SQL Injection

ISM 6218

Due on October 8th

The Avengers Team

"We will avenge every problem on our way"

Aitemir Yeskenov (Team Lead)

Nagarjuna Kanneganti

Sai Suraj Argula

Vinay Kumar Reddy Baradi

Table of Contents

Business Process Supported

For this experimental analysis we have created Inventory table for a warehouse containing various products and the stock left. We have inserted some values in this table.

Requirements Described

Exam 1: Subjects Evaluated

1. Create a SPROC that allows a SQL Injection attack.

2. Recreate the SPROC to prevent the attack using any of the methods discussed.

3. Must support a user story.

Table Diagram:

In this case we used a database with only one table – we created this table from scratch.

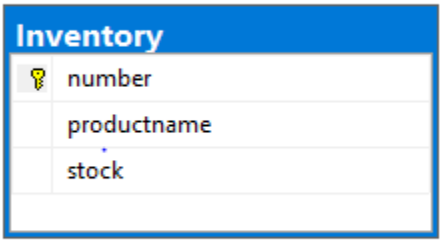


Table Creation:

```
CREATE TABLE Inventory (
number int not null,
ProductName Varchar (50) not null,
stock int,
Primary key (number)
);
```

Data entry:

```
Insert into Inventory values (1,'iPhone',10), (2,'biscuits',10),(3,'breads',10),(4,'cakes',10);
Select * from inventory;
```

Results | Messages

	number	productname	stock
1	1	choclates	10
2	2	biscuits	10
3	3	breads	10
4	4	cakes	10

User Story:

Warehouse admin wants to check the stock left for different products through a portal where he can type just the product name to get the product and stock left.

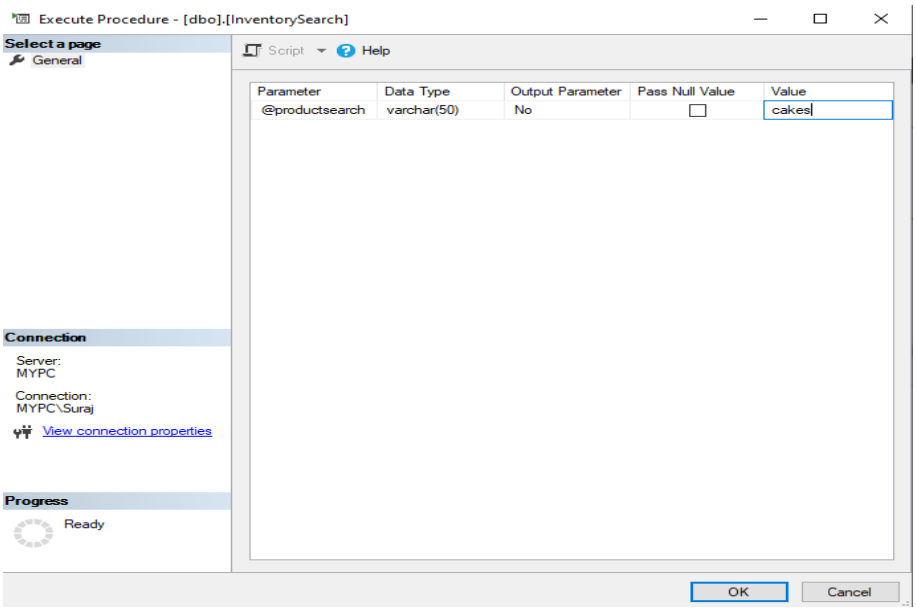
1) Create a SPROC that allows a SQL Injection attack.

