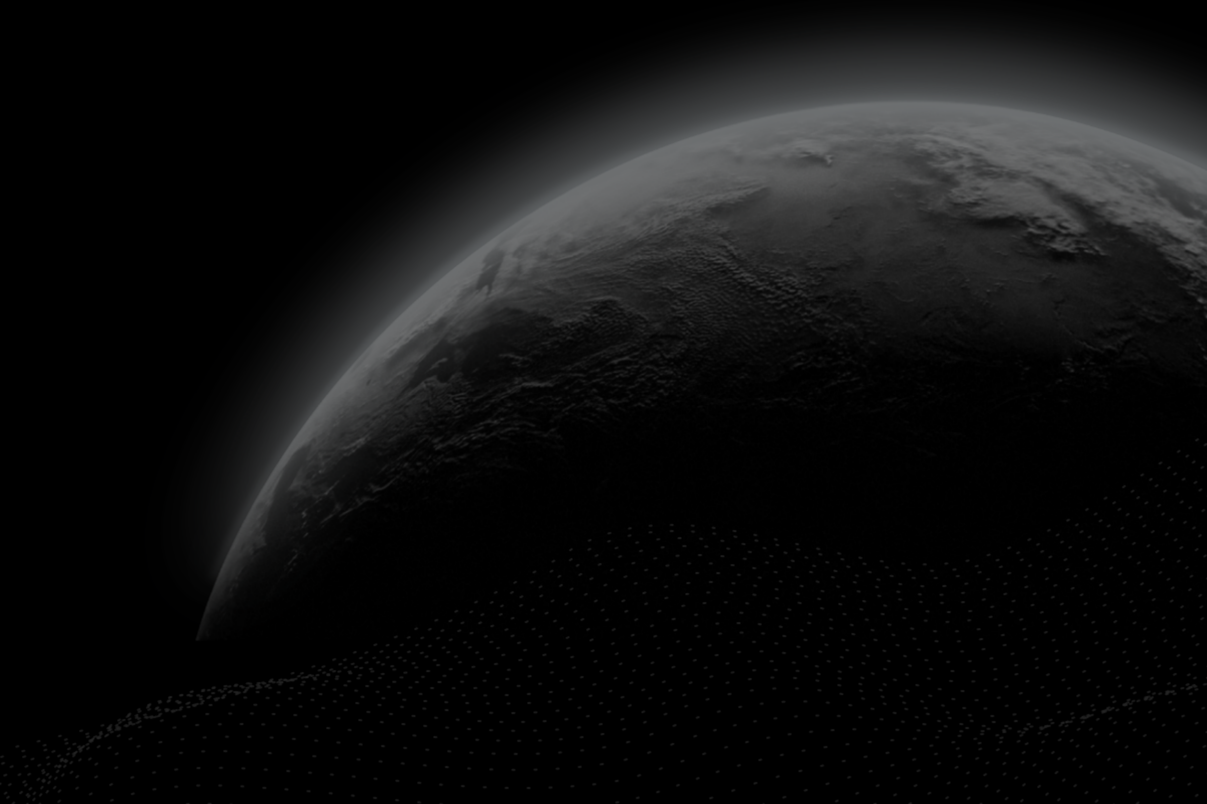
CERTIK

Security Assessment

PHI

CertiK Verified on Nov 15th, 2022



CertiK Verified on Nov 15th, 2022

PHI

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Ethereum	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 11/15/2022	N/A

CODEBASE

https://github.com/PHI-LABS-INC/Mumbai-phi-contract

...View All

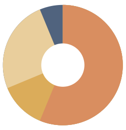
COMMITS

base: d508e4bcb84567fc2b1990e28bbaa7e61b409217

update: 93cb93dfc7fcda682f800dd37ed3d353d6d8f1ad

...View All

Vulnerability Summary



16	7	9	0	0	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.
🟧	9	Major	9 Mitigated	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 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	4	Minor	4 Resolved	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
🟦	1	Informational	1 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.
⬜	0	Discussion		The impact of the issue is yet to be determined, hence requires further clarifications from the project team.

TABLE OF CONTENTS | PHI

CODEBASE | PHI

Repository

https://github.com/PHI-LABS-INC/Mumbai-phi-contract

Commit

base: d508e4bcb84567fc2b1990e28bbaa7e61b409217

update: 93cb93dfc7fcda682f800dd37ed3d353d6d8f1ad

AUDIT SCOPE | PHI

11 files audited ● 8 files with Mitigated findings ● 1 file with Resolved findings ● 2 files without findings

ID	File	SHA256 Checksum
● BPM	contracts/object/BasePlate.sol	4c8ab914035e8c2b85dfd6113e782506ea5485a964fca7d74ce39f90280d73bd
● FOM	contracts/object/FreeObject.sol	1ef8b9e1e18ddfba0bd9d61bd593768f811f71479f5ce486bb46e4e6aa919ce5
● POM	contracts/object/PremiumObject.sol	a6fed7bac03f40910917263959c244d1f6f82aa0b737a852db7e89470195ef08
● QOM	contracts/object/QuestObject.sol	be0a837811c523e206a22a093b20710efd74c166c0895b54afb56b2fa5eec806
● BOM	contracts/utils/BaseObject.sol	fe55f7ec77769763cc1e7fbfbc3b1d6552257a2415ea899f0e501f982ba84478
● PCM	contracts/PhiClaim.sol	602ee9ce0f7c0378ceed208dd2b178cf9f8a43f14af7a6dffb9e88604281f963
● PMM	contracts/PhiMap.sol	31952b25c05a885d8b0d9590086f065f1b59751738c0d818c9e6b3e69986d6cb
● RMP	contracts/Registry.sol	7fffc2cc4b233be692a315f678bd57e8b0b77adb46b92dd8c47a153948893451
● WPM	contracts/object/WallPaper.sol	d95b772a0beeef802efe5b717a2589533aba5ea820e01dc9b0f6c0b83b933314
● MOM	contracts/utils/MultiOwner.sol	551728871e1e999663aef6b5c8fdc90b5a1cc38060a99b32905fcb61e9ec7655
● PSM	contracts/PhiShop.sol	630250d2ae254f3ccb8f39dfde8552c87c8553a670deb6117537ca40746751f9

