// HALBORN

Seascape Minigames

Webapp Pentest

Prepared by: **Halborn**

Date of Engagement: **December 30th, 2021 - January 14th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	12/30/2021	alessandro.cara
0.2	Document Amended	01/14/2022	Alessandro Cara
0.3	Draft Review	01/17/2022	Gabi Urrutia
1.0	Remedation Plan	02/15/2022	Alessandro Cara
1.1	Remedation Plan Review	02/15/2022	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 mini-games' application, with a focus on the Zombie mini-game. The assessment was carried out between December 30th, 2021 and January 17th, 2022.

The application exposed the following functionality:

- Connect a web3 wallet
- Change network between BSC, ETH, and Moonriver
- List of available minigames
- Play available minigames (stake, unstake and claim rewards)

The ZombieFarm mini-game comprised of different challenges which allowed users to stake their tokens and NFTs, and claim their rewards. The mini-game was written in JavaScript (JS) with an Elixir API and the backend application running on the ThinkPHP framework.

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

Halborn was able to identify multiple vulnerabilities which could allow an external attacker to access the application content management system

(CMS) and tamper with the application functionality, phish Seascape's users via malicious links and potentially further compromise Seascape's infrastructure.

In more details, the admin CMS password was set to an easy-to-guess value, which was set for both the testing and the production environments. Once logged in, a user could recover sensitive information about the hosting infrastructure and technology stack of the application, amend the application's pages with their own malicious content, add or remove mini-games, and send emails to arbitrary email addresses. It should be noted that the Seascape team promptly changed the compromised credentials on both environment to reduce the risk of an attacker from accessing the administrative interface.

Additionally, Halborn observed that the PHP framework in use was outdated and likely vulnerable to a publicly disclosed vulnerability. In addition, several JavaScript (JS) libraries imported by the application were out-dated. Attackers might be able to identify where vulnerable components are in use and exploit the application.

To remediate the identified security risks, the following high level remedial actions were recommended by Halborn:

- Enforce a strong password policy and enroll developers into security training to ensure that passwords are securely created and stored
- Implement a patch management policy to ensure that the latest and stable versions of software and third-party libraries are in use
- Add the secure and HttpOnly flag to session cookies
- Restrict the disclosure of information regarding the technology stack in use via debug messages and server headers
- Implement Security Related HTTP Headers such as Content Security Policy (CSP) and HTTP Strict Transport Security Header (HSTS)

In summary, Halborn identified some security risks that were mostly addressed by the Seascape team.

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.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH

7 - 6 - MEDIUM

5 - 4 - LOW

3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

The assessment was conducted on the web application available at Seascape test environment mini-games. It is worth mentioning that this was the testing environment. Vulnerabilities have been reproduced where possible on the production application available at Seascape production mini-games application.

It should be noted that depending on the blockchain network that a user would connect, the following three URLs for the testing environment and the production environment would be accessed:

- https://beta-bsc.seascape.network/ - https://bsc.seascape.network/
- https://beta-eth.seascape.network/ - https://eth.seascape.network/
- https://beta-moonriver.seascape.network/ - https://moonriver.seascape.network/

Finally, the backend application programming interface (API) was accessible at testnet API and production API.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
1	0	4	3	3

LIKELIHOOD

IMPACT

(HAL-05)				(HAL-01)
		(HAL-02) (HAL-03) (HAL-04)		
	(HAL-06) (HAL-07) (HAL-08)			
(HAL-09) (HAL-10) (HAL-11)				

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
WEAK CMS PASSWORD	Critical	SOLVED
ADMIN PANEL PUBLICLY EXPOSED	Medium	PARTIALLY SOLVED
SENSITIVE INFORMATION DISCLOSURE VIA VERBOSE ERROR MESSAGES	Medium	PARTIALLY SOLVED
OUTDATED THINKPHP VERSION	Medium	RISK ACCEPTED
USE OF PACKAGES WITH KNOWN VULNERABILITIES	Medium	SOLVED
MISSING HTTP SECURITY HEADERS	Low	RISK ACCEPTED
SESSION COOKIE WITHOUT SECURE AND HTTPONLY FLAGS SET	Low	RISK ACCEPTED
TLS MISCONFIGURATIONS	Low	RISK ACCEPTED
HARDCODED AND WEAK CREDENTIALS	Informational	ACKNOWLEDGED
FRONT-END DISPLAY ERRORS	Informational	SOLVED
WEB SERVER BANNER DISCLOSURE	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) WEAK CMS PASSWORD - CRITICAL

Description:

Testing revealed that the application Content Management System (CMS) management interface was accessible with an easy-to-guess username and password combination. The username was the same as the application name, as well as the password being trivial to guess. Additionally, the same credentials allowed to access both the testing and the production environments.

