**Author**: Shaival Dalal

**Domain**: Network Security

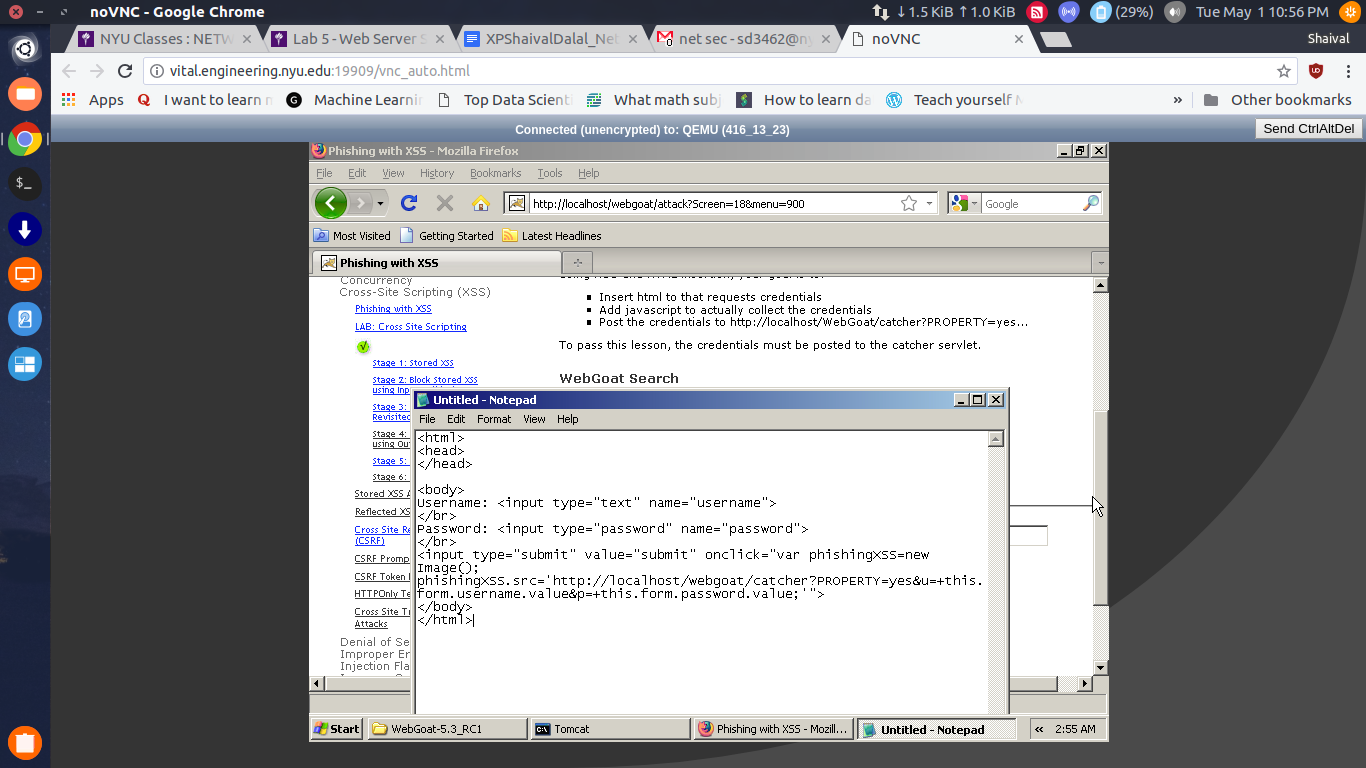
**Aim**: Exploring OWASP WebGoat v5.3

**Description:** The following document contains my explorations in the domain of applied network security. We use WebGoat v.5.3 in order to learn more about how exploits are detected and how web applications are exploited to return private information or perform malicious actions.

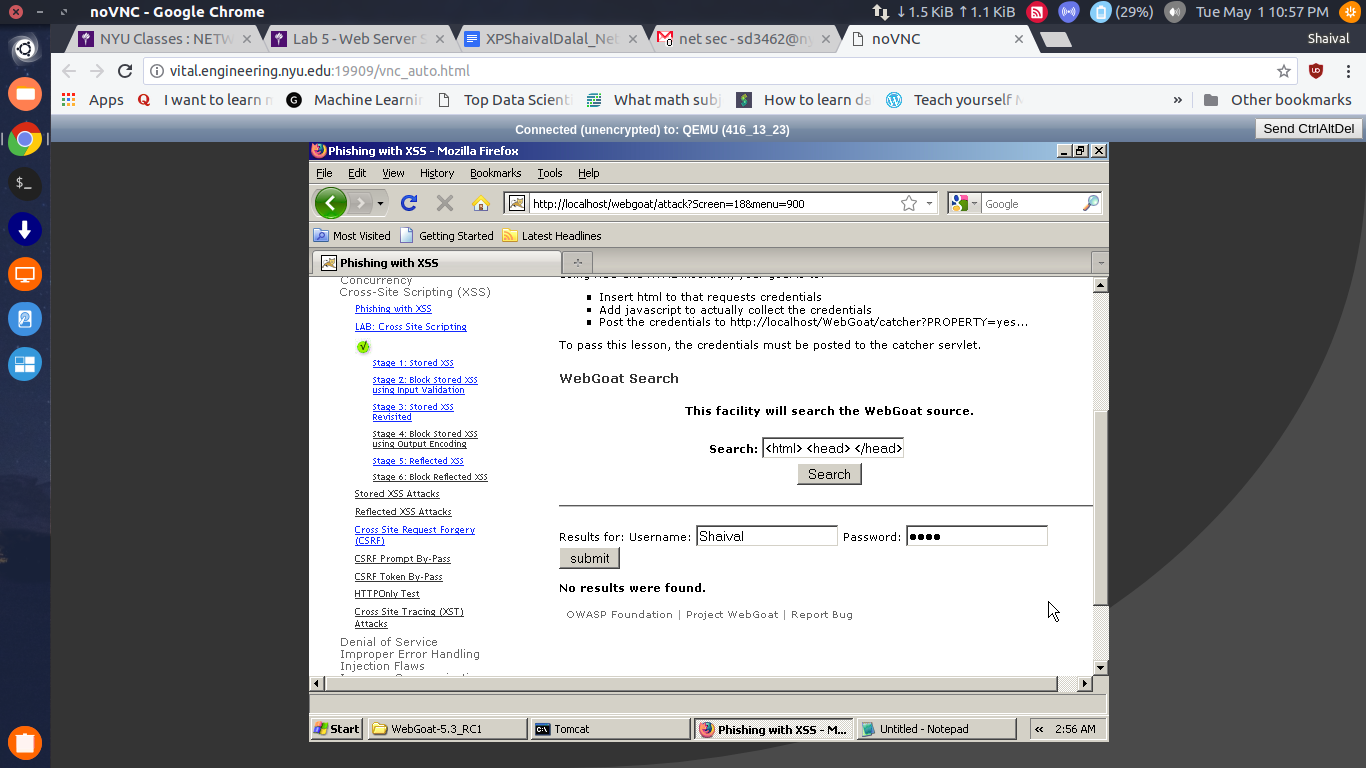
Although the document does not cover all the exploit domains, it contains helpful comments that may guide in similar analysis