APPROACH & METHODS | PHI

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

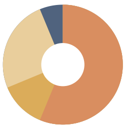
The auditing process pays special attention to the following considerations:

- 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 | PHI



16	0	9	2	4	1	0
Total Findings	Critical	Major	Medium	Minor	Informational	Discussion

This report has been prepared to discover issues and vulnerabilities for PHI. Through this audit, we have uncovered 16 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
BOM-01	**Centralization Risks In BaseObject.Sol**	**Centralization / Privilege**	**Major**	● **Mitigated**
BPM-01	**Centralization Risks In BasePlate.Sol**	**Centralization / Privilege**	**Major**	● **Mitigated**
FOM-01	**Centralization Risks In FreeObject.Sol**	**Centralization / Privilege**	**Major**	● **Mitigated**
MPH-01	**Centralized Control Of Contract Upgrade**	**Centralization / Privilege**	**Major**	● **Mitigated**
MPH-02	Potential Reentrancy Attack (Out-Of-Order Events)	Volatile Code	Minor	● Resolved
MPH-03	Missing Override Specifier	Compiler Error	Minor	● Resolved
PCM-01	**Centralization Risks In PhiClaim.Sol**	**Centralization / Privilege**	**Major**	● **Mitigated**
PCM-02	Old Coupons Will Fail Verification	Logical Issue	Minor	● Resolved
PMM-01	**Centralization Risks In PhiMap.Sol**	**Centralization / Privilege**	**Major**	● **Mitigated**
PMM-02	Potential Reentrancy Attack	Volatile Code	Medium	● Resolved

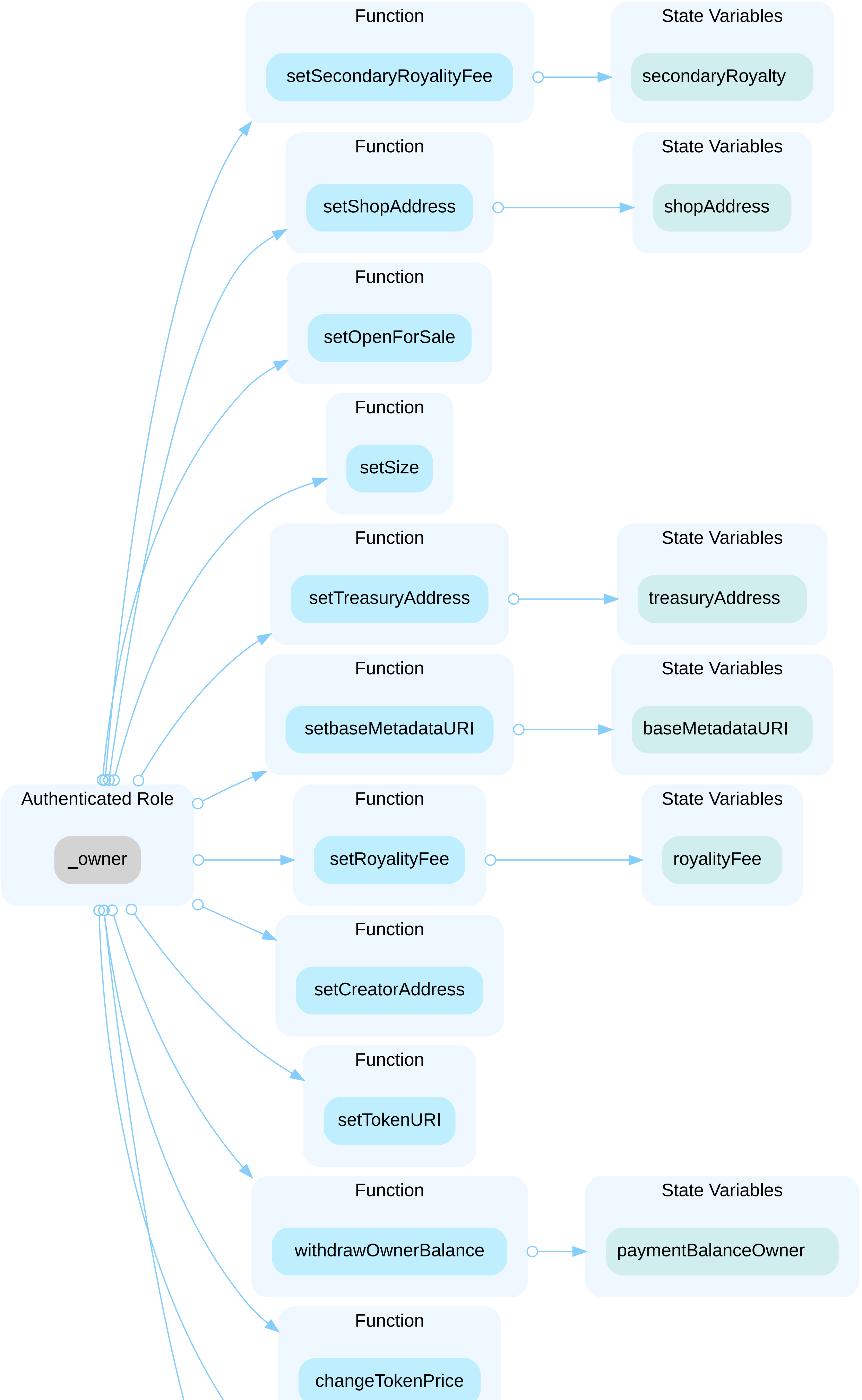
ID	Title	Category	Severity	Status
PMM-03	Lack Of Access Restriction On Function "_changeBasePlate()`	Logical Issue	Medium	● Resolved
PMM-04	Contract `ReentrancyGuardUpgradeable.sol` Is Not Used	Inconsistency	Minor	● Resolved
POM-01	**Centralization Risks In PremiumObject.Sol**	**Centralization / Privilege**	**Major**	● **Mitigated**
QOM-01	**Centralization Risks In QuestObject.Sol**	**Centralization / Privilege**	**Major**	● **Mitigated**
RMP-01	**Centralization Risks In Registry.Sol**	**Centralization / Privilege**	**Major**	● **Mitigated**
MPH-04	Missing Error Message	Coding Style	Informational	● Resolved

