HALBORN

Seascape NFT Marketplace

WebApp Pentest

Prepared by: **Halborn**

Date of Engagement: **January 17th, 2022 - February 11th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	01/17/2022	Alessandro Cara
0.2	Document Amended	02/10/2022	Alessandro Cara
0.3	Draft Review	02/11/2022	Gabi Urrutia
1.0	Remediation Plan	02/25/2022	Alessandro Cara
1.1	Remediation Plan Review	02/25/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Alessandro Cara	Halborn	Alessandro.Cara@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Halborn was engaged by Seascape to perform a security assessment of their NFT Marketplace. The assessment was conducted between January and February 2022. The application allowed its users to browse NFTs within the marketplace, sell and buy NFTs, explore additional Seascape offerings and change network between the supported ones. All the transactions were handled by the supported blockchain protocols presented below:

- BSC
- Ethereum
- Moonriver
- Moonbeam

1.2 AUDIT SUMMARY

The team at Halborn was provided two weeks for the engagement and assigned one full-time security engineers to audit the security of the application. A security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this assessment was to:

- Ensure that the application behaves as intended
- Identify potential security issues with the application

While the assessment did not reveal any issues which could lead to the full compromise of the application, several security issues were identified. These stemmed from the use of outdated and vulnerable software, the lack of security related headers, and the disclosure of sensitive information on the code repository.

It was identified that several packages in use by the frontend application and the API were outdated and contained publicly disclosed vulnerabil-

ities. Should an attacker understand how these components are used by the application, they would be able to execute code on the solution and potentially access sensitive data.

Furthermore, a private key was found on the GitHub repository. While this repository was private, this is considered bad practice. Secrets should not be shared on a code repository as this can be seen by more than the intended users, and these could potentially be committed to a public repository at a later date.

Halborn recommends that the following high-level remedial actions are implemented to mitigate the identified security issues:

- Implement a patch management policy to ensure that software packages are up-to-date at all times
- Enforce the use of HTTP Security headers
- Remove sensitive information from GitHub repositories
- Review the transport layer security (TLS) configuration to disable connections with outdated protocols

In summary, Halborn identified some security issues that were addressed in both the test and production environments.

1.3 TEST APPROACH & METHODOLOGY

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

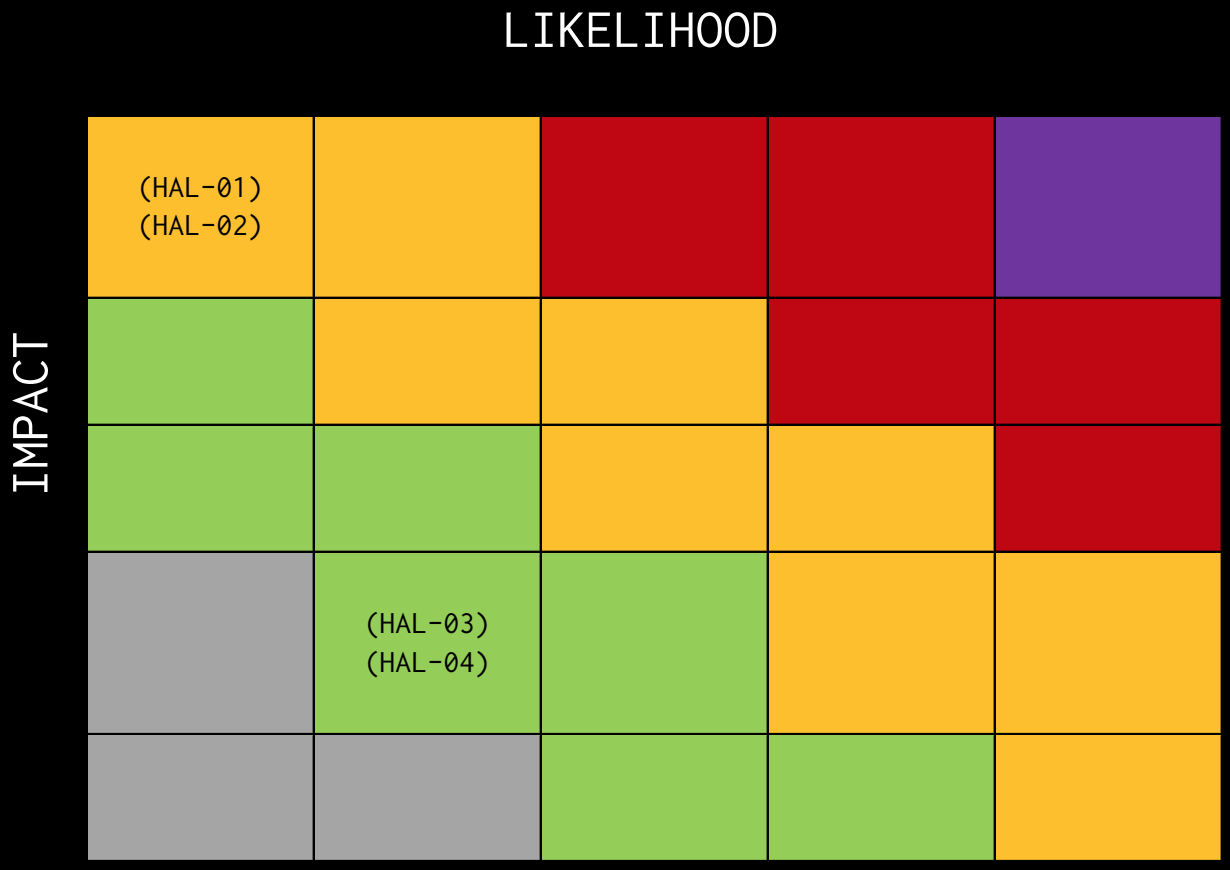
The assessment was scoped down to the application, accessible with the links below:

- Frontend
- API

It should be noted that this assessment was carried out on a test environment. However, the identified issues were confirmed to be present and finally **solved** by Seascape on the production environment too.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	2	2	0



EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - HARDCODED PRIVATE KEY IN THE REPOSITORY	Medium	SOLVED - 02/17/2022
HAL02 - USE OF PACKAGES WITH KNOWN VULNERABILITIES	Medium	SOLVED - 02/17/2022
HAL03 - MISSING HTTP SECURITY HEADERS	Low	SOLVED - 02/25/2022
HAL04 - TLS MISCONFIGURATIONS	Low	SOLVED - 02/25/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) HARDCODED PRIVATE KEY IN THE REPOSITORY - MEDIUM

Description:

Testing revealed that the backend private repository on GitHub contained a private key. While the repository was private and therefore only accessible by Seascape's team, this is considered bad practice as secrets should be securely stored and shared between the development team on a need to know basis. Additionally, these might be later re-used by Seascape and should they fall in the wrong hands, they could be used to access sensitive services and data.

Location:

The private key could be found by browsing to the seascape-antibot-d98baf808e54.json file within the backend source code.

Risk Level:

Likelihood - 1
Impact - 5

Recommendation:

Halborn recommends that the private key is replaced and that it is removed from the repository. Instead, environment variables should be used to pass on arguments to the code, should it be needed. Additionally, Seascape should research secret sharing solutions that could be used to share secrets used in development environments between engineers and developers. Finally, GitHub commits should be reviewed by a second person before being pushed to ensure that unintended data is not committed to the repository.

Remediation Plan:

SOLVED: This issue was solved by using environment variables.

3.2 (HAL-02) USE OF PACKAGES WITH KNOWN VULNERABILITIES - MEDIUM

Description:

The application uses third-party dependencies to delegate handling of different kind of operations, e.g., generation of document in a specific format, HTTP communications, data parsing of a specific format, etc. However, the dependencies bring forth an expected downside where the security posture of the real application is now resting on them. Several imported packages were found to not be updated to the latest version and presented various security risks, as described below.

Vulnerabilities List:

FRONTEND

Title	Package	Severity
Prototype Pollution	immer	CRITICAL
Cross-Site Scripting (XSS)	next-js	HIGH
Open Redirect	next-js	HIGH
Unexpected Server Crash	next-js	HIGH
Exposure of Sensitive Information	node-fetch	HIGH
Exposure of Sensitive Information	simple-get	HIGH
Denial of Service (DoS)	trim-newlines	HIGH

API

Title	Package	Severity
Exposure of Sensitive Information	simple-get	HIGH
Exposure of Sensitive Information	node-fetch	HIGH

Risk Level:

Likelihood - 1

Impact - 5

Recommendation:

It is highly recommended performing automated analysis of the dependencies from the birth of the project and if they happen to contain any security issues. The Seascape team needs to be aware of it and apply the required mitigation measures to secure the affected application.

Remediation Plan:

SOLVED: Seascape updated the highlighted packages on the frontend and the API.

3.3 (HAL-03) MISSING HTTP SECURITY HEADERS - LOW

Description:

The assessment revealed that several security headers were not enforced by the application. These headers are used by client browsers to ensure various security controls are appropriately implemented during the normal functioning of the application.

