PENETRATION TEST REPORT

Prepared by: Nursyaza Aida binti Azuan Amin

Prepared for: Juggy Bank Corporation

Report Issued: 22 June 2025

CONFIDENTIALITY NOTICE

Sensitive, privileged, and private information is included in this report. The confidentiality of the data in this document should be safeguarded by taking certain precautions. The release of this material could harm Juggy Bank's reputation or encourage attacks on the bank. Nursyaza will not be held responsible for any errors in it. Nursyaza makes no guarantees about the correction of all flaws in this work product. Nursyaza disclaims all obligations and liabilities, including but not limited to direct, indirect, incidental or consequential, special or exemplary damages resulting from the use of or reliance upon any information in this document, unless specifically stated in any master services agreement or project assignment. None of the businesses or goods listed in this document are impliedly endorsed.

Table of Contents

Document Control

Issue Control			
Document Reference	N/A	**Project Number**	N/A
Issue		**Date**	22/6/2025
Classification	Confidential	**Author**	Nursyaza Aida
Document Title	Web Application Penetration Test Report		
Approved by			
Released by	Nursyaza Aida		

Owner Details	
Name	Nursyaza Aida binti Azuan Amin
Email	syazaaida9@gmail.com

1.0 Executive Summary

This penetration test was successfully conducted on the web-based application https://www.juggybank.com to assess the security measures and ensure compliance with Requirement 11.3 of the Payment Card Industry Data Security Standard (PCI DSS). The goal was to identify vulnerabilities that could be exploited by unethical individuals, putting critical cardholder and company data at risk. The analysis focused on open-source programs and their associated interfaces, using both automatic and manual methods in accordance with widely acknowledged industry standards such as OWASP.

The assessment follows a black-box testing approach, representing an attacker with no system knowledge and no prior internal access. The testing focuses on publicly available web application components, including directory structures and resources that could potentially expose sensitive data or functionality. Below are the **main findings** of the vulnerabilities in this website:

Authentication bypass	**High**	Allow attackers to gain access to administrator page including user credentials and possibly cardholder data.
Vulnerable and outdated components	**High**	Attackers can take advantage of these vulnerabilities if they can launch malicious scripts or modify the application logic.
Security misconfiguration	**Medium**	Attackers can use it to obtain illegal access, retrieve sensitive information, or disrupt the entire system.
Broken authentication mechanism	**Medium**	Allow attackers to bypass login and impersonate legitimate users such as administrators.
Broken access control	**Medium**	Attackers can perform actions or access data that should not be permitted.

To evaluate the security of the web application www.juggybank.com, a penetration tester started with reconnaissance and host discovery. This procedure involved identifying the operating systems, applications, and services running on the target hosts utilizing port scanning tools, Nmap, and Dirb for directory enumeration. After

discovering open ports and services, a penetration tester listed possible vulnerabilities. Finally, to eliminate false positives and confirm any remaining vulnerabilities, a penetration tester tried to exploit all vulnerabilities affecting the target host. After extensive testing, only a few vulnerabilities were found to be present in the target host.

Test Scope and Objectives

Target URL	https://www.juggybank.com
Environment	Cardholder Data Environment (CDE) – Web Application
Type of Test	Black-box testing
Methodology	OWASP Testing Guide
Tools	Nmap, Dirb, OWASP Zap, Custom HTML
Timeframe	May 10 – June 22
Objectives	To discover exploitable vulnerabilities that could allow the attackers to access the cardholder data.

2.0 Statement of Scope

This penetration test covers all external systems, applications, and services that process, transmit, or store cardholder data, as well as security measures in these contexts. The study followed PCI DSS v4.0 guidelines and focused on vulnerabilities and misconfigurations that could jeopardize the confidentiality, integrity, and availability of systems in the cardholder data environment (CDE).

Since the engagement is a part of an external penetration test, the scope is limited to external-facing components. No internal network or physical systems were tested during this engagement.

In-scope System	System/Address	Details
External Web Applications	www.juggybank.com	The main public-facing client banking and payment portal. • **TLS Version:** TLS 1.3 • **Cert Subject:** Juggybank.com • **CA:** Valid EC certificate issued by Google Trust Services.
	https://script.viserlab.com/viserbank/	• Secure login and transaction management. • Authentication entry point for user accounts.
IP Address	172.67.166.161	Web Application and Firewall (Cloudflare)
	104.21.34.245	
Firewall	Cloudflare Http Proxy	Cloudflare acting as a reverse proxy and WAF for www.juggybank.com

The CDE systems in www.juggybank.com includes the system and network components that is store, process, and transfer the cardholder data. Based on the current testing, the following components were added and classified to the scope:

System/Component	Classification	Justification
https://www.juggybank.com	CDE System	The primary public-facing client banking and payment portal that handles the processing and transmission of cardholder data.
https://script.viserlab.com/viserbank	CDE System	It supports the login and transaction process management and act as authentication entry point.
Cloudflare HTTP Proxy (WAF)	Non-CDE System	Act as reverse proxy and WAF for the CDE. Does not store or process cardholder data but can impact CDE security.
IP Addresses: • 172.67.166.161 • 104.21.34.245	Non-CDE System	Public-facing IPs are linked to Cloudflare routing traffic to the CDE system.

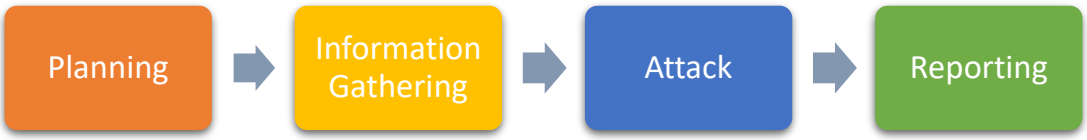
By using Cloudflare as a reverse proxy, www.juggybank.com used a secure design that separates sensitive backend systems from public-facing infrastructure. The origin web application server is still covered by CDE even though Cloudflare is not, particularly if it handles or saves CDE. This boundary is evident in all results and test coverage.

3.0 Methodology

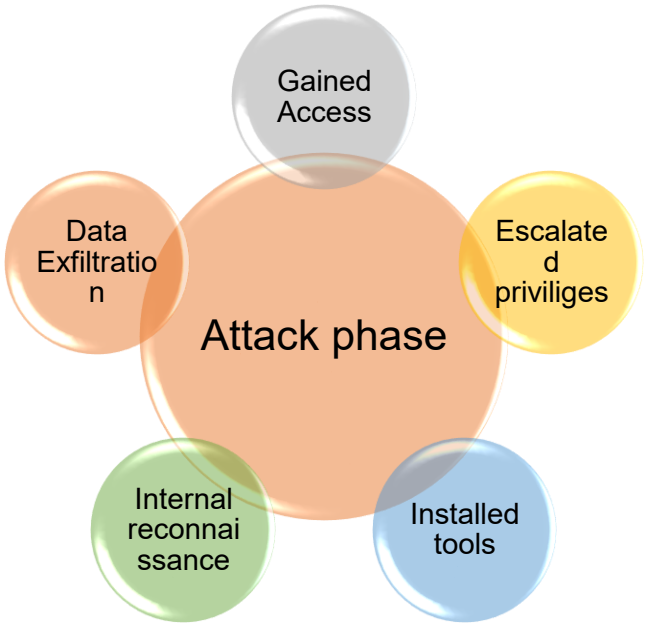
The testing was implemented in several stages.

