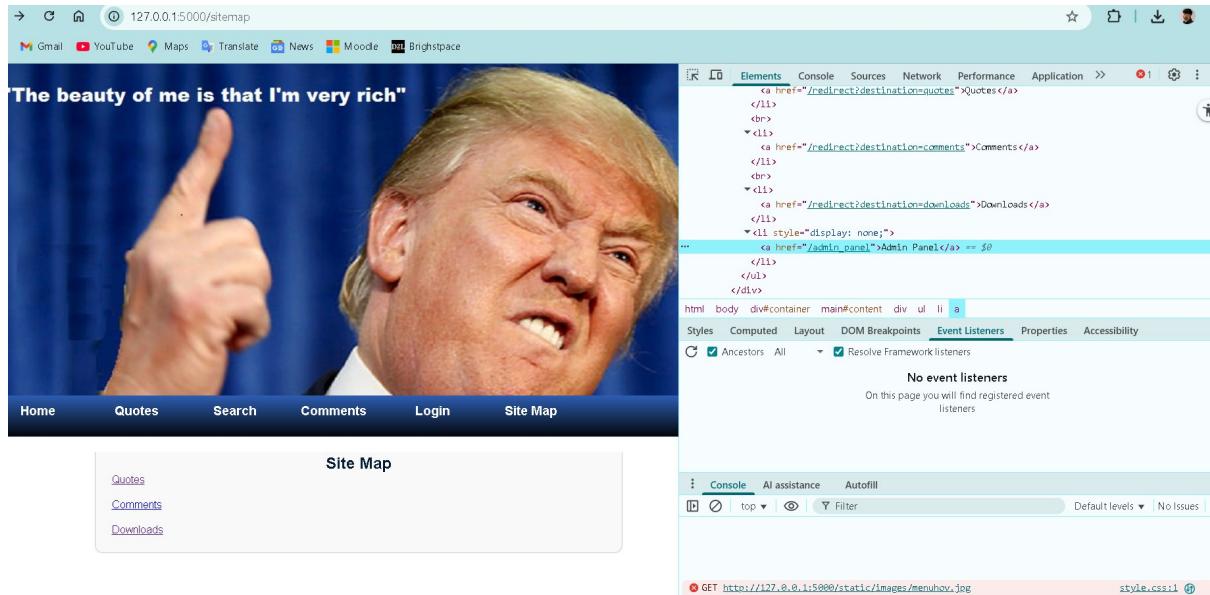
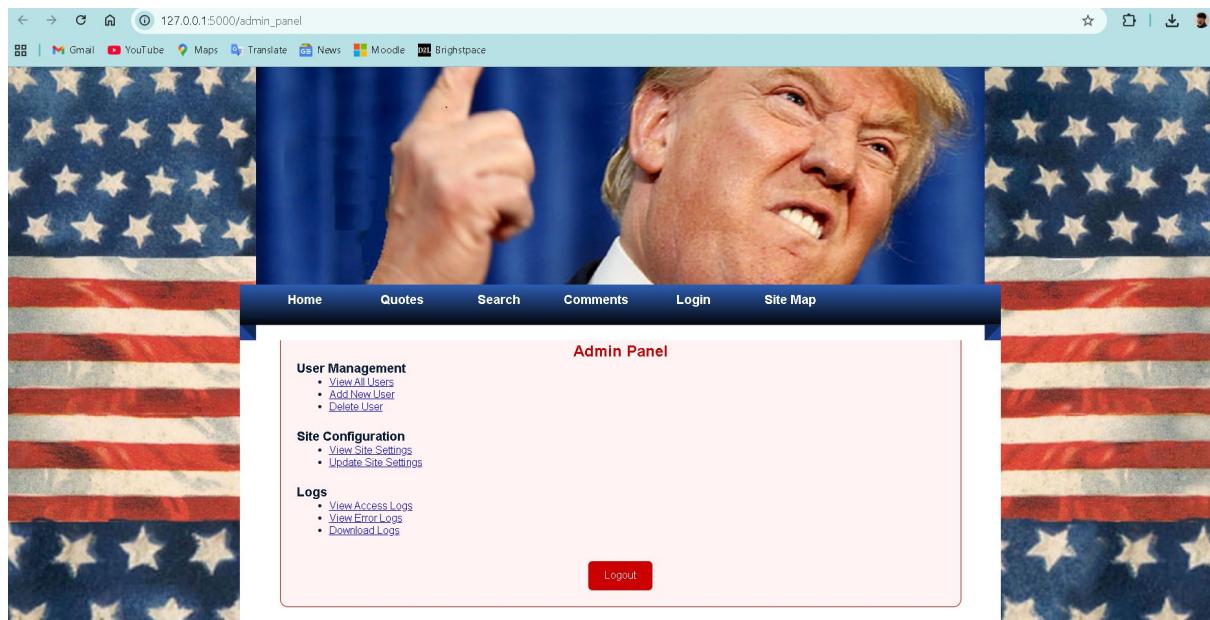


