OZYS
Account-Abstraction
Security Analysis Report

Prepared by

78ResearchLab

RedSpider

Sep 10, 2024

TABLE OF CONTENTS

PROJECT OVERALL

About Project

The project developed by Ozys focuses on extending the existing ERC-4337 standard for Account Abstraction (AA) by incorporating a custom Multisig Account (MSAccount). Account Abstraction is a concept that allows users to use smart contracts as their accounts. ERC-4337, in particular, defines a way to achieve this abstraction without requiring changes to the Ethereum protocol, enabling features such as account recovery, social recovery, and gas sponsorship.

To further enhance this functionality, Ozys has built upon the ERC-4337-contract repository, integrating a multisig account system (MSAccount). This approach allows multiple signers to collectively manage an account, providing an added layer of security and decentralization in account management. Additionally, a custom ExternalSigner was developed to facilitate the MSAccount's operation, enabling external parties to authorize transactions and participate in the multisig approval process.

Target Summary

Name	Account-Abstraction
Website	https://ozys.io/
Repository	
Commit	7822ec49d21e69b05b5d12aba8eee520a9be94eb
Network	Silicon, Klaytn
Languages	Solidity
Method	Source code auditing
Timeline	June 3, 2024 ~ June 28, 2024

SCOPE

Since the project is based on the erc4337-contract repository, we did not conduct a full analysis of the entire repository. Instead, we focused on analyzing the files within the "ms" directory, specifically MSAccountFactory.sol, MSAccount.sol, and ExternalSigner.sol.

Source code

Name	commit
Account-Abstraction	7822ec49d21e69b05b5d12aba8eee520a9be94eb

```
├── contracts
│   ├── core
│   │   ├── BaseAccount.sol
│   │   ├── BasePaymaster.sol
│   │   ├── EntryPointSimulations.sol
│   │   ├── EntryPoint.sol
│   │   ├── Helpers.sol
│   │   ├── NonceManager.sol
│   │   ├── SenderCreator.sol
│   │   ├── StakeManager.sol
│   │   └── UserOperationLib.sol
│   ├── external
│   │   ├── interfaces
│   │   │   └── IWebAuthn.sol
│   │   ├── ms
│   │   │   ├── ExternalSigner.sol
│   │   │   ├── MSAccountFactory.sol
│   │   │   └── MSAccount.sol
│   │   └── seedless
│   │       ├── SeedlessAccountFactory.sol
│   │       └── SeedlessAccount.sol
│   ├── interfaces
│   │   ├── IAccountExecute.sol
│   │   ├── IAccount.sol
│   │   ├── IAggregator.sol
│   │   ├── IEntryPointSimulations.sol
│   │   ├── IEntryPoint.sol
│   │   ├── INonceManager.sol
│   │   ├── IPaymaster.sol
│   │   ├── IStakeManager.sol
│   │   └── PackedUserOperation.sol
│   ├── package.json
│   ├── samples
│   │   ├── bls
│   │   │   ├── BLSAccountFactory.sol
```

```
|   |   |   ├── BLSAccount.sol
|   |   |   ├── BLSHelper.sol
|   |   |   ├── BLSSignatureAggregator.sol
|   |   |   ├── IBLSAccount.sol
|   |   |   └── lib
|   |   |       ├── BLSOpen.sol
|   |   |       └── hubble-contracts
|   |   |           └── contracts
|   |   |               └── libs
|   |   |                   ├── BLS.sol
|   |   |                   ├── BNPairingPrecompileCostEstimator.sol
|   |   |                   └── ModExp.sol
|   |   ├── callback
|   |   |   └── TokenCallbackHandler.sol
|   |   ├── LegacyTokenPaymaster.sol
|   |   ├── SimpleAccountFactory.sol
|   |   ├── SimpleAccount.sol
|   |   ├── TokenPaymaster.sol
|   |   ├── utils
|   |   |   ├── IOracle.sol
|   |   |   ├── OracleHelper.sol
|   |   |   └── UniswapHelper.sol
|   |   └── VerifyingPaymaster.sol
|   └── utils
|       └── Exec.sol
```

RISK CLASSIFICATION

Severity

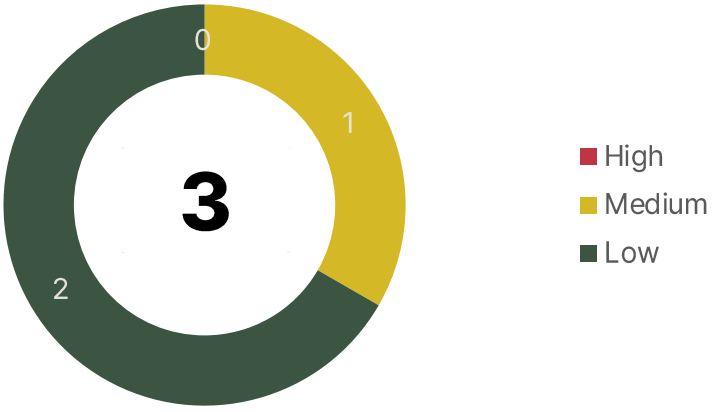
Our risk classification is based on Severity Categorization of code4ena.