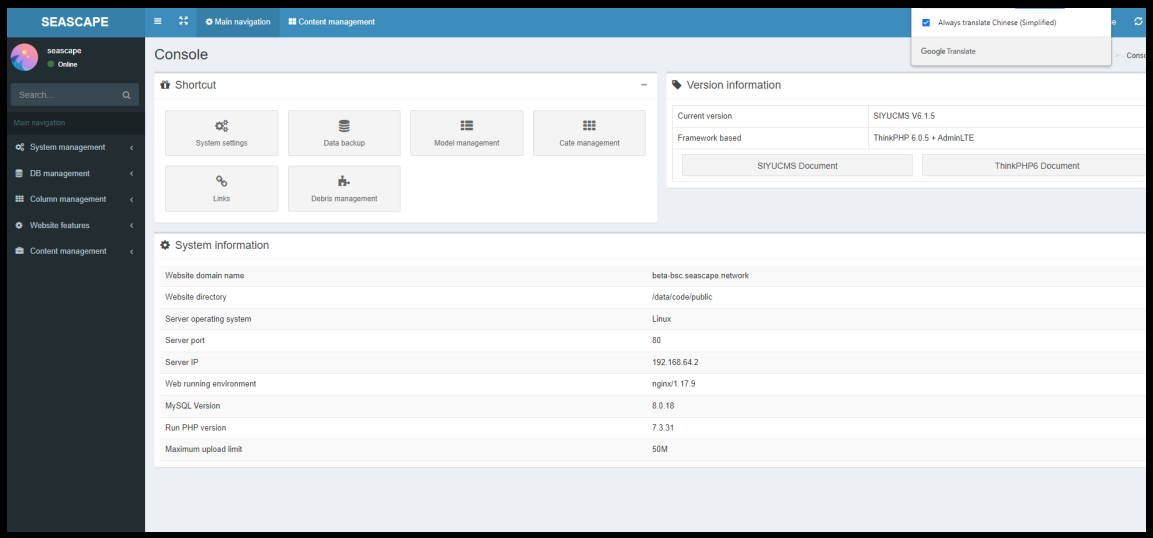
After logging in, users were presented with different interfaces that allowed to modify the content displayed to the users, as well as collecting sensitive information about the backend infrastructure.
For instance, users with access to the CMS could edit pages content, insert their own malicious links, send phishing emails to Seascape newsletter contacts, add and modify mini-games details.

It should be noted that the full criticality of this issue was not assessed by Halborn, as Seascape promptly amended the CMS credentials to prevent any unauthorized access to the platform.

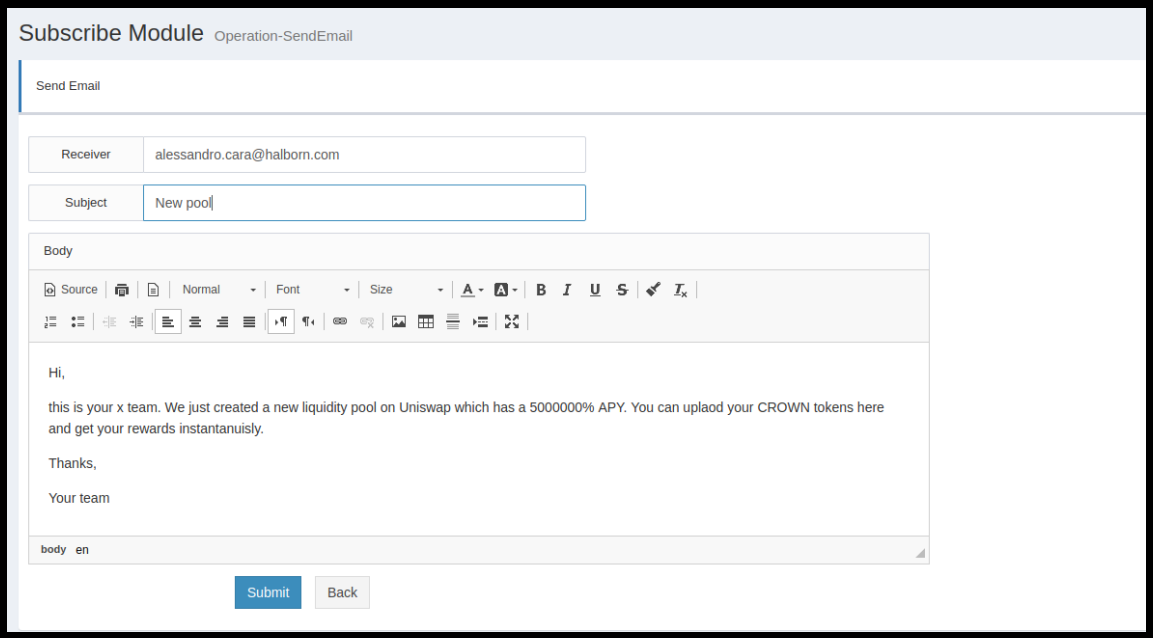
Code Location:

The following screenshots show some of the functionality which could be exploited by a malicious user with access to the CMS backend.

Below is presented the sensitive information that was disclosed about the server's technology stack in use:



The following functionality could be abused to send phishing emails that would come from Seascape's official domain. Thus, users would be more likely to fall victim to the attack:



The edit page functionality could be used to deface the application's front end as well as insert malicious code such as JavaScript (which could trigger Cross-Site Scripting (XSS) vulnerabilities) and malicious links.

Risk Level:

Likelihood - 5
Impact - 5

Recommendation:

Whilst the issue was remediated by Seascape during the assessment window, the following recommendations for secure password generation should be taken into consideration.

A strong password policy can significantly reduce the risk of an attacker being able to gain access through a password guessing attack.

A strong password should be at least 16 characters in length. Complexity requirements (i.e., upper and lowercase letters, numbers and special characters) will also increase a password's strength, but these requirements can make it more difficult for people to set and remember passwords, and do not contribute as much to password strength as pure length. Longer passwords without complex character requirements are considered both easier for humans to remember and more difficult to guess or brute-force via automated means. Therefore, it is recommended that, to reduce the burden on users' memories, complexity requirements are removed from the policy in tandem with the increase in minimum length.

To accommodate the increased length requirement, users should be encouraged to consider their passwords to be passphrases rather than single words. A phrase such as `todayiwillworkverywellindeed` is easier to remember and significantly less likely to be guessed through standard password guessing techniques than a short but complex password such as `Pa$$w0rd1`.

As administrative accounts are commonly targeted by attackers, it is recommended that these accounts are further secured by requiring a higher minimum number of characters, such as 20.

As an additional measure, it is advised to add commonly used passwords to a blacklist, which would significantly increase the resilience of user accounts to password guessing attacks. Users should not be able to

choose passwords containing words that appear in the blacklist, which should contain the following:

- The username
- The company name
- The application name
- Months and seasons
- Commonly used weak passwords (‘password‘‘, ‘secret‘, etc.)

Remediation Plan:

SOLVED: The Seascape team amended the password to be more secure.

3.2 (HAL-02) ADMIN PANEL PUBLICLY EXPOSED - MEDIUM

Description:

Testing revealed that the application exposed different login pages to the users. One of them was the CMS admin panel. It should be noted that one of the login pages allowed users to register an account. Seascape explained that users who log in with this interface would only be allowed to access extra content should an admin authorized them to do so.

Code Location:

The following three URLs allowed access to the aforementioned login pages:

- https://beta-bsc.seascape.network/admin/Login/index.html
- https://beta-bsc.seascape.network/index/user/login
- https://beta-bsc.seascape.network/index/login/login

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Where possible, admin panels should only be accessible to the required users, for instance by implementing IP address whitelisting. Additionally, it is recommended to remove any unused functionality that might be used for testing, as this opens up unnecessary attack surface.

Finally, strong password policy requirements should be followed to ensure that any unauthorized user cannot access sensitive content. Please refer to the **Weak CMS Password** recommendations above.

Remediation Plan:

PARTIALLY SOLVED: The Seascape team acknowledged the issue and informed that they have implemented IP address whitelisting only on the authenticated part of the application, but not on the panel itself.