Access Control Vulnerability was found as within the sitemap in the page source there is a link to the admin panel



The screenshot shows a browser window with the URL `127.0.0.1:5000/sitemap`. The page displays a quote from Donald Trump: "'The beauty of me is that I'm very rich'". Below the quote is a navigation bar with links: Home, Quotes, Search, Comments, Login, and Site Map. The Site Map link is highlighted. To the right of the page is a developer tools interface. The Elements tab is selected, showing the HTML structure of the page. A specific link in the list is highlighted with a red box, reading `Admin Panel`. The Styles, Computed, Layout, DOM Breakpoints, Event Listeners, Properties, and Accessibility tabs are also visible. The Event Listeners tab shows "No event listeners" and "On this page you will find registered event listeners". The Console tab shows a single log entry: "GET http://127.0.0.1:5000/static/images/menubck.jpg style.css:1".

When clicked on, the admin panel is displayed and there is no proper authorisation or authentication checks



The screenshot shows a browser window with the URL `127.0.0.1:5000/admin_panel`. The page features a large background image of Donald Trump giving a thumbs-up. The navigation bar at the top includes Home, Quotes, Search, Comments, Login, and Site Map. Below the navigation is a sidebar with three sections: User Management (View All Users, Add New User, Delete User), Site Configuration (View Site Settings, Update Site Settings), and Logs (View Access Logs, View Error Logs, Download Logs). At the bottom of the sidebar is a red "Logout" button. The main content area is entirely blank.

The backend has no authentication whatsoever, and who ever wants to access it is free to do so

```
50
51  @app.route('/admin_panel')
52  def admin_panel():
53      return render_template('admin_panel.html')
54
```

To fix this, the code now has a key for the administrator, so if anybody was now able to try and login without the code it will now not be possible.

```
@app.route('/admin_panel')
def admin_panel():
    key = request.args.get('key')
    if key != 'ADMINPASS1234':
        abort(403) # Forbidden
    return render_template('admin_panel.html')
```