SPROC

```
select * from Inventory;
Alter procedure InventorySearch
@productsearch varchar(50)
as
Begin
declare @sql nvarchar(100)
set @sql = 'SELECT * FROM Inventory WHERE productname='" + @productsearch +"';
exec(@sql)
end
```

Analysis - 1 Functional Requirement:

For example, if he wants to check the cake stock, he will enter **Cakes,** which triggers the stored procedure and returns the stock of cakes.




```
DECLARE @return_value int

EXEC    @return_value = [dbo].[InventorySearch]
        @productsearch = N'cakes'

SELECT  'Return Value' = @return_value

GO
```

110 %

Results | Messages

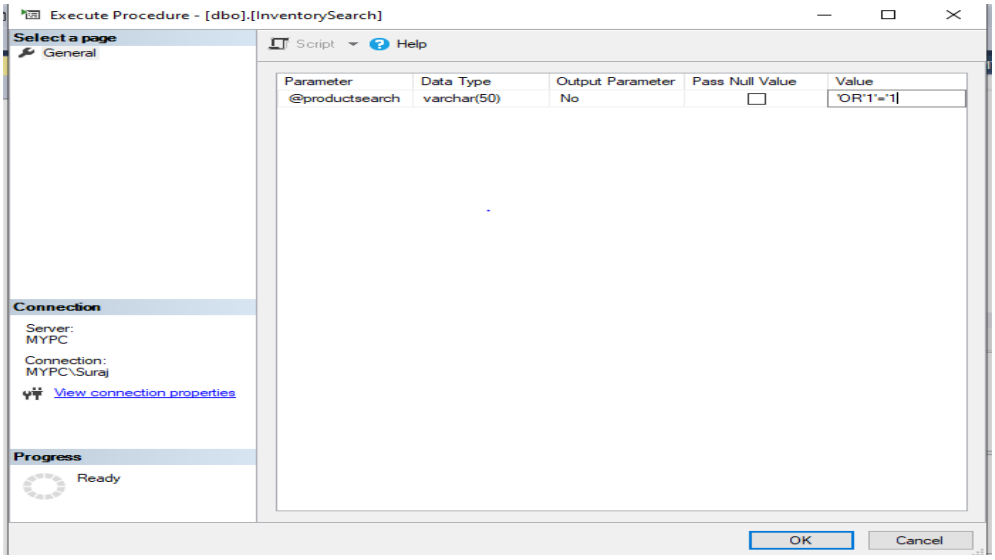
	number	productname	stock
1	4	cakes	10

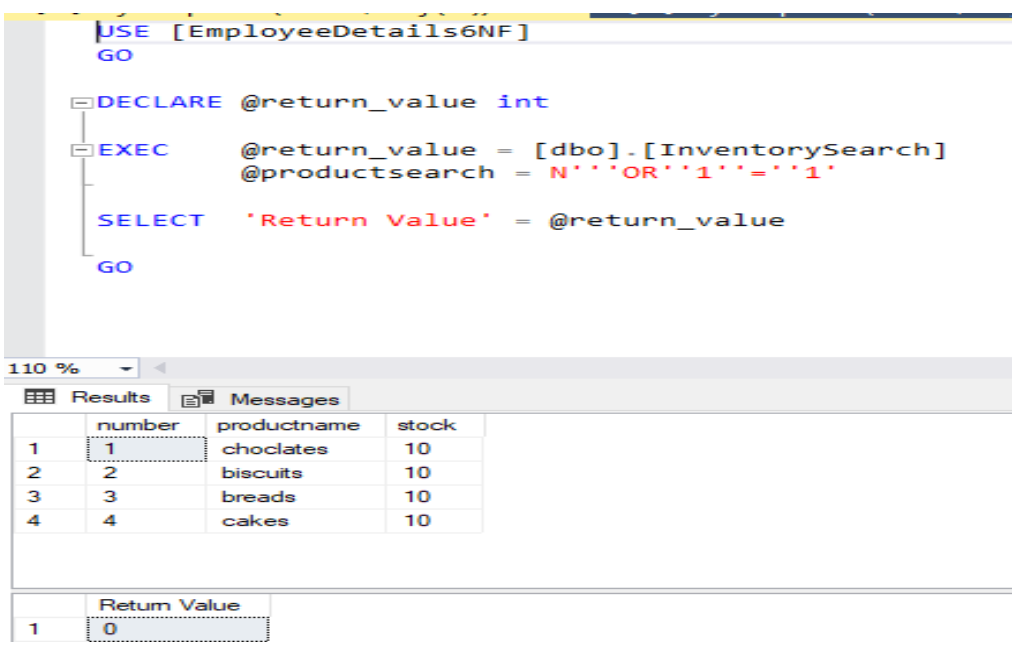
	Return Value
1	0

Analysis 2:

However, there are many security issues in the above stored procedure that make the database vulnerable from Hackers which is called SQL injection.