3.3 (HAL-03) SENSITIVE INFORMATION DISCLOSURE VIA VERBOSE ERROR MESSAGES - MEDIUM

Description:

Within the testing environment, it was identified that by accessing nonexistent pages, a stack trace would be presented to the user. Within the content of the server response, sensitive information was identified, such as database hostname, username, and password. While the database host was inaccessible, these credentials might be used on other environments, and their disclosure increased Seascape's attack surface.

Code Location:

It was possible to generate a stack trace by accessing unavailable content such as the following page:

https://beta-bsc.seascape.network/admin/reset

The extract from the server response is presented below:

Listing 1

```
TIMEZONE    America/New_York
HOSTNAME    [Redacted]
ethplatform CWS-ETH
bsc PancakeSwap
bscplatform pCWS-BNB
api https://beta-api.seascape.network/
nftAddress  0x7115ABcCa5f0702E177f172C1c14b3F686d6A63a
burning 0x3Cd60dEc3F8623B61537A8A681850c776709Ac5A
PASSWORD    [Redacted]
crownsAddress   0x168840Df293413A930d3D40baB6e1Cd8F406719D
USERNAME    beta_seascape
PHP_MEMORY_LIMIT    512M
PATH    /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/
    bin
```

```
14 default_lang    en-us
15 DATABASE    beta_eth_seascape
16 PWD /data/code
17 domain  beta
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Halborn recommends that the testing environment is either made accessible only to whitelisted IP addresses or that the debug feature is disabled. Additionally, Seascape should ensure that the identified password is not used elsewhere in their infrastructure, to prevent attackers from accessing unauthorized content.

Remediation Plan:

PARTIALLY SOLVED: The issue is not present on the production environment however users would be able to recover the sensitive information via the messages disclosed in the testing environment. It should be noted that the production environment is a replica of the testing one therefore the information would still be of use.

3.4 (HAL-04) OUTDATED THINKPHP VERSION - MEDIUM

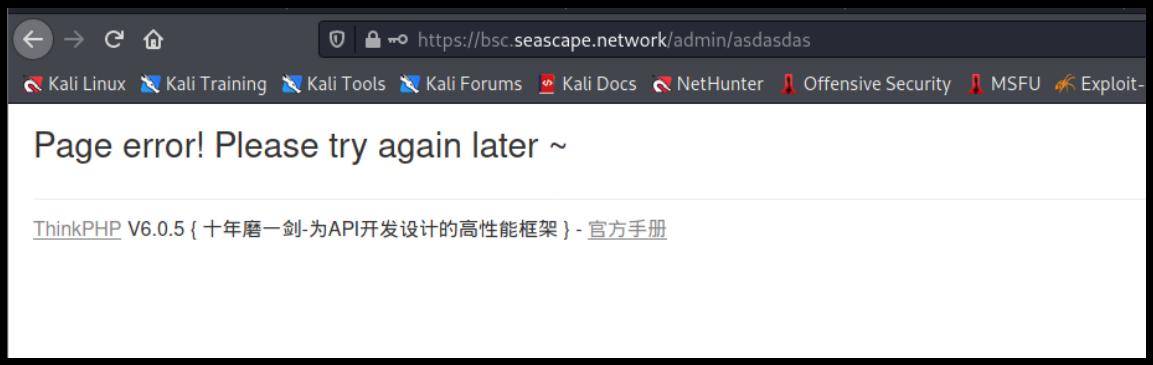
Description:

By observing the server response on nonexistent pages, it was possible to identify the exact ThinkPHP version number. The version in use by Seascape was version 6.0.5, which is outdated. At the time of writing this report, the most recent version is 6.0.11.

One critical vulnerability was identified in ThinkPHP version 6.0.8. However, it was not confirmed if this applied to the version in use by Seascape.
By using an outdated software version, Seascape might not benefit from features and security enhancements available on the new version of this framework.

Code Location:

The following screenshot shows the framework version being disclosed on the production environment:



Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Halborn recommends that a patch management policy is implemented to ensure that all software components are updated to the latest stable and secure versions at all times. This would minimize the window of opportunity that an attacker would have to exploit any vulnerabilities affecting these components.

For ThinkPHP, the latest stable version is version 6.0.11 - available at ThinkPHP GitHub.

Remediation Plan:

RISK ACCEPTED: The development team weighted the risk and time resources required to upgrade the version and are happy to continue using the current one.

3.5 (HAL-05) 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:

Title	Package	Severity
Denial of Service (DoS)	global-parent	HIGH
Prototype Pollution	ini	HIGH
Denial of Service (DoS)	is-svg	HIGH
Command Injection	lodash	HIGH
DoS	ssri	HIGH
Arbitrary File Creation	tar	HIGH
Prototype Pollution	y18n	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: The Seascape team performed an update of the dependencies in use.

3.6 (HAL-06) MISSING HTTP SECURITY HEADERS - LOW

Description:

The assessment revealed 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.
- Cache-Control - instructs the browser: if, how and which items should be stored in the temporary memory.
- 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:

RISK ACCEPTED: The development team informed that they will look into enabling the recommended headers.

3.7 (HAL-07) SESSION COOKIE WITHOUT SECURE AND HTTPONLY FLAGS SET - LOW

Description:

After logging to the CMS, the application server set a PHP cookie. Cookies are used by web applications to store information relating to the site within the user's browser. In particular, cookies typically contain session information that ties individual requests to the application with a specific user.

The following observations were made on the application's session cookie.

The HttpOnly flag was not set for the cookie used by the application, which could allow an attacker to impersonate users by stealing their sessions, should cross-site scripting vulnerabilities be discovered in the application in the future, potentially allowing them to impersonate the application's users. Additionally, without the secure flag, the cookie's contents could potentially traverse a clear text channel, which could result in an attacker gaining access to a user's session information.

The Secure flag is an attribute of cookies that instructs a browser whether a cookie can be transmitted over a clear text channel, such as HTTP. Cookies that are sent over a clear text channel can be intercepted by an attacker that is suitably positioned on the network. Once intercepted, the cookie's content can be read or altered to suit the attacker's needs. If the Secure flag is set, the browser will only send the cookie over an HTTPS connection. Due to the encrypted nature of HTTPS, similar attacks would no longer be possible.

The application only communicated over an encrypted HTTPS channel. Therefore, it was not possible for an attacker to create a condition in which the cookie would be transmitted over a clear text channel. However, security best practice dictates that cookies containing session tracking information should be protected with the Secure flag.

Code Location:

The Secure and HttpOnly flags are stipulated in the Set-Cookie header, as the value is assigned. As can be seen in the extract below, this flag was not present.

Listing 2

