ZEALYNX SECURITY

Penetration Testing

All Your Base

Security Assessment

July 12th, 2024 - Prepared by Zealynx Security

Jose Fernando @0xMrjory

Disclaimer

The information in this document is confidential and meant for use only by the intended recipient. Every effort has been made to ensure that the information contained in this document is true and correct at the time of publication. However, the products, specifications, and content in general that are described in this document are subject to continuous development and improvement, and therefore the reporter cannot accept liability for any loss or damage of any nature whatsoever arising or resulting from the use of or reliance on outdated information or particulars.

Changelog

Revision	Date	Change
1	June 10, 2024	Initial revision

Contents

Disclaimer 1

Executive Summary 3

Summary 3

Methodology 3

Findings Overview 4

Key Findings 4

General Recommendations 4

Risk Score 5

Assessment Objectives 5

Assessment Scope 5

Assessment Approach 5

Findings Summary 6

Application summary 7

Technical Details 7

Decimals not handled by the application frontend 7

Content spoofing via parameters `url`, `w` and `q` (unauthenticated and authenticated [wallet connected]) 12

Improper error handling in parameters `url`, `w` and `q` (unauthenticated and authenticated [wallet connected]) 17

Strict transport security not enforced 20

Appendix A 22

Executive Summary

Summary

The reporter performed a security assessment of the allyourbase site between June 10, 2024 and June 17, 2024. The purpose of the assessment was to identify security vulnerabilities and recommend remediations.

The assessment was performed with a black-box, dynamic (browser based) approach.

The reporter does not warrant that the material contained in this documentation is free of errors, please note that it is not possible to find all vulnerabilities and vectors during an assessment. This report should be taken as-is and not as an exhaustive list of all security issues. With the ever-changing environment of information technology, tests performed will exclude vulnerabilities in software or systems that are unknown at the time of the assessment.

Methodology

The assessment is conducted with the following phases:

- Pre-engagement Interactions
- Enumeration
- Vulnerability Discovery
- Exploitation
- Post Exploitation
- Reporting
- Post-Engagement interaction

The reporter uses a combination of automated and manual methods and follows a testing methodology based on the PTES Technical Guidelines and OWASP Testing Guide.

Findings Overview

Vulnerability Classification (see Appendix A)	No. of active vulnerabilities
Critical	0
High	1
Medium	2
Low	0
Informational	1

Key Findings

The penetration test identified **7 VULNERABILITIES** that require remediation:

1. Content Spoofing
3. Strict Transport not enforced
4. Improper error handling
5. Rounding issues

General Recommendations

To increase the security posture, the reporter recommends the following actions be taken:

1. Develop a plan of action and mitigation to remediate all other vulnerabilities according to a specific process of software patching. For more info: https://owaspsamm.org/model/operations/environment-management/stream-b/
2. Perform routine testing for the applications on a semi-annual basis.

Risk Score

The risk score for Bitsight response scan is 7 of a possible 25, which is rated at **LOW RISK**.

A LOW risk score indicates the target system or data is at a very low risk of being compromised and no immediate action is required. (See Risk score calculation at the end of the report)

Assessment Objectives

The security assessment attempted to gain information in three areas:

1. Identify security risks and gain system level access.
2. Identify areas of infrastructure weakness.
3. Recommend remediations to mitigate risks and eliminate vulnerabilities.

Assessment Scope

The assessment was performed on allyourbase..virtual.tech.

Assessment Approach

The assessment was conducted in five phases:

1. Reconnaissance and information gathering.
2. Review reconnaissance data and perform analysis.
3. Using Tools like proxies and interceptors to test injections and other issues.
4. Assess systems and determine which may be vulnerable to exploitation.
5. Documentation of findings and recommendations.

Findings Summary

The tables below summarize vulnerabilities discovered during the assessment. More information about vulnerability classification can be found in PortSwigger.

	Severity	Remediated	Finding
	High	No	Rounding issues
	Medium	No	Content Spoofing
	Medium	No	Improper error handling
	Informational	No	Strict transport not enforced

Application summary

The allyourbase (allyourbase.virtual.tech) website is a Web2 application powered by the NestJS framework, which interfaces with a smart contract on the backend. This platform facilitates a game where players can choose from various tokens to participate. Players select a token and place bets on whether the USD price will increase (up) or decrease (down). Currently, betting is limited to USDT, although additional betting pools are planned for the future.

