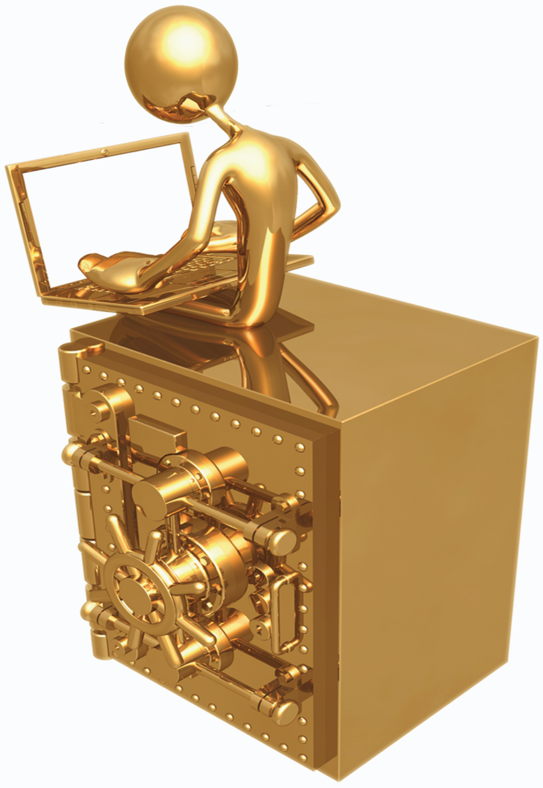
CRC Press
Taylor & Francis Group
AN AUERBACH BOOK

HOWTO Secure and Audit Oracle 10g and 11g



Ron Ben Natan

Foreword by Pete Finnigan

HOWTO

Secure and Audit

Oracle 10g and 11g

HOWTO Secure and Audit Oracle 10g and 11g

Ron Ben Natan

Foreword by Pete Finnigan

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
AN AUERBACH BOOK

CRC Press
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Version Date: 20150120

International Standard Book Number-13: 978-1-4200-8413-9 (eBook - PDF)

Dedication

To my father Danny

Contents

Foreword

In recent years, Oracle security has assumed a whole new meaning for many people in organizations around the world; we have seen the rise of regulatory requirements and worse still a huge rise in the reporting of data loss. While not from an Oracle database, the recent loss of two CDs in the United Kingdom containing all the child benefit details of over 25 million people could not be a more graphic example for people who want their data to remain secure. Securing Oracle databases is more important today than it was many years ago when I started dedicating my business and research life purely to thinking about and providing companies and the community with assistance in securing their data held in Oracle databases.

Companies have also been more widely reporting an increase in internal rather than external attacks to their systems. This is of the highest significance for the security of data held in an Oracle database as in today's world the use of perimeter security is of little value when it comes to securing the data. Unfortunately, most companies have open network architectures and most employees have indirect or direct access to the database (whether intended or not) due to open routing policies and also standard desktop builds. As we have seen, the biggest threat in recent times comes from internal attacks; this could again be intentional or unintentional. Remember, in the world of securing data, the adversary that the Database Administrator (DBA) has to compete with may not be a hacker; he or she may be a fellow employee who has too much access that allows damage and unauthorized changes to data and the database itself to occur, or any other of the myriad of situations that allow people unauthorized access, again intended or not.

Securing an Oracle database and the data held in it should be of utmost importance to all of the people within an organization—from the managers who write the checks to the DBAs, developers, security analysts, users, and owners of the data. Securing data held in an Oracle database is not "rocket science" but it is complex because the Oracle database itself is complex and infinitely configurable and also because the applications, data needs, and use are also infinite and different for each organization. Wow! This sounds like a big problem, doesn't it? If every system, application, configuration, use, people, and so on is different we clearly need best practices to secure Oracle and the data.

I should make a clear distinction here. The task of securing the "data" should be held separately from the task of securing the "Oracle software." Why is this? Well, simply put, following a checklist or "tip sheet" is not good enough. We must ensure that this is done as practitioners of "securing data"; yes, securing data must come first and not securing the software; we must understand the data, understand its use, understand what I call its "flow"; how does the data get into the database, how does it leave the database, and where is it at rest. Only then can we start to secure the data using the tools provided by Oracle.

I am a believer in best practices and ideas—good ideas, not just simply ticking off checklists. I believe that if you understand your data you can secure it. I have helped define best practice and helped many clients secure their data. A methodology should teach the principles and obviously discuss the features and tools that Oracle provides, but it must also teach how to understand the risks to the data. Ron's book is clearly written and focuses on the core technology available from Oracle to secure your data, but, importantly, it discusses why you should secure your data and then provides guidelines as to how you should do it using out-of-the-box tools. This is important as it means the book is useful to any customer of Oracle since Ron uses the techniques and tools provided with the Oracle license (additional cost options are discussed, as well such as Audit vault or Virtual Private Database). This is a practical book that any layman can follow and the core concepts are well explained. Did I mention that Oracle security is complex…? ; well Ron cuts to the quick and covers all the core issues. I like to think an ideal book teaches the reader the "how" and the "why," which they can then apply to all aspects of the security of their data. I think Ron has achieved this. I was particularly impressed with the clear discussions on privileges; this has been my main bugbear with lots of customers; I see lots of sites with excessive privileges, serious privileges, wide-ranging access for lots of people; if we can teach users of data the importance of access to only the data they should access and only at the times they should access it, we would have made massive progress toward secure data. Enjoy this book but most importantly learn from it and use it to secure your data.

Pete Finnigan
Managing Director,
PeteFinnigan.com Limited

Acknowledgments

I would like to thank my wife and kids for tolerating my vanishing acts during nights, weekends, and any other time that should have been spent with them and went instead into writing this book.

I would like to thank the whole crew at Guardium for helping me deal with the complex topics of security and Governance, Risk and Compliance (GRC) in large database environments every single day.

Finally, I would like to thank the many people I have worked with over the years on Oracle security—those that I have met as Guardium customers, and others who I have interacted with in Oracle forums. The many hundreds of such interactions helped me understand what is important from a very real and pragmatic point of view, and not merely from a "tooling" perspective. The list of people is too long to mention here but if you have ever taken the time to sit down with me and explain what you need, I thank you for that.

Ron Ben-Natan

Author

Ron Ben-Natan has more than 20 years of experience developing enterprise applications and security technology for blue-chip companies. Ron is currently the chief technical officer at Guardium, the leader in database security and auditing. Prior to this he has worked for companies such as Merrill Lynch, J.P. Morgan, Intel, and AT&T Bell Laboratories. He is an IBM GOLD consultant with a PhD in computer science. Ron is an expert in distributed application environments, application security, and database security. He has authored 11 technical books including *Implementing Database Security and Auditing*. He speaks frequently at database and security seminars including sessions run by Oracle University.

Chapter 1

Introduction: How This Book Will Help You Be Secure and Compliant

The word Oracle means (from Wikipedia):

> An **oracle** is a person or agency considered to be a source of wise counsel or prophetic opinion; an infallible authority, usually spiritual in nature. It may also be a revealed prediction or precognition of the future, from deities, that is spoken through another object (e.g.: runemal) or life-form (e.g.: augury and auspice). In the ancient world many sites gained a reputation for the dispensing of oracular wisdom: they too became known as "oracles", and the oracular utterances, called khrēsmoi in Greek, were often referred to under the same name — a name derived from the Latin verb ōrāre, to speak.

A pretty good name for a database management system (DBMS). But there is another important bit of history behind the use of the word Oracle to name the database engine. Much before Oracle was a company as we know it today, Larry Ellison and Bob Miner, two of the founders of Software Development Labs (which later became Oracle), were working on a consulting project for the Central Intelligence Agency (the CIA in the United States). The CIA wanted to use a new Structured Query Language (SQL) that IBM had written a white paper about. The code name for the project was Oracle (the CIA saw this as the system to give all answers to all kind of questions intelligence analysts had). Larry Ellison and Bob Miner saw the opportunity to take what they had started as part of this project and market it. So they used that project's code name of Oracle to name their new database engine and later the company.

Today, Oracle is the number one database engine based on any metric and it dominates many geographies and many verticals/industries. Perhaps the vertical that it dominates most is that of military organizations, agencies such as the CIA, National Security Agency (NSA), Federal Bureau of Investigation (FBI), and other organizations where security is of utmost importance. This is part of the company's legacy and it is evident in the product. Maybe this origin is the reason that Oracle has more security-related functions, products, and tools (both built by Oracle as well as available as third-party products) than any other comparable DBMS in the world.

Unfortunately, the fact that these capabilities exist does not mean that they are always well-known and used correctly. In fact most users of the Oracle database, even those who have been working with Oracle for many years, are often familiar with less than 20 percent of the security mechanisms within Oracle. This leads sometimes to insecure Oracle environments and other times to implementation which use the wrong tools—meaning a lot of unnecessary work and solutions which require too much effort to sustain over time. One of the main goals of this book is to review the methods and tools available for securing Oracle so that when you have to implement any security-related requirement (be it access control, audit trails, configuration assessment, encryption, etc.), you know how to navigate the options, which tool to use for which scenario, and how to avoid common pitfalls.

1.1 Why Secure the Data?

Let's start with understanding why you must invest in Oracle security. This sounds like a silly question—but you'll have to answer this question in one form or another once you start asking for budgets. It's quite obvious why you need to secure the data—right? Everything you care about sits in a database (and if you're reading this book most likely a lot of it is in an Oracle database). Whether it is financial company data, data about customers, data about employees, credit card information—it is usually stored in a database. There are many other forms that this data can take—data in documents, data in e-mails, data in IM messages, etc. These do not usually live inside the database and there are other tools and techniques (not covered in this book) for securing these endpoints. These data elements are often permutations of data that was sourced from a database. Almost all information that you and your organization consider to be valuable resides in a database in some form or at some point in time. Obviously, you need to secure these "crown jewels."

So far so good. But now let's start asking slightly different questions. Suppose that you did your analysis and go to your management with a proposal to secure the database that will cost $50,000 (or Euros, or Pounds, or whatever your currency may be) per database. What if your proposal required you to add 10 to your headcount? Will it still be so obvious that you need to secure the data (to that level)? Management will most likely not want to have the data "so secure" at that point. If you map your requirements and request $1000 per database; will it then be approved? That depends on what value this investment provides at a business level and what business problem this investment solves. Security, like any other IT investment needs to be justified and being able to answer the question of "why secure the data" (or the less simplistic variants of this question) is important.

At a very high level, the reasons for securing your data fall into two broad categories—you need to avoid a data breach and you need to remain in compliance with some internal or external set of requirements. Both data breaches and noncompliance can cost an organization dearly. A data breach can damage a company's brand, can lead to loss of customers, and always costs a lot of money—money spent on remediation, compensation, investigations, audits, notification, etc. Noncompliance too can be very costly. Noncompliance leads to fines, can lead to loss of a license to operate the business, can lead to continual expensive audits for many years to come, etc.

The justification for investing in securing Oracle is simple—it is a far lower cost than the cost involved with a data breach or noncompliance.

When you build your business case for elevating the security of your data you may have to justify it with a return on investment (ROI). When you build your business justification you should first list the essential elements that must be performed to achieve an acceptable level of security and compliance. You usually do not need to justify this part with an ROI. Security is in many ways like buying insurance. There is no ROI on buying home insurance. If nothing happens during that year (and normally nothing does) then you get no return on a sizable investment. But if your house burns down—well, then you'll be very happy you bought insurance. Security is similar—if there is an attempt to hack your database and it is foiled, your investment has paid off. What is usually more important than ROI is the total cost of ownership (TCO) of the proposal.

