the variable holding the location where the log file is written, full administrator access to the database could be obtained relatively easily[20]. To mitigate this vulnerability the listener could be protected with a password. In recent versions of Oracle Database, remote access to the Listener service can only be obtained after local authentication. However, disclosures of Listener related security issues still go around[21]. Also a vulnerability was discovered by which an adversary could crash the Listener Server by sending a specially crafted request[22]. Several tools have been developed to scan the Listener for vulnerabilities. Figure 6.4 shows a screenshot of one of these tools.

An Oracle Database contains many users accounts by default. In principle, each default user account does have a default password. If these default user accounts are not locked or disabled, this can be a dangerous vulnerability. There are programs and scripts[23] available that can check for database access using default accounts and passwords. Of course, each default account should be either provided with non-default password, or should be disabled at all.

Changing default passwords and disabling default accounts does however not solve all problems with usernames and passwords. Wright and Cid reveal in [WC05] the way in which Oracle 'protects' users' passwords: each individual password is simply *hashed* before it is stored in the database or sent over the network. This hashing operation is performed by means of a proprietary one-way function that takes the username and the password as an input, and that outputs 8 bytes that represent the resultant password hash. The problem is that, at least for short passwords, this hashfunction is not very secure: Generally, offline dictionary attacks on hashes are deterred by using a secret value, that is called salt, as second input for the hash function. According to [WC05], Oracle uses a hashfunction that is unsecure since it is directly based on the username of the account which password is hashed. Nowadays, several tools[24] are available that can be used to recover plain text passwords from the hashed values. Therefore, access to password hashed should

[20] See http://tinyurl.com/37vgmq, accessed on 31/10/2007.
[21] See for instance http://tinyurl.com/36k5z, accessed on 31/10/2007.
[22] See http://tinyurl.com/2vxx9v, accessed on 31/10/2007.
[23] See for instance http://tinyurl.com/ynq2rq, accessed on 31/10/2007.
[24] See for instance http://tinyurl.com/28ytcl and http://tinyurl.com/53vmz, both accessed on 2/11/2007.

be restricted as much as possible by restricting users' privileges to the OS and the database and by encrypting network traffic. Also enforcing relatively long passwords and setting relatively short expiration dates can help in mitigate the risk of plain text passwords being recovered from hash values. (This may however annoy users, resulting in a decrease in security as they are incited to write down these long and hard-to-remember user names and passwords).

In this context, two other issues may not be left unmentioned. In the first place, database encryption can be used as an extra measure to mitigate attacks regarding the confidentiality of the information i the database. As a drawback, enabling database encryption can lead to a significant performance penalty, especially for searching operations.

In the second place, auditors are faced with the problems of an all-powerful database administrator: since this person can access the database with administrating privileges, he can view and modify literally every piece of information that is stored in it. For instance in the case of a database holding salary information, a manager could want to restrict the administrators privileges to view or modify information. Oracle has tried to solve this problem by developing a product called *Oracle Database Vault*[25] that was designed to curtail administrators' privileges. However, since administrators normally have full privileges at the OS level this problem, that is related to the classical issue that is expressed in the famous question "who will guard the guardians?"[26], actually remains unsolved.

Regarding the Oracle Database, network segmentation should tentatively be advised as the key solution to mitigate inherent vulnerabilities. Many vulnerabilities can be mitigated by only allowing database access via OAS. In this context it is advisable to use at least a dedicated schema user with minimal privileges for access via an OAS that can be approached via the Internet. In this way, the risk of SQL injections[27] can be mitigated.

6.6 Mid-Tier

As depicted in Figure 4.1, the mid-tier can be seen as the heart of OAS: the Web Cache, the HTTP server, and the environment where (custom) applications are executed are part of this mid-tier.

6.6.1 Oracle HTTP Server

As mentioned in Section 4.1.1, Oracle HTTP Server is in fact a modified version of the open source Apache HTTP Server. Over the years a large number of vulnerabilities[28] has been discovered in this product. The Apache HTTP Server is developed and maintained by the Apache Software Foundation.

The Apache Foundation releases security patches on an irregular basis, whereas Oracle only brings out updates quarterly (see Section 6.4). This implies that security flaws in the Oracle HTTP Server can remain unpatched for a relatively long time even after the official patch is released by the Apache Software Foundation. This problem can be mitigated by responding to vulnerability reports that appear on the website[29] of the Apache Software Foundation or other

[25] See http://tinyurl.com/ztvnc,accessedon19/11/2007

[26] As posed by Plato in the *Republic*. This issue is also captured in the famous Latin phrase "*quis custodiet ipsos custodes?*" from the *Satires of Juvenal*.

[27] See Section 6.6.3.

[28] See for instance http://tinyurl.com/ycuafx, accessed on 8/11/2007.

[29] http://tinyurl.com/ycuafx, accessed on 8/11/2007.

vulnerability-tracking websites[30].

The main possible attacks on Oracle HTTP Server are cross-site scripting (XSS), command injection, buffer overflows, and (D)DoS[31] attacks. The issues of XSS and command injection are discussed in Section 6.6.3 in more detail. A special form of command injection is directory traversal, an attack that exploits a vulnerability in the web server for accessing files and folders that should not be accessible via the HTTP Server. Basically this attack consists of requesting specially crafted URLs that cause the HTTP server to walk out of its default directory tree. In the early days, this could be performed by using URLs involving /../ or \..\, but more recent attacks involve URLs in which these commands are encoded in for instance UTF-8[32]. By means of this attack, the adversary can not only obtain sensitive information, but he can also execute code remotely.

On the Internet, various[33] (patched) buffer overflow vulnerabilities for Apache HTTP Server can be found, also several[34] (patched) weaknesses can be found that make the Oracle HTTP Server extra vulnerable for (D)DoS attacks. Applying the appropriate patches and disabling unneeded components are the most important countermeasures to perform. Again, also an IPS can be helpful here.

The core Apache HTTP Server can be extended by (installing and) enabling different *modules*. These extensions add certain functionality to the Apache HTTP Server. By default, the Oracle HTTP Server is supplied with several modules[35] that are developed and maintained by Oracle. To mitigate vulnerabilities as much as possible, all unused modules should be disabled.

Inherent vulnerabilities of the Oracle HTTP Server	**Touches on**	**Remedies to mitigate these vulnerabilities**
Vulnerabilities in Oracle HTTP Server may remain unpatched for a long time, even after the Apache Foundation has disclosed and patched these specific vulnerabilities	C I A M	Respond to Apache HTTP Server's vulnerability reports that appear on the Internet, disable all unneeded modules, implement an IPS

Table 6.3: Summary of inherent Oracle HTTP Server vulnerabilities, the information criteria Confidentiality (C), Integrity (I), Availability (A), and auditability (M) vulnerabilities where these vulnerabilities touch on, and the remedies to mitigate these vulnerabilities.

6.6.2 Web Cache

Implementing Oracle Web Cache[36] does relieve the Oracle HTTP Server. This Web Cache can however also introduce new vulnerabilities. Besides the difficult issue of (distributed) denial-of-service attacks[37], several vulnerabilities and exploits can be found on the Internet. Most of

[30] For instance http://cve.mitre.org (international) or http://www.govcert.nl (Dutch).

[31] These (Distributed) Denial of Service attacks are performed by sending such a massive amount of requests (from different computers) to the HTTP Server, so that it is no longer able to process legitimate requests. An IPS might mitigate this vulnerability, but in general it is very hard or even impossible to cover these attacks completely.

[32] For instance see http://tinyurl.com/28o2sp, accessed on 8/11/2007.

[33] See for instance http://tinyurl.com/zf48p and http://tinyurl.com/ynjra9, both accessed on 9/11/2007.

[34] See for instance http://tinyurl.com/yrn2y3 and http://tinyurl.com/2eholb, both accessed on 9/11/2007.

[35] See Section 4.1.1.

[36] See Section 4.1.2.

[37] In fact, implementing Oracle Web Cache can mitigate the overall impact of (D)DoS attacks since the Web Cache is more robust than the HTTP Server. However, Oracle Web Cache itself also is particularly not invulnerable for (D)DoS

these were reported and solved before the year 2006. Examples of these vulnerabilities include buffer overflows[38] that can be triggered by an adversary by sending overly long requests to the Oracle Web Cache. As a result of this, the adversary can write custom data into the memory of the computer running the Web Cache. This can result in a crash of the Oracle Web Cache, the disclosure of sensitive information, and even in remote execution of arbitrary code.

In addition, vulnerabilities were discovered in the web-based administrating interface of the Web Cache[39], this specific vulnerability could lead to integrity violation of files located on the computer on which the Web Cache is installed.

The vulnerabilities mentioned above are inherent to the design and the implementation of Oracle Web Cache. Patching the system, hardening the implementation (for instance by disabling the web-based administrating interface), and using an (up-to-date) IPS are in principle the only ways in which the exploitation risk of these vulnerabilities can be mitigated. An issue that is more related to the internals of a Web Cache in general is the issue of leakage of sensitive information. Oracle has equipped the Web Cache with a session binding mechanism [Ste06], however, the possibility that data is leaked to an other session-compartment can never be completely ruled out. According to recommendations of the National Security Agency, administrators should "*configure the OAS Web Cache with caching rules to restrict caching of sensitive information*" or "*if the OAS Web Cache is configured to cache both sensitive and non-sensitive content, regard all cached data as sensitive content stored in a high threat environment.*"[CKRW06].