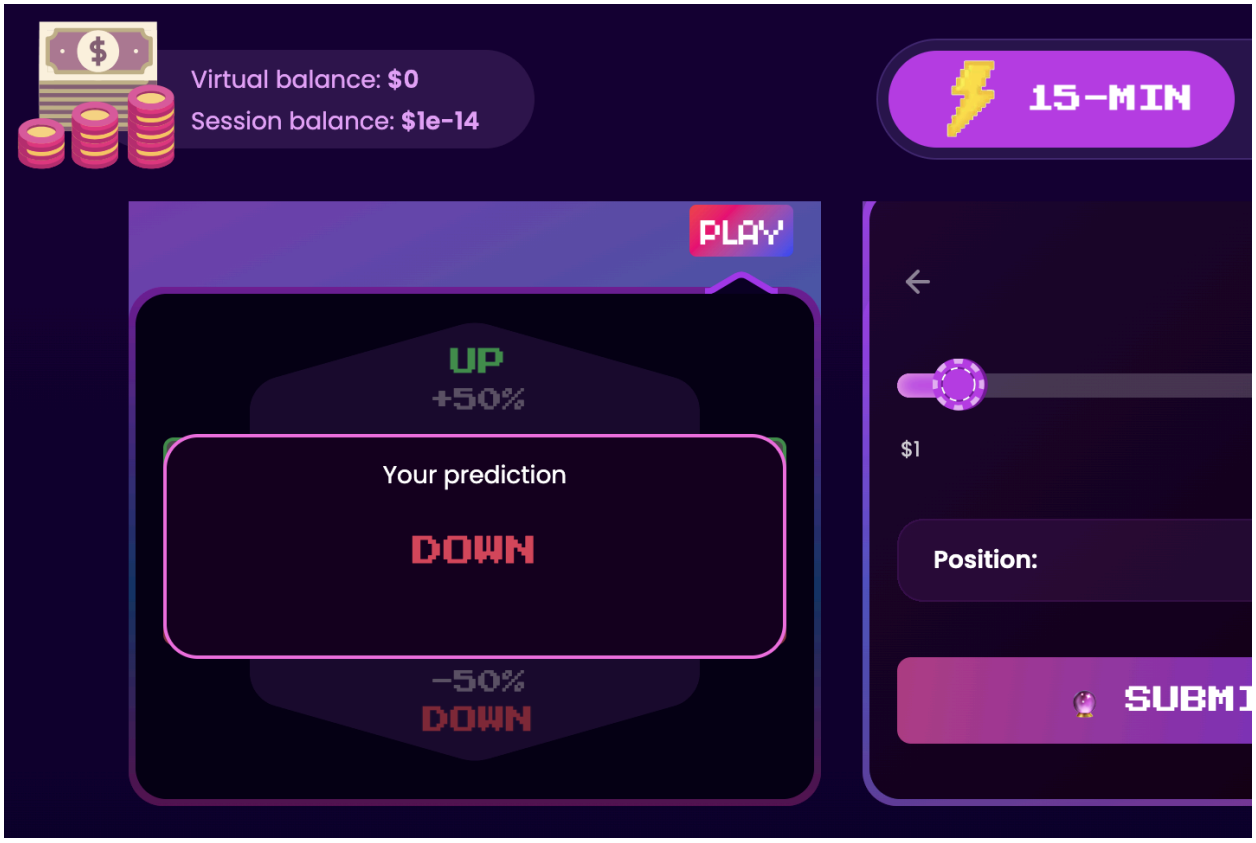
After placing their bets, users await the application's response to determine the outcome (win or lose). Depending on the result, their balance is adjusted accordingly, either increasing or decreasing based on the game's outcome.