Once you have defined the essential elements that have to be implemented comes the second part—your implementation plan. This is where ROI comes in. Given a set of requirements, there are usually many ways to achieve them. Some may involve tools and others may involve manual work done by staff. At this point, you will have to justify your decisions to implement one method over another—and this is done by showing that one method has a much better ROI than another.

Data Breaches

There are many resources and Web sites that track data breaches. For example, the Privacy Rights Clearinghouse keeps a chronology of data breaches that have been reported that involve data that can be useful to identify thieves—including Social Security numbers, account numbers, and driver's license numbers. This list enumerates more than just database-related breaches—it lists all kinds of data-related breaches (which include also things like stolen laptops, etc.). This list only covers reported incidents—and not all incidents are reported by any stretch. The list covers only incidents reported in the United States. This includes a running total of the number of records that have been compromised. Here too, this number is understated because in many incidents the number of affected records is unknown and therefore not counted. Just to give you an idea of the magnitude of the problem—between 2005 and June 2008 there have been at least 225 million records compromised as part of the reported data breaches in the United States. Another good list is the Data Breach Blog that is maintained by SC magazine.

Here is a small sampling of some known incidents involving database breaches (not necessarily Oracle databases):

- In February 2003 the BBC reported an attack on a database that held credit card accounts where the attacker gained access to more than five million Visa and Mastercard accounts.
- In October 2004 a hacker compromised a database containing sensitive information on more than 1.4 million California residents. The breach occurred on August 1 but was not detected until the end of the month. The database in question contained the names, addresses, Social Security numbers, and dates of birth of caregivers and care recipients participating in California's In-Home Supportive Services (IHSS) program since 2001. The data was being used in a UC Berkeley study of the effect of wages on in-home care and was obtained with authorization from the California Department of Social Services. The hacker had reportedly taken advantage of an unpatched system and although officials declined to state which vendor's database was the subject of the attack they did report that it was a "commercially available product with a known vulnerability that was exploited."

- In August 2005 an Air Force spokesman reported that a hacker tapped into a U.S. military database containing Social Security numbers and other personal information for 33,000 Air Force officers and some enlisted personnel.
- In April 2006 Computerworld reported on a case in which an employee at Progressive Casualty Insurance wrongfully accessed information on foreclosure properties she was interested in buying. There was no hacking involved—just a misuse of insider privileges. When the incident was discovered the company sent out letters informing people that confidential information had been accessed by this employee who was fired. The incident was discovered because a local woman complained about receiving calls from a Progressive agent inquiring about her house being under foreclosure—not because there was any database monitoring or auditing in place.
- In September 2006, the virtual reality game SecondLife reported that one of its databases containing unencrypted user information was breached.
- In October 2006, an official with the Illinois Ballot Integrity Project, a not-for-profit organization dedicated to the correction of election system deficiencies, reported that the organization hacked into the voter database for the 1.35 million voters in the city of Chicago and could have not only stolen private information but also create election problems by modifying status and data.
- In June 2007, eWeek reported that Fidelity National Information Services, an electronic payment processor, fired a database administrator (DBA) after they found that the DBA stole and sold customer data exposing as many as 2.3 million bank and credit card records. The DBA, who worked at the company's Certegy Check Services unit, sold the information to a data broker who in turn sold some of it to direct marketers. These activities led to customers receiving marketing solicitations—which was how the incident was exposed.
- In July 2007, credit card information, names, addresses, and phone numbers were hacked from a Western Union database.
- In August 2007, Monster.com reported that details of over 1.6 million job seekers and information on 146,000 subscribers to usajobs.gov residing in a resume database managed by monster.com has been stolen.
- In September 2007, TD Ameritrade reported that information on 6.3 million customers was stolen from one of its database. The stolen information included names, addresses, and e-mail addresses plus a variety of account activity information. The company reported that it discovered and eliminated unauthorized code. Although it did not provide further details, it is likely that this was done by a privileged user. TD Ameritrade said it discovered the breach after customers said they had received spam offering unsolicited investment advice.
- In January 2008, a hacker broke into a database of the Davidson Companies, a financial services firm. The hacker obtained the names and Social Security numbers of practically all of the company's clients as well as information relating to account numbers and balances.
- In March 2008, Harvard University reported that the database containing summaries of GSAS applicant data had been compromised and that about 10,000 of 2007 applicants' personal information may have been compromised. At least 6000 Social Security numbers were exposed and a compressed 125 MB file containing the stolen data is available through BitTorrent, a peer-to-peer network. The file included server backups of three databases and a note was attached which reads "maybe you don't like it but this is to demonstrate that persons like … (admin of the server) in that they don't know how to secure …".

- In May 2008 Computerworld reported that a professional penetration tester (an ethical hacker) managed to hack his way through to a major FBI crime database within a mere six hours.

Data breaches have, unfortunately, become part of our daily lives. In addition to looking at long lists of incidents, it is instrumental to look at some commonalities. One of the best sources for learning about these is the 2008 Data Breach Investigations Report—a study published by the Verizon Business RISK Team. This report draws from over 500 forensic engagements handled by the Verizon Business Investigative Response team over a period of four years. The report provides statistics on how breaches occur, where they occur most (in terms of verticals), what methods were used, and more. For example, the report lists that most breaches resulted from a combination of events rather than a single event but in almost all cases some form of error contributed to a compromise, whereas less than a quarter of attacks exploited vulnerabilities. This shows how important it is to know how to use the security options correctly. Of the incidents due to vulnerabilities, 90 percent of the vulnerabilities exploited by these attacks had patches available for at least six months prior to the breach (hence, you'll learn how to read Critical Patch Updates in Chapter 2).

The report also finds that nine of ten breaches involved something that is unknown to the owner—the most common one being data that was not known to exist on the compromised system. This is shown in Figure 1.1. The report calls these recurring situations as the "Achilles heel in the data protection efforts of every organization." This is perhaps the most important theme in data breaches—you cannot protect that which you do not know about and you cannot secure what you cannot see. The importance of visibility—visibility into where sensitive data resides, visibility into who or what is accessing sensitive data, visibility into controls—is perhaps the most important element that must exist for a database to be secure. Two other very disturbing facts that the report finds is that most breaches go undetected for a long time and are discovered more often by a third party than the breached organization and that most

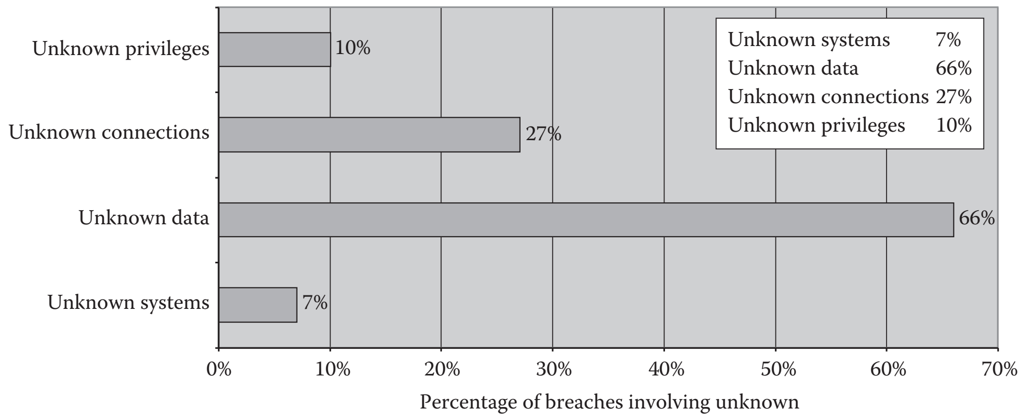


Figure 1.1 Breaches involving unknown factors.

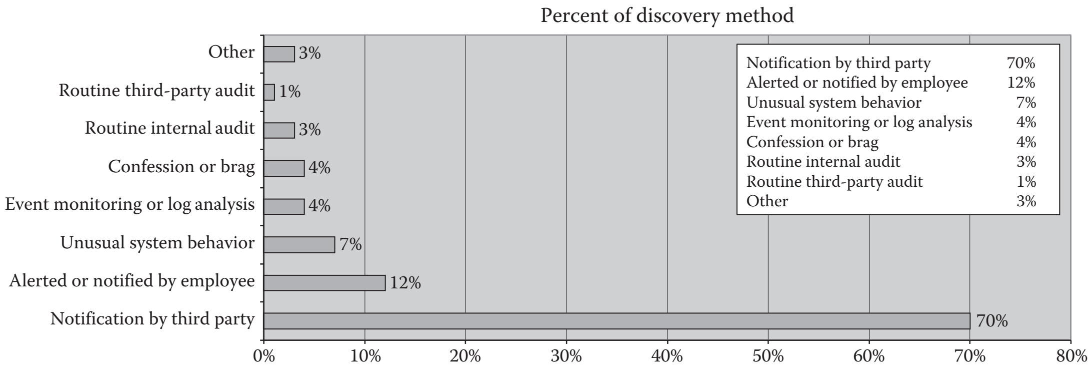


Figure 1.2 Discovery of breaches.

attacks are low to moderate in difficulty—i.e., the attacker does not have to work too hard. Specifically, the report finds that

- ■ 66 percent of attacks involved data the victim did not know was on the system.
- ■ 75 percent of the breaches were discovered by a third party and not the breached organization—as shown in Figure 1.2. The report goes on to present the data shown in Figure 1.3 regarding the time until discovery. This is a very serious finding—it shows that there is a great deficiency in monitoring and auditing. There is a huge difference between a breach that lasts for a day versus a breach that goes on for months, and between a breach that is discovered and handled versus one where the victims (the people who's data is stolen) cannot defend themselves because they don't even know they are victims.
- ■ 83 percent of the attacks were not difficult to perform.
- ■ 87 percent of the attacks could have been avoided through reasonable controls.

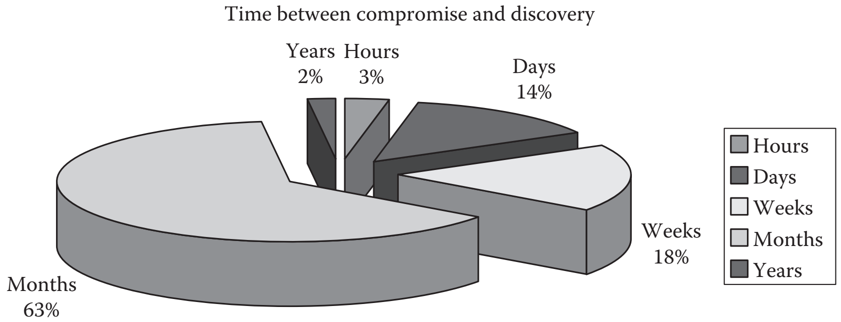


Figure 1.3 Time until discovery.