If we check now, the website does not allow access to the admin panel

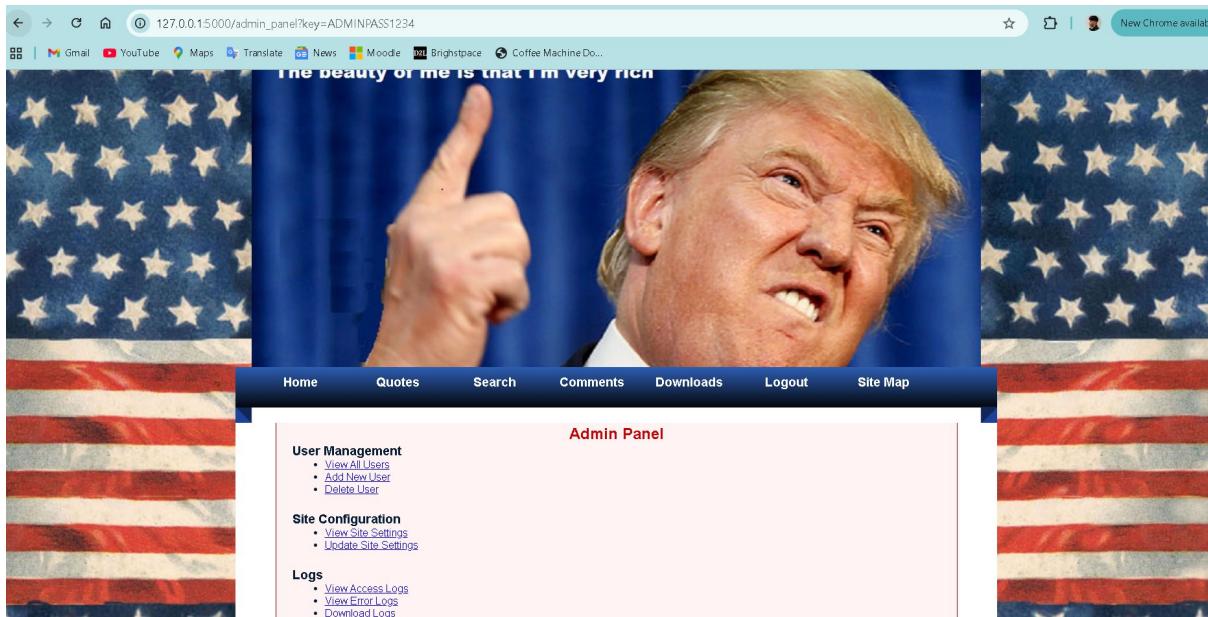


Forbidden

you don't have the permission to access the requested resource. It is either read-protected or not readable by the server.

+3

However if we try with the key in the address bar



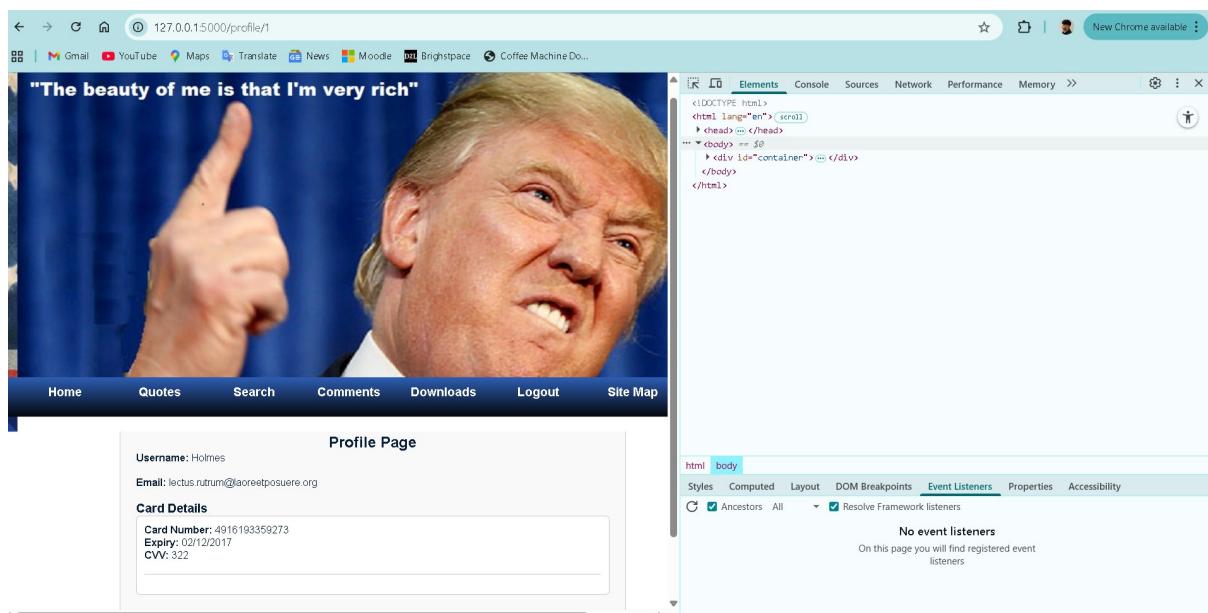
We now have access to the admin panel

Insecure Direct Object Reference vulnerability was found. To do this we first needed any login detail. since the website does not have a sign up option, using the sql file provided we got one login,

