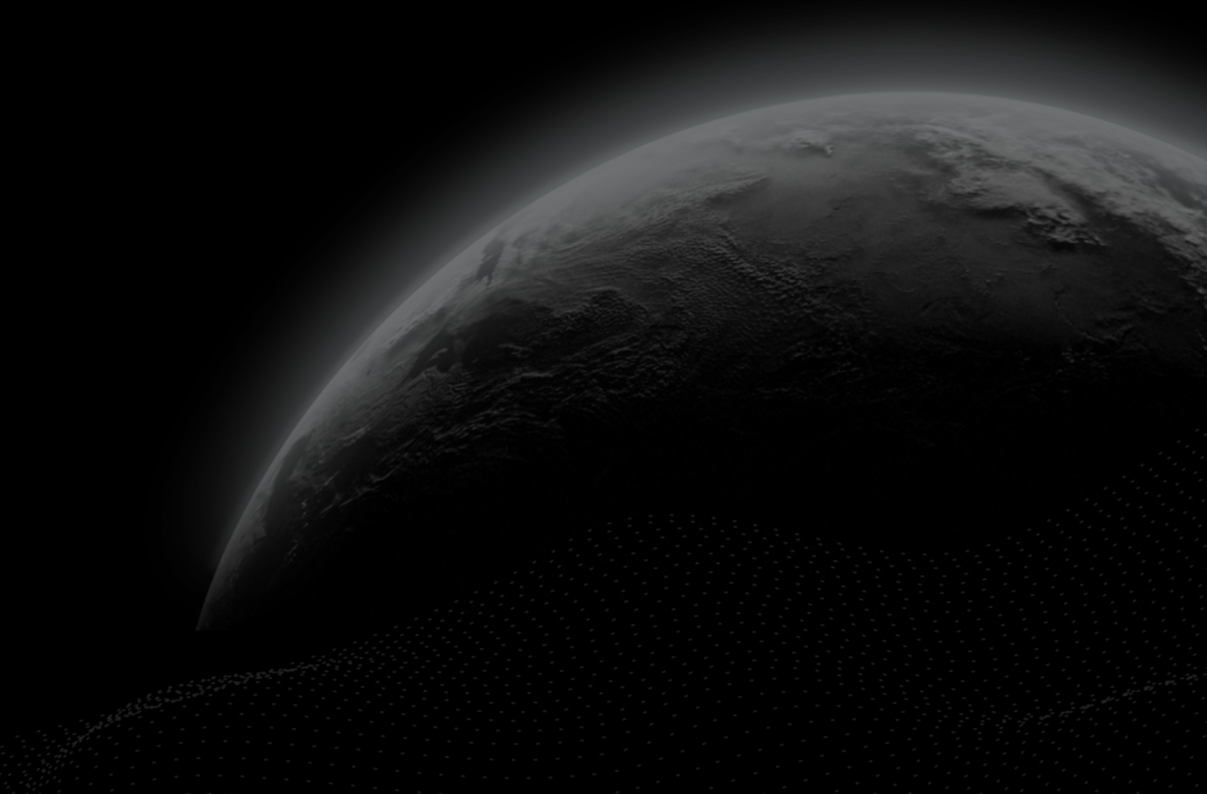
CERTIK

Security Assessment

JOJO - Ⅲ

CertiK Verified on Oct 11th, 2022



CertiK Verified on Oct 11th, 2022

JOJO - Ⅲ

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
Trading	BSC	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 10/11/2022	N/A

CODEBASE

update2 18e4f2a1e6790bdd8d9a799848f811bdf2860f65

update1 597798a3b12bbb6831a309f2121616885a3e32ef

base 23403f169a903c8c238ff34803807ac178c660cc

...View All

Vulnerability Summary

13	7	0	2	4	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.
🟨	4	Minor	2 Resolved, 1 Partially Resolved, 1 Acknowledged	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.
🟦	6	Informational	5 Resolved, 1 Partially Resolved	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 - III

Summary

Findings

Optimizations

Formal Verification

Appendix

Disclaimer

CODEBASE | JOJO - III

Repository

update2 18e4f2a1e6790bdd8d9a799848f811bdf2860f65

update1 597798a3b12bbb6831a309f2121616885a3e32ef

base 23403f169a903c8c238ff34803807ac178c660cc

AUDIT SCOPE | JOJO - III

73 files audited ● 7 files with Acknowledged findings ● 4 files with Partially Resolved findings ● 5 files with Resolved findings ● 57 files without findings