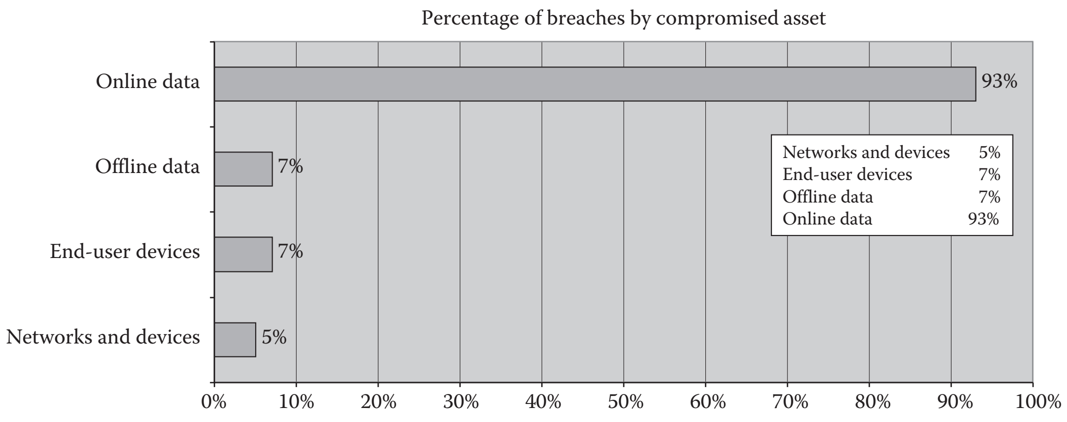


Figure 1.4 Which data is most often targeted?

Finally, if you have any doubt about the importance of database security in the battle against data breaches, the report brings some statistics on the assets that were compromised. Data sits in many places and takes many forms. Data can be in a database but can also reside on USB keys. Data that resides in a database also exists in backups and taken offline. As Figure 1.4 shows, compromises to online data repositories occurred more than five times more often than all other asset classes combined!

Compliance

It would be wonderful if we invested in security because we were so disturbed by the many data breaches we've seen and because we always want to do the right thing—but the truth is that we usually invest in security because we're told to do so. Most of the investment in Oracle security is made because of the need to comply with a regulation or a requirement.

There are two kinds of compliance requirements—internal and external. Internal requirements are policies that are defined within the company. They include policies set by the information security or internal audit groups and they are usually derived from industry best practices, from a regulation, or from preparation for an external audit. External requirements derive from a regulator or from external auditors. There are numerous examples of regulations that affect database security—including Sarbanes Oxley (and its derivatives), the Payment Card Industry Data Security Standard (PCI DSS), various data privacy laws, and many others. Some of these regulations are industry-specific, some are national (i.e., affect only companies operating in a certain country), and others relevant to companies of a certain size. Most of these regulations do not directly set requirements related to database security—they need to be interpreted and mapped to IT terms and these interpretations determine what is required to be in compliance. Luckily, most of these regulations have been around long enough to have a standard interpretation and one that is consistent with requirements set by industry best practices.

Compliance is a very important driver—especially when it comes from an external source. If you need to comply with a certain regulation, it is very hard to shut down a project for lack of funding or other priorities. This is the great power of compliance and the reason that so much of database security is driven by compliance requirements. When you know you have to pass an audit or face certain penalties and when you know that the auditor is an external entity, you are usually bound to make the investment and elevate the security of your data. Compliance and regulation is a very positive factor in data security and much of the improvements in the last five years can be attributed to auditors and the processes they put us through.

1.2 Taxonomy of Best-Practice Database Security

Before you start reading the HOWTOs, you should realize that securing an Oracle database is not a trivial task. Oracle has many capabilities. The Oracle SQL language is richer than most DBMSs. You can connect to Oracle using a great many networking libraries and authentication methods. There are many thousands of packages and procedures in any database. There is a Java virtual machine and an Hypertext Transfer Protocol (HTTP) server. You can call out from the database using external procedures or various utility packages. All these are examples of functionality that is great when you look at productivity and building applications. But from a security standpoint, the more options and capabilities a server has, the harder it is to secure and monitor it properly. The reason is simple—each such option can be used by an attacker to gain unauthorized access. Securing Oracle requires a lot of know-how and involves doing work in a variety of categories.

There is a large body of knowledge by now on what activities are required to secure Oracle and to comply with regulations and requirements. There are checklists you can follow and you will learn about them in Chapter 2. This is good—it means you can adhere to a set of best practices and achieve security and compliance by investing in the following:

- Discovery—you can't secure that which you don't know. You need to have a good mapping of your assets—both of your instances and of your sensitive data. Plus, you need to automate discovery because the state of this "asset map" six or twelve months into the future will be different from what it is today.
- Vulnerability and configuration assessment—you need to assess the configuration of your databases to ensure that they don't have security holes in them. This verification includes both the way the database is installed on the operating system and the configuration options within the database itself. You need to verify that you are not running versions of the database with known vulnerabilities.
- Hardening—the result of an assessment is often a set of recommendations. This is the first step in hardening the database. Other elements of hardening involve removing all functions and options that you do not use.
- Change auditing—once you have a hardened configuration you must continually track it to ensure that you don't digress from a secure configuration. You do this using change auditing tools which compare snapshots of the configurations (at both the operating system level and at the database level) and alert you when a change is made that may affect the security of the database.
- Database activity monitoring—although changes can and should be tracked using change auditing, you can also use database activity monitoring to alert on changes made through an SQL interface. Additionally, database activity monitoring lets you detect intrusions and misuse, detect fraud, and discover problems at real-time, limiting your exposure considerably.

- Auditing—audit trails must be generated and maintained for database activity that may have an impact on security, integrity, or on access to sensitive data.
- Authentication, access control, and entitlement management—not all data and not all users are created equal. You must authenticate users, you must ensure full accountability per user, and you must manage privileges to limit access to data. You need to enforce these privileges even for the most privileged database user. You also need to review entitlement reports periodically as part of an audit process.
- Encryption—use encryption to render sensitive data unreadable. Use encryption so that an attacker cannot gain unauthorized access from outside the database. This includes both encryption of data-in-transit so that an attacker cannot eavesdrop at the networking layer and gain access to the data when it is sent to the database client as well as encryption of data-at-rest so that an attacker cannot use the media files and extract the data there.

1.3 Using HOWTOs to Secure Oracle

Oracle is the largest and most-functional DBMS platform today. As such it has many aspects that need to be secured—the bigger the "surface area" of a server, the more work is required to secure it properly. Luckily, Oracle has more security features than any other database and often there is more than one way to address a problem. This book aims to help you understand what these different options how, when to use them, and how to use them. To achieve this goal the book is structured as HOWTOs—specific "recipes" that show you how to use each of these security functions in the context of Oracle 11g and Oracle 10g. Most of these HOWTOs focus on the tools that exist within the Oracle database but some extensions are provided when relevant.

The HOWTOs are organized into chapters by security category and by topic. In Chapter 2, you will learn how to harden the database—that is, remove unnecessary components and choose configuration settings that make it harder for an intruder to gain unauthorized access. You'll learn how to choose a checklist and how to create a secure configuration baseline. You'll learn how to read Oracle Critical Patch Updates and the importance of patching. In Chapter 3, you will focus on listener security and see what a secure listener configuration looks like. Chapter 4 deals with account security. You'll learn about user management, password policies and complexity, and what the standard logon process looks like.

Because Chapters 6 through 8 all deal with topics that use cryptographic functions, Chapter 5 provides an overview of cryptography, including encryption, digests, and digital signatures. In this chapter you'll also learn how to use an Oracle wallet and how to use Oracle Wallet Manager and orapki for generating and managing certificates and other secrets. With this background you'll learn about the various authentication options available with and without Advanced Security Options in Chapter 6, how to encrypt database communications in Chapter 7, and how to encrypt the data within the database in Chapter 8.

The next set of chapters deal with the important topic of auditing. In Chapter 9, you'll learn how to use standard auditing, how to use and manage audit trails, and about their impact on the database. Chapter 10 extends this with a discussion on administrator auditing and Chapter 11 augments with fine-grained auditing. In Chapter 12, you'll learn some advanced techniques for auditing before and after values and for discovering what data was extracted from the database.

Auditing is usually the first step in any Oracle security implementation. Additionally, there are some basic requirements that an audit trail must satisfy for it to be considered valid—requirements that are architectural in nature. For example, you need to ensure separation of duties—that is, the DBAs need to be audited and cannot have control over the audit trail. For this reason, an external product—Oracle Audit Vault has been created. You'll learn about this product and how to use it in Chapter 13. Because monitoring and auditing access to the database is such an important topic, an entire space has emerged for addressing this need called database activity monitoring. In Chapter 14, you'll learn what these third-party tools provide and when you can use them.

Chapters 15 through 17 deal with access control and with enforcing privileges. In Chapter 15, you'll learn about the Oracle privilege model, how to authorize access, and how to audit what privileges have been assigned. You'll learn about roles and about secure application roles that can be used by an application to implement an authorization structure directly within the database. In Chapter 16, you'll learn how to use Virtual Private Database to enforce row-level access control and how to mask sensitive data. Finally, in Chapter 17 you'll learn how to use Oracle Database Vault to limit users even if they have system privileges and how to implement an external security overlay.

The organization of the chapters follows a logical order of implementation steps, but most of the chapters can be read independently. You can read through the chapters in the order that suits the tasks you need to perform. The HOWTOs within the chapter can usually also be read independently and where possible, are self-contained. So, feel free to read the book in order or skip through it based on your needs at work.

Chapter 2

Hardening the Database

System hardening is the process by which you securely configure a system to protect it from unauthorized access. System hardening is necessary in any system that has a range of configuration options and is viable in any system that has enough security measures to make them suitable for usage in security-oriented environments. Oracle database falls into both of these categories.

The purpose of system hardening is to eliminate as many security risks as possible. This is done by removing all nonessential elements from the system and by selecting configuration options that limit access and reduce risk. As Oracle has evolved, more and more options have become available and these options offer new ways to access data—sometimes by unauthorized users if used inappropriately. The larger the footprint and capabilities of a system, the harder it is to harden and the more security risks may be present. Therefore, as the Oracle database grows in size and functionality, hardening becomes even more important. Luckily, as Oracle evolves, there are also more and more security options available that you can use to secure the data—Chapters 3 through the end of the book outline these capabilities and how you should use them. But first, you need to harden the database.

Oracle hardening covers a wide range of activities and involves many types of configuration options. The most important guideline is that if there is a feature that you do not use, remove it. The fact that you don't use something does not mean that an attacker won't. The smaller the surface area of a system, the more secure it is. Examples of this include:

- Remove or lock predefined accounts that you do not use and change the password for accounts you do use that have a predefined password (Oracle 11g already comes configured that way).
- Remove predefined roles that you do not use.
- Remove components in the database software that you do not use.
- Remove options that you do not use—for example, remove EXTPROC from your listener if you do not use external procedures.
- Remove privileges from PUBLIC that you do not require.

Because Oracle has so many capabilities and configuration options, hardening is usually an exercise that involves hundreds of activities. Coming up with a list of these required activities is a monumental task. Luckily, you don't have to come up with this list. Lists have been created and entire books are dedicated to this topic—so all you have to do is pick a source—the topic of the first HOWTO of this chapter.

2.1 HOWTO Choose a Hardening Guideline

Hardening of any complex system involves many little details. The more a system can be configured, the lengthier the list of tasks you need to perform (or validate) to create a hardened configuration. The list for Oracle is long, but openly available and free.

There are two documents that provide very mature guidelines for implementing a secure Oracle configuration and that you should look at when forming your standard hardening process. One is the Database Security Technical Implementation Guide (STIG) developed by Defense Information Systems Agency (DISA) for the Department of Defense (DOD) and the second is the Center for Internet Security (CIS) Benchmark for Oracle developed by the CIS. Both are excellent and very comprehensive; use one of them (or both) rather than develop your own hardening checklist.