1. During the planning phase, rules of engagement, test scope, test web application and goals were established.
2. During the discovery phase, automated vulnerability scanning and manual testing were used to analyse the test target and identify potential vulnerabilities.
3. The attack phase includes attempts to take advantage of any vulnerabilities found and to combine information about the environment, its technology, its users, and its functionality to escalate privileges beyond what the client intended.

4. The final phase documents all findings for the client to review and address potential risks. This includes writing this report.



During the attack phase, there are several steps included following the initial reconnaissance during the discovery phase:



1. **Planning and Rules of Engagement**

The planning phase lays the foundation for a secure, structured, and authorized penetration test for www.juggybank.com. During this phase, the scope of the engagement was specifically specified as encompassing the web application and its supporting interfaces. Before testing began, the types of tests, testing approach, and target systems were classified. This assessment mainly focused on **external network penetration testing**, following the **OWASP Web Security Testing Guide (WSTG)** methodology, covering issues such as authentication, session management, and access control. A **black-box testing approach** was implemented, where no internal knowledge or credentials were provided beforehand. The aim was to identify vulnerabilities that could be exploited from outside the organization's internal

environment. Industry standard tools such as Nmap, Dirb and OWASP ZAP were used, along with manual techniques including source code review and browser developer tool. All activities were designed in accordance with ethical and legal standards, to guarantee the interaction remained professional and non-intrusive.

2. Information Gathering

The information gathering phase involves obtaining publicly available data and observing the behaviour of the target application, www.juggybank.com to determine potential entry points and application components. Methods for active and passive reconnaissance were used. Analysing DNS records, checking through public search engine results, and looking at a website's SSL certificate and HTTP headers are examples of passive approaches. A penetration tester began with searching for **WHOIS Domain Lookup** to check for domain information and registrar information.

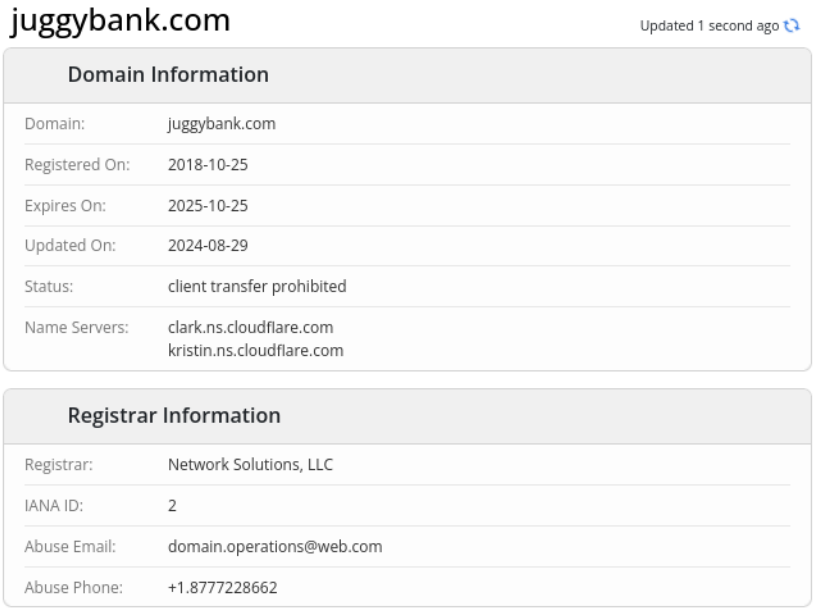


Figure 1: WHOIS Domain Lookup's Information

Additionally, DNS enumeration was performed to discover the IP address and hosting details in www.juggybank.com using **nslookup**. The DNS record's Time to Live (TTL) value, which indicates how frequently Cloudflare revalidates its cache with authoritative servers, was set to five minutes. This data demonstrates that Cloudflare is used by the domain to optimize security and performance, potentially reducing the number of direct lookups of the origin server.

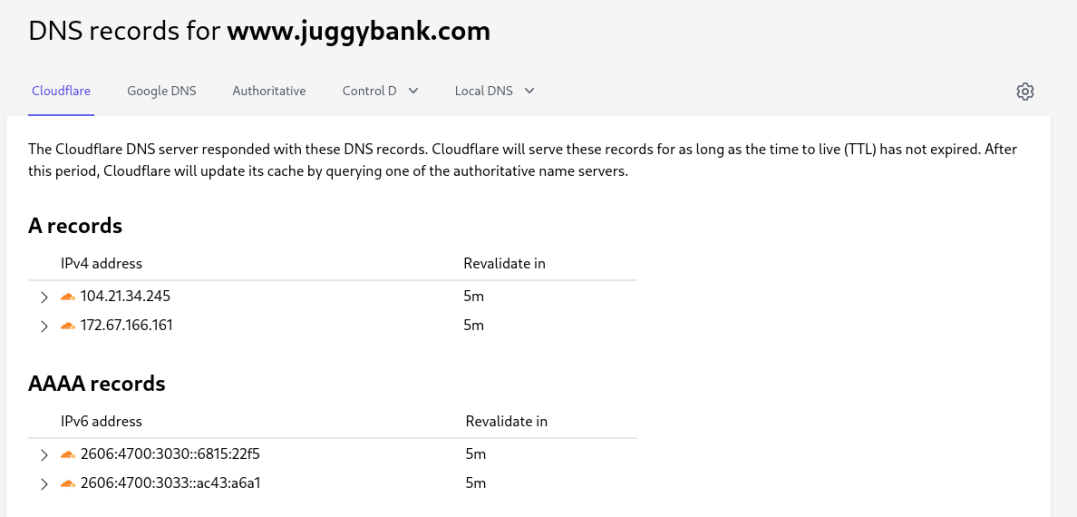


Figure 2: DNS Records

A passive SSL Certificate Analysis was conducted using **Qualys SSL Labs**. The analysis reveals that the certificate was issued for juggybank.com and its subdomain www.juggybank.com, with a 256-bit ECC key and the SHA256withECDSA signature method. The certificate is now valid, since it was issued by Let's Encrypt (R3) via Let's Encrypt Authority X1, and will become invalid on September 3, 2025. This research demonstrates that the website uses strong cryptography, however it should disable older protocols to improve its security posture.