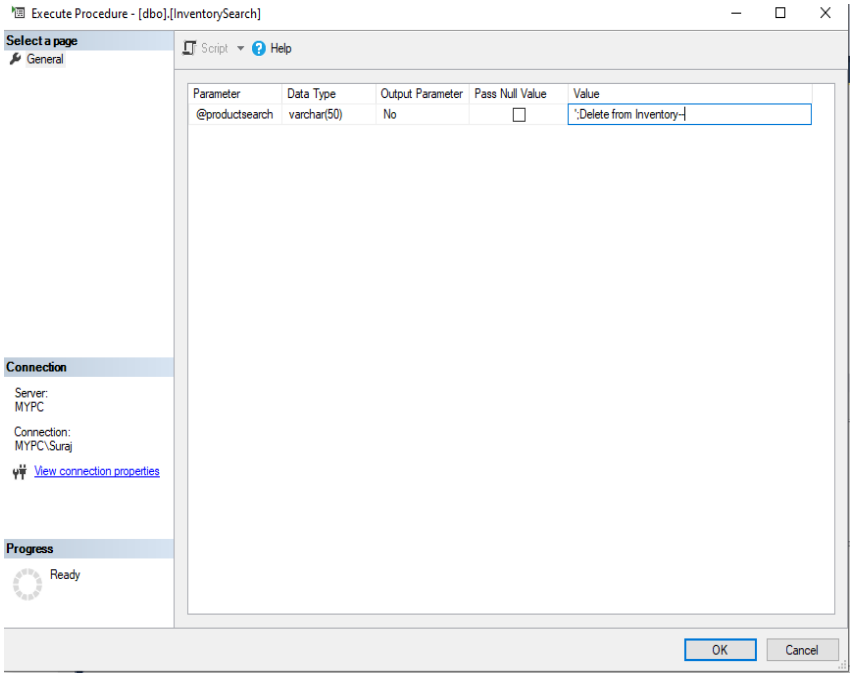
For instance, if someone enters 'OR'1'='1 into the field. Below output of all the products with the stocks is displayed. This is because the field has no restrictions to prevent a user from entering the "wrong" input. Since 1 =1 is always true it returns all the information. This is one of SQL injection technique.





Analysis 3:

This is another type of SQL injection. Not only the data is vulnerable to view, but also the intruder can update and delete. From the below screenshot, we can see that the data deleted by entering the ';Delete from Inventory— into the field which because the colon ; in the above statement ends the first statement to search and starts a new statement, which would delete the table.



```
USE [EmployeeDetails6NF]
GO

DECLARE @return_value int

EXEC    @return_value = [dbo].[InventorySearch]
        @productsearch = N''';Delete from Inventory--'

SELECT  'Return Value' = @return_value

GO
```

110 %

Results | Messages

number	productname	stock

	Return Value
1	0

```
SELECT * FROM Inventory
```

10 %

Results | Messages

number	productname	stock

2) Recreate the SPROC to prevent the attack using any of the methods discussed.

In the Above experiment we have found there are some issues with the SPROC, we Recreated the SPROC to prevent SQL injection by introducing parameterized inputs into the SPROC instead of concatenating the input with the query.