1. **XSS**
   1. **Phishing with XSS**

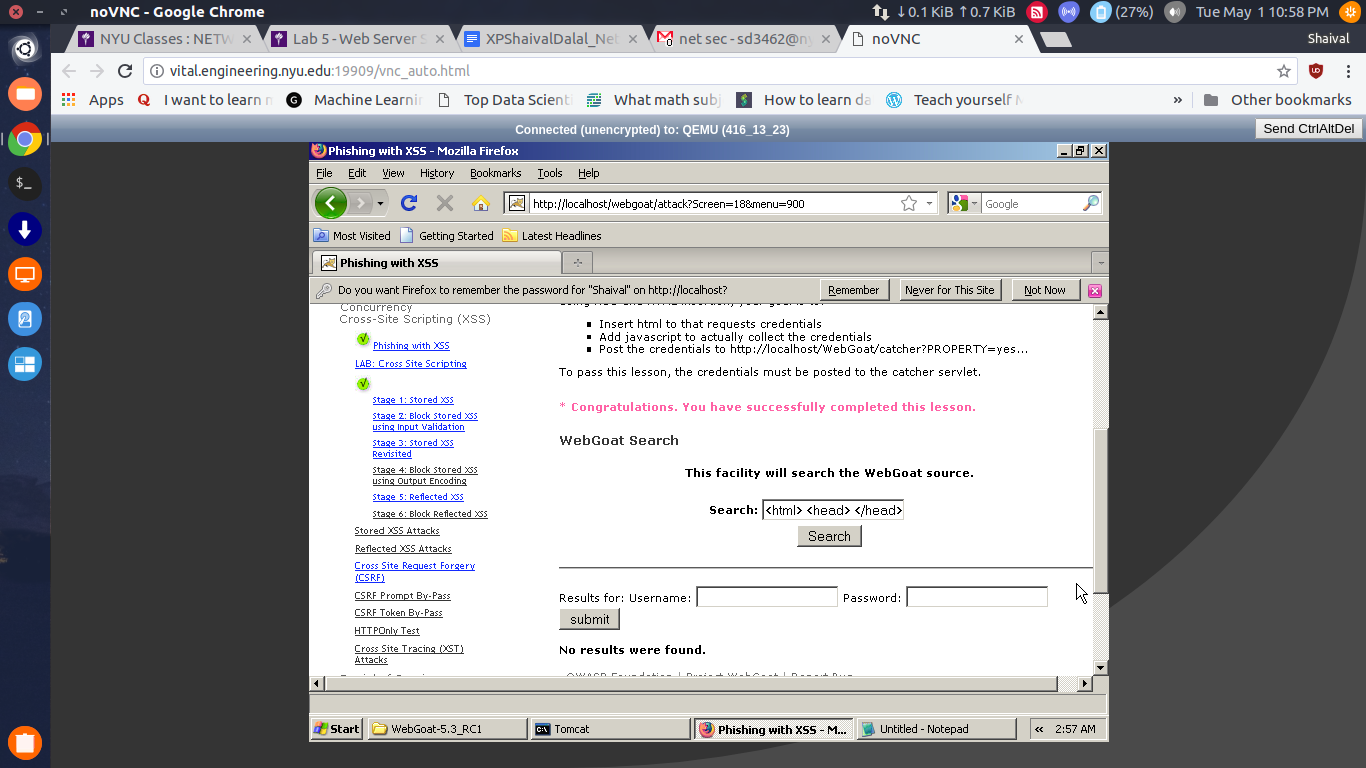
We insert the following HTML code in the input field in order to allow us to POST data

