CERTIK

Code Security Assessment

ClearDAO

Jan 18th, 2022

Table of Contents

Summary

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

- 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.

Overview

Project Summary

Project Name	ClearDAO
Platform	ethereum
Language	Solidity
Codebase	https://github.com/DerivStudio/contracts/tree/main/contracts
Commit	7ca94af91446646cb1315a3c5d47c204f30e7e1a

Audit Summary

Delivery Date	Jan 18, 2022
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

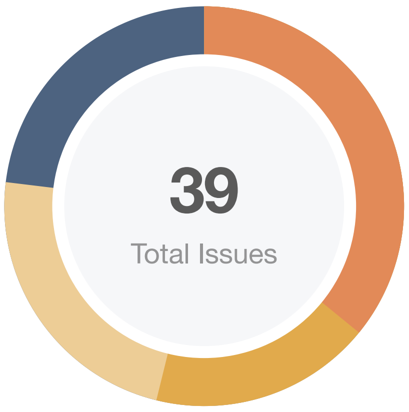
Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● Major	14	0	0	14	0	0
● Medium	7	0	0	2	0	5
● Minor	9	0	0	5	0	4
● Informational	9	0	0	1	0	8
● Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
ERC	interfaces/ERC20detail.sol	67f0f720e7e30f51c472a2b8d4773e9abb4a309a8b14ca7f1c11204111989f4a
ICR	interfaces/ICRCN.sol	b1f5ba3fa53b8a74211c1b6c24e7041c214b609d60e8924b8088c9f34e766de1
IIC	interfaces/IInvest.sol	ccc57e8490dbbdfd0f86e330d439db77a1adfc7fe65b0adc564cd7d133c845d6
IOO	interfaces/IOpenOraclePriceData.sol	256dec93c17bd5365d7a0553ff36f4a3a2456b788bb616264022311317950986
IOI	interfaces/IOptionIndex.sol	a655929e178056154555ade612945210059ba2439672e7d0f6f9f94aa38771c2
IOM	interfaces/IOptionMain.sol	d76f3251c09d67b16e64f881cb92c0c6d415c8beb46180363ff8c1edb4af9542
IOC	interfaces/IOptionMarket.sol	1b06ed8d9755630bcbf3cc6ca000543eef292e637760c48cd8573d34ac4ee8cc
IOV	interfaces/IOptionV2.sol	fda89f50aaf1c5091dd36dce5a2247e64d60c3ee140e90aea1748d9fe36e1331
IOP	interfaces/IOptionV2Maker.sol	5c64b03379064f4b9639b4a131592be8ce0a2a623d4ffeec5a9cf98bc70ae7bf
IOK	interfaces/IOracle.sol	28dab63d9f9f78f25c825bb33d6907aaf31703638494997a85e5b30a7380bff8
IPC	interfaces/IPriceCoefficient.sol	76b58432e9a56bdef73a7b467cd216c20c9b1410dd703aebf88e403bc0eda6e1
ISR	interfaces/IStdReference.sol	e0eb9077eded984cd5ee4de89949e711f8fbf2e62aaf07c89ad38a83310a4787
OVH	library/OptionV2Helper.sol	fffaf60b623e62acf870e130022d7ea534abb76c2891b926e696a39cfd3e3d60
SVC	library/StringsV2.sol	bf8e5b1e146c0d15242aff63f3f835443a62aa3d41b6587fd88400e2044a7ce8
CAC	ClearAccessControl.sol	f3e4eae50615527b8539be6b4b4b06fb8e3b247b2bd0555b59021f58566eca11
CCP	ClearProxy.sol	8461292d437318fa2b041e462dd18ddbc22daf5c86c25822acda637cf21c75ea
CPA	ClearProxyAdmin.sol	72aa83d80421f9f6353fe540c99a1a56cfe48825133190e9e8ad132553fa0c90
MSW	MultiSigWallet.sol	737ec7feb02e8734204e9926b8767f0640676cf925023d692ea5aaeaf052470a
NVC	NoteV2.sol	c5c1910c5106d103a1010880069eff8e02aaf0a0cacd75f86f25b2fcfc6c7f28
NVI	NoteV2Invest.sol	f402967d59dc712378c3751398dcd19e33e54b53b846b785c7fbbc01e8546149
OIC	OptionIndex.sol	69eb2a3c7805dbbf02173a667d72a66d48b237bbd4bf7121e71e0e273dcdcee7
OTC	OptionToken.sol	3fca664a2841fb1363a3093771957a39e35c7d8a74718d42dbd222518743b1da

ID	File	SHA256 Checksum
OVC	OptionV2.sol	884ffecc468bd3a31c796a7360711d2f45746ba7e8af184effc0b0346011762e
OVM	OptionV2Maker.sol	7a2eec8cc778d8bb85ebed082a9369036e30d331c5a5943002173c059c4e4463
OCP	Oracle.sol	c0ad423c87b859978e78266834da89978e2c01bcdff5eb2acf207d21229df2c2
OVP	OwnableV2.sol	54e072f1653f5863a90f5c78edb8b88853463623d777ed20177ac5e20f3f4960
PCP	Pausable.sol	c992a8689cfd0331b9eefbcb80fea01c18c9f31b30f7e84e906d9b22a4d162bf
PCC	PriceCoefficient.sol	c1aa29311c4bac4a4b42ba06697e0173e86b09be6287107c5d2208a96e162358

Findings



🟥 **Critical**	**0** (0.00%)
🟧 **Major**	**14** (35.90%)
🟨 **Medium**	**7** (17.95%)
🟨 **Minor**	**9** (23.08%)
🟦 **Informational**	**9** (23.08%)
🟩 **Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Calling mechanism of function `setExercisePrice()`	Logical Issue	● Informational	✓ Resolved
CAC-01	Centralization Risk: Privileged Role, _owner, in Contract, ClearAccessControl.	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
CPC-01	Redundant Code Components	Volatile Code	● Informational	✓ Resolved
CPC-02	Missing emit events	Coding Style	● Informational	✓ Resolved
NVC-01	Check effect interaction pattern violated	Logical Issue	● Minor	✓ Resolved
NVC-02	Price is determined off-chain	Logical Issue	● Medium	ⓘ Acknowledged
NVC-03	No upper limit for `rate`	Logical Issue	● Minor	ⓘ Acknowledged
NVC-04	Centralization Risk: Privileged Role, _owner, in Contract, NoteV2.	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
NVC-05	Centralization Risk: Privileged Role, manager, in Contract, NoteV2.	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
NVC-06	Third party dependencies	Volatile Code	● Minor	ⓘ Acknowledged
NVC-07	Missing input validation of `_cycleIndex`	Logical Issue	● Informational	✓ Resolved
NVI-01	Potential wrong logic of modifier `onlyOwner()`	Logical Issue	● Minor	✓ Resolved

ID	Title	Category	Severity	Status
NVI-02	Logical issue of function `setPayToken()`	Logical Issue, Control Flow	● Minor	ⓘ Acknowledged
NVI-04	Third party dependencies	Volatile Code	● Minor	ⓘ Acknowledged
NVI-05	Centralization Related Risks	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
OCP-01	Centralization Risk: Privileged Role, _owner, in Contract, Oracle.	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
OIC-01	Centralization Risk: Privileged Role, _owner, in Contract, OptionIndex.	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
OIC-02	Logical issue about token decimals	Logical Issue	● Informational	ⓘ Acknowledged
OIC-03	Lack of access control	Logical Issue	● Medium	✓ Resolved
OIC-04	Potential flashloan attack	Control Flow	● Major	ⓘ Acknowledged
OIC-05	Third party dependencies	Volatile Code	● Minor	ⓘ Acknowledged
OTC-01	Centralization Risk: Privileged Role, _owner, in Contract, OptionToken.	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
OTC-02	Centralization Risk: Privileged Role, optionMain, in Contract, OptionToken.	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
OVC-01	Centralization Risk: Privileged Role, _owner, in Contract, OptionV2.	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
OVC-02	Logical issue of function `buyOption()`	Logical Issue	● Informational	✓ Resolved
OVC-03	Function `sellOption()` does not `addMakerBalance()`	Logical Issue	● Medium	✓ Resolved
OVC-04	Cases of selling option	Logical Issue	● Informational	✓ Resolved
OVC-05	Weak control on order's status	Logical Issue	● Medium	✓ Resolved
OVC-06	Logical issue of function `userExercise()`	Logical Issue	● Medium	✓ Resolved
OVC-07	Logical issue about `knockoutRebate`	Logical Issue	● Informational	✓ Resolved

ID	Title	Category	Severity	Status
OVC-08	Insufficient funds due to high `knockoutRebate`	Logical Issue	● Minor	Resolved
OVC-09	Logical issue of function `sellOption()`	Logical Issue	● Medium	Acknowledged
OVC-10	Logical issue of function `userKnockout()`	Logical Issue	● Informational	Resolved
OVC-11	Logical issue of function `makerKnockout()`	Logical Issue	● Medium	Resolved
OVC-12	Control flow of the option	Control Flow	● Minor	Resolved
OVM-01	Centralization Risk: Privileged Role, _owner, in Contract, OptionV2Maker.	**Centralization / Privilege**	● **Major**	Acknowledged
OVM-02	Centralization Risk: Privileged Role, optionMain, in Contract, OptionV2Maker.	**Centralization / Privilege**	● **Major**	Acknowledged
OVP-01	Centralization Risk: Privileged Role, _owner, in Contract, OwnableV2.	**Centralization / Privilege**	● **Major**	Acknowledged
PCP-01	Centralization Risk: Privileged Role, _owner, in Contract, Pausable.	**Centralization / Privilege**	● **Major**	Acknowledged

GLOBAL-01 | Calling mechanism of function `setExercisePrice()`

Category	Severity	Location	Status
Logical Issue	● Informational	Global	✓ Resolved

Description

The prices of the index should be updated every day at 0 o'clock. So the server should trigger a daily call to this function.

Alleviation

The client stated that the prices of the indexes will be updated at 0 o'clock every day.

