

# Assignment-1 Report

Akhilesh Gowda Mandya Ramesh

**Abstract**—The Task given in Assignment-1 was to write a Automated script to login in as Tom for the SQL injection(Advanced) Task 5.

## I. INTRODUCTION

THE Task requires us to write as automated tool which sends requests to Webgoat and performs SQL injection to get relevant information from the tables in the database to log in as Tom. The challenge is to exploit the vulnerable field to get these information. The field has to be exploited manually as the tool will not be able to. Up on determining the vulnerable field or section we need to make use of it to get the relevant information, that might be needed to login as tom.

## II. DETERMINING THE VULNERABLE FIELD

The vulnerable field can be found by exploring the available pages the login page and the Register page. After exploring i found that the Register page is vulnerable as it displays messages like please proceed to a login page and so on. Whereas the login page does not provide any useful information. Hence the Register page has vulnerable field in the page which can be made use for SQL injection. It is crucial to determine the vulnerable field as this is where we will be asking the true or false questions using SQL injection. This is the most basic step in the process as we will have to manually determine which field is vulnerable by exploring the input fields.

### SQL Injection (advanced)

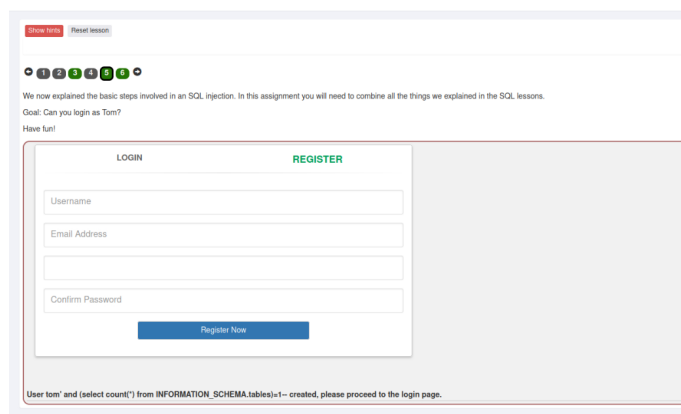


Fig. 1. Vulnerable field

Advisor: Luyi Xing, Jiale Guan.

## III. FINDING THE TABLE NAMES

first i started off by writing a query in the username field of the register page to find the total number of tables. We know that all the tables can be found in INFORMATION\_SCHEMA.tables hence we can ask true or false question here using blind SQL injection to exploit the vulnerabilities. The query was username='tom' and (select count(\*) from INFORMATION\_SCHEMA.tables)= count --, iteratively increment count until it throws an error. This way i was able to get the total number of tables which was 121. The number of tables which was found to be 121 was found again by asking true or false question in the username field or registration. By using this length we can now iterate so many finite times. Then i was able to fetch the names of all 121 table names using the query. The table names can be fetched by writing a query with a substring operator to try all the possible characters for each letter of the table name by changing the substring's start parameter. The SUBSTRING is of the form SUBSTRING(string,start,length). This was we can get the names of all the 121 tables iteratively. This part of the tool is automatic as it displays all the table names and the number of tables in the database. It took a significant amount of time to get the names of all the tables, the challenge was to be patient until it got all the table names. I copied the names of all the tables into notepad for the next step, where we determine which table is vulnerable that provides password for tom.

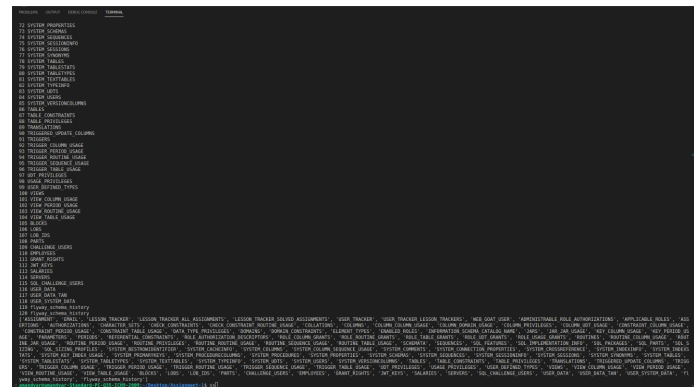


Fig. 2. Getting all the table names

## IV. FINDING THE VULNERABLE TABLE

After finding all the table names now the task is to find the vulnerable table in which we can find tom's password. This is a fairly laborious task as we will have to go through the all the 121 table names randomly and explore their columns to check if they contain the password column for the users. I manually went through the table names and explored them

based on their column names as it would determine the kind of data they often contain. This is challenging and laborious as we need to go through each table until we find a suitable table with password. Upon investing some time for checking table names checked for CHALLENGEUSERS columns and found out that it had the password column. This part of the tool is manual as we have to manually determine which table needs to be used to get the column names and to identify which table has the column which contains the password for tom. The challenge here was to check the columns of random tables who's name is suspicious enough to be vulnerable. After significant number of trial and error i found that the vulnerable table was CHALLENGE USERS as it had the USERID and PASSWORD columns in it.