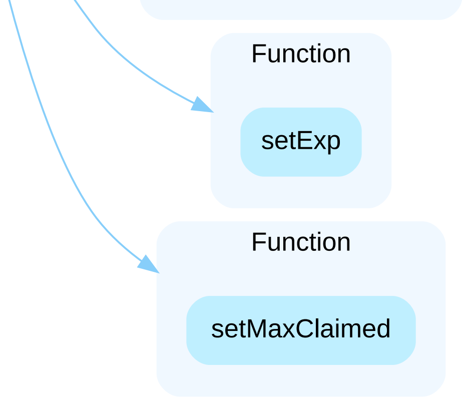
BOM-01 | CENTRALIZATION RISKS IN BASEOBJECT.SOL

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**contracts/utils/BaseObject.sol: 89, 94, 100, 105, 110, 115, 120, 126, 131, 136, 211, 216, 223**	● **Mitigated**

Description

In the contract `BaseObject` 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 collect royalties from tokens they did not create as well receive funds that should be directed to the treasury address.


Function
setSecondaryRoyalityFee
State Variables
secondaryRoyalty
Function
setShopAddress
State Variables
shopAddress
Function
setOpenForSale
Function
setSize
Function
setTreasuryAddress
State Variables
treasuryAddress
Function
setbaseMetadataURI
State Variables
baseMetadataURI
Authenticated Role
_owner
Function
setRoyalityFee
State Variables
royalityFee
Function
setCreatorAddress
Function
setTokenURI
Function
withdrawOwnerBalance
State Variables
paymentBalanceOwner
Function
changeTokenPrice



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

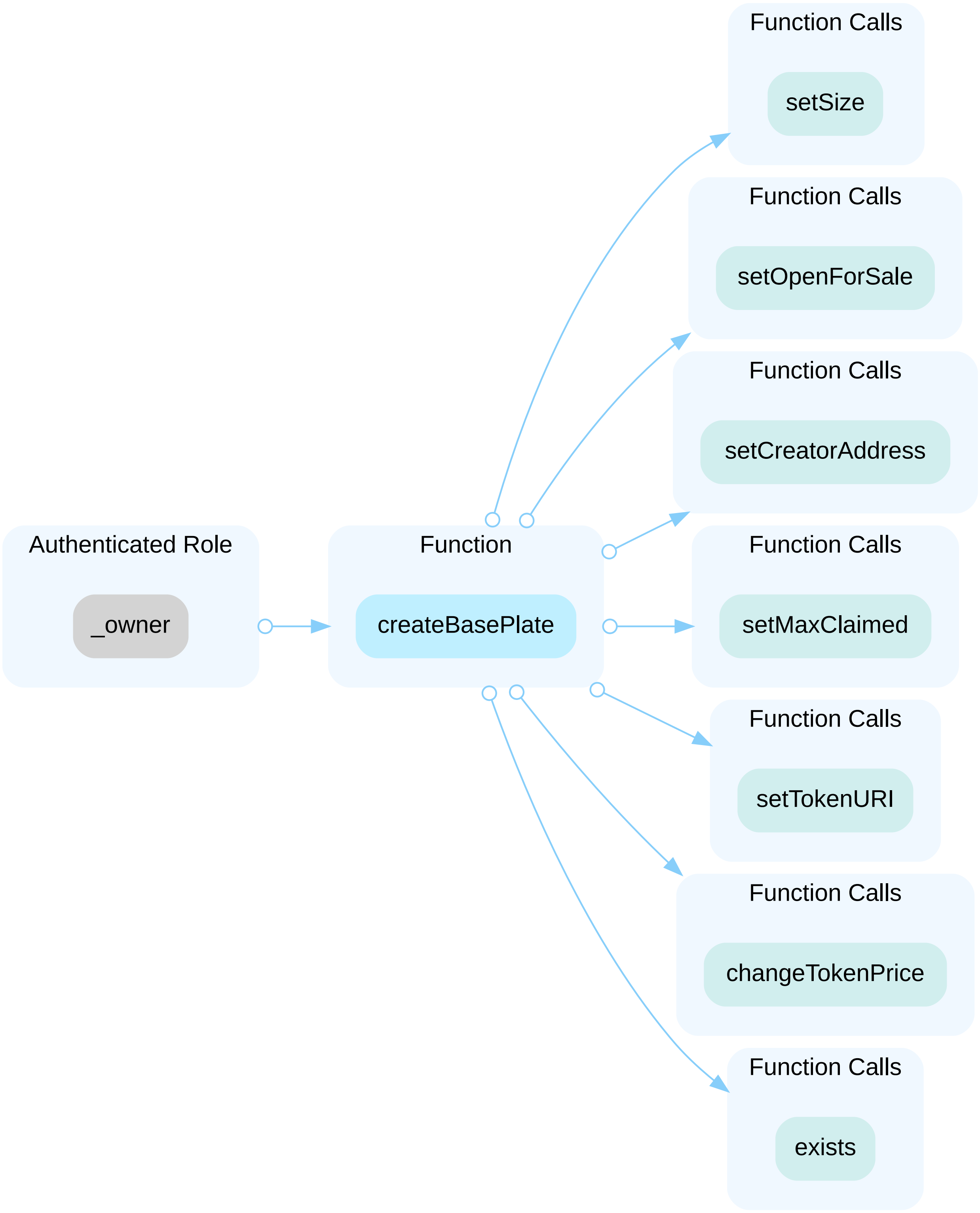
[Certik] - BaseObject is an abstract contract that BasePlate inherits. The centralization issue on BasePlate has been mitigated therefore the issues here are mitigated as well.

