Cyberscope

Penetration Test Report

CharlieTheUnicoin

May 2024

Audited by © cyberscope

Table of Contents

Review

Domain	https://www.charlietheunicoin.com/
Initial Report	01 May 2024

Overview

Cyberscope has conducted a comprehensive penetration test on the web application "Charlie the Unicoin" hosted at https://www.charlietheunicoin.com/. This report focuses on evaluating the security and performance aspects of the web application. The assessment encompasses various facets of the application, including but not limited to authentication and authorization mechanisms, data handling and storage practices, network security measures, and response to high traffic volumes.

The expansion of blockchain technology has introduced a myriad of innovative applications, each with its own unique security challenges. Charlie the Unicoin, as a prime example within the realm of digital currency ecosystems, ensures robust protection of user data and system integrity.

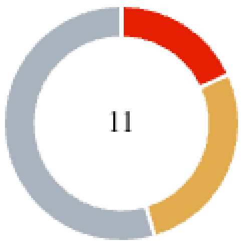
Penetration Assessment Scope

The scope of this assessment extends to identifying vulnerabilities and weaknesses in the application's architecture and functionality, with the aim of providing actionable recommendations to enhance its security posture. The report aims to offer a comprehensive understanding of the application's strengths and areas for improvement, facilitating informed decision-making to mitigate risks, fortify against potential cyber threats, and bolster overall security resilience.

Webapp Technologies

Technology	Version	Category
Squarespace	7.1	CMS
Squarespace Commerce	7.1	Ecommerce
Stimulus	N/A	JavaScript Frameworks
reCAPTCHA	N/A	Security
HSTS	N/A	Security
Google Font API	N/A	Font Scripts
Typekit	1.21.0	Font Scripts
YUI	3.17.2	JavaScript Libraries
Modernizr	2.8.3	JavaScript Libraries
Lodash	4.17.21	JavaScript Libraries
core-js	3.26.0	JavaScript Libraries
Priority Hints	N/A	Performance
Webpack	N/A	Miscellaneous
Open Graph	N/A	Miscellaneous
Module Federation	N/A	Miscellaneous

Findings Breakdown



🟥	Critical	2
🟧	Medium	3
⬜	Minor / Informative	6

Severity		Unresolved	Acknowledged	Resolved	Other
●	Critical	2	0	0	0
●	Medium	3	0	0	0
●	Minor / Informative	6	0	0	0

Diagnostics

🔴 Critical 🟠 Medium ⚪ Minor / Informative

Severity	Code	Description	Status
🔴	CEV	Credentials Exposure Vulnerability	Unresolved
🔴	SIV	SQL Injection Vulnerability	Unresolved
🟠	ATA	Anti-CSRF Tokens Absence	Unresolved
🟠	MACH	Missing Anti-Clickjacking Header	Unresolved
🟠	MCSPH	Missing Content Security Policy (CSP) Header	Unresolved
⚪	BPC	Best Practices Compliance	Unresolved
⚪	CNHF	Cookie No HttpOnly Flag	Unresolved
⚪	CWSA	Cookie Without SameSite Attribute	Unresolved
⚪	IDRC	Incomplete DNS Records Configuration	Unresolved
⚪	MSTSH	Missing Strict Transport Security Header	Unresolved
⚪	PDUL	Performance Degradation Under Load	Unresolved

CEV - Credentials Exposure Vulnerability

Criticality	Critical
Status	Unresolved

Description

There are scripts embedded in the webapp that contain sensitive credentials, such as a username and password, in plain text. These credentials are used to authorize requests to API endpoints, allowing the script to interact with external services. However, exposing credentials in this manner poses a significant security risk as they can be easily accessed and abused by malicious actors.

```
<script>
    const username = '*********';
    const password = '*********';
    ...
    function searchMemes() {
        const query = prompt('Enter meme query:');
        const body = {
            username: username,
            password: password,
            query: query,
        };

        fetchFromAPI('https://api.imgflip.com/search_memes', 'POST',
body).then((data) => {
            document.getElementById('output').innerText =
JSON.stringify(data, null, 2);
        });
    }
    ...
</script>
```

Recommendation

To mitigate the security risk associated with visible script credentials, it is essential to implement secure credential management practices. Specifically, the team should avoid hardcoding sensitive credentials, such as usernames and passwords, directly into client-side scripts. Instead, credentials should be stored securely on the server-side or utilize secure methods for accessing sensitive information, such as environment variables. Additionally, the team could implement secure authentication mechanisms, such as OAuth or token-based authentication, to authenticate requests to API endpoints without exposing sensitive credentials in client-side scripts. By implementing secure credential management practices and adhering to recommended security guidelines, the application can effectively mitigate the risk of unauthorized access and protect sensitive data from exploitation by malicious actors.

