CERTIK

Security Assessment

JOJO-reaudit

CertiK Verified on Jan 9th, 2023

CertiK Verified on Jan 9th, 2023

JOJO-reaudit

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Ethereum (ETH)	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 01/09/2023	N/A

Vulnerability Summary

7	1	0	1	5	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
1	Major	1 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
2	Medium	2 Acknowledged	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
1	Minor	1 Partially Resolved	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
3	Informational	1 Resolved, 2 Acknowledged	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | JOJO-REAUDIT

Summary

Executive Summary

Vulnerability Summary

Codebase

Audit Scope

Approach & Methods

Findings

EVM-01 : Centralization Risks

EVJ-01 : Potential Reentrancy Attack

OEJ-01 : Logic issue In `setPerpRiskParams()`

EVJ-02 : Missing Zero Address Validation

EMJ-01 : Wrong Comments

FEV-01 : Incompatibility with Deflationary Tokens

OEV-01 : Unused Contract

Optimizations

SEV-01 : Comparison to Boolean Constant

Appendix

Disclaimer

AUDIT SCOPE | JOJO-REAUDIT

25 files audited ● 6 files with Acknowledged findings ● 2 files with Partially Resolved findings
● 2 files with Resolved findings ● 15 files without findings

ID	File	SHA256 Checksum
OEV	contracts/adaptor/constOracle.sol	73bf7eccf9f29d63f4cae57e73f02a68bfaf8f243a96b00dea705f169a0f415e
OEM	contracts/adaptor/emergencyOracle.sol	3da1a7b194f13353835b74202f60d1e93197fc152540fab36c7cf3e2de946166
FRU	contracts/fundingRateKeeper/FundingRateUpdateLimiter.sol	45813e3ea32f9de446ba28ded666a11ffe5f59a97894319a2a01812013ae546d
PEV	contracts/impl/Perpetual.sol	aa10d24d771f20e8e7c43adb75cd8281b003b8edd718087ec32f23d34f0cd1cf
FEV	contracts/lib/Funding.sol	258de419c817c81f6aeab232ea830e39bd11e380502cafe30d35706545eb9df0
OEJ	contracts/lib/Operation.sol	03dcc009ac491931d1f06e4bbd0197f1cb2b05502b094041aced0efbcb2e9db7
AEV	contracts/adaptor/chainlinkAdaptor.sol	206d821a6c8c97ed09bf1f9ad935a94dc8ef2a26639c5ad6c9dc322b24cd1a18
SEV	contracts/subaccount/Subaccount.sol	1c85ab57fdd82109a7d2ca130984cba125ab9f1d7b57216a429326681fa989be
TEV	contracts/lib/Trading.sol	db034a64ba689c3c4572f2dcbc6569b5be5ad7ffa2e447d95f33a80ec5c74706
TEM	contracts/lib/Types.sol	6a582c7d0531f119a3844c151b632bb5bdcee8c2c05abc7a4097b54f6bae9892
JOJ	contracts/impl/JOJODealer.sol	b82dd416bd1d41d8d98cba22e42065eafa340f8bda82962e17755836d0e29e69
JOO	contracts/impl/JOJOExternal.sol	48b64e366689925fdbeed1e45e77bcb3d39284ee6a8f1b2e359c150cfee643ea
JOE	contracts/impl/JOJOOperation.sol	e74c0e42dcabdc8f6721d30adec5b3763c20738d8517b8f22fd97f62b23ab223
JOS	contracts/impl/JOJOStorage.sol	4c55de87b588e2794cc926b91d9baf2242290a71df2c6b4552521788cddbfe2a