Create procedure InventorySearch2

@productsearch varchar(50)

as

Begin

SELECT * FROM Inventory WHERE productname=@productsearch;

End

Analysis 1:

For checking main functional requirement. Our main requirement is met.

```
Create procedure InventorySearch2
@productsearch varchar(50)
as
Begin
SELECT * FROM Inventory WHERE productname=@productsearch;
end
EXEC [dbo].[InventorySearch] @productsearch = N'cakes';
```

110 %

Results | Messages

	number	productname	stock
1	4	cakes	10

```
DECLARE @return_value int

EXEC    @return_value = [dbo].[InventorySearch2]
        @productsearch = N'cakes'

SELECT  'Return Value' = @return_value

GO
```

110 %

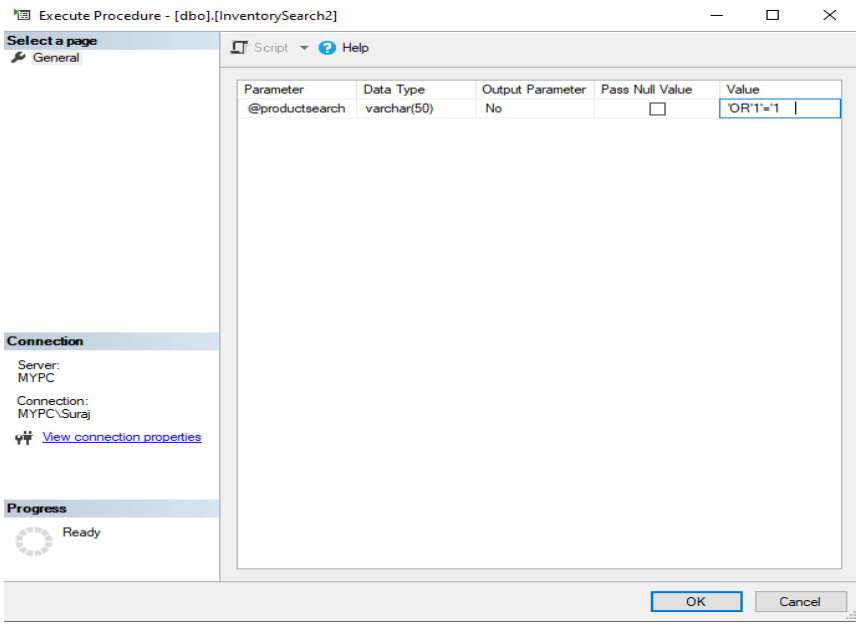
Results | Messages

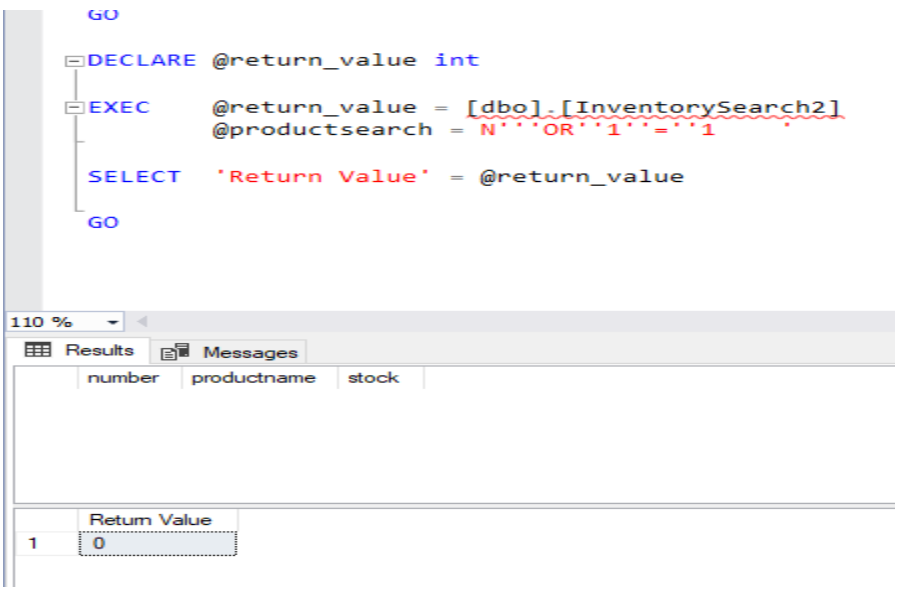
	number	productname	stock
1	4	cakes	10

	Return Value
1	0

Analysis 2:

Checking the cases where found issues with 'OR'1'='1 . We were able to see that, the data cannot be viewed.