Database STIG

STIGs are documents published by the DISA to assist in improvement of the security of DOD information systems. There are numerous STIG documents—all of them are accessible at http://iase.disa.mil/stigs/stig/index.html. The checklists can be downloaded from http://iase.disa.mil/stigs/checklist/index.html. The Database STIG focuses on relational databases. The Database STIG has a generic section which outlines guidelines relevant to any database management system (DBMS) and has an Oracle-specific section which adds steps relevant for Oracle only. The sections within the general document address:

1. Integrity
   a. Software integrity
   b. Database software development
   c. Ad-hoc queries
   d. Multiple services host systems
   e. Data integrity—including file integrity, software baseline, and file backup and recovery
2. Discretionary access control
   a. Account control
   b. Authentication
   c. Database accounts
   d. Authorizations
   e. Protection of sensitive data
   f. Protection of stored applications
   g. Protection of database files
3. Database auditing
   a. Audit data requirements
   b. Audit data backups
   c. Audit data reviews
   d. Audit data access
   e. Database monitoring

4. Network access
   - a. Protection of database identification parameters
   - b. Network connections to the database
   - c. Database replication
   - d. Database links
5. Operating system (OS)
   - a. File access
   - b. Local database accounts
   - c. Administrator accounts
   - d. OS groups

The Oracle-Specific Policy and Implementation appendix specifically addresses:

6. Oracle access control
   - a. Oracle identification and authentication
   - b. Oracle connection pooling
   - c. Secure distributed computing
   - d. Oracle administrative connections
   - e. Oracle administrative OS groups
   - f. Default accounts
   - g. Default passwords
   - h. Oracle password management requirements
7. Oracle authorizations
   - a. Predefined roles
   - b. System privileges
   - c. Object privileges
   - d. Administration of privileges
8. Oracle replication
9. Network security
   - a. Encrypting network logins
   - b. Protecting network communications
   - c. Listener security
   - d. XML DB protocol server
10. Oracle Intelligent Agent/Oracle Enterprise Manager (OEM)
11. Oracle account protections
12. ARCHIVELOG
13. Securing SQL*Plus
14. Protecting stored procedures
15. Oracle trace utility
16. Auditing in Oracle—includes standard auditing, fine-grained auditing, mandatory auditing, and architectural discussions
17. File and directory permissions at the OS level
18. Critical file management—including control files, redo log files, and data files
19. Optimal Flexible Architecture (OFA)
20. Initialization parameters
21. Miscellaneous OS requirements—including Unix, Window, and z/OS

The Database STIG is published as an unclassified document and is made available to all. DISA also publishes a set of evaluation scripts and these can help you check the security strength of your database—download these from http://iase.disa.mil/stigs/SRR/index.html.

CIS Oracle Benchmark

The CIS (www.cisecurity.org) publishes the CIS Benchmark for Oracle as part of a set of benchmarks, scoring tools, software, data, and other services that are made public as a service to all users worldwide. You can download the benchmark from http://www.cisecurity.org/bench_oracle.html. The recommendations contained in the Oracle benchmark result from a consensus-building process that involves the leading Oracle security experts. The CIS benchmark takes the form of a checklist partitioned into a number of sections. Within each section is a list of items that should be validated. Each such item includes a description of the item, the action or recommended setting for parameters, comments, which Oracle version it applies to, and whether it is relevant to Unix, Windows, or both. The main sections in the CIS Oracle benchmark are

1. OS-specific settings
2. Installation and patch
3. Oracle directory and file permissions
4. Oracle parameter settings
5. Encryption-specific settings
6. Startup and shutdown
7. Backup and disaster recovery
8. Oracle user profile setup settings
9. Oracle user profile access settings
10. Enterprise Manager/Grid Control/Agents
11. Items relevant to specific subsystems
12. General policy and procedures
13. Auditing policy and procedures
14. Appendix A—additional settings

Both documents take a broad approach to hardening. They do not have a narrow interpretation that hardening only involves certain configuration settings, removing default components, locking users, etc. They provide a full checklist that also includes what activities should be audited, where separation of duties is required, what activities need to be performed, etc. Of the two—the STIG puts even more focus on the general implementation, process, roles that need to be involved in securing an Oracle environment, etc.

Two Things to Remember about Choosing a Hardening Guideline

1. Don't build your own checklists—hardening an Oracle database is no longer an art; there are good mature guidelines for you to choose from such as the CIS Oracle benchmark or the Database STIG.
2. Use these documents not only as a hardening guide but also as the basis for putting together a complete Oracle security implementation. These documents outline configuration settings but also outline process, procedures, and what to focus on. In many ways, all the chapters in this book explain how to use Oracle tools to implement what these two documents suggest that you do.

2.2 HOWTO Use a Vulnerability Assessment Tool

Using hardening checklists is simple but tedious. From a pure hardening perspective, the checklists contain many checks and modifications that you need to perform and these can be automated. In fact, without automation this tasks quickly becomes unmanageable—especially if you have tens and hundreds of instances and they do not all conform to a single "gold build." Tools that you can use to automate this process are called vulnerability assessment (VA) tools or vulnerability scanners.

For almost any type of system there are VA tools and Oracle is no exception. VA tools will scan your database instances and come back with a report showing what changes you need to perform to make your database more secure. These results are presented in the form of a security report where each problem is classified and a recommendation provided for what changes you need to make (e.g., see Figure 2.1). These checks and recommendations usually cover the items specified in the various hardening checklists—meaning that a VA tool can save you most of the tedious work involved in reviewing your databases and their alignment with the checklists.

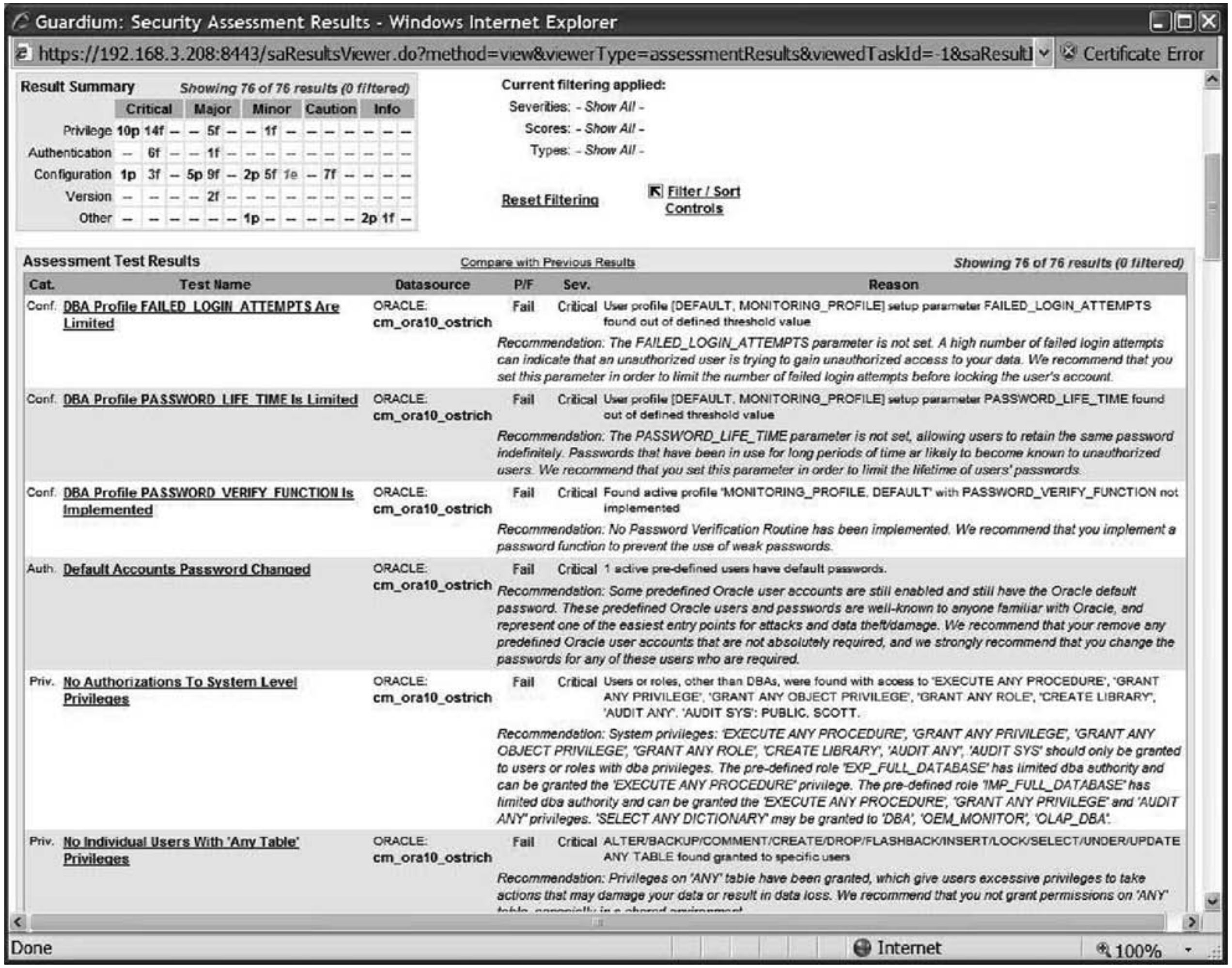
There are many VA tools for Oracle including AppDetective, AppSentry, Guardium, IPLocks, and NGS Squirrel. Some of these tools are stand-alone VA scanners and some tools are part of a larger suite of products that address multiple aspects of Oracle security. From a pure VA perspective all these tools scan your database to recommend changes you need to make to harden your instances. The tools that are integrated within a larger suite have an advantage—in the same way that STIG and CIS take the wider interpretation to securing the database environment and define practices for auditing, practices for review, etc. (see the previous section), so do these suite-based tools. This allows you to become secure and fully compliant within a single implementation thus saving a lot of time.

VA tools perform many types of checks. These checks can be classified into three main groups:

1. Checks for software vulnerabilities
2. Checks for misconfigurations
3. Checks for misuse of the database

All of these checks are necessary to check for vulnerabilities in your database. An attacker can gain unauthorized access to your database by using a code vulnerability that exists because you have not patched your server using the latest Critical Patch Update (CPU) (an example of type 1), through the use of a default account that has not been locked and still has a default password (an example of type 2), because you have REMOTE_OS_AUTHENT set to true (an example of type 2), or because everyone knows the password for SYSTEM and multiple people make use of this account constantly (an example of type 3).

