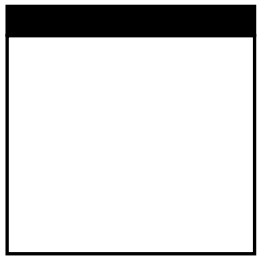
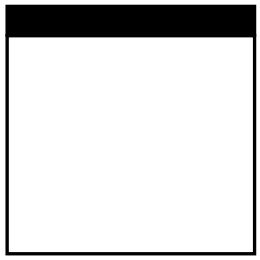


The Oracle® Hacker's Handbook: Hacking and Defending Oracle

David Litchfield

Wiley Publishing, Inc.

The Oracle® Hacker's Handbook: Hacking and Defending Oracle



The Oracle® Hacker's Handbook: Hacking and Defending Oracle

David Litchfield

Wiley Publishing, Inc.

The Oracle® Hacker's Handbook: Hacking and Defending Oracle

Published by
Wiley Publishing, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN-13: 978-0-470-08022-1
ISBN-10: 0-470-08022-1

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

1MA/QT/QR/QX/IN

Library of Congress Cataloging-in-Publication Data:

Litchfield, David, 1975-
The Oracle hacker's handbook : hacking and defending Oracle / David Litchfield.
p. cm.
Includes index.
ISBN-13: 978-0-470-08022-1 (paper/website)
ISBN-10: 0-470-08022-1 (paper/website)
1. Oracle (Computer file) 2. Database security. I. Title.
QA76.9.D314.L58 2007
005.8--dc22

2006036733

With love, for Sophie and our two "girls," Susie and Katie.
Adopt a greyhound!

About the Author

David Litchfield is the founder and Chief Research Scientist of NGSSoftware Ltd., a U.K.-based security solutions provider. David is known as the world's premier expert on Oracle database security, having gained that reputation when he uncovered a security hole in Oracle 9 Database Servers, which disproved Oracle's multimillion dollar "unbreakable" marketing campaign. He has lectured at both the National Security Agency in the U.S. and G.C.H.Q. in the U.K. on emerging threats and information assurance. He is a regular speaker at Blackhat Security Briefings and Microsoft Bluehat and Microsoft TechEd. Previously, he was Director of Security Architecture at @stake, since acquired by Symantec. David has designed NGSSQuirreL, a powerful tool for advanced database vulnerability and risk assessment.

Credits

Executive Editor
Carol Long

Development Editor
Kenyon Brown

Production Editor
William A. Barton

Copy Editor
Luann Rouff

Editorial Manager
Mary Beth Wakefield

Production Manager
Tim Tate

Vice President & Executive Group Publisher
Richard Swadley

Vice President and Publisher
Joseph B. Wikert

Project Coordinator
Jennifer Theriot

Graphics and Production Specialists
Carrie A. Foster
Brooke Graczyk
Denny Hager
Stephanie D. Jumper
Alicia B. South

Quality Control Technician
Jessica Kramer

Proofreading and Indexing
Linda Quigley
Techbooks

Anniversary Logo Design
Richard Pacifico

Contents

Acknowledgments

Firstly, I'd like to extend my gratitude to my wife, Sophie, for her understanding and putting up with my odd sleeping times. I'd also like to thank the team at Wiley, with special thanks going to both Carol Long and Kenyon Brown for putting up with the long periods of "blackouts" followed by an avalanche of material.

Introduction

It's terribly important that Oracle get security right, and so far their record has been poor. The Oracle RDBMS has had more critical security vulnerabilities than any other database server product. By critical, I mean those flaws that can be exploited by a remote attacker with no user ID and password and which gives them full control over the database server. To put these critical security vulnerabilities in context, IBM's DB2 has had 1; Informix has had 2; and Microsoft's SQL Server has had 2. Oracle has had 9. That's more than the other database servers put together. In terms of flaws that require a user ID and password but yield full control when exploited, again Oracle outstrips the rest by far. These facts stand in stark contrast to Oracle's marketing campaigns claiming that their product is "unbreakable." When Oracle executives say, "We have the security problem solved. That's what we're good at . . . ," it makes you wonder what they're talking about. So far the problem is not solved, and complacency should have no home in an organization that develops software that is installed in most governments' networks. This is why it is absolutely critical for Oracle to get it right—national security is at stake.

Oracle's idea of what security means is formed largely on the U.S. Department of Defense's assurance standards. This is why Oracle can state that they "get security." This may have worked 15 years ago, but the security landscape has entirely changed since then. Let me explain further. The Oracle RDBMS was evaluated under the Common Criteria to EAL4—assurance level 4—which is no mean feat. However, the first few versions of Oracle that gained EAL4 had a buffer overflow vulnerability in the authentication mechanism. By passing a long username to the server, a stack-based buffer is overflowed, overwriting program control information,

and allowing an attacker to take complete control. How on earth did this get through and how was it missed? The answer is that there is a vast divide between what "standards" security means and what real security means. There is, of course, an important place for standards, but they are not the be all and end all, and Oracle would do well to learn this lesson. Standards imply rules but hackers don't play by the rules.