ID	File	SHA256 Checksum
JOV	contracts/impl/JOJOView.sol	c5fe567c2d1b9f3744d40e5032960a173678ab8d7e138fe49fc793ed93042b97
IDE	contracts/intf/IDealer.sol	616045c5cc356de6a9bcaab37b44036294a4af83fa1373140aa0095d3743ad89
IDR	contracts/intf/IDecimalERC20.sol	0d3ce2265048d422279b1f80115d3823707e2ead7cd77e8a52f2e444229a6cd4
IMP	contracts/intf/IMarkPriceSource.sol	502ce5041c08cc9b5bb0b4657c8eae76e0ff88ca60c7e630eeaebf15705a11aa
IPE	contracts/intf/IPerpetual.sol	97c53ab14cc0fe4f94e1c91fa39d29905b7324a6598fc41c65ef5f678d7c6523
EIP	contracts/lib/EIP712.sol	e48ccaa07de9d498cdbc1dc901366bc11ea8c1fc7c226babfe46dba125b7e4a2
LEV	contracts/lib/Liquidation.sol	45fd190de35bf4b062cb96edc161b51a43285b766433ea671ea234d40dce1387
PEM	contracts/lib/Position.sol	b63882e7e6f248196cd1b3f93ca79170e8e71741b60ba00ad2652871f59bdf27
SFE	contracts/subaccount/SubaccountFactory.sol	5838791c716cae71c727d39cf1ba8541bc0ee3fd42aebc17710189293dab1ffe
EEV	contracts/utils/Errors.sol	6b7ce762d7f7aaa7494045035debb8d17214fa8f3375a69ffd2c4dc0d04000ae
SDM	contracts/utils/SignedDecimalMath.sol	ac7f29a2b3f892ac7b700ad337098d3f1f75898a00b527dc7b4e70f6475e995c

APPROACH & METHODS | JOJO-REAUDIT

This report has been prepared for JOJO to discover issues and vulnerabilities in the source code of the JOJO-reaudit project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

FINDINGS | JOJO-REAUDIT

7	0	1	2	1	3
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for JOJO-reaudit. Through this audit, we have uncovered 7 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

