shieldify

Dedprz

Penetration Testing

Date: 18 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 Dedprz

The Dedprz website (dedprz.virtual.tech) is a Web2 application developed using the NextJS framework, specifically designed to integrate with a smart contract on the backend. This platform allows players to engage in a coin flip game, where they can place predetermined or customized bets using the $SUSA balance stored in a connected wallet. Players select either heads or tails along with their specified bet amount, awaiting the application's response to determine if they have won or lost. Depending on the outcome, their wallet balance is adjusted accordingly, either increasing or decreasing based on the result of the coin flip game.

4. Methodology

Security assessment of the beecasino 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: https://owaspsamm.org/model/operations/
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 dedprz.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: **3**
- **Low** issues: **2**
- **Informational** issues: **1**

9. Findings

ID	Title	Severity	Status
[H-01]	Decimals not Handled by the Application Frontend	High	-
[M-01]	Content Spoofing	Medium	-
[M-02]	Denial of Service due to Business Logic Flaw (Authenticated)	Medium	-
[M-03]	Improper Error Handling	Medium	-
[L-01]	No Minimum Session Balance is Enforced (Business Logic Flaw)	Low	-
[L-02]	“Recent flips” can Deface the Application	Low	-
[I-01]	Strict Transport Security not Enforced	Informational	-

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

Severity

High Risk

Description

In dedprz.virtual.tech a user can manually enter **0.(decimal)** numbers in the wager amount. Consequently, when a user flips the coin, the result will always show that the user either lost or won $0 (the front end rounds it to 0). Despite this, the session balance will be updated with the decimal values entered, and the Recent Flips table will also reflect these changes.

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.

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).

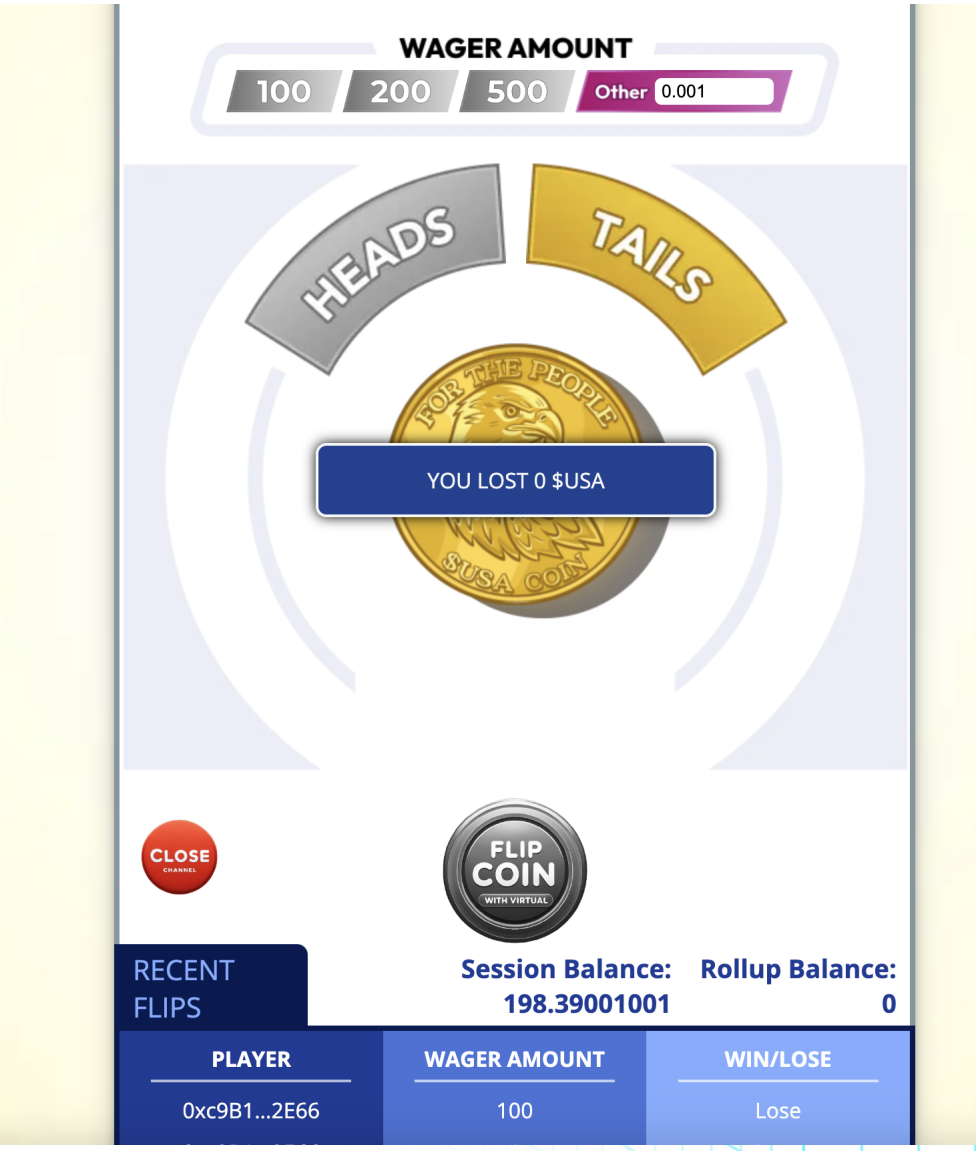

Figure 1: In the image: A user introduced a decimal value **0.001** in the wager amount. The application still processes that amount when flipping the coin but reports **0$** win/loss Session balance and Recent Flips are still updated according to the decimals added/subtracted.