****

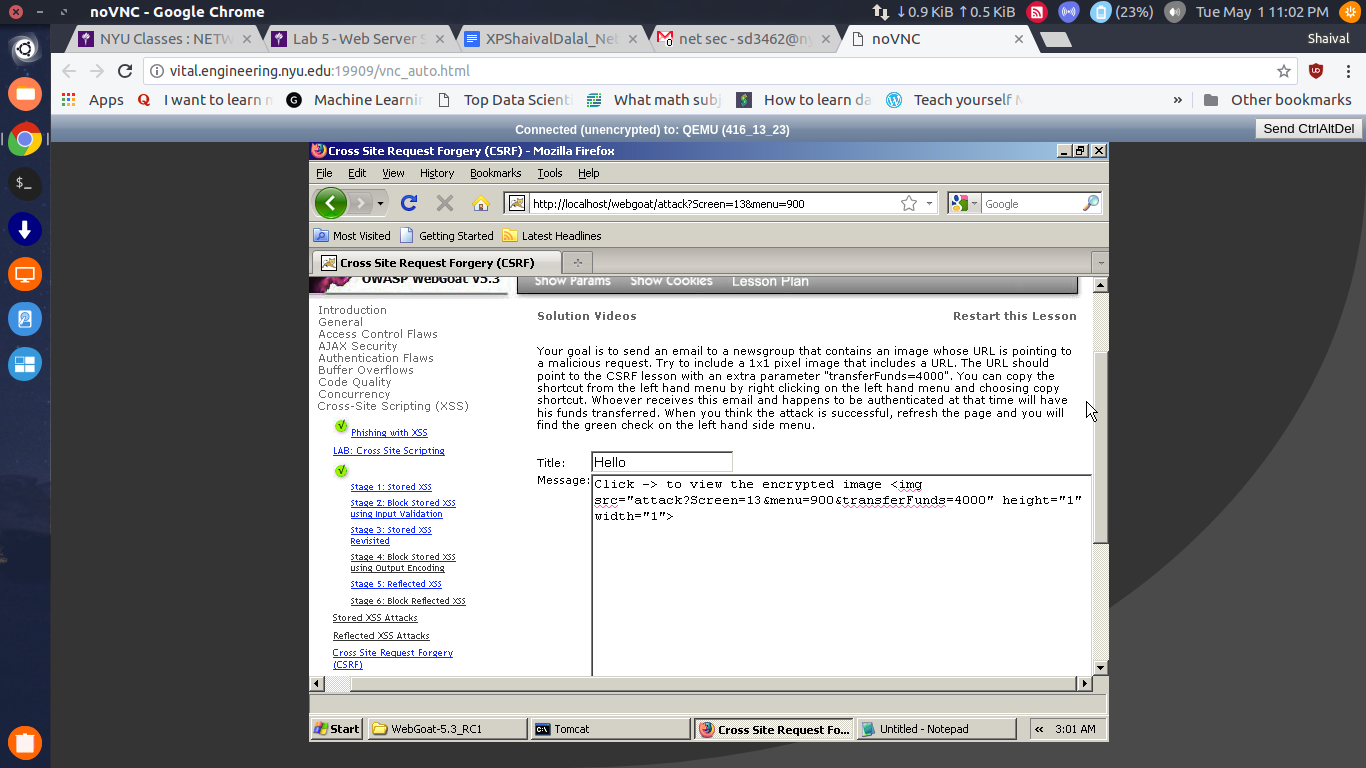
On entering the HTML, we see

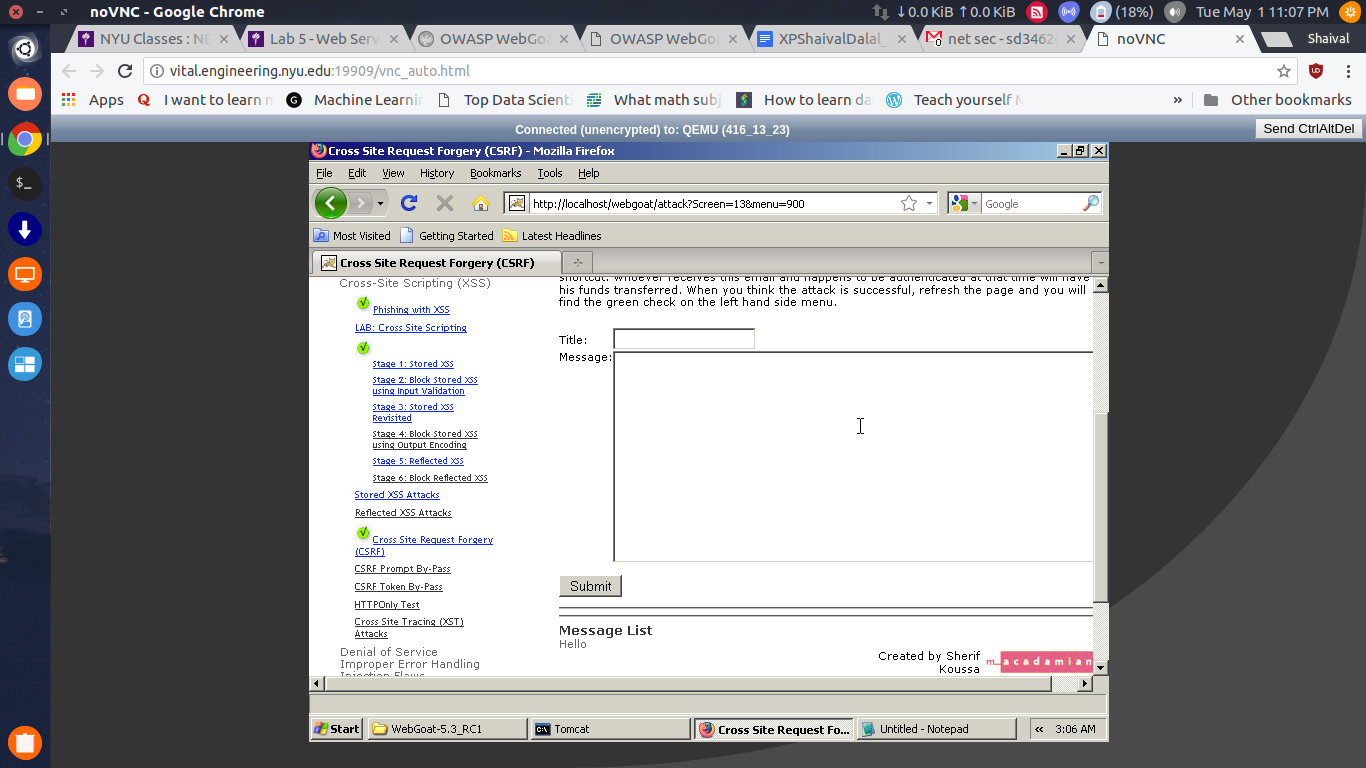


Result



* 1. **Cross Site Request Forgery (CSRF)**

****

****

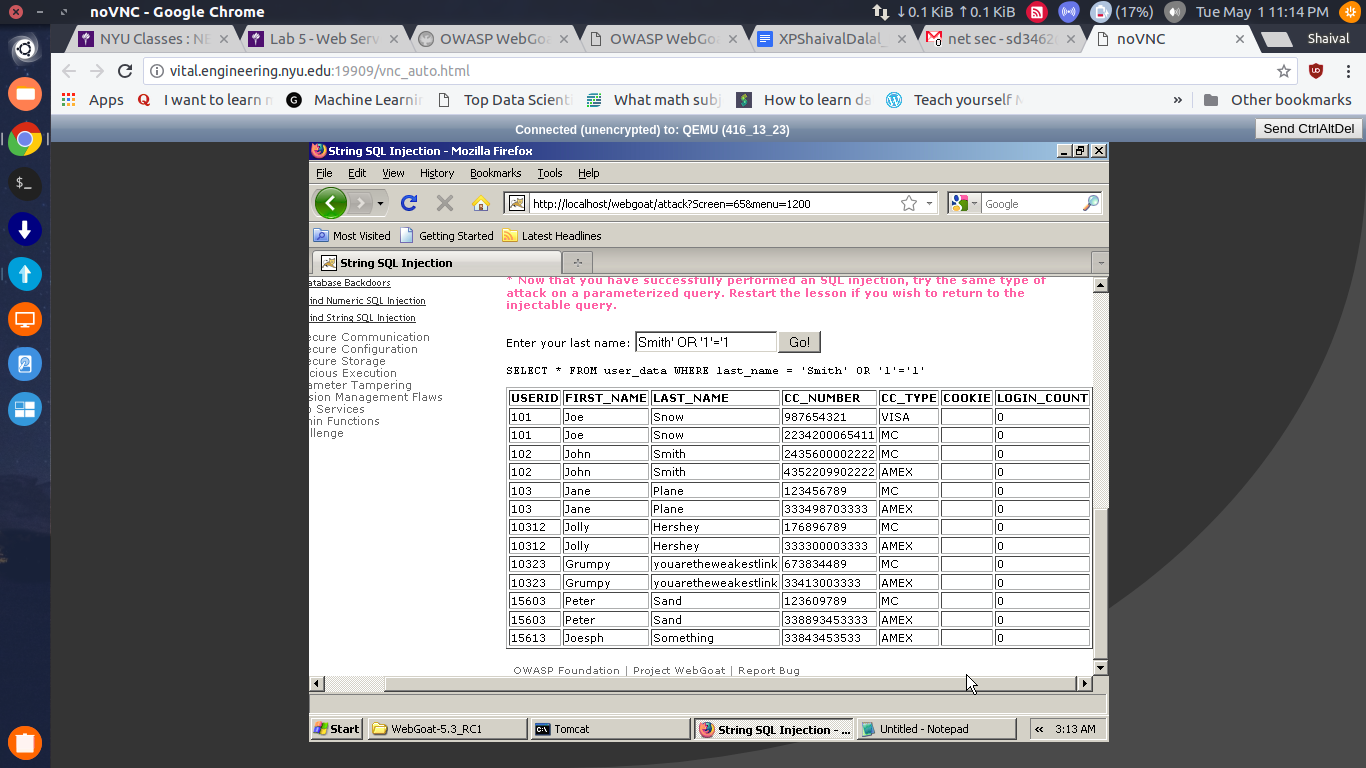
1. SQL Injection Flaws
   1. **Stage 1: String SQL Injection**

