SHERLOCK

Security Review For
vVv

Public contest prepared for: **vVv**
Lead Security Expert: **bughuntoor**
Date Audited: **November 14 - November 17, 2024**

Introduction

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

Scope

Repository: vvvdevs/vvv-platform-smart-contracts

Branch: main

Audited Commit: 29fdceaeed9a4174039b66d85a5d4ce5d0ed14bf

Final Commit: ea46d90c4019432f342c1eacc13b4f9def00f41e

For the detailed scope, see the contest details.

Findings

Each issue has an assigned severity:

- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.

Issues found

High	Medium
1	0

Issues not fixed or acknowledged

High	Medium
0	0

Security experts who found valid issues

rsam_eth
bughuntoor
wildflowerzx
y4y
future2_22
gr8tree
justAWanderKid
X12
TessKimy
prgzro
vladi319
udo
irresponsible
m4k2
redbeans
Waydou
0xNirix
Japy69
0xAadi
X0sauce
jsmi
0xShahilHussain
0xlemon
dobrevaleri
h2134
Aamirusmani1552
eeyore
ami
oot2k
mgf15
Atharv
oxwhite
Motomoto
MohammadX2049
plairfx
DeltaXV
hunter_w3b
matejdb
DharkArtz
shaflow01
nour99
BengalCatBalu
Weed0607
4th05
0xnbvc
whitehair0330
0xjarix
0xmujahid002
Le_Rems
neko_nyaa
merlinboii
imkapadia
37H3RN17Y2
merlin
pashap9990
Chonkov
novaman33
Ragnarok
056Security
hals
PNS
cawfree
ljj
Galturok
mgnfy.view
pepocpeter
0xaxaxa
dany.armstrong90
POB
covey0x07
0xLeveler
SUPERMAN_I4G
Pro_King
Adotsam

Issue H-1: Anyone can call `VVVVCTokenDistribution::claim` function by utilizing `ClaimParams` signed by the `signer`

Source: https://github.com/sherlock-audit/2024-11-vvv-exchange-update-judging/issues/187

Found by

056Security, 0xAadi, 0xLeveler, 0xNirix, 0xShahilHussain, 0xaxaxa, 0xjarix, 0xlemon, 0xmujahid002, 0xnbvc, 37H3RN17Y2, 4th05, Aamirusmani1552, Adotsam, Atharv, BengalCatBalu, Chonkov, DeltaXV, DharkArtz, Galturok, Japy69, Le_Rems, MohammadX2049, Motomoto, PNS, POB, Pro_King, Ragnarok, SUPERMAN_I4G, TessKimy, Waydou, Weed0607, X0sauce, X12, ami, bughuntoor, cawfree, covey0x07, dany.armstrong90, dobrevaleri, eeyore, future2_22, gr8tree, h2134, hals, hunter_w3b, imkapadia, irresponsible, jsmi, justAWanderKid, ljj, m4k2, matejdb, merlin, merlinboii, mgf15, mgnfy.view, neko_nyaa, nour99, novaman33, oot2k, oxwhite, pashap9990, pepocpeter, plairfx, prgzro, redbeans, rsam_eth, shaflow01, udo, vladi319, whitehair0330, wildflowerzx, y4y

Summary

Legimate users can claim their reward with the `ClaimParas` signed by the signer. However, The `VVVVCTokenDistribution::claim` function just checks the validity of the signed `params` and doesn't check the caller is the right claimer. Hence anyone can claim the legimate users' rewards by front-running the signed `ClaimParams` and because of the nonce increasement legimate users can't claim their reward.

Root Cause

https://github.com/sherlock-audit/2024-11-vvv-exchange-update/blob/1791f41b310489aaa66de349ef1b9e4bd331f14b/vvv-platform-smart-contracts/contracts/vc/VVVVCTokenDistributor.sol#L133

Internal pre-conditions

N/A

External pre-conditions

Legimate users call `claim` function with signed params.

Attack Path

The `VVVVCTokenDistribution::claim` function is as follows:

```
File: VVVVCTokenDistributor.sol
106: function claim(ClaimParams memory _params) public {
107:         if (claimIsPaused) {
108:             revert ClaimIsPaused();
109:         }
110:
111:         if (_params.projectTokenProxyWallets.length !=
↪  _params.tokenAmountsToClaim.length) {
112:             revert ArrayLengthMismatch();
113:         }
114:
115:         if (_params.nonce <= nonces[_params.kycAddress]) {
116:             revert InvalidNonce();
117:         }
118:
119:         if (!_isSignatureValid(_params)) {
120:             revert InvalidSignature();
121:         }
122:
123:         // update nonce
124:         nonces[_params.kycAddress] = _params.nonce;
125:
126:         // define token to transfer
127:         IERC20 projectToken = IERC20(_params.projectTokenAddress);
128:
129:         // transfer tokens from each wallet to the caller
130:         for (uint256 i = 0; i < _params.projectTokenProxyWallets.length; i++) {
131:             projectToken.safeTransferFrom(
132:                 _params.projectTokenProxyWallets[i],
133:                 msg.sender,
134:                 _params.tokenAmountsToClaim[i]
135:             );
136:         }
137:
138:         emit VCClaim(
139:             _params.kycAddress,
140:             _params.projectTokenAddress,
141:             _params.projectTokenProxyWallets,
142:             _params.tokenAmountsToClaim,
143:             _params.nonce
144:         );
145:     }
```

At L119, the function check the validity of the input params.

The `_isSignatureValid` function is as follows:

```
File: VVVVCTokenDistributor.sol
157: function _isSignatureValid(ClaimParams memory _params) private view returns
↪  (bool) {
158:         bytes32 digest = keccak256(
159:             abi.encodePacked(
160:                 "\x19\x01",
161:                 DOMAIN_SEPARATOR,
162:                 keccak256(
163:                     abi.encode(
164:                         CLAIM_TYPEHASH,
165:                         _params.kycAddress,
166:                         _params.projectTokenAddress,
167:                         _params.projectTokenProxyWallets,
168:                         _params.tokenAmountsToClaim,
169:                         _params.nonce,
170:                         _params.deadline
171:                     )
172:                 )
173:             )
174:         );
175:
176:         address recoveredAddress = ECDSA.recover(digest, _params.signature);
177:
178:         bool isSigner = recoveredAddress == signer;
179:         bool isExpired = block.timestamp > _params.deadline;
180:         return isSigner && !isExpired;
181:     }
```

The function just checks if the signature is signed by the signer. The `VVVVCTokenDistribution::claim` function doesn't check if the `msg.sender` is the `kycAddress` and this leads to anyone can claim legimate `kycAddress` reward by front-running the params. Besides at L124, it increases the nonce of the `kycAddress` to prevent the double claim, hence the legimate `claimer` can't claim their rewards.

Though, front-running maybe hard on L2 but this leads to loss of fund to users and it will be deployed on `Ethereum` also.

Impact

Anyone can claim other legimate users' rewards and the legimate users can't claim their rewards.

PoC

Mitigation

It is recommended to send the reward to the `kycAddress` not the `msg.sender`.

```
File: VVVVCTokenDistributor.sol
 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.kycAddress,
                _params.tokenAmountsToClaim[i]
            );
        }

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

Discussion

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits:
https://github.com/vvvdevs/vvv-platform-smart-contracts/pull/99

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.