BPM-01 | CENTRALIZATION RISKS IN BASEPLATE.SOL

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**contracts/object/BasePlate.sol: 102, 225**	● **Mitigated**

Description

In the contract `BasePlate` 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 renounce ownership of the contract. This will prevent the creation of `BasePlates tokens` and hence keep anyone from purchasing a `BasePlate`.



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

`[CertiK]` : `BasePlate` is deployed at 0xE83d625E021f8238f418c068D256FeceDE705970

The timelock contract address is 0xf8a6d37ab13700c317d5011cc6bd88bdd726e13f

This transaction proves that the timelock is an owner of the contract: tx

This transaction proves that the previous owner `0xb7caa0ed757bbfaa208342752c9b1c541e36a4b9` has been revoked of the owner role by the timelock, tx

The multisig address is deployed at this address 0x913A8aa07B5Ed9e5ba73ee193696A27638ECD742 and it has the executor role in the timelock contract.

The multsig uses 4 signers that we list below :

Owner 1 : 0xFe3DdB7883c3f09e1fdc9908B570C6C79fB25f7C

Owner 2 : 0xE95330D7CDcd37bf0Ad875C29e2a2871FeFa3De8

Owner 3 : 0xA756641a137eb3F85C75Cb20D3fc4F69D4d3FD2f

Owner 4 : 0x1f9eFa166EE8Da21332DC5A55F7e267456cfd099

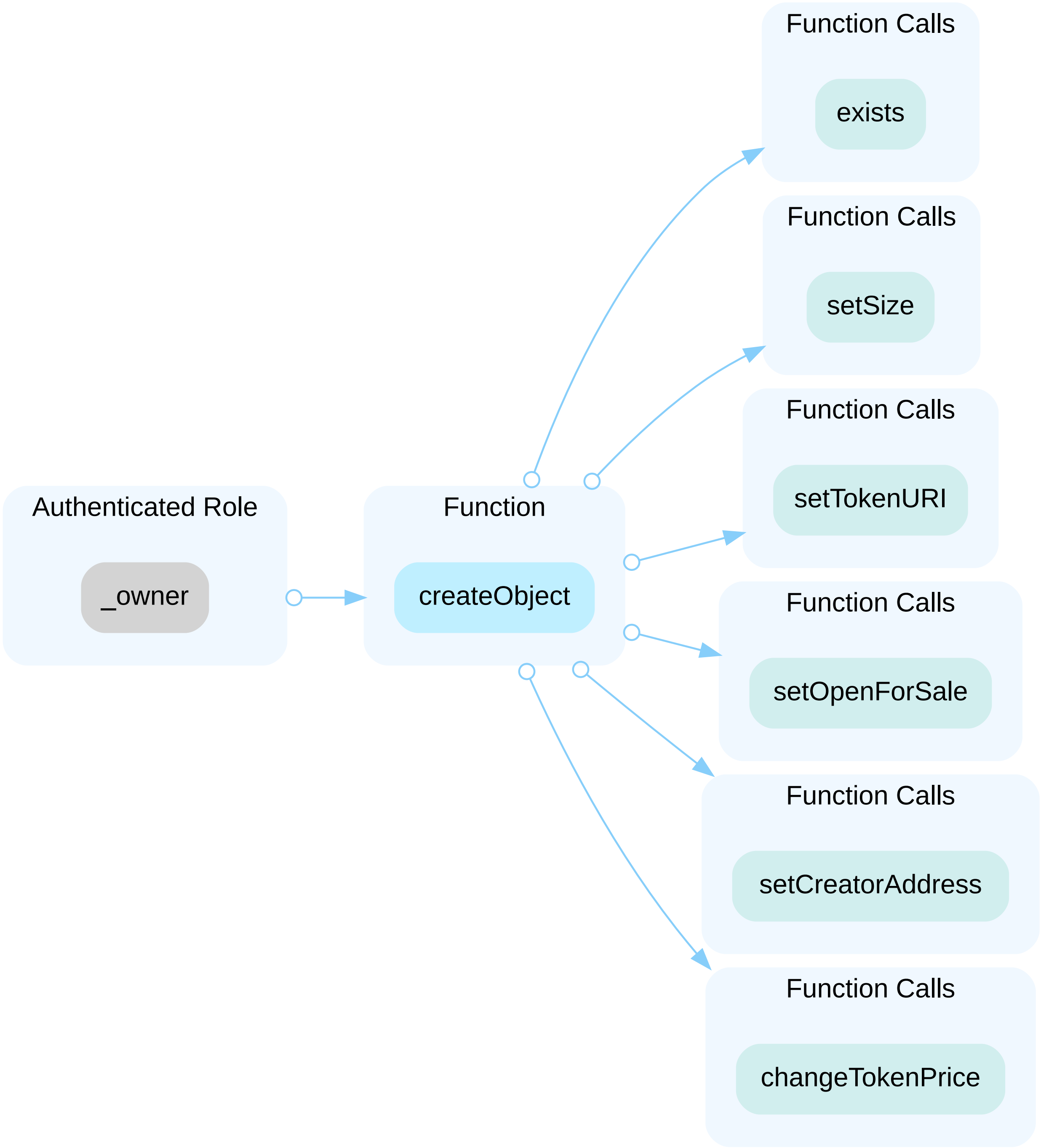
The information above is posted in this link: https://docs.philand.xyz/user-guide/contract-info.

FOM-01 | CENTRALIZATION RISKS IN FREEOBJECT.SOL

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**contracts/object/FreeObject.sol: 88, 158**	● **Mitigated**

Description

In the contract `FreeObject` 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 renounce ownership which prevents the creation of free objects to be distributed to buyers.