I decided to use username " Holmes" and his password "MEC15DBF3XD" in this instance

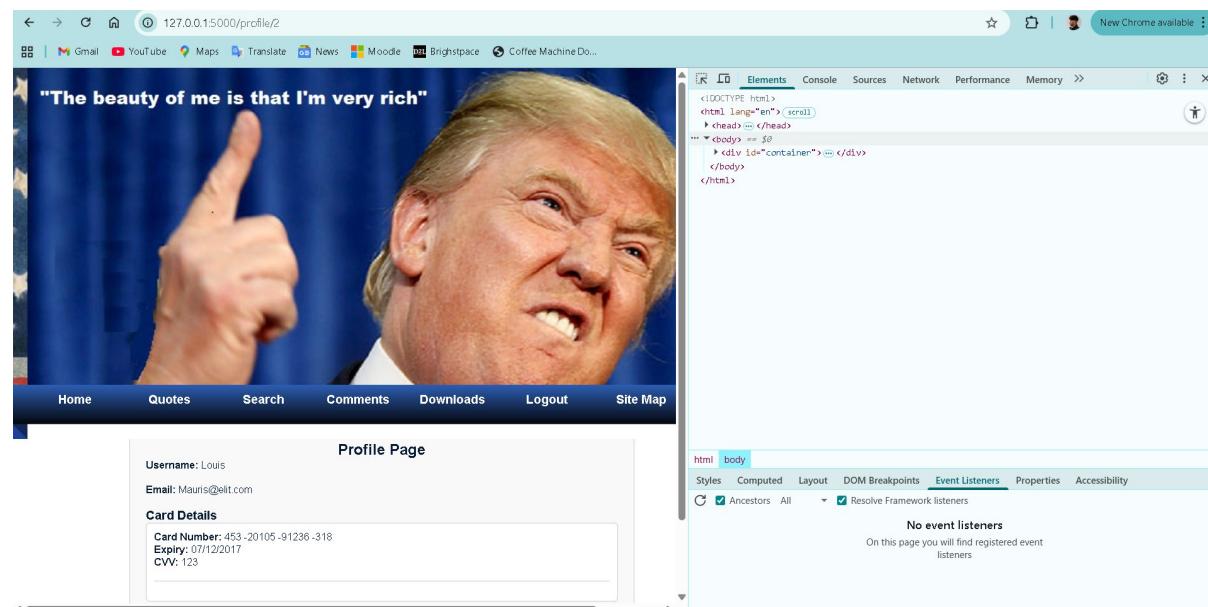
```
13 | 'user' TEXT DEFAULT NULL
14 |
15
16 • INSERT INTO `users` (`username`, `password`, `email`, `about`) VALUES ("Holmes", "MEC15DBF3X
17 • INSERT INTO `users` (`username`, `password`, `email`, `about`) VALUES ("Jakeem", "FXL380BR0A
18 • INSERT INTO `users` (`username`, `password`, `email`, `about`) VALUES ("Stuart", "A0042CAE6S
19 • INSERT INTO `users` (`username`, `password`, `email`, `about`) VALUES ("Kamal", "XUG95SXT6EY
20 • INSERT INTO `users` (`username`, `password`, `email`, `about`) VALUES ("Bruno", "RKV77YJT3UO
```

Once I logged in this was the page I had access to:



The screenshot shows a web browser window with the URL `127.0.0.1:5000/profile/1`. The main content area displays a profile page for a user named 'Holmes'. The page includes a large image of Donald Trump, a navigation bar with links like Home, Quotes, Search, Comments, Downloads, Logout, and Site Map, and a 'Profile Page' section with fields for Username, Email, and Card Details. The developer tools (Elements tab) are open in the bottom right corner, showing the DOM structure of the page.

