shieldify

All Your Base

Penetration Testing

Date: 16 July 2024

CONTENTS

1. About Shieldify

Positioned as the first hybrid Web3 Security company, Shieldify shakes things up with a unique subscription-based auditing model that entitles the customer to unlimited audits within its duration, as well as top-notch service quality thanks to a disruptive 6-layered security approach. The company works with very well-established researchers in the space and have secured multiple millions in TVL across protocols, also can audit codebases written in Solidity, Vyper, Rust, Cairo, Move and Go.

Learn more about us at shieldify.org.

2. Disclaimer

The information in this document is confidential and meant for use only by the intended recipient. This security review does not guarantee bulletproof protection against a hack or exploit. The protocol, which this report is intended for, indemnifies Shieldify Security against any responsibility for any misbehavior, bugs, or exploits affecting the audited code during any part of the project's life cycle. It is also pivotal to acknowledge that modifications made to the audited code, including fixes for the issues described in this report, may introduce new problems and necessitate additional auditing.

3. About All Your Base

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.

4. Methodology

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 assessment is conducted with the following phases:

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

A combination of automated and manual methods and follows have been used as a testing methodology.

5. 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:
2. Perform routine testing for the applications on a semi-annual basis.

6. Risk Classification

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.

Table 1: Classification and Description of Vulnerabilities

Classification	Description
Catastrophic	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 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.

7. 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.

7.1 Assessment Scope

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

7.2 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.

8. Findings Summary

The following number of issues have been identified, sorted by their severity:

- **Critical** and **High** issues: **1**
- **Medium** issues: **2**
- **Low** issues: **0**
- **Informational** issues: **1**

9. Findings

ID	Title	Severity	Status
[H-01]	Decimals not Handled by the Application Frontend	High	-
[M-01]	Content Spoofing via Parameters	Medium	-
[M-02]	Improper Error Handling in Parameters	Medium	-
[I-01]	Strict Transport Security Not Enforced	Low	-

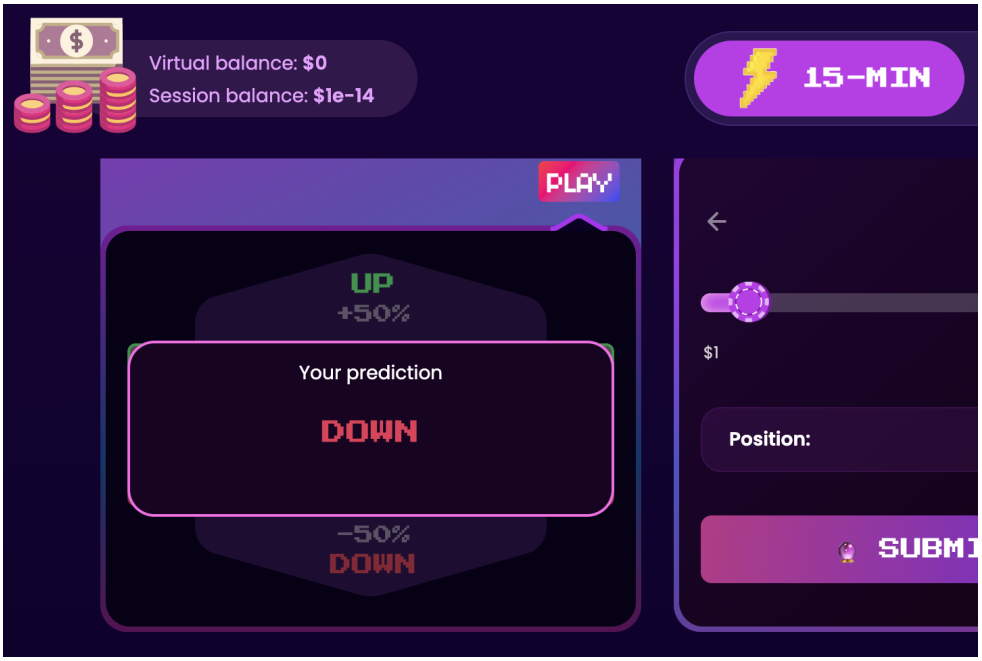
[H-01] Decimals not Handled by the Application Frontend

