First Exploit:

The first exploit was pretty simple. After seeing update 3 which implied that there were other updates, I saw the URL and noticed that “num=3”. I changed to 2 and 1 to get the other update pages. Those pages gave me additional information that would be useful for future exploits. The fact that I could obtain the earlier updates through means not envisioned by the people who made the website was a “vulnerability”.

Second Exploit:

The second exploit involved a series of SQL injections. The vulnerability was that on the items page, I was able to inject SQL statements into the search-the-store query. That was because my input was not properly sanitized. Thus I first began by testing if I could get any sort of exploit going - even one not particularly useful to me. “user_id = 2%' OR 1= 1;#” was the first query that worked. Then I realized that I could obtain the other tables on this website through certain exploits. Thus on the internet, I read that SELECT table_name FROM information_schema.tables would give me a list of all the tables. I could theoretically use a UNION to append that to my first query. However, that didn’t work as unions need the same amount of columns selected in both queries. Through “user_id = 2%' ORDER BY 5;#”, and similar queries I was able to determine that the item table had 5 columns so I would have to pad my second query with 4 extra columns. The query that ended up working was “user_id = 2%' OR 1= 1 UNION SELECT table_name, 4,4,4,4 FROM information_schema.tables;# “. After receiving the list of tables I saw that there was “user” table. I then began to query the user table with my UNION and my padding to get the passwords and usernames. The username was determined to be evil_enthusiast. However, the passwords were hashes so I need to utilize another exploit to figure out what the underlying password was.

Third Exploit:

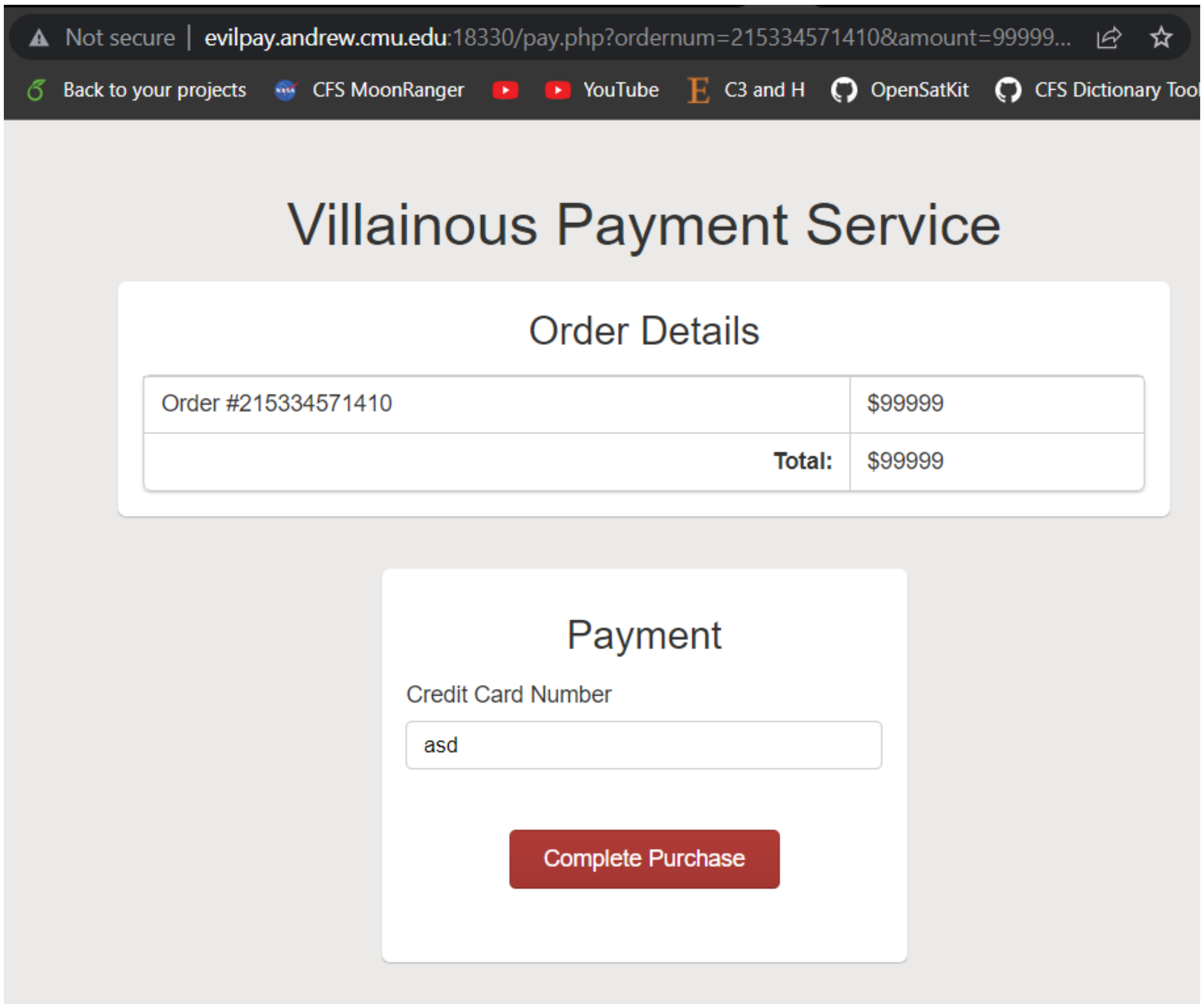
Now that I had hashes representing the passwords, I needed to unhash them and get them to their original form. Even though unhashing them is realistically not feasible there was a vulnerability.The inherent vulnerability was that the password was a widely used password and was actually in the 10 million most commonly used passwords. I found this out after essentially making a python script to go through the 10 million most commonly used words and then quadruple hashing them and checking if any of them matched the hashed passwords found on via the SQL injection. Eventually, I got a match and I knew that the password was bunnies. Now I could actually make purchases as I was no longer a visitor.