Technical Details

Decimals not handled by the application frontend

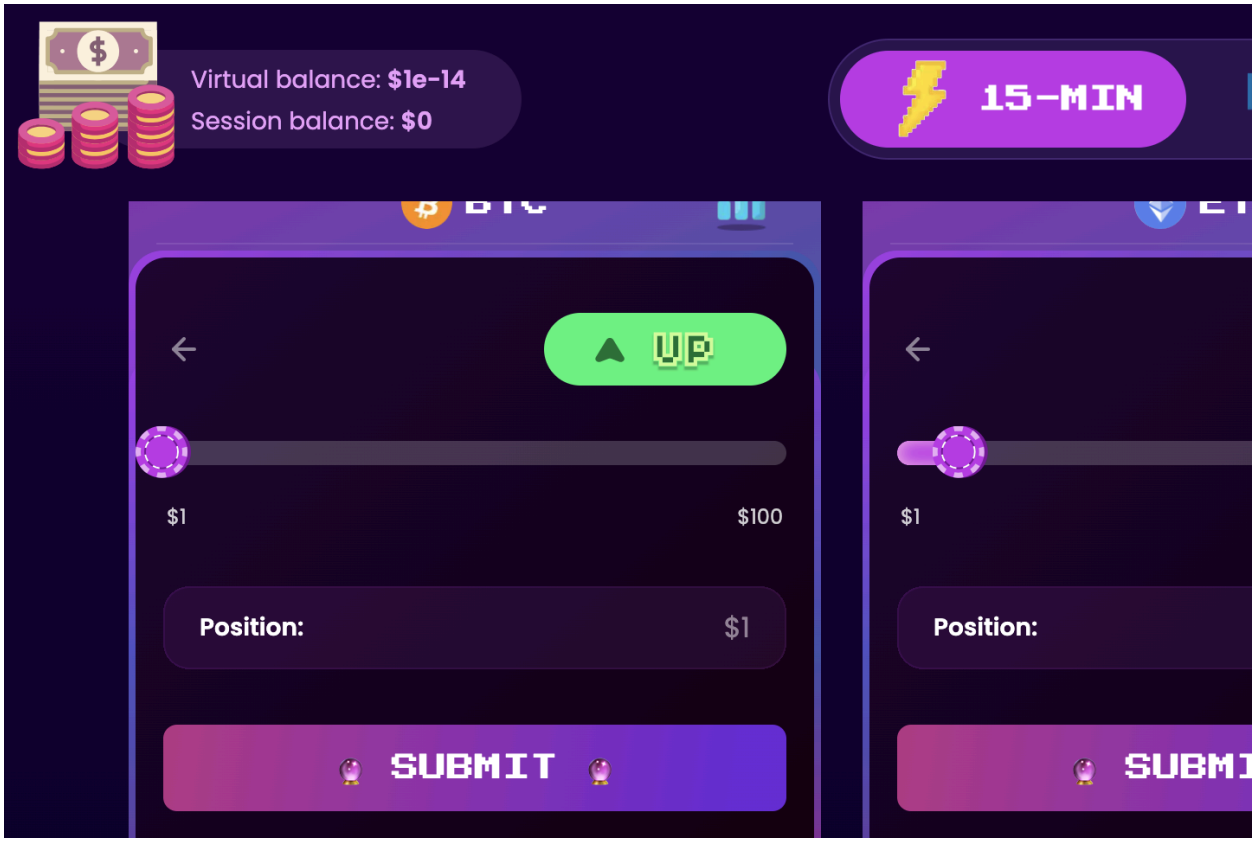
Summary

In allyourbase.virtual.tech a user can manually enter **1.(decimal)** numbers in the deposit amount. Consequently the frontend will display an exponential amount (see image) and the session balance will be displayed using this convention.