On this page in the url bar it shows us that profile id is 1, using this, we decided to change profile by changing the 1 to a 2 and refreshing the page.



The screenshot shows a web browser window with the URL `127.0.0.1:5000/profile/2`. The main content area displays a profile page for a user named 'Louis'. The page includes a large image of Donald Trump, a navigation bar with links like Home, Quotes, Search, Comments, Downloads, Logout, and Site Map, and a 'Profile Page' section with fields for Username, Email, and Card Details. The developer tools (Elements tab) are open in the bottom right corner, showing the DOM structure of the page.

Just by doing that we had access to another users account this was because of IDOR, the site only checked what the user asked for and it never checked whether the user was allowed to ask for it, ie no authorisation – code below is missing authorisation

```
117     @app.route('/profile/<int:user_id>', methods=['GET'])
118     def profile(user_id):
119         query_user = text(f"SELECT * FROM users WHERE id = {user_id}")
120         user = db.session.execute(query_user).fetchone()
121
122         if user:
123             query_cards = text(f"SELECT * FROM carddetail WHERE id = {user_id}")
124             cards = db.session.execute(query_cards).fetchall()
125             return render_template('profile.html', user=user, cards=cards)
126         else:
127             return "User not found or unauthorized access.", 403
128
129 from flask import request
```

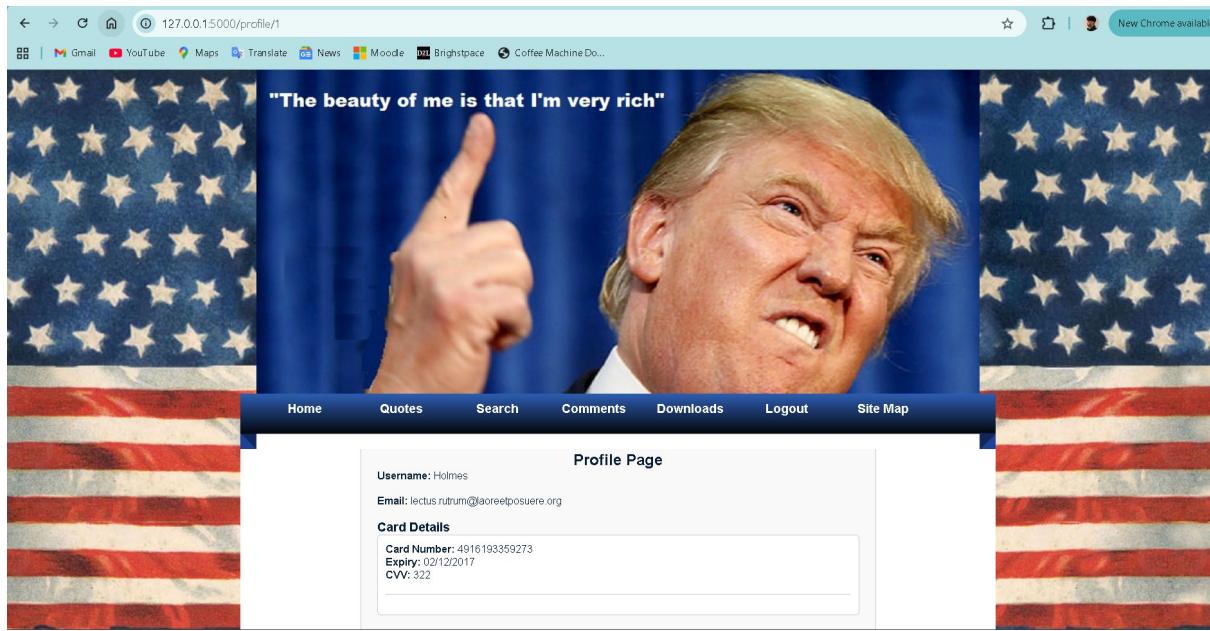
To allow for authorisation we have to add a line to the script

```
if session.get("user_id") != user_id:
    abort(403)
```