ID	File	SHA256 Checksum
● AEV	contracts/adaptor/chainlinkAdaptor.sol	c455e670bd969fea2b7195dcd08e039641f4d479561f7f817729e72b1a7fdee9
● AEM	contracts/adaptor/witnetAdaptor.sol	40a167663a2f1369504633a133a8e0ba073948a39c2297a5d01620e9556ed879
● JOE	contracts/impl/JOJOOperation.sol	4a35e4eb61ca6ae60f8224294a54447612c878edb5ceef87e97f5a5acbf48290
● OEJ	contracts/lib/Operation.sol	3316f4fde8ef3950a645556e0b1f8c40d5682430aef0c38c2a7b611487c75f71
● TEM	contracts/lib/Types.sol	9ad1fac79e661cee6e5a59ef95cb851df2771663f9daea45fb6c450b9895cd07
● SEV	contracts/subaccount/Subaccount.sol	ddb155c991177fdab39818a5e1d981c6242a2d1228802f65aff70c7157f25b9e
● SFE	contracts/subaccount/SubaccountFactory.sol	5965c08052bffb4bedd29396d202385e8321045086964c0963e9b5b8b99d950b
● PEV	contracts/impl/Perpetual.sol	318ee12543a8e2136b120c469f4c78920073cacc7b6365fc89bcff5655266b2f
● IDE	contracts/intf/IDealer.sol	4bd6110a2eb190a5e13085aafc9a6dbaa652bf7dc4ed1a91fd3ab9d67f56c95a
● LEV	contracts/lib/Liquidation.sol	e4919ea2a2fc6a0d71e7a5bf709bb1cdfc918a2ea259096433a1acf704958b22
● TEV	contracts/lib/Trading.sol	6deeca414375e5e0c9125c9ce1bddca0b6681f5ff26a62b1117fe1f9675cd688
● OEM	contracts/adaptor/emergencyOracle.sol	2f7faf71da1d11b7f2d6975a1571caa045e60197dab8d6dcfd2480511c69c7d1
● JOJ	contracts/impl/JOJODealer.sol	1d379426ac18cd3aa05e1a9731df0c98ca7082b01993535999e7601d4bfbab4d
● JOO	contracts/impl/JOJOExternal.sol	aa0fb0dbcb85745e4f7ab6279729df64f9f673ed4afe373a7b8b31b11d2405f3

ID	File	SHA256 Checksum
● FEV	contracts/lib/Funding.sol	5253143ba3549f3eb29ab0d065d590fc8fbe460310534c7fc1ce8b7ab62dcfe6
● EEV	contracts/utils/Errors.sol	93b64c5acbdf86c7d8b5bd40658b5fb593a03f6c926145360ea46e36d87f8b63
○ OEV	contracts/adaptor/constOracle.sol	73bf7eccf9f29d63f4cae57e73f02a68bfaf8f243a96b00dea705f169a0f415e
○ JOS	contracts/impl/JOJOStorage.sol	0ad2b06f76d177c1e6de75c07202ad080e39fe7f86f01f6a5cd36cc32fae36d6
○ JOV	contracts/impl/JOJOView.sol	1432b0fbf55a208a5545f653a8693fa2259d8bf9fb6ccb37ffc5c0c4a58d3ebc
○ IMP	contracts/intf/IMarkPriceSource.sol	502ce5041c08cc9b5bb0b4657c8eae76e0ff88ca60c7e630eeaebf15705a11aa
○ IPE	contracts/intf/IPerpetual.sol	fd13a44a4db1197950533b170ee194feaceef91bf7c5228fc024a09995b4a760
○ EIP	contracts/lib/EIP712.sol	e48ccaa07de9d498cdbc1dc901366bc11ea8c1fc7c226babfe46dba125b7e4a2
○ PEM	contracts/lib/Position.sol	81595028e4ee193d7e3fa6165bd7a1cbbc68a14eaab8826523737094570a1d9a
○ SDM	contracts/utils/SignedDecimalMath.sol	0f8607bc88f34e226fe80d3fea473124898330df47d17be1455cc5c77476c12b
○ AEJ	contracts/adaptor/chainlinkAdaptor.sol	c455e670bd969fea2b7195dcd08e039641f4d479561f7f817729e72b1a7fdee9
○ OEO	contracts/adaptor/constOracle.sol	73bf7eccf9f29d63f4cae57e73f02a68bfaf8f243a96b00dea705f169a0f415e
○ OVM	contracts/adaptor/emergencyOracle.sol	1df52adc0cb47594f8facd57a4abb6abde0b230d405784b8909a78366d535675
○ JOD	contracts/impl/JOJODealer.sol	c517183bc628a85583cdf6bf1b1b190e2b706cd4125f860e6dd303bc34e6883d
○ JOM	contracts/impl/JOJOExternal.sol	2d84acca1430e12a9017f536b1aa579c332ec4f156c3b46a3790ff837dbc629d
○ JJO	contracts/impl/JOJOOperation.sol	dabfc5444aeb28cfc7ba5d21dc4a20ed58772761c69abc7afbe750fca3a2f4cc
○ JJS	contracts/impl/JOJOStorage.sol	4c55de87b588e2794cc926b91d9baf2242290a71df2c6b4552521788cddbfe2a