Checking for vulnerabilities (of all types) is done using a multipronged approach. Some things can be checked from the outside-in and other checks are done from within the database. VA tools have multiple modes in which they work to provide you with the full picture. Many checks need to be performed from the inside. For example, to check for bad configurations the VA tool needs to be able to access V$PARAMETER. To check for bad privilege assignment (as an example—too many privileges assigned to PUBLIC can be a serious vulnerability), VA tools need certain SELECT privileges to the catalog. These tools come with scripts that grant these privileges to a user or a role (that is then assigned to a user you create for the tool to use). Review these scripts when you evaluate the tool to ensure that it does not assign itself too many privileges. The better VA tools also inspect and check files at the OS level. For example, lax file permissions or a wrong owner or group used for important Oracle files (such as data files, software files, configuration files, and log files) are a serious vulnerability. Contents of files such as sqlnet.ora can affect how your database behaves and checking a configuration must include checks on files, on registry values (on Windows), environment variables, etc. Finally, comprehensive checks will often include inspecting output from scripts and OS commands—e.g., inspecting the


Guardium: Security Assessment Results - Windows Internet Explorer

https://192.168.3.208:8443/saResultsViewer.do?method=view&viewerType=assessmentResults&viewedTaskId=-1&saResultI

Certificate Error

Result Summary *Showing 76 of 76 results (0 filtered)*

	Critical			Major			Minor			Caution			Info		
Privilege	10p	14f	–	–	5f	–	–	1f	–	–	–	–	–	–	–
Authentication	–	6f	–	–	1f	–	–	–	–	–	–	–	–	–	–
Configuration	1p	3f	–	5p	9f	–	2p	5f	1e	–	7f	–	–	–	–
Version	–	–	–	–	2f	–	–	–	–	–	–	–	–	–	–
Other	–	–	–	–	–	–	1p	–	–	–	–	–	2p	1f	–

Current filtering applied:

Severities: *- Show All -*

Scores: *- Show All -*

Types: *- Show All -*

Reset Filtering

Filter / Sort Controls

Assessment Test Results

Compare with Previous Results

Showing 76 of 76 results (0 filtered)

Cat.	Test Name	Datasource	P/F	Sev.	Reason
Conf.	DBA Profile FAILED_LOGIN_ATTEMPTS Are Limited	ORACLE: cm_ora10_ostrich	Fail	Critical	User profile [DEFAULT, MONITORING_PROFILE] setup parameter FAILED_LOGIN_ATTEMPTS found out of defined threshold value *Recommendation: The FAILED_LOGIN_ATTEMPTS parameter is not set. A high number of failed login attempts can indicate that an unauthorized user is trying to gain unauthorized access to your data. We recommend that you set this parameter in order to limit the number of failed login attempts before locking the user's account.*
Conf.	DBA Profile PASSWORD_LIFE_TIME Is Limited	ORACLE: cm_ora10_ostrich	Fail	Critical	User profile [DEFAULT, MONITORING_PROFILE] setup parameter PASSWORD_LIFE_TIME found out of defined threshold value *Recommendation: The PASSWORD_LIFE_TIME parameter is not set, allowing users to retain the same password indefinitely. Passwords that have been in use for long periods of time ar likely to become known to unauthorized users. We recommend that you set this parameter in order to limit the lifetime of users' passwords.*
Conf.	DBA Profile PASSWORD_VERIFY_FUNCTION Is Implemented	ORACLE: cm_ora10_ostrich	Fail	Critical	Found active profile 'MONITORING_PROFILE, DEFAULT' with PASSWORD_VERIFY_FUNCTION not implemented *Recommendation: No Password Verification Routine has been implemented. We recommend that you implement a password function to prevent the use of weak passwords.*
Auth.	Default Accounts Password Changed	ORACLE: cm_ora10_ostrich	Fail	Critical	1 active pre-defined users have default passwords. *Recommendation: Some predefined Oracle user accounts are still enabled and still have the Oracle default password. These predefined Oracle users and passwords are well-known to anyone familiar with Oracle, and represent one of the easiest entry points for attacks and data theft/damage. We recommend that your remove any predefined Oracle user accounts that are not absolutely required, and we strongly recommend that you change the passwords for any of these users who are required.*
Priv.	No Authorizations To System Level Privileges	ORACLE: cm_ora10_ostrich	Fail	Critical	Users or roles, other than DBAs, were found with access to 'EXECUTE ANY PROCEDURE', 'GRANT ANY PRIVILEGE', 'GRANT ANY OBJECT PRIVILEGE', 'GRANT ANY ROLE', 'CREATE LIBRARY', 'AUDIT ANY'. 'AUDIT SYS': PUBLIC, SCOTT. *Recommendation: System privileges: 'EXECUTE ANY PROCEDURE', 'GRANT ANY PRIVILEGE', 'GRANT ANY OBJECT PRIVILEGE', 'GRANT ANY ROLE', 'CREATE LIBRARY', 'AUDIT ANY', 'AUDIT SYS' should only be granted to users or roles with dba privileges. The pre-defined role 'EXP_FULL_DATABASE' has limited dba authority and can be granted the 'EXECUTE ANY PROCEDURE' privilege. The pre-defined role 'IMP_FULL_DATABASE' has limited dba authority and can be granted the 'EXECUTE ANY PROCEDURE', 'GRANT ANY PRIVILEGE' and 'AUDIT ANY' privileges. 'SELECT ANY DICTIONARY' may be granted to 'DBA', 'OEM_MONITOR', 'OLAP_DBA'.*
Priv.	No Individual Users With 'Any Table' Privileges	ORACLE: cm_ora10_ostrich	Fail	Critical	ALTER/BACKUP/COMMENT/CREATE/DROP/FLASHBACK/INSERT/LOCK/SELECT/UNDER/UPDATE ANY TABLE found granted to specific users *Recommendation: Privileges on 'ANY' table have been granted, which give users excessive privileges to take actions that may damage your data or result in data loss. We recommend that you not grant permissions on 'ANY'*

Done

Internet

100%

Figure 2.1 Sample recommendation report from an Oracle VA scanner.

output of orapatch to ensure that you have the latest CPUs installed to address known code vulnerabilities. At the end of the day, a VA tool is the only way to ensure that you have hardened your databases properly and that you remain compliant over time.

Three Things to Remember about Using an Assessment Tool

1. Some VA tools are stand-alone scanners and others are part of a larger database security suite. If you're implementing the full set of recommendations presented by within the Database STIG or within the CIS checklist you should consider the suite products that can also support your auditing implementation, intrusion detection implementation, separation of duties, etc.
2. VA scanners check both vulnerabilities and CPUs installed (or that should be installed) as well as the configurations of your database.
3. Some of the checks that you need to perform are at the OS level. Make sure the VA tool you choose can perform checks on file ownership, file permissions, etc.

2.3 HOWTO Create and Maintain a Secure Configuration Baseline

Once you have finished hardening your database, you have a tight configuration, but you need to ensure that it remains tight and does not degrade over time. There are two things you can do to ensure a sustained secure configuration—(1) run assessments on a scheduled basis to find new vulnerabilities as they are created, and (2) create a baseline for the configuration once you are happy with it and track any changes from this configuration using an alert that needs to be reviewed and approved. The best practice suggests that you do both of these because they complement each other.

If you have a VA tool, then running an assessment periodically is easy. All VA tools have a scheduler that allows you to test your databases every month or every week. Most VA tools also have a "diff report" that shows you changes to the assessment results between one run and another—so once you are happy with the results of a scan you can monitor only the differences in future scans.

The second set of tools is called change tracking tools. These tools create a baseline of your configurations and alert you on any change that is made. Baselines are created by computing a digest on configuration elements and then periodically recomputing them to see if there are any changes. Digests (or hash values) can be computed on files (to ensure for example that no one modifies the software files themselves), on the output of a script (e.g., the output of a script that greps certain values from sqlnet.ora), on Structured Query Language (SQL) result sets (e.g., the output of a query that checks assignment of system privileges), etc. A change tracking tool tells you when there is a deviation from the hardened configuration.

Change tracking tools will always be part of a database security implementation. These tools are required to comply with the broad hardening checklists because this is the only way you can ensure that no one replaces your files with Trojan versions. They are also the most effective way to ensure that scripts run periodically are not used as a point-of-compromise; tracking changes made to scripts is much simpler and more effective than reviewing audit trails that show what a script did.

Because you'll have access to these tools if you're responsible for database security, use them in the context of VA change tracking tools—once you've completed your hardening process use them to ensure that you don't deviate from this standard. If there are changes over time (and there always will be changes) you can reauthorize and update the baseline and track changes from that point on. Look for a VA tool that includes a change tracking tool to ensure sustained and continuous compliance.

Three Things to Remember about Creating and Maintaining a Secure Configuration Baseline

1. Change tracking tools have multiple uses within an Oracle security implementation. One of these is to create and track a secure baseline following the hardening phase. VA tools that incorporate a change tracking tool offer you more options in terms of continued compliance.
2. Baselines are generated by creating digests that uniquely identify a file or a result of a script. Any change to the underlying entity will cause the digest to change and a change report to be issued. Digests act as digital fingerprints of the underlying entity.
3. Baselines include digests for files that should not change, digests for the result of an OS script, digests for the result of a query, digests for values of environment variables or registry entries, and more.

2.4 HOWTO Understand Critical Patch Updates

Always install patches to security issues as soon as they are available from Oracle. When code vulnerabilities are discovered (and there always will be code vulnerabilities—any large piece of software has bugs), Oracle issues patches—you must install these patches to remove these vulnerabilities from your environment.

Security patches come in the form of Oracle CPUs. An Oracle CPU is a bundle of patches that are released on a quarterly basis to fix security issues. CPUs have been around since 2005 (before that there were called Security Alerts) and they come out at 1 p.m. Pacific time on the Tuesday closest to the 15th of January, April, July, and October. The fact the CPUs are released at a known date is important—it allows you to plan ahead and define change management windows accordingly. Before CPUs were used, security alerts were issued when issues were discovered and fixed. This made installing these security patches very difficult and sometimes as people were in the processing of testing one fix, another one would be announced. Knowing when the patches are published makes it easier for you to build a process around applying them.

The Oracle CPU includes fixes for all Oracle software components. One patch is released per version of the database, application server, Enterprise Manager, etc. This has changed as of 2007 with the introduction of the n-apply process (more on that later). Patches are also released for Oracle E-Business Suite, PeopleSoft, Siebel, and other applications. Patches for the database are cumulative so that the latest CPU includes fixes for all earlier CPUs unless stated otherwise.

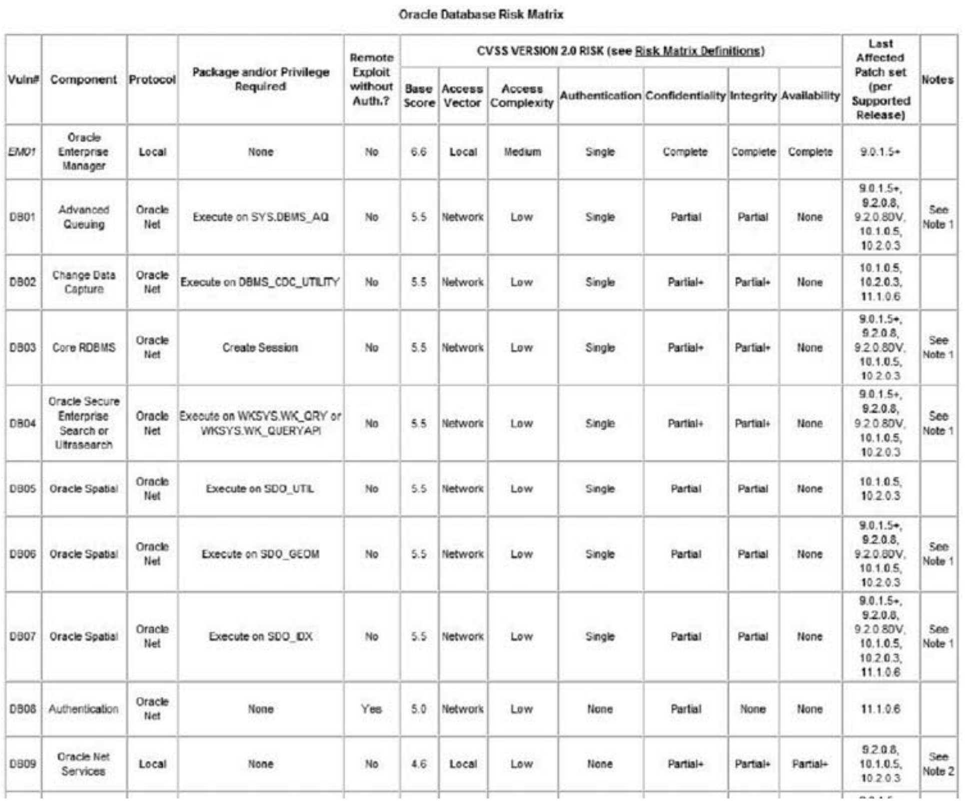
Each CPU includes a set of patches, an advisory, preinstallation notes, and release notes. The CPU advisory contains information which helps you evaluate the impact of the fixed vulnerabilities so that you may assess the criticality of issues and how quickly you need to install patches on production systems. The advisory includes a set of risk matrices—one per software system—in which Oracle reports on the risks of the discovered issues.