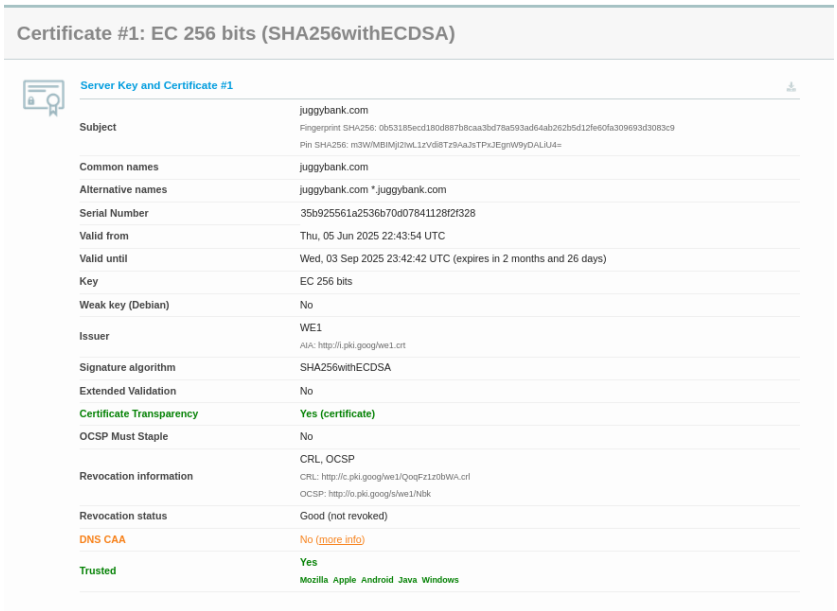


Figure 3: SSL Certificate Analysis in Qualys SSL Labs

A penetration tester also conducted the active scanning to explore the open port for www.juggybank.com using Nmap to scan the target system. The IP addresses, which are 104.21.34.245 and 172.67.166.61, were discovered. The scan indicated that the

host was active and located behind Cloudflare, indicating a virtual or cloud-based environment. Several open ports were found which is 80/tcp, which indicates a conventional web server 443/tcp, which indicates an SSL/TLS-enabled web server 8080/tcp, which is typically used for proxies or web application admin panels and 8443/tcp, an additional HTTPS port often used for secure admin interfaces.

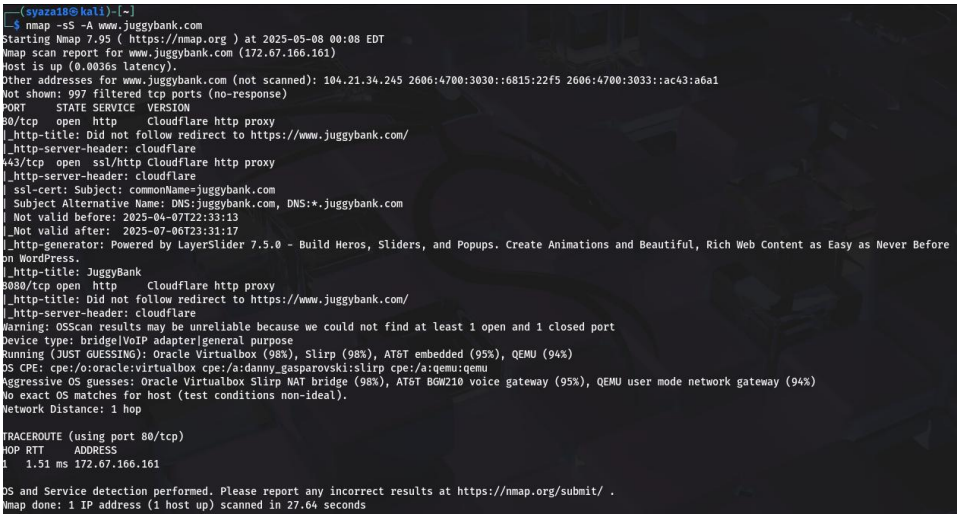


Figure 4: Nmap scan for open port

In addition to port scanning, a penetration tester used Dirb tools to enumerate hidden directories on the web application in order to locate potentially sensitive or unindexed content that may be accessible via HTTP/HTTPS services. One of the directories discovered on www.juggybank.com is the **admin.html** page.

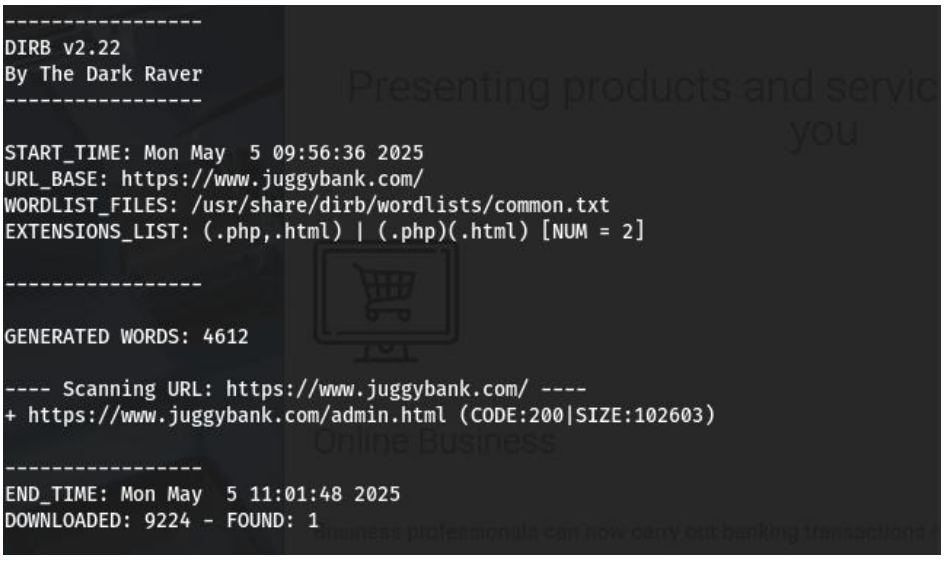


Figure 5: Utilized Dirb Tools to enumerate hidden directories

3. Vulnerabilities Analysis

During the vulnerability analysis phase, the www.juggybank.com/admin.html web application was scanned automatically using OWASP ZAP (Zed Attack Proxy). ZAP is an open-source application that specifically developed to discover vulnerabilities in web application that allows for identifying the common vulnerabilities aligned with OWASP 10 and OWASP Testing Guide. The common vulnerabilities found in the Juggy Bank website includes **security misconfigurations, vulnerable and outdated components, broken authentication mechanism and broken access control.** The tool performed passive and active scans, evaluating HTTP requests and responses to identify potential security issues. Passive scanning initially aims to identify vulnerabilities without changing the request, thus minimizing the risk to the live system. This was followed by active scanning, in which ZAP attempted to recreate actual attack situations in order to identify exploitable flaws. The OWASP ZAP findings gave a preliminary list of security concerns, which were verified and further investigated during the finding phase. Manual testing methods were also conducted using browser developer tools, parameter manipulation and inspection of HTTP responses. This method assists in detecting and validating potential vulnerabilities in input validation, session handling, and web application settings. All findings are recorded in the findings and testing narrative section, along with technical evidence and solution recommendations. The vulnerabilities that were identified are thoroughly evaluated to determine their real-world impact and usefulness.

- **Outdated Software and Libraries**

  The vulnerability of outdated software and libraries were also found in the website. These components could include old versions of frameworks, plugins, or third-party dependencies that have not been patched or modified. The inclusion of such components increases the possibility of exploitation by attackers who can use publicly identified vulnerabilities to compromise the system. A penetration tester begins by searching the National Vulnerability Database (NIST) for known vulnerabilities connected to the website. She looks for the keyword “moment.js” in the category of vulnerable JavaScript libraries.