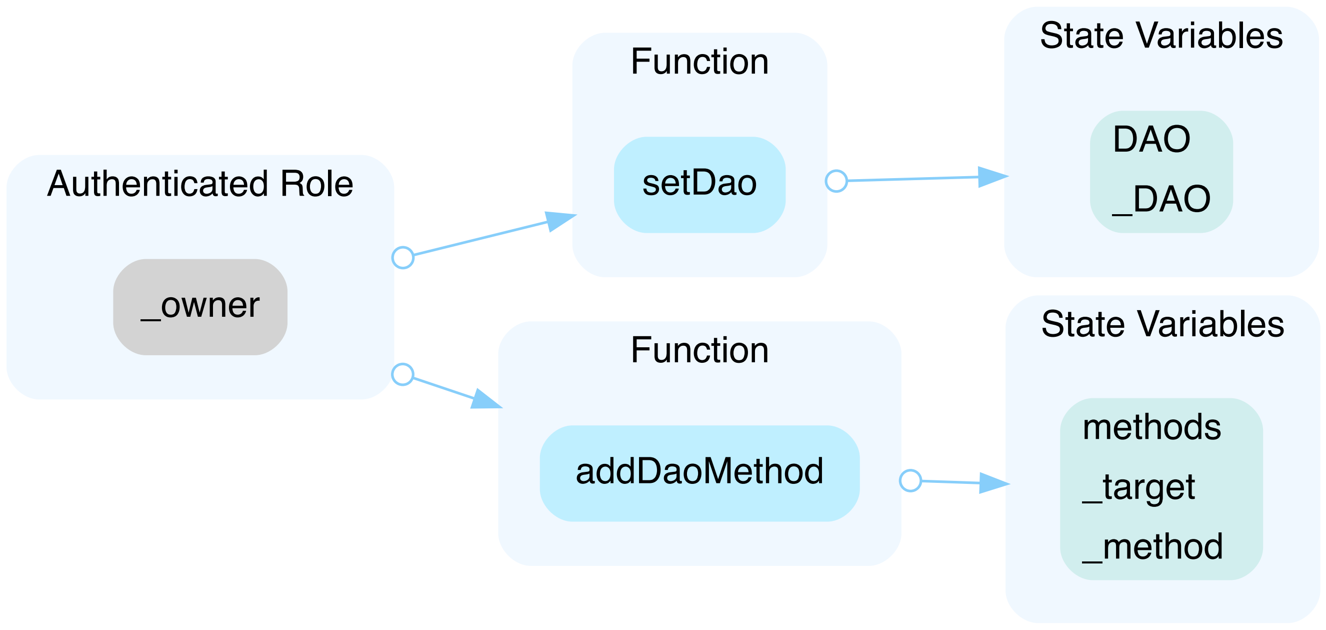
CAC-01 | Centralization Risk: Privileged Role, _owner, in Contract, ClearAccessControl.

Category	Severity	Location	Status
Centralization / Privilege	● Major	projects/Clear Protocal/contracts/ClearAccessControl.sol (826e704): 19~22, 24~28	ⓘ Acknowledged

Description

In the contract, `ClearAccessControl`, the role, `_owner`, has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `_owner` may allow the hacker to take advantage of this.



Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged the issue and stated the following.

The owner of the contract `ClearAccessContrl` is a `MultiSigWalelt` address, so its privileges will not be abused. Later on, all rights management will be handed over to DAO governance.

CPC-01 | Redundant Code Components

Category	Severity	Location	Status
Volatile Code	● Informational	projects/Clear Protocal/contracts/MultiSigWallet.sol (826e704): 28~32, 64~68 projects/Clear Protocal/contracts/OptionToken.sol (826e704): 53~60	✓ Resolved

Description

The linked statements do not affect the functionality of the codebase and appear to be either leftovers from test code or older functionality.

Recommendation

We advise to remove the redundant statements for production environments.

Alleviation

The team heeded our advice and fixed the issue in commit 902b0e2d349e9c727da19ac6d879cb697a68ad5e.

CPC-02 | Missing emit events

Category	Severity	Location	Status
Coding Style	● Informational	projects/Clear Protocal/contracts/ClearAccessControl.sol (826e704): 19~22 projects/Clear Protocal/contracts/NoteV2.sol (826e704): 330~332, 339~341, 356~358, 365~371, 373~381 projects/Clear Protocal/contracts/Oracle.sol (826e704): 44~53, 55~60, 62~70 projects/Clear Protocal/contracts/OptionIndex.sol (826e704): 119~121 projects/Clear Protocal/contracts/OptionToken.sol (826e704): 25~28, 30~32, 43~51, 62~70 projects/Clear Protocal/contracts/OptionV2.sol (826e704): 439~441, 443~445, 447~449, 451~453, 455~458, 463~468, 473~478, 480~482 projects/Clear Protocal/contracts/OptionV2Maker.sol (826e704): 76~81, 155~194, 196~215, 276~278, 280~286, 288~293	✓ Resolved

Description

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

Recommendation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Alleviation

The team heeded our advice and resolved the issue in commit a696e63029f1744db09ffc0ebcea910a3cad3312.

NVC-01 | Check effect interaction pattern violated

Category	Severity	Location	Status
Logical Issue	● Minor	NoteV2.sol: 127	✓ Resolved

Description

The order of external call/transfer and storage manipulation must follow the check-effect-interaction pattern.

Recommendation

We advise the client to check if storage manipulation is before the external call/transfer operation. LINK

Alleviation

The team heeded our advice and fixed the issue in commit 902b0e2d349e9c727da19ac6d879cb697a68ad5e.

NVC-02 | Price is determined off-chain

Category	Severity	Location	Status
Logical Issue	● Medium	projects/Clear Protocal/contracts/NoteV2.sol (826e704): 299, 242	ⓘ Acknowledged

Description

The price of buying or selling options is determined off-chain. Rules and calculations are not found in the contract. If it does not equal `startPrice`, there may be problems with the allocation of funds in the order.

Recommendation

We recommend the client check the logic and fix the issue.

Alleviation

The team acknowledged the issue and stated the following.

The formula of calculating the prices of the options uses the BS model and requires public library `quantlib`. The process is complex and can not be completed on-chain. The calculation process and calculation method can be published to the users.

NVC-03 | No upper limit for `rate`

Category	Severity	Location	Status
Logical Issue	● Minor	projects/Clear Protocal/contracts/NoteV2.sol (826e704): 331	ⓘ Acknowledged

Description

The owner can set the `rate` when deploying the contract and there is no upper limit on what the rate can be. In the extreme case, the rate can be as high as 100%.

Recommendation

We recommend the team set a reasonable upper limit for `rate`, such as 30%.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

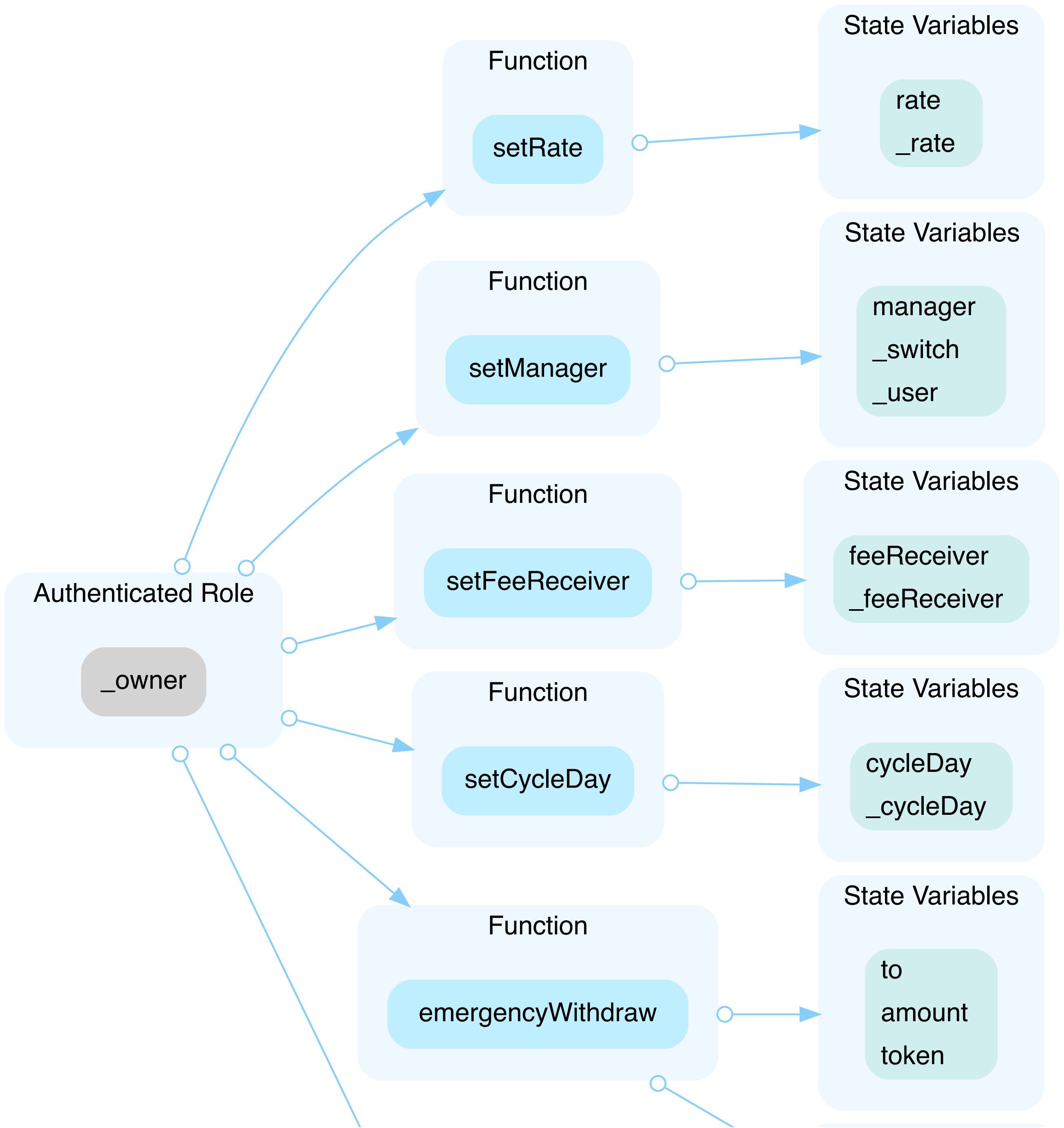
NVC-04 | Centralization Risk: Privileged Role, _owner, in Contract, NoteV2.

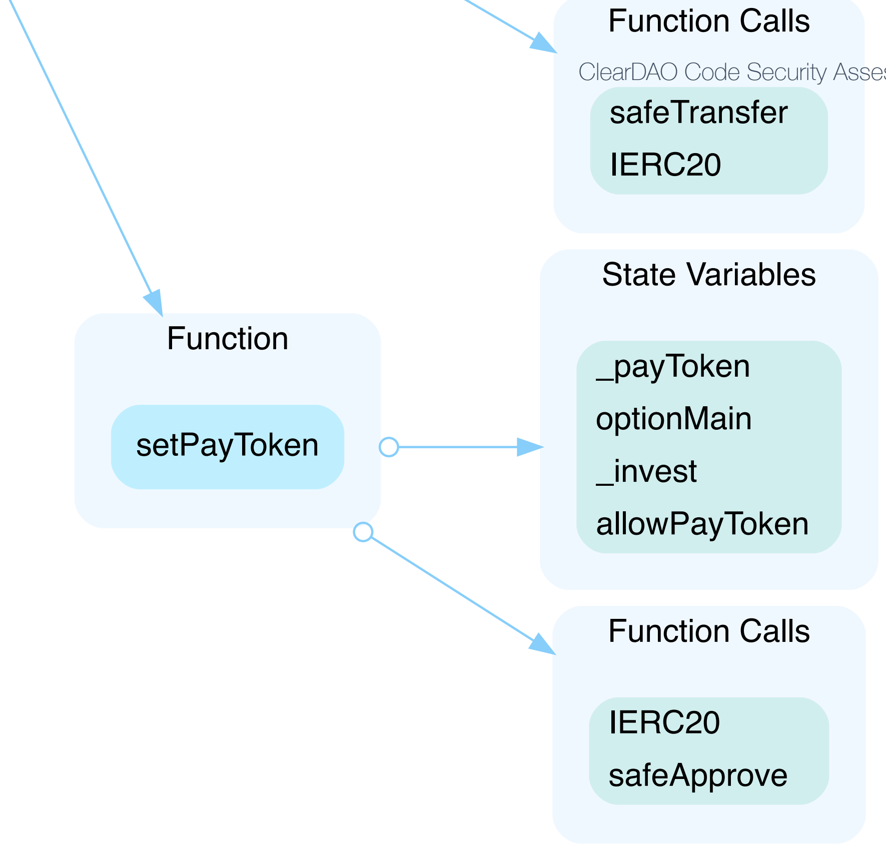
Category	Severity	Location	Status
Centralization / Privilege	● Major	projects/Clear Protocal/contracts/NoteV2.sol (826e704): 330~332, 334~337, 339~341, 356~358, 365~371, 373~381	ⓘ Acknowledged

Description

In the contract, `NoteV2`, the role, `_owner`, has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `_owner` may allow the hacker to take advantage of this.





Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged the issue and stated the following.

The owner of the contract `NoteV2` is the contract `ClearAccessControl` and the owner of the `ClearAccessControl` is a `MultiSigWalelt` address, so its privileges will not be abused. Later on, all rights management will be handed over to DAO governance.

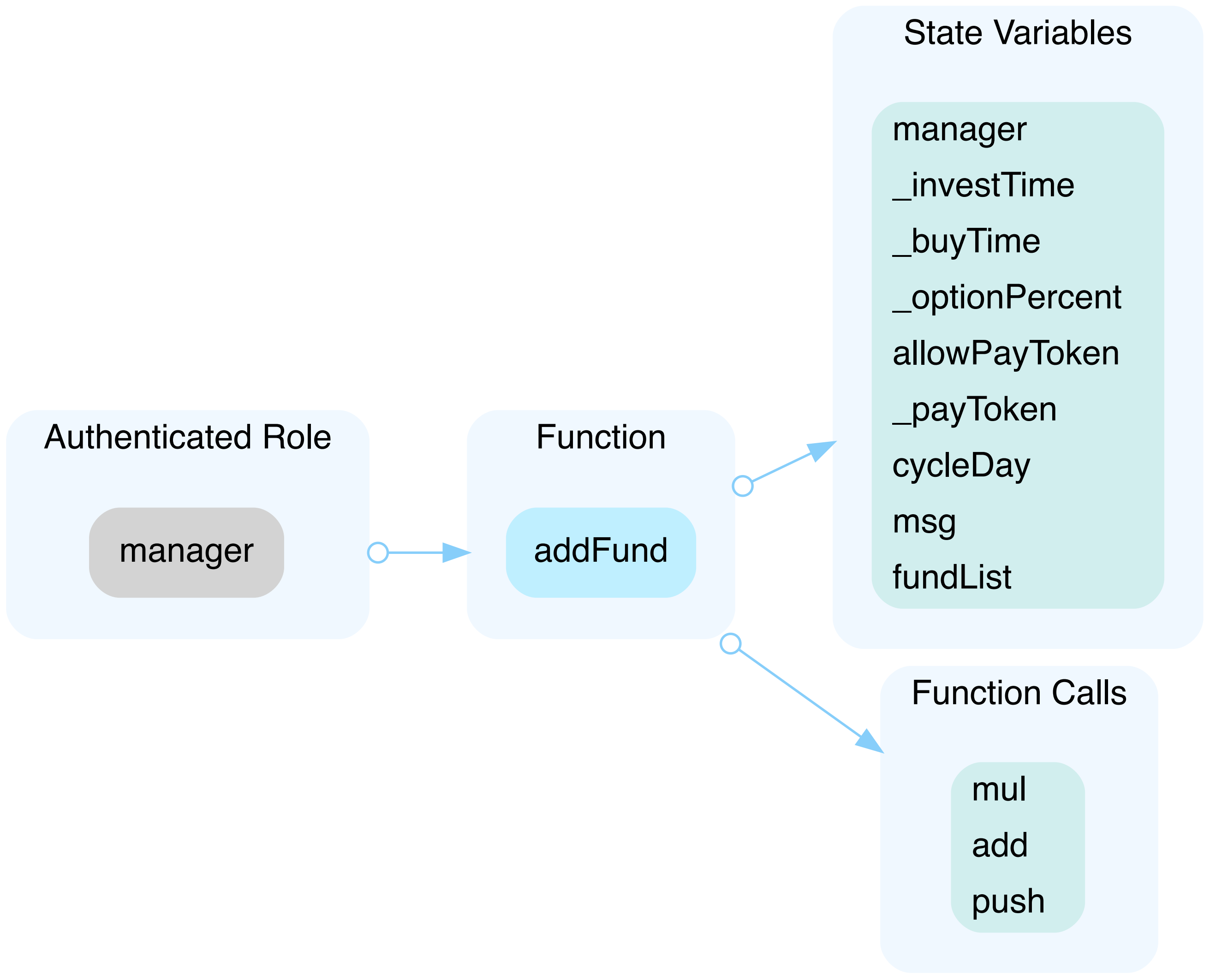
NVC-05 | Centralization Risk: Privileged Role, manager, in Contract, NoteV2.

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	projects/Clear Protocal/contracts/NoteV2.sol (826e704): 93~116	ⓘ Acknowledged

Description

In the contract, `NoteV2`, the role, `manager`, has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `manager` may allow the hacker to take advantage of this.



Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged the issue and stated the following.

The manager role will be granted to `ClearAccessControl`, its owner is a multisig wallet now. Later on, all rights management will be handed over to DAO governance.

NVC-06 | Third party dependencies

Category	Severity	Location	Status
Volatile Code	● Minor	projects/Clear Protocal/contracts/NoteV2.sol (826e704)	ⓘ Acknowledged

Description

The contract is serving as the underlying entity to interact with third party `invest` protocols. The scope of the audit treats 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of [NoteV2] requires interaction with `invest` protocols. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

The team acknowledged this issue and stated the following.

We will listen to the status of the third party invest protocols in real time for exceptions.

NVC-07 | Missing input validation of `_cycleIndex`

Category	Severity	Location	Status
Logical Issue	● Informational	projects/Clear Protocal/contracts/NoteV2.sol (826e704): 97	✓ Resolved

Description

It is recommended to check if the value of `_cycleIndex` is out of range to avoid that the `endTime` of the fund equals `investTime`.

Recommendation

We recommend the team check the logic and fix the issue.

Alleviation

The team heeded our advice and fixed the issue in commit a696e63029f1744db09ffc0ebcea910a3cad3312.

NVI-01 | Potential wrong logic of modifier `onlyOwner()`

Category	Severity	Location	Status
Logical Issue	● Minor	NoteV2Invest.sol: 68~71	✓ Resolved

Description

The modifier `onlyOwner()` has the same implementation with the modifier `whenNotPaused`. And it is widely used in permission control. We would like to confirm with the client if the current implementation aligns with the original project design.

Recommendation

We recommend the team check the logic and fix the issue.

Alleviation

The team heeded our advice and fixed the issue in commit a696e63029f1744db09ffc0ebcea910a3cad3312.

NVI-02 | Logical issue of function `setPayToken()`

Category	Severity	Location	Status
Logical Issue, Control Flow	● Minor	NoteV2Invest.sol: 256	ⓘ Acknowledged

Description

Changes of the mapping between invest tokens and `payToken` may impact funds' withdrawing assets from previous invest token. There should be more checks to safeguard assets of the fund against loss.

Recommendation

We recommend the team add checks to safeguard assets of the fund against loss.

Alleviation

The team acknowledged this issue and they will leave it as it is for now.

NVI-04 | Third party dependencies

Category	Severity	Location	Status
Volatile Code	● Minor	NoteV2Invest.sol	ⓘ Acknowledged

Description

The contract is serving as the underlying entity to interact with third party `invest` protocols. The scope of the audit treats 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of `NoteV2Invest` requires interaction with `invest`, etc. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

The team acknowledged the issue and stated the following.

"We monitor the status of the contract at all times in the background, and in the event of an unforeseen event, multi-signature users use multi-signature to urgently transfer project funds to a secure address."

NVI-05 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	NoteV2Invest.sol	ⓘ Acknowledged

Description

In the contract NoteV2Invest.sol the role `owner` has authority over the functions shown in the diagram below.

- function setPayToken()
- function emergencyWithdraw()

Any compromise to the `owner` account may allow the hacker to take advantage of this authority and change `payToken` or withdraw tokens from the contract.

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 can't 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, were able to *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-functionalities

Alleviation

The team acknowledged this issue and stated the following.

The owner of the contract `NoteV2Invest` is the contract `ClearAccessControl` and the owner of the `ClearAccessControl` is a `MultiSigWalelt` address, so its privileges will not be abused. Later on, all rights management will be handed over to DAO governance.

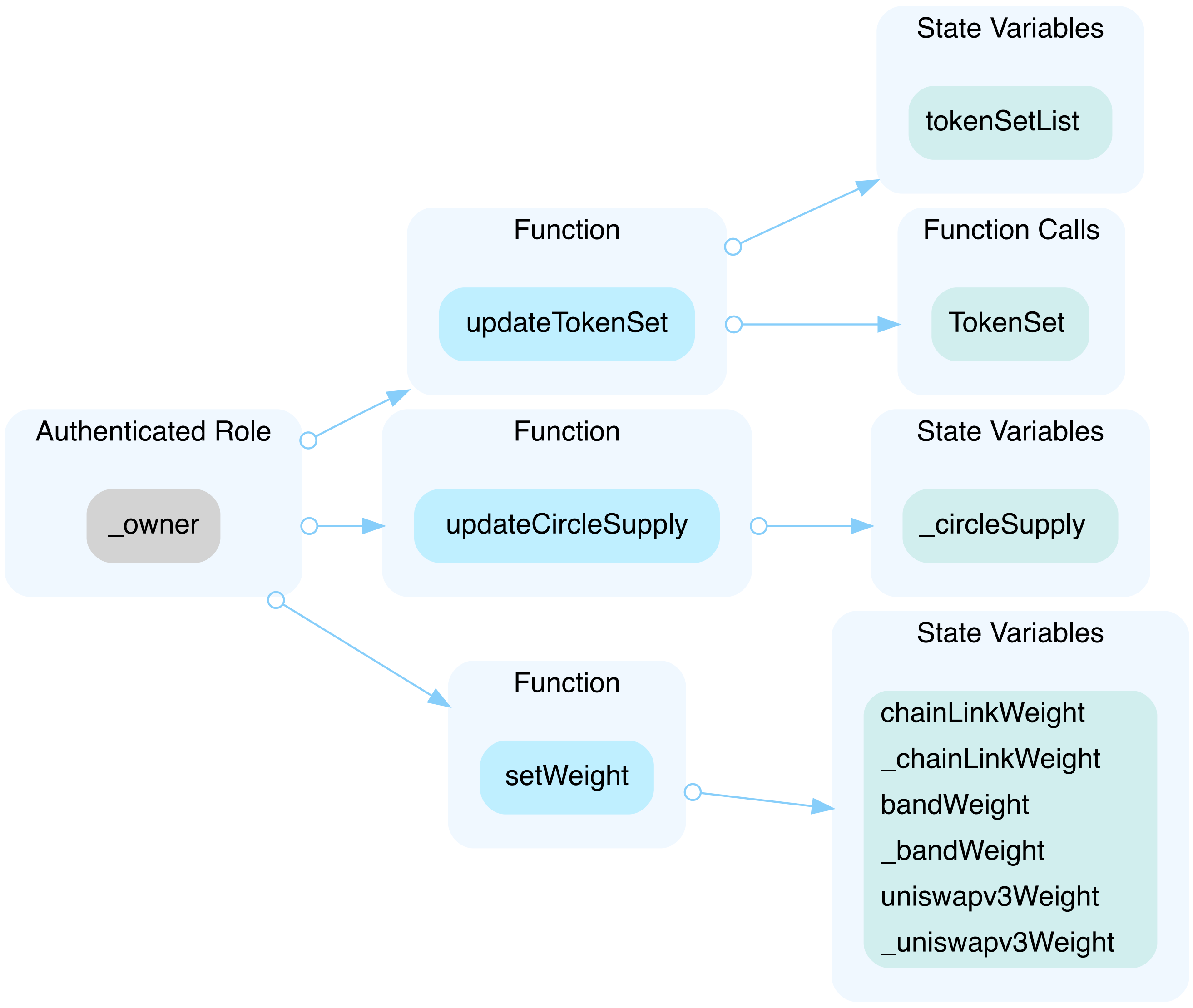
OCP-01 | Centralization Risk: Privileged Role, _owner, in Contract, Oracle.

