Playground Labs - Self-Custody

Node Security Audit

Prepared by: **Halborn**

Date of Engagement: **September 9th, 2022 - October 7th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	09/29/2022	Hossam Mohamed
0.2	Document Edits	09/29/2022	Hossam Mohamed
0.3	Draft Review	09/29/2022	Gabi Urrutia
1.0	Remediation Plan	11/17/2022	Hossam Mohamed
1.1	Remediation Plan Review	11/17/2022	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
Hossam Mohamed	Halborn	Hossam.Mohamed@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Playground Labs engaged Halborn to conduct a security assessment on their Self-Custody on September 9th, 2022 and ending October 7th, 2022. Playground Labs developed the Self-Custody node that has API Interface and interacts with external chains.

1.2 AUDIT SUMMARY

The team at Halborn was provided four weeks for the engagement and assigned a full-time security engineer to audit the security of the Self-Custody in scope. The security engineer is a blockchain and smart contract security expert with advanced penetration testing and smart contract hacking skills, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Identify potential security issues within the Self-Custody

In summary, Halborn identified multiple security issues are mostly related to the API of the Self-Custody, successful exploitation of these issues will directly lead to funds/financial loss.

Halborn's findings, descriptions and remediations have been redacted at the request of Playground Labs.

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 review was focused on the Python codebase in the halborn-audit branch in the Kapital-Self-Custody-Node repository.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
3	0	2	4	1

LIKELIHOOD

IMPACT

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

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
API - AUTHENTICATED INTERNAL USERS CAN CREATE ADMIN USER	Critical	SOLVED - 09/22/2022
API -AUTHENTICATED INTERNAL USER CAN DELETE ANY OTHER USER	Critical	SOLVED - 09/22/2022
ARCHITECTURE - MISSING INTERNAL MFA CONTROLS	Critical	SOLVED - 11/07/2022
CRYPTOGRAPHY - PLAIN TEXT GAME PASSWORDS	Informational	ACKNOWLEDGED
ARCHITECTURE - MISSING RATE LIMIT	Medium	SOLVED - 11/09/2022
API - USER ENUMERATION	Medium	SOLVED - 11/07/2022
LOG MANAGEMENT - LOG INJECTION	Low	SOLVED - 11/07/2022
ARCHITECTURE - WEAK PASSWORD POLICY	Low	SOLVED - 11/07/2022
ARCHITECTURE - MISSING TWO FACTOR AUTHENTICATION	Low	SOLVED - 11/07/2022
HTTP - MISCONFIGURED CORS	Low	SOLVED - 11/07/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) API - AUTHENTICATED INTERNAL USERS CAN CREATE ADMIN USER - CRITICAL

Description:

Inside the user_router within the create_user function which is responsible for creating new users, it was noted that the function requires only an authenticated user without any special permissions.
The user parameter within create_user is based on the UserCreate schema which includes user_role, the attacker can provide any role within the request and create a user with any role, including the admin role.

Risk Level:

Likelihood - 5
Impact - 5

Recommendation:

It is recommended to restrict user creation to the admin role only and remove user_role from UserCreate request schema.

Remediation Plan:

SOLVED: The Playground Labs team solved the issue by adding the permission checking class.

6e0368d70f706452d4bc940c08beec485ebd6093

3.2 (HAL-02) API - AUTHENTICATED INTERNAL USERS CAN DELETE ANY OTHER USER - CRITICAL

Description:

Within the user_router inside the delete_user function which is responsible for deleting any user on the system, it was noted that the function requires only an authenticated user without any special permissions to delete any user on the system, including admin users.

Risk Level:

Likelihood - 5
Impact - 5

Recommendation:

It is recommended to restrict the delete user function to the admin role only.

Remediation Plan:

SOLVED: The Playground Labs team solved the issue by adding the permissions check class.

6e0368d70f706452d4bc940c08beec485ebd6093

3.3 (HAL-03) ARCHITECTURE - MISSING INTERNAL MFA CONTROLS - CRITICAL

Description:

The authorization scheme within the API was observed, which allows an attacker to interact with the Node and perform privileged actions such as axie_claim or even sign transactions and interact with wallets by only sending the wallet address as an HTTP parameter.
Relying on addr just to interact with a wallet object and retrieve the private key for that wallet is not enough, as it does not provide an authentication layer for wallets. Multiple critical functions were found for this issue, such as wallet_web3_sign and wallet_web3_interaction.