Inherent vulnerabilities of the OAS Web Cache	Touches on	Remedies to mitigate these vulnerabilities
(D)DoS attacks on OAS Web Cache	A	Implement an IPS, cluster Web Cache servers, install OAS Web Cache on a dedicated server
Buffer overflows and other design/implementation flaws in OAS Web Cache or its administration interface	C I A M	Carefully patch OAS Web Cache, implement an IPS
Breaches in data compartmentalizing within OAS Web Cache	C	Restrict caching of sensitive information

Table 6.4: Summary of inherent OAS Web Cache vulnerabilities, the information criteria Confidentiality (C), Integrity (I), Availability (A), and auditability (M) vulnerabilities where these vulnerabilities touch on, and the remedies to mitigate these vulnerabilities.

6.6.3 Applications

Just as Database Security, application security is a topic that is worth a thesis on its own. Basically three kinds[40] of applications can be discerned in the context of OAS: Forms & Reports, Portal, and Java. OAS Forms & Reports and Portal applications are bound to the framework that Oracle has supplied. Java applications, however, can be used to write full custom applications (both websites and Web Services). Therefore, Java applications probably are worth the most attention from a security perspective.

attacks. The impact, however, can be limited by installing Oracle Web Cache on a dedicated (grid of) computer(s).

[38] See for instance http://tinyurl.com/29xlds, accessed on 5/11/2007.

[39] See for instance http://tinyurl.com/2grxaz, accessed on 5/11/2007.

[40] See Section 4.1.3.

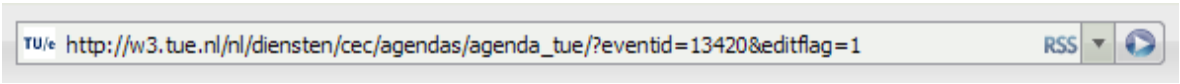

Figure 6.5: Example of an URL with parameters

Java

The most well known attacks for websites include parameter modification, cross-site scripting (XSS), and SQL injection. Parameter modification means that an adversary modifies or adds a parameter that is sent to the HTTP server. As a –very trivial– example, Figure 6.5 shows a address bar in a web browser. Obviously, the parameters eventid and editflag can be modified easily. In this specific case, it seems that nothing harmful can be performed by modifying these parameters. In other examples, however, modifying similar parameters can for instance lead to the disclosure of confidential information[41].

Cross-site scripting and SQL injection are in fact special instances of the parameter modification attack. XSS is performed by leaving a message containing a script that is executed when a user clicks on a link or visits the web page that contains that message. Therefore, pages like digital guestbooks and journals (where user data is presented on a web page) are often faced with this attack. XSS can also be performed when a user clicks on a specially crafted URL that exploits a vulnerability in the target website. By means of XSS malicious scripts can be executed on the users' computer. This may even result in hacked user sessions when *cookies*[42] can be stolen. By stealing these cookies by means of crafted client-side Java code, an adversary can in principle take over a user's session which can result in the violation of the information (security) criteria.

SQL injection[43] is in fact a very specific instance of parameter modification. As explained in Section 4.1.1, the OAS can use the PL/SQL language to communicate with the database. In order to set this up, the mod_plsql module for the Oracle HTTP Server can be used. In this way, a webpage can contain PL/SQL code that is executed in the database when that webpage is requested. Allowing user input in this PL/SQL code opens the opportunity of SQL injecting. An example of a bit of PL/SQL code that is used to check a username - password combination (for instance as part of a login process on a website) can be:

```
SELECT * FROM users
WHERE username = 'parameter1' and PASSWORD = 'parameter2';
```

Where the parameters can be modified by the user. Of course it was intended that parameter2 could only contain the users' password. However, by sending a crafted parameter2, an adversary could turn that into:

```
SELECT * FROM users
WHERE username = 'bob' and PASSWORD = 'pass' or 1 = 1;
```

so that the user 'bob' is logged on without supplying the correct password. SQL injections are

[41] In practice, more 'sensitive' parameters often can not be seen or modified in the URL address bar. Still, they can be modified using for instance a client side proxy like Paros (see http://parosproxy.org).

[42] See Section 4.4.1.

[43] This term is also used to address an attack that consists of injecting SQL statements in *stored procedures* (i.e. functions) in Oracle Database. See for instance http://tinyurl.com/3l2p2, accessed on 21/11/2007.

in principle not possible when Enterprise Java Beans[44] are used instead of mod_plsql.

All three types of attacks boil down to the processing of user import. The solution therefore is rather straightforward: user input should never be trusted. At the same time, this is very complex: in many real-word applications, much user data needs to be processed. For programmers that design the applications, it is seemingly very difficult to work under the assumption that all user input potentially can be of a malicious nature. The key solution to these problems is 'security by design': security has to be considered from the early design phase of each (web) application. Normally, this can however not be enforced by IT auditors since they are not involved in the design-phase. However, many tools[45] are available for testing websites and Web Services for these vulnerabilities.

As a kind of makeshift measure, the mod_security[46] module for Apache can be used. This open-source extension is installed (only not enabled) in Oracle HTTP Server by default. It is developed by Breach Security[47] and it works as an application firewall that is designed to block or mitigate several attacks including cross-site scripting, and SQL injection. Administrators should enable this module in order to mitigate these vulnerabilities. [CKRW06] however recommends not to use mod_security "in lieu of commercial-grade IDS solutions". Some of these commercial solutions probably indeed may offer a greater level of security. Nevertheless, using mod_security definitely is a good starting point.

Web Services are also vulnerable to parameter modification: cross site scripting is not possible since the XML files are not executed in the client browser. However, PL/SQL communication with the database via the Web Service is possible. As shown by [Lit02], OAS Web Services are prone to SQL injections when this type of communication is used. Using well-designed programs that carefully sanitize user input is –again– the key solution to mitigate this vulnerability.

OAS Forms & Reports

As described in Section 4.1.3, OAS Forms & Reports are two independent products that are developed by Oracle to interact with an Oracle Database via OAS using standard web pages that are developed by Oracle. Obviously, OAS Forms & Reports should be designed and programmed carefully so that users only can only see information that they are allowed to see and modify information that they are allowed to modify. In practice, many Forms & Reports applications are transformed two-tier applications. In these two-tier applications, database accounts and roles were used to regulate users' privileges. After transformation to three-tier applications, this user validation mechanism still is used by default. However, as explained in Section 4.4, also SSO can be used for validating users in OAS. When SSO is used, then user validation takes place using the OID. To utilize the advantage of one central user administration, only one (or at most a few) database accounts are used and all information about roles and privileges is stored in OID. Then, in order to maintain the security of the information in the database, it is required that OAS Forms & Reports applications are modified so that the SSO accounts can be used for users' authorization. For security reasons, it is then very important that SSO users' authorizations are checked carefully in the OAS Forms & Reports applications since these authorizations are no longer checked in the database. This is a potential vulnerability, especially since several vulnera-

[44] See Section 4.1.3.

[45] See for instance http://tinyurl.com/y8htsd and http://tinyurl.com/8hm32, both accessed on 14/11/2007.

[46] See http://tinyurl.com/7vv4j, accessed on 9/11/2007.

[47] http://www.breach.com

bilities[48] have been discovered in the standard web pages that are developed by Oracle to make OAS Forms & Reports applications available.

So, in spite of the advantages of a central user administration in OID, implementing SSO for OAS Forms & Reports may introduce extra inherent vulnerabilities. Therefore, it may be advisable that OAS Forms & Reports applications should not be accessible by external clients. However, advising against SSO for OAS Forms & Reports probably is too radical since such a central user administration also can help to increase security (at least the auditability).

OAS Portal

As explained in Section 4.1.3, OAS Portal is a web application that is developed by Oracle in order to supply an environment to enable central access to various sources of information. OAS Portal is inextricably bound up with SSO since only this mechanism can be used to regulate access to OAS Portal. Obviously, permissions ro read or modify information in OAS Portal should be granted and documented carefully. Further, OAS Portal should be patched carefully. At the time of writing (fall 2007), an IPS system is essential since OAS Portal contains unpatched[49] XSS vulnerabilities.

Default Web Applications

OAS is shipped with various default web applications, some of these web pages are demo applications, others are part of administrating interfaces (see Section 6.9, or are part of products like OAS Portal or Forms & Reports. On the Internet, several vulnerabilities in these default applications can be found [50]. There even is a free tool[51] to scan for these vulnerabilities.

Inherent vulnerabilities of web applications running on OAS	Touches on	Remedies to mitigate these vulnerabilities
Parameter modification, cross site scripting and SQL injections in Java applications	C I A	Demand 'security by design', test web applications extensively, enable mod_security or any other IDS / IPS solution that mitigates these vulnerabilities
Bypassing program permissions may lead to database access with a powerful database account when using SSO for OAS Forms & Reports	C I A M	Demand 'security by design', test OAS Forms & Reports applications extensively, enable an IPS solution that mitigates these vulnerabilities
Parameter modification, cross site scripting and SQL injections in OAS Portal	C I A M	Patch OAS Portal carefully, enable an IPS solution that mitigates these vulnerabilities, scan for vulnerabilities in default applications, remove as much default applications as possible

Table 6.5: Summary of inherent OAS application vulnerabilities, the information criteria Confidentiality (C), Integrity (I), Availability (A), and auditability (M) vulnerabilities where these vulnerabilities touch on, and the remedies to mitigate these vulnerabilities.