Category	Severity	Location	Status
Centralization / Privilege	● Major	projects/Clear Protocal/contracts/Oracle.sol (826e704): 44~53, 55~60, 62~70	ⓘ Acknowledged

Description

In the contract, `Oracle`, the role, `_owner`, has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `_owner` may allow the hacker to take advantage of this.



Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged the issue and stated the following.

The role `_owner` of the contract `Oracle` will be granted to `ClearAccessControl`.The owner of the `ClearAccessControl` is a `MultiSigWalelt` address, so its privileges will not be abused. Later on, all rights management will be handed over to DAO governance.

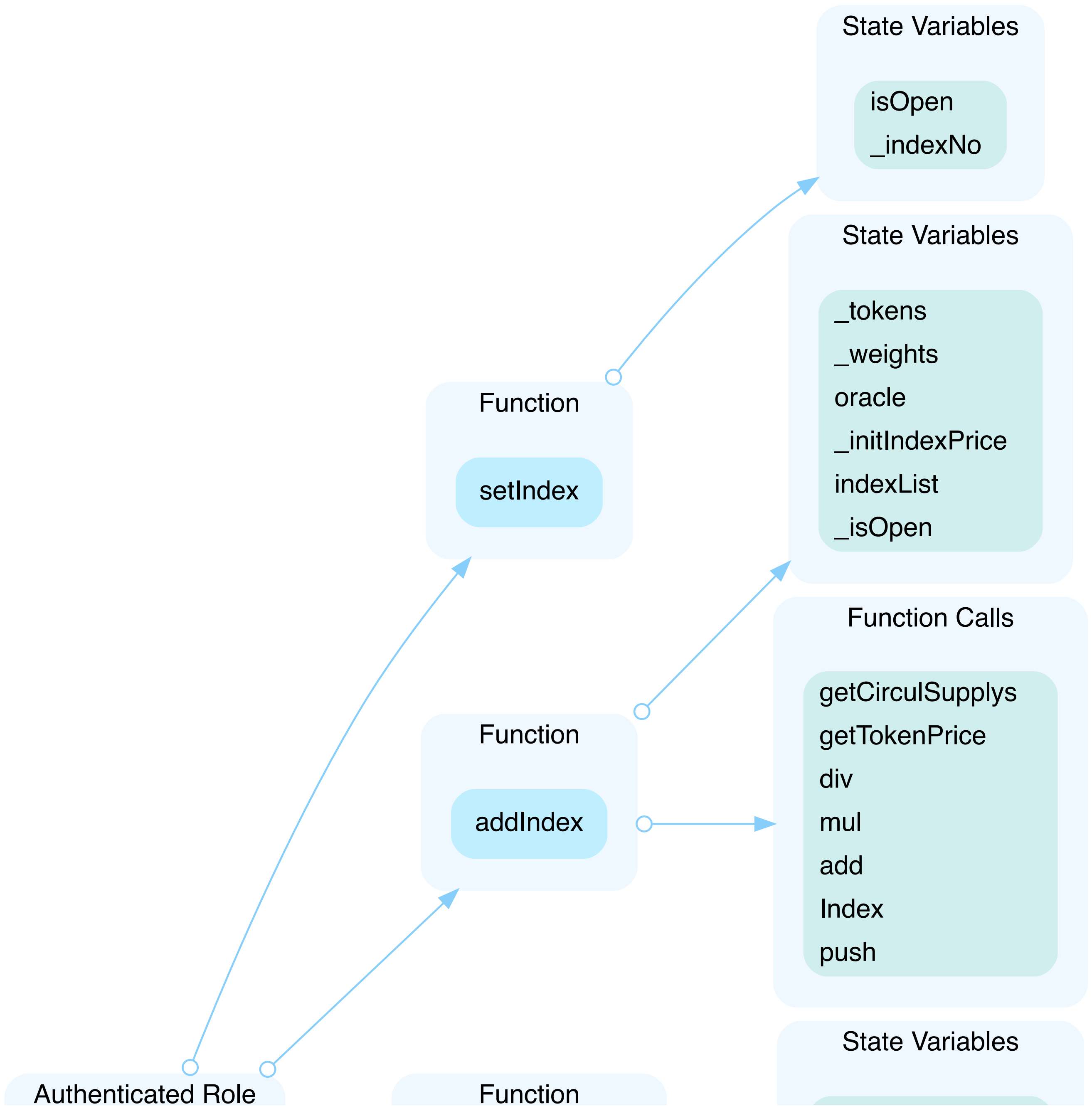
OIC-01 | Centralization Risk: Privileged Role, _owner, in Contract, OptionIndex.

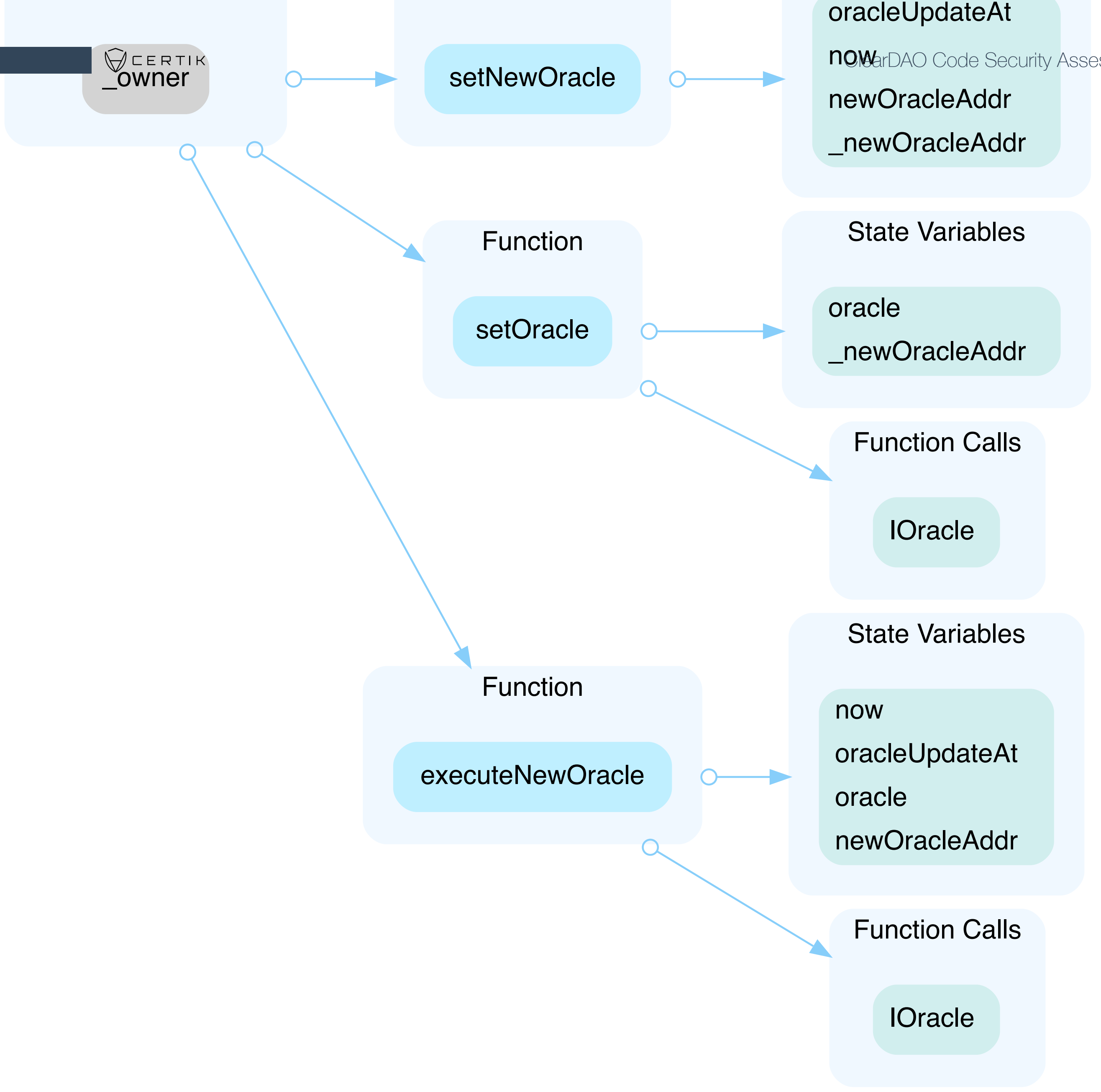
Category	Severity	Location	Status
Centralization / Privilege	● Major	projects/Clear Protocal/contracts/OptionIndex.sol (826e704): 43~46, 48~80, 113~117, 119~121, 123~132	ⓘ Acknowledged

Description

In the contract, `OptionIndex`, the role, `_owner`, has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `_owner` may allow the hacker to take advantage of this.





Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged the issue and stated the following.

The authenticated role will be granted to `ClearAccessControl`. And the owner of the `ClearAccessControl` is a `MultiSigWalelt` address, so its privileges will not be abused. Later on, all rights management will be handed over to DAO governance.

OIC-02 | Logical issue about token decimals

Category	Severity	Location	Status
Logical Issue	● Informational	projects/Clear Protocal/contracts/OptionIndex.sol (826e704): 58~63	ⓘ Acknowledged

Description

When calculating the price of the tokens, it seems that decimals of these tokens are ignored.

Recommendation

We recommend the client make sure the weights of the tokens are set according to their decimals.

