SHERLOCK

SHERLOCK SECURITY REVIEW FOR

Contest type:	Public
Prepared for:	Tokensoft
Prepared by:	Sherlock
Lead Security Expert:	jkoppel
Dates Audited:	May 31 - June 4, 2024
Prepared on:	August 1, 2024

Introduction

Adding "Per Address" functionality to existing distribution contracts on Tokensoft’s platform.

Scope

Repository: SoftDAO/contracts

Branch: per-user-config

Commit: fb34000f9b2e0863c5e9fb001caf52de0e47cfee

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

Medium	High
2	0

Issues not fixed or acknowledged

Medium	High
0	0

Security experts who found valid issues

jkoppel	BiasedMerc	aman
samuraii77	Ironsidesec	1337web3
nfmelendez	Honour	0xboriskataa

SHERLOCK

Varun_05	robertodf
merlin	hunter_w3b

Issue M-1: AdvancedDistributorInitializable sets claim data to empty, making claims fail

Source: https://github.com/sherlock-audit/2024-05-tokensoft-distributor-contracts-update-judging/issues/11

Found by

0xboriskataa, 1337web3, BiasedMerc, Honour, Ironsidesec, Varun_05, aman, hunter_w3b, jkoppel, merlin, nfmelendez, robertodf, samuraii77

Summary

AdvancedDistributorInitializable.claim overrides the passed-in data with new bytes(0). This data is needed by both PerAddressTrancheVestingMerkleDistributor and PerAddressContinuousVestingMerkleDistributor for claims to work. Therefore, claims do not work.

Vulnerability Detail

Impact

Claims do not work

Both PerAddressTrancheVestingMerkleDistributor and PerAddressContinuousVestingMerkleDistributor also pass in `new bytes(0)` to this argument, causing the same issue. However, this is a separate issue per my understanding of Sherlock rules; it is in separate code, and will stay broken if the others are fixed.

Code Snippet

AdvancedDistributorInitializable calls DistributorInitializable._executeClaim with data=new bytes(0)

https://github.com/sherlock-audit/2024-05-tokensoft-distributor-contracts-update/blob/main/contracts/packages/hardhat/contracts/claim/factory/AdvancedDistributorInitializable.sol#L106-L113

```
function _executeClaim(address beneficiary, uint256 totalAmount, bytes memory)
    internal
    virtual
    override
    returns (uint256 _claimed)
{
    _claimed = super._executeClaim(beneficiary, totalAmount, new bytes(0));
    _reconcileVotingPower(beneficiary);
}
```

DistributorInitializable._executeClaim forwards this data to getClaimableAmount, which forwards it to getVestedFraction

https://github.com/sherlock-audit/2024-05-tokensoft-distributor-contracts-update/blob/main/contracts/packages/hardhat/contracts/claim/factory/DistributorInitializable.sol#L75 https://github.com/sherlock-audit/2024-05-tokensoft-distributor-contracts-update/blob/main/contracts/packages/hardhat/contracts/claim/factory/DistributorInitializable.sol#L113

Many implementations of getVestedFraction attempt to decode this data in ways that break if the data is empty.

E.g.:

https://github.com/sherlock-audit/2024-05-tokensoft-distributor-contracts-update/blob/main/contracts/packages/hardhat/contracts/claim/factory/PerAddressContinuousVestingInitializable.sol#L30-L35

```
function getVestedFraction(
    address beneficiary,
    uint256 time, // time is in seconds past the epoch (e.g. block.timestamp)
    bytes memory data
) public view override returns (uint256) {
    (uint256 start, uint256 cliff, uint256 end) = abi.decode(data, (uint256,
↪   uint256, uint256));
```

Putting these together, attempting to claim will be a lot like this Chisel section.

```
 bytes memory data = new bytes(0)
 (uint256 start, uint256 cliff, uint256 end) = abi.decode(data, (uint256,
↪   uint256, uint256));
Traces:
  [805] 0xBd770416a3345F91E4B34576cb804a576fa48EB1::run()
    ← "EvmError: Revert"

 Chisel Error: Failed to execute REPL contract!
```

Tool used

Manual Review

Recommendation

Pass the data parameter along properly

Discussion

ArshaanB

Yes, can confirm, this seems like an issue.

