

Objective--Badge Manipulation (Part 4)

Solution (at last)

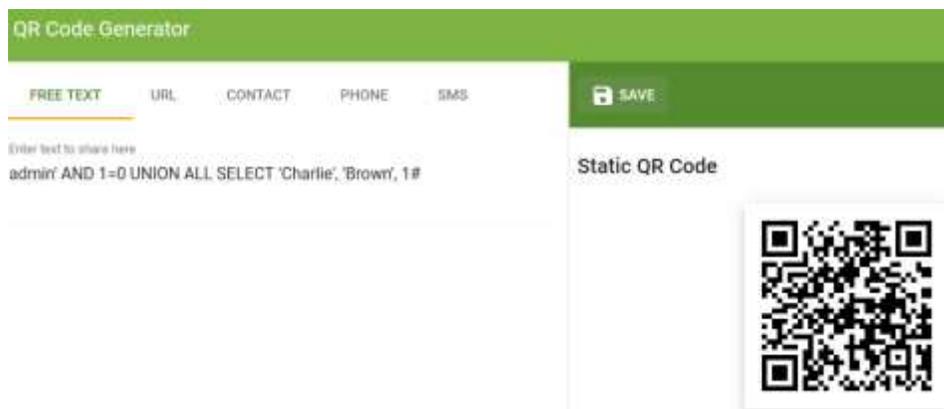
This example from the [Netsparker](#) link is very close to what we want.

Bypassing MD5 Hash Check Example (MSP)

```
Username: 'admin' AND 1=0 UNION ALL SELECT 'admin',  
'81dc9bdb52d04dc20036dbd8313ed055'
```

The first part, `admin '`, doesn't matter except that we need the single quote to close out the `uid=` ' that was already in the query. This query returns two strings, `admin` and `81dc9bdb52d04dc20036dbd8313ed055`. Our query needs to return a string for first name, a string for last name, and the number 1 for enabled. So, we can change their SQLI to this. Don't forget the comment symbol `#` at the end, though.

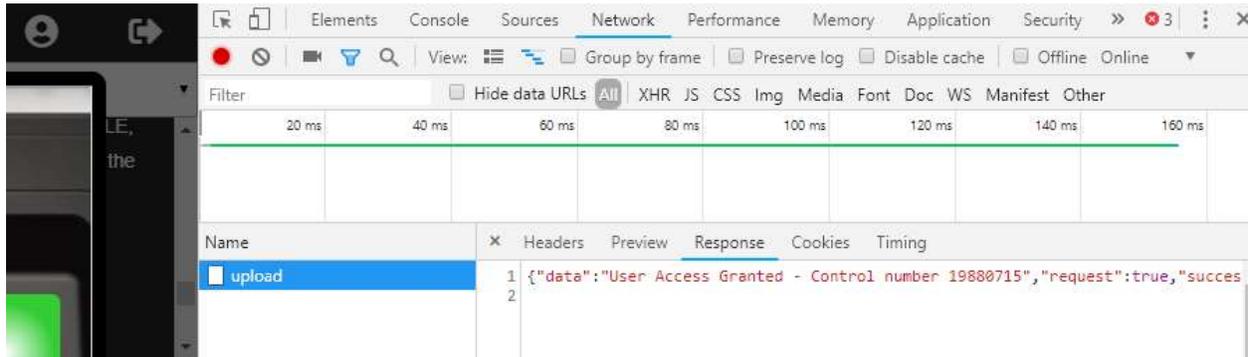
```
admin' AND 1=0 UNION ALL SELECT 'Charlie', 'Brown', 1#
```



It works!



I'm not fast enough to copy the entire message, so let's fall back to Chrome developer tools (Firefox Web Developer works too). We'll need to enter the access number (19880715) into the Objective.



A note: The SQLi we used could have been shortened considerably. This works too.
' UNION SELECT 'Charlie', 'Brown', 1#

Thanks to user Justintime on the CentralSec Slack SANS Channel for help on this one!

Preventing SQL Injection

SQL Injection is #1 on [OWASP's list of the top 10 web vulnerabilities](#), and has been for years. This is shameful because there is a simple way to prevent SQL Injection called [Parameterized Queries](#).

This is an example of an SQL query in Java that is vulnerable to SQL Injection.

```
String custname = request.getParameter("customerName");
String query = "SELECT account_balance FROM user_data WHERE user_name
= custname";
stmt = connection.createStatement();
ResultSet results = stmt.executeQuery(query);
```

You can see that there is nothing that checks the custname variable to detect attempts to embed SQL or other commands. While it is possible to use blacklists (hard to write, easy to defeat) or whitelists (better) to sanitize the input, parameterized queries are easier and safer.

Hand in

- 1) Using the [OWASP Query Parameterization Cheat Sheet](#), fix the Java SQL code so that it uses parameterized queries.

Up Next

The next objective, HR Incident Response, requires that we visit Sparkle Redberry and help her with the Dev Ops Fail terminal challenge. Sparkle is on the left side of the second floor. Off we go!