```
1  HTTP/1.1 200 OK
2  Date: Fri, 07 Jan 2022 15:26:34 GMT
3  Content-Type: application/json; charset=utf-8
4  Connection: close
5  Cf-Railgun: c4a0eb76c8 stream 0.000000 0210 57da
6  Set-Cookie: think_lang=en-us; path=/; domain=seascape.network
7  Set-Cookie: PHPSESSID=[Redacted]; path=/; domain=seascape.network
8  X-Powered-By: PHP/7.3.31
9  CF-Cache-Status: DYNAMIC
10 Expect-CT: max-age=604800, report-uri="https://report-uri.
       cloudflare.com/cdn-cgi/beacon/expect-ct"
11 Server: cloudflare
12 CF-RAY: 6c9e3485db8b71d8-LHR
13 alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400, h3
       -28=":443"; ma=86400, h3-27=":443"; ma=86400
14 Content-Length: 9630
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

The Secure flag and the HttpOnly flag should be set on all cookies containing sensitive data such as user session tokens. Once enabled, these flags will be visible in the web server's Set-Cookie headers.

The source code of the application can either be amended, or the configuration changes can be applied to the hosting server and/or the reverse proxy. The directives would depend on the server in use.

Remediation Plan:

RISK ACCEPTED: The Seascape team confirmed that they will amend the server side configuration to add the HttpOnly and secure flag to cookies.

3.8 (HAL-08) TLS MISCONFIGURATIONS - LOW

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.

```
Testing protocols via sockets except NPN+ALPN

SSLv2      not offered (OK)
SSLv3      not offered (OK)
TLS 1      offered (deprecated)
TLS 1.1    offered (deprecated)
TLS 1.2    offered (OK)
TLS 1.3    offered (OK): final
NPN/SPDY   not offered
ALPN/HTTP2 h2, http/1.1 (offered)

Testing cipher categories

NULL ciphers (no encryption)                      not offered (OK)
Anonymous NULL Ciphers (no authentication)        not offered (OK)
Export ciphers (w/o ADH+NULL)                     not offered (OK)
LOW: 64 Bit + DES, RC[2,4], MD5 (w/o export)      not offered (OK)
Triple DES Ciphers / IDEA                         offered
Obsoleted CBC ciphers (AES, ARIA etc.)            offered
Strong encryption (AEAD ciphers) with no FS       offered (OK)
Forward Secrecy strong encryption (AEAD ciphers)  offered (OK)
```

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:

ACKNOWLEDGED: The development team will restrict access with TLS v1.0 and TLS v1.1.