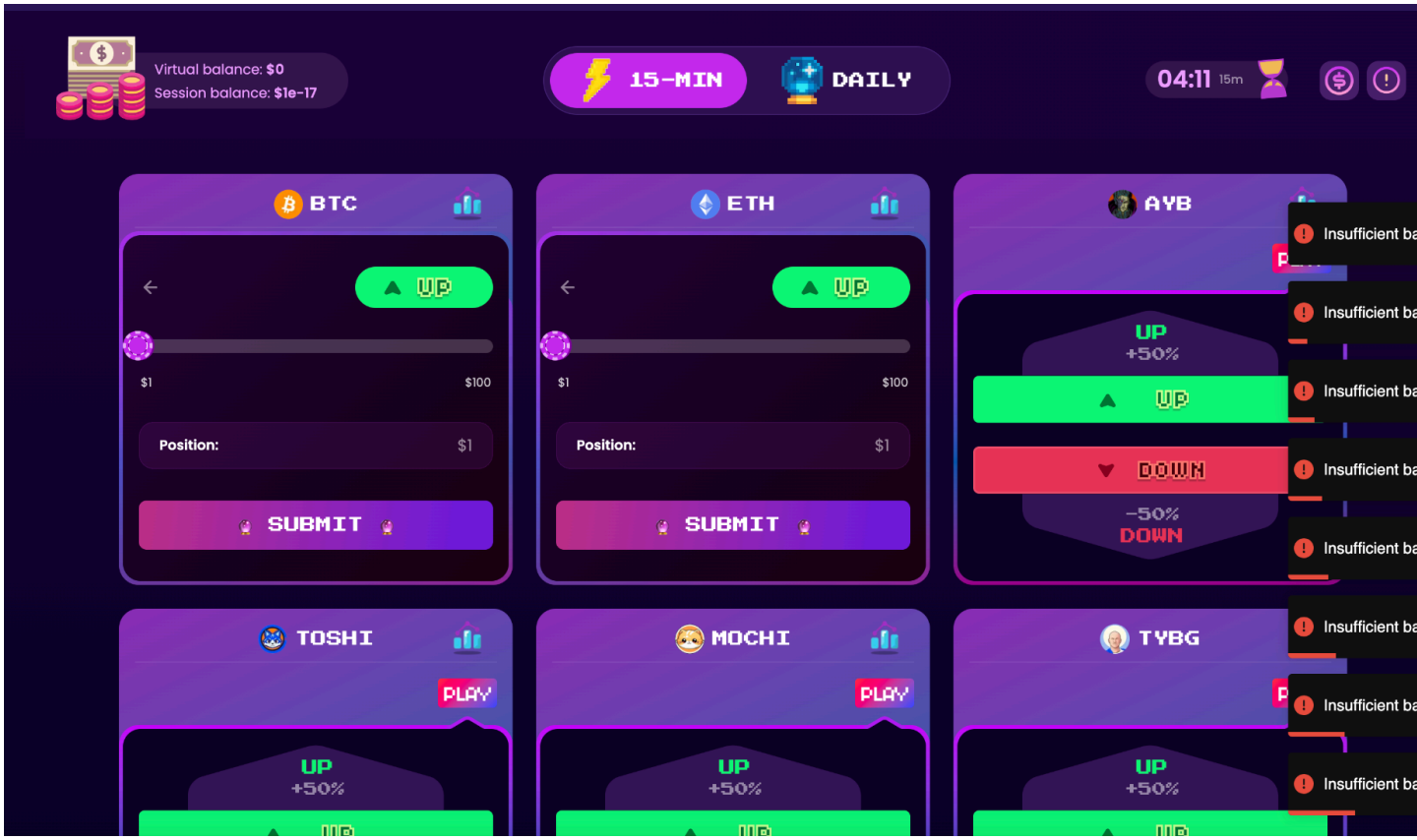
In the image: A user introduced a decimal value **1.0…0001 (14 decimals)** in the session balance. When submitting the app processes that amount as an exponential. If the user guesses an incorrect prediction the “Session balance” is still displaying the exponential value without any change.

Since the application relies on blockchain technology, these rounding errors could lead to issues in the calculations or even just gas griefing, as the backend might not be able to handle decimal values properly.



In the image: After finishing the session, the virtual balance is updated with exponentials.

In the image: Another example with 17 decimals.



In the image: The frontend is displaying at least 1 in session balance and the bet is for exactly $1. Hence, when hitting submit the app displays insufficient balance. The introduction of decimals are not handled by the app.

This particular finding is classified as high since it is related to gaming finance (at least from the frontend perspective), and the app is displaying amounts in decimals (something that might not be expected by the backend).

Classification

High: CVSS:3.1/AV:N/AC:L/PR:L/UI:N/S:U/C:N/I:H/A:L

Affected Hosts

allyourbase.virtual.tech

Recommendation

Revise the permitted input for the deposit amount. Although users are expected to enter whole numbers, the application currently accepts decimals. This could lead to

rounding errors, as the application does not notify users that decimals might not be supported.