Alleviation

The team stated that all prices decimals will be converted to 18 in the `Oracle`.

OIC-03 | Lack of access control

Category	Severity	Location	Status
Logical Issue	● Medium	projects/Clear Protocal/contracts/OptionIndex.sol (826e704): 25	✓ Resolved

Description

The function `setK()` lacks an access control. Any user can call the function to impact the price of the index.

Alleviation

The team fixed the issue and removed the function in commit 902b0e2d349e9c727da19ac6d879cb697a68ad5e.

OIC-04 | Potential flashloan attack

Category	Severity	Location	Status
Control Flow	● Major	projects/Clear Protocal/contracts/OptionIndex.sol (826e704): 60	ⓘ Acknowledged

Description

Flash loans are a way to borrow large amounts of money for a certain fee. The requirement is that the loans need to be returned within the same transaction in a block. If not, the transaction will be reverted.

An attacker can use the borrowed money as the initial funds for an exploit to enlarge the profit and/or manipulate the token price in the decentralized exchanges.

We find that the `OptionIndex` rely on price calculations that are based on-chain, meaning that they would be susceptible to flash-loan attacks by manipulating the price of given pairs to the attacker's benefit.

Recommendation

If a project requires price references, it needs to be caution of flash loans that might manipulate token prices. To minimize the chance of happening, we recommend the client to consider following according to the project's business model.

1. Use multiple reliable on-chain price oracle sources, such as Chainlink and Band protocol.
2. Use Time-Weighted Average Price (TWAP). The TWAP represents the average price of a token over a specified time frame. If an attacker manipulates the price in one block, it will not affect too much on the average price.
3. If the business model allows, restrict the function caller to be a non-contract/EOA address.
4. Flash loans only allow users to borrow money within a single transaction. If the contract use cases allowed, force critical transactions to span at least two blocks.

Alleviation

The team acknowledged the issue and stated the following.

"The price data is provided by chainLink and Band protocol in a 4:2 ratio. The users have to generate signatures for selling after buying options."

OIC-05 | Third party dependencies

Category	Severity	Location	Status
Volatile Code	● Minor	projects/Clear Protocal/contracts/OptionIndex.sol (826e704)	ⓘ Acknowledged

Description

The contract is serving as the underlying entity to interact with third party oracle protocols. The scope of the audit treats 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of `OptionIndex` requires interaction with oracle, etc. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

The team acknowledged this issue and stated the following.

We will constantly listen to the status of third party contracts.

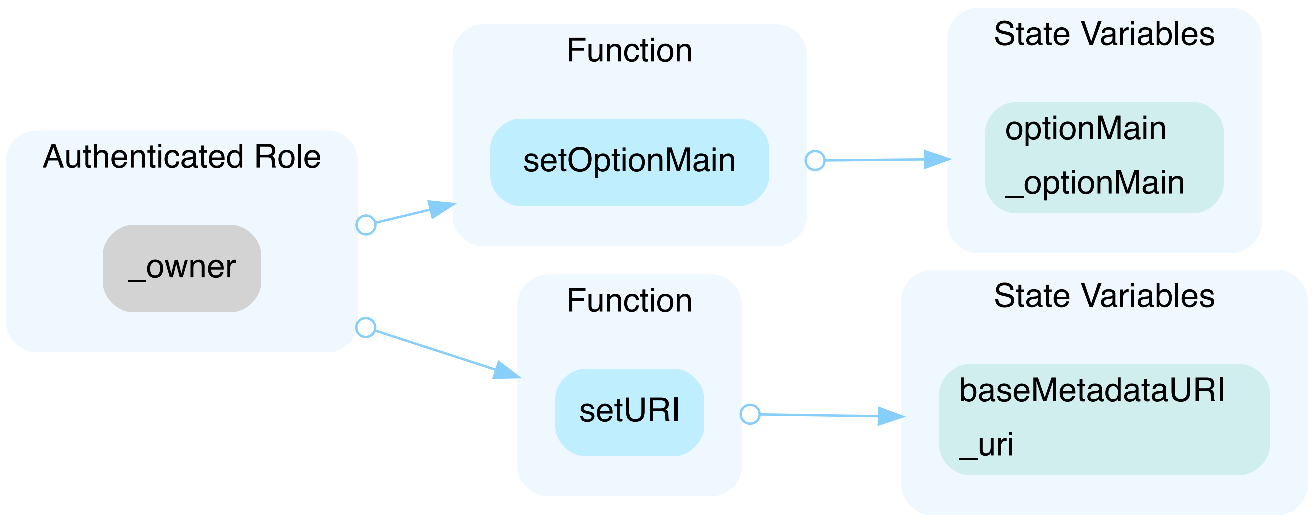
OTC-01 | Centralization Risk: Privileged Role, _owner, in Contract, OptionToken.

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	projects/Clear Protocal/contracts/OptionToken.sol (826e704): 25~28, 30~32	ⓘ Acknowledged

Description

In the contract, `OptionToken`, the role, `_owner`, has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `_owner` may allow the hacker to take advantage of this.



Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged the issue and stated the following.

The `_owner` role will be granted to `ClearAccessControl`. And the owner of the `ClearAccessControl` is a `MultiSigWalelt` address, so its privileges will not be abused. Later on, all rights management will be handed over to DAO governance.

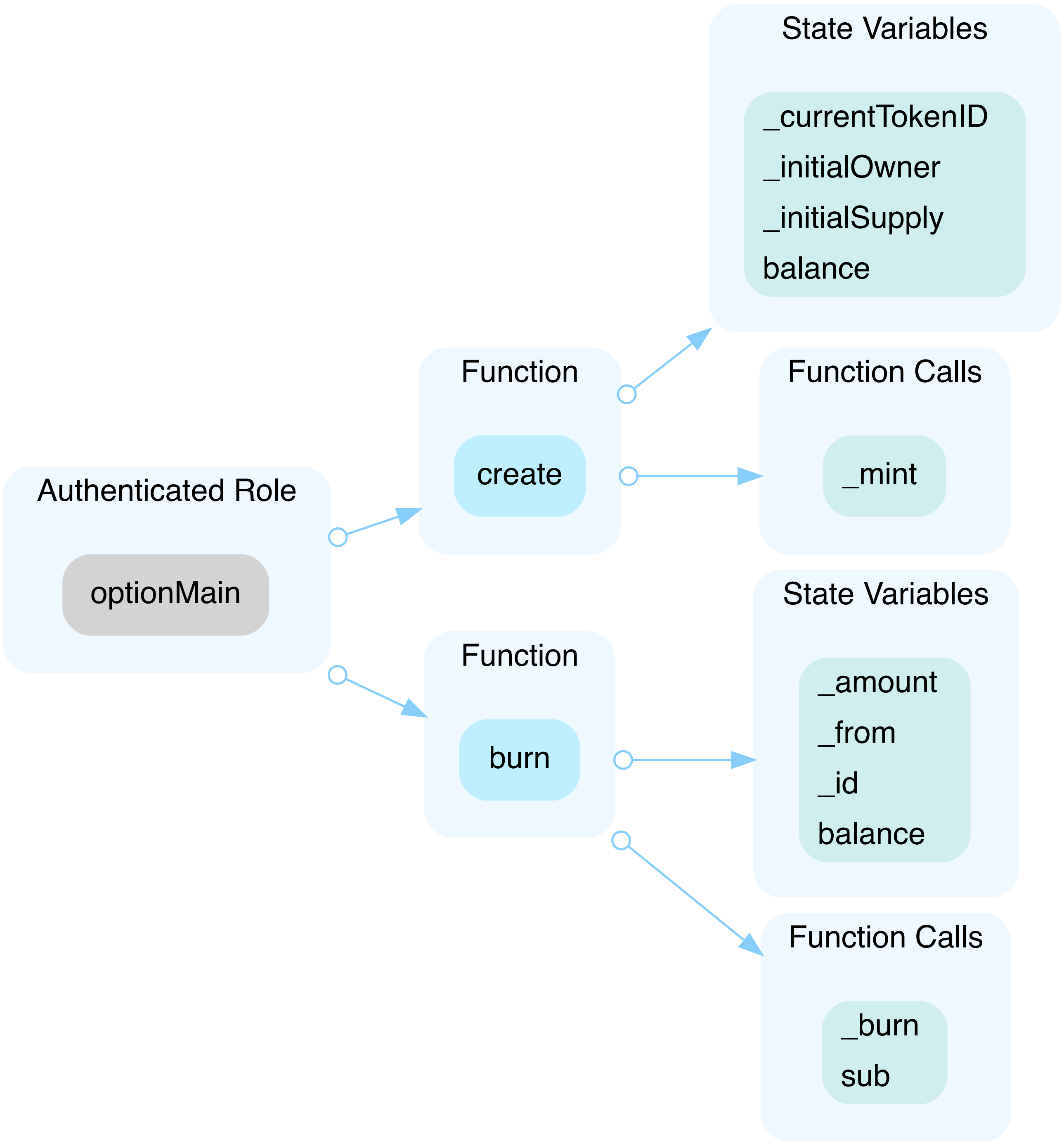
OTC-02 | Centralization Risk: Privileged Role, optionMain, in Contract, OptionToken.

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	projects/Clear Protocal/contracts/OptionToken.sol (826e704): 43~51, 62~70	ⓘ Acknowledged

Description

In the contract, `OptionToken`, the role, `optionMain`, has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `optionMain` may allow the hacker to take advantage of this.



Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged the issue and stated the following.

The role `optionMain` will be granted to `optionV2`, so the privileges won't be abused. The owner of `optionV2` is `ClearAccessControl`. And the owner of the `ClearAccessControl` is a `MultiSigWalelt` address, so its privileges will not be abused. Later on, all rights management will be handed over to DAO governance.

