STATE MIND

Instadapp Avocado v3

20-07-2023 - 28-09-2023

Table of contents

1. Project Brief

Title	Description
Client	Instadapp
Project name	Instadapp Avocado v3
Timeline	20-07-2023 - 28-09-2023
Initial commit	0eec33a45784e5f4b3b45ac2c9dfefc5225bda0a
Final commit	dbe165d01994b496acdbaa9523460e1596de066d

Short Overview

The audit covers the core contracts for Instadapp Avocado.

There are 5 main parts to the Avocado architecture:

- AvocadoMultisig: logic contract that executes multiple actions for owner based on auth logic such as EIP712 signature
- Avocado: minimalistic simple proxy that falls back to AvocadoMultisig
- AvoFactory: deploys Avocado contracts with implementation set to AvocadoMultisig logic contract deterministcally if necessary and computes deterministic address
- AvoForwarder: main interaction point, especially for relayers. Forwards transactions with signature and list of actions to Avocado of a user.
- AvoRegistry: holds valid implementation versions for Avocados and AvoForwarder & sets fee config for gas cost markup for direct owner transactions. Configurable by owner

Helpers:

- AvoDepositManager: Handles USDC deposits to pay for multichain gas fees
- AvoSignersList: tracks the mappings of Avocado <> allowed signers

Project Scope

The audit covered the following files:

- AvoMultiSafe.sol
- AvoDepositManager.sol
- AvoAuthoritiesList.sol
- AvoAuthoritiesListProxy.sol
- AvoWalletErrors.sol
- AvoWalletVariables.sol
- AvoWalletEvents.sol
- AvoWalletInitializable.sol
- AvoWallet.sol
- AvoSignersList.sol
- AvoSignersListProxy.sol
- AvoAdmin.sol
- AvoFactory.sol
- AvoDepositManagerProxy.sol
- AvoVersionsRegistry.sol
- InstaFlashAggregatorInterface.sol
- IWeth9.sol
- ICREATE3Factory.sol
- InstaFlashReceiverInterface.sol
- AvoCoreEvents.sol
- AvoCore.sol
- AvoCoreStructs.sol
- AvoCoreErrors.sol
- AvoCoreVariables.sol
- AvoFactoryProxy.sol
- AvoMultisigErrors.sol
- AvoMultisigInitializable.sol
- AvoMultisig.sol
- AvoMultisigVariables.sol
- AvoMultisigEvents.sol
- AvoVersionsRegistryProxy.sol
- AvoGasEstimationsHelper.sol
- IAvoFactory.sol
- IAvoVersionsRegistry.sol
- IAvoMultisigV3.sol
- IAvoWalletV2.sol
- IAvoWalletV3.sol
- IAvoWalletV1.sol
- IAvoSignersList.sol
- IAvoForwarder.sol
- IAvoAuthoritiesList.sol
- AvoSafe.sol
- AvoForwarder.sol
- AvoForwarderProxy.sol

2. Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss of funds to be transferred to any party.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss of funds.
Informational	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

3. Summary of findings

Severity	# of Findings
Critical	1 (1 fixed, 0 acknowledged)
High	2 (2 fixed, 0 acknowledged)
Medium	6 (3 fixed, 3 acknowledged)
Informational	27 (22 fixed, 5 acknowledged)
Total	36 (28 fixed, 8 acknowledged)

4. Conclusion

During the audit of Avocado v3 codebase, 36 issues were found in total:

- 1 critical severity issue (1 fixed)
- 2 high severity issues (2 fixed)
- 6 medium severity issues (3 fixed, 3 acknowledged)
- 27 informational severity issues (22 fixed, 5 acknowledged)

The final reviewed commit is dbe165d01994b496acdbaa9523460e1596de066d

Deployment

File name	Contract deployed on mainnet
AvoMultisigAdmin	0x91fFc68D5021A08b54E5bC9903Cac640aC271F0b
AvoAdmin	0x99a73A3cc7965baE8024e0707332327E8E1Fa656
AvoConfigV1	0x1412121E487478498b355C227c77D5ED6CF1798d
EmptyImplementation	0x33428CA4Fc1c2372547791E4F16F4106f06b3227

AvoFactoryProxy	0xe981E50c7c47F0Df8826B5ce3F533f5E4440e687
AvoRegistryProxy	0x779385EC7A04242259AdD4990e3130846f80eA69
AvoForwarderProxy	0x46978CD477A496028A18c02F07ab7F35EDBa5A54
AvoDepositManagerProxy	0xE62C1eF23C30C3FF1738e9E91A5674062314D171
AvoSignersListProxy	0xA27A71DD0348b5C69370eF9Da3743197DC006848
AvoRegistry	0xa2A467f3C5B5427F9447117e38b5D36422ce4621
AvoFactory	0x732D01C9b3C0A8EcF6a586DfFE940d3883c3B5DD
AvoForwarder	0xA9910e485b1c789443d504D4E40c559e32A56452
AvoDepositManager	0x940029f3475556d7492cef1968210C4FAAe56922
AvoSignersList	0x6e7E2A4e77706Fac5a392E42c3c66C1fdA9De17A
AvocadoMultisig	0xaa282C8aB681fbD501A2B8fA0Ea558cbF5785F73
Avocado	0xE95EC44a32CFca82Cd9B88e267C4C0d68Ec5127a

5. Findings report

CRITICAL-01	Ability to change the implementation of the wallet during the upgrade from v2 to v3	Fixed at bd8f96

Description

In AvoCore.sol some restrictions are enforced through the usage of **_transientAllowHash** and **_transientId**. There is manipulation of this variables during the upgrade process from V2 version. Here is the plan:
All actions performed by the owner (they should be done in the same block):

1. Calling the **cast** function to modify the slot 53 where **_transientAllowHash** is stored with value equal to **bytes31(keccak256(abi.encode(data_, block.timestamp, EXECUTE_OPERATION_SELECTOR)))** and **_transientId** equal to **1** (to enable **delegatecall** actions).
2. Calling **upgradeTo(V3Wallet)**

Now we have an access to **executeOperation** function.

3. Calling **executeOperation** and change slot0 that stores the implementation.

We avoid storage slot0 snapshot checking, because **isFlashloanCallback_ == true**.
In the final step, we can set any implementation. For example, without any fees or set the smart contract as the owner. The same process could be done if there would be possibility to change versions to old ones.

Recommendation

We recommend resetting **_transientAllowHash** and **_transientId** during the upgrade process and taking snapshots even if it is **flashloan callback**.

HIGH-01	Possibility of DoS from the AvoVersionsRegistry contract	Fixed at bf393d

Description

Admin's controlled contract **AvoVersionsRegistry** can DoS all wallets using the **calcFee** function. It is called by wallets in the internal **_payAuthorizedFee** function of the **AvoCore** contract on L300 which is called every time when an owner of a wallet or signers of a multisig call the **castAuthorized** function (wallet and multisig).
This can happen in several ways:

1. The function **calcFee** spends unlimited amount of gas. In this case, transaction with this call can't be processed.
2. The function **calcFee** returns less than 64 bytes. In this case there will be a revert on L305 because **abi.decode** will ask for 64 bytes (**uint256 + address**) and there will be less amount of return data.
3. The function **calcFee** returns a wrong data for the **feeCollector_** address. For example, if the **calcFee** function return a value that is equal to the **uint256(type(uint160).max) + 1** value, then the **abi.decode** instruction will revert decoding this value into the **feeCollector_** variable of the type **address**.

This opens a possibility for an admin block all transactions that aren't casted by forwarders.