SIV - SQL Injection Vulnerability

Criticality	Critical
Status	Unresolved

Description

A SQL injection attack consists of insertion or “injection” of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data, execute administration operations on the database, recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands.

The webapp introduces potential SQL injection vulnerabilities on several endpoints. By using various techniques, including boolean conditions and query time manipulation, these manipulations resulted in unintended data retrieval or prolonged query execution times, indicating the presence of injection vulnerabilities.

The following URLs demonstrated this vulnerability:

1. https://www.charlietheunicoin.com/*&author=
2. https://www.charlietheunicoin.com/*&format=ical
3. https://www.charlietheunicoin.com/*&format=json
4. https://www.charlietheunicoin.com/*&format=page-context
5. https://www.charlietheunicoin.com/*&month=
6. https://www.charlietheunicoin.com/*&tag=
7. https://www.charlietheunicoin.com/*&view=
8. https://www.charlietheunicoin.com/*?author=+AND+1%3D1+---+
9. https://www.charlietheunicoin.com/*?format=json%27+AND+%271%27%3D%271%27+---+
10. https://www.charlietheunicoin.com/*?month=+AND+1%3D1+---+
11. https://www.charlietheunicoin.com/*?reversePaginate=+AND+1%3D1+---+
12. https://www.charlietheunicoin.com/*?tag=+AND+1%3D1+---+
13. https://www.charlietheunicoin.com/*?view=%22+AND+%221%22%3D%221

14. https://www.charlietheunicoin.com/*?author
15. https://www.charlietheunicoin.com/*?month
16. https://www.charlietheunicoin.com/*?tag
17. https://www.charlietheunicoin.com/ai-chat-bot
18. https://www.charlietheunicoin.com/mini-games
19. https://www.charlietheunicoin.com/soundboard
20. https://www.charlietheunicoin.com/white-paper-1

Recommendation

To mitigate the SQL injection vulnerabilities:

- Implement server-side validation and sanitization of all user-supplied input to prevent injection attacks.
- Utilize parameterized queries (e.g., PreparedStatement or CallableStatement in JDBC) to ensure that user input is treated as data rather than executable code.
- Refrain from dynamically constructing SQL queries using string concatenation, as this can inadvertently introduce injection vulnerabilities.
- Assign the least privileged database user role necessary for the application to limit the potential impact of successful injection attacks.
- Enforce strict input validation by implementing allowlists or denylists of permissible characters to mitigate injection risks.

By implementing these recommendations, the application can significantly reduce the risk of SQL injection attacks and enhance its overall security posture. For more detailed guidance on preventing SQL injection vulnerabilities, refer to the OWASP SQL Injection Prevention Cheat Sheet.

ATA - Anti-CSRF Tokens Absence

Criticality	Medium
Status	Unresolved

Description

The absence of Anti-CSRF (Cross-Site Request Forgery) tokens poses a significant security risk to the application. CSRF attacks involve an attacker tricking a user into performing actions on a web application without their knowledge or consent. This vulnerability arises due to the lack of protection mechanisms, such as Anti-CSRF tokens, which prevent unauthorized requests from being executed.

It was observed that no Anti-CSRF tokens were present in the HTML submission forms across various endpoints of the application. Without Anti-CSRF tokens, attackers can forge requests and manipulate user sessions to perform malicious actions, potentially leading to unauthorized data modification, account takeover, or other security breaches.

The following URLs is a sample of all the occurances that demonstrated this vulnerability:

1. https://www.charlietheunicoin.com
2. https://www.charlietheunicoin.com/ai-chat-bot
3. https://www.charlietheunicoin.com/ai-image-generation
4. https://www.charlietheunicoin.com/ai-video-generation-1
5. https://www.charlietheunicoin.com/api/
6. https://www.charlietheunicoin.com/api/ui-extensions/
7. https://www.charlietheunicoin.com/cart
8. https://www.charlietheunicoin.com/charlies-pad
9. https://www.charlietheunicoin.com/charlies-vortex
10. https://www.charlietheunicoin.com/charlieswap
11. https://www.charlietheunicoin.com/home