> High ●
>
> Assets can be stolen, lost, compromised directly or indirectly via a valid attack path (e.g. Malicious Input Handling, Escalation of privileges, Arithmetic).

> Medium ●
>
> Assets not at direct risk, but the function of the protocol or its availability could be impacted, or leak value with a hypothetical attack path with stated assumptions, but external requirements.

> Low ●
>
> Assets are not at risk. User mistake, misuse of privileges, governance risk fall under this grade.

FINDINGS BREAKDOWN



Severity	Acknowledged	fixed	Total
● High	0	0	0
● Medium	1	0	1
● Low	0	2	2
			3

* Fixed : Risk is fixed by Ozys.
* Acknowledged : Ozys has recognized the risk but has not addressed it, as it poses only a minor impact.

FINDINGS

MEDIUM

M-01. Signature less than required number can change owner in ExternalSigner

Acknowledged

IMPACT

The MSAccount owner can delegate a new owner without verifying signatures greater than the `required` number through the ExternalSigner.

DESCRIPTION

To change the `owner` of the `MSAccount`, the `transferOwnership` function must be called, which can only be invoked by the ExternalSigner. The ExternalSigner can execute transactions via the `execute` function, which requires a number of signatures greater than or equal to the `required` value. Therefore, to change the owner of the MSAccount, approval from the ExternalSigner owners in numbers greater than or equal to the `required` value is needed.

```
function execute(address dest, uint256 value, bytes calldata func, uint256 validUntil,
bytes[] calldata signatures) external onlyOwner {
    require(block.timestamp <= validUntil);
    _validateSignature(dest, value, func, validUntil, signatures);
    _call(dest, value, func);
}

function _validateSignature(address dest, uint256 value, bytes calldata func, uint256
validUntil, bytes[] calldata signatures) internal {
    require(signatures.length >= required);

    bytes32 hash = keccak256(abi.encode(address(this), getChainId(), dest, value, func,
validUntil));
    bytes32 signingHash = MessageHashUtils.toEthSignedMessageHash(hash);
    require(!usedHash[signingHash]);

    uint256 validatedCount = 0;
    for(uint256 i = 0; i < signatures.length; i++){
        address owner = ECDSA.recover(signingHash, signatures[i]);
        require(isOwner[owner]);

        require(!confirmation[signingHash][owner]);
        confirmation[signingHash][owner] = true;
        validatedCount += 1;
    }
```

```
    require(validatedCount >= required);
    usedHash[signingHash] = true;
}
```

File 1 : ExternalSigner Function: execute

The `MSAccount` has a `setExternalSigner` function to set an ExternalSigner, but this function can only be called through the `execute` function of the `MSAccount`. The signature validation process for calling `execute` is as follows:

```
function _validateSignature(PackedUserOperation calldata userOp, bytes32 userOpHash)
internal override virtual returns (uint256 validationData) {
    if(address(externalSigner()) == address(0)) return _validateSingleSignature(userOp,
userOpHash);

    if(userOp.signature.length <= 65) return SIG_VALIDATION_FAILED;

    (bytes memory signatureOwner, bytes memory signatureExternalSigner) =
parseSignature(userOp.signature);

    bytes32 signingHash = MessageHashUtils.toEthSignedMessageHash(userOpHash);

    address recoverOwner = ECDSA.recover(signingHash, signatureOwner);
    if(recoverOwner != owner) return SIG_VALIDATION_FAILED;

    bytes4 externalSignerMagicValue = externalSigner().isValidSignature(userOpHash,
signatureExternalSigner);
    if(externalSignerMagicValue != IERC1271.isValidSignature.selector) return
0xffffffff;

    return SIG_VALIDATION_SUCCESS;
}
```

File 2 : ExternalSigner Function: _validateSignature

When there is no ExternalSigner, the signature is verified only by the `owner`. However, if an ExternalSigner exists, the signature is validated using the `isValidSignature` function of the ExternalSigner. The issue is that the function only checks whether the recovered signature belongs to one of the owners.

If the MSAccount owner holds control over one of the owners of the ExternalSigner, they could:

1. Call the `setExternalSigner` function to set a new ExternalSigner and then arbitrarily change the owner.
2. Upgrade the proxy to arbitrarily change the owner.
3.

RECOMMENDATIONS

When an ExternalSigner exists, only the ExternalSigner should be allowed to upgrade the proxy or call the `setExternalSigner` function.

STATUS Acknowledged

OZYS :

Hello,

It seems there was some confusion due to our failure to share the policy regarding this matter.

First, the permissions are set up as follows:

- MS account owner: the user

- MS account required: owner signature + one signature from ExternalSigner members

- ExternalSigner members: signing server / admin A / admin B / admin C

- ExternalSigner required: 3 out of 4 signatures

- Operator 1: manages signing server and admin A

- Operator 2: manages admin B and admin C

The basic structure ensures that no single party can act alone.