Location of Affected Code

dedprz.virtual.tech

Recommendation

Revise the permitted input for the Wager amount “Other”. 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 and displays always a 0$ in winning or losing when user introduces amounts less than 1.

[M-01] Content Spoofing

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 dedprz.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.

Location of Affected Code

dedprz.virtual.tech

Proof of Concept

Using a proxy intercept the request when browsing to dedprz.virtual.tech. The

following GET request will be available:

GET /_next/image?url=%2Ftails-button-active.png&w=640&q=75 HTTP/2

Host: dedprz.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: dedprz.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: dedprz.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=%2Ftails-button-active.png&w=640&q=**7500** HTTP/2

Host: dedprz.virtual.tech

Response:

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

Request:

GET /_next/image?url=%2Ftails-button-active.png&w=**6400**&q=750 HTTP/2

Host: dedprz.virtual.tech

Response:

"w" parameter (width) of 6400 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:

Request: GET /_ next/image?url=%2Ftails-button-active.png&w=640&q=75 HTTP/2 // Change this to heads! Host: dedprz.virtual.tech

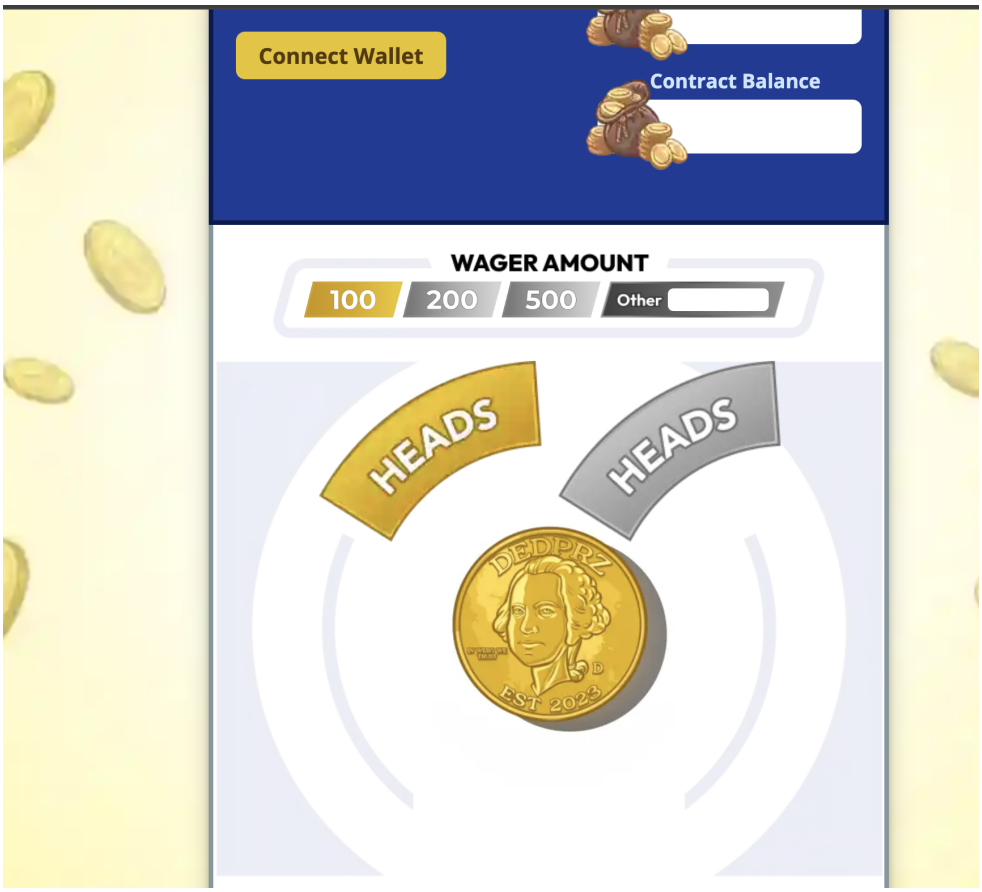

Figure 2: Response (rendered):

Figure 3: Another defacement rendered example:

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] Denial of Service due to Business Logic Flaw (Authenticated)

Severity

Medium Risk

Description

When a user connects their wallet to dedprz.virtual.tech, the application does not adequately handle the scenario where the user might accidentally or intentionally press the withdraw button as their

first interaction. When this happens, all remaining options/buttons become locked, except for the "Get $USA to Play" button. While this functionality may be convenient during an ongoing game, it creates the impression that the entire application is frozen if it occurs during the user's initial interaction. Although refreshing the page resolves the issue, this situation effectively results in a form of Denial of Service (DoS), as there is no in-app mechanism for recovery.