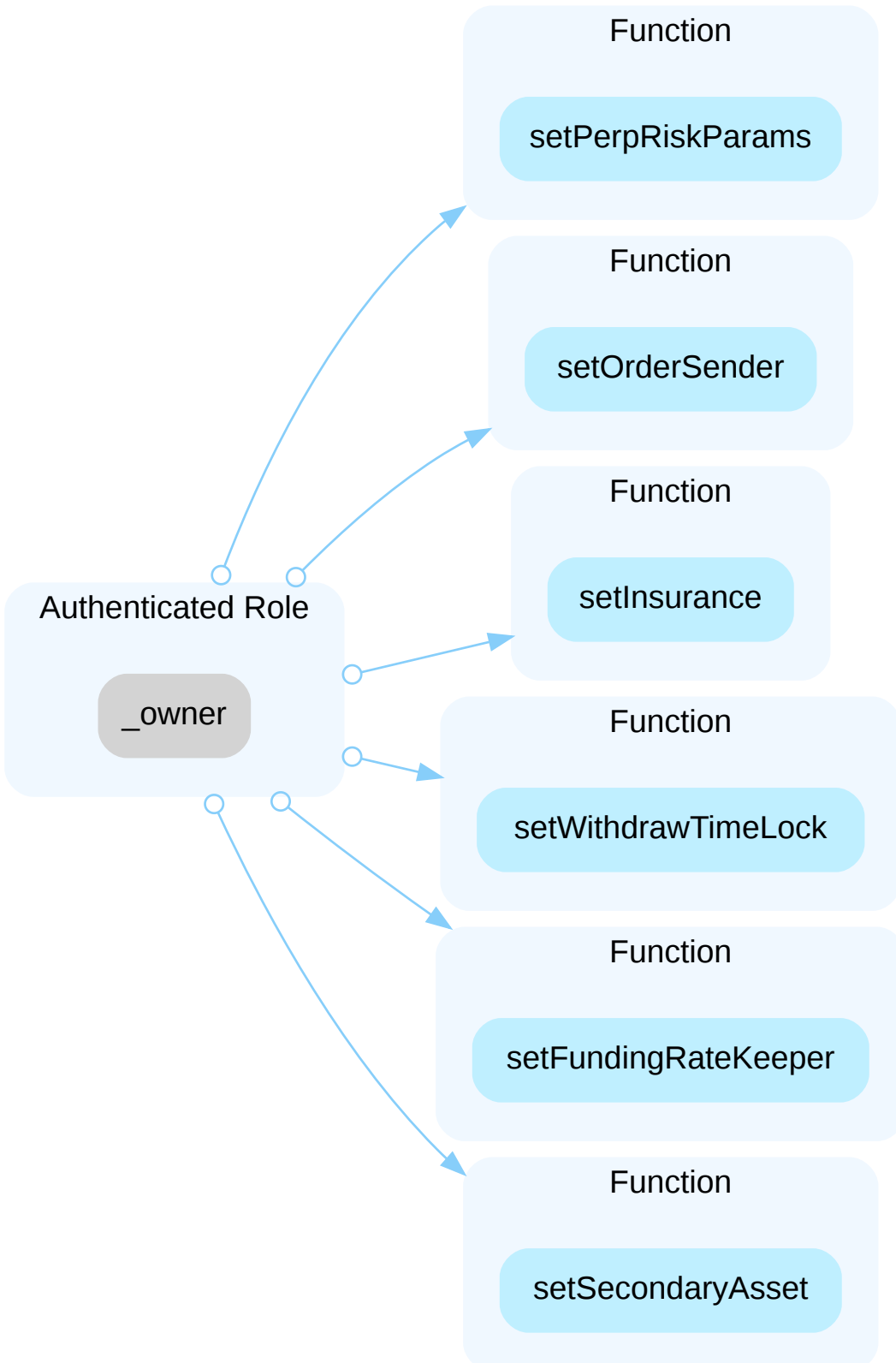
ID	Title	Category	Severity	Status
EVM-01	**Centralization Risks**	**Centralization / Privilege**	**Major**	**Acknowledged**
EVJ-01	Potential Reentrancy Attack	Volatile Code	Medium	Acknowledged
OEJ-01	Logic Issue In `setPerpRiskParams()`	Logical Issue	Medium	Acknowledged
EVJ-02	Missing Zero Address Validation	Volatile Code	Minor	Partially Resolved
EMJ-01	Wrong Comments	Inconsistency	Informational	Resolved
FEV-01	Incompatibility With Deflationary Tokens	Logical Issue	Informational	Acknowledged
OEV-01	Unused Contract	Coding Style	Informational	Acknowledged

EVM-01 | CENTRALIZATION RISKS

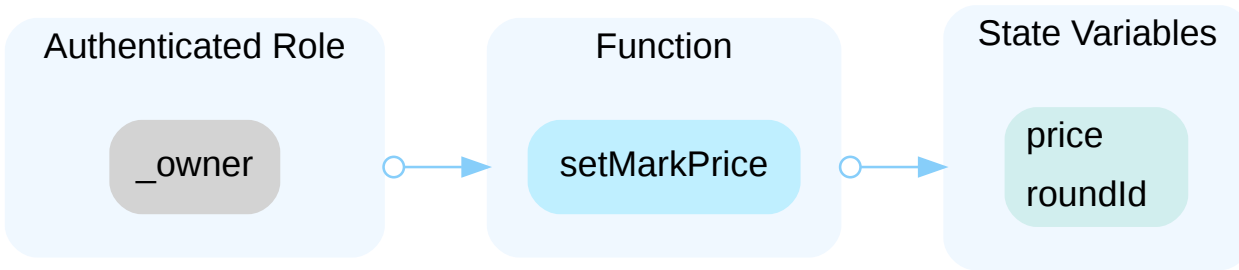
Category	Severity	Location	Status
Centralization / Privilege	**Major**	**contracts/adaptor/emergencyOracle.sol: 35; contracts/fundingRateKeeper/FundingRateUpdateLimiter.sol: 38; source/contracts/impl/JOJOOperation.sol: 33, 40, 44, 48, 55, 64**	**Acknowledged**