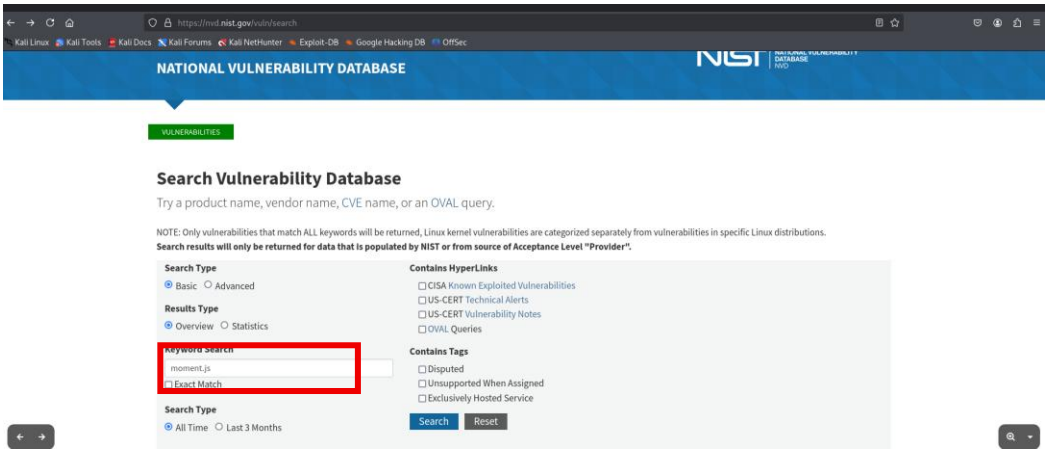


Figure 6: Moment.js search in National Vulnerability Database (NIST)

The search results show that the library is outdated. The screenshot below contains extensive information about the identified vulnerabilities.

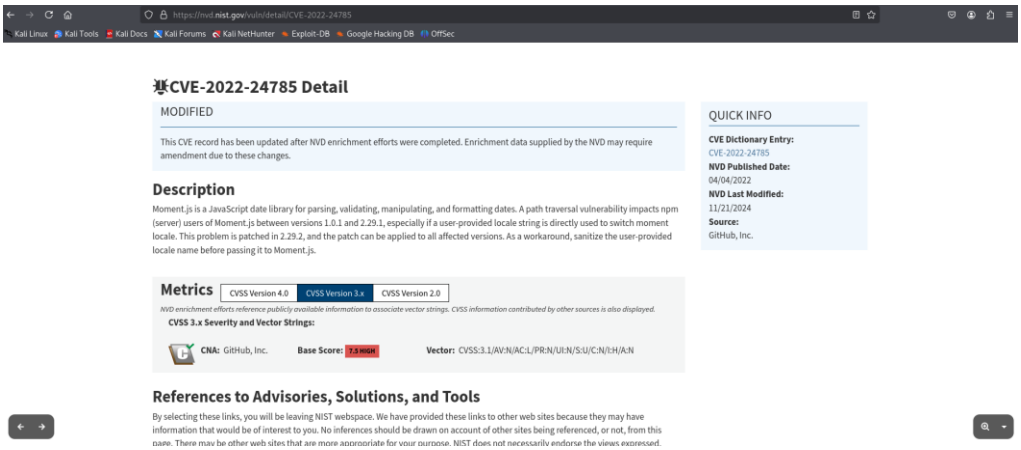


Figure 7: Vulnerable JavaScript Libraries Information

- **Broken Access Control and Broken Authentication Mechanism**

  The critical finding was **broken access control and broken authentication mechanism.** During the manual discovery of the target web application www.juggybank.com, it was found that the administration page which is admin.html could be accessed without authentication. A user can directly access the administrator page by entering the URL intro the browser or by sending a direct HTTP request with no login credentials. The server responds a status of 200 OK and shows the entire administrative interface, suggesting that no access control or session authentication is implemented on this sensitive endpoint.

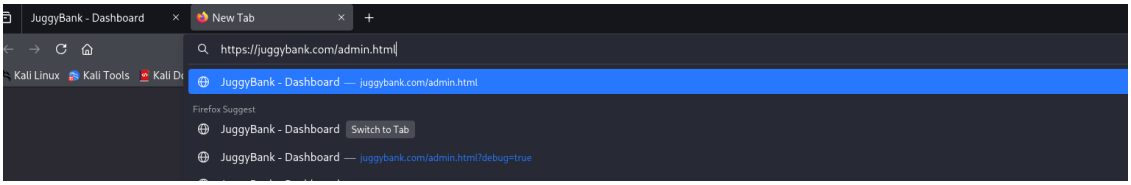


Figure 8: Direct access to admin.html page without authentication

Additionally, it was found that the authentication mechanism was implemented entirely on the client side in JavaScript, with hard-coded credentials of "juggybank" for both username and password accessible in the browser source code. This vulnerability allows an attacker to quickly bypass logins by analysing the page and exploiting the disclosed credentials. This is also known as **authentication bypass**, and it poses a significant security concern since it allows uncontrolled access to protected regions, such as admin.html, without the need for server-side authentication.

```
  <script>
function validateLogin() {
  var username = document.getElementById('username').value;
  var password = document.getElementById('password').value;

    // Simple validation for demonstration purposes
  if (username === 'juggybank' && password === 'juggybank') {
    alert('Login successful!');
    // Redirect to Google after successful login
    window.location.href = 'admin.html';
  } else {
    alert('Invalid username or password. Please try again.');
  }
}
```

Figure 9: Page Source Viewed in Login Page

Additionally, a penetration tester found that administrators could log in as users without entering a username or password, from the website https://script.viserlab.com/viserbank/ as shown in the screenshot below.

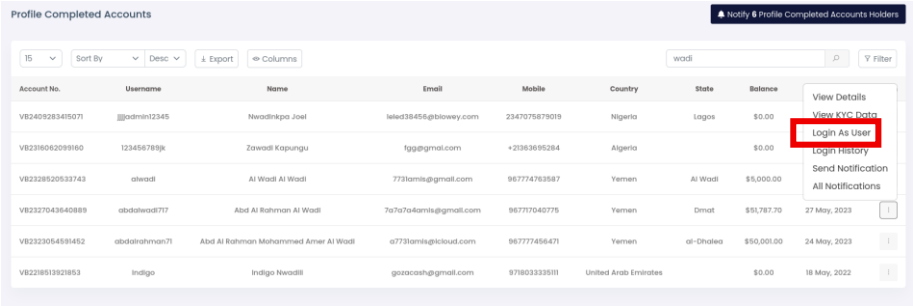


Figure 10: Administrator Dashboard

After a successful login, sensitive cardholder and personal information is displayed directly on the user's profile page. The login mechanism does not

enforce role-based constraints, which should restrict access to administrative accounts only. This vulnerability was successfully exploited by logging in using legitimate user credentials from administrator login form, hence circumventing the intended separation of user and administrator roles. This may enable unauthorized individuals to monitor or change administrative functions. The following screenshots demonstrate how an attacker could potentially log in as a user to capture credentials and compromise bank account information.