Content spoofing via parameters `url`, `w` and `q` (unauthenticated and authenticated [wallet connected])

Summary

Content spoofing, also known as content injection, arbitrary text injection, or virtual defacement, is an attack targeting a user through a vulnerability in a web application. This occurs when the application improperly handles user-supplied data, allowing an attacker to inject content, typically via a parameter value, which is then reflected back to the user. This results in the user seeing a modified page under the trusted domain's context. Often, this type of attack is combined with social engineering tactics, exploiting both a code-based vulnerability and the user's trust.

The impact of a content spoofing attack varies based on context. If user-supplied information is reflected in a way that is correctly escaped and clearly visually marked, such as in error messages, it may be harmless. However, if the input is not clearly visually distinguished from the legitimate content, it can be used in social engineering attacks. Moreover, if the input is not correctly escaped, it may contain active components, enabling attacks similar to Cross-site Scripting (XSS).

In the specific case of allyourbase.virtual.tech, parameters **url**, **w** and **q** can be modified to reflect the frontend changes/defacements directly in the response. The parameters can be used initially to self-deface the website as the application allows the user to change the values on the fly while the components are loading.

Classification

Medium : CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:L/A:N

Affected Hosts

allyourbase.virtual.tech

Proof of Concept

Using a proxy intercept the request when browsing to allyourbase.virtual.tech. The following GET request will be available:

GET /_next/image?url=%2Ficons%2Fclassic.png&w=48&q=75 HTTP/2
Host: allyourbase.virtual.tech
// Other headers not shown.

From the request above, you can see that parameter url, w and q are having predictable values that can be modified in the request and overalls are parsed by the frontend. First of all if we modify the parameter url with different values, the following error messages appear:

Request:

GET /_next/image?url=%2F../../../&w=828&q=011-10 HTTP/2
Host: allyourbase.virtual.tech

Response:

HTTP/2 400 Bad Request
Unable to optimize image and unable to fallback to upstream image

Request:
GET /_next/image?url=%2Ffoo.png&w=828&q=011-10 HTTP/2
Host: allyourbase.virtual.tech

Response:

HTTP/2 400 Bad Request
The requested resource isn't a valid image.

For parameters q and w:

Request:
GET /_next/image?url=%2Ficons%2Fclassic.png&w=48&q=7500 HTTP/2
Host: allyourbase.virtual.tech

Response:

"q" parameter (quality) must be a number between 1 and 100

Request:
GET /_next/image?url=%2Ficons%2Fclassic.png&w=4800&q=75 HTTP/2
Host: allyourbase.virtual.tech

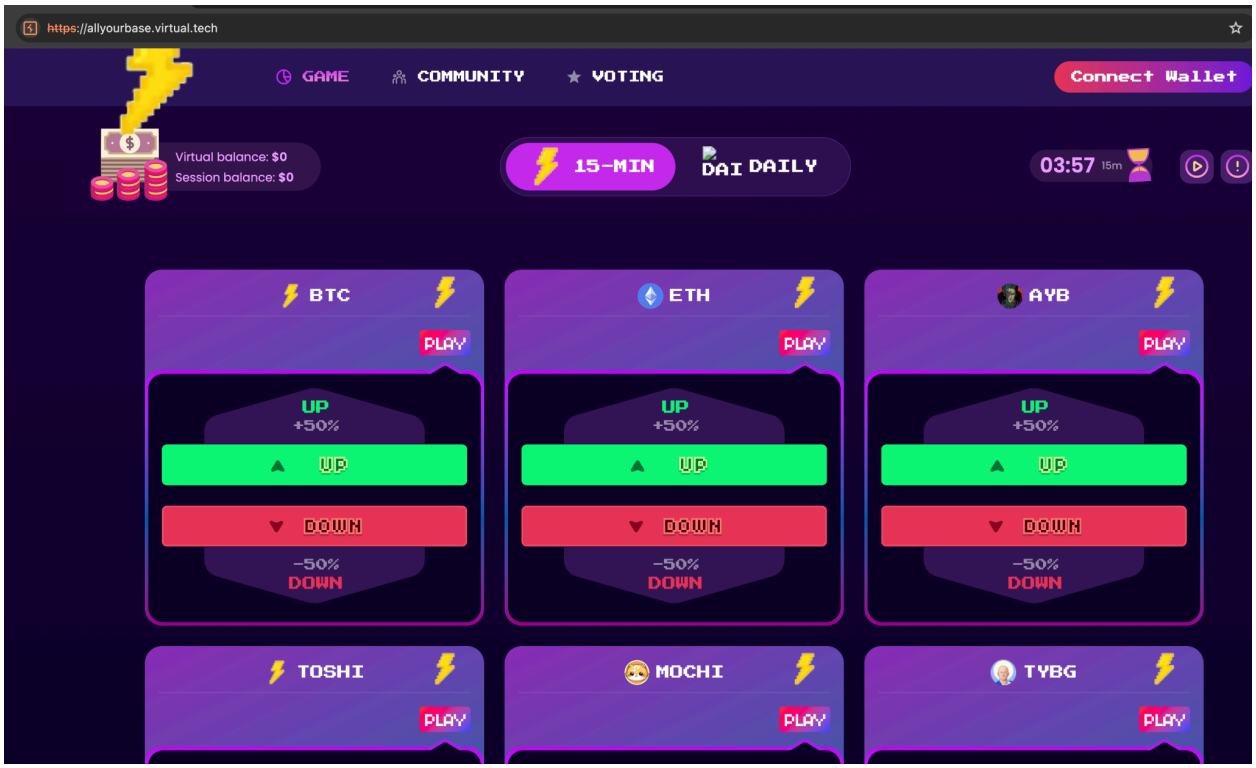
Response:

"w" parameter (width) of 4800 is not allowed

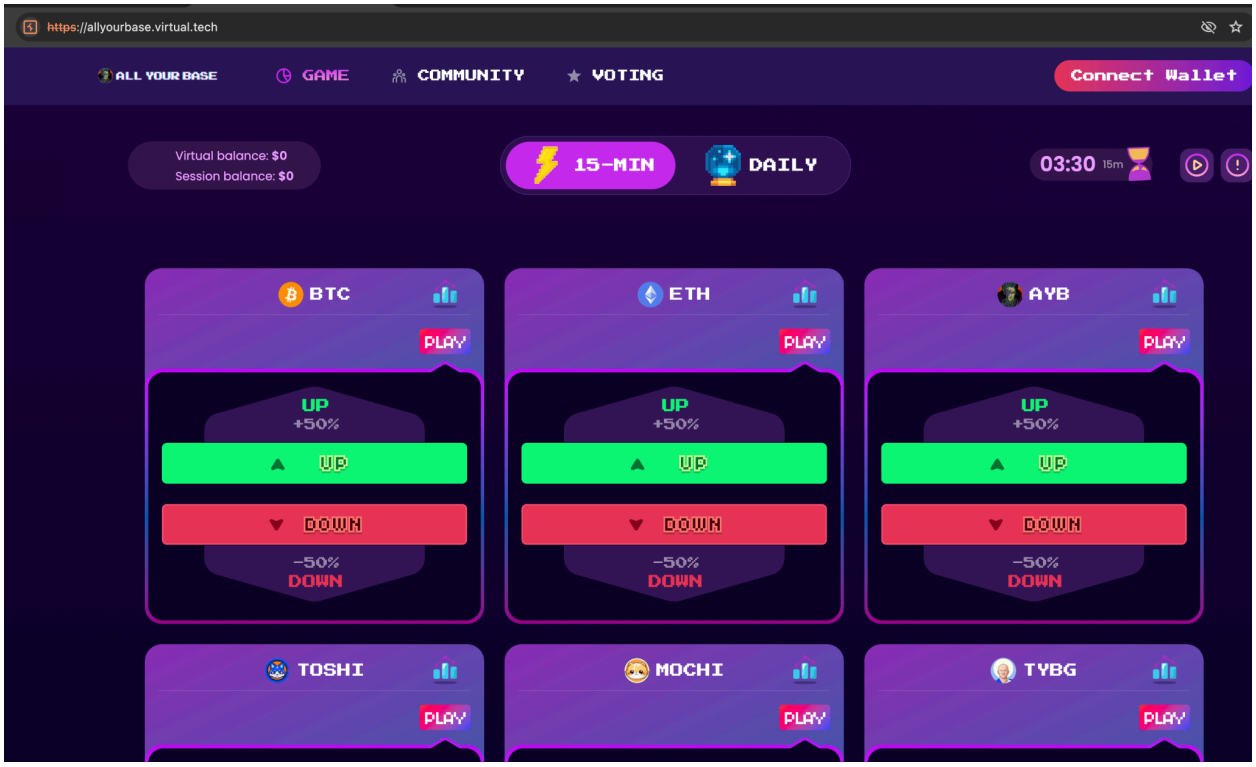
It is clear that application response changes according to the url, q and w values which are allowed to be modified on the fly. In this way, when content is loading, it is possible to execute a defacement by changing the values of the images loading:

We modify the tails image for heads on the fly (which will be processed by the application) in this example:

Response (rendered):



Original:



Recommendation

Filter all supplied content to the parameters `url`, `w`, and `q`, especially since these are programmatically defined. Although this may seem like self-defacement, users should not be able to modify the contents of a defined web application, even within their own session.