Recommendation

It is recommended to add gas limit for the call to the **calcFee**. Also, add this checks to the **_payAuthorizedFee** function:

```
if (success_ && result_.length >= 64) {
    uint256 addressValue;
    assembly {
        addressValue := mload(add(result_, 0x40))
    }
    if (addressValue <= type(uint160).max) {
        (feeAmount_, feeCollector_) = abi.decode(result_, (uint256, address));
        if (feeAmount_ > AUTHORIZED_MAX_FEE) {
            // make sure AvoVersionsRegistry fee is capped
            feeAmount_ = AUTHORIZED_MAX_FEE;
        }
    } else {
        feeCollector_ = AUTHORIZED_FEE_COLLECTOR;
        feeAmount_ = AUTHORIZED_MIN_FEE;
    }
} else {
    ...
}
```

HIGH-02	One signer can takeover the multisig	Fixed at 002eef

Description

After contract deployment, the multisig has one signer who is the owner. The owner can add signers to the multisig using **addSigners()**, but it doesn't update **requiredSigners**. If the owner does not call **setRequiredSigners()** within the same **cast** as **addSigners()**, this can lead to one added signer to takeover the multisig, since **requiredSigners == 1**. A malicious signer can frontrun any transaction and transfer the funds from the multisig.
The same applies to **removeSigners()**. For example, to remove all signers except the owner, **requiredSigners** has to be set to **1** separately from **removeSigners()** call.

Recommendation

We recommend to add an option to update **requiredSigners** value in the **addSigners()** and **removeSigners()** functions.

MEDIUM-01	State violation via call	Fixed at 86ce51

Description

In function _executeActions() if operation is delegate call or flashloan callback, then contract at the end of function resets values for **_transientAllowHash** and **_transientId** to prevent reentrancy and further state violation. But there is possibility to change these variables in simple call. Firstly, we should make delegate call to change implementation with special function:

```
function setSlot(bytes32 slot, bytes32 value) public {
        assembly {
            sstore(slot, value)
        }
}
```

Then we can change contract state using simple call operation.

Recommendation

It is recommended to reset variables **_transientAllowHash** and **_transientId** and add checks for slot shapshots after every action.

Client's comments

Implemented in PR https://github.com/Instadapp/avocado-contracts/pull/160 The check is after every delegateCall now, but not after every action. I think probably that's what you meant anyway? Please confirm.

MEDIUM-02	Violation of gas calculations in case of failed cast	Fixed at 8e2adc

Description

In **AvoCore** contract immutable variables **CAST_EVENTS_RESERVE_GAS** and **CAST_AUTHORIZED_RESERVE_GAS** are used to estimate gas consumption on events emitting at Lines 404-408. But in case of failure, cost of **CastFailed** is dynamic and depends on variables **revertReason_** and **params_.metadata** and could be more than **11_000** gas, because their size is not limited. This could lead to incorrect gas calculation and possible revert due to **out of gas** and incorrect fee calculations in _payAuthorizedFee().

Recommendation

It is recommended to add restriction on length of variables of **revertReason_** and **params_.metadata**.

MEDIUM-03	Custom errors aren't decoded in proper way	Fixed at 421dd3

Description

In the contract **AvoCore** in the function **_getRevertReasonFromReturnedData** in the case when **returnedData_.length > 68** the contract suggests that an error has the signature **Error(string)** which is not true in all cases. For example, in the **_executeActions** there was a call that reverted with a custom error with the signature **CustomError(uint256,uint256,uint256)** and with values **1e18, 1e18, 1e18**. In that case, the reverted data will come into the function **_getRevertReasonFromReturnedData** that will try to decode error data, but the decoding will revert in this case because there is no data with at the **1e18** offset.

Recommendation

It is recommended to check that an error selector equals to the selector of the **Error(string)** error before decoding data.

Client's comments

Implemented in https://github.com/Instadapp/avocado-contracts/pull/161, along with some other fixes. please confirm

MEDIUM-04	Possible function selector collision	Acknowledged

Description

At the lines AvoMultiSafe.sol#L65, AvoMultiSafe.sol#L72, AvoMultiSafe.sol#L83, AvoSafe.sol#L53, the proxy contracts check if the calldata matches the selector and return values if it does. However, if the implementation contract has functions with the same function selector, then it will be impossible to call those functions on the implementation.

Recommendation

We recommend to move these view functions from proxy contracts to implementation or use a transparent proxy pattern.

Client's comments

this is acceptable to us, but we added comments and tests to make sure we will not accidentally have such methods on the logic contracts: https://github.com/Instadapp/avocado-contracts/pull/162 (https://github.com/instadapp/avocado-contracts/commit/e9d62e1fed8370d6386612e42180fb691abe9ca4)

MEDIUM-05	A user can make AvoGasEstimationsHelper to estimate gas incorrectly	Acknowledged

Description

The contract **AvoGasEstimationsHelper** uses **tx.origin == 0x000000000000000000000000000000000000dEaD** for backend gas estimations. A user can abuse this by checking the **tx.origin** in the **action.target**, i.e. if **tx.origin == 0x000000000000000000000000000000000000dEaD** then it does nothing, otherwise it executes normally. **AvoGasEstimationsHelper** will estimate gas incorrectly, and if this estimation is used for gas payment in the backend, a user can pay less fees.

Recommendation

We recommend to use the **eth_call** RPC method to evaluate gas usage.

Client's comments

interesting point. But if user would do this, then the broadcaster would not send enough gas and the tx would fail when actually executed? Also, we don't use the estimated gas fees as payment, but rather the actual tx cost after execution. Can't use the eth_call RPC because we want (and need) to estimate before signature. By action.target you simply mean the target contract for the action right?

MEDIUM-06	Broadcaster can send the cast transaction with a lot more gas than the user instructed	Acknowledged

Description

At the lines AvoWallet.sol#L372, AvoMultisig.sol#L580, **forwardParams_.gas** is the user instructed minimum amount of gas that the relayer (AvoForwarder) must send for the execution. However, there is no maximum amount of gas that the relayer can send. For example, the user expects **cast** to consume **100_000** gas and sets **forwardParams_.gas = 100_000**. But after the user signed the message and before the broadcaster sends the transaction, due to external factors the cast consumes **1_000_000** gas. The broadcaster sends the transaction with **1_000_000** gas. Then the user has to pay for gas 10 times more than they expected.
It would be better to let the user set the maximum gas they are willing to pay and enforce it on-chain to make sure the broadcasters cannot abuse their role to force users to pay more gas fees.

Recommendation

We recommend to add a parameter to **CastForwardParams** for maximum gas the relayer is allowed to send.

Client's comments

We could add the maxGas param in addition to current "gas" (minGas) param. There will always be some implicit trust from the user though to work with the unified gas tank that is one of the main features of Avocado. The user has to trust the architecture behind it to charge the correct amount in the end. The problem is, user gives instructions for gas amounts on his contract, but part of gas that is also charged is for execution on Forwarder - any check there could be removed (upgradeable)...

INFORMATIONAL-01	Possibility of DoS DepositManager	Fixed at 88f7c0

Description

In AvoDepositManager.sol **withdraw** function is open for everyone. So anybody can withdraw **balanceOf(contract) – withdrawLimit**. If there is a transaction in mempool with **processWithdraw** for amount close to **withdrawLimit** users can frontrun this transaction and contract balance become 0. There is no any check in **requestWithdraw()** and **withdrawLimit** don't decrease or increase in the code, only setter available for owner.

Recommendation

If it's expected behavior we recommend adding access control to **withdraw()** if withdrawLimit going to be small or implementing increasing/decreasing of Limit when users deposit or auths call **requestWithdraw**.

INFORMATIONAL-02	Little gas optimizations	Fixed at b65663

Description

1.**isAllowedSigner_** can be stored outside of the loop

2.**depositToken** can be used in withdraw function instead of creating local variable, because it declared as **immutable**. It will decrease gas cost of usage but increase size of contract.

Recommendation

We recommend implementing optimization.