ID	File	SHA256 Checksum
● JJV	contracts/impl/JOJOView.sol	cef9a986215febdc72bc2b968f35abcbdd4c0db8df60f51c48fdaac4c0e79147
● PEJ	contracts/impl/Perpetual.sol	c0407f6d4006d0d618055e5d536439022ec805fb2afa46b31fb859d1f6e7cc76
● IDV	contracts/intf/IDealer.sol	2aac181d88a4a348ac78f85ff3589fa2c5afa9a8005ada6b9e5d9b0165e51e80
● IDR	contracts/intf/IDecimalERC20.sol	0d3ce2265048d422279b1f80115d3823707e2ead7cd77e8a52f2e444229a6cd4
● IMS	contracts/intf/IMarkPriceSource.sol	502ce5041c08cc9b5bb0b4657c8eae76e0ff88ca60c7e630eeaebf15705a11aa
● IPV	contracts/intf/IPerpetual.sol	fd13a44a4db1197950533b170ee194feaceef91bf7c5228fc024a09995b4a760
● EIE	contracts/lib/EIP712.sol	e48ccaa07de9d498cdbc1dc901366bc11ea8c1fc7c226babfe46dba125b7e4a2
● FEM	contracts/lib/Funding.sol	5253143ba3549f3eb29ab0d065d590fc8fbe460310534c7fc1ce8b7ab62dcfe6
● LEM	contracts/lib/Liquidation.sol	c5cfedf6b9ba7cf1c13c2f9d12829763b7e265261aa265e83d048d6de628efae
● OVJ	contracts/lib/Operation.sol	03dcc009ac491931d1f06e4bbd0197f1cb2b05502b094041aced0efbcb2e9db7
● PEO	contracts/lib/Position.sol	81595028e4ee193d7e3fa6165bd7a1cbbc68a14eaab8826523737094570a1d9a
● TEJ	contracts/lib/Trading.sol	88852874123244cb0447588f1d235561374161994b009fca9f2c602b0df76139
● TEO	contracts/lib/Types.sol	42d47fd5736e9d28c44aac42148794c95111226cc1727b8c16fae2e70985da63
● SEM	contracts/subaccount/Subaccount.sol	ba822c6788d8247046295afc48ff1738794179fac49ae2c8292fa255c5cb1cd7
● SFV	contracts/subaccount/SubaccountFactory.sol	5965c08052bffb4bedd29396d202385e8321045086964c0963e9b5b8b99d950b
● EEM	contracts/utils/Errors.sol	7f7afa849c729e61e60bbb05aca80a164cd4fb445c44827c6fc41c3dca5dd28d
● SDE	contracts/utils/SignedDecimalMath.sol	ac7f29a2b3f892ac7b700ad337098d3f1f75898a00b527dc7b4e70f6475e995c

ID	File	SHA256 Checksum
AEO	contracts/adaptor/chainlinkAdaptor.sol	ad625848eaff5b5cb19d95d95318cee51540d347e7b9f531af315052a2233157
OVO	contracts/adaptor/constOracle.sol	73bf7eccf9f29d63f4cae57e73f02a68bfaf8f243a96b00dea705f169a0f415e
OMJ	contracts/adaptor/emergencyOracle.sol	1df52adc0cb47594f8facd57a4abb6abde0b230d405784b8909a78366d535675
FRU	contracts/fundingRateKeeper/FundingRateUpdateLimiter.sol	45813e3ea32f9de446ba28ded666a11ffe5f59a97894319a2a01812013ae546d
JJD	contracts/impl/JOJODealer.sol	b82dd416bd1d41d8d98cba22e42065eafa340f8bda82962e17755836d0e29e69
JJE	contracts/impl/JOJOExternal.sol	2d84acca1430e12a9017f536b1aa579c332ec4f156c3b46a3790ff837dbc629d
JJM	contracts/impl/JOJOOperation.sol	e74c0e42dcabdc8f6721d30adec5b3763c20738d8517b8f22fd97f62b23ab223
JJJ	contracts/impl/JOJOStorage.sol	4c55de87b588e2794cc926b91d9baf2242290a71df2c6b4552521788cddbfe2a
JVE	contracts/impl/JOJOView.sol	c26bec409993d80aee2861ca7c05dfa191da73b1cfcba751bf2868d83df3cc3a
PVM	contracts/impl/Perpetual.sol	c0407f6d4006d0d618055e5d536439022ec805fb2afa46b31fb859d1f6e7cc76
IDM	contracts/intf/IDealer.sol	2f17171124d184bb9d414bd5d5d64ace6a626118ff8ba46a4066dbc639a5017d
IDC	contracts/intf/IDecimalERC20.sol	0d3ce2265048d422279b1f80115d3823707e2ead7cd77e8a52f2e444229a6cd4
IME	contracts/intf/IMarkPriceSource.sol	502ce5041c08cc9b5bb0b4657c8eae76e0ff88ca60c7e630eeaebf15705a11aa
IPM	contracts/intf/IPerpetual.sol	fd13a44a4db1197950533b170ee194feaceef91bf7c5228fc024a09995b4a760
EIV	contracts/lib/EIP712.sol	e48ccaa07de9d498cdbc1dc901366bc11ea8c1fc7c226babfe46dba125b7e4a2
FEJ	contracts/lib/Funding.sol	5253143ba3549f3eb29ab0d065d590fc8fbe460310534c7fc1ce8b7ab62dcfe6
LEJ	contracts/lib/Liquidation.sol	93475c350e4181e41885e79794ea55b162954617d8111719c5a549c397bbc734