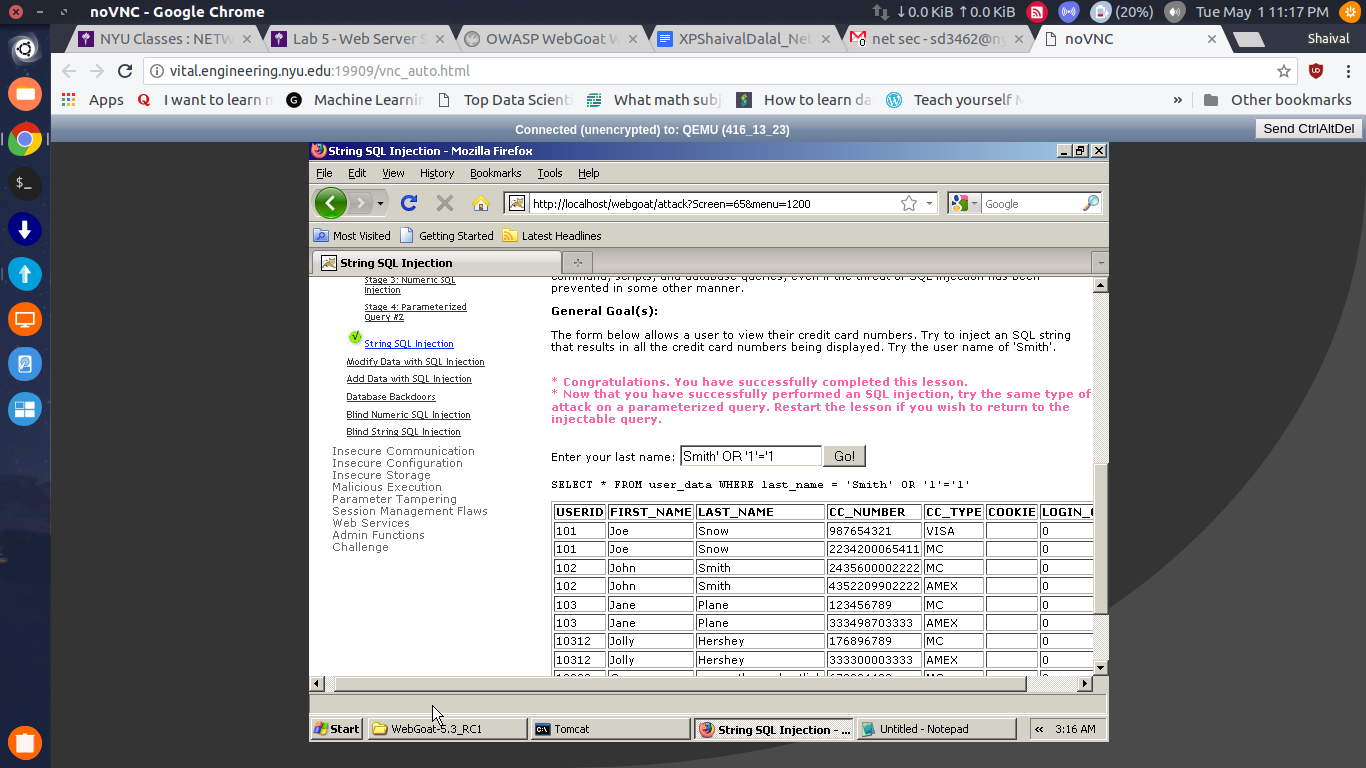
Statement used:

|  |
| --- |
| Smith' OR '1'='1 |

We use apostrophes in order to create manually end the statement and insert a new condition of 1=1 which will always be true. Hence returning all the records.