Description

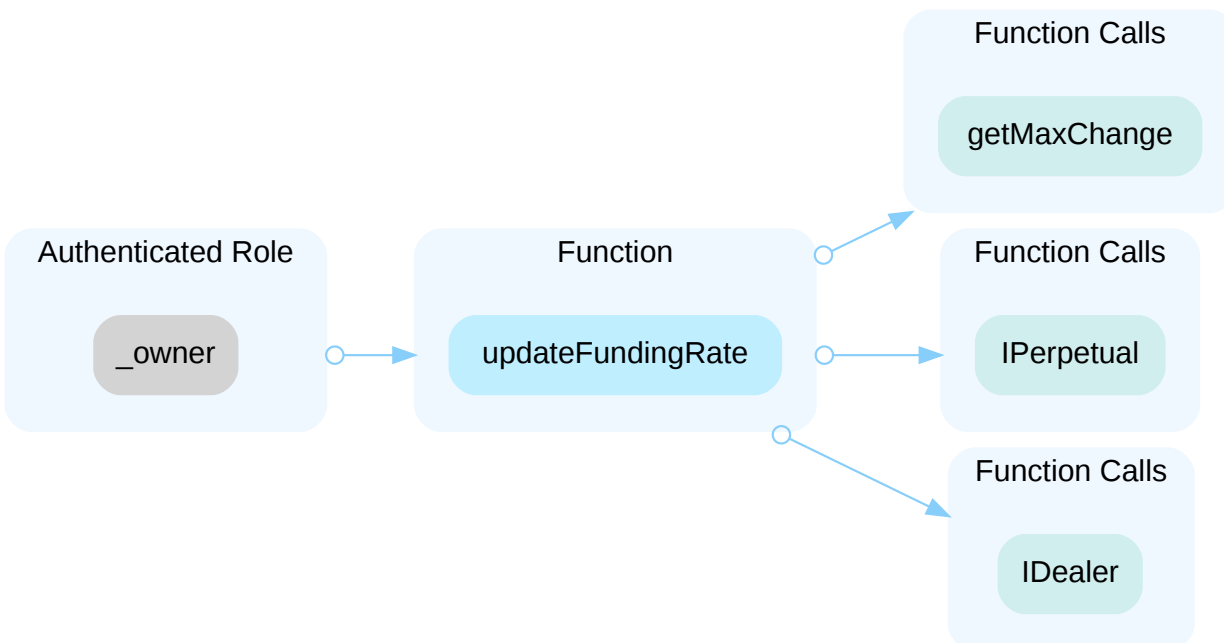
In the contract `JOJOOperation` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and extract all the funds via setting of bad `RiskParams` (fake Oracle, unexpected `liquidationThreshold` and `insuranceFeeRate`).



In the contract `EmergencyOracle` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and maliciously manipulate emergency prices.



In the contract `FundingRateUpdateLimiter` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and maliciously manipulate the FundingRate of perpetual.



Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

[JOJO Team]:

1. The perpetual's owner will always be JOJODealer, so no worry about it.
2. Subaccount's owner will be the one who created it. It's totally permissionless and won't influence JOJO's trading system.
3. FundingRateKeeper will be an EOA account managed by JOJO's team. We admit it is centralized by design.
4. And the owner of JOJOOperation (it is also the owner of JOJODealer) will be a 2of3 gnosis safe wallet. Will provide the address before the product launch.