ID	File	SHA256 Checksum
OMO	contracts/lib/Operation.sol	03dcc009ac491931d1f06e4bbd0197f1cb2b05502b094041aced0efbcb2e9db7
PVJ	contracts/lib/Position.sol	81595028e4ee193d7e3fa6165bd7a1cbbc68a14eaab8826523737094570a1d9a
TVM	contracts/lib/Trading.sol	88852874123244cb0447588f1d235561374161994b009fca9f2c602b0df76139
TVJ	contracts/lib/Types.sol	42d47fd5736e9d28c44aac42148794c95111226cc1727b8c16fae2e70985da63
SEJ	contracts/subaccount/Subaccount.sol	ba822c6788d8247046295afc48ff1738794179fac49ae2c8292fa255c5cb1cd7
SFM	contracts/subaccount/SubaccountFactory.sol	5965c08052bffb4bedd29396d202385e8321045086964c0963e9b5b8b99d950b
EEJ	contracts/utils/Errors.sol	7f7afa849c729e61e60bbb05aca80a164cd4fb445c44827c6fc41c3dca5dd28d
SDV	contracts/utils/SignedDecimalMath.sol	ac7f29a2b3f892ac7b700ad337098d3f1f75898a00b527dc7b4e70f6475e995c

APPROACH & METHODS | JOJO - Ⅲ

This report has been prepared for JOJO to discover issues and vulnerabilities in the source code of the JOJO - Ⅲ 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.

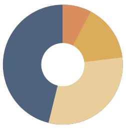
The auditing process pays special attention to the following considerations:

- 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 - Ⅲ



Total Findings	Critical	Major	Medium	Minor	Informational
13	0	1	2	4	6

This report has been prepared to discover issues and vulnerabilities for JOJO - Ⅲ. Through this audit, we have uncovered 13 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
AEM-01	Failed `ERC2362.valueFor()` Request Is Not Handled	Volatile Code	Minor	● Resolved
EVM-01	Missing Zero Address Validation	Volatile Code	Minor	● Acknowledged
JOE-01	**Centralization Risks In JOJOOperation.Sol**	**Centralization / Privilege**	**Major**	● **Acknowledged**
JOE-02	Secondary Asset `decimals` Is Not Checked	Volatile Code	Minor	● Resolved
OEJ-01	Open Positions Are Discarded If `Perpetual` Is Deregistered	Logical Issue	Medium	● Acknowledged
PEV-01	Potential Reentrancy In `_settle()`	Volatile Code	Minor	● Partially Resolved
TEM-01	`validOrderSender` Can Manipulate The Market	Control Flow	Medium	● Acknowledged
EVM-03	Typos	Coding Style	Informational	● Resolved
EVM-04	Incorrect Comments	Coding Style	Informational	● Partially Resolved
FEV-01	Incompatibility With Deflationary Tokens	Logical Issue	Informational	● Resolved

ID	Title	Category	Severity	Status
JOO-01	`view` Functions Can Be Declared In `JOJOView`	Inconsistency	Informational	● Resolved
LEV-02	`liquidationThreshold` Scaling Factor Can Be Declared As A Constant	Magic Numbers	Informational	● Resolved
OEM-01	Uninitialized State Variable `roundId`	Coding Style	Informational	● Resolved

AEM-01 | FAILED `ERC2362.valueFor()` REQUEST IS NOT HANDLED

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/adaptor/witnetAdaptor.sol (base): 33~34	● Resolved

Description

`EIP-2362` standard defines the status codes returned by `valueFor()`. `valueFor()` will return a status code of 404 if the value for an id is not available yet, 400 in case of bad request.

`getMarkPrice()` doesn't handle the status codes.

Recommendation

We recommend reverting if `statusCode != 200` to ensure the correct behavior.

EVM-01 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/adaptor/chainlinkAdaptor.sol (base): 32; contracts/adaptor/witnetAdaptor.sol (base): 28; contracts/subaccount/Subaccount.sol (base): 41, 42; contracts/subaccount/SubaccountFactory.sol (base): 33	● Acknowledged

Description

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

```
32          chainlink = _chainlink;
```

```
28          witnet = _witnet;
```

```
41          owner = _owner;
```

```
42          dealer = _dealer;
```

```
33          dealer = _dealer;
```