[48] For instance see http://tinyurl.com/yo8cea, accessed on 26/11/2007.

[49] See http://tinyurl.com/yqrw4b, accessed on 26/11/2007

[50] For instance see http://tinyurl.com/2z35mh, accessed on 17/12/2007

[51] This tool, called OAPScan, is developed by Deniz Cevik from the Turkish company inTellectPRO. It can be downloaded via http://tinyurl.com/37fth6 (accessed on 17/12/2007.)

6.7 OAS Infrastructure

Besides the mid-tier, also the infrastructure components of OAS have some inherent vulnerabilities. As depicted in Figure 4.1, the core components of the OAS infrastructure are the Metadata-repository (see Section 6.7.1), the Certificate Authority (see Section 6.7.3), and the Oracle Internet Directory (see Section 6.7.2).

6.7.1 OAS Metadata-Repository

A file based repository can only be used when the installation type[52] of OAS is *J2EE and Web Cache*. In that case, the most important weakness is the fact that these local files can be modified relatively easy. Moreover, increasing the infrastructure's 'availability' by clustering of metadata is not possible when a file based repository is used.

In practice, it is more likely that the metadata is located in an Oracle Database. Only when that is the case, the place where the metadata is stored is actually called the OAS *Metadata-Repository*. By default, a dedicated Oracle Database is installed as part of the the OAS Infrastructure. Besides the metadata, this database also contains information for OID. It is, however, also possible to use one or more existing Oracle databases for this purpose: both [RWD+04] and [PH05] advice to use a clustered Oracle Database for holding both the metadata and the data for OID. Without clustering, OAS Metadata-Repository can be identified as a single point of failure.

Obviously, the inherent vulnerabilities for Oracle Databases (see Section 6.5) also apply to OAS Metadata-Repository.

6.7.2 Oracle Internet Directory

As explained in Section 4.2.2, OID offers a directory service that provides enterprise information including account information. In this way, OID can for instance be used to authenticate and authorize users (also see Section 6.8).

Information from OID is used for Oracle Portal (also see Section 6.6.3), and for web based administration interfaces (see Section 6.9). Furthermore, it can be used for other applications including Forms & Reports and custom Java web applications. the OC4J component communicates with OID via an LDAP interface. The OC4J component communicates with OID via an LDAP interface when an application uses OID to authenticate or authorize an user. By default, this communication is not encrypted, although passwords are always transferred in a hashed format (that easily can be hacked when a user name is known [WC05]). Therefore, this LDAP interface should be secured by means of SSL.

This does however not solve all problems: in October 2000, a serious buffer overflow vulnerability[53] was found in the directory server of OID. Oracle responded to the disclosure of the vulnerability by releasing[54] a workaround and a patch in the summer of 2001. It is remarkable that this workaround only included a downgrade of the operating system privileges of OID's service. This can at best mitigate the vulnerability, whereas it is presented by Oracle as "a workaround to fix the security vulnerability". Of course, this is just an (relatively old) example of a vulnerability that can not be solved or mitigated by the implementation of SSL. However, on the Internet various

[52] See section 4.1.

[53] See for example http://tinyurl.com/33n6d5, accessed on 4/12/2007.

[54] See http://tinyurl.com/2k7op7, accessed on 4/12/2007.

(vague) descriptions of (unfixed) vulnerabilities can be found. Also in the Critical Patch Update of October 2007, very serious, but unspecified, vulnerabilities in OID were reported[55].

After all, this gives reason to be suspicious about the security of OID. Therefore it is, as a kind of makeshift measure, advisable to segment the network so that OID only can be reached by the OC4J component. This takes away the necessity of implementing SSL for the LDAP communication. Nevertheless, this also takes away the functionality that clients (and other directory servers) can communicate with OID directly using dedicated LDAP client software (see Section 4.4).

Inherent vulnerabilities of OID	Touches on	Remedies to mitigate these vulnerabilities
Disclosure or modification of information that is sent via LDAP protocol	C I A	Use SSL to encrypt (and sign) LDAP communication
Buffer overflows and other design/implenentation flaws in OID or its administration interface	C I A M	Carefully patch OAS Web Cache, implement an IPS

Table 6.6: Summary of inherent OID vulnerabilities, the information criteria Confidentiality (C), Integrity (I), Availability (A), and auditability (M) vulnerabilities where these vulnerabilities touch on, and the remedies to mitigate these vulnerabilities.

6.7.3 OAS Certificate Authority

As explained in Section 4.2.3, the CA of OAS is used to create signed digital certificates that can be used in a PKI. In the previous sections, SSL is presented as a key solution in mitigating risks in OAS by providing secrecy, integrity and authenticity of communication. OAS CA plays a central role in SSL communication since it provides the certificates that are used in this configuration. SSL is however not the silver bullet in information security: this protocol can be of great help in securing communication, but it is not completely flawless. According to [WS96], various so called man-in-the-middle attacks can successfully be performed on SSL. In such a man-in-the-middle attack an adversary is rerouting traffic between two honest parties via a computer that is under his control. A conceptual example of this type of attack is depicted in Figure 6.6. In this example, Bob thinks that he is communicating directly with the server. All traffic between Bob and the server is in this case however rerouted via the adversary. The adversary for instance can now hold back, modify, and replay this traffic. As explained by [WS96], version 3 of the SSL protocol (which is used in OAS by default) is not directly vulnerable to this attack. However, by default it is possible for clients (and so for adversaries) to perform a cipher rollback to version 2 of SSL. This version is vulnerable for this type of attack: the attacker possibly can decrypt all traffic between Bob and the server. [WS96] also points out some (potential) vulnerabilities in SSL version 3, but these can be strongly mitigated by implementing mutual authentication.

The OAS CA has a web interface that listens by default on port 6600 (SSL with server authentication) and port 6601 (SSL with mutual authentication). Like all web interfaces, this server is in principle vulnerable to (D)DoS attacks.

Considering the flaws in other OAS components, it cannot be ruled out that OAS CA contains some exploitable vulnerabilities[56] The consequences of a successful attack on OAS CA can be

[55] See http://tinyurl.com/23tvd7, accessed at 4/12/2007.

[56] However, I was not able to find any vulnerability report on OAS CA on the Internet on 10/12/2007.

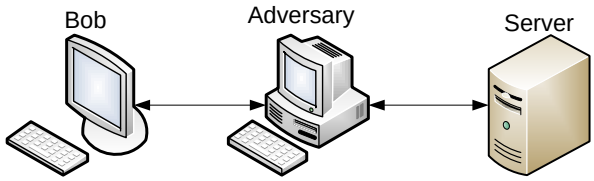

Figure 6.6: Conceptual example of a man-in-the-middle attack

far-reaching since the complete PKI structure depends on it. Therefore it is at least advisable to prohibit access to OAS CA by external clients.

Further, careful administration and consultation of the revocation list is essential: when the compromise of a key is presumed, then the corresponding certificate needs to be added to the revocation list as soon as possible. Additionally, all certificates that are signed by the public key of this certificate should be regarded as compromised.

[Mis05] advises to use a unique certificate (and so a different key pair) for each purpose, but the implementation does not prohibit users from mixing these purposes. In fact, the default setting is that a key pair can be used for all three purposes (authentication, signing, and encryption). This can be dangerous: secret keys for encryption generally are escrowed so that encrypted messages always can be read in case of an emergency[57]. Keys that are used for signing should however never be subject to escrowing because that may increase the risk of impersonating attacks. Moreover, using the same key pair for both authentication and signing can even lead to successful so called 'chosen ciphertext attacks' in which an attacker uses a prepared ciphertext to get it signed by a victim in order to decrypt a victim's encrypted message[58]. Hence this mixture should be forbidden.

Inherent vulnerabilities of OAS CA	Touches on	Remedies to mitigate these vulnerabilities
Successful man-in-the-middle attacks can be performed after rolling back the ciphersuite (from SSL version 3 to version 2)	C I	Prohibit ciphersuite rollbacks
The OAS CA's web interface can in principle be vulnerable to (D)DoS attacks, also the existence of other vulnerabilities (as a result of design flaws) cannot be ruled out	C I A	Prohibit access to OAS CA by external clients
According to [WS96] various (theoretical) vulnerabilities exsist in SSL version 3	C I	Demand mutual authentication
The same certificate can be used for both authentication and signing	C I	Demand and check that each certificate is used for at most one purpose (i.e. authentication or signing or encryption)

Table 6.7: Summary of inherent OAS CA vulnerabilities, the information criteria Confidentiality (C), Integrity (I), Availability (A), and auditability (M) vulnerabilities where these vulnerabilities touch on, and the remedies to mitigate these vulnerabilities.

[57]The current (at the time of writing, fall 2007) version of OAS CA does not have options for key escrow. For availability reasons, administrators should implement such an escrow system.

[58]According to [Ble98], this was possible in old versions of the SSL protocol.