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

`[Certik]` : `FreeObject` is deployed at 0x017BD973FE4D3E0F81bAB69BcccCb44679D86eab

The timelock contract address is 0xf8a6d37ab13700c317d5011cc6bd88bdd726e13f

This transaction proves that the timelock is an owner of the contract: tx

This transaction proves that the previous owner `0xb7caa0ed757bbfaa208342752c9b1c541e36a4b9` has been revoked of the owner role by the timelock, tx

The multisig address is deployed at this address 0x913A8aa07B5Ed9e5ba73ee193696A27638ECD742 and it has the executor role in the timelock contract.

The multsig uses 4 signers that we list below :

Owner 1 : 0xFe3DdB7883c3f09e1fdc9908B570C6C79fB25f7C

Owner 2 : 0xE95330D7CDcd37bf0Ad875C29e2a2871FeFa3De8

Owner 3 : 0xA756641a137eb3F85C75Cb20D3fc4F69D4d3FD2f

Owner 4 : 0x1f9eFa166EE8Da21332DC5A55F7e267456cfd099

The information above is posted in this link: https://docs.philand.xyz/user-guide/contract-info.

MPH-01 | CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**contracts/PhiClaim.sol: 18; contracts/PhiMap.sol: 20; contracts/Registry.sol: 18**	● **Mitigated**

Description

`PhiClaim` , `PhiMap` , and `Registry` are upgradeable contracts, the owner can upgrade the contracts without the community's consent. If an attacker compromises the account, they can change the implementation of the contract and drain 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 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

All centralized related risks in PhiClaim, PhiMap, and Registry are mitigated due to the implementation of a time-lock and use of a multi-signature.

The following link includes information regarding the contracts deployed addresses, the time-lock and multi-signature address and all of signers.

Click Here

MPH-02 | POTENTIAL REENTRANCY ATTACK (OUT-OF-ORDER EVENTS)

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/PhiClaim.sol: 167, 168; contracts/PhiMap.sol: 354, 361, 362, 382, 389, 390, 589, 592, 594, 827, 828, 899, 900; contracts/Registry.sol: 158, 159, 174, 175	● Resolved

Description

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

This finding is considered minor because the reentrancy only causes out-of-order events.

External call(s)

```
167            _questObject.getObject(msg.sender, tokenId);
```

Events emitted after the call(s)

```
168            emit LogClaimObject(msg.sender, tokenId);
```

External call(s)

```
354                _lastWallPaper.safeTransferFrom(address(this), msg.sender,
lastWallPaperTokenId, 1, "0x00");
```

```
361            _object.safeTransferFrom(msg.sender, address(this), tokenId, 1,
"0x00");
```

Events emitted after the call(s)

```
362            emit ChangeWallPaper(name, contractAddress, tokenId);
```

External call(s)

```
                _lastBasePlate.safeTransferFrom(address(this), msg.sender,
lastBasePlateTokenId, 1, "0x00");
```

```
            _object.safeTransferFrom(msg.sender, address(this), tokenId, 1,
"0x00");
```

Events emitted after the call(s)

```
            emit ChangeBasePlate(name, contractAddress, tokenId);
```

External call(s)

```
                _changeWallPaper(name, wcontractAddress, wtokenId);
```

- This function call executes the following external call(s).
- In `PhiMap._changeWallPaper` ,
    - `_lastWallPaper.safeTransferFrom(address(this),msg.sender,lastWallPaperTokenId,1,0x00)`
- In `PhiMap._changeWallPaper` ,
    - `_object.safeTransferFrom(msg.sender,address(this),tokenId,1,0x00)`

```
                _changeBasePlate(name, bcontractAddress, btokenId);
```

- This function call executes the following external call(s).
- In `PhiMap._changeBasePlate` ,
    - `_lastBasePlate.safeTransferFrom(address(this),msg.sender,lastBasePlateTokenId,1,0x00)`
- In `PhiMap._changeBasePlate` ,
    - `_object.safeTransferFrom(msg.sender,address(this),tokenId,1,0x00)`

Events emitted after the call(s)

```
            emit Save(name, msg.sender);
```

```
            emit ChangeBasePlate(name, contractAddress, tokenId);
```

- Executed via the following function call(s):
    - `_changeBasePlate(name,bcontractAddress,btokenId)`

External call(s)

```
_object.safeTransferFrom(msgSender, address(this), tokenId, amount, "0x00");
```

Events emitted after the call(s)

```
emit DepositSuccess(msgSender, name, contractAddress, tokenId, amount);
```

External call(s)

```
_object.safeTransferFrom(address(this), msg.sender, tokenId, amount, "0x00");
```

Events emitted after the call(s)

```
emit WithdrawSuccess(msg.sender, name, contractAddress, tokenId, amount);
```

External call(s)

```
_phimap.create(name, msg.sender);
```

Events emitted after the call(s)

```
emit LogCreatePhiland(msg.sender, name);
```

External call(s)

```
_phimap.changePhilandOwner(name, msg.sender);
```

Events emitted after the call(s)

```
emit LogChangePhilandOwner(msg.sender, name);
```

Recommendation

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

Alleviation

`[CertiK]` : The `Phi` team resolved this issue by including the `nonReentrant` modifier to the function `save()` which mitigates re-entrancy risks. The inclusion of the modifier can be seen on line 576 in `PhiMap.sol` at commit 06a7dd9fcef8a766bf3e219dba6cb831c14e8f70.

MPH-03 | MISSING OVERRIDE SPECIFIER

Category	Severity	Location	Status
Compiler Error	● Minor	contracts/PhiMap.sol: 924~925, 934~935; contracts/utils/BaseObject.sol: 188	● Resolved

Description

The function `royaltyInfo()` defined on line 188 does not override `royaltyInfo()` inherited from `IERC2981.sol`. This will cause failure when compiling the contract.