`ContinuousVestingInitializable`, `TrancheVestingInitializable` — the `getVestedFraction()` doesn't care about `data`
`PerAddressContinuousVestingInitializable`, `PerAddressTrancheVestingInitializable` — the `getVestedFraction()` does care about `data`

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits: https://github.com/SoftDAO/contracts/pull/47

ArshaanB

An additional PR covering a case that I didn't cover in PR 47 above: https://github.com/SoftDAO/contracts/pull/51

ArshaanB

PR covering a case for PACVM https://github.com/SoftDAO/contracts/pull/52

sherlock-admin2

The Lead Senior Watson signed off on the fix.

Issue M-2: PerAddressTrancheVestingMerkleDistributor.claim always reverts because it checks the Merkle proof incorrectly

Source: https://github.com/sherlock-audit/2024-05-tokensoft-distributor-contracts-update-judging/issues/15

Found by

jkoppel, samuraii77

Summary

Both the `claim` and `initializeDistributionRecord` methods of PerAddressTrancheVestingMerkleDistributor do not include the tranches when checking the Merkle proof (and do not even accept the tranches as a parameter). Both methods therefore always revert, before the function body is even entered.

Vulnerability Detail

We know from PerAddressTrancheVestingMerkle that the Merkle trees for this kind of distributor include the list of tranches for each address (and also because they have to be; there's no other way to specify the tranches). However, PerAddressTrancheVestingMerkleDistributor omits the tranches when checking the Merkle proofs. The Merkle proofs therefore always revert.

Impact

PerAddressTrancheVestingMerkleDistributor is completely inoperable.

Code Snippet

Notice the lack of tranches here:

https://github.com/sherlock-audit/2024-05-tokensoft-distributor-contracts-update/blob/main/contracts/packages/hardhat/contracts/claim/factory/PerAddressTrancheVestingMerkleDistributor.sol#L42-L58

```
function initializeDistributionRecord(
    uint256 index, // the beneficiary's index in the merkle root
    address beneficiary, // the address that will receive tokens
    uint256 amount, // the total claimable by this beneficiary
    bytes32[] calldata merkleProof
) external validMerkleProof(keccak256(abi.encodePacked(index, beneficiary,
↪   amount)), merkleProof) {
    _initializeDistributionRecord(beneficiary, amount);
}

function claim(
    uint256 index, // the beneficiary's index in the merkle root
    address beneficiary, // the address that will receive tokens
    uint256 totalAmount, // the total claimable by this beneficiary
    bytes32[] calldata merkleProof
)
    external
    validMerkleProof(keccak256(abi.encodePacked(index, beneficiary,
↪   totalAmount)), merkleProof)
```

Contrast the code in PerAddressTrancheVestingMerkle, which checks the Merkle proof like this:

https://github.com/sherlock-audit/2024-05-tokensoft-distributor-contracts-update/blob/main/contracts/packages/hardhat/contracts/claim/PerAddressTrancheVestingMerkle.sol#L45

```
validMerkleProof(keccak256(abi.encodePacked(index, beneficiary, amount,
↪   abi.encode(tranches))), merkleProof)
```

Tool used

Manual Review

Recommendation

Add the tranches parameter to both methods and check in the Merkle proof

Discussion

ArshaanB

Yes, `tranches` should be passed to `validMerkleProof()` in order to properly validate for these two functions in `PerAddressTrancheVestingMerkleDistributor.sol`.

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits:
https://github.com/SoftDAO/contracts/pull/49

sherlock-admin2

The Lead Senior Watson signed off on the fix.

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.