INFORMATIONAL-03	Unchecked delegatecall in upgradeToAndCall	Fixed at 9959a1

Description

In function upgradeToAndCall() after changing **implementation** there is **delegatecall** to new **implementation**. This may lead to state violation and bypassing **onlySelf** modifier.

Recommendation

It is recommended to add sanity checks for important slots after calling this method, because in **cast()** method snapshots won't be taken due to simple call.

INFORMATIONAL-04	Misleading name of parent contract of AvoFactory	Fixed at bbb0f0

Description

At Line 130 **AvoFactory** contract is inherited from **AvoForwarderCore** which leads to misleading with actual one in **AvoForwarder.sol**.

Recommendation

It is recommended to rename abstract contract **AvoForwarderCore** in **AvoFactory.sol** to **AvoFactoryCore**.

INFORMATIONAL-05	Check for set value for avoWalletImpl and avoMultisigImpl variables	Fixed at b9cd50

Description

the state would remain). Calling deploy functions for **AvoSafe** and **AvoSafeMultisig** would set implementation slot to zero and would revert during initialization.

Recommendation

It is recommended to check implementation variables **avoWalletImpl** and **AvoMultisigImpl** before deploying proxies.

INFORMATIONAL-06	Old verify function could be marked as view	Fixed at 097c78

Description

Functions verifyV1, verify() and verifyV2 are not marked as **view**, because they potentially may deploy wallets. But inside function _getDeployedLegacyAvoWallet() is called which just computes wallet address and checks if it is contract. So in old versions there is no deploy for wallet and they can be marked as **view**.

Recommendation

It is recommended to mark old verify functions as **view** functions.

INFORMATIONAL-07	Sanity check for feeConfig	Fixed at 3ca1cf

Description

In function updateFeeConfig() new config is set, but it is not checked for correct values. When **mode** equals **0**, then **fee** parameter is in percents and should be checked to be less than or equal **100% (10**8 in code)**

Recommendation

It is recommended to add sanity checks for **feeConfig**.

Client's comments

Note top-up fee can be > 100% in percentage mode (it could be e.g. 3 times the used gas cost). We added a sanity check that it must be < 1000% though: https://github.com/Instadapp/avocado-contracts/pull/162

INFORMATIONAL-08	Outdated interface for IAvoWalletV2	Fixed at 19b218

Description

Currently, working version of **AvoWallet** is **2.0.1**, other versions are deprecated in **AvoVersionsRegistry**. In interfaces section there is only interface for **2.0.0** and it differs from current one. It misses **StorageSnapshot** struct and in new version there is no **castAuthorized()** method (its logic moved to **cast()** function). Also in **deployments** directory only one address is provided (for version **2.0.0**).

Recommendation

It is recommended to update **interface** and **deployments** sections to match currently deployed and working versions of **AvoWallet**.

Client's comments

Interface fixed in https://github.com/Instadapp/avocado-contracts/pull/162. The interface is the same for versions 2.0.0 and 2.0.1 (we follow semver). But with past changes we had planned to release a 2.1.0, which never happened and it looks like changes got messed up. Adjusted the interface to be correct for 2.0.x. Deployments have moved to a separate repo with better tooling, where 2.0.1 (and other versions) is tracked correctly. In fact we will remove deployments completely soon after migration to the new repo is confirmed: https://github.com/Instadapp/avocado-contracts/pull/129

INFORMATIONAL-09	Too much gas providing for feeCollector	Fixed at e47a58

Description

At Line 331 previously calculated **feeAmount** is transferred to **feeCollector** and **22000** gas is provided for this action. But in EVM, if **value** is set and is transferred with **CALL** opcode, then automatically **2300** gas is added to provided gas to ensure transfer could happen. So actually, there is no need to provide any gas along with **call** for simple transfer.

Recommendation

It is recommended to leave a comment if it is intended or reduce provided gas to prevent any actions on **feeCollector** side.

INFORMATIONAL-10	Code duplication	Fixed at 9fd06d

Description

In the contracts **AvoMultisig** and **AvoWallet** there are pairs of functions that are identical. Pairs of functions:

1. The function **getSigDigest** in the **AvoMultisig** and **AvoWallet** contracts
2. The function **getSigDigestAuthorized** in the **AvoMultisig** and **AvoWallet** (named with different name **nonSequentialNonceAuthorized**) contracts

Recommendation

These functions can be moved into the **AvoCore** contract to remove code duplication.

INFORMATIONAL-11	Variables can be made internal	Fixed at 854ec1

Description

In the **AvoForwarder** contract there are global variables **broadcasters** and **auths** that can be made **internal** instead of **public** because there are external functions **isBroadcaster** and **isAuth** that can be used to check values in that mappings.

Recommendation

It is recommended to make global variables **broadcasters** and **auths internal** instead of **public**.

INFORMATIONAL-12	Gas optimization	Fixed at e247ce

Description

There are some gas optimizations that can reduce gas consumption of the contracts:

1. The **AvoCore** contract
    1. In the function **_domainSeparatorV4** the value **keccak256(abi.encodePacked(block.chainid))** can be cached into an immutable variable inside the constructor
2. The **AvoMultisigVariables** contract
    1. In the function **_getSigners** the global variable **_signersPointer** is read double times

Recommendation

It is recommended to make gas optimizations to reduce gas consumption

INFORMATIONAL-13	Implicit conversion to address	Fixed at 1b47b9

Description

At the line AvoCore.sol#L79, **sload(_avoImplementation.slot)** takes the implementation address from slot 0. However, the implementation address is packed with other variables in that slot. It is better to explicitly convert slot value to **address** using a bit mask.

Recommendation

We recommend to change to **and(sload(0), 0xffffffffffffffffffffffffffffffffffffffff)**.

INFORMATIONAL-14	No revert message	Fixed at c75316

Description

At the line AvoCore.sol#L496, there is no revert message. Currently, the function **_toHexDigit()** is not used with values **> 15** so it shouldn't revert. However, if it reverts in a future update, it will be useful know why it reverted.

Recommendation

We recommend to add a custom error for this function to be more consistent with the entire codebase.

INFORMATIONAL-15	Typos	Fixed at 7b41e7

Description

There are several typos, at the lines:

- AvoCoreVariables.sol#L54, it should be **address**,
- AvoWallet.sol#L115, it should be **signal**,
- AvoSafe.sol#L31, there is no **hash**.

Recommendation

We recommend to fix these typos.

INFORMATIONAL-16	Magic numbers can be replaced with constant variables	Fixed at 3a0251

Description

There are magic numbers which can be replaced with constant variables at the lines:

- AvoWalletVariables.sol#L39-L40
- AvoMultisigVariables.sol#L50-L51
- AvoMultisig.sol#L260
- AvoMultisig.sol#L270

Recommendation

We recommend to replace magic numbers with constant variables.

INFORMATIONAL-17	Ownable contracts can renounce ownership	Fixed at b9cd50

Description

The contracts **AvoAdmin**, **AvoDepositManager**, **AvoForwarder**, **AvoVersionsRegistry** have the function **renounceOwnership()** which can be called to renounce ownership. If it is called by mistake, the contracts will be left without an owner. If the owner is **address(0)**, important functions will not be available.

Recommendation

We recommend to override the function **renounceOwnership()** and revert if called.

INFORMATIONAL-18	No address(0) check in AvoFactory	Fixed at b9cd50

Description

In **AvoFactory**, the modifier **onlyEOA()** doesn't check if the address **owner_ == address(0)**. This makes it possible to deploy an **AvoSafe** or **AvoMultiSafe** with **address(0)** as the owner.

Recommendation

We recommend to check if the **owner** is not **address(0)**.

