·Decurity·

UI Security Assessment Report

SYMBIOSIS LABS LTD.

Decurity, 2022

Contents

1. General Information

This report contains information about the results of the security audit of the security audit of the dApp frontend UI of Symbiosis Labs Ltd. (hereafter referred to as “Customer”) smart contracts, conducted by Decurity in the period from 03/18/2022 to 03/23/2022.

1.1. Introduction

Tasks solved during the work are:

- Security assessment of the Customer’s web applications, including attempts to gain an access to the critical assets and confidential data,
- Development of the short-term and long-term technical and organizational recommendations based on the findings and industry best practices,
- Evaluation of applied bug fixes and security improvements, control checks of the findings.

1.2. Scope of Work

The testing scope included the Symbiosis web app (Swap/Pools interfaces) accessible on the https://app.symbiosis.finance/ and https://testnet.symbiosis.finance/ URLs. The source code for the app has been provided by the Customer in the following repository: https://github.com/symbiosis-finance/web-app. Initial review was done for the commit 538aaa8bf630de0bcab8e17ea8a6bf25a8f82914 and the re-testing was done for the commit c5b92099692d6ad80368da9820e89d7d08c84625.

1.3. Threat Model

The assessment presumes actions of an external threat actor whose purpose is to damage the application or its users.

1.4. Weakness Scoring

An expert evaluation scores the findings in this report, an impact of each vulnerability is calculated based on its ease of exploitation (based on the industry practice and our experience) and severity (for the considered threats).

2. Summary

During the testing of the Customer's system no exploitable security issues have been identified, the overall security level of the Customer's application can be described as **high**.

The suggestions included fixing the low-risk issues and some best practices (see 3.1).

The Symbiosis Finance team has given the feedback for the suggested changes and explanation for the underlying code.

2.1. Suggestions

The table below contains the discovered issues, their risk level, and their status as of Aug 15, 2022 .

Table. Discovered weaknesses

Issue	Target	Risk Level	Status
Lack of Content Security Policy (CSP)	https://testnet.symbiosis.finance/	Low	Partially Fixed
Window opener hijacking (Tabnabbing)	https://testnet.symbiosis.finance/	Informational	Acknowledged (Fixed in the major web browsers)
Undefined variable	https://testnet.symbiosis.finance/	Informational	Fixed

3. Findings

3.1. Lack of Security Headers

Risk Level: Low

Status:

During the re-test it has been confirmed that a version of Content Security Policy has been implemented in the commit https://github.com/symbiosis-finance/web-app/commit/e5d12d160e3bfa20fe3397fff075d4e1bc4f7401. However, it still allows evaluation and inline execution of the JavaScript code.

Other security headers also have been added, and the following tool marks them as safe: https://securityheaders.com/?q=https%3A%2F%2Fapp.symbiosis.finance%2F&followRedirects=on

It's worth noting that it's hard to fully implement the CSP because of the limitations it imposes on the code. So adding a loose policy is a first step in refactoring the application.

Resources::

- https://testnet.symbiosis.finance/

References:

- https://csp-evaluator.withgoogle.com/
- https://securityheaders.com/

Remediation:

- Enable CSP using one of the methods such as an HTTP header or a meta-tag,
- Make the policy as strict as possible without breaking the app,
- Evaluate the policy using the websites https://csp-evaluator.withgoogle.com/ and https://securityheaders.com/.

Description:

Content Security Policy is an additional feature than enables mitigation of some types of Cross-Site Scripting (XSS) or similar attacks. If CSP is not implemented, there's no direct risk but it makes it easier for the attackers to carry out Cross-Site-Scripting attacks.

Proofs:

The web application has no Content Security Policy implemented.

The screenshot below shows the HTTP response without the CSP:

3.2. Window Opener Hijacking (Tabnabbing)

Risk Level: Informational

Status:

The tabnabbing attack has been made obsolete by the major browser updates that added the *noopener* attribute by the external links by default.