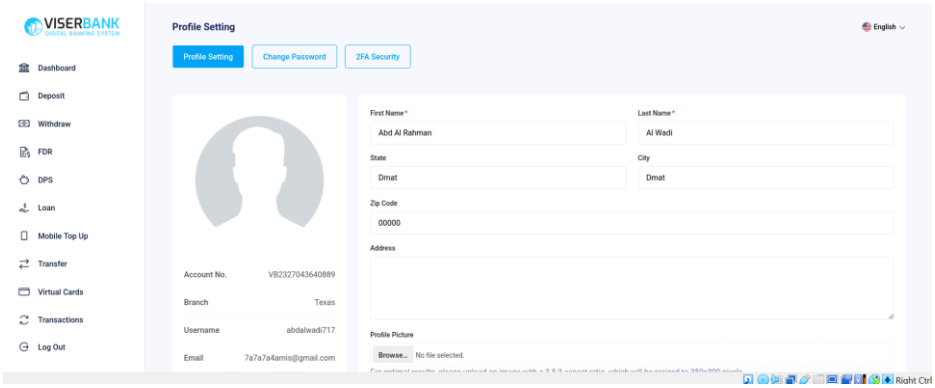


Figure 11: User Profile

- **Security Misconfigurations**

  Several security misconfigurations were found when the SSL configuration for the subdomain https://script.viserlab.com/viserbank/demo/ was manually reviewed. Important security measures were found to be disabled, even though the server uses strong signature techniques (ECDSA-P256-SHA256) and a secure cipher suite (TLS_AES_128_GCM_SHA256) and supports the latest TLS version 1.3.

  HTTP Strict Transport Security (HSTS) has been disabled, which leads to possible man-in-the-middle (MITM) attacks and SSL stripping. Furthermore, **Public Key Embedding (PKE) and Certificate Transparency** are disabled, reducing visibility and trust methods for detecting rogue certificates or illegal issuance. To ensure strong HTTPS enforcement, enable HSTS with the correct max-age directive and add subdomains. Additionally, audit TLS settings regularly and use modern security headers to protect applications from common transport layer threats.

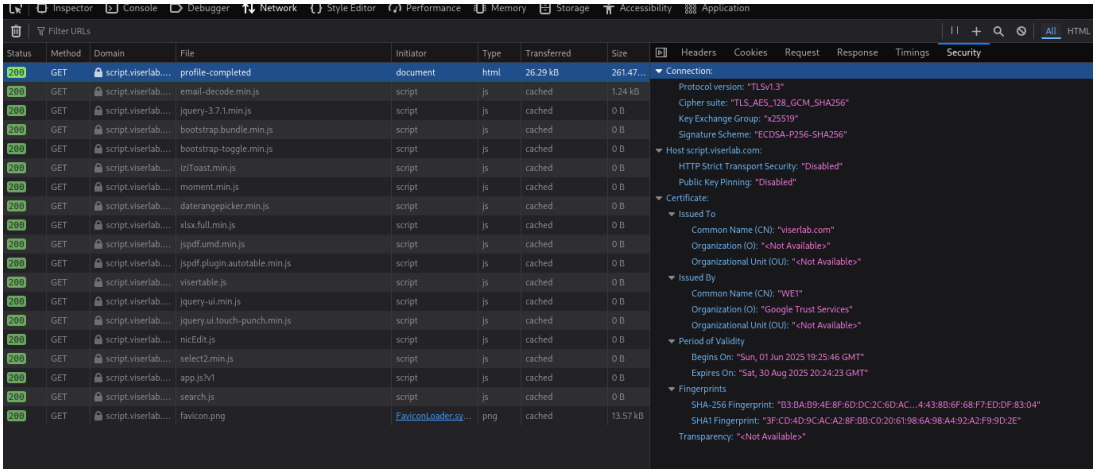


Figure 12: Security Misconfiguration Inspect in Web Developer Tools

4.0 Segmentation Test Results

To evaluate the efficiency of the segmentation restrictions for www.juggybank.com, a set of tests was conducted that simulated access attempts from a non-Cardholder Data Environment (non-CDE). A ping test to the IP address 172.67.166.161 failed, indicating that the ICMP query was stopped, possibly by the Cloudflare firewall. A thorough TCP port scan with nmap revealed that only ports 80 (HTTP), 443 (HTTPS), and 8080 (HTTP-proxy) were visible, with all other ports restricted, indicating limited exposure. A DNS query for cde.juggybank.com returns a response of NXDOMAIN, indicating that no internal or sensitive subdomains can be publicly resolved. Attempts to establish an SSH connection to the host on port 22 failed with a "connection refused" message, and visiting the administrative endpoint, potentially via curl at /admin, led to a TLS handshake failure, indicating proper access control and encryption enforcement. These findings indicate that perimeter security segmentation and controls are effective, which supports the claim of scope reduction for PCI DSS compliance.

5.0 Findings and Testing Narrative

Below are the detailed findings of vulnerabilities found in the Juggy Bank website.

The following sites were included:

- https://fonts.gstatic.com
- https://script.viserlab.com
- https://fonts.googleapis.com
- https://www.juggybank.com

Risk levels

Included: High, Medium, Low

Confidence levels

Included: User Confirmed, High, Medium, Low, False Positive

#	Vulnerability Summary	Risk level	Recommendations
1	Vulnerable JS Library	High	Upgrade to the latest version of the affected library.
2	Content Security Policy (CSP) Header Not Set	Medium	Ensure that web server, application server, load balancer is configured to set the Content-Security-Policy header.
3	Cross-Domain Misconfiguration	Medium	• Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance). • Configure the “Access-Control-Allow-Origin” HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.
4	Missing Anti-clickjacking Header	Medium	Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by site.
5	Cross-Domain JavaScript Source File Inclusion	Low	Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.
6	Strict-Transport-Security Header Not Set	Low	Ensure that web server, application server, load balancer is configured to enforce Strict-Transport-Security.

Details

1. Vulnerable JS Library	
Risk	**High**
Location	https://script.viserlab.com
Description	The identified library moment.js, version 2.24.0 is vulnerable

Observations

Retire.js is a tool that were used to check the website https://script.viserlab.com for outdated JavaScript libraries and known vulnerabilities. The scan identified various components that pose security threats as shown in the screenshot below. Most significantly, **moment.js version 2.29.0** has been identified as having high-severity vulnerabilities, including a **Regular Expression Denial of Service (ReDoS)** problem, as specified in CVE-2022-24785 and CVE-2022-31129. Other libraries found are Bootstrap 5.2.3, jQuery 3.6.0, and Select2 4.0.13-beta.1.