INFORMATIONAL-19	Can upgrade the implementation to the same implementation	Fixed at 13b420

Description

In the functions **AvoWallet.upgradeTo()**, **AvoMultisig.upgradeTo()**, there is no check if **avoImplementation_** is equal to the current implementation **_avoImplementation**. This makes it possible to upgrade the implementation to the same implementation and delegate call.

Recommendation

We recommend to check if **avoImplementation_ == _avoImplementation**.

INFORMATIONAL-20	Gas optimizations	Fixed at 5f4980

Description

1. It is possible to save gas by removing **== true** and replacing **== false** with **!** in **if** conditions:
    - AvoAuthoritiesList.sol#L179
    - AvoAuthoritiesList.sol#L190
    - AvoDepositManager.sol#L170
    - AvoFactory.sol#L176
    - AvoForwarder.sol#L390
    - AvoForwarder.sol#L484
    - AvoForwarder.sol#L547
    - AvoForwarder.sol#L618
    - AvoMultisig.sol#L625
    - AvoMultisig.sol#L734
    - AvoWallet.sol#L404
    - AvoWallet.sol#L476
    - AvoSignersList.sol#L146
    - AvoSignersList.sol#L156
    - AvoSignersList.sol#L206
    - AvoSignersList.sol#L224
    - AvoSignersList.sol#L237

    - AvoSignersList.sol#L274
2. Replace **i++** with **++i**:
    - AvoForwarder.sol#L682
    - AvoForwarder.sol#L711
    - AvoForwarder.sol#L762

Recommendation

We recommend to implement these gas optimizations.

INFORMATIONAL-21	Missing isAvoSafe checks	Fixed at 76aeb9

Description

1. At the line AvoSignersList.sol#L93, AvoSignersList.sol#L112, there is a check if **avoMultiSafe_** is a contract, but it would be better to use the **AvoFactory.isAvoSafe()** check.
2. At the line AvoAuthoritiesList.sol#L91, there is no check if **avoSafe_** is deployed by **AvoFactory**.
3. At the lines AvoAuthoritiesList.sol#L89, AvoAuthoritiesList.sol#L99, AvoAuthoritiesList.sol#L119, AvoSignersList.sol#L85. If the **AvoSafe** or **AvoMultiSafe** is selfdestructed, it will remove signers and authorities without calling **sync** functions in **AvoAuthoritiesList** and **AvoSignersList**. It will be impossible to call **sync** functions because **AvoFactory** checks if there is code at the address. These functions will return wrong old values.

Recommendation

We recommend to add **isAvoSafe** checks.

INFORMATIONAL-22	Broadcaster can send msg.value more or less than required in actions	Fixed at 370b44

Description

The broadcaster sets **msg.value** at the line AvoForwarder.sol#L541, to send to **cast**. It is unclear how this value is set by the broadcaster off-chain. If the user pays for **msg.value** by sending Ether to the broadcaster, the broadcaster can accidentally send more than he received and the leftover Ether will stay in the wallet. The broadcaster can set **msg.value = 0**, then the Ether for actions will be used from the user's **AvoWallet** balance. Then the user will pay **msg.value** twice.

Recommendation

We recommend to add a **CastForwardParams** parameter value set by the user that has to equal **msg.value** set by the broadcaster. The user should choose whether to use **AvoWallet** Ether balance for target actions.

INFORMATIONAL-23	Open reinitialize() function in AvoForwarder	Acknowledged

Description

In new **AvoForwarder** version function reinitialize() is open. After setting new implementation, anyone can call this function and set custom broadcasters.

Recommendation

It is recommended to update implementation version via **upgradeToAndCall()** method of **TransparentProxy** or restrict access to **reinitialize()** function.

Client's comments

Yes, we are already doing this in upgradeToAndCall in our deployment cli tool that we use for upgrade to v3.

INFORMATIONAL-24	New operation	Acknowledged

Description

In the **AvoCore** contract in the **_executeActions** function there are three types of actions: call, delegate call, and flashloan, but there is no operation for contract creation.

Recommendation

It is recommended to add contract creation operation type to the **_executeActions** function.

Client's comments

This is a good idea, we have added it to our backlog. If needed to deploy a contract from the Avo smart wallet in the mean time there would always be solutions via e.g. delegateCall that does this?

INFORMATIONAL-25	Unclear call to IAvoWalletV2	Acknowledged

Description

At the line AvoForwarder.sol#L487, the function **_callTargets()** is being called if the cast execution failed. However, from the code it is unclear why and which function is being called.

Recommendation

We recommend to add comments documenting the reason for the call and the function selector.

Client's comments

We added this code to make sure the status flag on AvoWallets resets for sure even if the action execution via cast reverts.

INFORMATIONAL-26	Race condition for 3 required signers out of 6 signers	Acknowledged

Description

At the line AvoMultisigAdmin.sol#L18, **AvoMultisigAdmin** has **REQUIRED_CONFIRMATIONS = 3** and 6 signers. This creates a race condition when the signers cannot come to a consensus and the signers are split. For example, 2 signers confirm, 2 signers revoke, then the last 2 signers race to confirm or revoke the transaction.

Recommendation

We recommend to set **REQUIRED_CONFIRMATIONS = 4**.

INFORMATIONAL-27	Signer can confirm and revoke the same transaction	Acknowledged

Description

At AvoMultisigAdmin.sol#L119, the signer can confirm a transaction then revoke the same transaction. For example, if 2 signers confirm and revoke the transaction, the third signer will confirm or revoke the transaction either way.

Recommendation

We recommend to disallow confirming and revoking the same transaction.

6. Appendix A. Linter

Error/contract-name-camelcase

- contracts/AvoAuthoritiesList.sol:24 – Contract name must be in CamelCase
- contracts/AvoDepositManager.sol:19 – Contract name must be in CamelCase
- contracts/AvoFactory.sol:25 – Contract name must be in CamelCase
- contracts/AvoForwarder.sol:26 – Contract name must be in CamelCase
- contracts/AvoMultisig/AvoMultisig.sol:51 – Contract name must be in CamelCase
- contracts/AvoSignersList.sol:25 – Contract name must be in CamelCase
- contracts/AvoVersionsRegistry.sol:19 – Contract name must be in CamelCase
- contracts/AvoWallet/AvoWallet.sol:49 – Contract name must be in CamelCase
- contracts/helpers/AvoGasEstimationsHelper.sol:16 – Contract name must be in CamelCase

Error/code-complexity

- contracts/AvoAuthoritiesList.sol:153 – Function has cyclomatic complexity 9 but allowed no more than 5
- contracts/AvoCore/AvoCore.sol:123 – Function has cyclomatic complexity 14 but allowed no more than 5
- contracts/AvoCore/AvoCore.sol:287 – Function has cyclomatic complexity 7 but allowed no more than 5
- contracts/AvoCore/AvoCore.sol:567 – Function has cyclomatic complexity 6 but allowed no more than 5
- contracts/AvoMultisig/AvoMultisig.sol:139 – Function has cyclomatic complexity 10 but allowed no more than 5
- contracts/AvoMultisig/AvoMultisig.sol:244 – Function has cyclomatic complexity 7 but allowed no more than 5
- contracts/AvoMultisig/AvoMultisig.sol:355 – Function has cyclomatic complexity 7 but allowed no more than 5
- contracts/AvoMultisig/AvoMultisig.sol:427 – Function has cyclomatic complexity 9 but allowed no more than 5
- contracts/AvoSignersList.sol:144 – Function has cyclomatic complexity 12 but allowed no more than 5
- contracts/AvoSignersList.sol:222 – Function has cyclomatic complexity 12 but allowed no more than 5
- contracts/AvoWallet/AvoWallet.sol:118 – Function has cyclomatic complexity 6 but allowed no more than 5
- contracts/AvoWallet/AvoWallet.sol:169 – Function has cyclomatic complexity 6 but allowed no more than 5