Severity

High risk

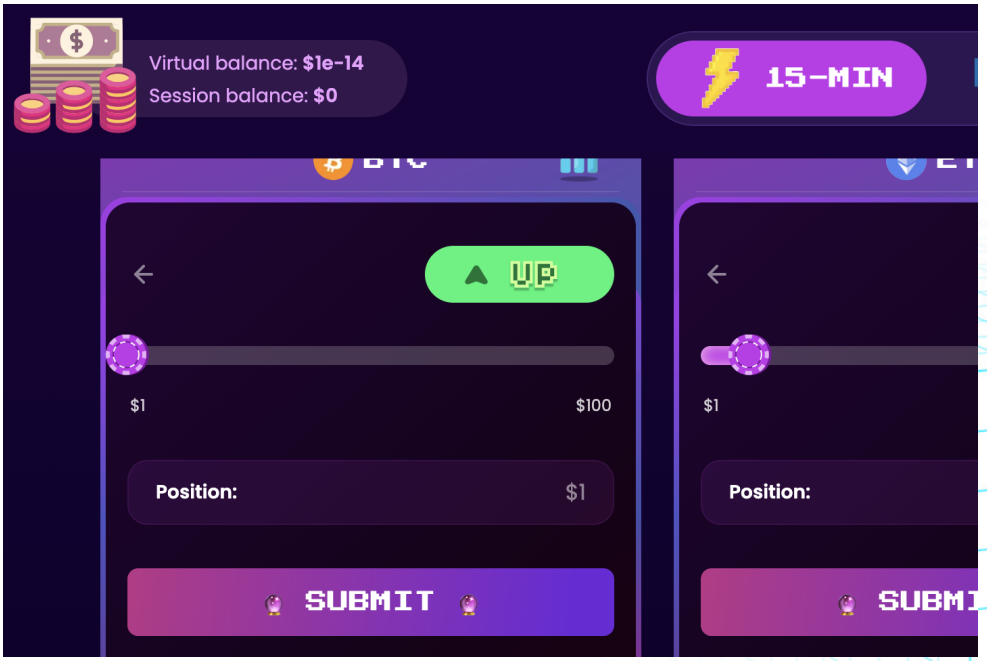
Description

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.



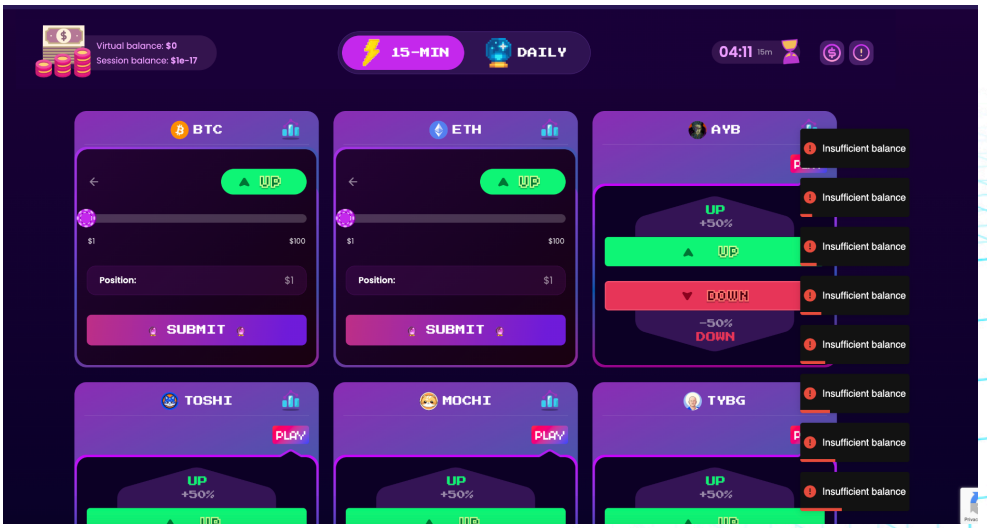
In the image: A user introduced a decimal value **1.0...0001 (14decimals)** in the session balance. When submitting the app processes that amount as an exponential. If the user guesses an incorrect prediction the "Session balance" is still displaying the exponential value without any change.

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).

Location of Affected Code

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.

[M-01] Content Spoofing via Parameters

Severity

Medium risk

Description

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.