## V. FINDING THE COLUMN NAMES FOR THE TABLE

The column names can be found for the table by performing the blind SQL injection by asking the true or false question using the SUBSTRING feature in SQL. After manually determining which table contains the password field. This part of the tool is automated provided we know the name of the table as we pass the name of the table which is CHALLENGE USERS as part of the query. we have to determine the length of the columns as we will have to iterate as many times as the length We will ask the true or false question character by character to determine the names of each column using all the possible combination of letters. After querying i found out that challenge users had 3 columns which were USERID, EMAIL and PASSWORD. Hence we can use this table to find the password for User id tom. If the name of the table does not return the PASSWORD and USERID field we have to keep changing the names of the table manually to determine the column names. Hence this part of the tool is fairly challenging and pushes the limits to try out suspicious table names.

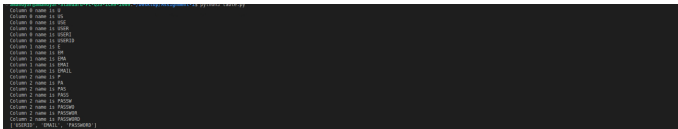


Fig. 3. Getting the column names from Challenge Users

## VI. GETTING THE PASSWORD

Now that we know the columns of the CHALLENGE USERS table which were USERID, EMAIL and PASSWORD. We can ask true or false question again here to perform blind SQL injection to determine the password by using the SUBSTRING in the query. The query asks true or false question character by character for the USERID = tom. Before this we have to determine the length of the password as we will have to iterate as many times as the length. This part of the tool is automated as well as we already know the columns in the CHALLENGE USERS. The figure getting password determines the password which is thisisasecretfortomonly. This password can now be used for logging in as tom by entering the username as tom and the password as thisisasecretfortomonly. The figure Task completion determines that

the task was completed upon entering the password found using the automated tool. The getting part of the password using the tool is automated, But we need to manually enter the password in the webgoat page to complete the task and to finish the assignment.

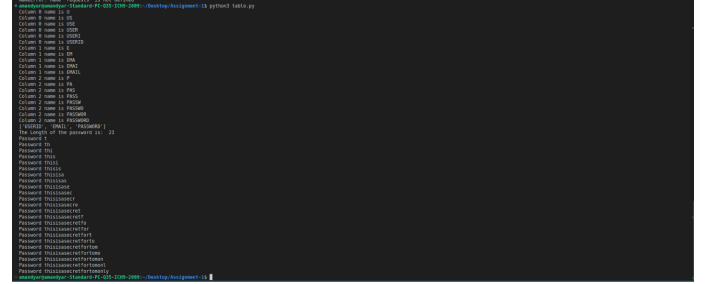


Fig. 4. getting the password

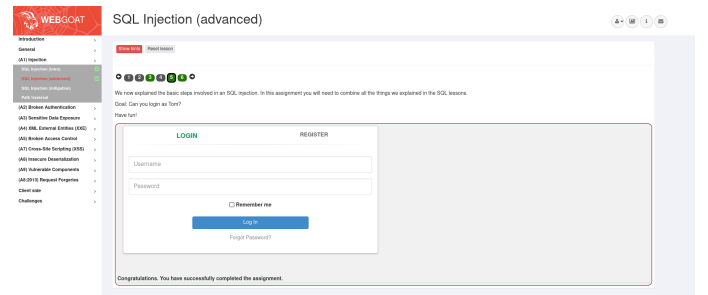


Fig. 5. Task completion screenshot

## VII. CONCLUSION

In Summary Automated scripts such as this can be written to exploit the vulnerabilities using Blind SQL Injection. Web-Goats vulnerabilities can be exploited to study and bolster the attacks. The CIA Triads can be compromised by performing Blind SQL Injection by using such as automated tool. This can be very harmful or costly for the organizations as they will result in a very harsh damage. This was we can used the TRUE or FALSE result from SQL Injection technique to appropriately guess the range of the user input, leading a way to get the value. This can be avoided by making the application to not display the generic error output, without which we cannot guess the value using SQL injection. There are various challenges that we face while designing the tool, right from determining the vulnerable field to determining the password. The series of steps has to be carefully planned as there are steps in the automated scripts that need to be done manually. We can make the tool better by trying to convert manual steps into automated ones, but there are numerous challenges with it and the automated script might take a significant amount of time to run. Thus making the script run efficiently by minimizing manual steps and reducing the run time is a huge challenge.

## REFERENCES

- [1] <https://github.com/WebGoat/WebGoat>
- [2] <https://owasp.org/>
- [3] <https://www.zaproxy.org>
- [4] [https://en.wikipedia.org/wiki/Information\\_schema](https://en.wikipedia.org/wiki/Information_schema)
- [5] <https://www.mssqltips.com/sqlservertutorial/196/information-schema-tables/>