- contracts/AvoWallet/AvoWallet.sol:437 – Function has cyclomatic complexity 6 but allowed no more than 5

Error/ordering

- contracts/AvoCore/AvoCore.sol:34 – Function order is incorrect, modifier definition can not go after public view function (line 29)
- contracts/AvoCore/AvoCore.sol:507 – Function order is incorrect, external function can not go after external view function (line 502)
- contracts/AvoCore/AvoCore.sol:540 – Function order is incorrect, external payable function can not go after public function (line 532)
- contracts/AvoDepositManager.sol:348 – Function order is incorrect, external view function can not go after public function (line 320)
- contracts/AvoFactory.sol:172 – Function order is incorrect, external view function can not go after public function (line 162)
- contracts/AvoForwarder.sol:226 – Function order is incorrect, internal view function can not go after internal pure function (line 220)
- contracts/AvoForwarder.sol:436 – Function order is incorrect, external payable function can not go after public function (line 416)
- contracts/AvoMultisig/AvoMultisig.sol:355 – Function order is incorrect, external function can not go after public view function (line 333)
- contracts/AvoMultisig/AvoMultisig.sol:540 – Function order is incorrect, external view function can not go after public view function (line 518)
- contracts/AvoMultisig/AvoMultisig.sol:658 – Function order is incorrect, external view function can not go after public view function (line 636)
- contracts/AvoMultisig/AvoMultisig.sol:798 – Function order is incorrect, receive function can not go after public function (line 790)
- contracts/AvoVersionsRegistry.sol:158 – Function order is incorrect, external function can not go after public view function (line 129)
- contracts/AvoVersionsRegistry.sol:193 – Function order is incorrect, external view function can not go after public function (line 182)
- contracts/AvoWallet/AvoWallet.sol:234 – Function order is incorrect, external function can not go after public view function (line 229)
- contracts/AvoWallet/AvoWallet.sol:331 – Function order is incorrect, external view function can not go after public view function (line 309)
- contracts/AvoWallet/AvoWallet.sol:437 – Function order is incorrect, external payable function can not go after

public view function (line 415)

- contracts/AvoWallet/AvoWallet.sol:564 – Function order is incorrect, receive function can not go after public function (line 545)
- contracts/external/IWeth9.sol:20 – Function order is incorrect, receive function can not go after external view function (line 18)
- contracts/helpers/AvoGasEstimationsHelper.sol:38 – Function order is incorrect, state variable declaration can not go after custom error definition (line 31)

Error/not-rely-on-time

- contracts/AvoCore/AvoCore.sol:57 – Avoid making time-based decisions in your business logic
- contracts/AvoCore/AvoCore.sol:197 – Avoid making time-based decisions in your business logic
- contracts/AvoCore/AvoCore.sol:279 – Avoid making time-based decisions in your business logic
- contracts/AvoCore/AvoCore.sol:279 – Avoid making time-based decisions in your business logic
- contracts/AvoCore/AvoCore.sol:640 – Avoid making time-based decisions in your business logic
- contracts/AvoCore/AvoCore.sol:670 – Avoid making time-based decisions in your business logic
- contracts/AvoDepositManager.sol:205 – Avoid making time-based decisions in your business logic

Error/private-vars-leading-underscore

- contracts/AvoCore/AvoCore.sol:666 – '_callTargets' should not start with _
- contracts/AvoCore/AvoCoreVariables.sol:19 – 'DOMAIN_SEPARATOR_NAME_HASHED' should start with _
- contracts/AvoCore/AvoCoreVariables.sol:20 – 'DOMAIN_SEPARATOR_VERSION_HASHED' should start with _
- contracts/AvoCore/AvoCoreVariables.sol:23 – 'CAST_AUTHORIZED_RESERVE_GAS' should start with _
- contracts/AvoCore/AvoCoreVariables.sol:25 – 'CAST_EVENTS_RESERVE_GAS' should start with _
- contracts/AvoCore/AvoCoreVariables.sol:28 – 'IS_MULTISIG' should start with _
- contracts/AvoCore/AvoCoreVariables.sol:104 – 'EIP1271_MAGIC_VALUE' should start with _
- contracts/AvoCore/AvoCoreVariables.sol:117 – 'RESET_BYTES31' should start with _
- contracts/AvoCore/AvoCoreVariables.sol:119 – 'EXECUTE_OPERATION_SELECTOR' should start with _
- contracts/AvoFactory.sol:57 – 'RESET_ADDRESS' should start with _

- contracts/AvoMultiSafe.sol:7 – '_avoMultisigImpl' should not start with _
- contracts/AvoMultiSafe.sol:9 – '_data' should not start with _
- contracts/AvoMultiSafe.sol:11 – '_owner' should not start with _
- contracts/AvoMultisig/AvoMultisigVariables.sol:30 – 'PER_SIGNER_RESERVE_GAS' should start with _
- contracts/AvoSafe.sol:7 – '_avoWalletImpl' should not start with _
- contracts/helpers/AvoGasEstimationsHelper.sol:21 – '_callTargets' should not start with _
- contracts/helpers/AvoGasEstimationsHelper.sol:25 – '_callTargets' should not start with _

Error/var-name-mixedcase

- contracts/AvoCore/AvoCoreVariables.sol:19 – Variable name must be in mixedCase
- contracts/AvoCore/AvoCoreVariables.sol:20 – Variable name must be in mixedCase
- contracts/AvoCore/AvoCoreVariables.sol:23 – Variable name must be in mixedCase
- contracts/AvoCore/AvoCoreVariables.sol:25 – Variable name must be in mixedCase
- contracts/AvoCore/AvoCoreVariables.sol:28 – Variable name must be in mixedCase
- contracts/AvoCore/AvoCoreVariables.sol:31 – Variable name must be in mixedCase
- contracts/AvoCore/AvoCoreVariables.sol:34 – Variable name must be in mixedCase
- contracts/AvoCore/AvoCoreVariables.sol:36 – Variable name must be in mixedCase

Error/max-states-count

- contracts/AvoDepositManager.sol:44 – Contract has 6 states declarations but allowed no more than 3
- contracts/AvoFactory.sol:74 – Contract has 5 states declarations but allowed no more than 3
- contracts/AvoVersionsRegistry.sol:32 – Contract has 4 states declarations but allowed no more than 3
- contracts/AvoWallet/AvoWalletVariables.sol:61 – Contract has 4 states declarations but allowed no more than 3

Error/const-name-snakecase

- contracts/AvoFactory.sol:46 – Constant name must be in capitalized SNAKE_CASE
- contracts/AvoFactory.sol:50 – Constant name must be in capitalized SNAKE_CASE

- contracts/AvoFactory.sol:53 – Constant name must be in capitalized SNAKE_CASE

Error/no-complex-fallback

- contracts/AvoMultiSafe.sol:59 – Fallback function must be simple

7. Appendix B. Slither

High/High/name-reused

AvoForwarderCore is re-used: - AvoForwarderCore - AvoForwarderCore

High/High/uninitialized-state

AvoMultisigVariables.requiredSigners is never initialized. It is used in: - AvoMultisigVariables._setSigners(address[])

High/High/unprotected-upgrade

AvoWallet is an upgradeable contract that does not protect its initiliaze functions: IAvoWalletV3Base.initialize(address)AvoWallet.initialize(address). Anyone can delete the contract with: AvoCoreProtected.executeOperation(address[],uint256[],uint256[],address,bytes)AvoCoreProtected._callTargets(AvoCoreStructs.Action[],uint256)AvoCoreSelfUpgradeable.upgradeToAndCall(address,bytes,bool) AvoMultisig is an upgradeable contract that does not protect its initiliaze functions: IAvoMultisigV3Base.initialize()AvoMultisig.initialize(). Anyone can delete the contract with: AvoCoreProtected.executeOperation(address[],uint256[],uint256[],address,bytes)AvoCoreProtected._callTargets(AvoCoreStructs.Action[],uint256)AvoCoreSelfUpgradeable.upgradeToAndCall(address,bytes,bool)

