

Smart contract audit

EthGild

Content

Project description	3
Executive summary	3
Reviews	3
Technical analysis and findings	5
Security findings	6
**** Critical	6
*** High	6
** Medium	6
M01 - Incorrect validation of total shares	6
* Low	7
Informational	7
Approach and methodology	8



Project description

This repository implements a decentralized financial system that leverages receipt-based vaults to manage tokenized assets. It provides mechanisms for minting ERC20 shares and ERC1155 receipt NFTs, enabling transparent tracking of deposits and withdrawals. The system supports dynamic share-to-asset ratios, off-chain asset integration, and price oracle-based calculations. It incorporates features like certification, asset confiscation, and fine-grained access control to ensure compliance and trust. The design emphasizes flexibility for issuers to maintain asset pegs and profitability while offering robust auditability and user protections.

Executive summary

Type	Vault
Languages	Solidity
Methods	Architecture Review, Manual Review, Unit Testing, Functional Testing, Automated Review
Documentation	README.md
Repository	https://github.com/gildlab/ethgild

Reviews

Date	Commit
30/07/25	81d3021ff0d77d9fdbcdb4a20154eaea39f0bf095
31/07/25	b44855c2f0f4e31e26649ad1902c6633f9395c2e

Scope

Contracts
src/concrete/authorize/OffchainAssetReceiptVaultPaymentMintAuthorizerV1.sol
src/concrete/authorize/OffchainAssetReceiptVaultAuthorizerV1.sol



Technical analysis and findings

Critical	0
High	0
Medium	0
Low	0
Informational	0



Security findings

**** Critical

No critical severity issue found.

*** High

No high severity issue found.

** Medium

M01 - Incorrect validation of total shares

Impact	Medium
Likelihood	Medium

The *authorize()* function of the *OffchainAssetReceiptVaultPaymentMintAuthorizerV1* contract is also used for validation during the deposit process.

The step-by-step call stack during a deposit looks like this:

1. OffchainAssetReceiptVault.deposit() → [ReceiptVault.deposit()]
2. ReceiptVault._deposit()
3. ReceiptVault._beforeDeposit()
 - 3.1. OffchainAssetReceiptVault._beforeDeposit()
4. ReceiptVault._mint()
 - 4.1. *ERC20._mint()* → increases totalSupply !
5. ReceiptVault._afterDeposit()
 - 5.1. OffchainAssetReceiptVault._afterDeposit()
 - 5.2. OffchainAssetReceiptVaultPaymentMintAuthorizerV1.authorize() → totalSupply is already increased.

As shown in the call chain, at step 4.1. the totalSupply is already increased. Therefore, at step 5.2., the call to *IERC20(IReceiptVault).totalSupply()* will return the updated value. This causes the check at line L236 to be inaccurate.

Path: src/concrete/authorize/OffchainAssetReceiptVaultPaymentMintAuthorizerV1.sol

Recommendation: Update the validation like the following:

```
require(IERC20(IReceiptVault).totalSupply() <= sMaxSharesSupply,  
"MaxSharesSupplyExceeded").
```



Found in: 81d3021

Status: Fixed (b44855)

* Low

No Low severity issue found.

Informational




No Informational severity issue found.



Approach and methodology




To establish a uniform evaluation, we define the following terminology in accordance with the OWASP Risk Rating

Methodology:

	Likelihood Indicates the probability of a specific vulnerability being discovered and exploited in real-world scenarios
	Impact Measures the technical loss and business repercussions resulting from a successful attack
	Severity Reflects the comprehensive magnitude of the risk, combining both the probability of occurrence (likelihood) and the extent of potential consequences (impact)

Likelihood and impact are divided into three levels: High H, Medium M, and Low L. The severity of a risk is a blend of these two factors, leading to its classification into one of four tiers: Critical, High, Medium, or Low.

When we identify an issue, our approach may include deploying contracts on our private testnet for validation through testing. Where necessary, we might also create a Proof of Concept PoC to demonstrate potential exploitability. In particular, we perform the audit according to the following procedure:

	Advanced DeFi Scrutiny We further review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs
	Semantic Consistency Checks We then manually check the logic of implemented smart contracts and compare with the description in the white paper.
	Security Analysis The process begins with a comprehensive examination of the system to gain a deep understanding of its internal mechanisms, identifying any irregularities and potential weak spots.