Recommendation

We recommend including the override keyword to the function definition. The following change suffices:

```
function royaltyInfo(uint256, uint256 salePrice) external override view returns
(address receiver, uint256 royaltyAmount) {
        return (address(this), (salePrice * secondaryRoyalty) / 10000);
    }
```

Alleviation

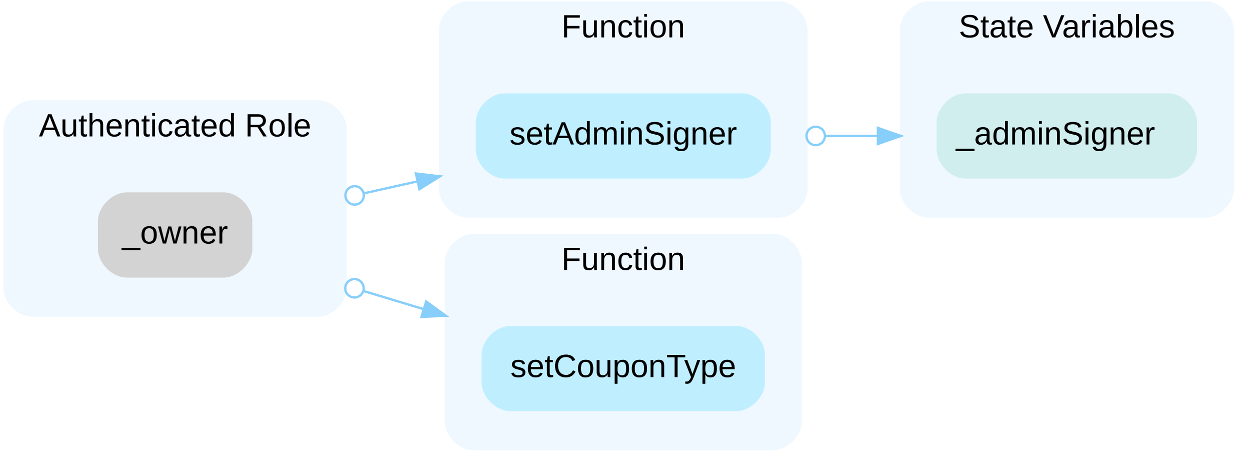
`[CertiK]`: The `PHI` team included the missing override keyword to `royaltyInfo()`, `onERC1155Received()` and `onERC1155BatchReceived()`. These changes resolve this issue and can be seen at commit 93cb93dfc7fcda682f800dd37ed3d353d6d8f1ad.

PCM-01 | CENTRALIZATION RISKS IN PHICLAIM.SOL

Category	Severity	Location	Status
Centralization / Privilege	● Major	contracts/PhiClaim.sol: 113, 125	● Mitigated

Description

In the contract `PhiClaim` 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 change the admin signer so that coupons won't verify. This causes any call to `claimQuestObject()` to fail.



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

`[CertiK]` : `PhiClaim` is deployed at 0x754e78bC0f7B487D304552810A5254497084970C

The timelock contract address is 0xf8a6d37ab13700c317d5011cc6bd88bdd726e13f

The following transaction proves that the previous owner `0xb7caa0ed757bbfaa208342752c9b1c541e36a4b9` has been revoked of the owner role by the timelock and that the timelock is the owner of the contract : tx

The multisig address is deployed at this address 0x913A8aa07B5Ed9e5ba73ee193696A27638ECD742 and it has the executor role in the timelock contract.

The multsig uses 4 signers that we list below :

Owner 1 : 0xFe3DdB7883c3f09e1fdc9908B570C6C79fB25f7C

Owner 2 : 0xE95330D7CDcd37bf0Ad875C29e2a2871FeFa3De8

Owner 3 : 0xA756641a137eb3F85C75Cb20D3fc4F69D4d3FD2f

Owner 4 : 0x1f9eFa166EE8Da21332DC5A55F7e267456cfd099

The information above is posted in this link: https://docs.philand.xyz/user-guide/contract-info.

PCM-02 | OLD COUPONS WILL FAIL VERIFICATION

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/PhiClaim.sol: 113, 164	● Resolved

Description

A call to `claimQuestObject()` will only pass if the coupon passed is signed by the address stored in `_adminSigner` . Therefore coupons signed by previous addresses stored in `_adminSigner` will not pass the check `isVerifiedCoupon()` .

Recommendation

We recommend ensuring that all coupons are verified before invoking a call to `setAdminSigner` .

Alleviation

`[CertiK]` : The team acknowledges the finding and ensures that this finding is not issue for their system. Please see below.