6.8 Identity Management and Access Control

Section 4.4 and Section 4.5 explained the way in which identity management and access control is implemented in OAS. In fact, the basic idea of Oracle Identity Management is that each user has an SSO account by which al necessary tasks can be performed. However, in many organizations already database schema users have been created in the past. These organizations have to choose if they are willing to switch to an SSO enabled structure. When switching to SSO completely, not only schema users are replaced by SSO users, but also roles are moved from the database to OID. It is, however, also possible to use a mixed form of SSO accounts and database roles. In addition to this, it is also possible to use a full user administration in both the database (by means of schema users and roles) and in OID (by means of SSO accounts and roles). In any case, the implementation of SSO accounts has to be performed very carefully: as mentioned in Section 4.5.2, switching to SSO accounts for instance also has consequences for the way in which audit data should be collected.

Further, enabling SSO introduces some potential risks. Section 6.6.3 already explained that stealing cookies via for instance XSS attacks can lead to the violation of information (security) criteria.

Additionally, just like in any system for identity management and access control, it is essential that user accounts and roles are defined in the right way. To evaluate the way in which access management is regulated, auditors should of course examine both all schema users and roles from the database as well as all SSO accounts and roles from OID.

6.9 OAS Administration

Various OAS's components are provided with (web based) administration interfaces. These interfaces can introduce some inherent vulnerabilities that are mentioned briefly in this section.

As referred to in Section 4.5.2, each installation of OAS contains a product called *'Enterprise Manager'* (EM). This tool is in fact a Java servlet that offers a website that can be used to both monitor and administrate OAS (see Figure 6.7 for an example). These administrating tasks include deployment and configuration of applications, shutting down and starting up OAS's components, and changing various parameters. By default, EM uses a simple username (by default: ias_admin) and password (chosen at installation time). Also plain HTTP (without SSL) is used for the communication. From a security perspective, this obviously is not recommendable. Fortunately, it is possible to enable SSO[59] for access management and (mutual) SSL for securing the communication. In fact, a combination of SSO and mutual SSL enables a reasonable level of security for EM. However, it can never be ruled out[60] that EM contains web pages that are vulnerable to parameter modification attacks.

It is also possible to install an optional product called *'Enterprise Manager Grid Control'* that can be used to manage different OASs and Oracle Databases that are installed in a grid. This grid control application can send various kinds of alerts per email. According to a database administrator of a department of the Netherlands government, it is a challenge to find the appropriate reporting level: a balance has to be found between getting noticed about serious alerts on the one hand, while preventing from being overwhelmed by massive amounts of alerts on the other.

[59] See Section 4.4.1 and Section 6.8.

[60] Also see Section 6.6.3

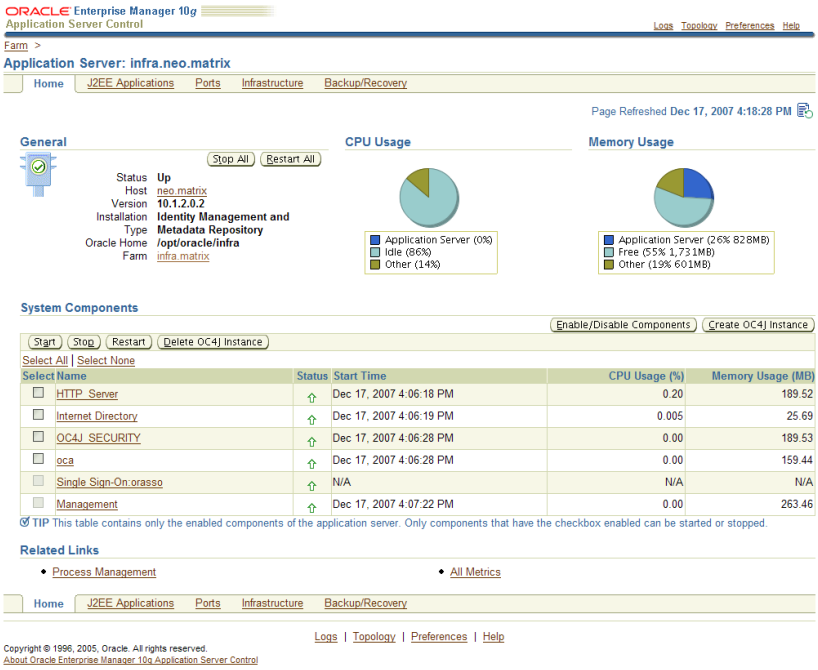

Figure 6.7: Screenshot of main screen of the Enterprise Manager web application

So, with some effort, even web based administration interfaces can be used in a reasonable secure way. However, a database administrator of a department of the Netherlands government told me that he had removed all remote administration interfaces from OASs that are accessible via the Internet. Obviously, there is a tradeoff here between security and the easy of administration: removing all remote administration possibilities indeed can increase OAS's security, but managing OAS then will be more difficult since physical access then is required to the computers on which OAS is installed. Considering the fact that some vulnerabilities were found in several default OAS applications (see Section 6.6.3), disabling these interfaces may be advisable, at least for the systems that are reachable via the Internet.

Also various OAS's components are supplied with specific administration interfaces. The CA web interface was already mentioned in Section 6.7.3, and the SSO management pages was addressed in Section 4.4.1. In general, it may be advisable to enable mutual SSL authentication and encryption for all administration interfaces. Additionally, it is advisable to prohibit access to these interfaces via the Internet.

6.10 Blended Vulnerabilities

The previous sections described various (more or less) independent vulnerabilities. In practice, however, an adversary will combine these vulnerabilities into a *blended threat*. In this way, various *attack vectors* can be identified: an adversary can step by step attack the system, getting closer and closer to his ultimate target. He might for instance start with exploiting an XSS vulnerability in a default application (see Section 6.6.3). Then, after stealing a user's session, he might find an application that is vulnerable to SQL-injections so that he can obtain access to the database with OAS's privileges. Of course, several other attack vectors can be made up: also an operating system

vulnerability can for instance be used to (ultimately) get control over the full OAS architecture.

Considering this, it is clear that making an OAS architecture secure is not an easy task: the strength of the chain is in the weakest link, hence each and every component of OAS should be made as secure as possible. On the other hand, there is often a tradeoff between security and useability and between security and investments in security measures. For this reason, a per-case risk investment should be carried out: a tradeoff between security risk and security investment should be made. However, some critical inherent vulnerabilities (including some that are described in this chapter) can be identified that should be fixed anyway. In addition, some general best practices can be composed.

6.11 Judgement

Oracle has been improving its security policy over the last years. However, there is much work that still needs to be done. In fact, Oracle's advice to segment the network and to use IDSs and IPSs can be seen as a weak bid: if all components and all communication were made secure, then all these measures would in principle not have been necessary.

The fact that even default applications are vulnerable for XSS attacks gives reason to be suspicious about the position of security concerns in the design and implementation of OAS components. Regarding Oracle's patching policy, it seems that the tradeoff between convenience and security is settled in favor of the first mentioned term.

Chapter 7

Conclusion

Oracle Application Server is in fact a collection of components on which applications can run that can make information from a database accessible via the web (i.e. via a website or a Web Service). Traditionally, clients were accessing the database directly. By Implementing this extra layer between the database and the clients, several advantages can be achieved for the security and the performance of IT systems.

However, OAS has some security issues that are worth a closer examination. This examination is performed in this thesis from the perspective of IT auditing –the field of performing risk-oriented evaluations of IT systems and other IT-related components and aspects within organizations.

In performing this examination, IT auditors often use 'general audit objectives' like VIR, BS7799, ITIL, and CobiT that are discussed in Chapter 5. These 'general audit objectives' were used in this project as a starting point in evaluating the 'reliability' (the extent to which an organization can rely on OAS for its information supply) of OAS. This evaluation was performed by first identifying a number of vulnerabilities that are inherent to the design and the implementation of OAS. These 'inherent vulnerabilities' were then used to compose a collection of 'concrete audit objectives' and an audit plan.

Just like 'general audit objectives' also 'concrete audit objectives' are not a silver bullet in guaranteeing a 'reliable' IT system. It is, however, a good starting point for the evaluation of this complex IT system. So, by using these 'concrete audit objectives' and this audit plan, various aspects of the 'reliability' of OAS can be measured in practice.

In fact, this thesis also indicates that Oracle has some work to do in making OAS secure: in the past, several serious security issues were discovered in various components of OAS. The fact that Oracle now is more transparent about security bugs and patches is quite an improvement: earlier it was unclear which patch was fixing which security issue. For administrators it is now possible to decide which patches need to be installed and which patches can be left aside. Thanks to Oracle's policy change, this certainly is improved. However, still a lot of work needs to be done before OAS indeed can be qualified as a truly 'reliable' IT product.

Bibliography

[AG06] Lance Ashdown and Paul Gavin, *Oracle Secure Backup Administrator's Guide*, Tech. report, Oracle Corporation, 2006.

[And01] Ross J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, John Wiley & Sons, Inc., New York, NY, USA, 2001.

[BLC90] T. Berners-Lee and R. Cailliau, *WorldWideWeb: Proposal for a HyperText Project*, European Laboratory for Particle Physisca (CERN), November 1990.

[Ble98] Daniel Bleichenbacher, *Chosen Ciphertext Attacks Against Protocols Based on the RSA Encryption Standard PKCS #1*, Lecture Notes in Computer Science **1462** (1998).

[BP02] Nick Bakker and Louk Peters, *ITIL in de praktijk*, PinkRoccade IT Management B.V., 2002.