Recommendation

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

Alleviation

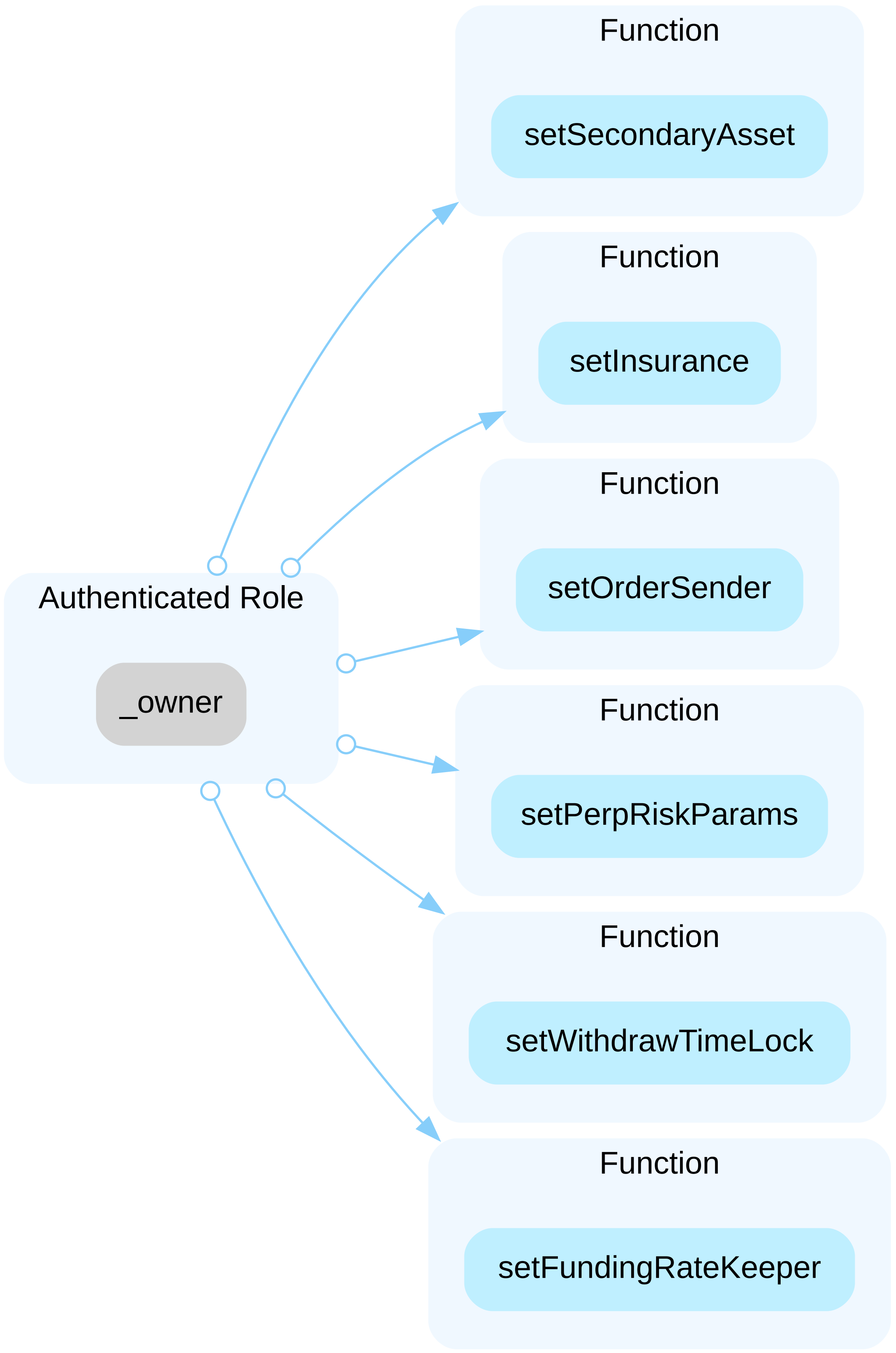
[JOJO]: Issue acknowledged. We won't make any changes for the current version.

JOE-01 | CENTRALIZATION RISKS IN JOJOOPERATION.SOL

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**contracts/impl/JOJOOperation.sol (base): 34, 41, 45, 49, 56, 65**	● **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`) .



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., multisignature 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]: We will use a 2 of 3 multisig wallet as the owner. But the multisig wallet won't have timelock for the purpose of fast reaction.

JOE-02 | SECONDARY ASSET `decimals` IS NOT CHECKED

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/impl/JOJOOperation.sol (base): 64~65	● Resolved

Description

`setSecondaryAsset()` allows to set any address as `secondaryAsset`. It can't later be reassigned. `decimals` is not checked, however, expected to be the same as `primaryAsset`. Wrong `secondaryAsset` will break the contract.

Recommendation

We recommend checking that `IERC20(_secondaryAsset).decimals() == IERC20(state.primaryAsset).decimals()` and calling `setSecondaryAsset()` with care.

OEJ-01 | OPEN POSITIONS ARE DISCARDED IF `Perpetual` IS DEREGISTERED

Category	Severity	Location	Status
Logical Issue	● Medium	contracts/lib/Operation.sol (base): 45	● Acknowledged

Description

Function `setPerpRiskParams()` allows to deregister the Perpetual from the Dealer even if there are open positions. Users with positive `credit` will suffer loss, users with negative will gain profit. It would be more fair to forcefully close all the positions before deregistering.

Recommendation

We recommend forbidding deregistering of Perpetual with open positions or implementing the auto-closing logic.

Alleviation

[JOJO]: We will remove the perp only when no position open.

PEV-01 | POTENTIAL REENTRANCY IN `_settle()`

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/impl/Perpetual.sol (base): 201~206	● Partially Resolved

Description

`Perpetual._settle()` first makes an external call to `IDealer.realizePnl()` and only after that nullifies `reducedCredit`. This opens a risk of reentrancy if `Position._realizePnl()` will have external calls in the future.

Recommendation

We recommend:

1. Update the `reducedCredit` before making the external call (use the Checks-Effects-Interactions Pattern)
2. In `_realizePnl()` check that `state.hasPosition[trader][msg.sender] == true`
3. Ensure that `state.openPositions[trader]` has the position in Perpetual `msg.sender`. Revert otherwise.

Alleviation

[CertiK]: `reducedCredit` still updated after `realizePnl()` call. Other protection measures were not implemented.

TEM-01 | `validOrderSender` CAN MANIPULATE THE MARKET

Category	Severity	Location	Status
Control Flow	● Medium	contracts/lib/Types.sol (base): 78~79	● Acknowledged

Description

`validOrderSender` can "replay" a cancelled off-chain order or fulfill makers' orders in wrong order.

`validOrderSender` prepares a set of orders to execute the `Perpetual.trade()` . Each order should be signed by `order.signer` or by her operator. However, the cancelled order can still be used until it expires. `nonce` part of the order is only used to distinguish orders, it is never checked on-chain and not "consumed" after cancellation.

