MultiSig Wallet Project Security Audit

Audit Resources:

Github Repository of the project was provided. (Link)

Project Author:

- Vinh Le (Github)

Project Auditor:

- Umair Mirza (Github)

Table of Contents

Audit Summary

The MultiSig Wallet project has been compiled, deployed and tested using the **Hardhat** smart contract development tool chain. The project is comprised of three smart contracts, namely:

- MultiSigFactory.sol
- MultiSigWallet.sol
- TestERC20Token.sol

Following libraries and interfaces have been integrated with the smart contracts:

- Chainlink Keepers
- OpenZeppelin
- ECDSA.sol util
- ERC20.sol Interface

The contracts have been audited by 1 resident from September 28th to October 1st. The repository was under active development during the audit.

Scope

The scope of this audit is limited to the smart contracts mentioned above. Frontend modules of the project have not been audited.
The commit that has been audited is: **102ba19b95a32fa519cf1df1ef4bb2649da2dc39**
This audit is about identifying potential vulnerabilities in the smart contracts. The audit may not identify all potential attack vectors or areas of vulnerability.

Findings Description

Findings have been broken down into sections by their respective impact:

- Critical, High, Medium, Low Impact
    - These are findings that range from attacks that may cause loss of funds, impact control/ownership of the contracts, or cause any unintended consequences/actions that are outside the scope of the requirements.
- Gas Savings
    - Findings that can improve the gas efficiency of the contracts.
- Informational
    - Findings including recommendations and best practices.

Medium Findings

1. Medium - Contract with a payable function, but without a withdrawal capacity

Proof of Concept

Contract MultiSigFactory (contracts/MultiSigFactory.sol#12-136) has payable function:
- MultiSigFactory.create2(uint256,address[],uint256,string)
(contracts/MultiSigFactory.sol#67-112)
But does not have a function to withdraw the Ether

```
function create2(
        uint256 _chainId,
        address[] calldata _owners,
        uint256 _signaturesRequired,
        string calldata _name
    ) public payable
```

Impact

Every Ether sent to the contract will be lost.

Recommendation

Add a Withdraw function.

2. Medium - Reentrancy in MultiSigFactory

Proof of Concept

Reentrancy in MultiSigFactory.create2(uint256,address[],uint256,string)
(contracts/MultiSigFactory.sol#67-112):
External calls:
- multiSig.init(_chainId,_owners,_signaturesRequired)
(contracts/MultiSigFactory.sol#98)
State variables written after the call(s):
- multiSigs.push(multiSig) (contracts/MultiSigFactory.sol#100)

```
        multiSig.init(_chainId, _owners, _signaturesRequired);
        multiSigs.push(multiSig);
```

Impact

An attacker can keep calling the function again and again because the state variable is being updated after the function call.

Recommendation

State variables should be updated before the function call.

Informational Findings

3. Informational - Different pragma directives are used

Proof of Concept

Different versions of Solidity are used:
- Version used: ['>=0.8.0<0.9.0', '^0.8.0']

```
pragma solidity ^0.8.0;
import {AutomationCompatibleInterface as KeeperCompatibleInterface}
from "./AutomationCompatibleInterface.sol";
```

Impact

Sometimes different version can cause compilation issues.

Recommendation

It’s better to use the same solidity version in all of the contracts and interfaces.

4. Informational - There is an Unused State Variable

Proof of Concept

MultiSigWallet.lowerThreshold (contracts/MultiSigWallet.sol#47) is never used in MultiSigWallet (contracts/MultiSigWallet.sol#23-298)

```
    address[] public owners;
    uint256 public signaturesRequired;
    uint256 public nonce;
    uint256 public chainId;
    string public name;
    uint lowerThreshold;
```

Impact

Unused state variables can be misused in some cases.

Recommendation

Remove unused state variables.

Gas Saving Findings

5. Gas Savings - Some functions should be declared external

Proof of Concept

Following functions are not called by the contract and hence, can be declared external to save gas:

- getMultiSig(uint256)
- create2(uint256,address[],uint256,string)
- computedAddress(string)
- init(uint256,address[],uint256)
- addSigner(address,uint256)
- removeSigner(address,uint256)
- updateSignaturesRequired(uint256)
- executeTransaction(address,uint256,bytes,bytes[])
- numberOfOwners()

Impact

If there are a lot of public functions in your contract and they are not being called inside the contracts then it can be very costly gas-wise.

Recommendation

Use the external attribute for functions never called from the contract, and change the location of immutable parameters to calldata to save gas.