[BSI99a] BSI/DISC Committee BDD/2, *BS 7799-1, Code of practice for information security management*, Tech. report, Britisch Standards Institute, 1999.

[BSI99b] ———, *BS 7799-2, Specification for information security management systems*, Tech. report, Britisch Standards Institute, 1999.

[CKRW06] John Carlson, Yi-Fang Koh, Harvey Rubinovitz, and Jackson Wynn, *Oracle Application Server Security Recommendations and DoDI 8500.2 IA Controls*, Tech. report, National Security Agency and MITRE Corporation, 2006.

[Des05] Ellen Desmond, *Oracle Identity Management Concepts and Deployment Planning Guide*, Tech. report, Oracle Corporation, 2005.

[DS05] Ellen Desmond and Richard Smith, *Oracle Internet Directory Administrator's Guide*, Tech. report, Oracle Corporation, 2005.

[GBS98] A. Goldberg, R. Buff, and A. Schmitt, *A Comparison of HTTP and HTTPS Performance*, Tech. report, Computer Science dept, Courant Institute, NYU, 1998.

[GSB04] Rick Greenwald, Robert Stackowiak, and Donald Bales, *Oracle Application Server 10g Essentials*, O'Reilly, 2004.

[ISO06] ISO/IEC, *Common Criteria for Information Technology Security Evaluation v3.1*, Tech. report, ISO/IEC, 2006.

[ITG07] ITGI / ISACA, *CobiT 4.1, Control Objectives, Management Guidelines, Maturity Models*, Tech. report, IT Governance Institute, 2007.

[KG95] Renoto Kuiper and Peter Groen, *Zijn Staatsgeheimen te beveiligen?*, Informatiebeveiliging (1995), 16–21.

[KM96] Sander Koppens and Bas Meyberg, *Operationeel beheer van Informatiesystemen*, Pink Elephant / Kluwer, 1996.

[KN93] John T. Kohl and B. Clifford Neuman, *The Kerberos network authentication service (V5)*, Tech. Report 1510, 1993.

[Lea00] Rick Leander, *Building Application Servers*, Cambridge University Press, 2000.

[Lee03] Spencer C. Lee, *An introduction to Identity Management*, Tech. report, SANS Institute, 2003.

[Lit02] David Litchfield, *Hackproofing Oracle Application Server*, Tech. report, NGSSoftware, 2002.

[Loo97] M. Looijen, *Beheer van Informatiesystemen*, Kluwer BedrijfsInformatie, 1997.

[Mis05] Vinaye H. Misra, *Oracle Application Server Certificate Authority*, Tech. report, Oracle Corporation, 2005.

[MSR06] Peter Mell, Karen Scarfone, and Sasha Romanosky, *Common Vulnerability Scoring System*, IEEE Security & Privacy **4** (2006), no. 6.

[NOR98] NOREA, *geschrift No. 1 - IT-auditing aangeduid*, Tech. report, Nederlandse Orde van Register EDP-Auditors, 1998.

[Per05] Elizabeth Hanes Perry, *Oracle Application Server Containers for J2EE*, Tech. report, Oracle Corporation, 2005.

[PF04] A.H. Phyo and S.M. Furnell, *A Detection-Oriented Classification of Insider IT Misuse*, Proceedings of the 3rd Security Conference, april 2004.

[PH05] Elizabeth Hanes Perry and Kevin Yu Hwang, *Oracle Application Server Security Guide*, Tech. report, Oracle Corporation, 2005.

[PM03] Andreas Pashalidis and Chris J. Mitchell, *A taxonomy of single sign-on systems*, ACISP, 2003, pp. 249–264.

[Pol04] Jennifer Polk, *Oracle Database Net Services Administrator's Guide*, Tech. report, Oracle Corporation, 2004.

[PW03] Birgit Pfitzmann and Michael Waidner, *Federated identity-management protocols*, Security Protocols Workshop, 2003, pp. 153–174.

[RWD+04] Thomas Van Raalte, Rodney Ward, Richard Delval, Xiang Liu, and Kai Li, *Oracle Application Server 10g High Availability Guide 10g (10.1.2)*, Tech. report, Oracle Corporation, 2004.

[Sch06] Harry Schaefer, *Oracle HTTP Server Administrator's Guide*, Tech. report, Oracle Corporation, 2006.

[Seb01] Sebasian Clauß and Marit Köhntopp, *Identity Management and its support of multilateral security*, Computer Networks **37** (2001), no. 2, 205–219.

[SG05] Saad Syed and Peter Goatly, *Security Target for Oracle Application Server 10g (9.0.4)*, Tech. report, Oracle Corporation, 2005.

[Ste06] Deborah Steiner, *Oracle Application Server Web Cache Administrator's Guide*, Tech. report, Oracle Corporation, 2006.

[vAZ07] Minister van Algemene Zaken, *Besluit voorschrift informatiebeveiliging rijksdienst 2007*, Staatscourant (2007), no. 122, 11.

[vBZ95] Ministerie van Binnenlandse Zaken, *Handboek Informatiebeveiliging Rijksdienst*, Ministerie van Binnenlandse Zaken, 1995.

[WC05] Joshua Wright and Carlos Cid, *An Assessment of the Oracle Password Hashing Algoritm*, SANS InfoSec Reading Room (2005).

[Wis06] Nina Wishbow, *Oracle Application Server Single Sign-On*, Tech. report, Oracle Corporation, 2006.

[WS96] David Wagner and Bruce Schneier, *Analysis of the SSL 3.0 Protocol*, The Second USENIX Workshop on Electronix Commerce Proceedings, USENIX Press, november 1996, pp. 29–40.

Abbreviations and Terms

API	Application Programming Interface: a library of building blocks for computer programs.
CA	Certificate Authority: a software component that issues, revokes, renews, and publishes digital certificates for a PKI infrastructure.
CIA	Acronym for information security criteria Confidentiality, Integrity, and Availability.
cookie	Small files that are located in web browsers and that for instance contain session information.
CPU	Central Processing Unit (within a computer).
	Critical Patch Update: quarterly released collection of Oracle patches.
CRL	Certificate Revocation List: a list containing identifiers to all revoked certificates from one or more CAs.
CVSS	Common Vulnerability Scoring System: a framework for assessing and quantifying the impact of vulnerabilities.
DIT	Directory Information Tree: a hierarchical structure that can represent the structure of entries and attributes within an LDAP-database.
(D)DoS	(Distributed) Denial of Service: an attack that consists of sending a massive amount of requests to a server (from different computers) so that it is no longer able to process legitimate requests.
HTML	HyperText Markup Language: language that is used to represent HyperText and its markup.
HTTP	HyperText Transfer Protocol: protocol for transferring HyperText over a TCP/IP network.
HTTPS	SSL-secured version of HTTP.
Hypertext	Text on a computer that is linked to other, related information so that related items of information are connected.
infrastructure	Collection of OAS' components that are needed to support the OAS midtier.

J2EE	Java 2 Enterprise Edition: an extension to the standard Java 2 edition of Sun Microsystems. J2EE adds support for so called 'Enterprise Java Beans' to the standard Java environment.
JSP	JavaServer Pages: a Java technology for dynamically generating content within (HTML) documents.
JVM	Java Virtual Machine: a program that is needed to interpret and execute compiled Java applications.
LDAP	Lightweight Directory Access Protocol: a standard that prescribes the way in which directory-servers and directory-clients can communicate with each other.
mid-tier	Collection of components that offer the core application server functionalities.
non-repudiation	Ensuring that an entity cannot repudiate the origin and authenticity of information.
OAS	Oracle Application Server.
OC4J	Oracle Containers for J2EE: runtime component of Oracle Application Server to run J2EE applications.
OID	Oracle Internet Directory: is a directory service, that is a framework that provides the listing of principals and their specific details.
PKI	Public Key Infrastructure: a framework of keys that can be exchanged to secure communication.
port	A logical data connection that is addressed by a number.
SBC	Server Based Computing (for instance Citrix or Windows Terminal Server).
servlet	a Java technology for dynamically generating (HTML) documents.
SOAP	Simple Object Access Protocol: an envelope that can contain XML messages.
SSL	Secure Socket Layer: cryptographic protocols that can be used secure various network based communication protocols.
SSO	Single sign-on: a method of access control that enables a user to authenticate once and gain access to the resources of multiple software systems.
TCP/IP	a collection of network protocols including the Transmission Control Protocol (TCP) and the Internet Protocol (IP). The Internet is for instance based on this TCP/IP protocol stack.
three tier	Architecture in which clients, an application server, and a database are involved.
two tier	Architecture in which clients and a database are involved.
UDDI	Universal Description, Discovery, and Integration: a registry that provides the opportunity of searching and accessing WSDL documents.

Web Service	A software application that supports direct interactions with other software applications using XML based messages via internet-based protocols.
WSDL	Web Services Description Language: a language that specifies the interface of a Web Service.
XML	eXtensible Markup Language: language that is focused on structured content representation only, without describing its layout.
XSS	Cross-site scripting: attack that is performed by leaving a message containing a script that is executed when a user clicks on a link or visits the web page that contains that message.