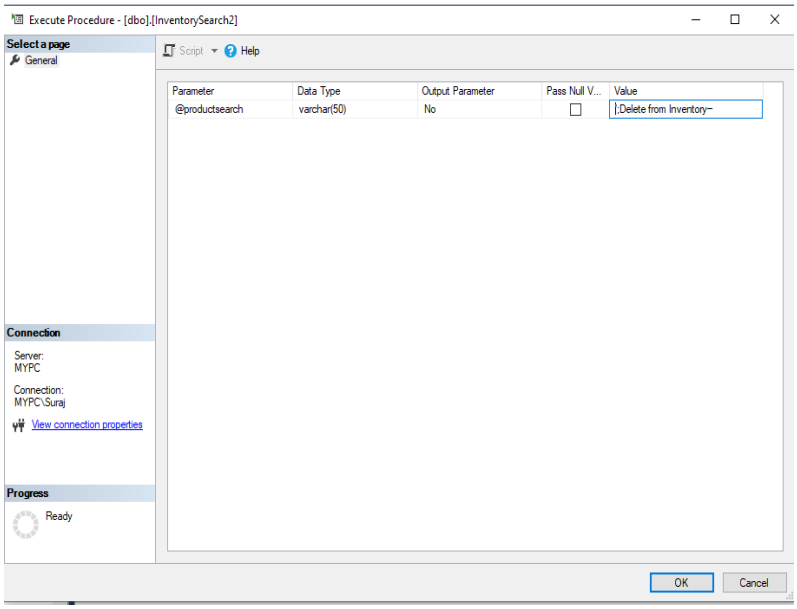


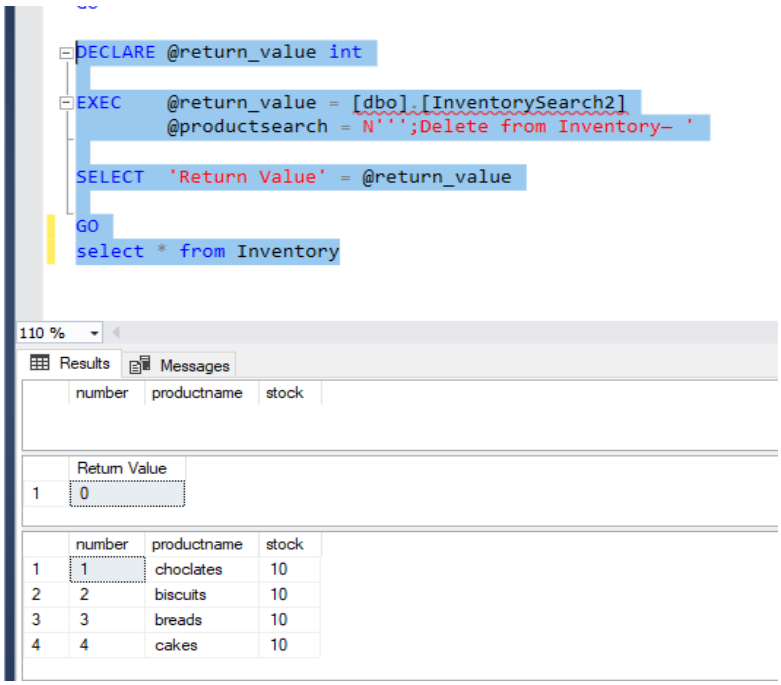


Analysis 3:

Checking the cases where found issues with ';Delete from Inventory—

We were able to see that, the data is not deleted from the table





Conclusion:

The Database is vulnerable to security threats by SQL injection, if the SPROCS are poorly designed by just concatenating the user input data into SQL statements string. We avoided SQL Injection threat by Parameterizing the input variables using SPROCS.