References

- Content Spoofing

Improper error handling in parameters `url`, `w` and `q` (unauthenticated and authenticated [wallet connected])

Summary

Improper error handling can introduce numerous security issues for a website. The most common issue arises when detailed internal error messages, such as stack traces, database dumps, and error codes, are displayed to users (potential attackers). These messages reveal implementation details that should remain confidential, providing malicious users with important clues about potential vulnerabilities in the site. Additionally, such messages can be disturbing for regular users.

Web applications often generate error conditions during normal operation, such as out-of-memory errors, null pointer exceptions, system call failures, database unavailability, and network timeouts. These errors must be managed according to a well-designed scheme that provides meaningful error messages to users, diagnostic information to site maintainers, and no useful information to attackers.

Even when error messages lack detail, inconsistencies in these messages can still reveal critical information about a site's inner workings and the data it contains. For example, an error message stating "file not found" when a user attempts to access a non-existent file, and "access denied" when trying to access a restricted file, can inadvertently disclose the existence of hidden files or the site's directory structure. This inconsistency allows users to infer the presence or absence of files they should not be aware of.

In the specific case of allyourbase.virtual.tech, the application does not gracefully handle the errors displayed when the user supplies values for the given parameters, as described in the PoC below.

Classification

Medium: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N

Affected Hosts

allyourbase.virtual.tech

Proof of Concept

Using a proxy intercept the request when browsing to allyourbase.virtual.tech. The following GET request will be available:

GET /_next/image?url=%2Ficons%2Fclassic.png&w=48&q=75 HTTP/2
Host: allyourbase.virtual.tech
// Other headers not shown.

From the request above, you can see that parameter url, w and q are having predictable values that can be modified in the request and overalls are parsed by the frontend. Similarly as the content spoofing vulnerability from above we are able to modify the parameter with different values and the following error messages appear:

Request:

GET /_next/image?url=%2F../../../&w=828&q=011-10 HTTP/2
Host: allyourbase.virtual.tech

Response:

HTTP/2 400 Bad Request
Unable to optimize image and unable to fallback to upstream image

Request:
GET /_next/image?url=%2Ffoo.png&w=828&q=011-10 HTTP/2
Host: allyourbase.virtual.tech

Response:

HTTP/2 400 Bad Request
The requested resource isn't a valid image.

For parameters q and w:

Request:
GET /_next/image?url=%2Ficons%2Fclassic.png&w=48&q=7500 HTTP/2
Host: allyourbase.virtual.tech

Response:

"q" parameter (quality) must be a number between 1 and 100

Request:
GET /_next/image?url=%2Ficons%2Fclassic.png&w=4800&q=75 HTTP/2
Host: allyourbase.virtual.tech

Response:

"w" parameter (width) of 4800 is not allowed

Recommendation

Effective error handling mechanisms must manage any feasible set of inputs while ensuring robust security measures. They should generate clear and concise error messages, which are then logged to facilitate the review of their causes, whether stemming from site errors or potential hacking attempts. Furthermore, error handling should not be limited to user inputs; it must also encompass errors arising from internal components, including system calls, database queries, and other internal functions.

References

- Improper error handling

Strict transport security not enforced.

Summary

The application fails to prevent users from connecting to it over unencrypted connections. An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP, so that if a targeted user follows a link to the site from an HTTP page, their browser never attempts to use an encrypted connection. The sslstrip tool automates this process.

To exploit this vulnerability, an attacker must be suitably positioned to intercept and modify the victim's network traffic. This scenario typically occurs when a client communicates with the server over an insecure connection such as public Wi-Fi, or a corporate or home network that is shared with a compromised computer. Common defenses such as switched networks are not sufficient to prevent this. An attacker situated in the user's ISP or the application's hosting infrastructure could also perform this attack. Note that an advanced adversary could potentially target any connection made over the Internet's core infrastructure.

