QuillAudits

Audit Report
March, 2024

For

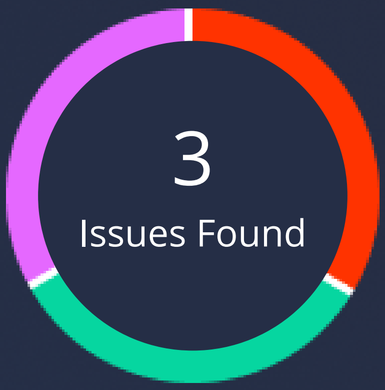
Vouch

Table of Content

Executive Summary

Project Name	Vouch
Overview	Vouch is designed to enhance the efficiency and security of escrow transactions in gig-based marketplaces using Ethereum blockchain technology. Aimed at automating the escrow process, the contract serves as a decentralized intermediary that securely holds funds until predefined conditions are met. This approach minimizes trust issues between parties in gig transactions, providing a more reliable and transparent method for handling payments.
Timeline	5th March 2024 - 29th March 2024
Updated Code Received	21st March 2024
Second Review	28th March 2024 - 29th March 2024
Method	Manual Review, Functional Testing, Automated Testing, etc. All the raised flags were manually reviewed and re-tested to identify any false positives.
Audit Scope	The scope of this audit was to analyze the Vouch Contracts for quality, security, and correctness.
Source Code	*https://github.com/vouch-app/vouch-smart-contract*
Branch	main
Commit Hash	5d8ccc2b6b8857de632f14f8edce59dce7e26263
Fixed In	*https://github.com/vouch-app/vouch-smart-contract/commit/b34a609b2f7c6361033184c615de8905d4d9eaff*

Number of Security Issues per Severity



🟥 High	🟨 Medium
🟩 Low	🟪 Informational

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

Checked Vulnerabilities

- ✓ Re-entrancy
- ✓ Timestamp Dependence
- ✓ Gas Limit and Loops
- ✓ DoS with Block Gas Limit
- ✓ Transaction-Ordering Dependence
- ✓ Use of tx.origin
- ✓ Exception disorder
- ✓ Gasless send
- ✓ Balance equality
- ✓ Byte array
- ✓ Transfer forwards all gas
- ✓ ERC20 API violation
- ✓ Malicious libraries
- ✓ Compiler version not fixed
- ✓ Redundant fallback function
- ✓ Send instead of transfer
- ✓ Style guide violation
- ✓ Unchecked external call
- ✓ Unchecked math
- ✓ Unsafe type inference
- ✓ Implicit visibility level

Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

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

Hardhat, Foundry.

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.

Types of Issues

Open

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

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

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.

High Severity Issues

1. Malicious requester can make the gig done Free

Path

VouchV1.sol

Function

Create function

Description

The **create** function in **VouchV1.sol** allows a malicious requester to exploit a logic flaw and receive an unintended refund.

The requester creates a gig where the voucher address is set to their own address. After the gig is completed, the requester initiates a dispute, changing the gig state to **REQUESTER_DISPUTED**. The requester has three options to force a resolution that benefits them:

- **Vouchee Concedes:** If the vouchee concedes the dispute, the requester receives a full refund (including the platform fee) even though they haven't paid the fee.
- **Vouchee Disputes:** If the vouchee disputes, the arbitration process requires a p from the voucher. However, a flaw allows the requester (~voucher) to set p = 0, essentially bypassing the deposit requirement and receiving a full refund if the vouchee doesn't respond within a specific timeframe.
- **Vouchee Inaction:** If the vouchee doesn't respond to the dispute, the Requester can recover the funds after the expiration of disputed time.

Due to the above issue, it leads to a loss of revenue for the platform due to fraudulent refunds. Discourages legitimate vouchees from participating in the platform due to the risk of unfair disputes.Potential loss of user trust in the platform's dispute resolution process.

Recommendation

Modify the create function to prevent setting the voucher address to the requester's address.

Status

Resolved

Resolved in Commit Hash

7783ffadc2d5454ee986ed66956a723571e3f9b2

Low Severity Issues

2. Incorrect data emission

Function

arbitrage, s_specialist

Description

In the function **arbitrage, s_specialist** address get emitted during the emission of the **Gig__Completed** event while the function is initiated by the voucher so the completer of the gig should be voucher address instead of **s_specialist**. Because of this incorrect data emission, dApps will show incorrect data and assume that voucher wasn't active to perform the arbitrage of the gig.

Recommendation

Replace the **s_specialist** address with the gig's voucher address in the **Gig__Completed** event.

Status

Resolved

Resolved in Commit Hash

224d91737f3204d4ed5a701836182de00c3e8ec6

Informational Issues

3. Throughout Codebase Inadequate Use of Safe Transfer Functions

Description

A pervasive security concern is identified throughout the codebase: the inconsistent use of the **safeTransfer** and **safeTransferFrom** functions from the OpenZeppelin SafeERC20 library.

The majority of instances involving stablecoin transfers do not leverage the recommended **safeTransfer** and **safeTransferFrom** functions. These functions offer crucial advantages:

- **Reentrancy Protection:** They prevent reentrancy attacks, a common exploit where a malicious function can call itself recursively to steal funds during a transfer. However, In the codebase **nonReentrant** modifier has been used so reentrancy is not an issue.
- **Error Handling:** They automatically revert the transaction in case of failure, providing valuable feedback and preventing unexpected behaviour.

The absence of these safeguards exposes the smart contract to Unexpected Transaction Reversion.

Recommendation

To mitigate these risks, it is imperative to systematically review and refactor the codebase to ensure that all payout token transfers utilize the **safeTransfer** and **safeTransferFrom** functions provided by the OpenZeppelin SafeERC20 library. This comprehensive update will significantly enhance the security and reliability of the smart contract by ensuring proper error handling, and safeguarding transferred funds.

Status

Resolved

Resolved in Commit Hash

https://github.com/vouch-app/vouch-smart-contract/commit/b34a609b2f7c6361033184c615de8905d4d9eaff

Sample code from our codebase:

```
bool success = payoutToken_.transferFrom(
            parties_.requester,
            address(this),
            payoutAmount_
);
if (!success) {
            revert Gig__TransferFailed();
}
```

Sample code from Openzeppelin's SafeERC20.sol:

```
function _callOptionalReturn(IERC20 token, bytes memory data) private {
            bytes memory returndata = address(token).functionCall(data);
            if (returndata.length != 0 && !abi.decode(returndata, (bool))) {
                        revert SafeERC20FailedOperation(address(token));
            }
}
```

Links

- **VouchV1.sol**
  (https://github.com/vouch-app/vouch-smart-contract/blob/main/contracts/VouchV1.sol#L228-L236)
- **SafeERC20.sol**
  (https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/utils/SafeERC20.sol#L140-L155)

We've acknowledged this proposal but decided to not move forward, as the recommendation is being enforced through a similar solution (manual check vs using the library).

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 Vouch codebase. We performed our audit according to the procedure described above.

One issue of High severity, one issue of low severity & one issue of informational severity were found. In the End Vouch Team, resolved all Issues.

Disclaimer

QuillAudits Smart contract security audit provides services to help identify and mitigate potential security risks in Vouch 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 Vouch smart contracts. 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 Vouch 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.

1000+
Audits Completed

$30B
Secured

1M
Lines of Code Audited

Follow Our Journey

Audit Report
March, 2024

For

Vouch

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com