Reversing:

After obtaining the password I was now able to make purchases. But I needed to buy my doom laser for $1 instead of $99999. The specific vulnerability was that I could edit the callback website in the url in Evil Pay. Thus I could go back to a callback that did not necessarily correspond to the item that I was buying in my current session. I would later exploit that to buy my laser for $1. My exploit would correspond to a Cross Site Request Forgery or CSRF

(Malicious Browser In The Middle) because I would be using the users browser to execute code that was not intended to be executed.

Successful Exploit:



Villainous Payment Service

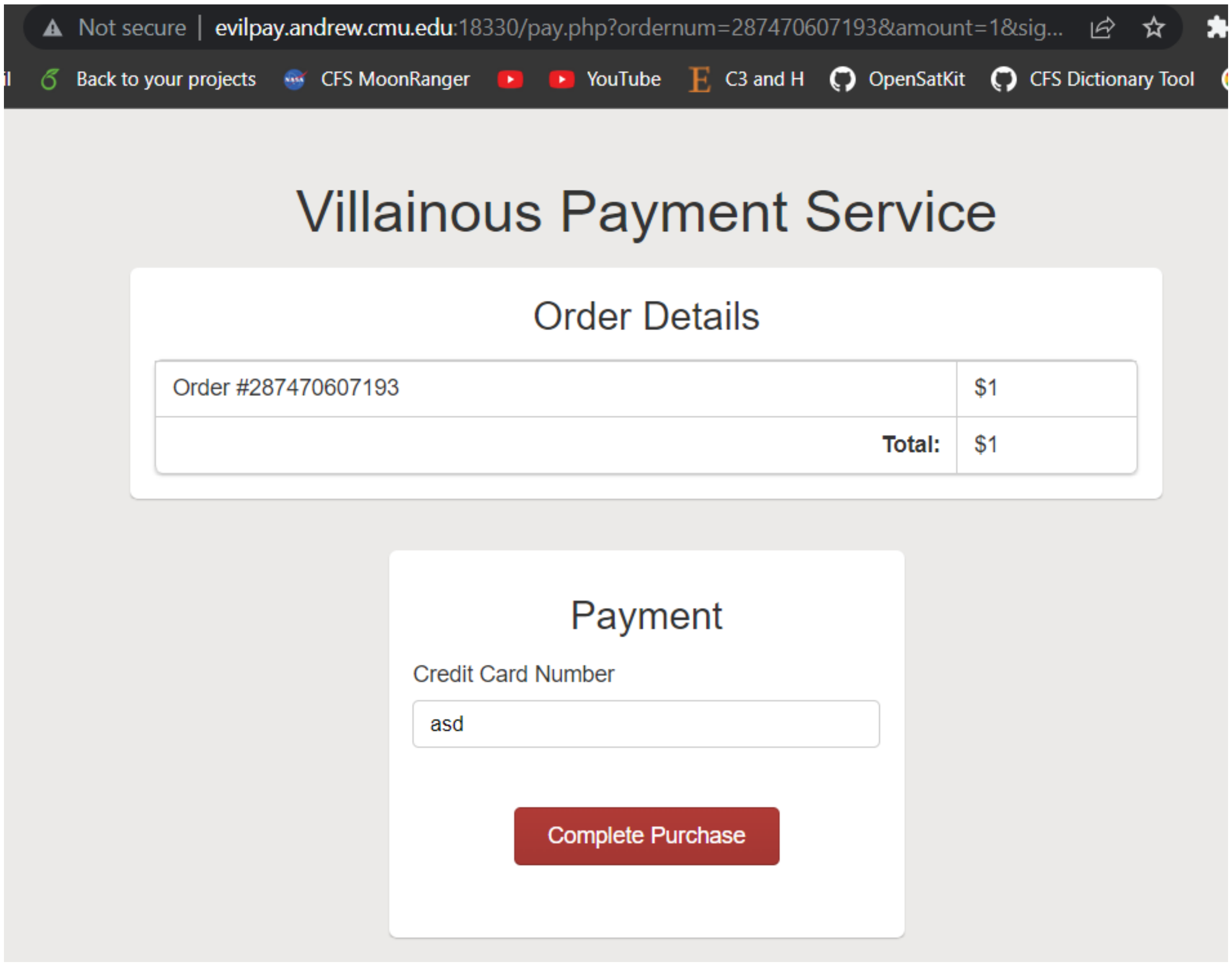
Order Details

Order #215334571410	$99999
Total:	$99999

Payment

Credit Card Number

asd



evilpay.andrew.cmu.edu:18330/pay.php?ordernum=287470607193&amount=1&sig...