- X-Content-Type-Options, which indicates that the MIME types advertised in the Content-Type headers should not be changed and be followed.
- X-Frame-Options, which indicates whether a browser should be allowed to render a page in a <frame>, <iframe>, <embed> or <object>.
- Content-Security-Policy, which allows website administrators to control resources the user agent is allowed to load for a given page.
- Strict-Transport-Security (HSTS) - which enforces secure transmission by letting a website tell browsers that it should only be accessed using HTTPS, instead of using HTTP.
- Referrer-Policy - specifies what information, or parts of it, should be sent in the Referer header with the request that prompted the redirection.
- Pragma - using the “no-cache” directive forces the browser to query the server before downloading a cached copy of the page, resulting in the download of the most recent version.
- Expires - includes a date, period, or value indicating when the server’s response is no longer correct.

It should be noted that in some situations Strict-Transport-Security and X-Content-Type-Options (and in some cases X-Frame-Options) do not necessarily have to be enabled by the application **directly**. These headers can be injected by supporting load balancers or web application accelerators.

Recommendation:

Seascape should review the above security headers and ensure that, where appropriate, these headers are included within all exposed endpoints and services. This allows Seascape to ensure the defence-in-depth approach is achieved thought the application.

References:

Strict-Transport-Security

X-Content-Type-Options

X-Frame-Options

Content-Type

Remediation Plan:

SOLVED: Security related headers were added to the platform.

3.4 (HAL-04) TLS MISCONFIGURATIONS - LOW

Description:

Several misconfigurations were identified within the application's SSL/TLS configuration that could compromise the security of communications. While attacks that target weak cipher suites or algorithms are complex to execute, with the steady progress of computational power these attacks become easier to achieve over time. As such, it is recommended to configure the connection to comply with security best practices.

Many protocols may be used in establishing a secure connection. In the past, various vulnerabilities have existed surrounding legacy protocols and their associated cipher suites. Newer versions of these secure communication protocols and modern browsers address the security vulnerabilities identified in older versions. For example, the POODLE (Padding Oracle On Downgraded Legacy Encryption) vulnerability allowed information to be extracted from communication headers and the DROWN attack allowed SSLv2 traffic to be decrypted. Neither of these attacks continue to pose a threat in modern browsers. However, if a user accesses the application with an outdated browser, these vulnerabilities may remain exploitable.

Encryption ciphers are used to protect communication channels between a client and a web server and are negotiated when a client initially connects to the server. This negotiation involves agreeing on a cipher suite and a combination of protocols, encryption algorithms, key lengths, and hashing algorithms supported by both parties. To support a wide range of browsers, web servers typically support ciphers of various strengths. However, some encryption algorithms do not provide adequate security, due to issues such as implementation flaws or inadequate key lengths. Additionally, permitting insecure cipher suites puts users with outdated software at greater risk, as an attacker could downgrade a user's connection to an insecure cipher.

The SSL cipher suites support a variety of key lengths. Only suites with sufficiently long keys should be permitted, as short keys may allow an

attacker to perform a brute-force attack to decrypt the traffic.

Analysis:

The application allowed connections with TLS v1.0 and TLS v1.1 to be established, as well as allowing the use of weak CBC cipher suites.

Listing 1

```
 1  SSLv2      not offered (OK)
 2  SSLv3      not offered (OK)
 3  TLS 1      offered (deprecated)
 4  TLS 1.1    offered (deprecated)
 5  TLS 1.2    offered (OK)
 6  TLS 1.3    offered (OK): final
 7  NPN/SPDY   not offered
 8  ALPN/HTTP2 h2, http/1.1 (offered)
 9
10 Ciphers:
11 ECDHE-RSA-AES128-SHA AES128-SHA ECDHE-RSA-AES256-SHA AES256-SHA
       DES-CBC3-SHA
```

Recommendation:

As the application functionality is hosted across multiple servers, the SSL/TLS configuration of each should be reviewed and updated to comply with security best practice and to minimize the risk of existing and future SSL vulnerabilities.

- It is recommended to disable TLSv1.0 and TLSv1.1. It should be noted that all major browser vendors coordinated to remove support for both TLSv1.0 and TLSv1.1 in March 2020, however these depreciated versions will still be available to outdated browser versions.
- Support for the weak cipher suites identified above should be removed.
- Key lengths of 256 bits and above should be used for symmetric encryption algorithms and a length of 4096 bits should be used for

RSA algorithms. For asymmetric keys generated with ECC algorithms, the minimum recommended key size is 512 bits.

Finally, it is recommended that forward secrecy is enabled on all hosts and that session renegotiation is not supported. If a compromise of the server's private key, forward secrecy will prevent an attacker from using it to decrypt previously recorded traffic.

Remediation Plan:

SOLVED: The team at Seascape amended the TLS configuration of the servers to only accept connections with version 2 and version 3.

THANK YOU FOR CHOOSING

// HALBORN