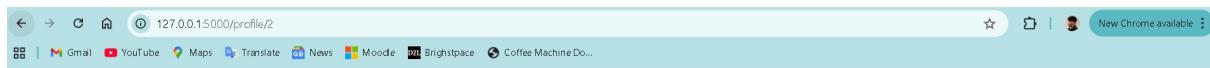
this compares the session ids and aborts if it is incorrect

```
117     @app.route('/profile/<int:user_id>', methods=['GET'])
118     def profile(user_id):
119
120         if session.get("user_id") != user_id:
121             abort(404)
122
123         query_user = text(f"SELECT * FROM users WHERE id = {user_id}")
124         user = db.session.execute(query_user).fetchone()
125
126         if user:
127             query_cards = text(f"SELECT * FROM carddetail WHERE id = {user_id}")
128             cards = db.session.execute(query_cards).fetchall()
129             return render_template('profile.html', user=user, cards=cards)
130         else:
131             return "User not found or unauthorized access.", 403
132
133 from flask import request
```

Now we can test to see if it works

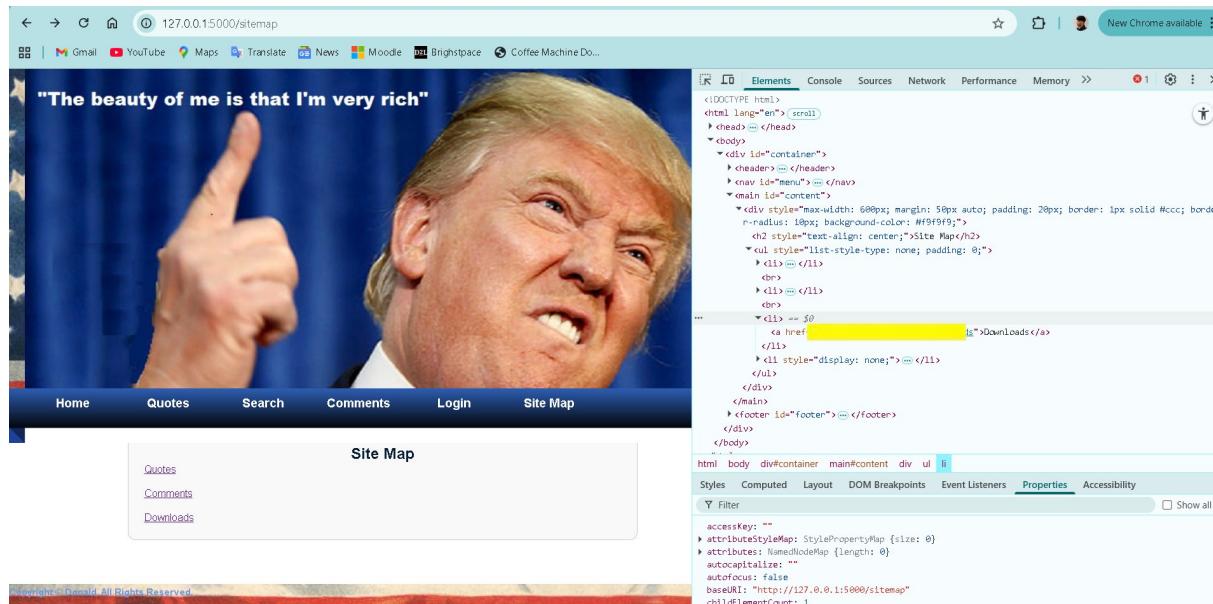


We are in profile 1, if we try change it to profile 2 like earlier:



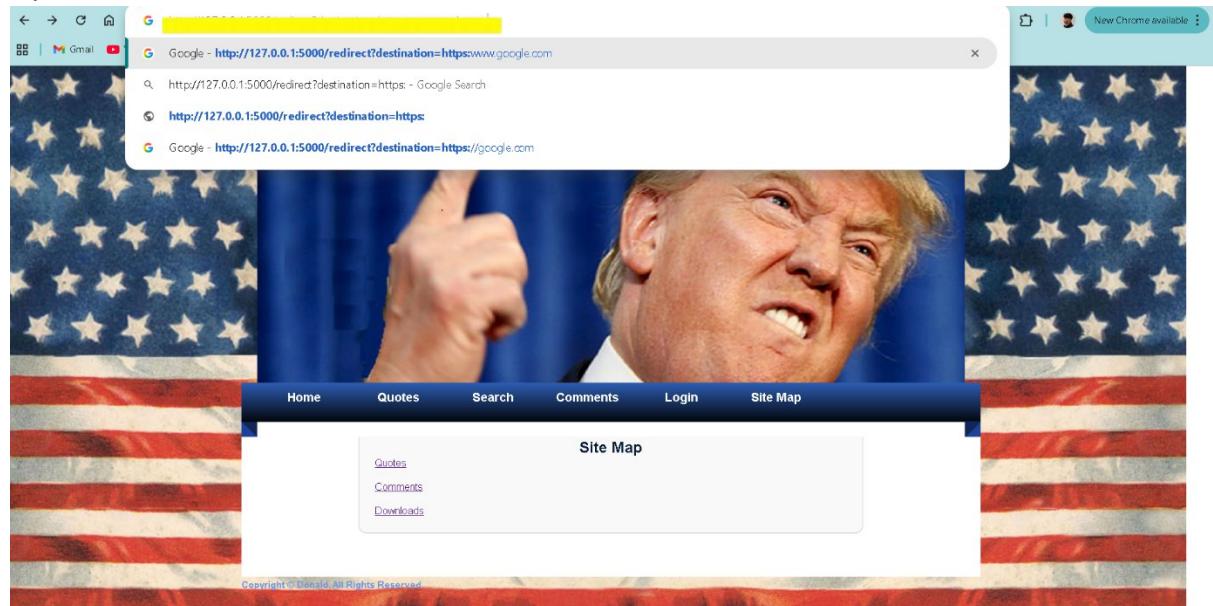
The System Aborts

An open redirect vulnerability was discovered within the sitemap of the page. The redirect element underneath(downloads/comments/quotes) allows for redirection of the webpage to an external source without any input validation



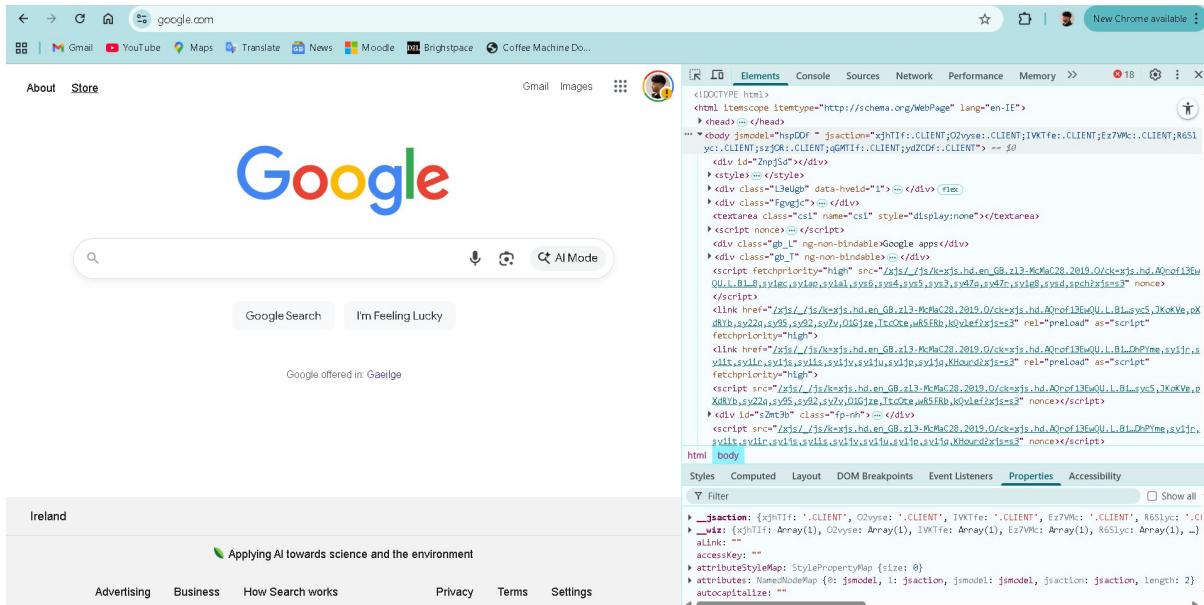
The screenshot shows a web browser window with the URL `127.0.0.1:5000/sitemap`. The main content features a large image of Donald Trump giving a thumbs-up. Below the image is a navigation bar with links: Home, Quotes, Search, Comments, Login, and Site Map. A sidebar titled "Site Map" contains three items: Quotes, Comments, and Downloads. The browser's developer tools are open, specifically the Elements tab, which displays the HTML code for the page. In the code, there is a `Downloads` link, which is highlighted with a yellow box. The browser's status bar at the bottom shows the URL `http://127.0.0.1:5000/sitemap`.