Perhaps Oracle is beginning to understand, though. By all accounts they have shaken up and improved their coding standards, and have invested in numerous tools to help them develop more secure code; and there *is* evidence to suggest that things are getting better on the security front. Oracle 10g Release 2 is a dramatic improvement over 10g Release 1. Security holes are still being discovered in 10g Release 2, but nowhere near the numbers that have been found with 10g Release 1. Oracle has also improved their security patch release mechanism. Every quarter, Oracle releases a Critical Patch Update (CPU), and up until July 2006 every CPU was reissued multiple times because of failings and missing fixes and other problems. The July 2006 CPU was different; it was released once—hopefully the start of a trend.

Considering that things are improving, where exactly is Oracle on this journey to "security" utopia—by which I mean a secure product that actually matches the marketing speak? In answering this question, for any vendor, a key pointer is to look at how they respond to security researchers. In the summer of 2006 at the Blackhat Security Briefing, I was on a panel that discussed the issues surrounding the disclosure of security flaws. The panel moderator, Paul Proctor from Gartner, insightfully suggested that "Microsoft is in the acceptance phase. Cisco is slowly moving out of the anger stage and into the acceptance stage. Oracle, on the other hand, is just coming out of the denial stage and into the anger stage."

This is an accurate assessment in my estimation. Like Microsoft a few years ago, when Scott Culp published his "Information Anarchy" paper, Oracle too had their say about security researchers when Mary-Ann Davidson, the Chief Security Officer of Oracle wrote her article "When Security Researchers Become the Problem." The difference between Mary-Ann's article and Scott's paper is that Scott's needed to be said, as it was published at a time when there *was* information anarchy and not much responsible disclosure going on; it was an attempt at convincing security researchers to work *with* the vendor. This is why Mary-Ann's article a few years later failed to hit home: The security researchers she disparaged were already working with Oracle to try to help improve their product. Oracle failed to see that they and security researchers were working toward the same goal—a more secure database server. Part of the article discusses security researchers making explicit and implicit threats, such as "Fix it in the next three weeks because I am giving a paper at Black Hat." However, Oracle

should understand that a security researcher is under no obligation to inform them that they are going to present a paper; and if they do tell them, Oracle should appreciate the heads up. Such information is a courtesy. Calling this an "implicit threat" is disingenuous and risks alienating the very people best placed to help them secure their product. It would be in the best interests of all for Oracle to get over their anger stage and embrace the acceptance phase.

Enough commentary on Oracle, however, at least for the time being. Let's look at why we need a book that details vulnerabilities in their RDBMS and examines how those flaws are exploited. In short, precisely because it is such a popular database server, it is a prime target for hackers, organized crime, and those involved in espionage, be it industrial or foreign. Therefore, there should be a reliable resource for database and security administrators that shows them how their systems are attacked and where they are vulnerable. This puts them in a position of strength when designing defense strategies and mitigations.

This book is that resource. Yes—such a book is, by nature, paradoxical: intended to aid defense, it arms not only the defender with the information but also the attacker. It is my experience, however, that most attackers already know much of this information already, whereas the defenders don't but should. Yet even today, given all the evidence to the contrary, you hear Oracle "experts" claiming that Oracle is secure, citing as proof that Oracle is "always installed behind a firewall" and that it "runs on Unix." Frankly, these "reasons" have nothing to do with whether Oracle is secure or not. It's as easy to break into an Oracle server running on Linux or Solaris as it is on Windows. A firewall becomes irrelevant as soon as you poke a hole through it to allow your business logic and web applications to connect to the database server—SQL injection is a major problem.

Furthermore, it is a myth that Oracle is always installed behind a firewall. According to the "Database Exposure Survey" I performed in December 2005 and published in June 2006, an estimated 140,000 Oracle database servers are out there accessible on the Internet, compared to 210,000 Microsoft SQL Servers. Given that many of these SQL Servers will be MSDE installs, one wonders what effect Oracle Express will have on the number. Oracle Express was released after the survey. Getting back to the core of the problem, however, there is not nearly enough understanding by those in the Oracle world that their servers are exposed to risk. When you consider that Oracle has committed to releasing a Critical Patch Update every three months until at least July 2007 (at the time of writing), this means that in the interim Oracle database servers are in a critically insecure state. Food for thought, indeed. This is the "why" then. If we are to take responsibility for the security of our own systems, knowing that they are

critically insecure, we need to know *how* they're insecure—only then can we take steps to prevent our systems from being compromised.

I hope that as well as finding this book useful and informative, you have fun reading it. I'm always willing to answer questions so please feel free to ask.

Cheers,

David Litchfield
david@databasesecurity.com

Code Samples from the Book

To download relevant code samples from the book, please visit the book's web site at www.wiley.com/go/ohh.