Output



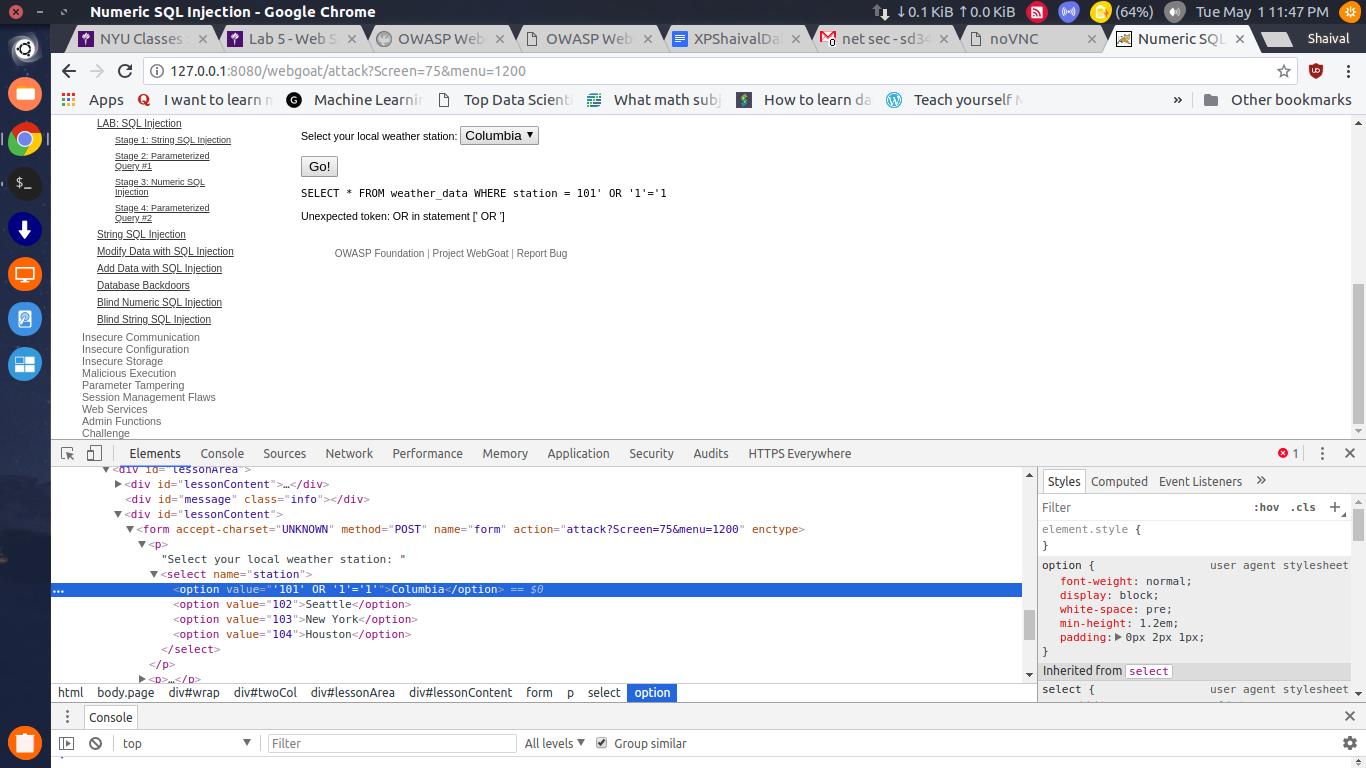


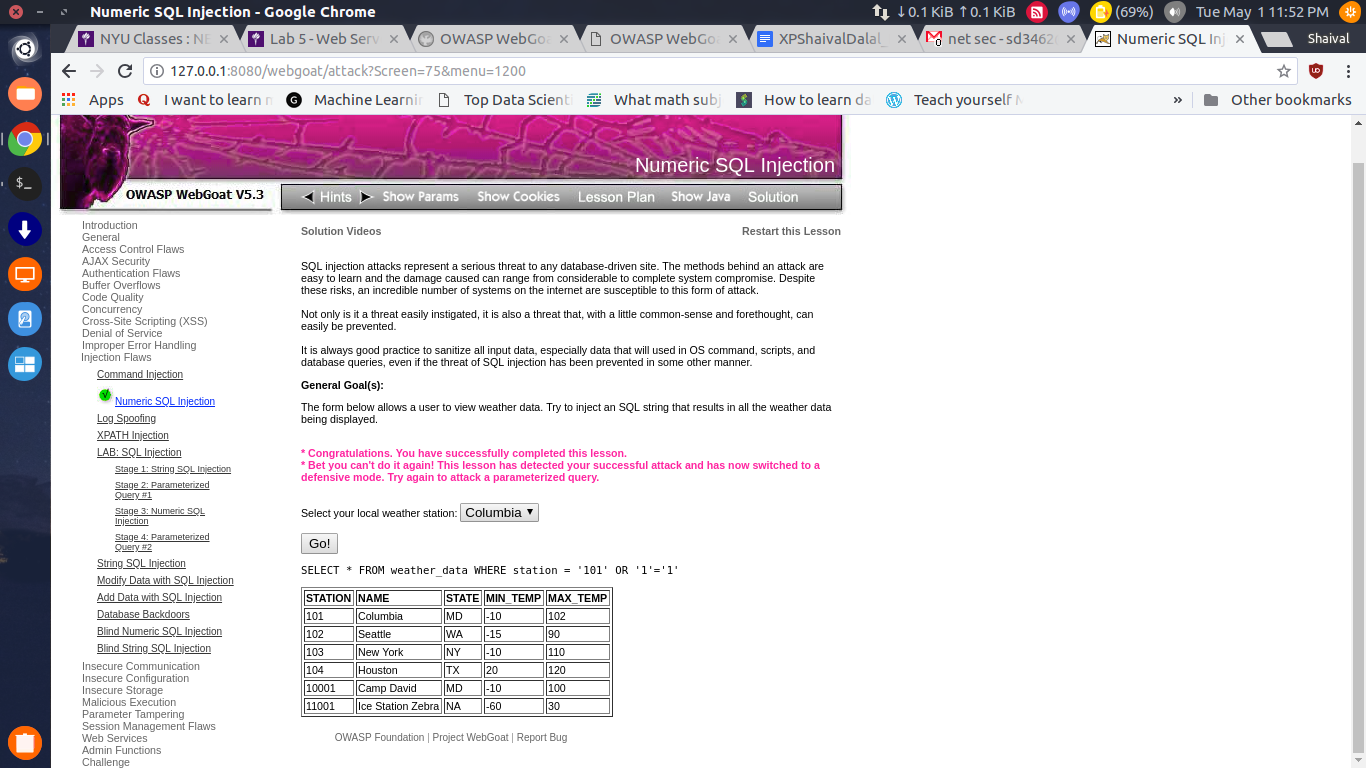
* 1. **Stage 3 Numeric SQL Injection**

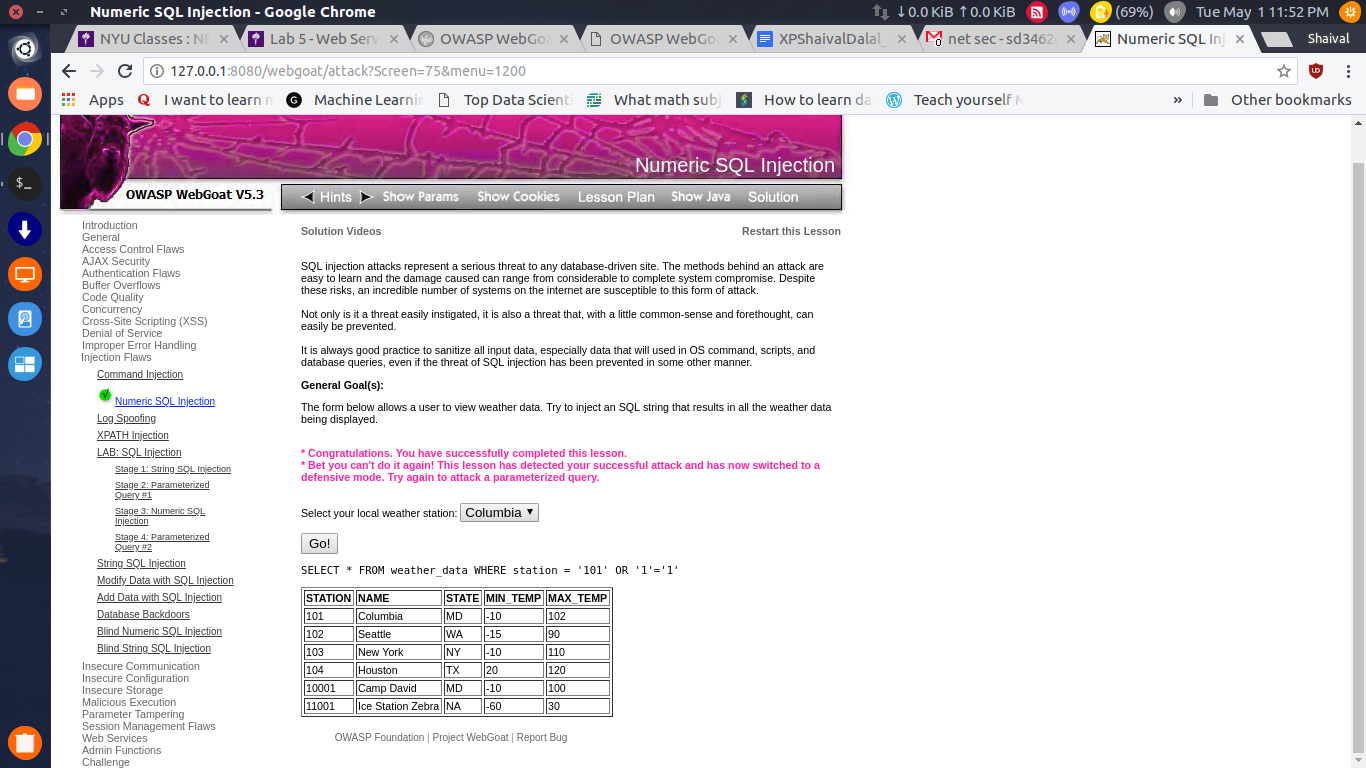
Statement Used:

Inserting it directly in Chrome’s page using Inspect Element

|  |
| --- |
| '101' OR '1'='1' |



Output



1. **Extra Credit**
   1. **Blind Numeric SQL Injection**

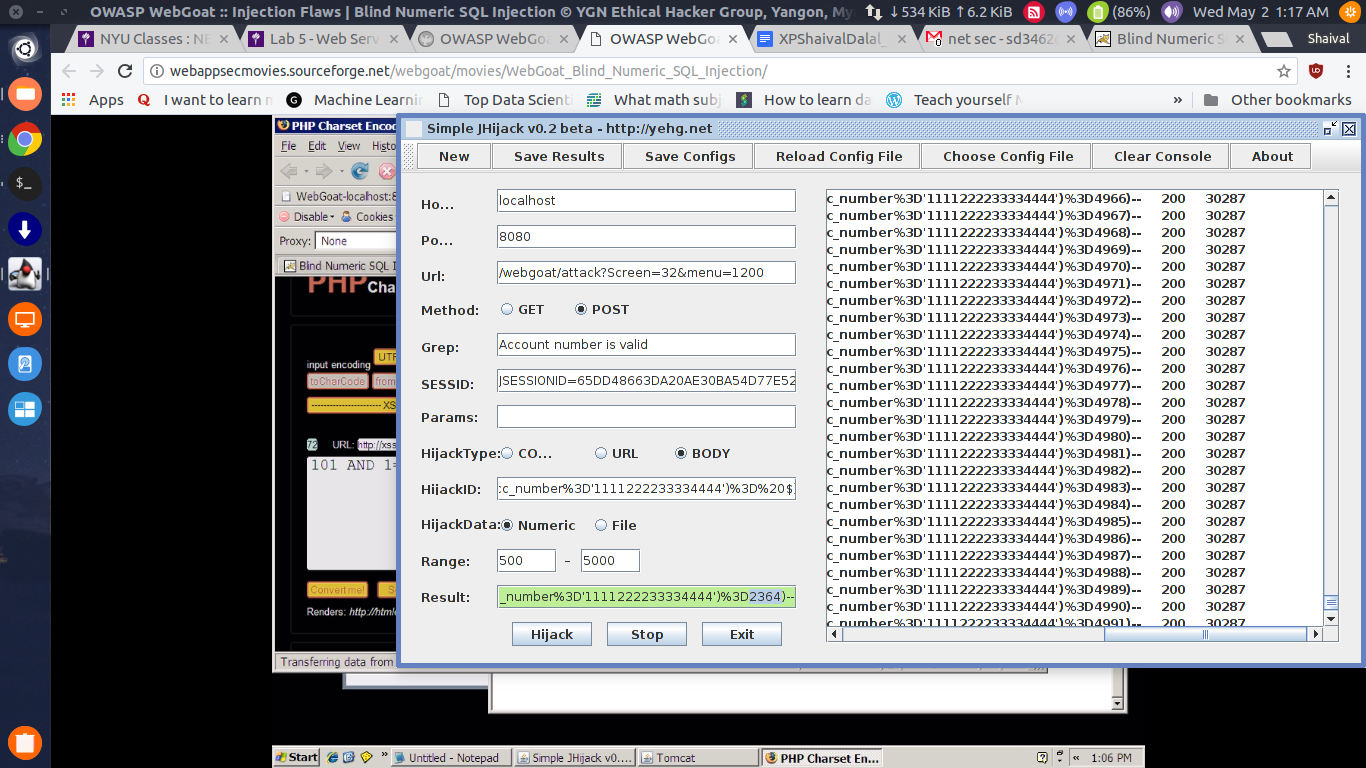
We use jHijack to run batch POST requests. We enter the success condition where the program will match the string entered in Grep field. In HijackID, we enter:

|  |
| --- |
| 101 AND 1=((SELECT pin FROM pins WHERE cc\_number='1111222233334444')=$ ) |

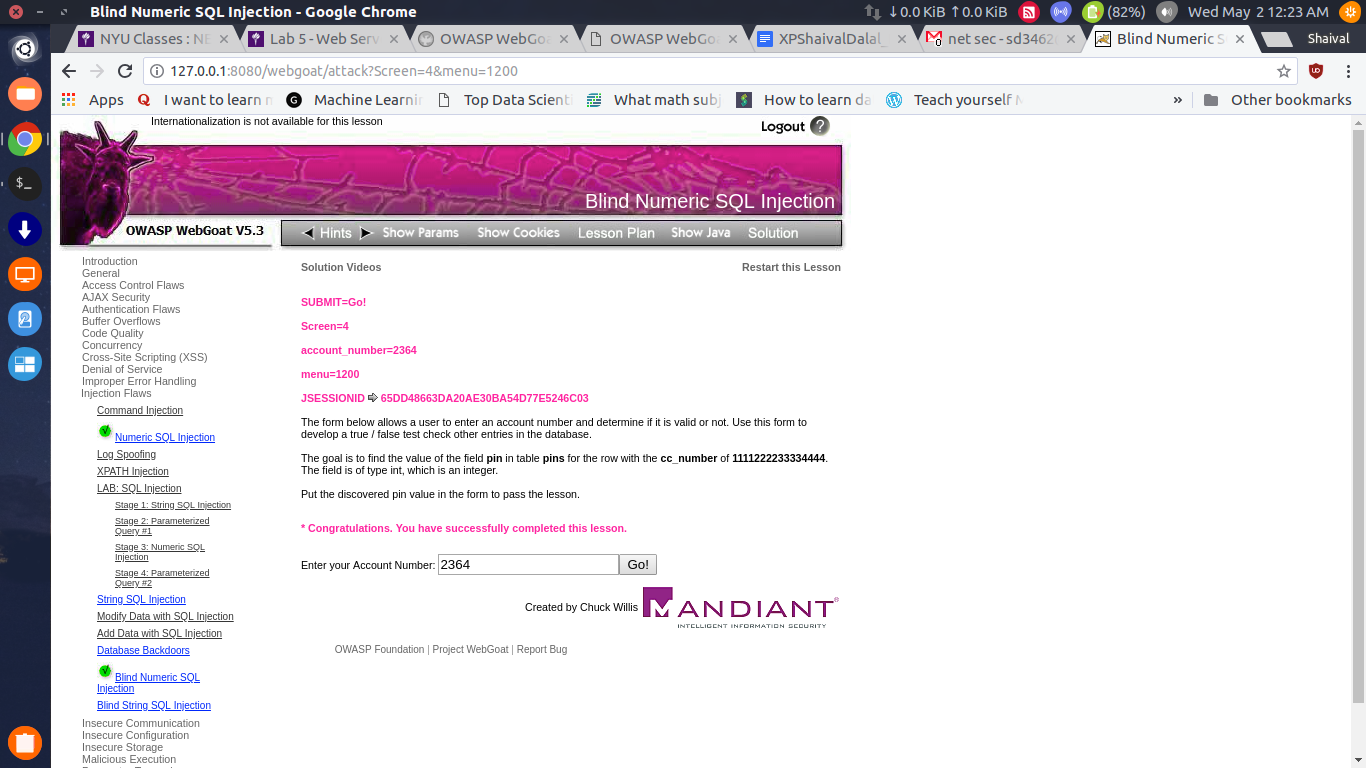
We encode it as:

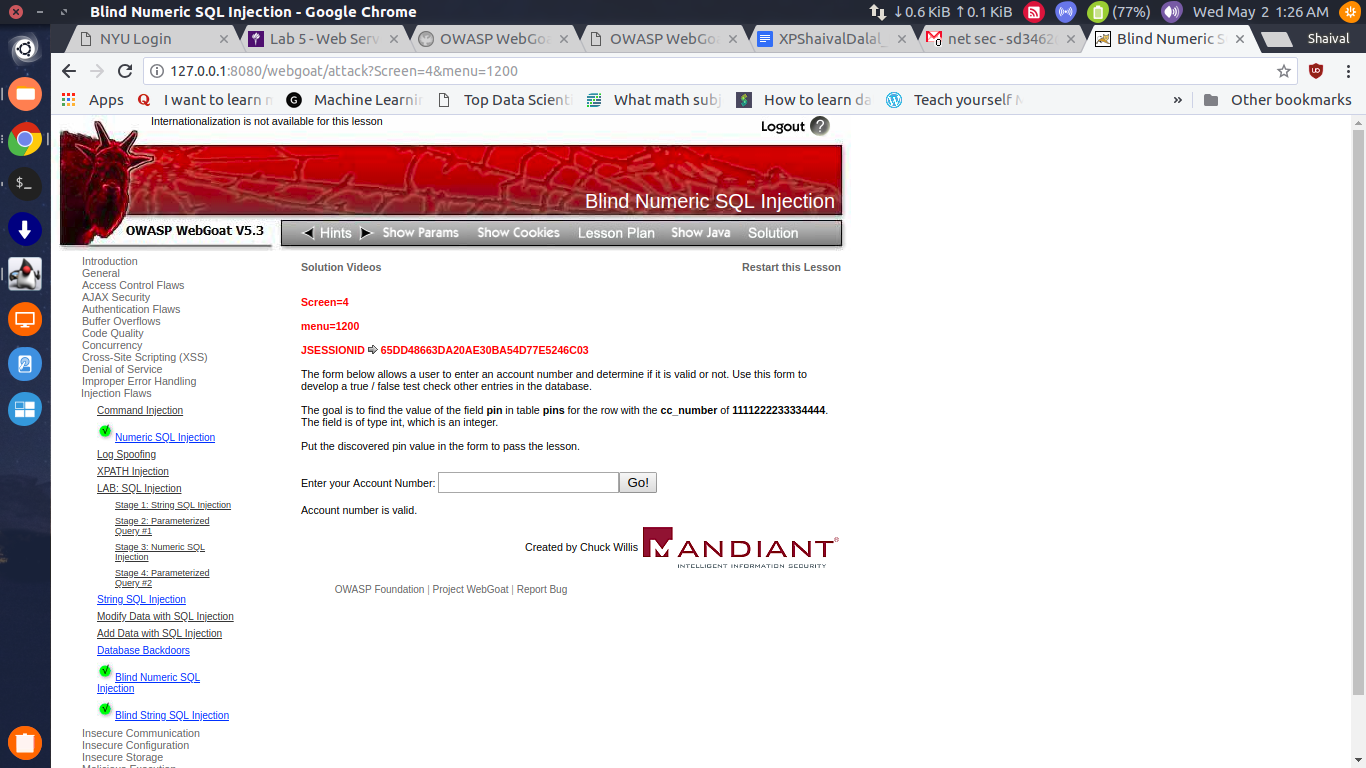
|  |
| --- |
| 101%20AND%201%3D((SELECT%20pin%20FROM%20pins%20WHERE%20cc\_number%3D'1111222233334444')%3D%20$) |