EVJ-01 | POTENTIAL REENTRANCY ATTACK

Category	Severity	Location	Status
Volatile Code	Medium	contracts/impl/Perpetual.sol: 128~138, 164, 165, 199, 200, 203, 207~210, 211; contracts/lib/Funding.sol: 141, 145, 147; contracts/lib/Operation.sol: 142~146, 147	Acknowledged

Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

External call(s)

```
128     (
129         liqtorPaperChange,
130         liqtorCreditChange,
131         liqedPaperChange,
132         liqedCreditChange
133     ) = IDealer(owner()).requestLiquidation(
134         msg.sender,
135         liquidator,
136         liquidatedTrader,
137         requestPaper
138     );
```

```
164     _settle(liquidatedTrader, liqedPaperChange, liqedCreditChange);
```

- This function call executes the following external call(s).
- In `Perpetual._settle` ,
  - `IDealer(owner()).openPosition(trader)`
- In `Perpetual._settle` ,
  - `IDealer(owner()).realizePnl(trader,balanceMap[trader].reducedCredit)`

```
165     _settle(liquidator, liqtorPaperChange, liqtorCreditChange);
```

State variables written after the call(s)

```
165    _settle(liquidator, liqtorPaperChange, liqtorCreditChange);
```

- This function call executes the following assignment(s).
- In `Perpetual._settle`,
  - `balanceMap[trader].paper = newPaper`
- In `Perpetual._settle`,
  - `balanceMap[trader].reducedCredit = newReducedCredit`
- In `Perpetual._settle`,
  - `balanceMap[trader].reducedCredit = 0`

External call(s)

```
141    IERC20(state.primaryAsset).safeTransfer(to, primaryAmount);
```

State variables written after the call(s)

```
145    state.secondaryCredit[payer] -= secondaryAmount;
```

```
147    state.secondaryCredit[to] += secondaryAmount;
```

External call(s)

```
142    require(
143      IDecimalERC20(_secondaryAsset).decimals() ==
144        IDecimalERC20(state.primaryAsset).decimals(),
145      Errors.SECONDARY_ASSET_DECIMAL_WRONG
146    );
```

State variables written after the call(s)

```
147    state.secondaryAsset = _secondaryAsset;
```

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

[JOJO Team]:

1. IDealer and Perpetual are all from our own project. So the contract can be trusted.
2. The primaryAsset is default by USDC ERC20 standard token, so we can trust this contract.
3. The decimals() function of secondaryAsset is a view function, and we can trust this contract.

In summary, we believe that there is no possibility of reentrancy attack.

OEJ-01 | LOGIC ISSUE IN `setPerpRiskParams()`

Category	Severity	Location	Status
Logical Issue	Medium	contracts/lib/Operation.sol: 46	Acknowledged

Description

There are two puzzling problems in the function `setPerpRiskParams` .

1. It is very strange that when the condition on line 51 is met, removing the current `perp` from `registeredPerp`, in the very common case, the function will do nothing and exit. Even in this case, there is no check on the perp contract to be deleted.
2. It makes more sense to check the validity of `param' beforehand than to assign it directly to the perp contract.

Recommendation

We suggest to review the logic of this method and check if there is still an open position when removing a perp contract.

Alleviation

[JOJO Team]: When the condition on line 51 is met, it means the system is removing the perp. Removing a perp corresponds to the process of delist a trading pair. In JOJO system, delist is not a liquidation process. It neither closes trading nor prohibit opening and closing of positions. The market is no longer changing and any one can close his position at anytime, with the liquidity provided by JOJO.

EVJ-02 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	contracts/adaptor/chainlinkAdaptor.sol: 36; contracts/fundingRateKeeper/FundingRateUpdateLimiter.sol: 34; contracts/subaccount/Subaccount.sol: 43, 47	Partially Resolved

Description

Addresses should be checked before assignment or external call to make sure they are not zero addresses.

```
36          chainlink = _chainlink;
```

- `_chainlink` is not zero-checked before being used.

```
34          dealer = _dealer;
```

- `_dealer` is not zero-checked before being used.

```
43          owner = _owner;
```

- `_owner` is not zero-checked before being used.

```
47          (bool success, bytes memory returnData) = to.call{value: value}(data);
```

- `to` is not zero-checked before being used.

