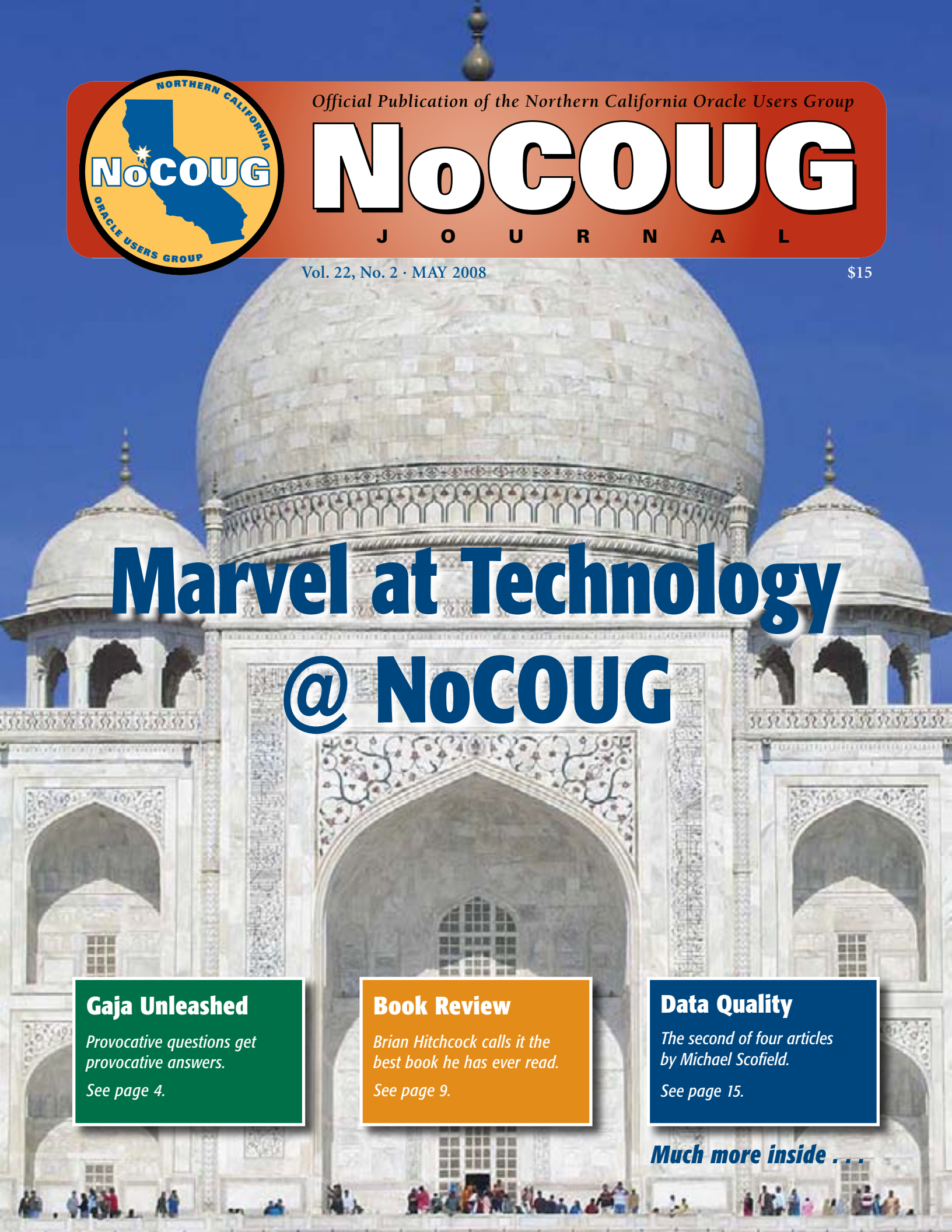
Official Publication of the Northern California Oracle Users Group

NoCOUG JOURNAL

Vol. 22, No. 2 · MAY 2008 $15



Marvel at Technology @ NoCOUG

Gaja Unleashed

Provocative questions get provocative answers.