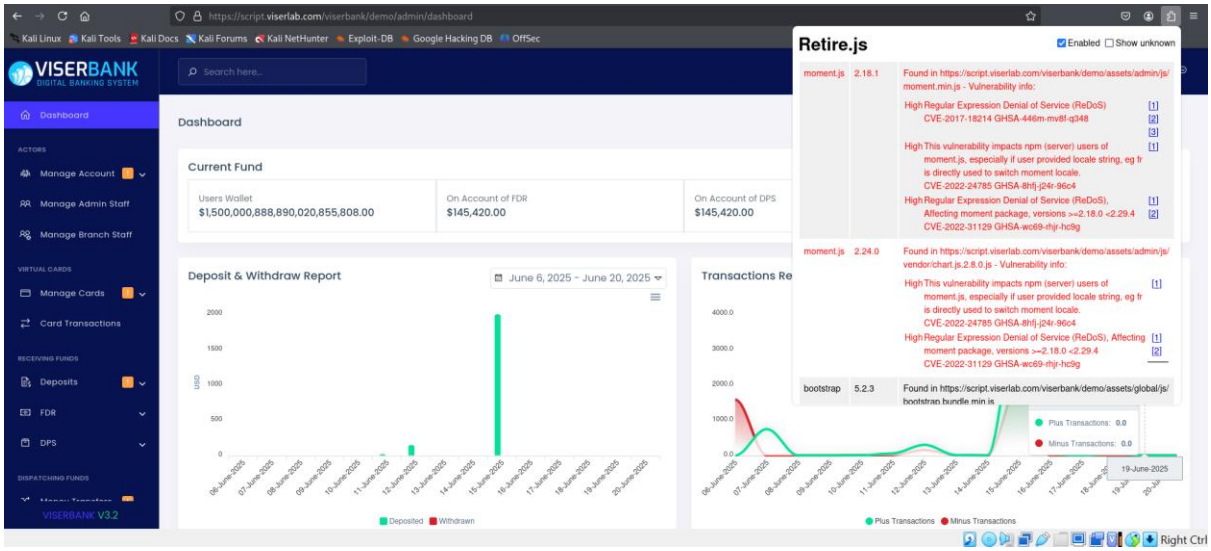


Figure 13: Observe JS Vulnerable Libraries using Retire.js

Impact: **CVSS Score 7.5**	
Confidentiality Impact	Complete
Integrity Impact	Partial
Availability Impact	Partial
Authentication Required	Not Required

Recommendations
It is advised to upgrade moment.js to the latest secure version, replace the beta version with a stable release, and maintain active dependency management procedures to ensure long-term security and stability.
Countermeasure proposal
1. Update **Moment.js v2.29.0** with **v2.29.4** or later. 2. Review and Update Other Libraries such as **Bootstrap** and **jQuery**. 3. Perform compatibility testing to ensure that the updated library does not affect any front-end functionality. 4. Deploy Dependency Monitoring Tools such as Retire.js, Snyk or OWASP Dependency-Check.
References
https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/ https://nvd.nist.gov/vuln/detail/CVE-2022-24785 https://nvd.nist.gov/vuln/detail/CVE-2022-31129

2. Content Security Policy (CSP) Header Not Set	
Risk	**Medium**
Location	https://www.juggybank.com/admin.html
Description	Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware. CSP provides a set of standard HTTP headers that allow website owners to declare approved sources of content that browsers should be allowed to load on that page covered types are JavaScript, CSS, HTML frames, fonts, images and embeddable objects such as Java applets, ActiveX, audio and video files.

Observations

During testing, a Cross-Site Scripting (XSS) vulnerability was found reflected on the www.juggybank.com website. To verify the vulnerability, HTML source in the website page was copied and tested locally by injecting a similar payload into a custom HTML file. The <script> alert ('CSP Header Not Set') </script> payload was injected into the HTML code through the URL and displayed in the created page as shown in the image below:

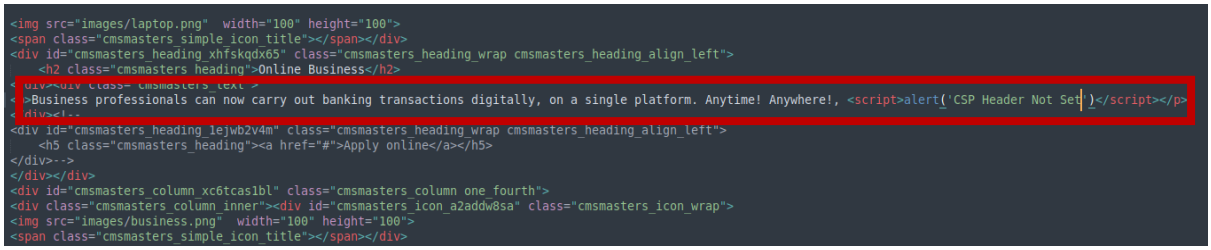


Figure 14: Injected Payload in HTML Code

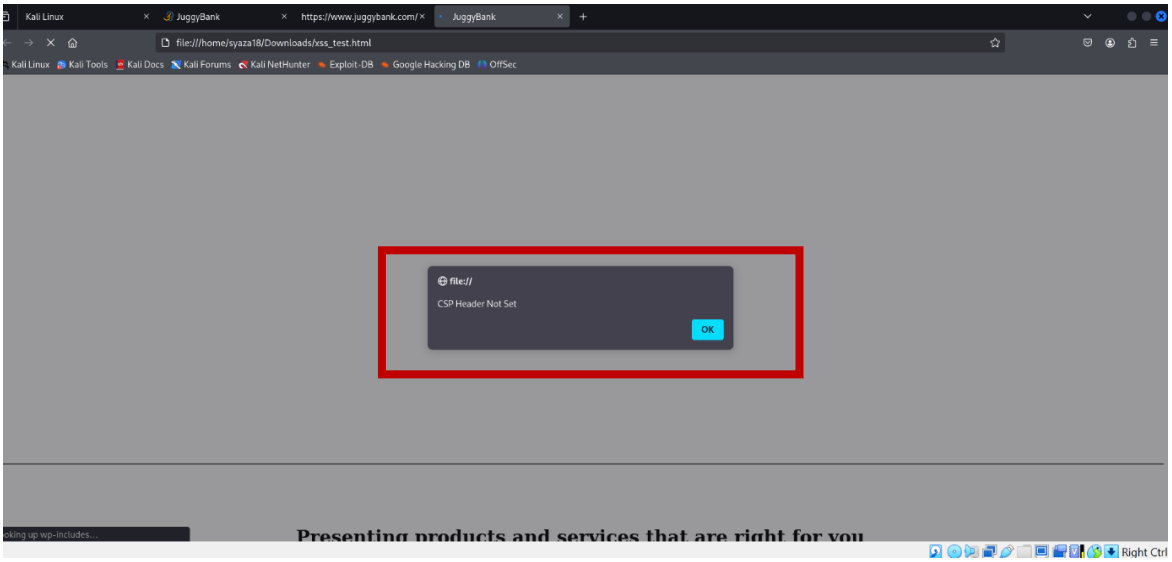


Figure 15: CSP Header Alert Appear

When viewed through a local browser, the script ran successfully, resulting in an alert pop-up. Although the local page loaded slower than the original website due to external resources, the load was still executed successfully. This demonstrates that if the input is not adequately sanitized and shown in a vulnerable situation, script injection is possible.