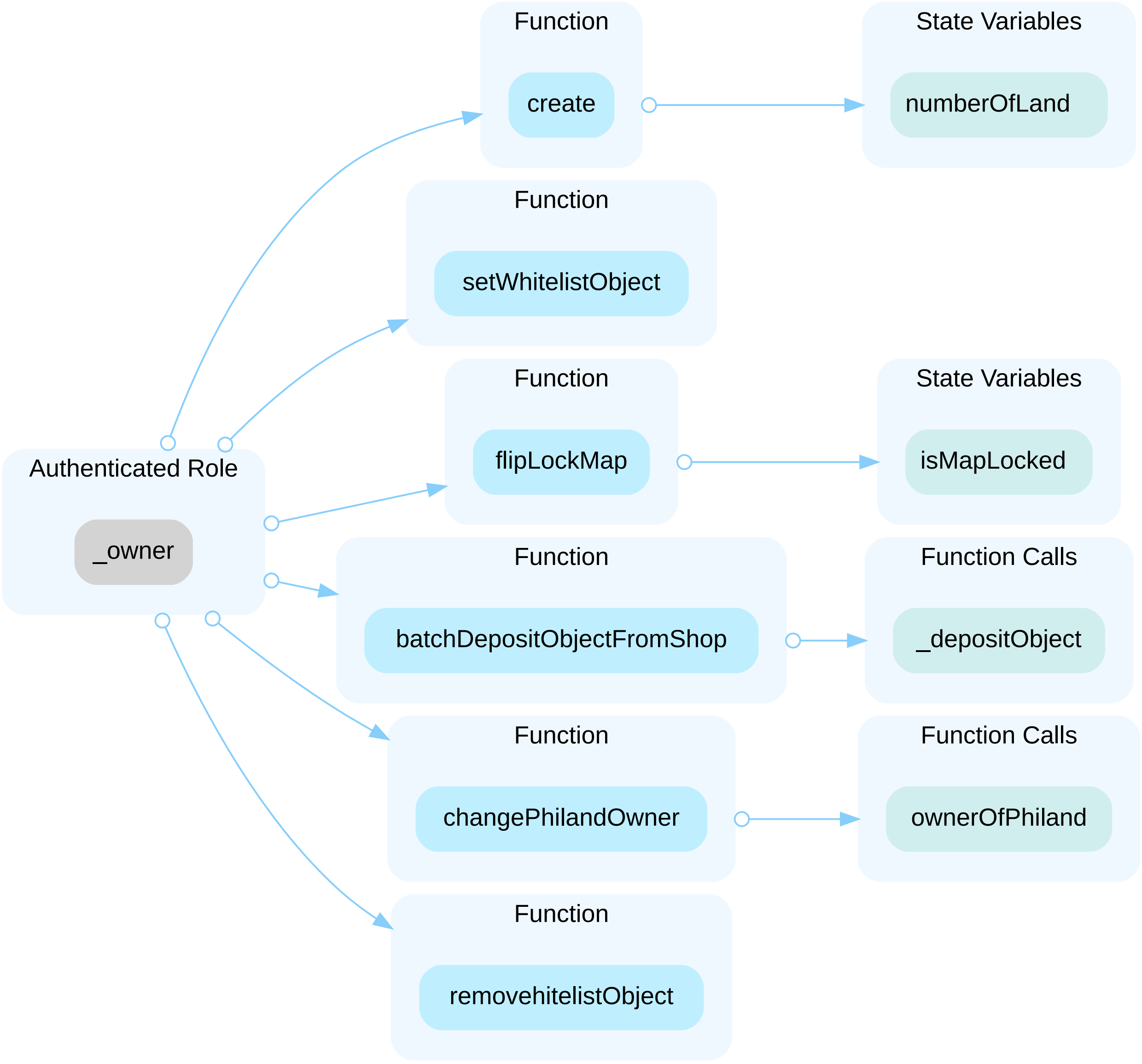
`[Philand]` : "If it doesn't go through, user can call to our backend to get the coupon(verified by newSigner key) again."

PMM-01 | CENTRALIZATION RISKS IN PHIMAP.SOL

Category	Severity	Location	Status
Centralization / Privilege	● Major	contracts/PhiMap.sol: 246, 261, 272, 280, 288, 860	● Mitigated

Description

In the contract `PhiMap` 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, change the owner of any land and write objects to land by calling the function `writeObjectToLand()` .



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

`[CertiK]` : `PhiMap` is deployed at 0xe8b6395d223C9D3D85e162f2cb2023bC9088a908

The timelock contract address is 0xf8a6d37ab13700c317d5011cc6bd88bdd726e13f

This transaction proves that the previous owner `0xb7caa0ed757bbfaa208342752c9b1c541e36a4b9` has been revoked of the owner role by the timelock and that the timelock is the owner of the contract tx

The multisig address is deployed at this address 0x913A8aa07B5Ed9e5ba73ee193696A27638ECD742 and it has the executor role in the timelock contract.

The multsig uses 4 signers that we list below :

Owner 1 : 0xFe3DdB7883c3f09e1fdc9908B570C6C79fB25f7C

Owner 2 : 0xE95330D7CDcd37bf0Ad875C29e2a2871FeFa3De8

Owner 3 : 0xA756641a137eb3F85C75Cb20D3fc4F69D4d3FD2f

Owner 4 : 0x1f9eFa166EE8Da21332DC5A55F7e267456cfd099

The information above is posted in this link: https://docs.philand.xyz/user-guide/contract-info.

PMM-02 | POTENTIAL REENTRANCY ATTACK

Category	Severity	Location	Status
Volatile Code	● Medium	contracts/PhiMap.sol: 354, 358, 382, 386	● Resolved

Description

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

External call(s)

```
354                 _lastWallPaper.safeTransferFrom(address(this), msg.sender,
lastWallPaperTokenId, 1, "0x00");
```

State variables written after the call(s)

```
358             wallPaper[name] = WallPaper(contractAddress, tokenId);
```

External call(s)

```
382                 _lastBasePlate.safeTransferFrom(address(this), msg.sender,
lastBasePlateTokenId, 1, "0x00");
```

State variables written after the call(s)

```
386             basePlate[name] = BasePlate(contractAddress, tokenId);
```

Recommendation

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

Alleviation

`[CertiK]` : The `Phi` team resolved this issue by including the `nonReentrant` modifier to the function `save()` which mitigates re-entrancy risks. The inclusion of the modifier can be seen on line 576 in `PhiMap.sol` at commit

06a7dd9fcef8a766bf3e219dba6cb831c14e8f70.

PMM-03 | LACK OF ACCESS RESTRICTION ON FUNCTION "_CHANGEBASEPLATE()`

Category	Severity	Location	Status
Logical Issue	● Medium	contracts/PhiMap.sol: 372, 376	● Resolved

Description

The function `_changeBasePlate()` is set as a public function meaning anyone can call this function without verification they are the original owner of the token they receive from the transaction. The concern is that `msg.sender` would be transferred the old baseplate token without any confirmation that it should be transferred to this address. It appears this function should only be called by other functions with the `onlyPhilandOwner()` modifier.

Recommendation

We recommend reviewing the function and setting the visibility to internal.

Alleviation

`[CertiK]` : On line 376 in `PhiMap.sol` at commit 1a67d3333776005d6ac77496d8f9dc8da35cf192 the function `_changeBasePlate` visibility is set to internal. This change resolves the issue.