Recommendation

To mitigate the absence of Anti-CSRF tokens and prevent CSRF attacks, the following recommendations are provided:

- Choose a reputable library or framework that includes built-in protections against CSRF attacks or provides features to easily implement Anti-CSRF mechanisms.
- Integrate Anti-CSRF packages such as OWASP CSRFGuard, which offer robust defense mechanisms against CSRF vulnerabilities.
- Ensure that the application is free from XSS vulnerabilities, as CSRF defenses can be bypassed using XSS attacks. Implement proper input validation and output encoding to mitigate XSS risks.
- Generate unique, unpredictable nonces for each form submission and embed them within the forms. Upon form submission, validate the nonce to verify the authenticity of the request. Be cautious of predictable nonces, as they can be exploited by attackers.
- For sensitive or high-risk operations, implement confirmation mechanisms to require users to confirm their actions. This adds an extra layer of security to prevent unauthorized requests.
- Incorporate ESAPI (OWASP Enterprise Security API) Session Management control, which includes components specifically designed to mitigate CSRF attacks.
- Refrain from using the GET method for requests that trigger state changes or sensitive operations, as GET requests can be easily manipulated and abused by attackers.
- Consider checking the HTTP Referer header to verify if requests originated from expected pages. However, be aware that this approach may not be foolproof and can be circumvented by certain user agents or proxies.

By implementing these recommendations, the application can significantly reduce the risk of CSRF attacks and enhance its overall security posture. For additional information and resources on CSRF vulnerabilities and mitigation strategies, refer to the following references.

1. http://projects.webappsec.org/Cross-Site-Request-Forgery
2. https://cwe.mitre.org/data/definitions/352.html

MACH - Missing Anti-Clickjacking Header

Criticality	Medium
Status	Unresolved

Description

The absence of an Anti-Clickjacking header exposes the application to potential Clickjacking attacks. Clickjacking is a malicious technique that tricks users into clicking on unintended elements by disguising them as legitimate UI elements. This can lead to unauthorized actions being performed without the user's knowledge or consent. Without proper protection mechanisms in place, attackers can exploit Clickjacking vulnerabilities to perform actions on behalf of users, such as making purchases, changing account settings, or clicking on malicious links.

The response from the https://www.charlietheunicoin.com/cart URL does not include either Content-Security-Policy with 'frame-ancestors' directive or X-Frame-Options to protect against 'ClickJacking' attacks.

Recommendation

Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. The team is advised to ensure one of them is set on all web pages returned by the site/app.

If the team expects the page to be framed only by pages on their server (e.g. it's part of a FRAMESET) then they'll want to use SAMEORIGIN, otherwise if the team never expects the page to be framed, they should use DENY. Alternatively, the team could consider implementing Content Security Policy's "frame-ancestors" directive.

Reference: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

MCSPH - Missing Content Security Policy (CSP) Header

Criticality	Medium
Status	Unresolved

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.

The following URLs are a sample of all the occurrences were a Content Security Policy (CSP) header was not set.

1. https://www.charlietheunicoin.com/nft-collection
2. https://www.charlietheunicoin.com/nft-marketplace
3. https://www.charlietheunicoin.com/soundboard
4. https://www.charlietheunicoin.com/staking
5. https://www.charlietheunicoin.com/standard-meme-generator
6. https://www.charlietheunicoin.com/static/
7. https://www.charlietheunicoin.com/white-paper
8. https://www.charlietheunicoin.com/white-paper-1
9. https://www.charlietheunicoin.com/
10. https://www.charlietheunicoin.com/*&format=page-context
11. https://www.charlietheunicoin.com/*?format=ical
12. https://www.charlietheunicoin.com/*?format=json
13. https://www.charlietheunicoin.com/*?month
14. https://www.charlietheunicoin.com/*?view
15. https://www.charlietheunicoin.com/nft-marketplace
16. https://www.charlietheunicoin.com/soundboard

Recommendation

To address the absence of Content Security Policy (CSP) headers and enhance the security of the application, the following steps are recommended:

- Verify that your web server, application server, load balancer, or any other relevant components are properly configured to set the Content-Security-Policy header in HTTP responses.
- Define a comprehensive CSP policy tailored to the specific requirements and functionalities of your application. Consider including directives such as default-src, script-src, style-src, img-src, font-src, connect-src, frame-src, media-src, object-src, and sandbox, among others, to restrict content loading from unauthorized sources.
- Utilize CSP reporting mechanisms to monitor policy violations and fine-tune your CSP directives over time based on real-world usage and detected issues.