Resources::

- https://testnet.symbiosis.finance/

References:

- https://developer.mozilla.org/en-US/docs/Web/API/Window/open
- https://developer.mozilla.org/en-US/docs/Web/HTML/Link_types/noopener

Remediation:

- Enable CSP using one of the methods such as an HTTP header or a meta-tag,
- Make the policy as strict as possible without breaking the app,
- Evaluate the policy using the websites https://csp-evaluator.withgoogle.com/ and https://securityheaders.com/.

Description:

When a user navigates to a new window created by the window.open call or clicks a link to an external site ("target"), the target="_blank" attribute causes the target site's contents to be opened in a new window or tab, which runs in the same process as the original page. The window.opener object records information about the original page that offered the link. If an attacker can run script on the target page, then they could read or modify certain properties of the window.opener object, including the location property — even if the original and target site are not the same origin.

Proofs:

The web application produces links to untrusted external sites outside of its sphere of control, but it does not properly prevent the external site from modifying security-critical properties of the window.opener object, such as the location property.

Examples of such links in the application's source code:

```
src/app/components/PageHeader/PageHeader.tsx:                    <a  href={infoLink}
target="_blank" rel="noreferrer">
src/app/pages/PendingTransactionsPage/PendingTransactionItem/PendingTransactionIte
m.tsx:              <a href={txLink} target="_blank" rel="noopener noreferrer">
```

Examples of window.open calls:

```
src/app/modals/ConnectWallet/ConnectWallet.tsx:
window.open('https://metamask.io/', '_blank')
src/app/modals/ConnectWallet/ConnectWallet.tsx:
window.open('https://www.xdefi.io/', '_blank')
src/app/modals/ConnectWallet/ConnectWallet.tsx:
window.open('https://coin98.com/', '_blank')
src/app/modals/ConnectWallet/ConnectWallet.tsx:
window.open('https://trustwallet.com', '_blank')
src/app/modals/ConnectWallet/ConnectWallet.tsx:
window.open('https://onto.app', '_blank')
```

The links are hardcoded and cannot be controlled by an attacker, therefore, the risk level is low. The only way this gets exploited is if an attacker controls one of the linked websites.

3.3. Undefined Variable

Risk Level: Informational

Status:

The error has been fixed.

Resources::

- https://testnet.symbiosis.finance/

References: -

Remediation:

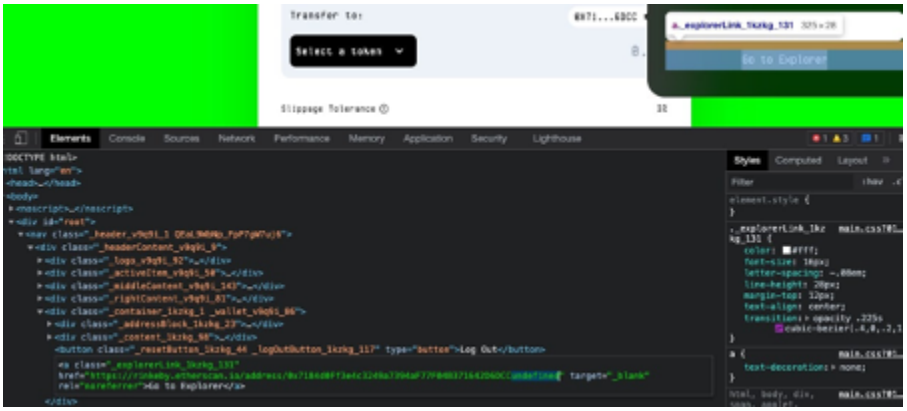
- Change the getExplorerLink implementation to avoid the incorrect links.

Description:

The non-functioning interfaces or undefined behaviour may lead to errors that affect the users' security.

Proofs:

Under certain circumstances the explorer link generated by the web application is incorrect as shown on the screenshot below:



4. Appendix

4.1. About us

The Decurity (former DeFiSecurity.io) team consists of experienced hackers who have been doing application security assessments and penetration testing for over a decade.

During the recent years, we've gained expertise in the blockchain field and have conducted numerous audits for both centralized and decentralized projects: exchanges, protocols, and blockchain nodes.

Our efforts have helped to protect hundreds of millions of dollars and make web3 a safer place.