Additionally, regarding the action of changing the ExternalSigner address, if there is a signature from the owner, it is considered a legitimate action intended by the user and will be treated as valid. The export feature we plan to provide to users will allow them to renounce the ExternalSigner address by setting it to the zero address.

Based on this policy, the issue you shared seems to be valid in a normal scenario, so we plan to maintain the current implementation.

Please confirm.

Thank you.

78ResearchLab:

Hello,

It seems that since the ExternalSigner can be changed with only the owner's signature, the owner value could also be changed. Based on the information from Mr. Jong-sik on Slack and your comment, I understand that changing the owner value should require agreement from all members of the ExternalSigner, which is why I believe this could be problematic.

Please confirm!

Thank you.

OZYS:

Hello,

For an exported (setExternalSigner call) account contract:

- We will consider that all authority has been transferred to the owner.

- Our internal system will no longer handle the account (e.g., the signing server will stop providing signatures).

Thus, the concern you raised about being able to change the owner value by changing the ExternalSigner address with only the owner's signature doesn't seem to apply here, as the account will no longer be under our management, and no further agreement from the ExternalSigner will be required.

Additionally, for convenience, when an account is exported (i.e., the ExternalSigner address is renounced to the zero address), the `transferOwnership` execution permission will be changed to `onlySelf`. You can refer to `4a1bbdb28c30a69e4323f75666155f317703bede` for the change.

Please provide your feedback!

Thank you.

78ResearchLab:

If changing the ExternalSigner means that it no longer needs to be managed, then that sounds fine to me!

LOW

L-01. Values initialized through 'initialize' are not properly verified

Fixed

IMPACT

There is a possibility that the `owner` or `required` values in `ExternalSigner` could be initialized with incorrect values.

DESCRIPTION

```
function initialize(address[] calldata owners_, uint256 required_) public virtual
initializer {
    _initialize(owners_, required_);
}

function _initialize(address[] calldata owners_, uint256 required_) internal virtual {
    for(uint256 i = 0; i < owners_.length; i++) _addOwner(owners_[i]);
    _changeRequirement(required_);
}
```

File 3 : ExternalSigner Function: initialize

The `initialize` function in `ExternalSigner` calls `_addOwner` and `_changeRequirement` to initialize the `owners` and `required` values. There are public wrapper functions for these functions, which performs validations for values being set. Therefore, during the initialization process, which calls `_addOwner` directly, can lead to invalid values being set.

```
function addOwner(address owner) public onlySelf {
    require(owner != address(0));
    _addOwner(owner);
}

function _addOwner(address owner) internal {
    require(!isOwner[owner]);

    owners[ownerCount] = owner;
    ownerCount += 1;
    isOwner[owner] = true;

    emit AddOwner(owner);
}
```

File 4 : ExternalSigner function: addOwner

RECOMMENDATIONS

Since incorrect values could be set during the initialization process, it is recommended to perform value validation within the internal functions.

STATUS Fixed

Ozys : fixed in commit `f4ecfb6cb00db84c183630bba74f354ab60ffea4`.

L-02. The Event is omitted when initializing externalSigner

Fixed

IMPACT

The event that should have been emitted during initialization process is missing.

DESCRIPTION

```
function _setExternalSigner(address newExternalSigner) internal virtual {
    emit SetExternalSigner(address(externalSigner()), newExternalSigner);
    _externalSigner = IERC1271(newExternalSigner);
}
```

File 4 : MSAccount Function: _setExternalSigner

In `MSAccount`, when the value of `_externalSigner` is updated, the `SetExternalSigner` event is emitted. However, if you look at the `_initialize` function, despite the change in the value of `_externalSigner`, the corresponding event is not emitted.

```
function _initialize(address anOwner, address anExternalSigner) internal virtual {
    owner = anOwner;
    _externalSigner = IERC1271(anExternalSigner);
    emit OwnershipTransferred(address(0), anOwner);
    emit MSAccountInitialized(_entryPoint, _externalSigner, owner);
}
```

File 5 : MSAccount Function: _initialize

RECOMMENDATIONS

An event should be emitted when the value of `_externalSigner` changes.

STATUS

Fixed

Ozys : fixed in commit `f5b655c1d04d49097143fa12a8a90434ed673f52`.

ABOUT 78ResearchLab

78ResearchLab is a offensive security corporation offering security auditing, penetration testing, education to enterprises, national organizations, and laboratories with the goal of making safe and convenience digital world. We have our own proprietary technology from system/security analysis and projects on various industries. We are working with the top technical experts who have won prizes in global Realword Hacking Competition/CTF, reported numerous security vulnerabilities, and have 10 years of experience in the information security.
Learn more about us at https://www.78researchlab.com/.

ABOUT RED SPIDER

Red Spider offers cyber security research and auditing service with our new R&D technologies and customized solution in IoT, OS, Web3.
Learn more about red spider at https://www.78researchlab.com/redSpider.html.