Furthermore, the Content Security Policy (CSP) header is not set, which increases risk because it would otherwise prevent inline or unauthorized script execution. The absence of this header amplifies the impact of client-side assaults like XSS. This vulnerability enables an attacker to inject malicious JavaScript, which might be used to steal session cookies, damage the website, or lead users to malicious domains. The exploitation requires no authentication and may be launched by just delivering a created link to the target, making the attack both effective and easy to execute.

Confidentiality Impact	Complete
Integrity Impact	Complete
Availability Impact	Partial
Authentication Required	Not Required

Recommendations
Ensure that the web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header. Sanitize and validate all user input, especially any data reflected back into HTML pages. Implement a strong Content Security Policy (CSP) to block inline script execution.

Countermeasure proposal
1. Implement strict server-side validation of the q parameter. 2. Before entering any dynamic content into an HTML page, encode it first. 3. Implement a strong Content Security Policy header such as: : Content-Security-Policy: default-src 'self'; script-src 'self'; object-src 'none'; 4. If JavaScript is used to render dynamic content, avoid using innerHTML. 5. Integrate automated security scanning tools such as OWASP ZAP and Burp Suite into the process to detect reflected XSS earlier.
References
https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html https://www.w3.org/TR/CSP/

---

3. Cross-Domain Misconfiguration	
Risk	**Medium**
Location	https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css
Description	Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server. The CORS misconfiguration on the web server permits cross-domain read requests from arbitrary third-party domains, using unauthenticated APIs on this domain. Web browser implementations do not permit arbitrary third parties to read the response from authenticated APIs, however. This reduces the risk somewhat. This misconfiguration could be used by an attacker to access data that is available in an unauthenticated manner, but which uses some other form of security, such as IP address white-listing.

Observations

A manual test was implemented to verify the vulnerability for Cross-Domain Misconfiguration in https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css. The

tester began by checking the HTTP response headers for an Access-Control-Allow-Origin: * entry, and found that the header was indeed present which means that the file can be accessed from any website.

```
┌──(syaza18㉿kali)-[~]
└─$ curl -I https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css
HTTP/2 200
date: Wed, 18 Jun 2025 17:09:28 GMT
content-type: text/css; charset=utf-8
cf-ray: 951c6504efda38ab-KUL
access-control-allow-origin: *
cache-control: public, max-age=30672000
etag: W/"5eb03e5f-7918"
last-modified: Mon, 04 May 2020 16:10:07 GMT
```

Figure 16: Checking HTTP response headers for an Access-Control-Allow-Origin

Then, the malicious HTML file was created which was used to send a cross-origin fetch request to the target resource.

```
<!DOCTYPE html>
<html>
  <body>
    <h2>Testing CORS on cdnjs</h2>
    <script>
      fetch("https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css")
        .then(response => response.text())
        .then(data => {
          alert("Got response:\n" + data.substring(0, 200) + "...");
        })
        .catch(error => {
          alert("Blocked or error:\n" + error);
        });
    </script>
  </body>
</html>
```

Figure 17: Custom HTML File to send a cross-origin fetch request

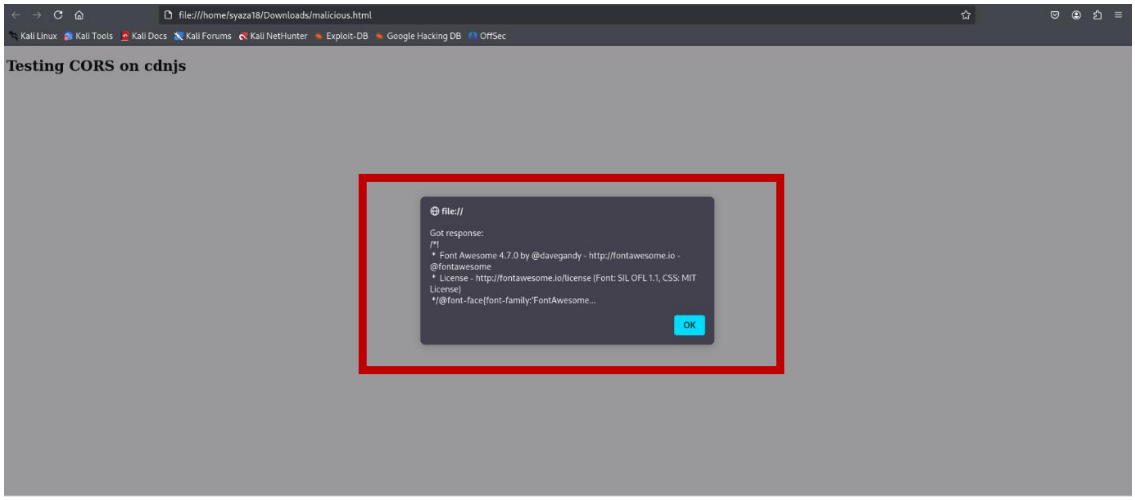


Figure 18: Successfully retrieved the response from the target

The response from the target was successfully retrieved and displayed, confirming that the file can be accessed cross-origin. However, the content retrieved was a static CSS file that did not contain critical or dynamic data, indicating that, even if CORS was set to permissive, the practical risk in this scenario was low.

Confidentiality Impact	Partial
Integrity Impact	Partial
Availability Impact	Low
Authentication Required	Not Required

Recommendations
Configure the “Access-Control-Allow-Origin” HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.
Countermeasure proposal
1. Avoid using Access-Control-Allow-Origin: * on resources that are not specifically intended for public usage and do not contain sensitive or dynamic content. 2. If cross-origin access is essential, whitelist only trusted domains with a certain origin value. 3. Avoid using Access-Control-Allow-Credentials: true with a free card origin (*), as this is both illegal and unsafe. 4. Evaluate and classify resources based on their sensitivity which apply stronger CORS rules to sensitive endpoints and relaxed restrictions just to static public assets as needed. 5. CDN providers should frequently audit their CORS headers to verify they are in line with their intended use and security standards.
References
https://vulncat.fortify.com/en/weakness

4. Missing Anti-clickjacking Header	
Risk	**Medium**
Location	https://www.juggybank.com/admin.html
Description	The response does not protect against “ClickJacking” attacks. It should include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options.