See page 4.

Book Review

Brian Hitchcock calls it the best book he has ever read.

See page 9.

Data Quality

The second of four articles by Michael Scofield.

See page 15.

Much more inside . . .

Practical Oracle Security: Your Unauthorized Guide to Relational Database Security

A Book Review by Brian Hitchcock

Details

Authors: Aaron Ingram and Josh Shaul
ISBN: 978-1-59749-198-3
Pages: 432
Year of Publication: 2007
Edition: 1
Price: $27.95
Publisher: Syngress

Overall Review

This book is among the best Oracle books I've ever read. It was quick and I learned many things I can apply to my work immediately. It didn't waste my time going over the basics (what is a database?) and got to the point immediately. It doesn't pad the content to get to 600 pages, which seems to be required now for most technical books. The authors are very good at conveying useful information rather than just trying to impress you.

Summary

Overall review: Excellent, the most useful book I've ever read on Oracle technology.

Target audience: Almost anyone that supports or manages Oracle databases.

Would you recommend to others: Yes.

Who will get the most from this book: DBAs and others managing the security aspects of Oracle databases.

Is this book platform specific: No.

Why did I obtain this book: I've been working on security issues in Oracle databases and Oracle applications for some time and have to learn about and deal with security issues all the time.

Chapter 1—Oracle Security: The Big Picture

This overview of Oracle database security history and issues covers a lot of ground. An excellent explanation is given of why the "table or view does not exist" error message actually is a good thing. The section on privilege controls explains where the CONNECT, RESOURCE, and DBA roles came from and why they were created in the first place. This provides valuable perspective on just how much has changed from the early versions of Oracle to the present security concerns. Also explained is why the introduction of SQL*Net in version 5 was the beginning of many of the security issues we still see today. With 8*i*, the authors explain how Oracle hacking went mainstream.

Other sections cover the history of auditing features in Oracle and how Fine Grained Auditing (FGA) in 10*g* was improved to audit DML. The PASSWORD_VERIFY_FUNCTION is discussed. I have experience with this, because when we applied Oracle Applications patches to support this, it broke our Oracle Discoverer reporting environment for Oracle Applications. As with all things, when you patch to fix one thing (in this case to support the password verify function), you may be breaking other things, which will require more patching and testing.

I had never heard of the Bell-LaPadula security model, but the description is interesting and, if nothing else, provides good job -interview trivia. The explanation of Virtual Private Database is very good. I didn't really understand what this was until I read this. Similarly, I didn't know that Label Security could be applied at either the schema or individual table level, offering complete flexibility.

There is also a discussion of the tradeoffs that come with more security features. Oracle 10*g* has lots of security fea-

"This book is among the best Oracle books I've ever read. It was quick and I learned many things I can apply to my work immediately. It didn't waste my time going over the basics (what is a database?) and got to the point immediately. It doesn't pad the content to get to 600 pages, which seems to be required now for most technical books."

tures; this means complexity, which means DBA overhead. As the authors put it, "Oracle is likely the most secure and most vulnerable database in existence today."

I highly recommend the next two sections, "The Regulatory Environment," which provides very good info describing the major pieces of legislation driving many of the current security projects, and "Major Data Theft Incidents," which fills in a lot of details about what has been reported in the popular media but not really explained.

In the section "Appropriate Security for Each Class of Database System," I disagree with the authors when they discuss Development, QA, and test databases. These are often a complete copy of a critical production system and therefore are just as sensitive from a security perspective as the primary system. As the authors point out in other parts of the book, a hacker is just as happy—probably happier actually—to get the sensitive data from a non-production system where no one is watching. Further, even your backup tapes, which are a complete copy of your databases, should be viewed as just as sensitive as the critical systems that were the source of the backups.