Appendix A

Audit Objectives for OAS 10g

This appendix provides a number of audit objectives[1] for auditing Oracle Application Server (OAS). These control objectives are formulated in a rather general way. When auditing complex objects like OAS, it is namely infeasible to use plain objectives can can be evaluated using a simple checklist. IT auditors need to think through the whole concept of OAS thoroughly when auditing a system in pratice. In this process, interpreting the philosophy behind the objectives can be of help in addressing the most important aspects of OAS.

Complying with these objectives is not a goal in itself, neither is it sufficient for obtaining a 'reliable' OAS. The aim of this document is addressing the most relevant 'quality aspects' (which are in particular aimed at security) of OAS. Hereby, it is advisable to use the objectives more as guidelines than to apply them literally. The organization that is evaluated can choose not to comply with certain objectives. In principle, this causes extra risks which should be considered and eventually be mitigated by supplementary measures.

The audit objectives in this document are based on several 'general audit objectives' (see Chapter 5) and 'inherent vulnerabilities' (see Chapter 6). The most relevant components from the *Business Intelligence and Forms* configuration type (which is the most extensive one) of OAS, version 10*G*, are addressed.

In order to make the audit objectives from this section applicable in practice, a so called audit plan is composed (see Appendix B). Each step from this audit plan corresponds with an objective from this section.

A.0 Preliminary Steps

Background:

Before the technical aspects of an IT system can be audited, it is necessary to acquire information about the organization and the general network architecture. Therefore, as a first step, IT auditors have to collect this information.

Objectives:

1. Acquire an overview of functions and staff.

[1] In Dutch, such a collection of audit objectives is called a '*normenkader*'.

2. Acquire a general network- and system topology (i.e. a network architecture and an overview of used hardware, operating systems and installed software).

A.1 Segregation of duties

Background:

Traditionally, in a database-enabled organizational environment five different roles can be discerned (this is an example, more fine-grained divisions of responsibilities are possible):

1. A **system administrator** (Dutch: *systeembeheerder*)who is responsible for the computer hardware, the operating system (OS), the installation of the Database software and other (client) applications.
2. A **database administrator** (DBA) who is responsible for creating and administrating the database. The DBA is also responsible for the backup- and restore process of the database.
3. An **operator** who is responsible for the availability of the hardware and software (and as a result also for backup- and restore procedures on OS-level).
4. A **security administrator** who is responsible for creating and managing user accounts and their authorizations.
5. A **security auditor** who is responsible for evaluating logical access controls and the system management processes.

This division of responsibilities is needed to comply with the concept of *segregation of duties* (in Dutch: *functiescheiding*) which can help in reducing the risk of accidental or deliberate system misuse.

OAS is often seen as an extension to a database environment. The traditional roles that are discerned in a database environment are however insufficient to enforce a sufficient level of separation of duties: naturally, the DBA is made responsible for creating and administrating OAS and the applications that run on top of this platform. This implies a considerable increase of the DBA's responsibilities. Therefore, it is a good idea to establish an extra role, namely that of the **application administrator** who is responsible for creating and administrating OAS and all its components (together with the applications that are running on OAS).

Objectives

1. The organization should have implemented a well-ordered structure of segregation of duties for reducing the risk of accidental or deliberate system misuse.
2. If different roles are performed by one person, than supplementary measures should be taken for reducing the risk of accidental or deliberate system misuse.

A.2 Network Communication

Background:

Various OAS components can be installed on different physical computers that are connected to each other via a TCP/IP network that can be used to communicate via various *application protocols*.

By itself, these *application protocols* are not secure: communication is not encrypted and even its authenticity cannot be guaranteed.

In fact two things can be done to secure (OAS-related) network communication:

1. Using (mutual) SSL to secure application protocols by encrypting and signing communication for security (and authenticity).

2. Segmentating the network by implementing firewalls.

Additionally, an Intrusion Detection System or, even better, an Intrusion Prevention System (IPS) can be installed to monitor and, in the case of an IPS, filter network traffic content.

In this context, it is advisable to discriminate between internal clients (that are inside the organization's LAN) and external client (that connect to the organization's LAN via the Internet). It is possible to restrict access external clients' access to the LAN by enforcing the use of SBC, a VPN and/or a proxy.

Objectives:

1. All sensitive information should only be sent over the network in encrypted form unless that information is sent over a dedicated (part of the) network so that it cannot be accessed by other parties than the sender or the receiver.

2. For high-sensitive information and information for which authenticity (and/or non-repudiation) is important, mutual authentication (using SSL) should be enforced unless that information is sent over a dedicated (part of the) network so that it cannot be accessed intercepted or modified by other parties than the sender or the receiver.

3. All OAS components and their underlying OS's should be *hardened*.

4. For access by external clients, the implementation of SBC, a VPN and/or a proxy should be considered.

5. At least for access by external clients, the implementation of and IDS (or better, an IPS) should be considered.

A.3 Physical Matters

Background:

Two different types of physical matters can be distinguished:

1. Physical attacks, for instance: destroying hardware, illegally plugging into the network, and cutting power. Here also social engineering and attacking the network via a wireless access point within the network of the organization are seen as 'physical' attacks.

2. Environmental calamities, for instance: fire, flood, and power failure.

Against both of these types, several preventive as well as reactive measures can be taken.

Objectives:

1. A tight physical access management should be implemented.
2. Obtaining wireless network access to the LAN should be impossible.
3. Users and administrators should have obtained a sufficient level of 'security awareness', meaning that they are behaving in a way that mitigates security risks.

A.4 Operating Systems

Background:

Even a full-secure implementation of OAS itself can be very vulnerable to various attacks when it is installed on top of an insecure operating system (OS). OSs' insecurity can for instance be the result of configuration errors, missing patches, or insecure applications that are installed. OS security actually is a topic in itself, therefore it is not treated here in detail.

Objectives:

1. The OS should be securely configured, well-patched and hardened.
2. No vulnerable applications should be installed on top of the OS.

A.5 Service Management

Background:

As described in Chapter 5, ITIL addresses the topic of *service management*. This *service management* is also important for OAS.

In ITIL, *service management* is de divided into *Service Support* and *Service Delivery*.
Service Support is then divided into seven processes:

1. Service Desk (communicate with users)
2. Incident Management (record and respond to incidents, see Section A.17)
3. Software Asset Management (optimize IT expenditures)
4. Problem Management (respond to problems that are caused by incidents)
5. Configuration Management (manage all configuration items)
6. Change Management (carefully implementing changes to the system)
7. Release Management (carefully upgrading the system (such an update can be seen as a collection of 'changes'))

The important issue of *patching* can be seen as part of the *change management* process. Patches can serve many purposes, the most important ones are fixing bugs and solving vulnerabilities. Therefore, it is necessary that newly released patches are implemented as quick as possible.

For an organization it is essential to devise a well-considered patch management plan. Patches are namely essential in fixing bugs and vulnerabilities: a missing patch can result in the violation of one or more information (security) criteria. On the other hand, installing patches can also have unwanted side effects: for instance applying a patch to OAS can result in J2EE applications that no longer work. Therefore, patches should be tested in a non-production environment before they are implemented in the production environment.

Service Delivery can be divided into five processes:

1. Service Level Management (composing and monitoring Service Level Agreements)
2. Capacity Management (Matching IT resources to the organizations' demands)
3. IT Service Continuity Management (see Section A.16)
4. Availability Management (see Section A.16)
5. Financial Management for IT Services

Objectives:

1. Service management should be implemented according to the corresponding ITIL processes. Hereby, it is advisable to use the objectives more as guidelines than to apply them literally.

A.6 Database

Background:

Just like OS security, auditing Oracle Databases is a separate issue (for which dedicated audit programs and audit objectives are available). As explained in Section 6.5, several vulnerabilities were found in Oracle Database. Therefore it is a good idea to use network segmentation so that the Oracle Database only can be reached by OAS so that other parties cannot connect to the database directly nor intercept traffic between the database and OAS. Further, for limiting the impact of an attack via OAS, it is advisable that an account with minimal needed privileges is used to connect to the database.

Objectives:

1. Access to the Oracle Database should be restricted to OAS only.
2. OAS should connect the this database with minimal privileges.

A.7 Oracle HTTP Server

Background:

The Oracle HTTP Server is in fact a modified version of the open source Apache HTTP Server. The Apache Foundation releases security patches on an irregular basis, whereas Oracle only brings out updates quarterly. This implies that security flaws in the Oracle HTTP Server can remain unpatched for a relatively long time even after the official patch is released by the Apache Software Foundation. This problem can be mitigated by responding to vulnerability reports that appear on the website of the Apache Software Foundation or other vulnerability-tracking websites.

The core Apache HTTP Server can be extended by (installing and) enabling different *modules*. These *modules* add certain functionality to the Apache HTTP Server. By default, the Oracle HTTP Server is supplied with several modules that are developed and maintained by Oracle. To mitigate vulnerabilities as much as possible, all unused modules should be disabled.

Objectives:

1. Vulnerability reports for Apache HTTP Server should be monitored, at least for (critical) systems that can be reached via the Internet.
2. All unused *modules* should be disabled.

A.8 Web Cache

Background:

OAS Web Cache can be placed in front of one or more web servers to cache any content that is transmitted via the HTTP protocol (See Figure 4.2). This means that the OAS Web Cache can offload the Oracle HTTP Server. This Web Cache can however also introduce new vulnerabilities:

1. The Web Cache can be brought down as the result of a (D)DoS attack (this in fact also holds for the HTTP Server). This issue is very hard to solve, but implementing an IPS can mitigate this vulnerability.
2. Buffer overflows and other design / implementation flaws in OAS Web Cache or its administration interface do also give attackers the opportunity of attacking OAS using these vulnerabilities. Also here, implementing an IPS and patching OAS Web Cache carefully can help.
3. Breaches in the data compartmentalizing mechanism of OAS Web Cache cannot be ruled out.

