4th05

vVv

Security Review Report

17 November 2024

Security Review Report

4th05

November 17, 2024

Table of Contents

Protocol Summary

vVv facilitates both seed & launchpad deals. This contest focusses on the contracts required for running these investments and distributing the tokens to the investors.

Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where I try to find as many vulnerabilities as possible. I can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

Risk Classification

		Impact		
		High	Medium	Low
Likelihood	High	H	H/M	M
	Medium	H/M	M	M/L
	Low	M	M/L	L

Overview

Contest platform	Sherlock
LOC	279
Language	Solidity
Commit	1791f41b310489aaa66de349ef1b9e4bd331f14b
Previous audits	n/a

Scope

- VVVVCInvestmentLedger.sol
- VVVVCTokenDistributor.sol

Issues found

Severity	Number of issues found
High	1
Medium	0
Low	0
Info	0

Findings

[H1] `VVVVCTokenDistributor::claim` replay/front-running/reorg attack

Summary

In `VVVVCTokenDistributor.sol` a valid call to the `VVVVCTokenDistributor::claim` function can be replayed by a malicious actor to stole the remaining funds from the wallets (the `VVVVCTokenDistributor::projectTokenProxyWallets`) of the OG caller bypassing all the requirements.

Relevant GitHub Links

https://github.com/sherlock-audit/2024-11-vvv-exchange-update/blob/main/vvv-platform-smart-contracts/contracts/vc/VVVVCTokenDistributor.sol#L42

https://github.com/sherlock-audit/2024-11-vvv-exchange-update/blob/main/vvv-platform-smart-contracts/contracts/vc/VVVVCTokenDistributor.sol#L51

Root Cause

To prevent replay attack of the `VVVVCTokenDistributor::claim` function the `uint256 nonce` has been added to the `VVVVCTokenDistributor::ClaimParams struct::ClaimParams`.

```
    struct ClaimParams {
        address kycAddress;
        address projectTokenAddress;
        address[] projectTokenProxyWallets;
        uint256[] tokenAmountsToClaim;
        uint256 nonce;
        uint256 deadline;
        bytes signature;
    }
```

This `nonce` is incremented by one off-chain before calling the `claim` function. However, this value, as of all the other input paramenters can be seen in a blockchain explorer, like `etherscan` (according to the chain this could be `avascan`,`bscscan`,`arbiscan`,`basescan`,`statescan`) by a malicious actor that can retrieve them by decoding the calldata. Since there are no requirements in place for the actual `msg.sender`, these parameters will satisfy all of them allowing, this way, malicious actors to stole funds by either replay the call function to `VVVVCTokenDistributor::claim` using these data but with a `nonce` value incremented by one (also the `ClaimParams.tokenAmountsToClaim` value could be changed if needed), or by front running the OG caller by paying more fees or by exploiting a `blockchain reorg` without the need (in these last 2 cases) to change any data.

```
function claim(ClaimParams memory _params) public {
        if (claimIsPaused) {
            revert ClaimIsPaused();
        }

        if (_params.projectTokenProxyWallets.length != _params.
            tokenAmountsToClaim.length) {
            revert ArrayLengthMismatch();
        }

        if (_params.nonce <= nonces[_params.kycAddress]) {
            revert InvalidNonce();
        }

        if (!_isSignatureValid(_params)) {
            revert InvalidSignature();
        }

        // update nonce
        nonces[_params.kycAddress] = _params.nonce;

        // define token to transfer
        IERC20 projectToken = IERC20(_params.projectTokenAddress);

        // transfer tokens from each wallet to the caller
        for (uint256 i = 0; i < _params.projectTokenProxyWallets.length
            ; i++) {
```

```
            projectToken.safeTransferFrom(
                _params.projectTokenProxyWallets[i],
                msg.sender,
                _params.tokenAmountsToClaim[i]
            );
        }

        emit VCClaim(
            _params.kycAddress,
            _params.projectTokenAddress,
            _params.projectTokenProxyWallets,
            _params.tokenAmountsToClaim,
            _params.nonce
        );
    }
```

Internal pre-conditions

A user has an amount of tokens in the `VVVVCTokenDistributor::projectTokenProxyWallets` and wants to withdraw a part of the funds by calling the `VVVVCTokenDistributor::claim`.

External pre-conditions

A malicious actor sees the transaction on the block explorer and retrieve all the input data used by the OG caller and use them just changing the `nonce` value (that has to be incremented by one) to claim/steal, in this way, other funds.

Impact

The user (OG caller) can potentially loose all his funds in `VVVVCTokenDistributor::projectTokenProxyWallets` which will be transfered to the attacker `address` as the new `msg.sender`.

```
        for (uint256 i = 0; i < _params.projectTokenProxyWallets.length; i++) {
            projectToken.safeTransferFrom(
                _params.projectTokenProxyWallets[i],
                msg.sender,
                _params.tokenAmountsToClaim[i]
            );
        }
```

Mitigation

Implement a requirement/check based on the `msg.sender` to prevent this kind of attack to happen.