Oracle and Security

In June of 1997, Larry Ellison and Robert Miner founded a company called Software Development Labs. Both had worked together at Ampex; Robert had been Larry's supervisor. Together they had a vision, inspired by the work of Edgar Codd. Codd worked as a researcher for IBM and developed ideas for relational database systems. In 1970 he published a paper entitled "Relational Model of Data for Large Shared Data Banks." While IBM was slow to see the potential of Codd's ideas, Larry and Robert were not. They changed their company's name to Relational Software, Inc., in 1979, and not long after that it again underwent a name change—becoming Oracle. "Oracle" had been the code name for a CIA project that both Larry and Robert had worked on while at Ampex. Indeed, by all accounts, in the early years, the biggest consumers of Oracle's software was the CIA and the NSA. Given this, one would assume that security would have been at the top of Oracle's agenda.

In 1999 Oracle started to gain the attention of the security research community. The first public record of a security bug in Oracle, according to SecurityFocus.com, was on April 29 of that year: Dan Sugalski posted that the oratclsh program was setuid root and executable by the *nix group, "others". This meant that anyone could run TCL scripts as the root user. Not long after this a number of flaws were revealed relating to the Oracle Web Listener, posted by the author and Georgi Guninski, as well as additional problems relating to default permissions. Oracle started releasing their own advisories in the year 2000, but these were released on an ad-hoc basis and often months after a flaw was announced publicly.

The graph shown in Figure I-1 shows the number of bugs reported in the "early" years. As you can see, the number of bugs grew exponentially, and indeed this is the only reason why 2003 and beyond are not shown on this graph—the numbers went through the roof, so to speak. Most of these bugs relate to buffer overflows in the RDBMS or the Application Server. One of the key weaknesses in Oracle, PL/SQL injection, didn't come to light until 2003, when on the fifth of November Oracle released Alert 61—an advisory that dealt with a number of injection flaws discovered by the author. This began a spate in the discovery of PL/SQL injection flaws, and even today most of the issues being fixed by Oracle are injection problems. As you'll see later in the book, flaws in PL/SQL are the Achilles' heel for the RDBMS, and when exploited allow an attacker to gain full control of the database server.

The "Unbreakable" Marketing Campaign

With the release of Oracle 9i, Oracle began a new marketing campaign in December 2001. They announced that their product was "unbreakable" and that no one could "break it" or "break in." To many in the security industry this was one of those Kennedy moments: Everyone can remember what they were doing when they first heard what Oracle announced. I certainly remember what I, and no doubt many others, did after—which was to download this "unbreakable" software and see just how tough it actually was. Within days of the announcement, my brother Mark and I had sent Oracle a bunch of reports detailing a great number of ways in which the server could be both broken and broken in to. After Oracle fixed these flaws I presented a paper on the issues at the Blackhat Security Briefings in the February of 2002. Oracle was thoroughly broken.

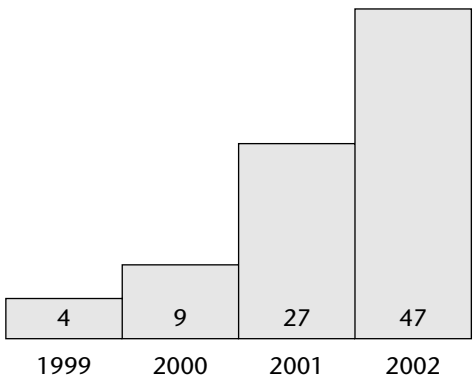


Figure I-1: The number of bugs grew steadily between 1999 and 2002.

Today, Oracle will tell you that the campaign spoke of their commitment to security, and was not to be taken as a statement about their product's security. Hmmm.

Independent Security Assessments

So what are we to make of Oracle's commitment to security? What do they mean by that? Well, Oracle has invested a great deal of money in having their products independently assessed. These are foisted upon the consumer public as proof positive that Oracle is secure. In "real" security world terms, however, being evaluated to EAL4 (Assurance Level 4) under the Common Criteria means nothing. How could it? Both Oracle and IBM's Informix (EAL2) were accredited under the Common Criteria yet both had a buffer overflow vulnerability due to a long username. All the right features are in the product to be able to get accredited but they're all holey. A castle is no castle if its door is made of cheese.

The first version of Oracle to gain EAL4 was 7.2 in September 1998. Next came Oracle 8.0.5 in October 2000, and then 8.1.7.0 in July 2001. In September 2003, Oracle 9iR2 was certified, followed by 10g Release 1 in September 2005. Since attaining certification, all of these versions have been weighed and found wanting—badly.

If you haven't guessed by now, I'm not a big fan of independent security evaluations but I suppose they do have their place and they give developers something to aim toward. This only holds true, though, if the development of the software is not a whitewash or mere window dressing.

The Future

Oracle 10g Release 2 is a good product. It's a vast improvement over 10g Release 1 when it comes to security. Oracle should be commended for that. However, it's not "mission accomplished" yet. There are still bugs in 10g Release 2—many of which are discussed in this book—and there are more to be found. Still, whereas it took only 5 to 10 minutes of searching to find a new bug on 10g Release 1, it takes a good day's effort or more to find one on 10g Release 2. The improvements are largely due to a heavy investment in source code auditing tools. While these tools do a great job of catching most flaws, they have a problem with boundaries—for example, when a PL/SQL procedure calls out to a C function or a Java function. The tools seem unable to pick up flaws that occur in these crossover points. Oracle needs to make