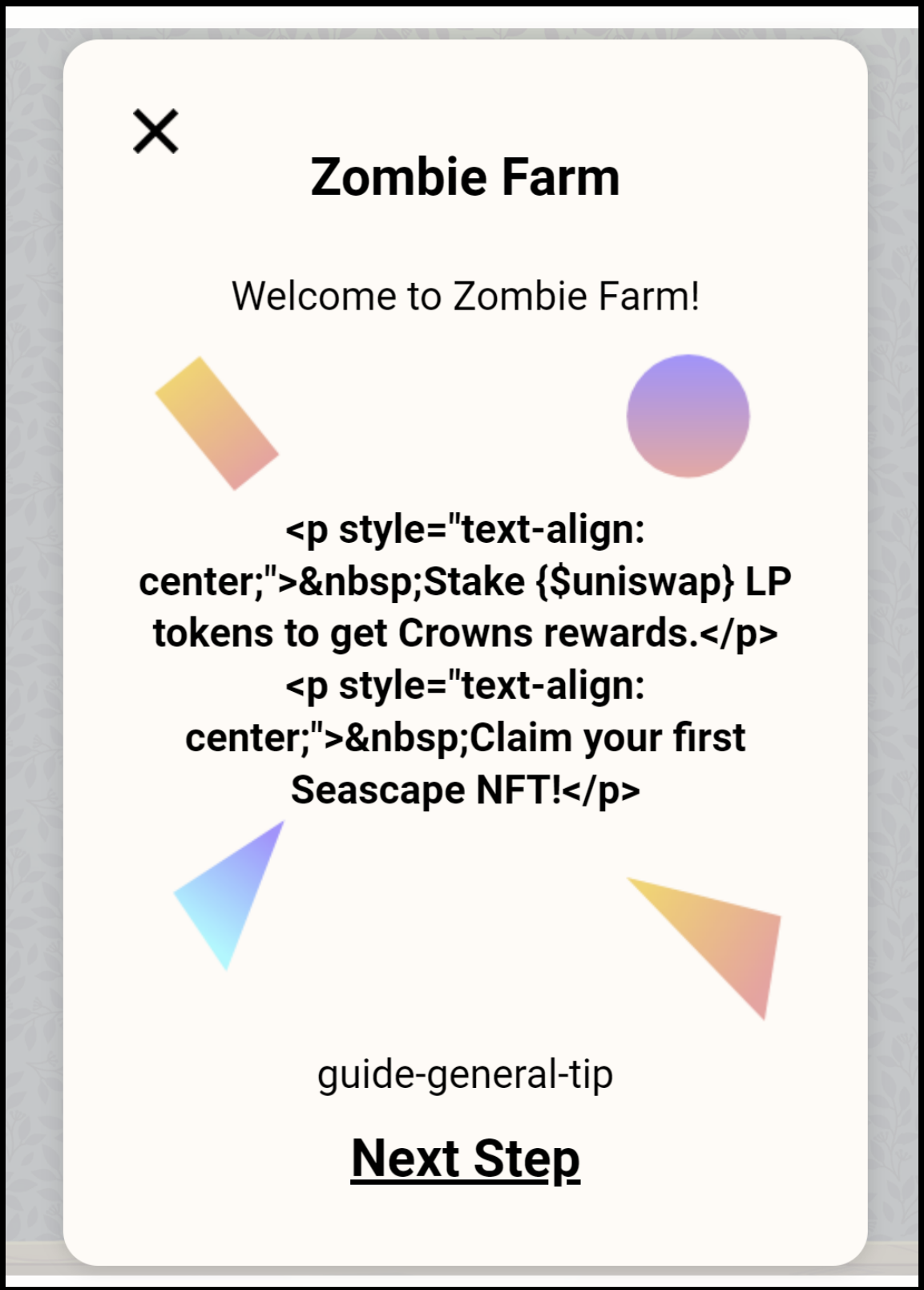
3.9 (HAL-09) FRONT-END DISPLAY ERRORS - INFORMATIONAL

Description:

Testing revealed that the application fronted did not display correctly formatted content when opening the Zombie minigame. Whilst this does not pose a direct security risk, it could lead an attacker to believe that the component is vulnerable to HTML injection or cross-site scripting (XSS) and expose the application to more targeted attacks.