By implementing a robust Content Security Policy (CSP) and adhering to best practices for CSP configuration and management, you can significantly reduce the risk of XSS attacks, data injection vulnerabilities, and other web security threats, thereby enhancing the overall security posture of your application.

References:

1. Mozilla Developer Network: Introducing Content Security Policy
2. OWASP Content Security Policy Cheat Sheet
3. W3C Content Security Policy Specification

BPC - Best Practices Compliance

Criticality	Minor / Informative
Status	Unresolved

Description

Several issues spanning performance, security, and best practices were identified as part of the assessment. Performance metrics including First Contentful Paint, Largest Contentful Paint, Speed Index, and Total Blocking Time indicate subpar performance levels, which could significantly impact user experience and engagement. Moreover, security vulnerabilities were uncovered, particularly concerning the use of deprecated APIs, which expose the application to potential attacks like man-in-the-middle and data interception. Additionally, best practices violations, such as missing meta descriptions and improper meta tag usage, were noted, adversely affecting the application's SEO and overall accessibility. These findings underscore the importance of addressing these issues promptly to ensure the application's usability, security, and compliance with industry standards.

Recommendation

The team is advised to address the identified issues and improve the overall quality of the application. Specifically, the team could ensure compliance with web development best practices by addressing the aforementioned issues. By addressing the identified issues, the application can improve its performance, security posture, and compliance with industry standards, ultimately enhancing user satisfaction and engagement.

CNHF - Cookie No HttpOnly Flag

Criticality	Minor / Informative
Status	Unresolved

Description

The presence of a cookie (crumb) without the HttpOnly flag poses a security risk to the application. When a cookie is set without the HttpOnly flag, it means that the cookie can be accessed by client-side JavaScript code. This opens up the possibility of malicious scripts accessing sensitive cookie information and transmitting it to unauthorized third-party sites. If the affected cookie is a session cookie, it could potentially lead to session hijacking, compromising user accounts, and exposing sensitive data.

The following URLs demonstrated this vulnerability:

1. https://www.charlietheunicoin.com
2. https://www.charlietheunicoin.com/robots.txt
3. https://www.charlietheunicoin.com/sitemap.xml

Recommendation

To mitigate the risk associated with cookies lacking the HttpOnly flag, it is essential to ensure that all cookies are configured with the HttpOnly flag enabled. Here are the recommended steps:

- Review the application's cookie settings and identify any cookies that do not have the HttpOnly flag set.
- Modify the cookie configuration to include the HttpOnly flag for all cookies, particularly those containing sensitive information or used for session management.

By ensuring that all cookies are configured with the HttpOnly flag enabled, the application can effectively mitigate the risk of client-side script attacks, safeguard sensitive cookie data, and enhance overall security.

Reference: OWASP: HttpOnly

CWSA - Cookie Without SameSite Attribute

Criticality	Minor / Informative
Status	Unresolved

Description

The presence of a cookie (crumb) without the SameSite attribute poses a security risk to the application. When a cookie is set without the SameSite attribute, it can be sent as part of cross-site requests, potentially exposing users to various web security threats, including cross-site request forgery (CSRF), cross-site scripting (XSS) inclusion, and timing attacks. The SameSite attribute provides an effective countermeasure against these types of attacks by restricting the cookie's behavior based on the origin of the request.

The following URLs demonstrated this vulnerability:

1. https://www.charlietheunicoin.com
2. https://www.charlietheunicoin.com/robots.txt
3. https://www.charlietheunicoin.com/sitemap.xml

Recommendation

To mitigate the risk associated with cookies lacking the SameSite attribute, it is essential to ensure that all cookies are configured with the SameSite attribute set to either Lax or ideally Strict. Here are the recommended steps:

- Review the application's cookie settings and identify any cookies that do not have the SameSite attribute set.
- Modify the cookie configuration to include the SameSite attribute for all cookies, particularly those containing sensitive information or used for session management.

By ensuring that all cookies are configured with the appropriate SameSite attribute (preferably Strict or Lax), the application can effectively mitigate the risk of cross-origin attacks and enhance overall security.