Recommendation

We advise adding a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

[JOJO Team]:

1. In the business logic, all subaccounts is created by subaccountFactory. and the owner of subaccount must be an EOA owned by someone, no need to check.
2. We will make sure the chanlink registered in `chainlinkAdaptor` correct.
3. we will make sure the dealer registere in `FundingRateUpdateLimiter` correct.
4. For the to address, this is the submission: https://github.com/JOJOexchange/smart-contract-EVM/commit/68a1a6d03c5a48085e322cd18f9dbdd5f725a516

EMJ-01 | WRONG COMMENTS

Category	Severity	Location	Status
Inconsistency	Informational	contracts/lib/Trading.sol: 45; contracts/lib/Types.sol: 66	Resolved

Description

```
44      /// at least 1 maker order.
45      /// orderList[0] is taker order and all others are taker orders.
```

It should be "orderList[0] is taker order and all others are maker orders."

```
66          // negative(positive) if you want to open short(long) position
67          int128 creditAmount;
```

It should be "negative(positive) if you want to open long(short) position".

Recommendation

We recommend updating the mentioned comments.

Alleviation

The team heeded the advice and resolved this issue in commit `f2600a1b3fa43a27de5e0f44b8a31b5fbe054c22`.

FEV-01 | INCOMPATIBILITY WITH DEFLATIONARY TOKENS

Category	Severity	Location	Status
Logical Issue	Informational	contracts/lib/Funding.sol: 73~77, 78	Acknowledged

Description

When transferring deflationary ERC20 tokens, the input amount may not be equal to the received amount due to the charged transaction fee. For example, if a user sends 100 deflationary tokens (with a 10% transaction fee), only 90 tokens actually arrived to the contract. However, a failure to discount such fees may allow the same user to withdraw 100 tokens from the contract, which causes the contract to lose 10 tokens in such a transaction.

Reference: https://thoreum-finance.medium.com/what-exploit-happened-today-for-gocerberus-and-garuda-also-for-lokum-ybear-piggy-caramelswap-3943ee23a39f

```
73              IERC20(state.secondaryAsset).safeTransferFrom(
74                  msg.sender,
75                  address(this),
76                  secondaryAmount
77              );
```

- Transferring tokens by `secondaryAmount`.

```
78              state.secondaryCredit[to] += secondaryAmount;
```

- The `secondaryAmount` appears to be used for bookkeeping purposes without compensating the potential transfer fees.

Recommendation

We advise the client to regulate the set of tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

Alleviation

[JOJO Team]: The secondaryAsset is USDJ, it is ERC20 standard token. So it won't appear the potential transfer fees.

OEV-01 | UNUSED CONTRACT

Category	Severity	Location	Status
Coding Style	Informational	contracts/adaptor/constOracle.sol	Acknowledged

Description

The contract `constOracle` is declared but never used.

Recommendation

We advise the client to remove or comment out the contract.

Alleviation

[JOJO Team]: JOJO will replaces the oracle to make the mark price a fixed value. From this moment onward, the perpetual price will no longer be anchored to the spot price.

OPTIMIZATIONS | JOJO-REAUDIT

ID	Title	Category	Severity	Status
SEV-01	Comparison To Boolean Constant	Coding Style	Optimization	Resolved

SEV-01 | COMPARISON TO BOOLEAN CONSTANT

Category	Severity	Location	Status
Coding Style	Optimization	contracts/subaccount/Subaccount.sol: 48	Resolved

Description

Boolean constants can be used directly and do not need to be compared to true or false.

```
48      if (success == false) {
```

Recommendation

We recommend removing the equality to the boolean constant.

Alleviation

The team heeded the advice and resolved this issue in commit `fcbc2be04faafd86986c39fbf33c32bd0069038a`.

APPENDIX | JOJO-REAUDIT

Finding Categories

Categories	Description
Centralization / Privilege	Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.
Logical Issue	Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.
Coding Style	Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.
Inconsistency	Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE

FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.