High/Medium/controlled-delegatecall

AvoCore._executeActions(AvoCoreStructs.Action[],uint256,bool) uses delegatecall to a input-controlled function id - (success_,result_) = action_.target.delegatecall(action_.data)

High/Medium/delegatecall-loop

AvoCore._executeActions(AvoCoreStructs.Action[],uint256,bool) has delegatecall inside a loop in a payable function: (success_,result_) = action_.target.delegatecall(action_.data)

Informational/High/boolean-equal

AvoSignersList.syncRemoveAvoSignerMappings(address,address[]) compares to a boolean constant: - safesPerSigner[removeSigners[0]].remove(avoMultiSafe_) == true

AvoForwarderV1.executeV1(address,IAvoWalletV1.Action[],uint256,uint256,address,bytes,bytes) compares to a boolean constant: -success_ == true

AvoSignersList.syncAddAvoSignerMappings(address,address[]) compares to a boolean constant: - safesPerSigner[addSigners[i]].add(avoMultiSafe_) == true

AvoFactory.isAvoSafe(address) compares to a boolean constant: –Address.isContract(avoSafe_) == false

AvoSignersList.syncAddAvoSignerMappings(address,address[]) compares to a boolean constant: –avoFactory.isAvoSafe(avoMultiSafe_) == false

AvoAuthoritiesList.syncAvoAuthorityMappings(address,address[]) compares to a boolean constant: –safesPerAuthority[authorities[i]].add(avoSafe_) == true

AvoForwarderV2.executeV2(address,IAvoWalletV2.Action[],IAvoWalletV2.CastParams,bytes) compares to a boolean constant: –success_ == true

AvoVersionsRegistry.requireValidAvoForwarderVersion(address) compares to a boolean constant: –avoForwarderVersions[avoForwarderVersion_] != true

AvoVersionsRegistry.requireValidAvoMultisigVersion(address) compares to a boolean constant: –avoMultisigVersions[avoMultisigVersion_] != true

AvoAuthoritiesList.syncAvoAuthorityMappings(address,address[]) compares to a boolean constant: –avoFactory.isAvoSafe(avoSafe_) == false

AvoSignersList.syncRemoveAvoSignerMappings(address,address[]) compares to a boolean constant: –safesPerSigner[removeSigners[i]].remove(avoMultiSafe_) == true

AvoForwarderMultisig.executeMultisigV3(address,AvoCoreStructs.CastParams,AvoCoreStructs.CastForwardParams,AvoCoreStructs.SignatureParams[]) compares to a boolean constant: –success_ == true

AvoWalletCastAuthorized.castAuthorized(AvoCoreStructs.CastParams,AvoCoreStructs.CastAuthorizedParams) compares to a boolean constant: –success_ == true

AvoAuthoritiesList.syncAvoAuthorityMappings(address,address[]) compares to a boolean constant: –safesPerAuthority[authorities[i]].remove(avoSafe_) == true

AvoForwarderV3.executeV3(address,AvoCoreStructs.CastParams,AvoCoreStructs.CastForwardParams,AvoCoreStructs.SignatureParams) compares to a boolean constant: –success_ == true

AvoWalletCast.cast(AvoCoreStructs.CastParams,AvoCoreStructs.CastForwardParams,AvoCoreStructs.SignatureParams) compares to a boolean constant: –success_ == true

AvoDepositManagerCore.onlyAvoSafe(address) compares to a boolean constant: –avoFactory.isAvoSafe(address_) == false

AvoSignersList.syncAddAvoSignerMappings(address,address[]) compares to a boolean constant: –safesPerSigner[addSigners[0]].add(avoMultiSafe_) == true

AvoSignersList.syncRemoveAvoSignerMappings(address,address[]) compares to a boolean constant: –avoFactory.isAvoSafe(avoMultiSafe_) == false

AvoVersionsRegistry.requireValidAvoWalletVersion(address) compares to a boolean constant: –avoWalletVersions[avoWalletVersion_] != true

AvoMultisigCastAuthorized.castAuthorized(AvoCoreStructs.CastParams,AvoCoreStructs.CastAuthorizedParams,AvoCoreStructs.SignatureParams[]) compares to a boolean constant: –success_ == true

AvoMultisigCast.cast(AvoCoreStructs.CastParams,AvoCoreStructs.CastForwardParams,AvoCoreStructs.SignatureParams[]) compares to a boolean constant: -success_ == true

Informational/High/low-level-calls

Low level call in AvoForwarderV2.executeV2(address,IAvoWalletV2.Action[],IAvoWalletV2.CastParams,bytes): - [(address(avoWallet_)).call(abi.encodeWithSelector(bytes4(0xb92e87fa),new IAvoWalletV2.Action,0))](https://github.com/instadapp/avocado-contracts/blob/0eec33a45784e5f4b3b45ac2c9dfefc5225bda0a/contracts/AvoForwarder.sol#L487)

Low level call in AvoCore._executeActions(AvoCoreStructs.Action[],uint256,bool): - (success_,result_) = action_.target.call{value: action_.value}(action_.data) - (success_,result_) = action_.target.delegatecall(action_.data) - (success_,result_) = action_.target.call{value: action_.value}(action_.data)

Low level call in AvoGasEstimationsHelper.estimateCallTargetsGas(address,AvoCoreStructs.Action[],uint256): - (success_,result_) = address(avoWallet_).call{value: msg.value}(abi.encodeCall(avoWallet_.callTargets,(actions,id_)))

Low level call in AvoGasEstimationsHelper.estimateCallTargetsGasMultisig(address,AvoCoreStructs.Action[],uint256): - (success_,result_) = address(avoMultisig_).call{value: msg.value}(abi.encodeCall(avoMultisig_.callTargets,(actions,id_)))

Low level call in AvoWalletAuthorities.removeAuthorities(address[]): - (success_) = address(avoAuthoritiesList).call(abi.encodeCall(IAvoAuthoritiesList.syncAvoAuthorityMappings,(address(this),authorities_)))

Low level call in AvoCore._payAuthorizedFee(uint256,uint256): - (success_,result_) = address(avoVersionsRegistry).staticcall(abi.encodeWithSignature(calcFee(uint256),gasUsed_)) - (success_) = feeCollector_.call{gas: 22_000,value: feeAmount_}()

Low level call in AvoGasEstimationsHelper.estimateCallTargetsGasWithVersionMultisig(address,AvoCoreStructs.Action[],uint256,address): - (success_,result_) = address(avoMultisig_).call{value: msg.value}(abi.encodeCall(avoMultisig_.callTargets,(actions,id_)))

Low level call in AvoSafe.constructor(): - (success_,data_) = msg.sender.call(bytes(0x8e7daf69))

Low level call in AvoMultiSafe.constructor(): - (deployData_) = msg.sender.staticcall(bytes(ŋ))

Low level call in AvoWalletAuthorities.addAuthorities(address[]): - (success_) = address(avoAuthoritiesList).call(abi.encodeCall(IAvoAuthoritiesList.syncAvoAuthorityMappings,(address(this),authorities_)))

Low level call in AvoGasEstimationsHelper.estimateCallTargetsGasWithVersion(address,AvoCoreStructs.Action[],uint256,address): - (success_,result_) = address(avoWallet_).call{value: msg.value}(abi.encodeCall(avoWallet_.callTargets,(actions,id_)))

Low level call in AvoMultisigSigners.removeSigners(address[]): - (success_) = address(avoSignersList).call(abi.encodeCall(IAvoSignersList.syncRemoveAvoSignerMappings,(address(this),removeSigners_)))