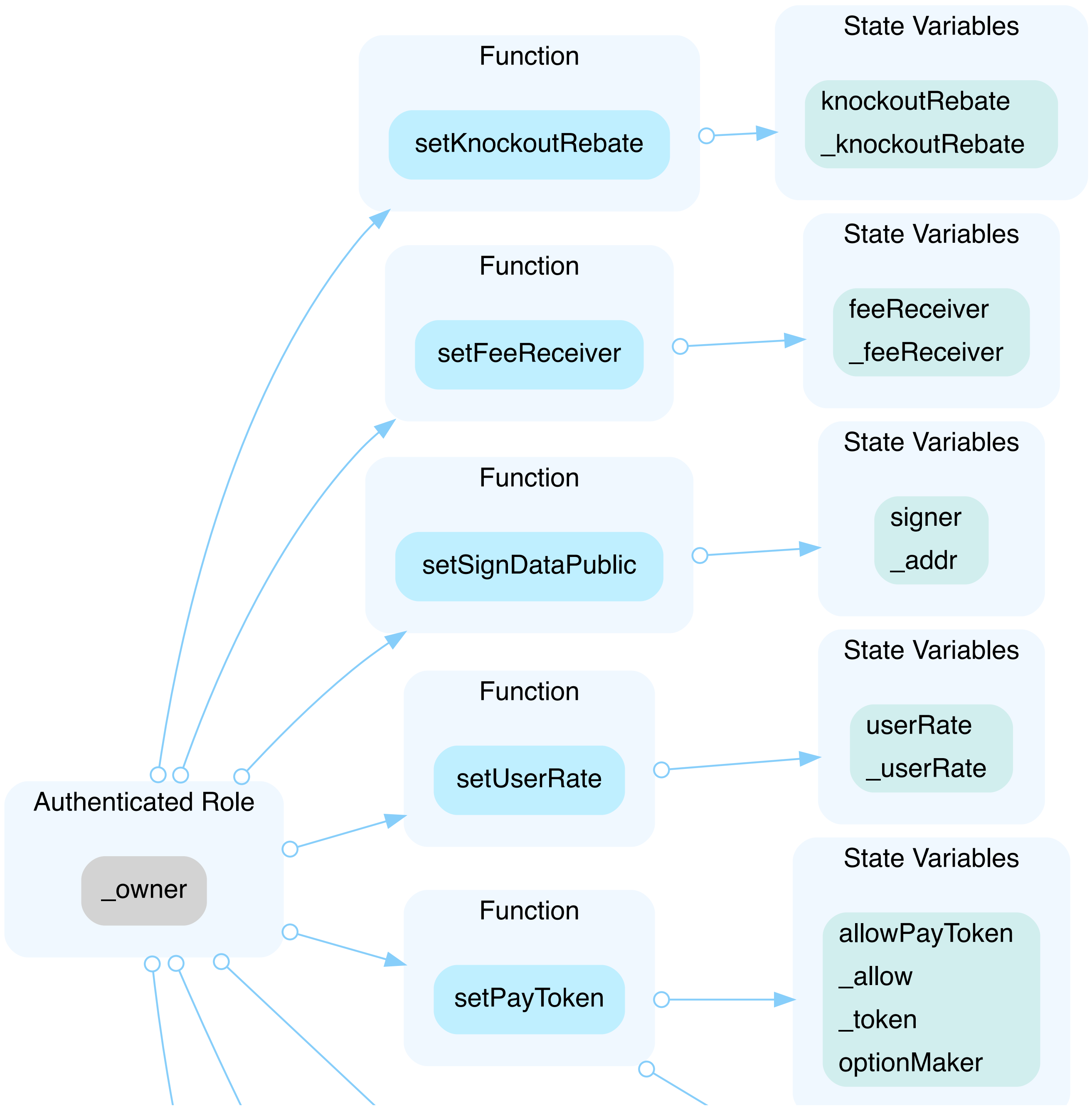
OVC-01 | Centralization Risk: Privileged Role, _owner, in Contract, OptionV2.

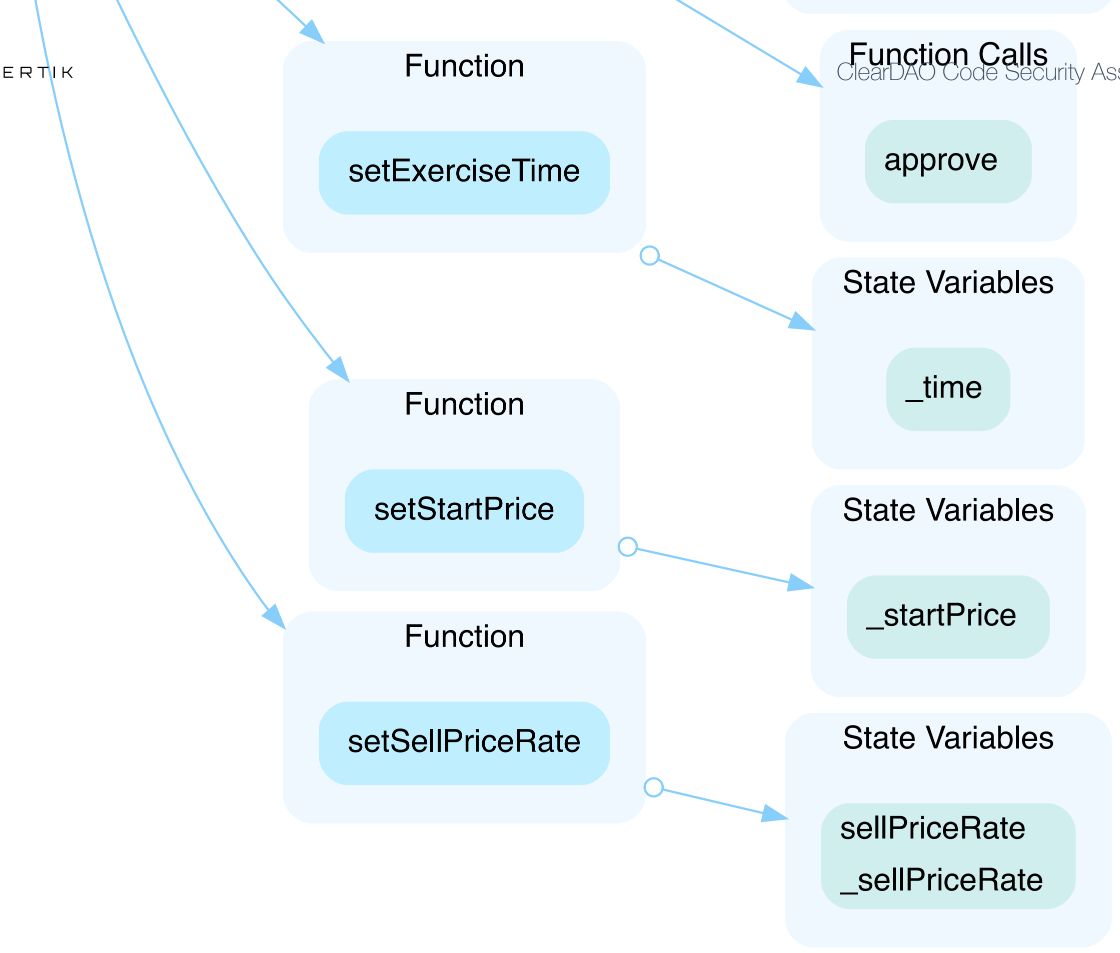
Category	Severity	Location	Status
Centralization / Privilege	● Major	projects/Clear Protocal/contracts/OptionV2.sol (826e704): 439~441, 443~445, 447~449, 451~453, 455~458, 463~468, 473~478, 480~482	ⓘ Acknowledged

Description

In the contract, `OptionV2`, the role, `_owner`, has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `_owner` may allow the hacker to take advantage of this.





Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged the issue and stated the following.

The owner of the contract `optionV2` is the contract `ClearAccessControl` and the owner of the `ClearAccessControl` is a `multisig` wallet, so the privileges will not be abused. Later on, all rights management will be handed over to DAO governance.

OVC-02 | Logical issue of function `buyOption()`

Category	Severity	Location	Status
Logical Issue	● Informational	projects/Clear Protocal/contracts/OptionV2.sol (826e704): 97	✓ Resolved

Description

If `_datetime` represents for buy time, the check should be `_datetime.add(signValidTime) < now`. If it represents for expire time, the check should be `now < _datetime`.

Alleviation

[Certik]

The check `_datetime > now` means that the signature is not expired yet. While `_datetime - now < signValidTime` means that the call can only happen in a time window, from `datetime - signValidTime` to `datetime`.

Assuming that `signValidTime` is 2 minutes and `datetime` is 12:00, then the function can be called during the time span from 11:58 to 12:00.

We would like to confirm with the client if the current implementation aligns with the original project design.

[ClearDAO]

The current implementation aligns with the original project design. We will rename the variable `datetime` to `expiredAt` to avoid misunderstandings.

OVC-03 | Function `sellOption()` does not `addMakerBalance()`

Category	Severity	Location	Status
Logical Issue	● Medium	projects/Clear Protocal/contracts/OptionV2.sol (826e704): 174	✓ Resolved

Description

The function `sellOption()` does not add maker balance via calling the function `addMakerBalance()`. So the makers' funds are lost during this procedure.

We would like to confirm with the client if the current implementation aligns with the original project design.

Recommendation

We recommend the team check the logic and fix the issue.

Alleviation

The team confirmed that the current implementation aligns with the original project design and stated the following.

“When the users call the function `sellOption()` to sell options, their positions are reduced. The calculation can only be determined after the investment stage. ”

OVC-04 | Cases of selling option

Category	Severity	Location	Status
Logical Issue	● Informational	projects/Clear Protocal/contracts/OptionV2.sol (826e704): 184	✓ Resolved

Description

Functions `sellOption()`, `userExercise()`, and `uesrKnockout()` are three ends of the options. What is the difference between `sellOption()` and the other two?

Alleviation

The team stated the following.

"Clear is a knockout option, which is a European option in terms of the exercise method. sellOption means that the user closes the position early, and in order to maintain fairness and not to impact the profits of makers, the selling price is calculated by the option price formula multiplied by `sellPriceRate`/10000; userExercise means that the option is exercised at the expiration date, and the user has not knocked out during the period; userKnockout means that the option is knocked out during the term, but the user is allowed to get back a portion of the purchase cost, calculated as `open_price`*`position`*knockoutRebate/10000"

OVC-05 | Weak control on order's status

Category	Severity	Location	Status
Logical Issue	● Medium	projects/Clear Protocal/contracts/OptionV2.sol (826e704)	✓ Resolved

Description

The contract `OptionV2` has a weak control on order's status.

Cases:

- Only `makerExercise()` and `makerKnockout()` check the order's status and change its status.
- If `userKnockout()` is called before `makerKnockout()`, the `_price` is calculated once more and different from that is used in `userKnockout()`. And the `_userProfit` is subbed twice from the `_order.balance`.
- If `makerKnockout()` is called, `userExercise()` and `sellOption()` can still be called.

The lack of control between the four function calls can bring unpredictable chaos.

Recommendation

We recommend the team check the logic carefully and complete the control flow.

Alleviation

[ClearDAO]

The value of knockout in case 2 is determined by the user's position. If `userKnockout()` is executed first, then the knockout part will be 0 as the user has burned all position.

And in case 3, we will add status checks.

The team fixed the case 3 in commit a696e63029f1744db09ffc0ebcea910a3cad3312.

OVC-06 | Logical issue of function `userExercise()`

Category	Severity	Location	Status
Logical Issue	● Medium	projects/Clear Protocal/contracts/OptionV2.sol (826e704): 255~277	✓ Resolved

Description

The function `userExercise()` does not return principal of the order. Function `userKnockout()` returns `position * price * rebaserate` while the function `userExercise()` just return `position * |exercisePrice - startPrice|`.

If it represents for pure profit, then NFT token should not be burned.

Recommendation

We recommend the client check the logic and fix the issue.

Alleviation

The team stated that the price of the order does not contain the principal(price of the index) as well, so the user can get all payout in the function `userExercise()`.

OVC-07 | Logical issue about `knockoutRebate`

Category	Severity	Location	Status
Logical Issue	● Informational	projects/Clear Protocal/contracts/OptionV2.sol (826e704)	✓ Resolved

Description

The state variable `knockoutRebate` is used in orders with different pools. The global `knockoutRebate` may not be able to cope with 5%, 7.5%, 15%(knockout margin) cases.