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

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:

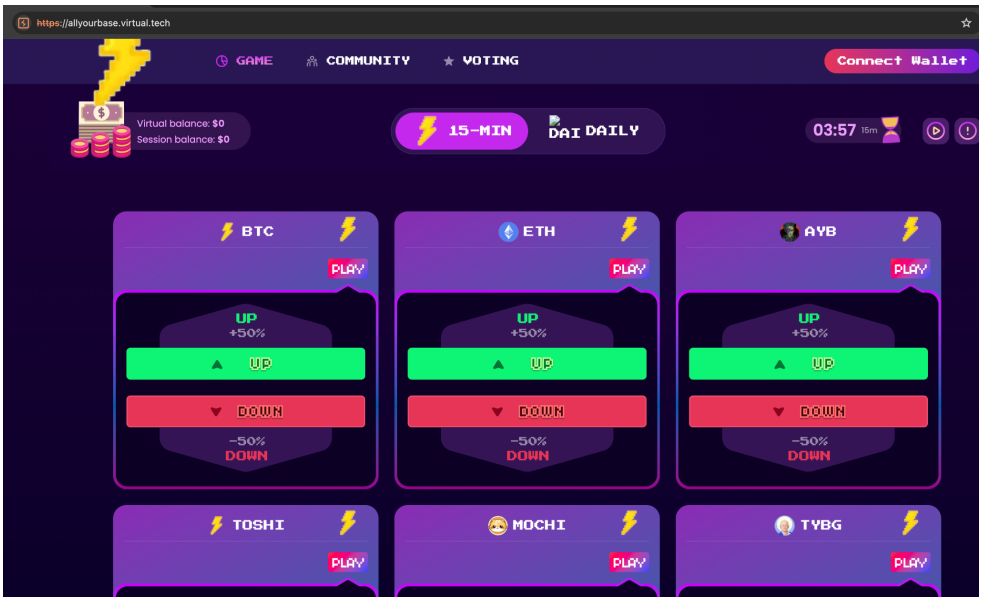


Figure 1: Response (rendered):

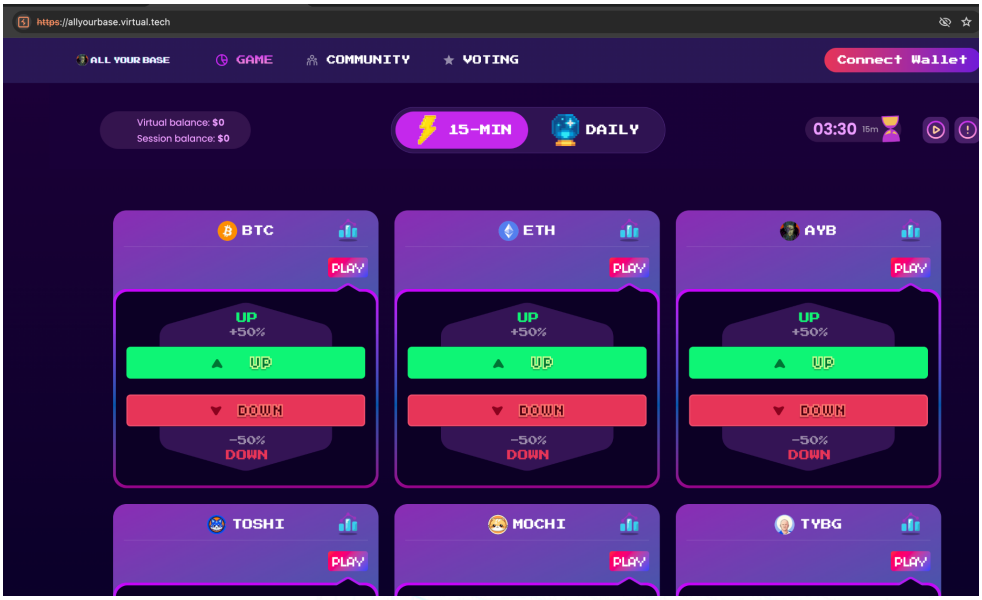


Figure 2: Original

Location of Affected Code

allyourbase.virtual.tech

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.

[M-02] Improper Error Handling in Parameters

Severity

Medium risk

Description

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.

Location of Affected Code

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.

[I-01] Strict Transport Security Not Enforced

Severity

Informational

Description

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.

Location of Affected Code

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.

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

shieldify

Thank you!