Reading a CPU Advisory Risk Matrix

A CPU advisory contains a database risk matrix. The risk matrix helps you understand the risk of discovered issues in terms of loss of confidentiality, loss of integrity, and loss of availability—the three dimensions that can be affected by security vulnerabilities.

Figure 2.2 shows a sample from the database risk matrix of the April 2008 CPU. The matrix summarizes the list of vulnerabilities fixed within the CPU and for each one provides information about risk. Each vulnerability is given a vulnerability number composed of four characters—the first two characters represent the system and the last two are an incremental numeric code starting from 01. Database vulnerabilities are tagged as DB##. Each CPU has its own numbering scheme so the vulnerability number is unique within a CPU. Sometimes a vulnerability in another system affects users of the Oracle database—in this case that vulnerability will be included in the database risk matrix so that the risk matrix is self-contained and you do not need to read the entire CPU if you only care about the database. As an example, in Figure 2.2 EM01 is listed within the database risk matrix even though the vulnerability is in Enterprise Manager. In such a case the vulnerability number appears in italics.

The risk matrix contains information about which component the vulnerability is in, what protocol is required to exploit the vulnerability, what package or privilege is required to exploit the vulnerability and whether an attacker can exploit the vulnerability from a remote node without



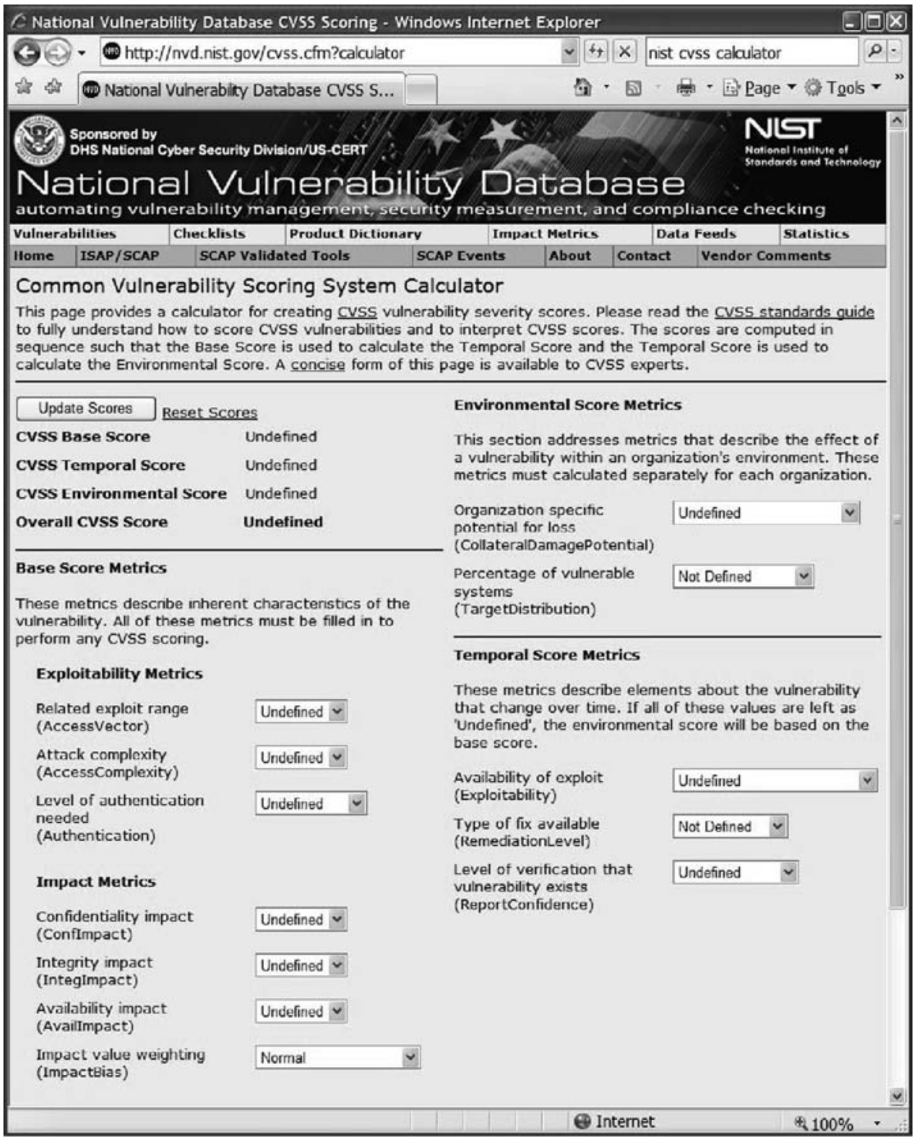
Oracle Database Risk Matrix

Vuln#	Component	Protocol	Package and/or Privilege Required	Remote Exploit without Auth.?	CVSS VERSION 2.0 RISK (see Risk Matrix Definitions)							Last Affected Patch set (per Supported Release)	Notes
					Base Score	Access Vector	Access Complexity	Authentication	Confidentiality	Integrity	Availability		
EM01	Oracle Enterprise Manager	Local	None	No	6.6	Local	Medium	Single	Complete	Complete	Complete	9.0.1.5+	
DB01	Advanced Queuing	Oracle Net	Execute on SYS.DBMS_AQ	No	5.5	Network	Low	Single	Partial	Partial	None	9.0.1.5+, 9.2.0.8, 9.2.0.8DV, 10.1.0.5, 10.2.0.3	See Note 1
DB02	Change Data Capture	Oracle Net	Execute on DBMS_CDC_UTILITY	No	5.5	Network	Low	Single	Partial+	Partial+	None	10.1.0.5, 10.2.0.3, 11.1.0.6	
DB03	Core RDBMS	Oracle Net	Create Session	No	5.5	Network	Low	Single	Partial+	Partial+	None	9.0.1.5+, 9.2.0.8, 9.2.0.8DV, 10.1.0.5, 10.2.0.3	See Note 1
DB04	Oracle Secure Enterprise Search or Ultrasearch	Oracle Net	Execute on WKSYS.WK_QRY or WKSYS.WK_QUERYAPI	No	5.5	Network	Low	Single	Partial+	Partial+	None	9.0.1.5+, 9.2.0.8, 9.2.0.8DV, 10.1.0.5, 10.2.0.3	See Note 1
DB05	Oracle Spatial	Oracle Net	Execute on SDO_UTIL	No	5.5	Network	Low	Single	Partial	Partial	None	10.1.0.5, 10.2.0.3	
DB06	Oracle Spatial	Oracle Net	Execute on SDO_GEOM	No	5.5	Network	Low	Single	Partial	Partial	None	9.0.1.5+, 9.2.0.8, 9.2.0.8DV, 10.1.0.5, 10.2.0.3	See Note 1
DB07	Oracle Spatial	Oracle Net	Execute on SDO_IDX	No	5.5	Network	Low	Single	Partial	Partial	None	9.0.1.5+, 9.2.0.8, 9.2.0.8DV, 10.1.0.5, 10.2.0.3, 11.1.0.6	See Note 1
DB08	Authentication	Oracle Net	None	Yes	5.0	Network	Low	None	Partial	None	None	11.1.0.6	
DB09	Oracle Net Services	Local	None	No	4.6	Local	Low	None	Partial+	Partial+	Partial+	9.2.0.8, 10.1.0.5, 10.2.0.3	See Note 2

Figure 2.2 Sample from a database risk matrix in a CPU.

first being authenticated. The next thing provided per vulnerability is a Common Vulnerability Scoring System (CVSS) score. CVSS is a standardized method for assessing security vulnerabilities in all systems. CVSS has been used by Oracle since October 2006—before then vulnerabilities were scored using a proprietary scoring system.

CVSS scores range between 0.0 and 10.0 with 10.0 being the worst possible score (implying the worst possible vulnerability, and the highest risk). Oracle currently uses CVSS version 2.0 to derive a base score and this score is included in the risk matrix. Scores are computed using a calculator available at nvd.nist.gov and shown in Figure 2.3. The score is computed after you enter a selection for each of the entries. When Oracle scores vulnerabilities they key in the answers to the base score


National Vulnerability Database CVSS Scoring - Windows Internet Explorer
http://nvd.nist.gov/cvss.cfm?calculator
nist cvss calculator
National Vulnerability Database CVSS S...
Page
Tools
Sponsored by DHS National Cyber Security Division/US-CERT
NIST National Institute of Standards and Technology
National Vulnerability Database
automating vulnerability management, security measurement, and compliance checking
Vulnerabilities
Checklists
Product Dictionary
Impact Metrics
Data Feeds
Statistics
Home
ISAP/SCAP
SCAP Validated Tools
SCAP Events
About
Contact
Vendor Comments
Update Scores
Reset Scores
Internet
100%

Common Vulnerability Scoring System Calculator

This page provides a calculator for creating CVSS vulnerability severity scores. Please read the CVSS standards guide to fully understand how to score CVSS vulnerabilities and to interpret CVSS scores. The scores are computed in sequence such that the Base Score is used to calculate the Temporal Score and the Temporal Score is used to calculate the Environmental Score. A concise form of this page is available to CVSS experts.

CVSS Base Score	Undefined
CVSS Temporal Score	Undefined
CVSS Environmental Score	Undefined
Overall CVSS Score	**Undefined**

Base Score Metrics

These metrics describe inherent characteristics of the vulnerability. All of these metrics must be filled in to perform any CVSS scoring.

Exploitability Metrics

Related exploit range (AccessVector)	Undefined
Attack complexity (AccessComplexity)	Undefined
Level of authentication needed (Authentication)	Undefined

Impact Metrics

Confidentiality impact (ConfImpact)	Undefined
Integrity impact (IntegImpact)	Undefined
Availability impact (AvailImpact)	Undefined
Impact value weighting (ImpactBias)	Normal

Environmental Score Metrics

This section addresses metrics that describe the effect of a vulnerability within an organization's environment. These metrics must calculated separately for each organization.

Organization specific potential for loss (CollateralDamagePotential)	Undefined
Percentage of vulnerable systems (TargetDistribution)	Not Defined

Temporal Score Metrics

These metrics describe elements about the vulnerability that change over time. If all of these values are left as 'Undefined', the environmental score will be based on the base score.

Availability of exploit (Exploitability)	Undefined
Type of fix available (RemediationLevel)	Not Defined
Level of verification that vulnerability exists (ReportConfidence)	Undefined

Figure 2.3 NIST CVSS calculator.

metrics and get a base score which then goes into the CPU risk matrix. Note that although the CVSS scores desire to be a single number through which you can tell right away whether it is critical or not, CVSS ratings depend on Oracle's interpretation of the problem. To fully understand how Oracle fills in the entries for computing CVSS scores see Metalink Note 394487.1.

It is especially important to understand how Oracle interprets the effect on confidentiality, availability, and integrity. The CVSS calculator allows you to enter one of three values—None, Partial, or Complete. Complete is defined to mean that the impact is to the "whole system." The question is what exactly this means. One interpretation of Complete can be that the impact is to all Oracle software running on a system and the other can be that the impact is to all software running on the system—OS and all. Oracle chooses to use the latter interpretation. This means for example that if the vulnerability affects all data within the database—all tables in all schemas—the CVSS score is still going to be based on a "Partial" selection in the calculator. Having chosen the first interpretation would have created higher CVSS scores. This is not uncommon and is the way most vendors interpret CVSS levels but you should understand this when you determine what your threshold for risk is.