`validOrderSender` can use the taker order and fulfill her own maker order with the worst possible price, ignoring all the other maker orders with better prices.

Recommendation

We recommend limiting the "expiration" field of the order with relatively small time. Off-chain frontend can recreate orders automatically after expiration. We recommend significantly limit the `validOrderSender` set.

Alleviation

[JOJO]: We will treat the order sender very carefully. The reason we don't allow the users to cancel orders onchain:

- From our observation, very few users want to pay gas for this onchain cancellation.
- The onchain cancellation will slow down our matching engine.

EVM-03 | TYPOS

Category	Severity	Location	Status
Coding Style	● Informational	contracts/impl/JOJODealer.sol (base): 14; contracts/impl/Perpetual.sol (base): 193; contracts/lib/Trading.sol (base): 42, 208; contracts/lib/Types.sol (base): 59, 93; contracts/subaccount/Subaccount.sol (base): 15; contracts/utils/Errors.sol (base): 16, 32	● Resolved

Description

"shoule" is supposed to be "should".

"happens" is supposed to be "happen".

"ALREASY" is supposed to be "ALREADY".

"LEASE" is supposed to be "LEAST".

"Operatiors" is supposed to be "Operators".

"implemnents" is supposed to be "implementation".

"newReducedCredkt" is supposed to be "newReducedCredit".

"mathcing" is supposed to be "matching".

"whold" is supposed to be "whole".

And some others.

Recommendation

We recommend fixing the typos.

EVM-04 | INCORRECT COMMENTS

Category	Severity	Location	Status
Coding Style	● Informational	contracts/intf/IDealer.sol (base): 10~12; contracts/lib/Liquidation.sol (base): 196~197, 233~234, 272~273; contracts/lib/Trading.sol (base): 44~45	● Partially Resolved

Description

```
10      /// @param primaryAmount is the amount of primary asset you want to
withdraw.
11      /// @param secondaryAmount is the amount of secondary asset you want to
withdraw.
```

The description of `deposit()` is about `withdraw`.

```
44      /// orderList[0] is taker order and orderList[1:] are taker orders.
```

All but first are maker orders.

```
196      /// safe or being liquidated if return 0.
```

```
233              If liqPrice<0, it should be considered as absolutely safe or being
liquidated.
```

"or" is likely supposed to be "of".

Recommendation

We recommend fixing the comments to reflect the code.

Alleviation

Liquidation.sol@272, Trading.sol@44 were not updated.

FEV-01 | INCOMPATIBILITY WITH DEFLATIONARY TOKENS

Category	Severity	Location	Status
Logical Issue	● Informational	contracts/lib/Funding.sol (base): 72~76, 77	● Resolved

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

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

- Transferring tokens by `secondaryAmount`.

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

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

Recommendation

We recommend carefully choosing of supported tokens only as `primaryAsset` and `secondaryAsset`.

Alleviation

[JOJO]: We will use supported tokens like USDC as primary credit. And we will launch a standard token as secondary asset.

JOO-01 | `view` FUNCTIONS CAN BE DECLARED IN `JOJOView`

Category	Severity	Location	Status
Inconsistency	● Informational	contracts/impl/JOJOExternal.sol (base): 52~69	● Resolved

Description

`JOJOExternal` implements several `view` functions: `isSafe()` , `isAllSafe()` , `getFundingRate()` . They can be moved to `JOJOView` for consistency.

Recommendation

We recommend moving `view` functions from `JOJOExternal` to `JOJOView` .

LEV-02 | `liquidationThreshold` SCALING FACTOR CAN BE DECLARED AS A CONSTANT

Category	Severity	Location	Status
Magic Numbers	● Informational	contracts/lib/Liquidation.sol (base): 262~264	● Resolved

Description

`liquidationThreshold` is represented as a fixed-point number with the scaling factor `10**18`. It can be declared as a constant to improve the code readability. Same factor is used by `liquidationPriceOff`, `insuranceFeeRate`.

Recommendation

We recommend declaring a constant `FIXED_POINT_FACTOR = 10**18`.

OEM-01 | UNINITIALIZED STATE VARIABLE `roundId`

Category	Severity	Location	Status
Coding Style	● Informational	contracts/adaptor/emergencyOracle.sol (base): 15, 34	● Resolved

Description

One or more state variables are used without being initialized in the constructor.

```
15      uint256 public roundId;
```

- `roundId` is never initialized, but used in `EmergencyOracle.setMarkPrice` .

Recommendation

We recommend removing of `roundId` state field.

OPTIMIZATIONS | JOJO - III

ID	Title	Category	Severity	Status
AEM-02	Variables That Could Be Declared As Immutable	Gas Optimization	Optimization	● Resolved
EVM-02	`external` Functions Can Accept `calldata` Arguments	Gas Optimization	Optimization	● Resolved
LEV-01	`params` Can Use `storage` Specifier	Gas Optimization	Optimization	● Resolved
PEV-02	Perpetual Asks The Dealer For `fundingRate`	Gas Optimization	Optimization	● Resolved

AEM-02 | VARIABLES THAT COULD BE DECLARED AS IMMUTABLE

Category	Severity	Location	Status
Gas Optimization	● Optimization	contracts/adaptor/witnetAdaptor.sol (base): 24	● Resolved

Description

The linked variables assigned in the constructor can be declared as `immutable`. Immutable state variables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. A big advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variables since they will not be stored in storage.