Risk Level:

Likelihood - 5
Impact - 5

Recommendation:

Only privileged accounts (Admin) should be able to interact with wallets for privileged operations.

Remediation Plan:

SOLVED: The Playground Labs team solved the issue by adding the OTP Password in critical functions.

815fc73bc5f0dac4c7cb2544bcd249f45aa08135

3.4 (HAL-04) CRYPTOGRAPHY - PLAIN TEXT GAME PASSWORDS - INFORMATIONAL

Description:

The Self-Custody, stores the wallet related data and private keys within a database,
however, it was observed that the wallet password was stored in a plain text format without any hashing or encryption, meanwhile, the response scheme was observed to return the password in the HTTP response.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended to remove the password from the HTTP response and not store the wallet password in plain text within the database and use that password as a second verification or authentication method within wallet related operations.

Remediation Plan:

ACKNOWLEDGED: The Playground Labs team acknowledged this finding and the severity has been changed to informational.

3.5 (HAL-05) ARCHITECTURE - MISSING RATE LIMIT - MEDIUM

Description:

The Self-Custody, authentication system and API implementation in general were found to lack the rate limit mechanism, which in turn exposes the Self-Custody, to brute force and denial of service attacks at the level of the application logic.

Risk Level:

Likelihood - 3
Impact - 4

Recommendation:

It is recommended to implement a rate limit mechanism and user lockout policy to defend against brute force attacks.

Remediation Plan:

SOLVED: The Playground Labs team solved the issue by adding an application level rate limit.

de794da9109a1f9fe8c1cdf74c9dba9511148834

3.6 (HAL-06) API - USER ENUMERATION - MEDIUM

Description:

A user enumeration vulnerability was observed within the login endpoint, a malicious actor can brute force valid usernames to find valid users on the system and use them in other attack scenarios.

Recommendation:

It is recommended to remove any response from the login process indicating that the user is valid or invalid and return a generic error message such as “invalid username or password.”.

Remediation Plan:

SOLVED: The Playground Labs team solved the issue in commit 815fc73bc5f0dac4c7cb2544bcd249f45aa08135

3.7 (HAL-07) LOG MANAGEMENT - LOG INJECTION - LOW

Description:

The HTTP Server within Self-Custody, was observed to be logging HTTP requests at an early stage with the HTTP body. The attacker can fill the disk with logs simply by sending simple HTTP requests with large chunks of data within the request body to the Custody HTTP Server.

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

It is recommended to log the request body only when debugging is enabled.

Remediation Plan:

SOLVED: The Playground Labs team solved the issue in commit fd692d0763b9696c24f036abbb25ec6c78dabb89

3.8 (HAL-08) ARCHITECTURE - WEAK PASSWORD POLICY - LOW

Description:

During the user creation process, it was observed that a weak password policy was applied during the creation of the user's password hash, as it is only checks if the password is longer than 8 characters, which is not enough as a password strength baseline.

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

It is recommended to enforce a stricter password policy that ensures that the user's password contains uppercase, lowercase, non-alphanumeric characters, and is longer than 8 characters.

Remediation Plan:

SOLVED: The Playground Labs team solved the issue in commit 3f325accdbc5824f147a986f35606ef049224e2f

3.9 (HAL-09) ARCHITECTURE - MISSING TWO FACTOR AUTHENTICATION - LOW

Description:

It was noted that the authentication system within the Self-Custody, lacks a two-factor authentication mechanism. In case of credentials theft, the attacker can perform privileged operations such as transactions and interact with other critical functions.

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

Two-factor authentication is recommended to reduce the impact of credential theft attacks.

Remediation Plan:

SOLVED: The Playground Labs team solved the issue by adding the OTP Password in critical functions. ebda4a69fee55bf67501eede7777a5d06fc77a4a

3.10 (HAL-10) HTTP - MISCONFIGURED CORS - LOW

Description:

Cross-Origin Resource Sharing (CORS) is an HTTP header-based mechanism that allows a server to indicate any origin (domain, scheme, or port) other than its own from which a browser should allow resource loading. Custody HTTP configurations are hardcoded for CORS, leaving custody vulnerable to CORS attacks.

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

CORS configurations should be loaded from environment variables.

Remediation Plan:

SOLVED: The Playground Labs team solved the issue by configuring the origins. 7470afbec3cec689bf0eef874763c1668f4da306

THANK YOU FOR CHOOSING

// HALBORN