Villainous Payment Service

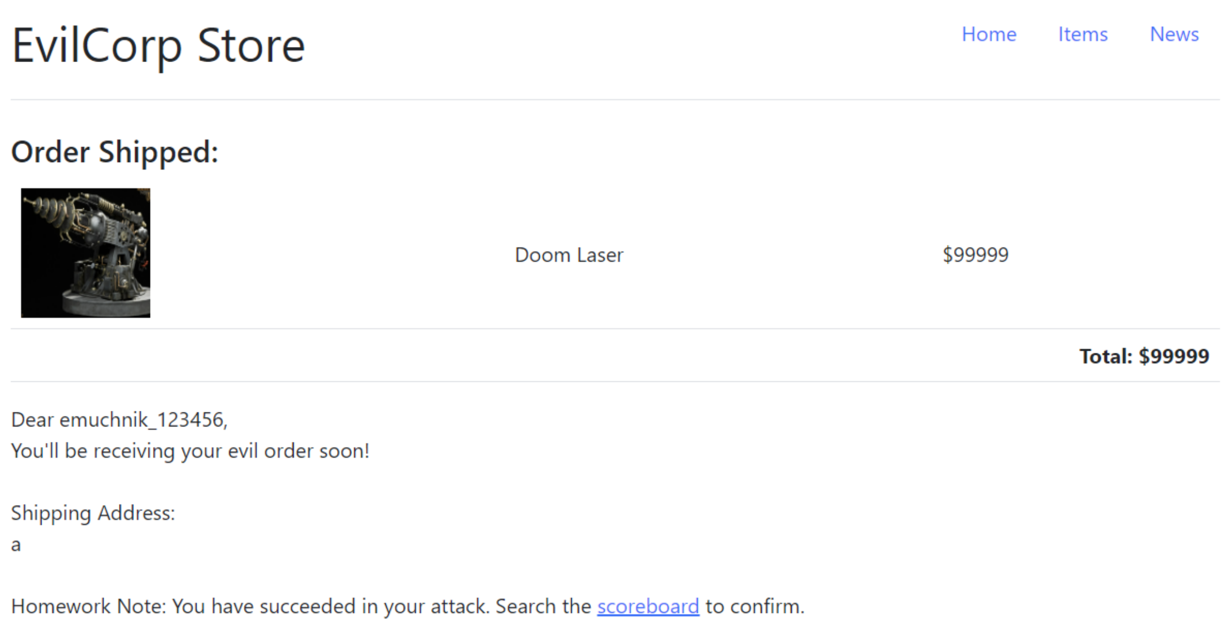
Order Details

Order #287470607193	$1
Total:	$1

Payment

Credit Card Number

asd



EvilCorp Store

Order Shipped:

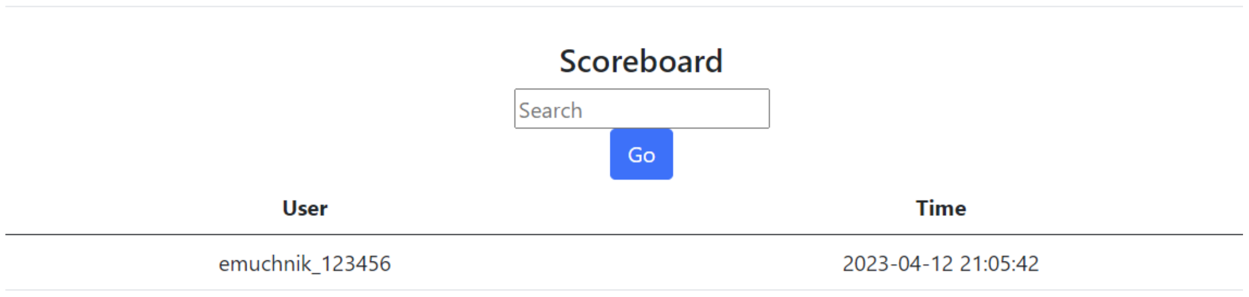
	Doom Laser	$99999

Total: $99999

Dear emuchnik_123456,
You'll be receiving your evil order soon!

Shipping Address:
a

Homework Note: You have succeeded in your attack. Search the scoreboard to confirm.


Scoreboard

User	Time
emuchnik_123456	2023-04-12 21:05:42

Exploit Explanation:

First I just went through a series valid purchases and noted that I was redirected to evilpay and then back to the evilstore during my purchase of both the pen and the laser. I suspected that in order to get redirected back from evilpay in order to give me my item, the url would have to store some data related to the order.

All part of the url except for the url had to do with the signature. Thus, it was impossible to modify without getting a signature-related error. But the callback url at the end was not a part of the signature so I was free to modify it. Importantly the callback url had the ordernum attached. After some more fiddling around I developed the following exploit.

I would get to the evilpay website for both an order of the pen and the laser. Thus I would see the callback url for both. I would copy the ordernum from the callback url of the laser to the pen. Then I went through with the purchase of the pen. Thus I ended up paying only $1 through evil pay. However, when I returned the website thought I bought the laser for 99,999 dollars. Thus I had only paid $1 for the laser.

Repairing:
The main issue is that the website is that it doesn't properly keep track of sessions and doesn't have proper defenses against attacks where callback urls are changed. Overall a general defense discussed is to not roll your own algorithms and to trust protocols made by others that were rigorously tested out. Overall a good protocol would not allow the user to edit callback url and insert another session/purchase by just modifying it.

Scoreboard

Search

Go

User	Time
emuchnik_123456	2023-04-12 21:05:42