In the Frequently Asked Questions section, the authors tell us that 70% of attacks involve an insider. What about consultants or a remote DBA that works on your critical systems for a brief time? Note that even if someone only has full access for a brief time, they can leave security holes that won't be apparent without careful scrutiny. Worse, what I've seen is that they may do things while they have access that open huge security holes just to get the job done on time. For example, granting SELECT ANY TABLE to give developers "read-only" access to the database. While the consultants are gone, they left behind a security issue that makes it easy for insiders to access sensitive data.

Chapter 2—File System

This chapter explains how database users can get access to the filesystem and why this is a security concern. The files in the filesystem are broken down into four categories: data, software, configuration, and logs; the chapter gives specific advice on securing each. A minor issue: the example showing how to list all datafiles doesn't include temp files supporting temporary tablespaces, yet a few pages later we're told how important it is to secure these files.

A good example of a seemingly innocent operation causing a security problem is given. When dropping a tablespace, Oracle doesn't remove the associated datafile(s) from the filesystem. This leaves the datafile(s) on disk where they could be accessed. While not part of the filesystems, the backups made of filesystems could give a hacker access to passwords, which then give access to live data. Backups need to be treated with same concern as the live data. Another related issue is that test machines often contain authentication info that is copied from production.

One security measure suggested is to compute and store hash values for all software and configuration files, so that any changes to these files could be detected. This may be a good idea, but who has the time and resources to implement and maintain this? Similarly, we are told to revoke permissions on Oracle software files, but how do we know the impact this may have on the operation of the software? Encrypting data is a good idea, but what happens if the encryption key is lost? There is a tradeoff between security and the cost of supporting a more secure environment.

"A hacker is just as happy—probably happier actually—to get the sensitive data from a non-production system where no one is watching. Further, even your backup tapes, which are a complete copy of your databases, should be viewed as just as sensitive as the critical systems that were the source of the backups."

Chapter 3—TNS Listener Security

The Oracle listener is the source of many security issues, and the authors tell us that while things are better than they were, there is still much to be concerned with. I didn't know about tnscmd and all the bad things you can do with it. 10*g* listener improvements are discussed but these improvements can all be turned off.

I found the following to be a surprise: If you remotely administer a database, the listener password is sent over the network in plain text, making it easy for anyone who can monitor the network to get the listener password. Further, before 10*g*, the listener password hash would be accepted by the listener as the password.

While not a direct attack on the database, the point is made that a denial of service (DoS) attack against the database listener is pretty much the same thing as a DoS attack against the database.

Given how many security issues are discussed with respect to the listener, the more general point is made that making software work is different from making it secure. This is another way of saying "we met the release date and anything else will get fixed later." Incomplete testing of applications is one of the main reasons why hackers can remain in business. This applies to both the Oracle software itself and the applications you build on top of it.

The authors suggest the best thing to do for listener security is upgrade to 10*g*. Perhaps this is a good idea, but is it really practical? Can most Oracle customers that are not on 10*g* now quickly upgrade to 10*g* just for the listener security improvements?

Another example of this book providing really useful information is the discussion of ADMIN_RESTRICTIONS. Oracle provided a good new security feature in a patch, but even after the patch is applied, these new features had to be enabled by the DBA. How many systems have this patch but don't have the benefit of these security improvements? Yes, the security improvements were patched into the system, but no one remembered to activate them. Again, better security requires more (and effective) resources.

A reference is given to a series of articles covering how to interpret the listener log file. I have read these articles and recommend them, although I haven't taken the SQL and used it on my own systems. The information in these articles is very worthwhile and addresses issues I've had for a long time. I've never been able to get much from the listener logs. Like the rest of this book, this is practical information that I can use on the job and that I haven't seen elsewhere.

Valid node checking is a lesser-known but more useful security feature of the listener. Note that Oracle thinks highly of this feature. When you install Oracle Applications 11*i*, by default this feature is configured so that the only host that can contact the database is the applications tier host. This feature should be used more often, as it prevents a lot of security issues by simply restricting the IP addresses that connection requests can come from.