Low level call in AvoMultisigCore._initialize(): - (success_) = address(avoSignersList).call(abi.encodeCall(IAvoSignersList.syncAddAvoSignerMappings,(address(this),signers_)))

Low level call in AvoMultisigSigners.addSigners(address[]): - (success_) = address(avoSignersList).call(abi.encodeCall(IAvoSignersList.syncAddAvoSignerMappings,(address(this),addSigners_)))

Informational/High/unused-state

AvoCoreSlotGaps.__gaps is never used in AvoMultisigInitializable

AvoAuthoritiesListVariables.__gap is never used in AvoAuthoritiesList

AvoWalletVariablesSlot1.__slot1Placeholder is never used in AvoWallet

AvoCoreConstants._CALL_TARGETS_SELECTOR is never used in AvoMultisigInitializable

AvoMultisigConstants.PER_SIGNER_RESERVE_GAS is never used in AvoMultisigInitializable

AvoCoreVariablesSlot0._avoImplementation is never used in AvoMultisigInitializable

AvoCoreTransient._transientId is never used in AvoMultisigInitializable

AvoCoreConstants.EXECUTE_OPERATION_SELECTOR is never used in AvoMultisigInitializable

AvoCoreConstants.EIP1271_MAGIC_VALUE is never used in AvoMultisigInitializable

AvoMultisigVariablesSlot1.__slot1Placeholder is never used in AvoMultisigInitializable

AvoCoreConstants.RESET_BYTES31 is never used in AvoMultisigInitializable

AvoSignersListVariables.__gap is never used in AvoSignersList

AvoWalletVariablesSlot0.__slot0Placeholder is never used in AvoWallet

AvoCoreTransient._transientAllowHash is never used in AvoMultisigInitializable

AvoCoreVariablesSlot0._avoSafeNonce is never used in AvoMultisigInitializable

AvoCoreVariablesSlot2._signedMessages is never used in AvoMultisigInitializable

AvoFactoryVariables.__gaps is never used in AvoFactory

AvoCoreSlotGaps.__gaps is never used in AvoWallet

AvoCoreSlotGaps.__gaps is never used in AvoMultisig

AvoMultisigVariablesSlot1.__slot1Placeholder is never used in AvoMultisig

Informational/Medium/dead-code

AvoMultisigVariables._getSigners() is never used and should be removed

AvoWalletInitializable._getInitializedVersion() is never used and should be removed

AvoMultisigInitializable._isInitializing() is never used and should be removed

AvoMultisigInitializable._disableInitializers() is never used and should be removed

AvoMultisigInitializable._getInitializedVersion() is never used and should be removed

AvoWalletInitializable._isInitializing() is never used and should be removed

AvoMultisigVariables._setSigners(address[]) is never used and should be removed

Low/High/shadowing-local

IAvoMultisigV3Base.signers().signers shadows: - IAvoMultisigV3Base.signers() (function)

Low/High/variable-scope

Variable 'AvoFactory.isAvoSafe(address).owner_' in AvoFactory.isAvoSafe(address) potentially used before declaration: computedAddress_ = computeAddress(owner_)

Variable 'AvoFactory.isAvoSafe(address).owner_' in AvoFactory.isAvoSafe(address) potentially used before declaration: computedAddress_ = computeAddressMultisig(owner_)

Low/Medium/calls-loop

AvoAuthoritiesList.syncAvoAuthorityMappings(address,address[]) has external calls inside a loop: isAuthority_ = IAvoWalletV3(avoSafe_).isAuthority(authorities_[i])

Medium/High/locked-ether

Contract locking ether found: Contract AvoSafe has payable functions: - AvoSafe.fallback() But does not have a function to withdraw the ether

Contract locking ether found: Contract AvoMultiSafe has payable functions: - AvoMultiSafe.fallback() But does not have a function to withdraw the ether

Medium/High/write-after-write

AvoFactoryVariables._transientDeployVersion is written in both transientDeployVersion = avoMultisigVersion _transientDeployVersion = RESET_ADDRESS

AvoFactoryVariables._transientDeployOwner is written in both transientDeployOwner = owner _transientDeployOwner = RESET_ADDRESS

AvoFactoryVariables._transientDeployVersion is written in both _transientDeployVersion = avoMultisigImpl _transientDeployVersion = RESET_ADDRESS

AvoFactoryVariables._transientDeployOwner is written in both transientDeployOwner = owner _transientDeployOwner = RESET_ADDRESS

Medium/Medium/unchecked-lowlevel

AvoForwarderV2.executeV2(address,IAvoWalletV2.Action[],IAvoWalletV2.CastParams,bytes) ignores return value by [(address(avoWallet_)).call(abi.encodeWithSelector(bytes4(0xb92e87fa),new IAvoWalletV2.Action,0))] (https://github.com/instadapp/avocado-contracts/blob/0eec33a45784e5f4b3b45ac2c9dfefc5225bda0a/contracts/AvoForwarder.sol#L487)

Medium/Medium/uninitialized-local

AvoSignersList.syncRemoveAvoSignerMappings(address,address[]).lastSkipSignerIndex_ is a local variable never initialized

AvoWalletCastAuthorized.castAuthorized(AvoCoreStructs.CastParams,AvoCoreStructs.CastAuthorizedParams).nonSequentialNonce_ is a local variable never initialized

AvoWalletAuthorities.removeAuthorities(address[]).i is a local variable never initialized

AvoCore._getSigDigest(AvoCoreStructs.CastParams,bytes32,bytes32).i is a local variable never initialized

AvoForwarder.reinitialize(address,address[]).i is a local variable never initialized

AvoCoreProtected.occupyNonSequentialNonces(bytes32[]).i is a local variable never initialized

AvoSignersList.syncRemoveAvoSignerMappings(address,address[]).i is a local variable never initialized

AvoSignersList.syncAddAvoSignerMappings(address,address[]).lastAllowedSignerIndex_ is a local variable never initialized

AvoCore._payAuthorizedFee(uint256,uint256).success__scope_0 is a local variable never initialized

AvoForwarderOwnerActions.updateAuths(AvoForwarderStructs.AddressBool[]).i is a local variable never initialized

AvoFactory.isAvoSafe(address).owner_ is a local variable never initialized

AvoForwarderOwnerActions.updateBroadcasters(AvoForwarderStructs.AddressBool[]).i is a local variable never initialized

AvoWalletAuthorities.addAuthorities(address[]).i is a local variable never initialized

AvoAuthoritiesList.syncAvoAuthorityMappings(address,address[]).i is a local variable never initialized

AvoSignersList.syncAddAvoSignerMappings(address,address[]).i is a local variable never initialized

AvoCoreProtected.occupyAvoSafeNonces(uint88[]).i is a local variable never initialized

AvoMultisigCore.verifySig(bytes32,AvoCoreStructs.SignatureParams[],bool).isContract is a local variable never initialized

AvoMultisigSigners.addSigners(address[]).i is a local variable never initialized

AvoMultisigSigners.addSigners(address[]).addedCount_ is a local variable never initialized

AvoMultisigCore._verifySig(bytes32,AvoCoreStructs.SignatureParams[],bool).i is a local variable never initialized

AvoMultisigSigners.isSigner(address).i is a local variable never initialized

AvoMultisigSigners.removeSigners(address[]).i is a local variable never initialized

AvoMultisigCore.verifySig(bytes32,AvoCoreStructs.SignatureParams[],bool).lastAllowedSignerIndex is a local variable never initialized

AvoMultisigSigners.removeSigners(address[]).removedCount_ is a local variable never initialized

AvoMultisigCore.verifySig(bytes32,AvoCoreStructs.SignatureParams[],bool).isAllowedSigner is a local variable never initialized