Recommendation

We recommend the team check the logic and fix the issue.

Alleviation

The team declined the issue and stated the following.

"The value of `knockoutRebate` is the percentage of the cost that can be retrieved by the user when knocked out. It must be less than 100% and have nothing to do with the margin of the pools."

OVC-08 | Insufficient funds due to high `knockoutRebate`

Category	Severity	Location	Status
Logical Issue	● Minor	projects/Clear Protocal/contracts/OptionV2.sol (826e704)	✓ Resolved

Description

If the `knockoutRebate` is higher than margin, then the maker balance that is locked in function `_addOrder` may be insufficient to cover the option rewards.

Recommendation

We recommend the client check the logic and fix the issue.

Alleviation

The team declined the issue and stated the following.

"The value of `knockoutRebate` is the percentage of the cost that can be retrieved by the user when knocked out. It must be less than 100% and have nothing to do with the margin of the pools."

OVC-09 | Logical issue of function `sellOption()`

Category	Severity	Location	Status
Logical Issue	● Medium	projects/Clear Protocal/contracts/OptionV2.sol (826e704): 174	ⓘ Acknowledged

Description

The price of the order is determined by the parameter `_price`. But the amount of selling option should be based on `order.price`.

Recommendation

We recommend the client check the logic and fix the issue.

Alleviation

The team acknowledged the issue and stated the following.

The option price is calculated based on the option price announcement and it is not possible to verify that the price is correct in the contract, only that it is signed and certified by the platform, and that the price may be profitable within a certain range, but it is not possible to exceed the margin plus premium.

OVC-10 | Logical issue of function `userKnockout()`

Category	Severity	Location	Status
Logical Issue	● Informational	projects/Clear Protocal/contracts/OptionV2.sol (826e704): 310~314	✓ Resolved

Description

The function returns `order.price * position * knockoutRebate` to the fund and burns the NFT tokens from the fund. So is it right that `knockoutRebate` contains the principal part(not only profit)?

Alleviation

The team confirmed that the current implementation aligns with the original project design and stated the following.

"userKnockout is a compensation when the user is knocked out and is returned based on a percentage of the original buy price."

OVC-11 | Logical issue of function `makerKnockout()`

Category	Severity	Location	Status
Logical Issue	● Medium	projects/Clear Protocal/contracts/OptionV2.sol (826e704): 353	✓ Resolved

Description

The price could be different from the price used in the function `userKnockout`. The price is not recorded in function `userKnockout` and the reread price could be different and leads to the check failed and reversion. And the makers cannot take their money back.

Recommendation

We recommend the client check the logic and fix the issue.

Alleviation

The team declined the issue and stated the following.

"In this case, there is no difference between `makerKnockout()` and `makerExercise()` because all the user's position has been removed and funds are claimed from the order."

OVC-12 | Control flow of the option

Category	Severity	Location	Status
Control Flow	● Minor	projects/Clear Protocal/contracts/OptionV2.sol (826e704)	✓ Resolved

Description

If the function `userExercise()` and `userKnockout()` are called before the function `makerExercise()` and `makerKnockout()`, then the `_order.balance` will be subbed twice the profit amount.

There need restrictions on these function to ensure these functions are called in order.

Recommendation

We recommend the team check the logic and fix the issue.

Alleviation

The team declined the issue and stated the following.

"UserKnockout and MakerKnockout have different logic inside. If the user has called the function `userKnockout`, then his/her position should be 0. The maker profit is the same whether makerExercise() or makerKnockout() is called."

OVM-01 | Centralization Risk: Privileged Role, _owner, in Contract, OptionV2Maker.

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	projects/Clear Protocal/contracts/OptionV2Maker.sol (826e704): 83~100, 276~278, 280~286, 288~293	ⓘ Acknowledged

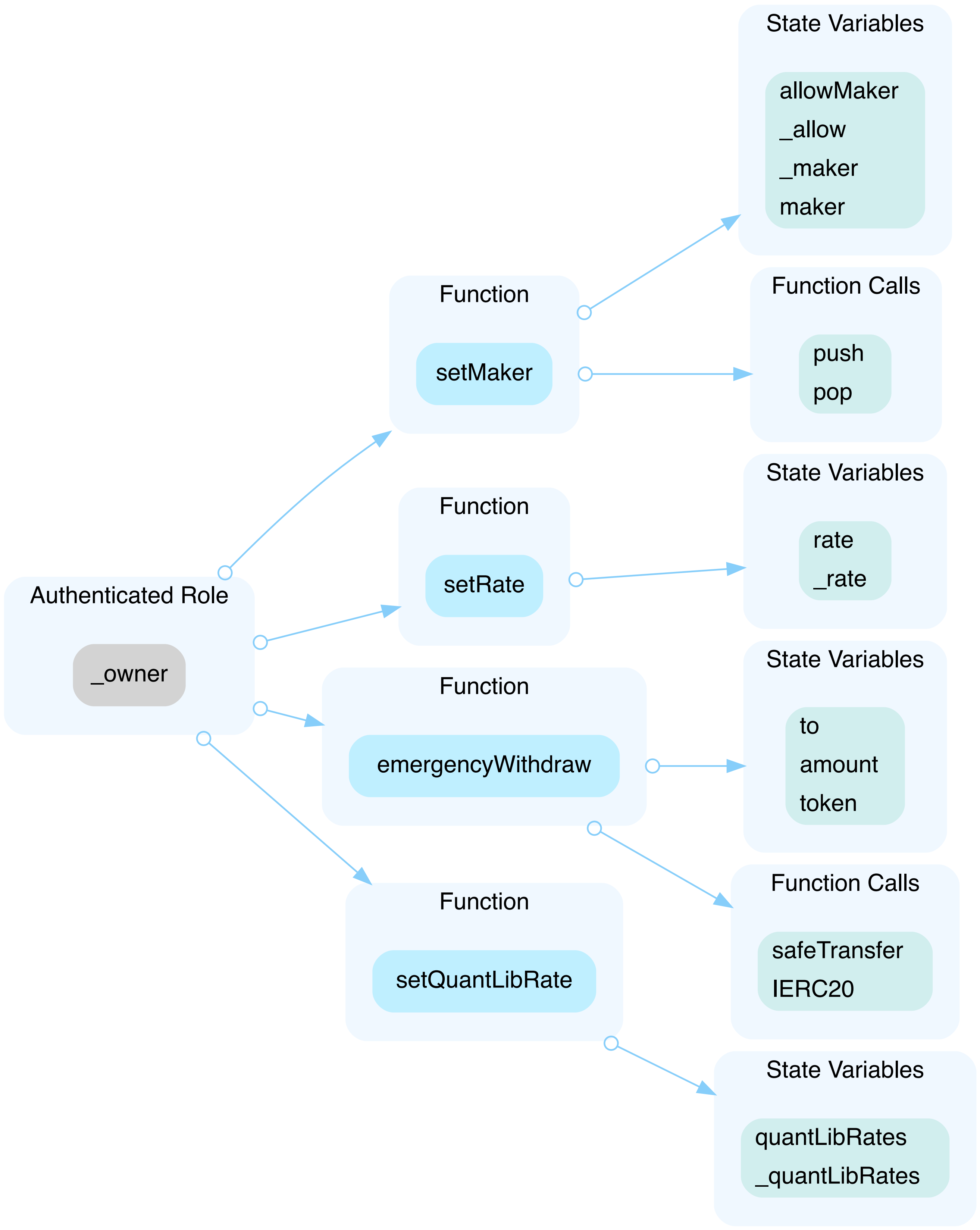
Description

Note: this finding is based on Certik Syntactic Analyzer.

Please review the client's code to see if any modification is required.

In the contract, `OptionV2Maker`, the role, `_owner`, has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `_owner` may allow the hacker to take advantage of this.



Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged the issue and stated the following.

"The management rights of the contract belong to ClearAccessControl."

OVM-02 | Centralization Risk: Privileged Role, optionMain, in Contract, OptionV2Maker.

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	projects/Clear Protocal/contracts/OptionV2Maker.sol (826e704): 76~81, 155~194, 196~215	ⓘ Acknowledged