On a practical note, while reviewing listener security issues and how to fix them, recall that the listener may use a range of ports. While the initial communication is done on the port assigned to the listener, after that the user connection to the database is handed off to one of a range of port numbers. This can cause issues if you have firewalls or other systems that only allow the listener to communicate on a specified port number.

Chapter 4—Managing Default Accounts

This chapter begins with some worrying observations. The success rates of breaking into a production Oracle database using default accounts and passwords are incredibly high. Default passwords are the most powerful and easiest to exploit vulnerability. Online lists of Oracle default accounts cover up to 600 combinations covering multiple versions of Oracle. 11*g* brings case-sensitive passwords as the default, which means case sensitivity was not the default for versions before 11*g*.

"The success rates of breaking into a production Oracle database using default accounts and passwords are incredibly high. Default passwords are the most powerful and easiest to exploit vulnerability."

With 10*g* all default accounts are locked at install except for SYS and SYSTEM, which no longer have default passwords. Note that all the other default accounts still have default passwords but they are locked at install. However, if they are unlocked, the default passwords will still be a problem.

The chapter contains Default Accounts lists that contain many that I had never been aware of—for example, ADAMS. We are advised to remove many of these default accounts. While it is good advice, readers should be told how to determine the impact of removing an account and how to re-create that account if needed in the future.

The DBSNMP account is discussed in detail with respect to Oracle Enterprise Manager (OEM). The advice is to change the account that OEM uses and then lock down the DBSNMP account. I would like to see more details of all the steps involved. Oracle tells us to lock and expire default accounts that we don't need. While this makes sense, it's too easy for one or more of these accounts to be unlocked. If this happens and the

default password isn't changed, the issue of default accounts and default passwords comes up again.

The authors recommend password management tools to generate and manage passwords. This is needed because really secure passwords are very difficult to remember. For example, I won't be able to remember a password that is a string of random characters. The authors don't offer advice on how secure these password management tools are. How do we know that we aren't making things worse by putting all our passwords in one place? How do we balance the need for secure (hard-to-remember) passwords against practicality if we can't use a password management tool?

I found the explanation of what a hash is to be very useful.

The section defining impossible passwords is great. I had seen this described elsewhere before but didn't know what it meant. This book was worth reading for this information alone. I also think this should have been presented earlier when default accounts were first being discussed. For me, setting an impossible password is a better choice for most default accounts where I'm not sure what the impact would be of removing that account. In the interest of time, I think an impossible password would be a better choice than removing accounts.

We're told to run a default password scan monthly, but I don't know who has the resources for this. Note that while the authors cover commercial products, they also tell us about the free default password scanner available from Oracle.

Chapter 5—PUBLIC Privileges

I have never understood what PUBLIC was. Again, this book was worth reading for this information alone, and this is another example of the valuable information I learned. PUBLIC is neither a Role nor a User. This chapter offers a very good explanation of not only what PUBLIC is, but also the history of how it came to be and how it was never fully realized.

We are told to use roles instead of granting permissions to PUBLIC. The sensible use of PUBLIC is also described. PUBLIC is granted access to SELECT from any tables or views that are prefixed with USER or ALL. I didn't know that. It can be challenging to remove system privileges from PUBLIC. This can affect all users of the database. Do not grant roles to PUBLIC. A password for a role is described but I would like to have seen more detail on how to create this password.

You need to review each database for any privileges that have been granted with the ADMIN option, since this allows a user to grant the privilege to other users. For system privileges, you need to check if any were granted with the GRANTABLE option, which means the grantee can grant the privilege to someone else. Note that DBA_TAB_PRIVS has a column TABLE_NAME, but this can refer to the name of a table as well as other objects such as views, stored procedures, and functions.