Code Location:

The following screenshot displays the frontend error:


Zombie Farm
Welcome to Zombie Farm!
 Stake {$uniswap} LP tokens to get Crowns rewards.
 Claim your first Seascape NFT!
guide-general-tip
Next Step

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Halborn recommends that the frontend is reviewed to ensure that all components work as expected and display the correct data.

Remediation Plan:

SOLVED: This issue only appeared in the testing environment and was fixed.

3.10 (HAL-10) HARDCODED AND WEAK CREDENTIALS - INFORMATIONAL

Description:

It is worth mentioning that this issue was added as informational only due to the functionality not being in use. Should this functionality be used, its criticality would greatly increase.

While reviewing the backend application's source code, Halborn was able to identify a hardocded JSON Web Token (JWT) secret and a weak username and password combination. It was not possible to identify if the JWT was in use by the application. However, it should be noted that if this is used to generate JWT tokens that allow access to sensitive functionality, attackers would likely be able to generate forged JWT tokens and access this content.

Finally, credentials for a login page were identified in the source code. While the source code is stored in a private repository, this is considered bad practice and should be avoided. It should be noted that it was not possible to identify what additional privileges a logged-in user would gain.

Code Location:

The following code snippet was found within the application's source code and contained a weak JSON Web Token (JWT) secret. Attackers with access to a JWT would likely be able to bruteforce the secret key and generate their tokens. It was not possible to determine where and if these tokens were used.

Listing 3

```
1     // claim uid
2     private $uid;
3
4     // secrect
```

```
5     private $secrect = '[Redacted]';
6
7     // decode token
8     private $decodeToken;
```

The credentials for the user test01 were hardcoded in the source code and trivial to guess should an attacker initiate a bruteforce attack to the application. These credentials allowed to log in to the panel available at the following link.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Halborn recommends that Seascape development team investigate the use of the aforementioned secret and credentials and that these are changed to more secure values. Additionally, it should be noted that it is bad practice to store sensitive data in source files. Instead, environmental variables should be used to import secrets into the code.

Remediation Plan:

ACKNOWLEDGED: This JWT library is not in use; therefore it should be ignored.

3.11 (HAL-11) WEB SERVER BANNER DISCLOSURE - INFORMATIONAL

Description:

Halborn was able to recover technical information regarding the application server via the server response headers. The exact PHP version in use was disclosed to the users.

The recovery of this information could allow a malicious user to target attacks specifically designed for the application's technology stack.

Code Location:

The following server response sample shows the exact PHP version in use being disclosed.

Listing 4

```
1  HTTP/1.1 200 OK
2  Date: Thu, 06 Jan 2022 11:26:04 GMT
3  Content-Type: text/html; charset=utf-8
4  Connection: close
5  Cf-Railgun: d839551eae stream 0.000000 0200 57da
6  Set-Cookie: think_lang=en-us; path=/; domain=seascape.network
7  Vary: Accept-Encoding
8  X-Powered-By: PHP/7.3.31
9  set-cookie: PHPSESSID=45f76c76b3a2e31dd3a291a4a42111f8; path=/;
       domain=seascape.network
10 CF-Cache-Status: DYNAMIC
11 Expect-CT: max-age=604800, report-uri="https://report-uri.
       cloudflare.com/cdn-cgi/beacon/expect-ct"
12 Server: cloudflare
13 CF-RAY: 6c9496dfec2306ae-LHR
14 alt-svc: h3=":443"; ma=86400, h3-29=":443"; ma=86400, h3
       -28=":443"; ma=86400, h3-27=":443"; ma=86400
15 Content-Length: 51765
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

To prevent attackers from easily determining technical information about the web server in use, these HTTP response headers should be obfuscated or removed.

The following line should be amended in the php.ini file to prevent the disclosure of the PHP version in use:

Listing 5

```
1 expose_php = Off
```

Remediation Plan:

ACKNOWLEDGED: The Seascape team confirmed that they will remove the display of the exact PHP version.

THANK YOU FOR CHOOSING

// HALBORN