To distinguish between cases in which the impact can be limited versus wide (which were the terms used before adoption of CVSS), Oracle introduces another level called Partial + (see Figure 2.2). A vulnerability that affects a limited set of resources (e.g., a specific database table) will have Partial in the risk matrix. If the vulnerability affects a wide range of resources (e.g., all tables in the database) then the impact is logged as Partial +. Note that this does not impact the CVSS score—the score in both cases is computed using Partial! It is therefore important to look at the entries in the risk matrix and not only the CVSS score.

Database n-Apply CPUs

Until 2007 CPUs included one monolithic patch for the database every quarter. It was impossible to install a subset of the fixes. This changed with the CPU of July 2007 when the n-apply patch format was introduced to CPUs. n-apply CPUs have the following benefits:

1. Customized patch conflict resolution.
2. Elimination of rollbacks and reinstallation of CPU patches that are already installed to limit downtime. CPUs are still cumulative but the installation process has been improved.
3. Ability to install only parts of the CPU fixes rather than have to install the whole CPU.

An n-apply CPU is a zip file that contains molecules and are installed using opatch. Each molecule is a group of security fixes. A molecule is an independent patch that does not conflict with any of the other molecules within the CPU.

Five Things to Remember about CPUs

1. CPUs are released every three months at specific dates so that you can plan ahead for testing and deploying fixes.
2. CPUs include security fixes to discovered vulnerabilities. It is very important to apply security fixes because this is the best way to protect yourself from attacks that exploit such vulnerabilities. Note that once a CPU has been released it is easier for an attacker to find the vulnerability because there is some information (e.g., the component) listed per vulnerability—therefore, apply CPUs as soon as possible while going through a standard testing and change management process.

(*continued*)

(continued)

3. CPUs include a risk matrix that allows you to determine how critical these fixes are for your environments. Risk matrices list which components are affected and how severe the issue is—all are there to help you decide whether or not you can tolerate the risk if you can't immediately apply the CPU.
4. CPUs are cumulative—if you apply the latest CPU you have included all fixes for all previous vulnerabilities too.
5. The new n-apply CPU packaging allows you to deploy fixes to only some vulnerabilities versus the old format which was delivered as a single patch.

2.5 HOWTO Sanitize Data for Test

As part of the section on Database Software Development the Database STIG states:

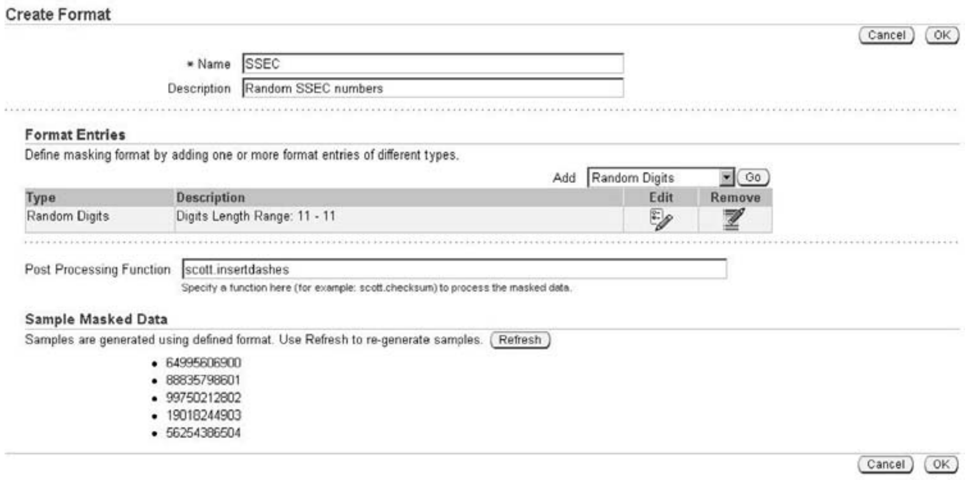
> (DG0076: CAT II) The DBA will ensure that export data from a production database used to populate a development database has all sensitive data such as payroll data or personal information, etc., removed or modified prior to import to the development database.

Production databases usually have better access controls and are monitored with higher scrutiny than development databases. This only makes sense if you assume that the sensitivity levels of data in development and test databases are lower than those of production servers. Developers have access to development and test databases but usually not to production servers (outside a "break glass," or emergency maintenance event). Therefore, you must sanitize data before you can load it to development servers.

Sanitizing data is hard. Not only do you need to know where all your sensitive data resides, you also need to change a lot of data in a way that does not invalidate your application and your tests. You can't change data randomly. If you have foreign keys (whether they are constraints in the database or managed by the application) you need to make sure that keys are preserved and that all references stay intact. Some fields encode application logic. For example, many account numbers follow a certain scheme in which there is a checksum or some algorithm for generating numbers—not any random set of digits is a valid account number. Another example is credit card numbers. When you sanitize data you need to preserve this property or your application will break. And finally, you need to make sure that after you sanitize the data, columns used for indexes maintain a statistical distribution which is close to that of the production data. Otherwise, performance tests on the sanitized data may not be indicative of the performance you will have on the production data. All in all, sanitizing test data is a hard problem to solve—that is why many development organizations don't do it and simply use the production data as-is. This is in violation of any security guideline, and tools do exist to help you accomplish this task.

The first tool available to you is the Data Masking option in Enterprise Manager. Follow these steps once you have enabled the data masking option in Enterprise Management Grid Control:

Step 1: Log onto EM
Step 2: Click on the Targets tab and the Databases subtab.
Step 3: Select the database where you want to mask sensitive data.



Create Format

* Name	SSEC
Description	Random SSEC numbers

Format Entries

Define masking format by adding one or more format entries of different types.

Add: Random Digits

Type	Description	Edit	Remove
Random Digits	Digits Length Range: 11 - 11		

Post Processing Function: scott.insertdashes

Specify a function here (for example: scott.checksum) to process the masked data.

Sample Masked Data

Samples are generated using defined format. Use Refresh to re-generate samples.

- 64995606900
- 88835798601
- 99750212802
- 19018244903
- 56254386504

Figure 2.4 Defining a masking format.

Step 4: Click on the Administration link. At the bottom right is a Data Masking section. The Definitions link lets you set the masking rules. The Format Library link lets you build up a library of data masking formats. A data masking format defines how you want to mask sensitive data. For example, you can use random numbers or random characters. For more advanced functions (and usually you need more advanced obfuscation techniques for real applications) you need to build PL/SQL routines. For example, if you want to test an application you probably will want data that may have indexes with the same statistical distribution as the real data—using random characters will certainly not preserve cardinalities.

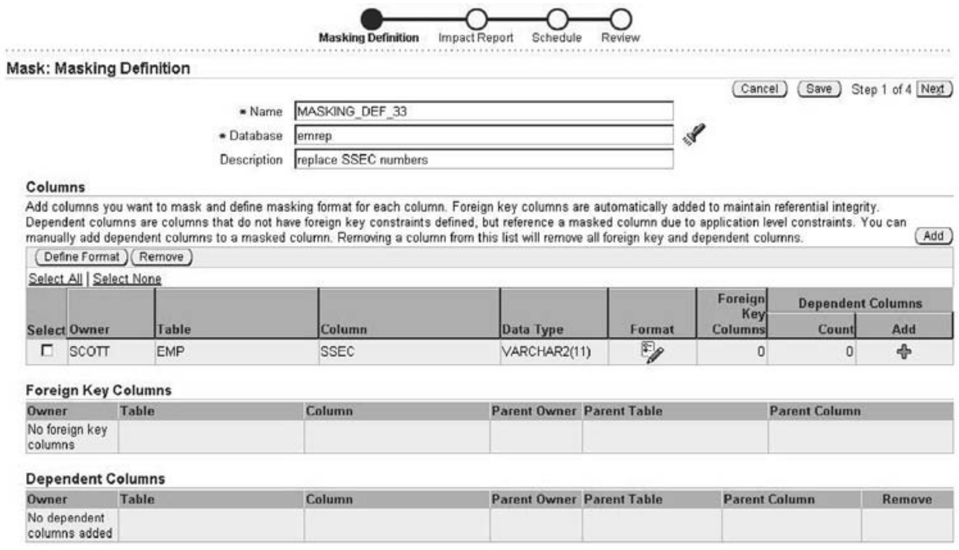
Step 5: Click on the Format Library link. This takes you to a page with a list of formats available to you. As this product matures, there will be many predefined formats for the most common identity patterns but for now click on Create.

Step 6: Figure 2.4 shows a format for masking social security numbers. These numbers have a pattern of [0–9]{3}-[0–9]{2}-[0–9]{4}. In this case you can choose Random digits from the drop down and click on Go. Enter 1 as the start and 11 as the end to ask Oracle to create 11 random digits for you. Click on OK. Then, you'll have to call a PL/SQL procedure to put in the dashes in locations 4 and 7, so enter the name of your procedure and click OK.

Step 7: Click on the Masking Definitions breadcrumb at the top to take you back to the masking definitions screen. Click on Mask to create a masking job. Give the job a name and select the database where the sensitive data resides.

Step 8: Click on Add to define which column to mask and how to mask it. Put in the schema name and click on the search icon. Select the sensitive column from the list (or multiple columns). Click on Define Format and Add.

Step 9: Click on Import From Library because you have already created the masking format. Select your format and click Import. You've now selected where the sensitive data is and how to mask it as shown in Figure 2.5. Click on Next.



Mask: Masking Definition

* Name: MASKING_DEF_33
* Database: emrep
Description: replace SSEC numbers

Columns

Add columns you want to mask and define masking format for each column. Foreign key columns are automatically added to maintain referential integrity. Dependent columns are columns that do not have foreign key constraints defined, but reference a masked column due to application level constraints. You can manually add dependent columns to a masked column. Removing a column from this list will remove all foreign key and dependent columns.

Select	Owner	Table	Column	Data Type	Format	Foreign Key Columns	Dependent Columns Count	Dependent Columns Add
☐	SCOTT	EMP	SSEC	VARCHAR2(11)		0	0	✚

Foreign Key Columns

Owner	Table	Column	Parent Owner	Parent Table	Parent Column
No foreign key columns					

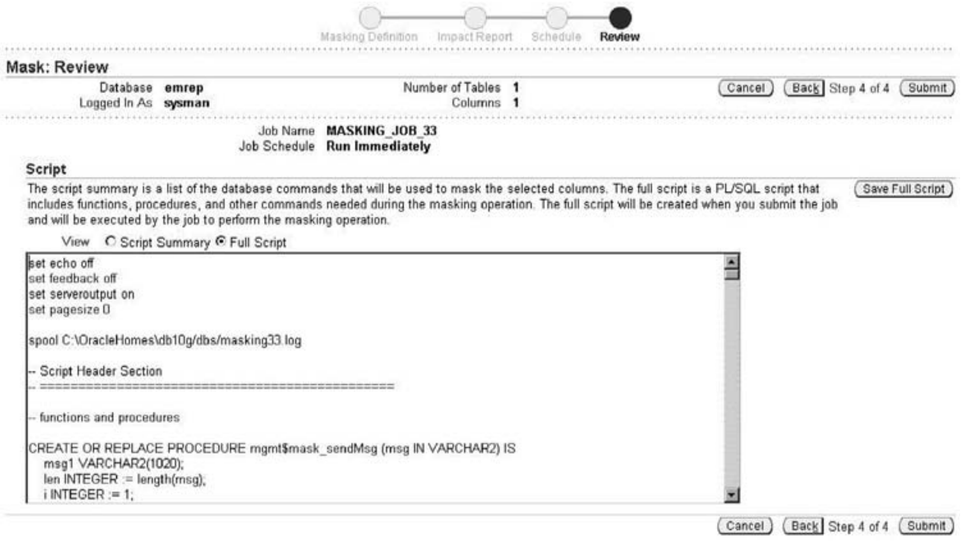
Dependent Columns

Owner	Table	Column	Parent Owner	Parent Table	Parent Column	Remove
No dependent columns added						

Figure 2.5 Defining which sensitive data to mask.