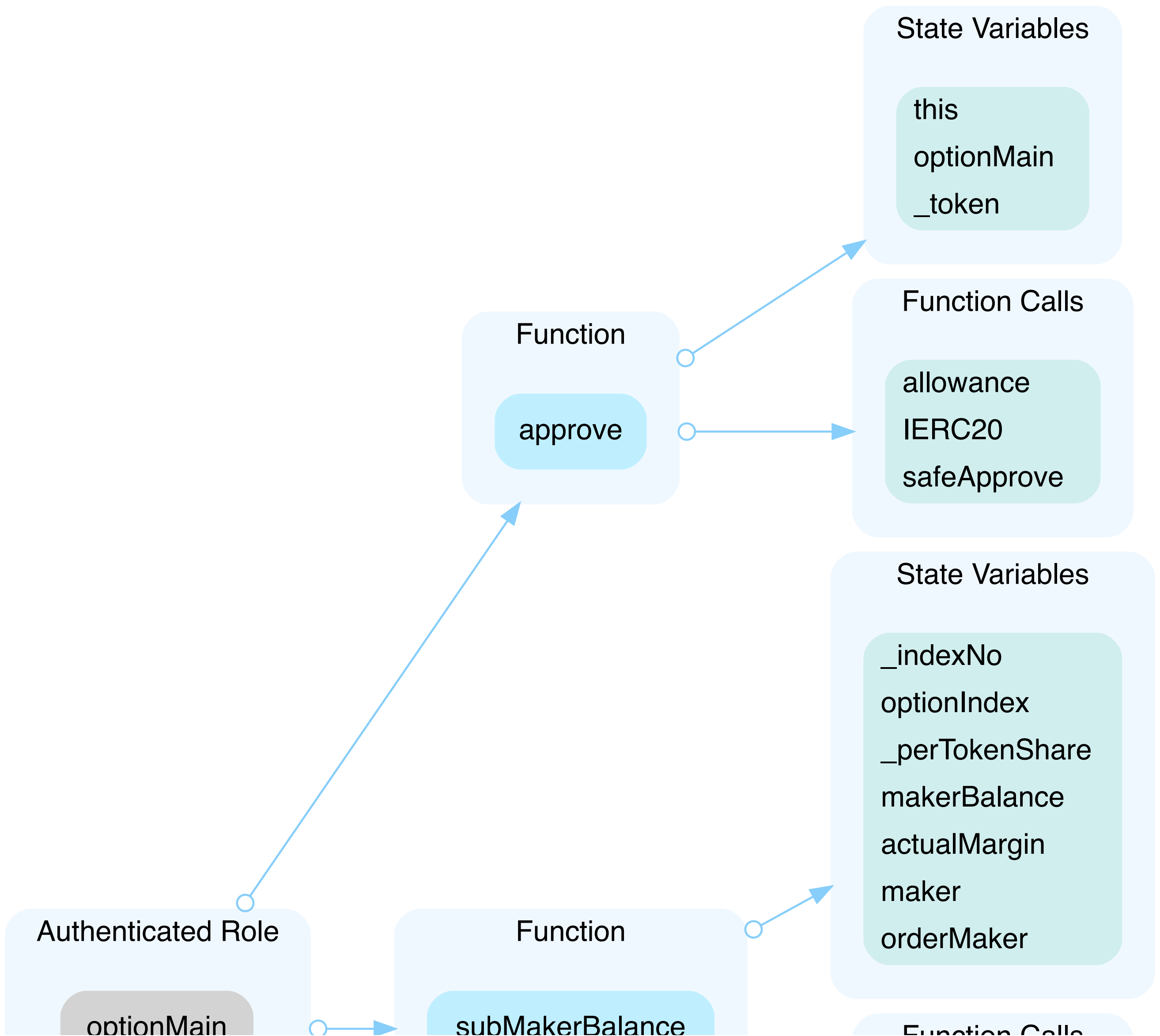
Description

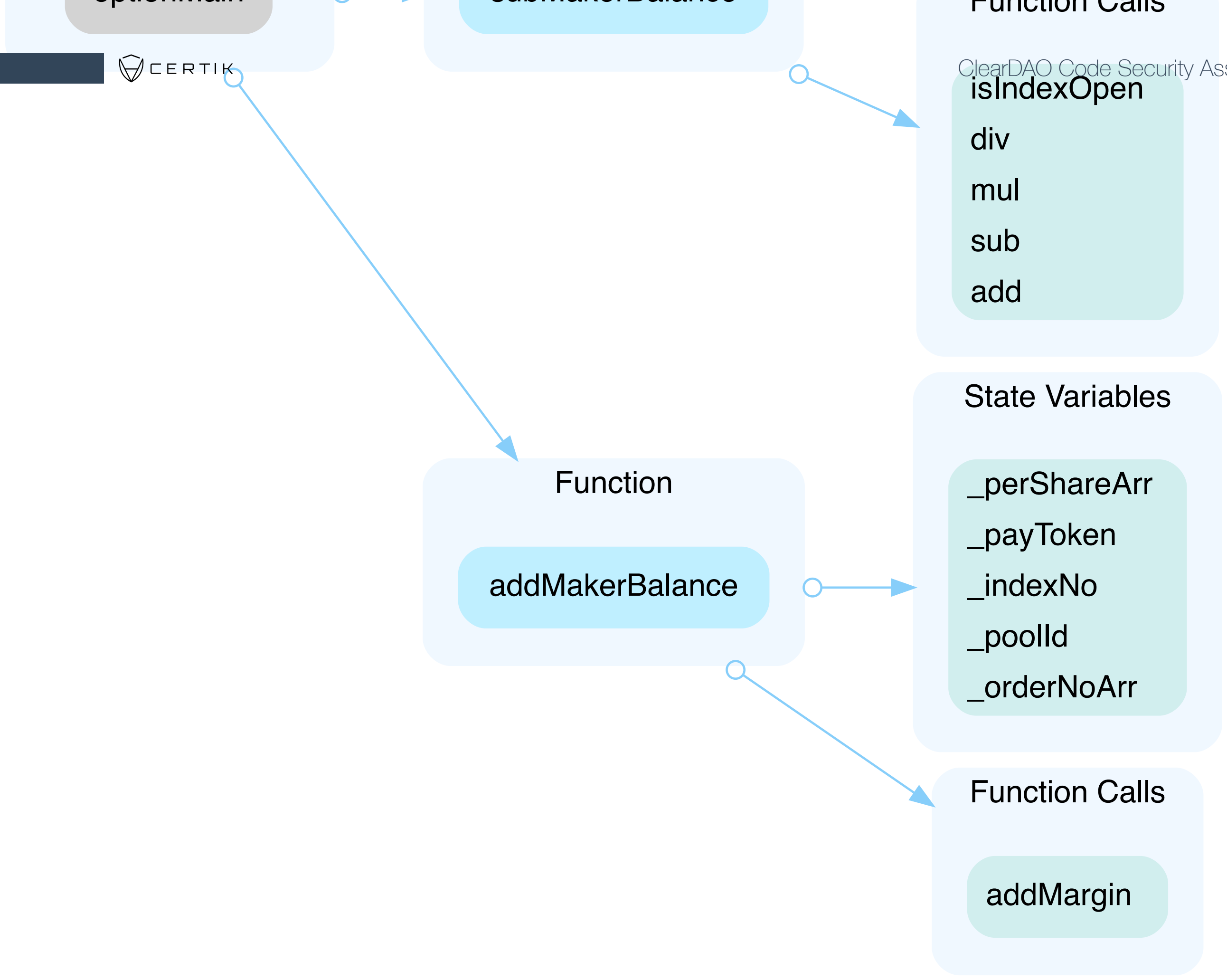
Note: this finding is based on Certik Syntactic Analyzer.

Please review the client's code to see if any modification is required.

In the contract, `OptionV2Maker`, the role, `optionMain`, has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `optionMain` may allow the hacker to take advantage of this.





Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged the issue and stated the following.

"The role `optionMain` in the contract is the contract `optionV2`, there is no abuse of privileges. The owner of `optionV2` is `ClearAccessControl`. And the owner of the `ClearAccessControl` is a `MultiSigWalelt`

address, so its privileges will not be abused. Later on, all rights management will be handed over to DAO governance."

OVP-01 | Centralization Risk: Privileged Role, _owner, in Contract, OwnableV2.

Category	Severity	Location	Status
Centralization / Privilege	● Major	projects/Clear Protocal/contracts/OwnableV2.sol (826e704): 47~50, 56~63	ⓘ Acknowledged

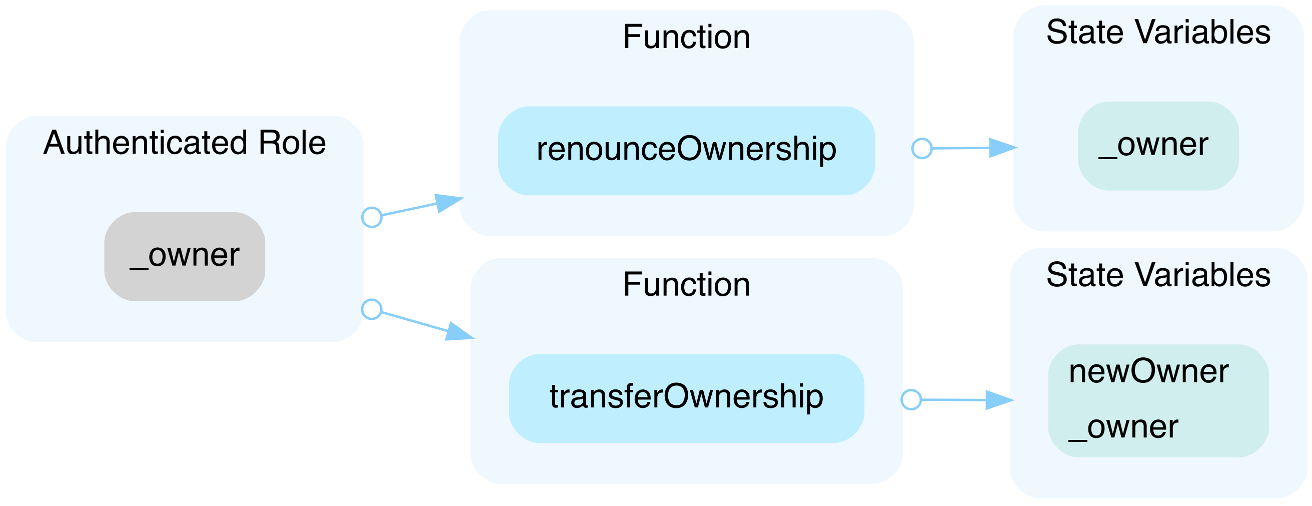
Description

Note: this finding is based on Certik Syntactic Analyzer.

Please review the client's code to see if any modification is required.

In the contract, `OwnableV2`, the role, `_owner`, has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `_owner` may allow the hacker to take advantage of this.



Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged this issue and they stated the following.

"The contract `OwnableV2` won't be deployed independently. As a part of the contract `OptionV2` or `NoteV2`, its owner is controlled and managed by `ClearAccessControl`."

PCP-01 | Centralization Risk: Privileged Role, _owner, in Contract, Pausable.

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	projects/Clear Protocal/contracts/Pausable.sol (826e704): 35~38, 43~46	ⓘ Acknowledged

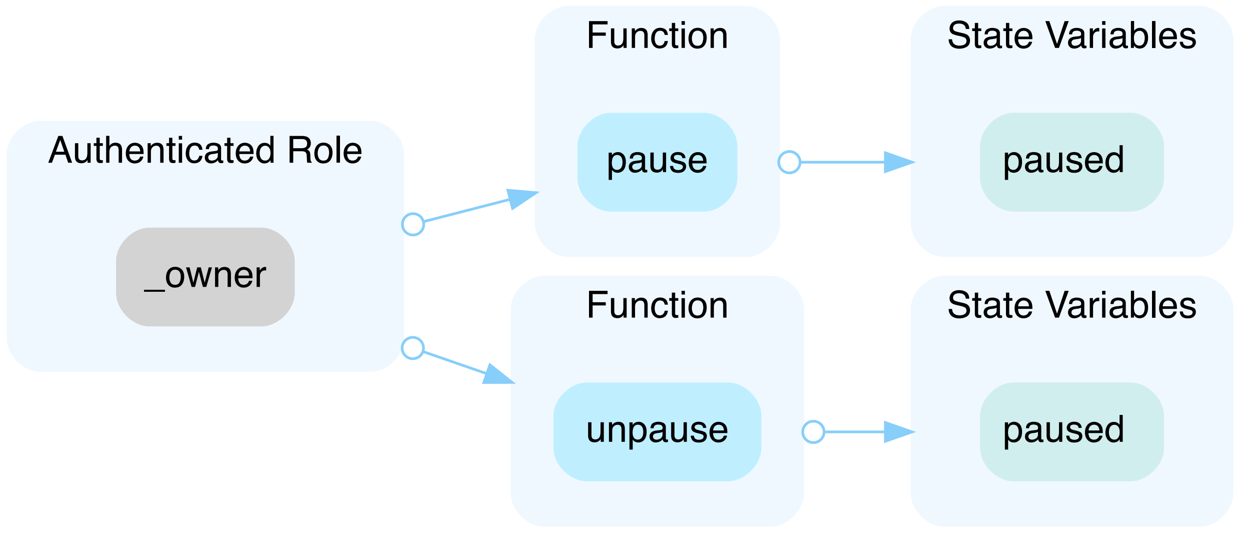
Description

Note: this finding is based on Certik Syntactic Analyzer.

Please review the client's code to see if any modification is required.

In the contract, `Pausable`, the role, `_owner`, has the authority over the functions shown in the diagram below.

Any compromise to the privileged account which has access to `_owner` may allow the hacker to take advantage of this.



Recommendation

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 to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged this issue and they stated the following.

"The contract `Pausable` won't be deployed independently. As a part of the contract `OptionV2` or `NoteV2`, its owner is controlled and managed by `ClearAccessControl`."

Appendix

Finding Categories

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.

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.

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

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

About

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.