The range is set from 500 to 5000

****

Output

****

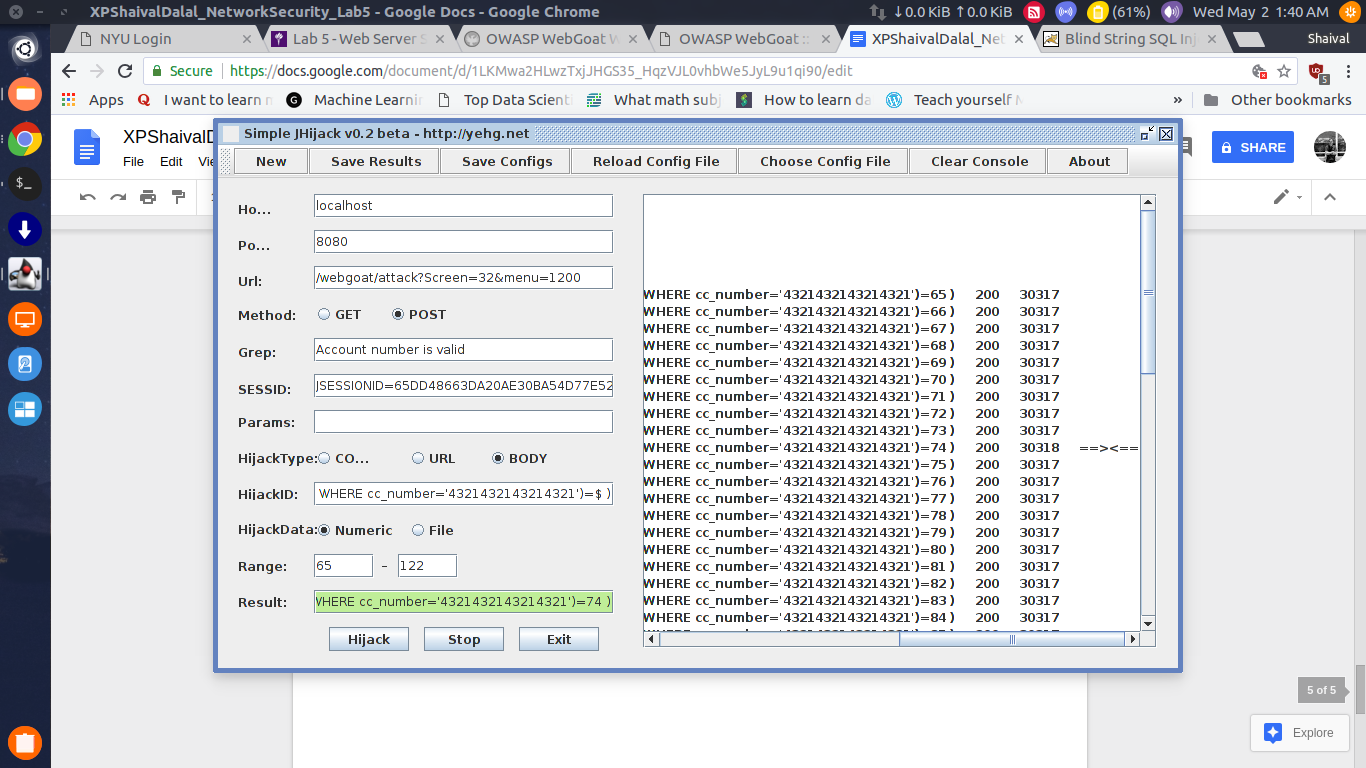
****

* 1. **Blind String SQL Injection**

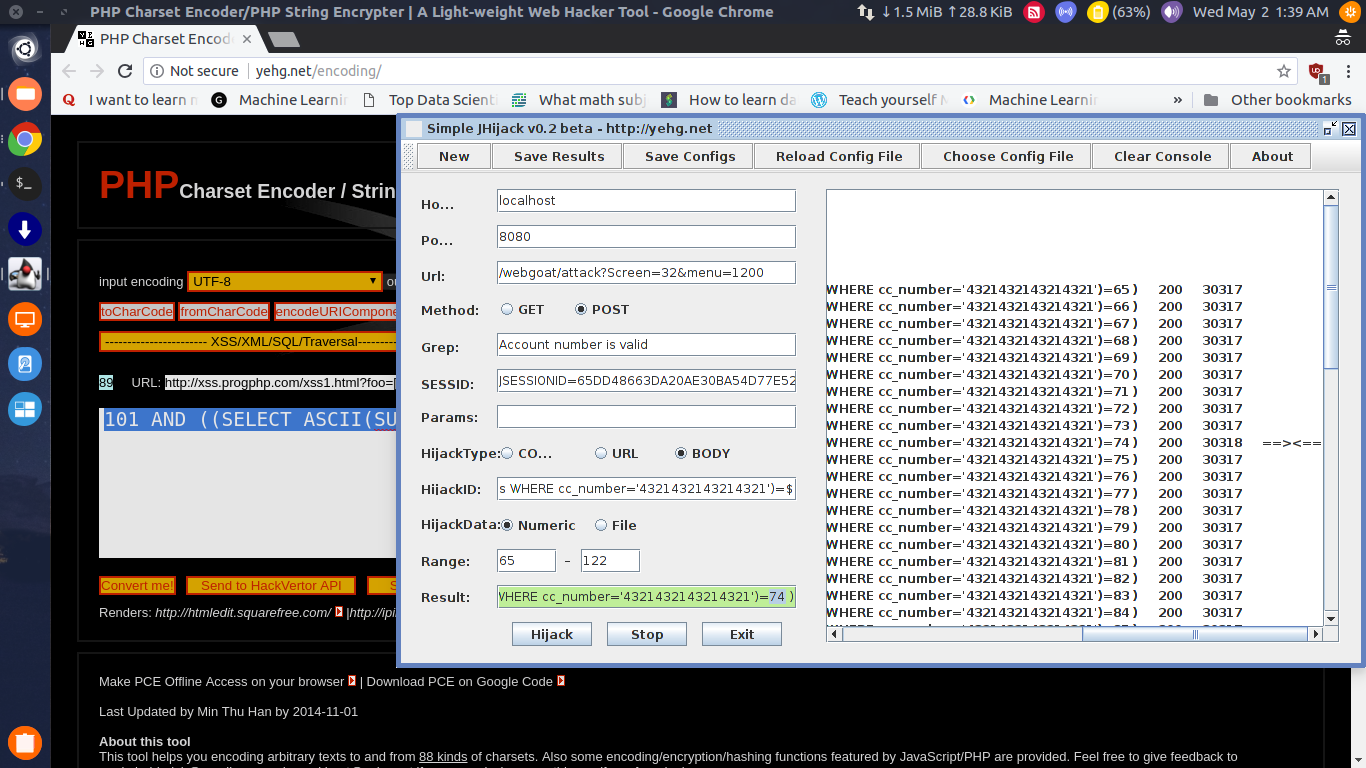
Similarly to Blind Numeric SQL Injection, we use jHijack to carry out Blind String SQL Injection

Statement used (HijackID):

|  |
| --- |
| account\_number=101 AND ((SELECT ASCII(SUBSTR(name,1,1)) FROM pins WHERE cc\_number='4321432143214321')=$ ) |

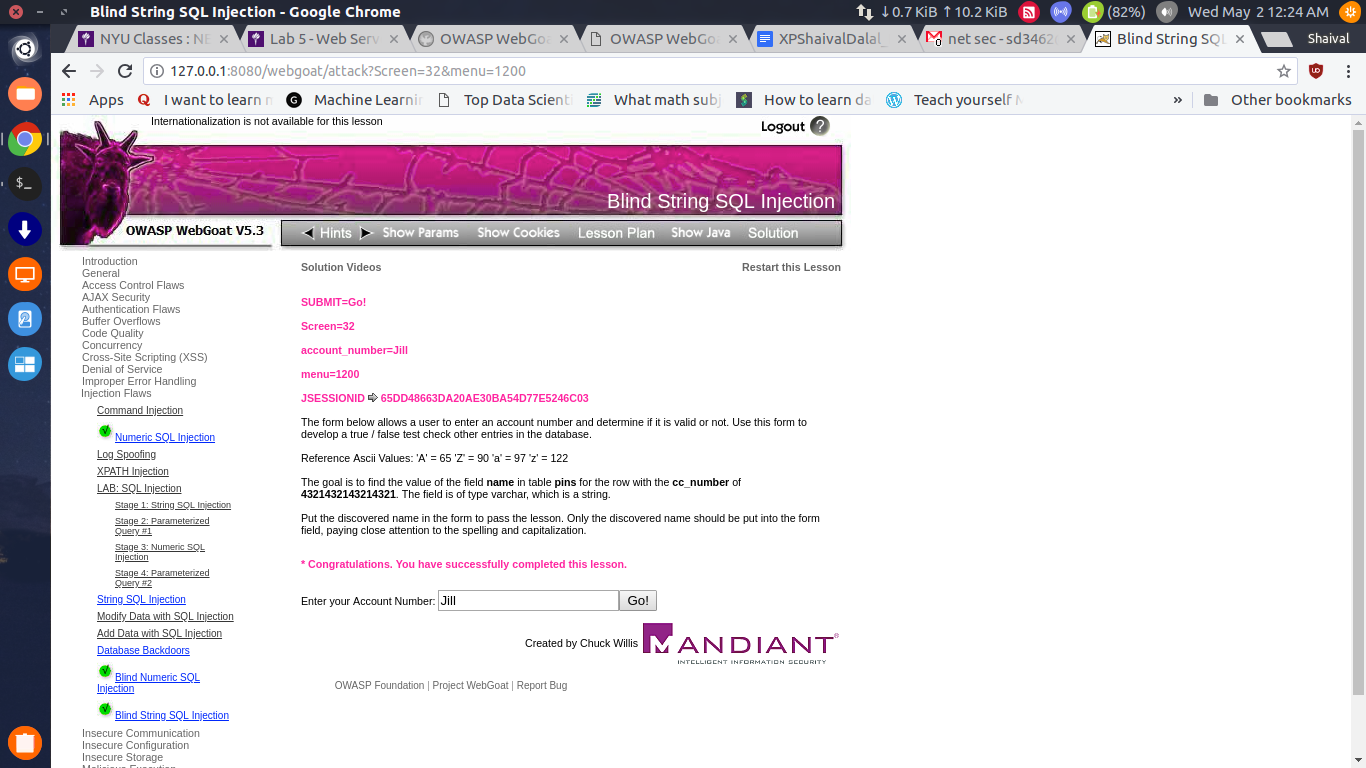


First alphabet’s ASCII code:



Similarly, by changing the substring position from 1,1 to 2,1 and subsequently 3,4, and 5. At “SUBSTR(name,5,1)” we get no result indicating that the name is 4 characters long

Our ASCII codes are: 74,105,108, and 108 which translate to “Jill”

****