Step 10: The script is generated. Review the generation information and click Next. Enter the host credentials where the script will be stored. Enter a schedule if the job should be scheduled or select Immediately. Click on Next.

Step 11: Oracle produces the script and you can review it as shown in Figure 2.6. Submit the masking job. You can then review the status (and possible errors) of masking jobs as shown in Figure 2.7.



Mask: Review

Database **emrep**
Logged In As **sysman**
Number of Tables **1**
Columns **1**

Job Name **MASKING_JOB_33**
Job Schedule **Run Immediately**

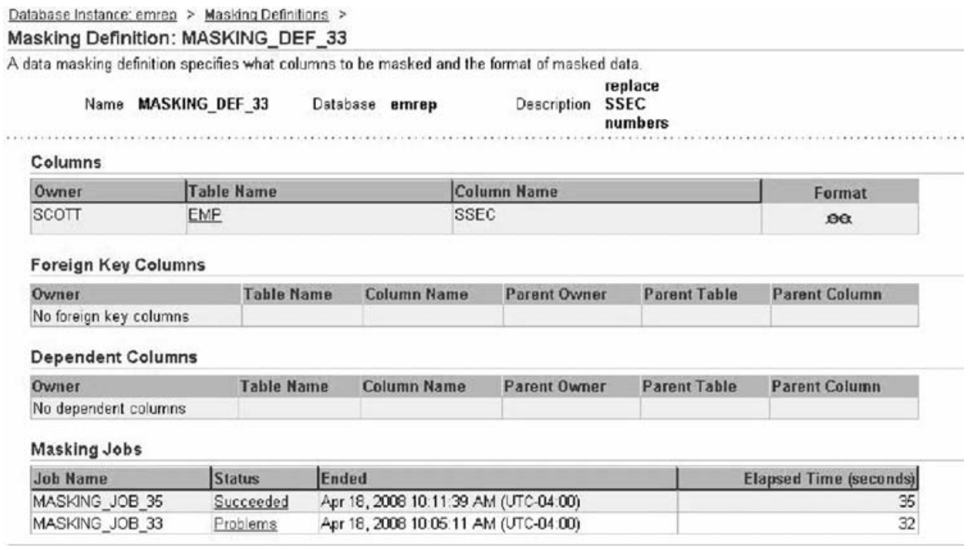
Script

The script summary is a list of the database commands that will be used to mask the selected columns. The full script is a PL/SQL script that includes functions, procedures, and other commands needed during the masking operation. The full script will be created when you submit the job and will be executed by the job to perform the masking operation.

View ( ) Script Summary [✓ Full Script]

```
set echo off
set feedback off
set serveroutput on
set pagesize 0

spool C:\OracleHomes\db10g/dbs/masking33.log

-- Script Header Section
-- ==============================================

-- functions and procedures

CREATE OR REPLACE PROCEDURE mgmt$mask_sendMsg (msg IN VARCHAR2) IS
   msg1 VARCHAR2(1020);
   len INTEGER := length(msg);
   i INTEGER := 1;
```

Figure 2.6 Reviewing the masking script and submitting the masking job.



Database Instance: emrep > Masking Definitions >

Masking Definition: MASKING_DEF_33

A data masking definition specifies what columns to be masked and the format of masked data.

Name	**MASKING_DEF_33**	Database	**emrep**	Description	**replace SSEC numbers**

Columns

Owner	Table Name	Column Name	Format
SCOTT	EMP	SSEC	👓

Foreign Key Columns

Owner	Table Name	Column Name	Parent Owner	Parent Table	Parent Column
No foreign key columns					

Dependent Columns

Owner	Table Name	Column Name	Parent Owner	Parent Table	Parent Column
No dependent columns					

Masking Jobs

Job Name	Status	Ended	Elapsed Time (seconds)
MASKING_JOB_35	Succeeded	Apr 18, 2008 10:11:39 AM (UTC-04:00)	35
MASKING_JOB_33	Problems	Apr 18, 2008 10:05:11 AM (UTC-04:00)	32

Figure 2.7 Reviewing the status of masking jobs.

As an example, if you use random digits and a PL/SQL procedure that adds the dashes, and if the data looks like:

```
SQL> select * from emp;

    EMPNO ENAME   JOB            MGR HIREDATE      SAL      COMM DEPTNO SSEC
--------- ------- ---------  -------- -------- -------- --------- ------ -----------
     7499  ALLEN  SALESMAN      7698 20-FEB-81      100       300     30 111-11-1111
     7876  ADAMS  CLERK         7788 23-MAY-87      101                20 222-22-2222
     7844  TURNER SALESMAN      7698 08-SEP-81      102         0     30 333-33-3333
     7782  CLARK  MANAGER       7839 09-JUN-81      103                10 444-44-4444
     7566  JONES  MANAGER       7839 02-APR-81      104                20 000-00-0000
     7839  KING   PRESIDENT          17-NOV-81      105                10 888-88-8888
     7900  JAMES  CLERK         7698 03-DEC-81      106                30 777-77-7777
     7788  SCOTT  ANALYST       7566 19-APR-87      108                20 555-55-5555
     7369  SMITH  CLERK         7902 17-DEC-80      110                20 999-99-9999
     7934  MILLER CLERK         7782 23-JAN-82      111                10 666-66-6666
```

Then, your masked data will look like:

```
SQL> select * from emp;

    EMPNO ENAME   JOB            MGR HIREDATE      SAL      COMM DEPTNO  SSEC
--------- ------- ---------  -------- -------- -------- --------- ------  -----------
     7844 TURNER  SALESMAN      7698 08-SEP-81      102         0     30  745-80-4600
     7782 CLARK   MANAGER       7839 09-JUN-81      103                10  857-87-7301
     7934 MILLER  CLERK         7782 23-JAN-82      111                10  522-91-5302
     7566 JONES   MANAGER       7839 02-APR-81      104                20  658-07-6603
     7839 KING    PRESIDENT          17-NOV-81      105                10  886-14-8204
     7788 SCOTT   ANALYST       7566 19-APR-87      108                20  661-97-3505
```

```
7369 SMITH   CLERK          7902 17-DEC-80     110                20  995-32-6006
7900 JAMES   CLERK          7698 03-DEC-81     106                30  821-79-0807
7876 ADAMS   CLERK          7788 23-MAY-87     101                20  420-95-5508
7499 ALLEN   SALESMAN       7698 20-FEB-81     100        300     30  375-11-5609
```

The Data Masking option is a new product and therefore has only rudimentary formats. With time the masking format library will grow and you will get logic-preserving and statistically preserving formats. However, there is a large set of third-party tools that perform this function and have a mature set of operators and formats. Examples include Princeton Softech (now IBM Optim), Application, Solix, and HP/Outerbay.

Three Things to Remember about Sanitizing Test Data

1. You must sanitize sensitive information if you generate test data by copying real data from production systems.
2. Sanitizing data is far from trivial—you cannot simply replace data with random strings or numbers. You have to preserve application logic which is often coded into data and you must preserve statistical distribution for performance tests to be valid.
3. You should use tools to sanitize data—use either the data making pack that is now part of Enterprise Management Grid Control or use third-party tools.

2.6 Discussion: Defense in Depth

All modern information security is founded on a concept called defense in depth. Defense in depth involves multiple layers of defense that increase the cost of an attack and places multiple barriers between an attacker and your computing resources. Multiple techniques and systems help mitigate the impact when one component of the defense is compromised or circumvented. The deeper an attacker tries to go the harder it gets. In addition to protecting your resources better, by the mere fact that there are multiple layers, defense in depth naturally provides areas in which you can put systems that can monitor and identify intrusions. This can often buy time to detect and respond to a breach and reduce its impact.

The term "defense in depth" is derived from a military strategy called defense in depth (also known as deep defense or elastic defense). This military strategy seeks to delay the advance of an attacker rather that prevent the advance. This buys time and causes enemy casualties—so rather than defeating an attacker with a single line, defense in depth relies on the tendency of an attack to lose momentum over a period of time or when it has to cover a larger area. The defender can yield some lines and territories causing the attacker to spread. This allows the defender to identify and mount counterattacks on the attacker's weak points.

Defense in depth is considered the only viable strategy for information systems. The reason is that there is no such thing as a perfect security layer. Anything can be cracked and every system has bugs. Moreover—the quality of both security and attacks are highly correlated with their cost. The goal of a good IT security implementation is to create an environment in which an attack will cost too much to be worth it—and do it all within a reasonable budget. Therefore, relying on a single super-duper security system just does not work. Instead, build multiple good (but perhaps not perfect) security layers. This has been described in an excellent paper called "Defense in Depth—A practical strategy for achieving information assurance in today's highly networked environments" published by the National Security Agency (NSA)—http://www.nsa.gov/snac/support/defenseindepth.pdf.

Securing Oracle environments must follow the same strategy. The hardening checklists clearly discuss multiple types of activities that, if done in concert, implement a best practice in Oracle security. Remember that the security techniques available within the database can (and should) be augmented with security systems sitting outside the database—be they network security systems, database activity monitoring systems or host security. Always think of the main thesis of defense in depth—don't rely on one layer only.

Chapter 3

Securing the Listener

Most Oracle activity occurs over the network. Oracle clients connect to oracle servers over a communication protocol such as Transmission Control Protocol/Internet Protocol (TCP/IP). Oracle clients and servers communicate using an Oracle protocol called the Transparent Network Substrate (TNS). TNS packets travel as TCP/IP payload. Setting up connections to the database is performed by the TNS listener—or listener for short. The listener manages network connections and as such, is the entry point to most database connections. It must be secured properly. Without the listener, your database will not be serving applications. Therefore, one important aspect of securing the listener involves ensuring that no one hijacks your listener, that no one shuts down your listener, and ensuring that no one cripples your listener by making it do a lot of work that you don't want it to do (e.g., writing trace information).

The listener has a number of components:

1. The $ORACLE_HOME/bin/tnslsnr executable itself.
2. The Listener Control Utility ($ORACLE_HOME/bin/lsnrctl) used to administer the listener, configure logging and tracing, set passwords, etc.
3. The $ORACLE_HOME/network/admin/sqlnet.ora configuration file which defines the networking parameters used by the listener.
4. The $ORACLE_HOME/network/admin/listener.ora configuration file which defines the IP addresses and ports that the listener should be listening on, the instances for which it is listening using a list of services or SIDs, and additional configuration options for that listener—options such as passwords, log levels, etc. You can change these options either by directly editing the listener.ora file or by using lsnrctl (which modifies this file). In both cases remember that you may need to bounce the listener for the changes to take effect.
5. The $ORACLE_HOME/network/admin/tnsnames.ora configuration file used to map TNS service names to connection descriptors so that a database client can use a convenient name to connect to the listener rather than having to use IP addresses, ports, protocol types, etc.

The listener is a highly privileged process and has an important role involving the execution of new processes at the operating system (OS) level, the loading of dynamically linked libraries (DLLs), etc.