An explanation of SQL92_SECURITY is provided. I didn't know anything about this and it was very interesting and useful. It turns out that without this feature enabled, a user who doesn't have SELECT but does have UPDATE can see the data in the table as if they *did* have SELECT privilege.

> ***"Don't ever grant a system privilege that contains ALL in the name . . . Granting SELECT ANY TABLE to database users allows users to select the password hash of any user and use external password cracking software to determine the password."***

The example given is very good. The security impact of backwards compatibility is highlighted with the explanation of why 07_DICTIONARY_ACCESSIBILITY exists and what it does.

Very specific advice is given to remove specific permissions from sensitive packages that are granted by default. We are told that patching the database in the future may actually reenable these grants. You should, in a perfect world, check for these issues regularly.

The discussion of how to use UTL_FILE to break into a database is very good. And the fact that Oracle continues to grant privileges on UTL_FILE to PUBLIC is hard to believe, but true. We are told that using DIRECTORY objects is more secure than UTL_FILE, but we also must make sure that the CREATE ANY DIRECTORY is only granted to those who really need it.

The authors are very clear. Don't ever grant system privileges to PUBLIC. Similarly, don't ever grant a system privilege that contains ALL in the name. The explanation of exactly how this can cause big security issues is very good. A list of specific system privileges that PUBLIC must not have is listed. The description of how each privilege could be used to cause security issues is fascinating. My favorite is using DROP USER as a DoS attack.

Chapter 6—Software Updates

This chapter focuses on the Critical Patch Update (CPU) patches that Oracle releases quarterly. Previously Oracle released security alerts (patches) whenever needed to fix severe problems. Note that CPU patches don't address all known security issues. The patches generally fix the most serious issues that can be patched. This means that even if you are current for all CPU patches, your system(s) still may not be patched for all known security issues.

The discussion of the flaws in CPU patching is excellent. Also explained is the methodology used to rank security issues and how this ranking process can be gamed to make some security issues seem less serious than they may be. A detailed discussion of how software development is managed gives great insight into how security issues get started and perpetuated in commercial software. I had no idea.

An excellent example of an issue that appears to have been fixed by CPU patching is a privilege escalation attack where the OBJECT_TYPE can be used to execute code with SYS privileges. There are issues that get "fixed" over and over, indicating that the method of developing the fix is not very effective. A specific example is given where a CPU patch ex-

plained a new vulnerability and (in theory) fixed it, but in reality it was six months later when the effective patch was released. For those six months, those that applied the CPU patches on time were exposed to a vulnerability that they had every reason to think they were protected from.

As always, it is recommended to always be current on CPU patches. This is sound advice but it ignores the reality that not all organizations have the resources to assign one or more persons to the CPU patching effort. It is my opinion that to really be on top of all CPU patches and to be constantly planning, patching, testing and releasing into multiple environments would indeed take one or more full-time experienced Oracle DBAs.

In the discussion of planning the CPU patching task, we are told that over time, the planning process will become easier as you can reuse the plan from quarter to quarter. I disagree, at least partially. Patching can be very different from patch to patch. You can reuse the high-level plan but the detailed planning will be different each time. Further, deciding how much testing must be done after a patch can take considerable time as well, since different patches will patch different Oracle software components. This is especially true where you have Oracle Applications running in addition to stand-alone databases.

This chapter also includes a description of molecules and "napply" technology that allows applying a subset of a CPU patch set. This became available as of July 2007. I had not heard of this before.

While planning the patch process, the use of multiple test systems and clones are advised. This assumes you have lots and lots of hardware and people to run it all. It is also hard to have a long enough downtime where you have time to patch, test the patch, and completely roll back the patch before the environment is needed again. Another good suggestion that is hard to do in the real world is to patch everything on the server that relates to Oracle. For a big server running many applications for different organizations all with different security requirements, this can be very hard to accomplish.

Chapter 7—Passwords and Password Controls

The discussion of what makes a password weak covers how Oracle stores the password hash, which is good, but this also points out the dangers of granting SELECT ANY TABLE to database users. This allows users to select the password hash of any user and use external password cracking software to determine the password.