Classification

Informational

Affected Hosts

allyourbase.virtual.tech

Recommendation

The application should instruct web browsers to only access the application using HTTPS. To do this, enable HTTP Strict Transport Security (HSTS) by adding a response header with the name 'Strict-Transport-Security' and the value

'max-age=expireTime', where expireTime is the time in seconds that browsers should remember that the site should only be accessed using HTTPS. Consider adding the 'includeSubDomains' flag if appropriate.

Note that because HSTS is a "trust on first use" (TOFU) protocol, a user who has never accessed the application will never have seen the HSTS header, and will therefore still be vulnerable to SSL stripping attacks. To mitigate this risk, you can optionally add the 'preload' flag to the HSTS header, and submit the domain for review by browser vendors.

References

- HTTP Strict Transport Security

Appendix A

Vulnerabilities Classification (based on Guidelines for Applying Security Patches)

Classification	Description
Catastrophic	This rating is assigned by the Vice President of Information Risk and Security. The rating will be rarely invoked, primarily for Known Exploited Vulnerabilities. Once a vulnerability is declared as Catastrophic, strict limits apply to the expected remediation timeline. Mitigations such as securing an asset behind a WAF or firewall and actively monitoring logs are the recommended immediate response.
Critical	This rating is given to flaws that could be easily exploited by a remote unauthenticated attacker and lead to system compromise (arbitrary code execution) without requiring user interaction. These are the types of vulnerabilities that can be exploited by worms. Flaws that require an authenticated remote user, a local user, or an unlikely configuration are not classed as critical impact.
Important	This rating is given to flaws that can easily compromise the confidentiality, integrity, or availability of resources. These are the types of vulnerabilities that allow local users to gain privileges, allow unauthenticated remote users to view resources that should otherwise be protected by authentication, allow authenticated remote users to execute arbitrary code, or allow local or remote users to cause a denial of service.
Medium	This rating is given to flaws that may be more difficult to exploit but could still lead to some compromise of the confidentiality, integrity, or availability of resources, under certain circumstances. These are the types of vulnerabilities that could have had a critical impact or important impact but are less easily exploited based on a technical evaluation of the flaw, or affect unlikely configurations.

Low	This rating is given to all other issues that have a security impact. These are the types of vulnerabilities that are believed to require unlikely circumstances to be able to be exploited, or where a successful exploit would give minimal consequences.

Risk rating scale calculation (based on NIST and CVSS 3.1)

Rating	Scale
Critical	9 – 10
High	7.0 - 8.9
Medium	4.0 - 6.9
Low	0 - 3.9

Risk score calculation

Risk Score is calculated using the following equation:

Risk = Likelihood * Business Impact,

where Likelihood is calculated by taking the highest CVSS rated issue and dividing the rating by two, and Business Impact is selected from range 1 to 5 based on the following table.

In this case 5.1 was the highest divided by 2 = 2.55 with BI of 2 = 5.1 of risk score which is Low.

Business Impact	Explanation
1	System is isolated, only Public data and no means of pivoting or reputational harm being done
2	Public data, and no reputation harm when system compromised
3	Internal data with limited impact, pivoting only to systems with BI 1 and almost no reputation harm when system compromised
4	Internal data with limited impact, pivoting only to system with BI 3, moderate reputation harm when system compromised
5	Restricted(+PII) or customer data, or system can be used for pivoting to get to system with high value data/resources, or can be used for great harm reputation

Qualitative Risk Score	Qualitative Risk Level	Rating Definition
25	Critical	Critical risk score indicates the target system or data is at a very high risk of being compromised, and immediate action is required.
15 - 24	High	High risk score indicates the target system or data is at a significant risk of being compromised, and prompt action is required.
8 - 14	Medium	Medium risk score indicates the target system or data is at a moderate risk of being compromised, and ongoing monitoring and improvement is recommended.
4 - 7	Low	Low risk score indicates the target system or data is at a low risk of being compromised, but ongoing monitoring and improvement is recommended.
1 - 3	Very Low	Very Low risk score indicates the target system or data is at a very low risk of being compromised and no immediate action is required.