

# Oracle Web Hacking Part I

By Chris Gates, CISSP, GCIH, C|EH, CPTS

Oracle applications are not what you'd call simple. I think any DBA or Oracle Application Server Administrator will be the first to attest to that fact. Oracle, with its great products, comes with some un-pleasantries. These are:

1. Oracle applications are complicated (hopefully we all agree on this).
2. They come with loads of default content and no clear way to remove that content. There is no IISLockdown equivalent for Oracle applications. Content you don't want must be removed manually. Some of this content can be used to run database queries, read documents, gather information via information leakage on the pages or perform XSS attacks.
3. Users have to pay for patches and extended advisory information (even then, no Proof of Concept code is released by Oracle).
4. And lastly, you have a fairly complicated patch/upgrade process which leads to an "it's working, don't touch it" mentality by a fair amount of admins.

This provides a target rich environment for pentesters and bad guys. Let's take a look.

del.icio.us

Discuss in Forums {mos\_smf\_discuss:Gates}

One of the issues that used to frequently come up on my penetration tests was running into random Oracle content pages. Here are some examples of default oracle pages (that don't actually have anything useful) that appear often, because they are the index pages.

Oracle HTTP Server Index Page

Oracle Application Server Portal Index Page

Obvious questions immediately come up when you discover pages like this. These include what default content/components exist, how do I find it, and more importantly how do I leverage this content to further penetrate into the network.

On public facing servers we may catch a break and use Google and Bing hacking [www.red-database-security.com/wp/google\_oracle\_hacking\_us.pdf ] to find some of those useful content/components. This is of no help to us internally though as internal servers won't be indexed.

A couple of default content scanners exist out in the world. Oapscan comes to mind, Inguma has one built-in, nikto has some oracle checks, etc. The problem I kept running into was that I would give a 200 for some random URL, but I'd have no idea what it did, what it was for, or how to exploit it. No fun.

In the end I used a combination of almost all the checks contained in various scanners that I could find and made a Metasploit auxiliary module out of them. I also added a "vuln" field to that output which would give me (and hopefully follow-on contributors) the ability to give some insight into why that URL was vulnerable or a reference on how to exploit it.

oracle\_oas\_scan.rb

The idea is that, as we find documents in the oracle web application, we get a clue as to why that can be useful. A couple of the more useful pieces of default content found by our scan are below.

Example1

/demo/sql/jdbc/JDBCQuery.jsp

Based on the results of our oracle\_oas\_scan.rb script, we see that /demo/sql/jdbc/JDBCQuery.jsp is vulnerable to SQL Injection and allows us query information from the oracle instance.

## SQL Injection in JDBQQuery.jsp and the Result Showing the Oracle SID Name

### Example 2

`/xsql/adhocsql/sqltoxml.html`

The oracle\_oas\_scanner searches for this useful piece of default content. This page allows us to run arbitrary SQL queries.

### Oracle Demo Page

### Example 3

#### iSQL\*Plus

iSQL\*Plus is a web interface to the TNS Listener. You log in with a username/password/database SID, and you are given a text box in which you can run sql commands, just like you were logged into the sqlplus interface.

#### iSQL\*Plus Login Page for Oracle DB 9 R2

#### Logged into iSQL\*Plus

Hopefully the value of the interface is shining through by now. Now the issue of any website that requires authentication is how to guess the credentials or bypass them.

Bring in isqlplus\_sidbrute.rb and isqlplus\_login.rb.

If you remember from the hacking oracle via the TNS listener material [ <http://www.slideshare.net/chrisgates/attacking-oracle-with-the-metasploit-framework> & <http://www.youtube.com/watch?v=Hj7u8Ja-mPM> ], we need 4 things to connect and log into an oracle instance: IP, port, usernames/password, and database SID. In the case of iSQL\*Plus IP & port are taken care of. Since it is a web application, we've obviously found the page. All that's left is username/password and SID.

Luckily for us the application responds differently for incorrect usernames/password with the right SID and incorrect usernames/passwords with the wrong SID. This allows us to throw password guessing for the SID field in the application.

Using error messages returned by Oracle determines valid SID:

Wrong SID:

ORA-12154: TNS: could not resolve service name

Right SID (wrong password):

ORA-01017: invalid username/password; logon denied

As an added bonus, iSQL\*Plus authenticates by default to the first SID in the tnsnames.ora file. This means we can \*usually\* pass no SID, and it will try to auth to the top SID in the tnsnames.ora file.

Isqlplus\_sidbrute in Action

Once we have a valid SID, or know that the application allows us to pass a blank SID in the POST request, we can repeat the process to guess valid username/password combinations.

Isqlplus\_login in Action

Once valid credentials are obtained you can login to the interface and run SQL commands to extract data or attempt privilege escalation attacks against the database and/or conduct further post exploitation activities against the database server.

Or for added lolz, you can read random files off the server:

Reading Arbitrary Files from the Host

Output of the File

That's it for now. Enjoy using vendor created content to dig further into the network!

To get a copy or contribute to the code mentioned in the article, grab it on github through carnal0wnage <https://github.com/carnal0wnage/carnal0wnage-code> and in the Web eXploitation Framework <https://github.com/WebExploitationFramework>.