Recommendation

We recommend declaring these variables as `immutable`.

EVM-02 `external` FUNCTIONS CAN ACCEPT `calldata` ARGUMENTS

Category	Severity	Location	Status
Gas Optimization	● Optimization	contracts/impl/JOJOExternal.sol (base): 57~58; contracts/intf/IDealer.sol (base): 55~56	● Resolved

Description

`external` functions can accept `calldata` arguments instead of `memory`, if the arguments are not modified. It allows to avoid copying and save gas.

Recommendation

We recommend accepting arguments as `calldata` wherever possible and pass them as `calldata` to `internal` functions.

LEV-01 | `params` CAN USE `storage` SPECIFIER

Category	Severity	Location	Status
Gas Optimization	● Optimization	contracts/lib/Liquidation.sol (base): 149~150, 245~246	● Resolved

Description

In `_isAllSafe()` the local variable `params` is declared as `memory`. However, only two fields are accessed. The variable can be declared as `storage`.

Recommendation

We recommend declaring the variable as `storage` to prevent copying and save gas.

PEV-02 | PERPETUAL ASKS THE DEALER FOR `fundingRate`

Category	Severity	Location	Status
Gas Optimization	● Optimization	contracts/impl/Perpetual.sol (base): 78	● Resolved

Description

In `Perpetual.balanceOf()` current `fundingRate` is retrieved via `IDealer(owner()).getFundingRate(address(this))` instead of direct state field access. The trades and liquidations are settled using `fundingRate` directly.

Recommendation

We recommend using of `fundingRate` directly.

FORMAL VERIFICATION | JOJO - III

Formal guarantees about the behavior of smart contracts can be obtained by reasoning about properties relating to the entire contract (e.g. contract invariants) or to specific functions of the contract. Once such properties are proven to be valid, they guarantee that the contract behaves as specified by the property. As part of this audit, we applied automated formal verification (symbolic model checking) to prove that well-known functions in the smart contracts adhere to their expected behavior.

Considered Functions And Scope

Verification of ERC-20 compliance

We verified properties of the public interface of those token contracts that implement the ERC-20 interface. This covers

- Functions `transfer` and `transferFrom` that are widely used for token transfers,
- functions `approve` and `allowance` that enable the owner of an account to delegate a certain subset of her tokens to another account (i.e. to grant an allowance), and
- the functions `balanceOf` and `totalSupply` , which are verified to correctly reflect the internal state of the contract.

The properties that were considered within the scope of this audit are as follows:

Property Name	Title
erc20-transfer-succeed-self	Function `transfer` Succeeds on Admissible Self Transfers
erc20-transfer-revert-zero	Function `transfer` Prevents Transfers to the Zero Address
erc20-transfer-succeed-normal	Function `transfer` Succeeds on Admissible Non-self Transfers
erc20-transfer-correct-amount	Function `transfer` Transfers the Correct Amount in Non-self Transfers
erc20-transfer-correct-amount-self	Function `transfer` Transfers the Correct Amount in Self Transfers
erc20-transfer-change-state	Function `transfer` Has No Unexpected State Changes
erc20-transfer-exceed-balance	Function `transfer` Fails if Requested Amount Exceeds Available Balance
erc20-transfer-recipient-overflow	Function `transfer` Prevents Overflows in the Recipient's Balance
erc20-transfer-false	If Function `transfer` Returns `false` , the Contract State Has Not Been Changed
erc20-transfer-never-return-false	Function `transfer` Never Returns `false`
erc20-transferfrom-revert-from-zero	Function `transferFrom` Fails for Transfers From the Zero Address

Property Name	Title

erc20-transferfrom-correct-amount	Function `transferFrom` Transfers the Correct Amount in Non-self Transfers
erc20-transferfrom-succeed-normal	Function `transferFrom` Succeeds on Admissible Non-self Transfers
erc20-transferfrom-succeed-self	Function `transferFrom` Succeeds on Admissible Self Transfers
erc20-transferfrom-correct-amount-self	Function `transferFrom` Performs Self Transfers Correctly
erc20-transferfrom-fail-exceed-balance	Function `transferFrom` Fails if the Requested Amount Exceeds the Available Balance
erc20-transferfrom-correct-allowance	Function `transferFrom` Updated the Allowance Correctly
erc20-transferfrom-change-state	Function `transferFrom` Has No Unexpected State Changes
erc20-transferfrom-fail-exceed-allowance	Function `transferFrom` Fails if the Requested Amount Exceeds the Available Allowance
erc20-transferfrom-false	If Function `transferFrom` Returns `false` , the Contract's State Has Not Been Changed
erc20-totalsupply-succeed-always	Function `totalSupply` Always Succeeds
erc20-transferfrom-fail-recipient-overflow	Function `transferFrom` Prevents Overflows in the Recipient's Balance
erc20-transferfrom-never-return-false	Function `transferFrom` Never Returns `false`
erc20-totalsupply-correct-value	Function `totalSupply` Returns the Value of the Corresponding State Variable
erc20-totalsupply-change-state	Function `totalSupply` Does Not Change the Contract's State
erc20-balanceof-succeed-always	Function `balanceOf` Always Succeeds
erc20-balanceof-correct-value	Function `balanceOf` Returns the Correct Value
erc20-balanceof-change-state	Function `balanceOf` Does Not Change the Contract's State
erc20-allowance-succeed-always	Function `allowance` Always Succeeds
erc20-allowance-correct-value	Function `allowance` Returns Correct Value
erc20-allowance-change-state	Function `allowance` Does Not Change the Contract's State
erc20-approve-succeed-normal	Function `approve` Succeeds for Admissible Inputs
erc20-approve-revert-zero	Function `approve` Prevents Giving Approvals For the Zero Address
erc20-approve-correct-amount	Function `approve` Updates the Approval Mapping Correctly
erc20-approve-change-state	Function `approve` Has No Unexpected State Changes
erc20-approve-false	If Function `approve` Returns `false` , the Contract's State Has Not Been Changed
erc20-approve-never-return-false	Function `approve` Never Returns `false`