Reference: IETF Draft: SameSite Cookies

IDRC - Incomplete DNS Records Configuration

Criticality	Minor / Informative
Status	Unresolved

Description

The app's domain lacks or has misconfigured crucial DNS records, which could potentially lead to email security vulnerabilities, such as email spoofing and phishing attacks. Specifically, the domain lacks the following essential records:

- **SPF (Sender Policy Framework):** This record helps prevent email spoofing and protects against phishing attacks by specifying authorized email servers for sending emails on behalf of the domain.
- **DMARC (Domain-based Message Authentication, Reporting, and Conformance):** DMARC provides email authentication and policy enforcement to prevent email spoofing and domain impersonation.
- **DKIM (DomainKeys Identified Mail):** DKIM adds a digital signature to outgoing emails, allowing recipients to verify the authenticity of the sender's domain and ensuring email integrity.

Recommendation

To address the incomplete DNS records configuration and improve domain security, the team is advised to configure SPF, DMARC, and DKIM records to enhance email security and prevent email spoofing and phishing attacks. The team should ensure that these records are properly configured according to best practices and align with the domain's email-sending policies. By implementing these recommendations, the domain can enhance its DNS security posture, mitigate email spoofing and phishing risks, and maintain trust and reliability with users and email recipients.

MSTSH - Missing Strict Transport Security Header

Criticality	Minor / Informative
Status	Unresolved

Description

The absence of the HTTP Strict Transport Security (HSTS) header poses a security risk to the application. HSTS is a crucial web security mechanism that instructs compliant web browsers to interact with the server using only secure HTTPS connections, thereby enhancing the overall security of communication between the client and the server. By enforcing HTTPS usage, HSTS helps mitigate various security threats, including man-in-the-middle attacks, network eavesdropping, and protocol downgrade attacks.

The following URL was missing a HSTS header: https://www.charlietheunicoin.com.

Recommendation

To enhance the security of the application and enforce secure communication over HTTPS, it is essential to ensure that the web server, application server, load balancer, or any other relevant components are configured to enforce Strict Transport Security (HSTS). By configuring the application to enforce Strict Transport Security (HSTS) and following best practices for HSTS implementation, the application can significantly reduce the risk of network-based attacks, protect sensitive data in transit, and enhance overall security posture.

References:

1. https://cheatsheetseries.owasp.org/cheatsheets/HTTP_Strict_Transport_Security_Cheat_Sheet.html
2. https://owasp.org/www-community/Security_Headers
3. http://en.wikipedia.org/wiki/HTTP_Strict_Transport_Security

PDUL - Performance Degradation Under Load

Criticality	Minor / Informative
Status	Unresolved

Description

The web app demonstrates performance degradation issues under load. The rate-limit test involved executing 11,690 requests within a span of 31.34 seconds, resulting in an average throughput of 50,176 requests per second. However, a significant number of errors were encountered, with 8,683 requests resulting in errors primarily due to timeouts. Additionally, no successful (2xx) responses were recorded during the assessment, indicating a high failure rate.

Recommendation

To address the performance issues identified during the rate limit test, the following recommendations are suggested:

1. Conduct a thorough performance analysis of the application to identify bottlenecks and optimize critical components such as database queries, server-side processing, and resource-intensive operations to improve overall system performance.
2. Perform comprehensive load testing using realistic scenarios to simulate production-level traffic and identify performance limitations. Adjust the concurrency levels, request distribution, and sampling intervals to accurately reflect real-world usage patterns.
3. Implement robust error-handling mechanisms to gracefully manage timeouts and prevent service disruptions. Configure appropriate timeout thresholds and implement retry strategies to handle transient errors and mitigate the impact of timeouts.
4. Evaluate the scalability of the infrastructure hosting the application and consider scaling horizontally or vertically to accommodate increased traffic and improve response times under load. Utilize cloud-based services and auto-scaling capabilities to dynamically adjust resource allocation based on demand.

By implementing these recommendations, the team can address the performance degradation observed during the rate limit test and ensure optimal performance under varying load conditions, thereby enhancing user experience and reliability of the application.

Summary

This report provides a thorough assessment of the web application's security and performance. Through meticulous analysis, the report identifies vulnerabilities and weaknesses in key areas such as data handling and network security. Recommendations are provided to address these issues and enhance the application's resilience against cyber threats.

Overall, the report serves as a valuable resource, offering insights into the application's security posture and actionable recommendations to fortify its defenses. By implementing the suggested measures, the team can strengthen the app’s security foundation and maintain trust among users.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an “endorsement” or “disapproval” of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io