Observations

It was determined that the website https://www.juggybank.com is vulnerable to clickjacking due to the lack of appropriate frame protection headers such as X-Frame-Options or Content-Security-Policy. This was demonstrated by embedding the website in an iframe within own HTML file which is test.html, where the page was successfully displayed without any limitations or restrictions. The absence of these security headers enables the site to be shown inside an iframe from any origin, making it vulnerable to clickjacking attacks in which visitors are misled into interacting with hidden or false content.

```
<iframe src="https://www.juggybank.com/admin.html" width="1000" height="600"></iframe>
```

Figure 19: Embedded the web URL in an iframe

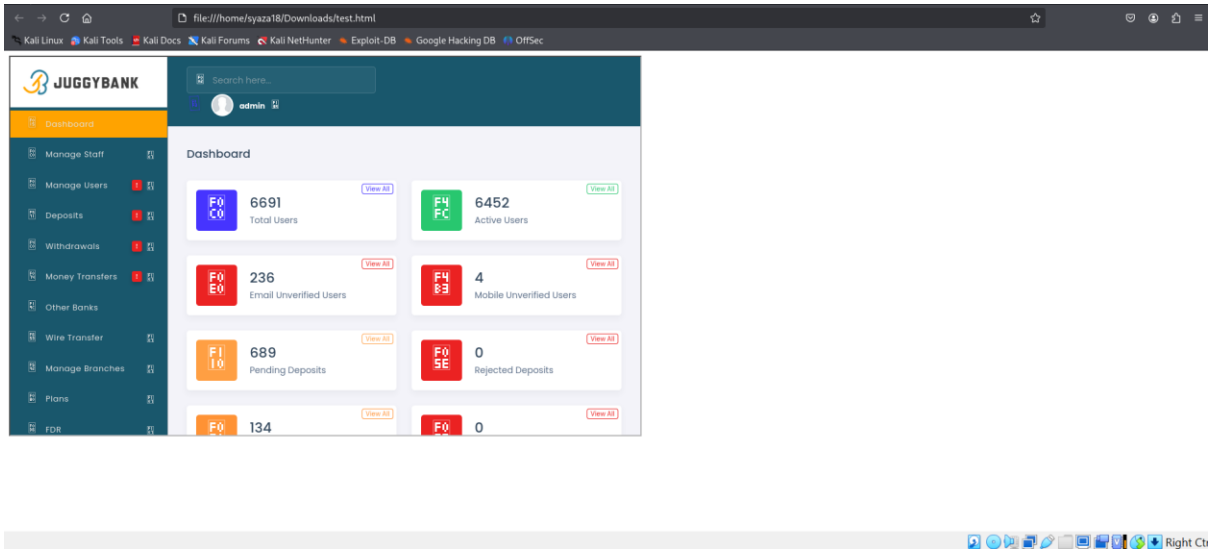


Figure 20: The web page displayed in an iframe which is vulnerable

Impact: **CVSS Score 4.3 Medium**	
Confidentiality Impact	Partial
Integrity Impact	Partial
Availability Impact	Low
Authentication Required	Not required

Recommendations
Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app. Implementing the X-Frame-Options: DENY header or utilizing Content Security Policy directives like “no ancestors” to stop websites from being embedded in external frames.
Countermeasure proposal
1. Set the X-Frame-Options HTTP Header such as X-Frame-Options: DENY 2. Implement Content Security Policy (CSP) Frame-Ancestors Directive 3. Periodically test HTTP response headers with security tools such as securityheaders.com to guarantee appropriate implementation and identify misconfigurations.
References
https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/11-Client-side_Testing/09-Testing_for_Clickjacking https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration.html https://nvd.nist.gov/vuln/detail/CVE-2025-49192 https://cwe.mitre.org/data/definitions/1021.html https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

6.0 Tools Used

A penetration tester applies a structured and methodical approach during testing, utilizing a range of tools, each with different functions in different stages of the testing lifecycle. The first step focuses on **gathering information**, which is vital for understanding the target area while discovering possible entry points. At this stage, a penetration tester uses **Nmap** which is a strong network scanning tool to find open ports, operating services, as well available vulnerabilities in network services. This helps her to plan the attack surface and achieved an understanding of the network’s basic structure.

To further list hidden directories and resources on a web server, a penetration tester used **Dirb**, **a web content scanner that brute forces URIs using a word list approach**. This was crucial in determining sensitive files, admin panels, or configuration folders that were not linked or indexed but were still publicly available. Furthermore, **DNS enumeration** was carried out using command-line tools such as **NsLookup**, to identify subdomains that likely expose new services or no longer in use assets.

After the reconnaissance phase was finished, a penetration tester continues to the **vulnerability assessment phase**. **OWASP ZAP (Zap Attack Proxy)** is the tool of choice for scanning for potential vulnerabilities in a website. It is a recognized open-source web application scanner that execute automatic scans against the target website. ZAP assisted in detecting common security concerns such as SQL injection, cross-site scripting (XSS), broken access controls, and insecure cookies. Its user-friendly interface and detailed reports were helpful in prioritizing the vulnerabilities for more manual validation.

Besides this, a penetration tester uses **Retire.js**, a tool created primarily for discovering **vulnerabilities in JavaScript frameworks**. Retire.js scans locally hosted and externally linked JavaScript files and analyse them to a database of known vulnerabilities. This allowed the identification of irrelevant or insecure JavaScript libraries that may be exploited, especially in client-side attacks.

Additionally, to help strengthen testing for vulnerability validation, custom **HTML files** were developed and used during manual testing especially for Cross-domain misconfiguration, Cross-Site Scripting (XSS) and Clickjacking Header vulnerabilities. This file creates a controlled environment in which a modified payload can be delivered to the target application, emulating attacker-controlled origins and user interaction scenarios. By using custom HTML file, the target server responds can be tested to unauthorized cross-origin requests to check if the malicious scripts will be running in the browser. In conclusion, combining automated scanning tools with manual testing methods can acquire a thorough insight of the target's security posture.

7.0 Statement of Limitations

Direct connection to the website via IP address is not available due to Cloudflare's proxy service. This limits certain network-level tests, such as determining origin servers or bypassing DNS-level security. Moreover, Penetration testing for www.juggybank.com was performed in a controlled and isolated environment using

custom HTML files rather than directly on the actual website. The test is intended to be non-intrusive, meaning that exploiting vulnerabilities is strictly controlled to avoid disrupting production services. The admin page should not be modified in any way during penetration testing. Assessment of specific vulnerabilities related to privilege escalation or administrative functions may be limited due to restrictions on testing actions such as editing, updating, or changing admin settings or content. Additionally, as a result, discovery was limited to only vulnerabilities that could be validated indirectly with the target system.

8.0 Cleaning up the Environment Post-Penetration Test

After finishing the penetration testing for https://www.juggybank.com, all test materials and adjustment that was made during test were properly delete to make sure no leftover impacts affected the target area or users.

Category	Action Taken
Test scripts and payload removal	All custom HTML pages, JavaScript payloads which are utilized for CORS, XSS and iframes used for clickjacking testing were deleted from the local or hosted environment to prevent from any accidental risk.
Sessions and network scan termination	OWASP ZAP for scanning, active sessions and open connections to the target site have been closed. All network discovery tools have been terminated, and all temporary scan data has been removed from the test machine.
Temporary test files removal	To prevent from unintended data exposure, all temporary files, logs, screenshots, or scripts that were saved locally during the testing stage were safely removed.
Review and guarantees	A final check was performed to ensure that no test components remain active in the environment and that the system is running properly without any signs of interference.

9.0 References

(Cyberkid), V. A. (2024, September 4). *Complete owasp zap guide*. Medium. http://medium.com/@redfanatic7/complete-owasp-zap-guide-384a080ff502

WSTG - v4.2. WSTG - v4.2 | OWASP Foundation. (n.d.-a). https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/