Objectives:

1. At least for availability-critical systems, an IPS should be installed to mitigate the risks of exploiting known vulnerabilities and (D)DoS attacks.
2. Sensitive data should not be cached (or all data that is cached should be considered as secret as the most classified data that is cached).

A.9 Applications

Background:

Also application security is a topic that is worth an audit plan on its own. Basically three kinds of applications can be discerned in the context of OAS:

JAVA can be used to build full-customized programs. In fact, JAVA can be used in different forms: it is possible to use J2EE that enables a transparent link between the program and the data in the database (that is can be cached in OAS in this manner). Several security issues play a role: SQL queries can for instance be included in JAVA programs. Given this situation, it can in principe possible for an adversary to read and / or modify these queries (this type of 'modifying' is called SQL injection). The possibility that JAVA source code is exposed can be mitigated by using *servlets* in favor of *JSP*s. Using J2EE can also mitigate the risk of SQL injection because then the JAVA programs in principal no longer contain SQL statements.

XSS (cross site scripting) also is important an important issue for Java-applications: By means of XSS malicious scripts can be executed on the users' computer. This may even result in hacked users' sessions when *session cookies* can be stolen. Both SQL injection and XSS can be prevented by designing and implementing and validating JAVA programs carefully. As an extra measure, the Apache module mod_security (or an commercial IPS solution) can be enabled to mitigate these vulnerabilities.

OAS Forms & Reports are two products that are developed by Oracle to interact with an Oracle Database via the OAS using standard web pages that are developed by Oracle. Since several vulnerabilities were found in these standard web pages, it is advisable that OAS Forms & Reports applications are made accessible for internal users only.

OAS Portal is a web application that is developed by Oracle in order to supply an environment to enable central access to various sources of information. OAS Portal is inextricably bound up with SSO (single sign-on) since only this mechanism can be used to regulate access to OAS Portal. Obviously, permissions to read or modify information in OAS Portal should be granted and documented carefully. Further, OAS Portal should be patched carefully since several vulnerabilities were found in the past.

Objectives:

1. JAVA
   (a) JAVA programs should be well-designed and should be implemented in a secure way (i.e. 'security' by design).
   (b) JAVA programs that contain SQL queries should be checked for SQL injections.
   (c) JAVA programs should be checked for XSS.
   (d) An IPS (or at least the *mod_security* module of Apache) should be implemented to mitigate the risk of SQL injections and XSS.
2. OAS Forms & Reports should be accessible for internal users only.
3. OAS Portal should be well-patched and -configured.

A.10 Metadata-Repository

Background:

The 'OAS Metadata Repository' is a collection of configuration data for OAS that is located in an Oracle Database. The information inside OAS Metadata Repository is used by OAS components. Communication with these components takes place over the proprietary Oracle NET protocol. It is possible to install the OAS Metadata Repository (and the other infrastructure components) on the same computer as the OAS midtier, but for 'availability'-reasons it is better to install the OAS Metadata Repository on a dedicated computer or cluster of computers. The OAS Metadata Repository contains application related metadata (for instance the datasources that are used by J2EE applications) and configuration related metadata (for instance which OASs are using that specific OAS Metadata Repository). One OAS Metadata Repository can be used by more instances of OAS. Obviously, the inherent vulnerabilities for Oracle Databases (see Section 6.5) also apply to OAS Metadata-Repository.

Objectives:

1. Access to Metadata Repository should be restricted to OAS only.

A.11 Oracle Internet Directory

Background:

Oracle Internet Directory (OID) offers a directory service that provides enterprise information including account information. In this way, OID can for instance be used to authenticate and to authorize users. Information from OID is used for Oracle Portal, and for web based administration interfaces. Furthermore, it can be used for other applications including Forms & Reports and custom Java web applications. When an application uses OID to authenticate or authorize an user, the OC4J component communicates with OID via an LDAP interface. By default, this communication is not encrypted, although passwords are always transferred in a hashed format (that easily can be hacked when a user name is know). Therefore, this LDAP interface should be secured by means of SSL.

Since several vulnerabilities were found in OID, it is (tentatively) advisable to segment the network so that OID only can be reached by the OC4J component. This takes away the necessity of implementing SSL for the LDAP communication. Nevertheless, this also takes away the functionality that clients (and other directory servers) can communicate with the LDAP server directly using dedicated LDAP client software.

Objectives:

1. The network should be segmented so that OID only can be reached via OAS. If that is not feasible, then communication with OID should be secured by SSL with mutual authentication.

2. OID should be well-patched.

A.12 Certificate Authority

Background:

The Certificate Authority (CA) of OAS is used to create signed digital certificates that can be used in a Public Key Infrastructure (PKI). OAS CA plays a central role in SSL communication since these certificates are used to both exchange and verify the public keys that are used in this configuration.

SSL can be of great help in securing communication. However, this protocol is not completely flawless. Especially older versions of SSL are found to be vulnerable for attacks, therefore ciphersuite rollbacks should be forbidden. In fact, SSL can be made more secure by demanding overall mutual authentication. This does however intensify the administrative and computational load of setting up and using an SSL-enabled infrastructure.

Considering that OAS CA is the centrepiece in secure OAS communication, it is advisable that access to OAS CA's web interface is restricted to internal clients only. OAS CA's web interface is namely vulnerable to (D)DoS attacks and it cannot be ruled out that other vulnerabilities are present.

Certificates can be used for different purposes purpose (i.e. authentication, signing, and encryption). For security reasons, certificates should be used for at most one purpose.

Objectives:

1. Ciphersuite rollbacks for SSL should be prohibited.
2. Mutual authentication for SSL should be considered.
3. Access to the OAS CA's web interface should be restricted to internal clients only
4. Certificates should be used for at most one purpose (i.e. authentication or signing or encryption).
5. OAS CA should be well-patched.

A.13 Identity Management and Access Control

Background:

The basic idea of Oracle Identity Management is that each user has an SSO account via which al necessary tasks can be performed. However, in many organizations database schema users have already been created in the past. These organizations have to choose if they are willing to switch to an SSO enabled structure. When switching to SSO completely, not only schema users are replaced by SSO users, but also roles are moved from the database to OID. It is, however, also possible to use a mixed form of SSO accounts and database roles. In addition to this, it is also possible to use a full user administration in both the database (by means of schema users and roles) and in OID (by means of SSO accounts and roles).

Enabling SSO introduces some potential risks: stealing cookies via for instance XSS attacks can lead to the violation of information (security) criteria. Furthermore, just like in any system for identity management and access control, it is essential that user accounts (whether based SSO

or schema users) and roles are defined in a way that is consistent with the organisational matters of section 1 of this document.

Objectives:

1. A conscious choice should be made about which 'form' of identity management is used: no SSO, partially SSO or full SSO.

2. SSO enabled web applications should be checked for XSS vulnerabilities.

3. The implementation of the chosen form of identity management should be consistent with the implementation of segregation of duties as described by Section A.1 of this document.

A.14 Administration Interfaces

Background:

Various OAS's components are provided with (web based) administration interfaces:

1. Each installation of OAS contains a product called the 'Enterprise Manager' (EM). This tool is in fact a Java servlet that offers a website that can be used to both monitor and administrate OAS. These administrating tasks include deployment and configuration of applications, shutting down and starting up OAS's components, and changing various parameters. By default, EM uses a simple username (by default: ias_admin) and password (chosen at installation time). Also plain HTTP (without SSL) is used for the communication. From a security perspective, this obviously is not recommendable: SSO and (mutual) SSL should be enabled.

2. It is also possible to install an optional product called 'Enterprise Manager Grid Control' that can be used to manage different OASs and Oracle Databases that are installed in a grid.

Considering the fact that some vulnerabilities were found in several default OAS applications it is advisable that these interfaces indeed should be disabled, at least for the systems that are reachable via the Internet.

Objectives:

1. Access to administration interfaces should be disabled or at least be restricted to internal clients (or better: administrators) only.

2. Mutual SSL authentication and encryption should be enabled for all administration interfaces.

A.15 Defaults

Background:

An OAS installation does contain some dangerous defaults in the form of vulnerable demo-applications, unprotected administration and monitoring services and default usernames (and passwords). OAPScanNG.pl (see Section 6.9) is a program that can be used for checking vulnerable demo-applications and unprotected administration and monitoring services. Additionally, on the Internet and in the literature (for instance, see [Lit02]) various 'default password lists' can be found to check if default username and password combinations are left unchanged. Of course, default usernames (like ias_admin and cn=orcl) can also be tried with the most obvious passwords of even with passwords that are generated from a standard password list or by a brute-force algorithm.

Objectives:

1. All OAS components and their underlying OSs should be hardened: all unnecessary features and extras (like demo applications) should be removed.
2. Default accounts should be disabled if possible (and if not, than at least their passwords should be set to a non-default and non-trivial value)
3. Default port numbers should be changed to mitigate the risk of 'fingerprinting' using a port-scan.