I was interested to learn that brute force password cracking isn't practical for passwords that are longer than 6 characters. This helps me understand why it is frequently required that passwords have a minimum of 8 characters. The point is made again that password management is just as important in non-production systems that are copies of critical systems. In reality, most organizations ignore this issue altogether.

We are told that there is no reason to disable account lockout in any Oracle database. Account lockout means that after a specified number of login failures the account is automatically locked. This sounds good, but you need to be aware that for an account used by an application, if the password gets changed, the application may get locked out if it is still trying to use the old password and this can affect all the users of the application.

Specific recommendations are given for each of the password controls such as FAILED_LOGIN_ATTEMPTS, PASSWORD_LIFETIME etc. The discussion of the issues with remote OS authentication is very good. Also valuable is knowing that the OS authentication prefix must not be NULL. I didn't know about either of these issues.

The authors point out that some security-related tasks may conflict. The example given is account lockout and password scanning software. The software will try to connect to various accounts using a series of passwords. This could result in all the accounts being locked out. You need to disable account lockout as this software runs.

Chapter 8—Database Activity Monitoring

This chapter discusses how to monitor your database(s) and tells us that some estimates show up to 70% of all database attacks are from insiders. This may all be true, but the discussion of monitoring database(s) assumes you have the resources to set up and maintain the monitoring software and to review and take action on all the logs that will be generated.

It is tough to know all the sources your users will connect from when those users can be on the company network, at home, on the road, connecting through their ISP, and so on. Similarly, we are told to define a SQL signature for what we expect the normal pattern of SQL access to be. I have no clue how to start this let alone maintain it. It sounds like a good idea. The description of the Sweeney attack is very good. I had no idea. This is yet another way that your data may not be as secure as you think.

The section discussing adhering to government and industry regulations is very good. It makes me wonder how anyone really does adhere to all of them for any length of time. The authors end the chapter by stating that the Sarbanes-Oxley (SOX) regulations are vague as far as technical requirements are concerned. Been there, done that, don't even have a T-shirt.

"The market needs more books like this—books that don't take months to read and don't go over basic information that the reader should already know . . . I hope this book is part of a larger trend in publishing towards specific and useful information and away from each book being a doorstop that almost never gets read."

Chapter 9—Implementation Guide

This final chapter states that the solutions presented so far are a lot to digest and taken together are a daunting task. I agree. While this chapter gives us a reasonable plan for implementing all the advice given in the preceding 8 chapters, it isn't clear who can afford the resources to make all this happen.

The point is repeated that CPU patches can reverse changes you made to improve security. We are told to remove all access to datafiles and redo logs from users other than Oracle. I don't know the impact of changing the permissions from those that were set up when the Oracle software was created. While it sounds like a good idea, without knowing the affect of these changes, how can I weigh the increased security against the possible downtime caused by these changes?

The point is made that making passwords so hard to guess and therefore virtually impossible to remember means users will write down the passwords. Have we really achieved better security or simply passed an audit?

Conclusion

The market needs more books like this—books that don't take months to read and don't go over basic information that the reader should already know. The authors have a point and they get to it quickly. This book targets a specific segment of the Oracle user base, namely those that already know about database security issues and need specific help working those issues. I hope this book is part of a larger trend in publishing towards specific and useful information and away from each book being a doorstop that almost never gets read. ▲

Brian Hitchcock has worked at Sun Microsystems in Newark, California, for the past 11 years. He is a member of a DBA team that supports 2400+ databases for many different applications at Sun. He frequently handles issues involving tuning, character sets, and Oracle applications. Other interests include Formula One racing, finishing his second Tiffany Wisteria lamp, Springbok puzzles, Märklin model trains, Corel Painter 8, and watching TV (TiVo rules!). Previous book reviews by Brian and his contact information are available at **www.brianhitchcock.net**.