Verification Results

For the following contracts, model checking established that each of the 38 properties that were in scope of this audit (see scope) are valid:

Contract TestERC20 (Source File contracts/testSupport/TestERC20.sol)

Detailed results for function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-succeed-self	● True	
erc20-transfer-revert-zero	● True	
erc20-transfer-succeed-normal	● True	
erc20-transfer-correct-amount	● True	
erc20-transfer-correct-amount-self	● True	
erc20-transfer-change-state	● True	
erc20-transfer-exceed-balance	● True	
erc20-transfer-recipient-overflow	● True	
erc20-transfer-false	● True	
erc20-transfer-never-return-false	● True	

Detailed results for function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-revert-from-zero	● True	
erc20-transferfrom-revert-to-zero	● True	
erc20-transferfrom-correct-amount	● True	
erc20-transferfrom-succeed-normal	● True	
erc20-transferfrom-succeed-self	● True	
erc20-transferfrom-correct-amount-self	● True	
erc20-transferfrom-fail-exceed-balance	● True	
erc20-transferfrom-correct-allowance	● True	
erc20-transferfrom-change-state	● True	
erc20-transferfrom-fail-exceed-allowance	● True	
erc20-transferfrom-false	● True	
erc20-transferfrom-fail-recipient-overflow	● True	
erc20-transferfrom-never-return-false	● True	

Detailed results for function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-succeed-always	● True	
erc20-totalsupply-correct-value	● True	
erc20-totalsupply-change-state	● True	

Detailed results for function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-succeed-always	● True	
erc20-balanceof-correct-value	● True	
erc20-balanceof-change-state	● True	

Detailed results for function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-succeed-always	● True	
erc20-allowance-correct-value	● True	
erc20-allowance-change-state	● True	

Detailed results for function `approve`

Property Name	Final Result	Remarks
erc20-approve-succeed-normal	● True	
erc20-approve-revert-zero	● True	
erc20-approve-correct-amount	● True	
erc20-approve-change-state	● True	
erc20-approve-false	● True	
erc20-approve-never-return-false	● True	

Contract ERC20 (Source File node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol)

Detailed results for function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-revert-zero	● True	
erc20-transfer-succeed-normal	● True	
erc20-transfer-succeed-self	● True	
erc20-transfer-correct-amount	● True	
erc20-transfer-correct-amount-self	● True	
erc20-transfer-change-state	● True	
erc20-transfer-exceed-balance	● True	
erc20-transfer-recipient-overflow	● True	
erc20-transfer-false	● True	
erc20-transfer-never-return-false	● True	

Detailed results for function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-revert-from-zero	● True	
erc20-transferfrom-revert-to-zero	● True	
erc20-transferfrom-succeed-self	● True	
erc20-transferfrom-succeed-normal	● True	
erc20-transferfrom-correct-amount	● True	
erc20-transferfrom-correct-amount-self	● True	
erc20-transferfrom-change-state	● True	
erc20-transferfrom-correct-allowance	● True	
erc20-transferfrom-fail-exceed-balance	● True	
erc20-transferfrom-fail-exceed-allowance	● True	
erc20-transferfrom-false	● True	
erc20-transferfrom-fail-recipient-overflow	● True	
erc20-transferfrom-never-return-false	● True	

Detailed results for function `totalSupply`

Property Name	Final Result	Remarks
erc20-totalsupply-succeed-always	● True	
erc20-totalsupply-correct-value	● True	
erc20-totalsupply-change-state	● True	

Detailed results for function `balanceOf`

Property Name	Final Result	Remarks
erc20-balanceof-succeed-always	● True	
erc20-balanceof-correct-value	● True	
erc20-balanceof-change-state	● True	

Detailed results for function `allowance`

Property Name	Final Result	Remarks
erc20-allowance-succeed-always	● True	
erc20-allowance-correct-value	● True	
erc20-allowance-change-state	● True	

Detailed results for function `approve`

Property Name	Final Result	Remarks
erc20-approve-revert-zero	● True	
erc20-approve-succeed-normal	● True	
erc20-approve-correct-amount	● True	
erc20-approve-change-state	● True	
erc20-approve-false	● True	
erc20-approve-never-return-false	● True	

APPENDIX | JOJO - III

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.
Gas Optimization	Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.
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.
Control Flow	Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.
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.
Magic Numbers	Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

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.