QuillAudits
Web3 Security

AUDIT REPORT

September, 2024

For

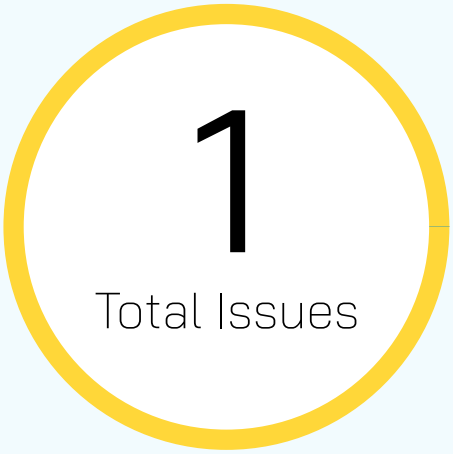
MATCHAIN

Executive Summary

Project name	Matchain
Overview	Matchain is a fork of Uniswap V2, It maintains the core structure and functionality of Uniswap V2, including the automated market maker (AMM) system, liquidity provision, token swaps, and flash swap capabilities. The main contracts are MSwapFactory for pair creation and management, MSwapPair for handling liquidity pairs and swaps, and MSwapERC20 for the implementation of liquidity tokens.
Timeline	26th August 2024 - 2nd September 2024
Second Review	NA
Method	Manual Review, Functional Testing, Automated Testing, etc. All the raised flags were manually reviewed and re-tested to identify any false positives.
Blockchain	BSC
Audit Scope	The scope of this audit was to analyse the Matchain for quality, security, and correctness.
Source code	https://github.com/MatchSwap/MSwap-contract/tree/main
Contracts In Scope	Factory.sol Router.sol WBNB.sol
Branch	Main
Fixed In	NA

Mainnet address	NA

Number of Issues per Severity



High	0 (0.00%)
Medium	1 (100.00%)
Low	0 (0.00%)
Informational	0 (0.00%)

Severity

Issues	High	Medium	Low	Informational
Open	0	0	0	0
Resolved	0	0	0	0
Acknowledged	0	**1**	0	0
Partially Resolved	0	0	0	0

Table of Content

Techniques and Methods

Throughout the pentest of Hurupay Mobile Wallet, care was taken to ensure:

- Information gathering – Using OSINT tools information concerning the web architecture,
- Information leakage, web service integration, and gathering other associated information
- related to web server & web services.
- Using Automated tools approach for Pentest like Nessus, Acunetix etc.
- Platform testing and configuration
- Error handling and data validation testing
- Encryption-related protection testing
- Client-side and business logic testing

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools And Platforms Used For Audit

Remix IDE, Truffle, Solhint, Mythril, Slither, Solidity Statistic Analysis.

Types of Issues

Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below

High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Issue Status

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

Security vulnerabilities identified that must be resolved and are currently unresolved.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/ impact of the security issue, but are not completely resolved.

Medium Severity Issues

A malicious user can grief and DOS a user while he is t remove Liquidity

Acknowledged

Description

The removeLiquidityWithPermit() and removeLiquidityETHWithPermit() use permit function to give approve to router contract for the LP tokens and removal of liquidity in the same transaction rather than approving the LP token manually from contract address.
1.User signs their LP tokens and provides the signature(v,r,s) to contract as part of approving token using permit.

2.When a user calls removeLiquidityWithPermit or removeLiquidityETHWithPermit function the malicious user see's the transaction in mempool and take the signature of user and calls the permit function himself before the user's transaction gets executed.

3.Since the signature provided by malicious user is correct the transaction get's successful and increases the nonce.
However, when user's transaction gets executed it would fail due to increase of nonce.

Recommendation

try IERC20Permit(pair).permit(msg.sender, address(this), amount, deadline, v, r, s) {} catch {}

No Issues Found

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Closing Summary

In this report, we have considered the security of the Matchain contracts. We performed our audit according to the procedure described above High, Medium, Low and informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in Matchain smart contracts. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of Matchain smart contract. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of the Matchain Team to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem

6+ Years of Expertise	**1M+** Lines of Code
$30B+ Secured in Digital Assets	**1K+** Projects

Follow Our Journey

AUDIT REPORT

September, 2024

For

MATCHAIN

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

mailto:audits@quillhash.com