Same vulnerability was discovered in the url tab, allowing for a redirect into other websites without input validation



The screenshot shows a web browser window with the URL `http://127.0.0.1:5000/redirect?destination=https://www.google.com`. The main content features a large image of Donald Trump giving a thumbs-up. Below the image is a navigation bar with links: Home, Quotes, Search, Comments, Login, and Site Map. A sidebar titled "Site Map" contains three items: Quotes, Comments, and Downloads. The browser's address bar shows the URL `http://127.0.0.1:5000/redirect?destination=https://www.google.com`. A tooltip in the address bar shows the expanded URL `http://127.0.0.1:5000/redirect?destination=https://www.google.com`. The browser's status bar at the bottom shows the URL `http://127.0.0.1:5000/redirect?destination=https://www.google.com`.

Both when clicked on or pressed enter redirected to google as shown below



Problem in the code:

As you can see down below the code not check for any input validation and just redirects when asked

```
55     # Route to handle redirects based on the destination query p
56
57     @app.route('/redirect', methods=['GET'])
58
59     def redirect_handler():
60
61         destination = request.args.get('destination')
62
63
64         if destination:
65             return redirect(destination)
66
67         else:
68             return "Invalid destination", 400
```

To fix this: we add a line in the code which only allows for redirection to happen if its within the website or followed by the " / " sign

```
18     # Route to handle redirects based on the destination query
19
20     @app.route('/redirect', methods=['GET'])
21
22     def redirect_handler():
23
24         destination = request.args.get('destination')
25
26
27         if not destination.startswith("/"):
28             abort(400)
29
30
31         return redirect(destination)
```

And finally to prevent the code from aborting when we click on the sitemap to download, comments or quotes we have to go into html file for sitemap and then add a "/" before each of the locations

```
1 ..html" %}
2
3 }site Map{% endblock %}
4
5 %}
6 max-width: 600px; margin: 50px auto; padding: 20px; border: 1px solid #ccc; b
7 e="text-align: center;">Site Map</h2>
8
9 e="list-style-type: none; padding: 0;">
10 <a href="{{ url_for('redirect_handler', destination='/quotes') }}">Quotes</a></li>
11 <a href="{{ url_for('redirect_handler', destination='/comments') }}">Comments</a></li>
12 <a href="{{ url_for('redirect_handler', destination='/downloads') }}">Downloads</a></li>
13 style="display: none;"><a href="/admin_panel">Admin Panel</a></li>
14
15
16
17
```

And now when we try to execute open redirect this time it fails



Bad Request

The browser (or proxy) sent a request that this server could not understand.