Inheritable Smart Contract Wallet - Findings Report

Table of contents

M-02. Timer Is Not Reset In InheritanceManager::contractInteractions Which Breaks Core Functionality Of The Contract

Contest Summary

Sponsor: First Flight #35

Dates: Mar 6th, 2025 - Mar 13th, 2025

See more contest details here

Results Summary

Number of findings:

- High: 1
- Medium: 2
- Low: 0

High Risk Findings

H-01. Improper Beneficiary Removal In InheritanceManager::removeBeneficiary Can Lead To Gap In Array Resulting In Unequal Distribution Of Funds

Summary: `InheritanceManager::removeBeneficiary` improperly deletes an element from the `beneficiaries` array, leaving a gap which can lead to unexpected behavior in functions requiring sequential indexing such as unequal funds distribution in `InheritanceManager::withdrawInheritedFunds`.

Description: `InheritanceManager::removeBeneficiary` deletes an element from the beneficiaries array without shifting remaining elements in the array, leaving an empty slot. This causes incorrect indexing.

Impact: Any function iterating over the `beneficiaries` array may fail due to unintended empty slots or behave not as intended. Unequal funds distribution in `InheritanceManager::withdrawInheritedFunds`.

Vulnerability Details: `InheritanceManager::removeBeneficiary` deletes an element from the beneficiaries array without shifting remaining elements in the array, leaving an empty slot. This causes incorrect indexing, which can lead to unequal distribution of funds in `InheritanceManager::withdrawInheritedFunds`.

POC: The vulnerable code:

```
//@audit --> Improper array deletion, can lead to gap in array
function removeBeneficiary(address _beneficiary) external onlyOwner {
uint256 indexToRemove = _getBeneficiaryIndex(_beneficiary);
delete beneficiaries[indexToRemove];
}
```

The following test was written, which returned false, proving that other elements in the array were not shifted after one element had been deleted:

```
function testRemoveBeneficiaryLeavesGap() public {
    address user2 = makeAddr("user2");
    address user3 = makeAddr("user3");
    vm.startPrank(owner);
```

```
    im.addBeneficiery(user1);
    im.addBeneficiery(user2);
    im.addBeneficiery(user3);
    vm.stopPrank();
    vm.startPrank(owner);
    im.removeBeneficiary(user2);
    vm.stopPrank();
    assertEq(0, im._getBeneficiaryIndex(user1));
    assertEq(1, im._getBeneficiaryIndex(user3));
}
```

Recommendations: Replace `delete beneficiaries[indexToRemove];` with a swap-and-pop method, which shifts subsequent elements in the array after one element has been deleted:

```
function removeBeneficiary(address _beneficiary) external onlyOwner {
    uint256 indexToRemove = _getBeneficiaryIndex(_beneficiary);
    uint256 lastIndex = beneficiaries.length - 1;

    if (indexToRemove != lastIndex) {
      beneficiaries[indexToRemove] = beneficiaries[lastIndex]; // Move last element
    }
   beneficiaries.pop(); // Remove last element to maintain array integrity
}
```

Medium Risk Findings

M-01. Timer Is Not Reset In InheritanceManager::removeBeneficiary Which Breaks Core Functionallity Of The Contract

Summary: The 90 days deadline reset isn't present in `InheritanceManager::removeBeneficiary` which breaks the core assumption of the contract

Vulnerability Details: The 90 days deadline is not set in `InheritanceManager::removeBeneficiary` which breaks the core assumption of the contract that all functions called by the owner will reset the 90 days deadline

```
   //@audit --> Function called by owner doesn't reset the 90 days timer
   function removeBeneficiary(address _beneficiary) external onlyOwner {
        uint256 indexToRemove = _getBeneficiaryIndex(_beneficiary);
        delete beneficiaries[indexToRemove];
    }
```

Impact: This breaks the core functionality of the contract that every transaction carried out by the owner would reset the 90 days timer

Tools Used: Foundry

Recommendations: Implement the `_setDeadline` on `InheritanceManager::removeBeneficiary`

```
 //@audit --> Added Deadline reset
        _setDeadline();
```

M-02. Timer Is Not Reset In InheritanceManager::contractInteractions Which Breaks Core Functionality Of The Contract

Summary: The 90 days timer reset isn't present in `InheritanceManager::contractInteractions` which breaks the core assumption that all transactions carried out by the owner should reset the 90 days timer.

Vulnerability Details: The 90 days timer reset isn't present in `InheritanceManager::contractInteractions` which breaks the core assumption that all transactions carried out by the owner should reset the 90 days timer. The vulnerable code:

```
//@audit --> Function called by owner doesn't reset the 90 days timer
   function contractInteractions(address _target, bytes calldata _payload, uint256
        external
        nonReentrant
        onlyOwner
    {
      (bool success, bytes memory data) = _target.call{value: _value}(_payload);
        require(success, "interaction failed");
        if (_storeTarget) {
            interactions[_target] = data;
        }
}
```

Impact: This breaks core functionality of the contract.

Tools Used: Foundry

Recommendations: Implement `_setDeadline()` to `InheritanceManager::contractInteractions`

```
 //@audit --> Added Deadline reset
        _setDeadline();
```