A.16 Availability

Background:

This topic also is an ITIL process. Since OAS demands specific measures, this process is discussed here in more detail.

Even mission critical computer systems, which need to be available anytime, are subject to planned and unplanned downtime. There are many possible causes of a system being down; maintenance plays a part (as a planned matter) but, in the main, the origin lies in unplanned matters like security breaches (attacks), system failure, disasters or human errors. At least for mission critical systems, measures need to be taken in order to overcome the fact that each computer system will be faced with downtime eventually. Basically, two things can be done to overcome problems with the downtime of computer systems:

Using redundant systems by means of clustering OAS components so that multiple installations of one component are presented as one single component to users and administrators as one OAS. This grouping of components is achieved by means of a load balancer that forwards requests to a clustered OAS instance. Such a load balancer can be a dedicated hardware device, but it is also possible to perform load balancing either by means of the operation system or the OAS Web Cache.

Scalability is a key issue for application servers. As explained in Chapter 2, increasing systems' performance was an important reason to come up with the 'middle-tier'. Even more important, clustering also can help to overcome the downtime of a computer systems

since it offers a form of redundancy: if one instance of OAS fails, the other OASs within the cluster will automatically take over its tasks. All OASs within a cluster do share the same infrastructure. This means that the failure of an infrastructure component does affect all the OASs within that specific cluster. To make the cluster capable of coping with the failure of an infrastructure component, it is possible to also group two or more infrastructure servers into a cluster.

Creating and maintaining backups to prevent data loss. Creating backups of OAS can be automated for the most part using Oracle Secure Backup. In order to get useful backups, it is essential to lay down what, when (i.e. the frequency) and how data is backed up.

File system backups are not stored in encrypted form. Database backups, however, are stored in encrypted form by default. The fact that file system backups are not encrypted by default gives reason for extra measures: at least for 'confidentiality'-reasons tapes containing this un-encrypted data should be stored in a safe environment or they should be encrypted manually.

In addition to these 'static' backups, there are, also more dynamic forms of performing the process of backup and recovery. A product called 'OAS Guard' can be used to facilitate this, even over geographically remote locations: the basic idea is that two identical OAS architectures are installed: a production architecture at the local site, and a standby architecture (including both the mid-tier and the infrastructure of OAS) at a geographically remote site. 'OAS Guard' then can be configured to sent data from the production site to the standby site so that in case of a failure at the production site, the standby site can take over its tasks relatively easy. Technically, this can for instance be realized by using a DNS-server that bounds a domain name to the production server by default, and that bounds this domain name to the stand-by server when a failure of the production site is detected. By default, SSL is used to encrypt the data that is transferred between the production site and the stand-by site. An important note is that backups (whether static or dynamic) should be tested regularly.

Objectives:

1. Redundancy of all components should be implemented for availability-critical systems so that no 'single point of failure' can be identified.
2. Creating and maintaining backups:
   (a) Backups should be created and tested regularly.
   (b) Backups should be stored in a safe manner.
   (c) For availability-critical systems, backups should be securely transferred to a (i.e.: at least one) safe geographically remote site.

A.17 Logging and Alerts

Background:

This topic also is an ITIL process. Since OAS demands specific measures, this process is discussed here in more detail.

Each component in OAS has its own logging mechanisms that record various types of events like errors, access attempts, warning messages and reports about start-ups and shutdowns. Oracle has developed a mechanism to make a number of these logs available for administrators via a standard web interface. This web interface is part of the Oracle Enterprise Manager that is installed as part of each OAS installation by default. Administrators can log on to this web interface using a browser to perform various administrative tasks. From an auditability perspective, it is a good idea to have such a cental log repository. However, it would have been even better if logs were written in a user-centric form so that an audit trail per user could be created. At the time of writing, OAS lacks this functionality.

OAS Grid Control can send various kinds of alerts per email. A balance has to be found between getting noticed about serious alerts on the one hand, while preventing from being overwhelmed by massive amounts of alerts on the other *Oracle Audit Vault* is a special Oracle product that can be used to collect audit data in a secure dedicated environment.

Objectives:

1. The directories where the logs are written should be protected so that users can not modify or delete log files easily.
2. When automatic notifications are enabled, then the sensitivity of this mechanism should be set to an appropriate level.
3. Log files should be reviewed periodically looking for critical events.
4. The implementation of *Oracle Audit Vault* should be considered.

Appendix B

Audit Plan for OAS 10g

This appendix provides a program[1] for auditing Oracle Application Server (OAS). This audit plan is based on the control objectives that are described in Appendix A.

The audit program is described in the table below. The number in the first column (#) corresponds to that section in Appendix A.

#	Description
0	Preliminary steps
0.1	Acquire an overview of functions and staff.
0.2	Acquire a general network- and system topology (i.e. network architecture and overview of physical components, operating systems and installed software).
1	Organizational Matters
1.1	Are different roles defined for: 1. the system administrator, 2. the database administrator (DBA), 3. the security administrator, and 4. the application administrator?
2	Network Communication
2.1	Is all 'sensitive' information encrypted when that is transferred over (compartments of) a network to which adversaries potentially can get access to?
2.2	Is mutual authentication enforced for high sensitive communication and for communication for which authenticity is important?
2.3	Is access by external clients restricted to SBC, a VPN and/or a proxy?
2.4	Is an IDS / IPS implemented?
3	Physical Matters
3.1	Is physical access to the building, the network and the physical computers regulated in a tight way?
3.2	Is it impossible to plug into the organization's LAN wirelessly?

[1] In Dutch, this is called a '*(controle)(werk)programma*'.

#	Description
3.3	Do users and administrators have a sufficient level of 'security awareness'?
4	Operation Systems (If desired, use dedicated OS audit plan)
4.1	Is the OS securely configured, well-patched and hardened?
4.2	Are no vulnerable applications installed on top of the OS?
5	Service Management
5.1	Is service management implemented according to (the ideas behind) the corresponding ITIL processes?
6	Oracle Database (If desired, use dedicated Oracle Database audit plan)
6.1	Is access to the Oracle Database restricted to OAS only? (by means of network segmentation)
6.2	Does OAS connect to the Oracle Database using a schema user account with minimal privileges?
7	Oracle HTTP Server
7.1	Are vulnerability reports for Apache HTTP Server monitored for critical systems that can be reached via the Internet?
7.2	Are all unused modules disabled?
8	Oracle Web Cache
8.1	Are well-functioning IPSs implemented in mission critical systems?
8.2	Is sensitive data excluded from caching or is all data that is cached considered as secret as the most classified data that is cached?
9	Applications
9.1.a	Are JAVA programs well-designed, validated, and implemented in a secure way (i.e. 'security' by design) (request for documentation)?
9.1.b	Are JAVA programs that contain SQL queries checked for SQL injections (also automated tools can help)?
9.1.c	Are JAVA programs checked for XSS (also automated tools can help)?
9.1.d	Is an IPS (or at least the mod_security module of Apache) implemented to mitigate the risk of SQL injections and XSS?
9.2	Is OAS Forms& Reports accessible for internal users only?
9.3	Is OAS Portal well-patched and –configured?
10	Metadata-repository
10.1	Is access to Metadata Repository restricted to OAS only?
11	Oracle Internet Directory
11.1	Is the network compartmentalized so that OID only can be reached by OAS?
	Is the communication with OID secured by SSL with mutual authentication if network compartmentalization is unfeasible?
11.2	Are security patches applied to OID?
12	Certificate Authority
12.1	Are ciphersuite rollbacks for SSL prohibited?
12.2	Is mutual authentication for SSL taken into consideration?
12.3	Is access to the OAS CA's web interface restricted to internal clients only?
12.4	Are certificates used for at most one purpose (i.e. authentication or signing or encryption)?
12.5	Are security patches applied to OAS CA?

#	Description
13	Identity Management and Access Control
13.1	Is the choice of which 'form' of identity management to use (no SSO, partially SSO or full SSO) made is a conscious way?
13.2	Are SSO enabled web applications checked for XSS vulnerabilities?
13.3	Is the implementation of the chosen form of identity management consistent with the organizational matters of step 1 of audit plan?
14	Administration interfaces
14.1	Is access to administration interfaces disabled or at least restricted to internal clients (or better: administrators) only?
14.2	Is mutual SSL authentication and encryption enabled for all administration interfaces?
15	Defaults
15.1	Are all unnecessary features and extras (like demo applications) removed (for instance use OAPScanNG.pl to check)?
15.2	Are default accounts disabled and if not, are their passwords set to a non-default and non-trivial value (for instance use default accounts and passwords list from [Lit02], also some guess work can be profitable)?
15.3	Are default port numbers changed (as far as that is reasonable)?
16	Availability
16.1	Is redundancy (i.e. mirroring) of all components implemented for availability-critical systems so that no 'single point of failure' can be identified?
16.2.a	Are backups created and tested regularly?
16.2.b	Are backups stored in a safe manner?
16.2.c	Are backups for availability-critical systems securely transferred to a (i.e.: at least one) safe geographically remote site?
17	Logging and alerts
17.1	Are the directories where the logs are written protected so that users can not modify or delete log files easily?
17.2	Is the sensitivity of automatic notifications (if this mechanism is enabled) set to an appropriate level?
17.3	Are log files reviewed periodically for critical events?
17.4	Is the implementation of *Oracle Audit Vault* taken into consideration?