Medium/Medium/unused-return

AvoAuthoritiesList.syncAvoAuthorityMappings(address,address[]) ignores return value by authoritiesPerSafe[avoSafe].remove(authorities_[i])

AvoCoreSelfUpgradeable.upgradeToAndCall(address,bytes,bool) ignores return value by Address.functionDelegateCall(avoImplementation_,data_)

AvoAuthoritiesList.syncAvoAuthorityMappings(address,address[]) ignores return value by authoritiesPerSafe[avoSafe].add(authorities_[i])

AvoFactory.isAvoSafe(address) ignores return value by IAvoWalletV3(avoSafe_).owner()

Optimization/High/constable-states

AvoWalletVariablesSlot1.__slot1Placeholder should be constant

AvoWalletVariablesSlot0.__slot0Placeholder should be constant

AvoMultisigVariablesSlot1.__slot1Placeholder should be constant

AvoMultisigVariables.requiredSigners should be constant

Optimization/High/external-function

avoMultiSafes(address) should be declared external: – AvoSignersListViews.avoMultiSafes(address)

signersCount(address) should be declared external: – AvoSignersListViews.signersCount(address)

authorities(address) should be declared external: – AvoAuthoritiesListViews.authorities(address)

reinitialize() should be declared external: – AvoFactory.reinitialize()

isAuthority(address) should be declared external: – AvoWalletAuthorities.isAuthority(address)

reinitialize(address,address[]) should be declared external: – AvoForwarder.reinitialize(address,address[])

initialize(address,address,uint96,uint96,uint96) should be declared external: – AvoDepositManager.initialize(address,address,uint96,uint96,uint96)

avoSafes(address) should be declared external: – AvoAuthoritiesListViews.avoSafes(address)

requireValidAvoForwarderVersion(address) should be declared external: – AvoVersionsRegistry.requireValidAvoForwarderVersion(address)

initialize() should be declared external: – AvoFactory.initialize()

avoSafesCount(address) should be declared external: – AvoAuthoritiesListViews.avoSafesCount(address)

calcFee(uint256) should be declared external: – AvoFeeCollector.calcFee(uint256)

initializeWithVersion(address,address) should be declared external: – AvoWallet.initializeWithVersion(address,address)

avoSafeNonce() should be declared external: – AvoCore.avoSafeNonce()

initialize(address) should be declared external: – AvoForwarder.initialize(address)

authoritiesCount(address) should be declared external: – AvoAuthoritiesListViews.authoritiesCount(address)

initialize(address) should be declared external: – AvoVersionsRegistry.initialize(address)

isSignerOf(address,address) should be declared external: – AvoSignersListViews.isSignerOf(address,address)

reinitialize() should be declared external: – AvoVersionsRegistry.reinitialize()

avoMultiSafesCount(address) should be declared external: – AvoSignersListViews.avoMultiSafesCount(address)

reinitialize() should be declared external: – AvoWallet.reinitialize()

isAuthorityOf(address,address) should be declared external: – AvoAuthoritiesListViews.isAuthorityOf(address,address)

domainSeparatorV4() should be declared external: – AvoWallet.domainSeparatorV4()

initialize(address) should be declared external: – AvoWallet.initialize(address)

signers(address) should be declared external: – AvoSignersListViews.signers(address) domainSeparatorV4() should be declared external: – AvoMultisig.domainSeparatorV4()

initialize() should be declared external: – AvoMultisig.initialize()

signers() should be declared external: – AvoMultisig.signers()

owner() should be declared external: – AvoMultisig.owner()

isSigner(address) should be declared external: – AvoMultisigSigners.isSigner(address)

8. Appendix C. Tests

Tests result

```
943 passing (8m)
```

Tests coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	96.25	93.51	98.35	96.4	
AvoAdmin.sol	0	100	0	0	10
AvoAuthoritiesList.sol	96.55	96.15	100	97.67	91
AvoAuthoritiesListProxy.sol	100	100	100	100	
AvoDepositManager.sol	100	95.59	100	100	
AvoDepositManagerProxy.sol	100	100	100	100	
AvoFactory.sol	100	97.06	100	98.44	68
AvoFactoryProxy.sol	100	100	100	100	
AvoForwarder.sol	91	87.5	100	91.2	... 489,674,696
AvoForwarderProxy.sol	100	100	100	100	
AvoMultiSafe.sol	100	100	100	100	
AvoSafe.sol	100	50	50	71.43	39,46
AvoSignersList.sol	100	91.67	100	100	
AvoSignersListProxy.sol	100	100	100	100	
AvoVersionsRegistry.sol	100	100	100	100	
AvoVersionsRegistryProxy.sol	100	100	100	100	
contracts/AvoCore/	96.91	90.74	100	96.76	
AvoCore.sol	96.81	90.82	100	96.47	... 441,442,496

AvoCoreErrors.sol	100	100	100	100	
AvoCoreEvents.sol	100	100	100	100	
AvoCoreStructs.sol	100	100	100	100	
AvoCoreVariables.sol	100	90	100	100	
contracts/AvoMultisig/	99.24	88.46	100	97.22	
AvoMultisig.sol	100	93.62	100	98.04	96,261,467,485
AvoMultisigErrors.sol	100	100	100	100	
AvoMultisigEvents.sol	100	100	100	100	
AvoMultisigVariables.sol	85.71	40	100	83.33	143,155
contracts/AvoMultisig/lib/	61.54	43.75	33.33	56.52	
AvoMultisigInitializable.sol	61.54	43.75	33.33	56.52	... 137,160,167
contracts/AvoWallet/	100	97.14	100	100	
AvoWallet.sol	100	97.06	100	100	
AvoWalletErrors.sol	100	100	100	100	
AvoWalletEvents.sol	100	100	100	100	
AvoWalletVariables.sol	100	100	100	100	
contracts/AvoWallet/lib/	76.92	56.25	50	82.61	
AvoWalletInitializable.sol	76.92	56.25	50	82.61	136,137,160,167
contracts/external/	100	100	100	100	
ICREATE3Factory.sol	100	100	100	100	
IWeth9.sol	100	100	100	100	
InstaFlashAggregatorInterface.sol	100	100	100	100	
InstaFlashReceiverInterface.sol	100	100	100	100	
contracts/helpers/	100	76.47	100	89.33	

contracts/interfaces/	100	100	100	100	
IAvoAuthoritiesList.sol	100	100	100	100	
IAvoFactory.sol	100	100	100	100	
IAvoForwarder.sol	100	100	100	100	
IAvoMultisigV3.sol	100	100	100	100	
IAvoSignersList.sol	100	100	100	100	
IAvoVersionsRegistry.sol	100	100	100	100	
IAvoWalletV1.sol	100	100	100	100	
IAvoWalletV2.sol	100	100	100	100	
IAvoWalletV3.sol	100	100	100	100	
contracts/mocks/	80.77	44.44	86.05	80.61	
EmptyImplementation.sol	100	100	100	100	
MockDelegateCallTargetMultisig.sol	75	25	80	80.56	... 106,110,111
MockDelegateCallTargetWallet.sol	78.57	50	84.62	80	... 76,85,90,95
MockDeposit.sol	100	100	100	100	
MockERC1967Proxy.sol	100	100	100	100	
MockERC20Token.sol	100	100	100	100	
MockERC721Token.sol	100	100	100	100	
MockFailingFeeCollector.sol	100	100	100	100	
MockSigner.sol	100	100	100	100	
MockSignerArbitrarySigLength.sol	100	50	100	100	
MockWETH.sol	66.67	33.33	75	60	... 27,28,43,44
------------------------------------ –	--------- –	--------- –	--------- –	--------- –	--------------- –
All files	95.35	88.43	94.02	94.13	

STATE
MIND