Location of Affected Code

dedprz.virtual.tech

Proof of Concept

With the connected wallet, navigate to dedprz.virtual.tech and hit the Withdraw button. After it, the remaining functionality is completely frozen and the user is unable to start a game until it refreshes the page.

Recommendation

Please revise the application's business logic to ensure that if the user presses the withdraw button at an early stage, the UI and browsing experience are not locked without a clear reason, and the user can recover without the need of refreshing the page.

[M-03] Improper Error Handling

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 dedprz.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

dedprz.virtual.tech

Proof of Concept

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

GET /_next/image?url=%2Ftails-button-active.png&w=640&q=75 HTTP/2

Host: dedprz.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: dedprz.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: dedprz.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=%2Ftails-button-active.png&w=640&q=**7500** HTTP/2 Host: dedprz.virtual.tech

Response:

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

Request:
GET /_next/image?url=%2Ftails-button-active.png&w=**6400**&q=750 HTTP/2 Host: dedprz.virtual.tech

Response:

"w" parameter (width) of 6400 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.

[L-01] No Minimum Session Balance is Enforced (Business Logic Flaw)

Severity

Low Risk

Description

The application does not require or clearly shows a minimum session balance to use the Flip Coin button. When a new user deposits the minimum allowed amount (which is 1), the application will repeatedly display the message "Deposit $USA to play" whenever the user tries to flip the coin. In this way, while the minimum balance is implied to be 100 it does not inform the user of the minimum amount needed to start the game precisely.

Figure 4: In the picture: The overall session balance is 3, but if a user tries to flip the coin the application will ask to deposit $USA to play without any further details on the minimum required amount.

Location of Affected Code

dedprz.virtual.tech

Recommendation

The application should clearly define the requirements for using the Flip Coin feature and always notify the user of the minimum balance needed to start the game. This is particularly important since each transaction incurs a gas fee for the user.

[L-02] "Recent flips" can Deface the Application

Severity

Low Risk

Description

Whenever a user flips the coin, the Recent Flips section is updated with a table displaying the player's name, the wager amount, and the result for the current session. However , a defacement issue arises as this table grows without limit, potentially disrupting the website's layout if a user engages in many rounds within a single session.

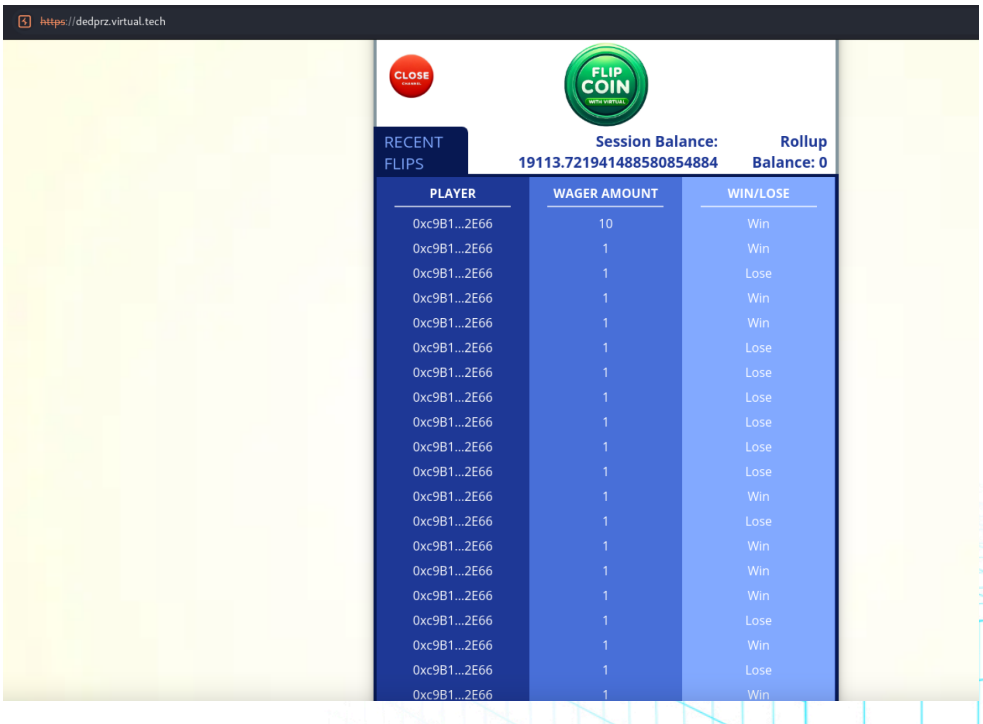

Figure 5: In the image: The recent flips table is growing without limit. The website might fall into an involuntary defacement.

Location of Affected Code

dedprz.virtual.tech

Recommendation

While defacement does not pose a significant security threat, functionality that impacts application integrity can negatively affect the user experience. It is recommended to manage the Recent Flips table using a controlled data structure or to limit the number of entries displayed in a single session.

[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

dedprz.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.

shieldify

Thank you!