PMM-04 | CONTRACT `ReentrancyGuardUpgradeable.sol` IS NOT USED

Category	Severity	Location	Status
Inconsistency	● Minor	contracts/PhiMap.sol: 16	● Resolved

Description

The contract `ReentrancyGuardUpgradeable.sol` contains a modifier `nonReentrant` which prevents functions from being re-entered. The modifier is not used in `PhiMap.sol` .

Recommendation

We recommend reviewing the codebase and ensuring the modifier is not needed. If the modifier is not needed, we recommend removing the inheritance.

Alleviation

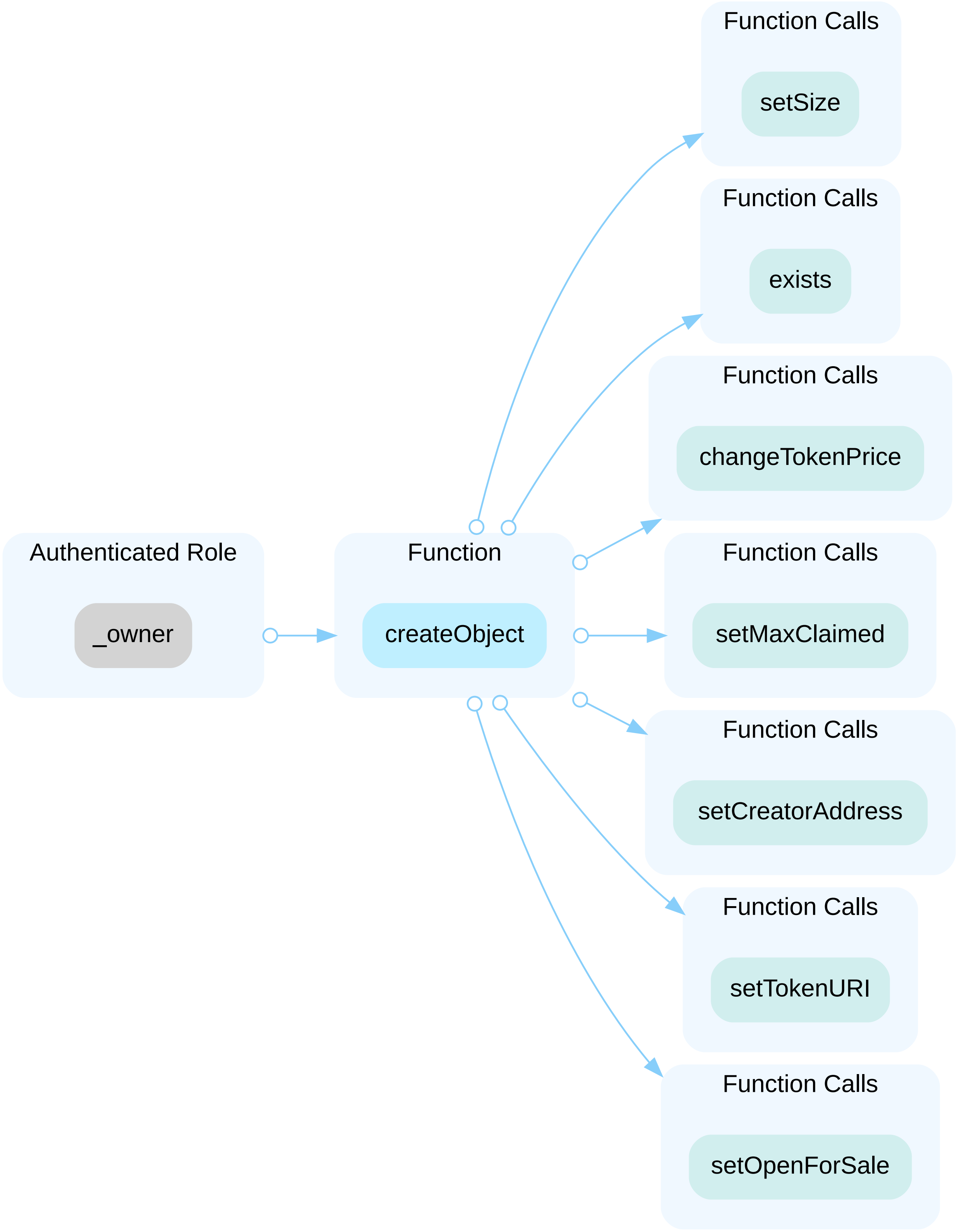
`[CertiK]` : At commit 06a7dd9fcef8a766bf3e219dba6cb831c14e8f70 the `nonReentrant` modifier is included. Thus the contract `ReentrancyGuardUpgradeable.sol` is now used. The changes implemented resolve this issue.

POM-01 | CENTRALIZATION RISKS IN PREMIUMOBJECT.SOL

Category	Severity	Location	Status
Centralization / Privilege	● Major	contracts/object/PremiumObject.sol: 102, 226	● Mitigated

Description

In the contract `PremiumObject` 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 renounce ownership of the contract. This will prevent the creation of `Premium tokens` and hence keep anyone from purchasing a `Premium object` .



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

`[Certik]` : `PremiumObject` is deployed at 0x312E2F8AD479BEDd32c7A752bCB68FA24550CAF7

The timelock contract address is 0xf8a6d37ab13700c317d5011cc6bd88bdd726e13f

This transaction proves that the timelock is an owner of the contract: tx

This transaction proves that the previous owner `0xb7caa0ed757bbfaa208342752c9b1c541e36a4b9` has been revoked of the owner role by the timelock, tx

The multisig address is deployed at this address 0x913A8aa07B5Ed9e5ba73ee193696A27638ECD742 and it has the executor role in the timelock contract.

The multsig uses 4 signers that we list below :

Owner 1 : 0xFe3DdB7883c3f09e1fdc9908B570C6C79fB25f7C

Owner 2 : 0xE95330D7CDcd37bf0Ad875C29e2a2871FeFa3De8

Owner 3 : 0xA756641a137eb3F85C75Cb20D3fc4F69D4d3FD2f

Owner 4 : 0x1f9eFa166EE8Da21332DC5A55F7e267456cfd099

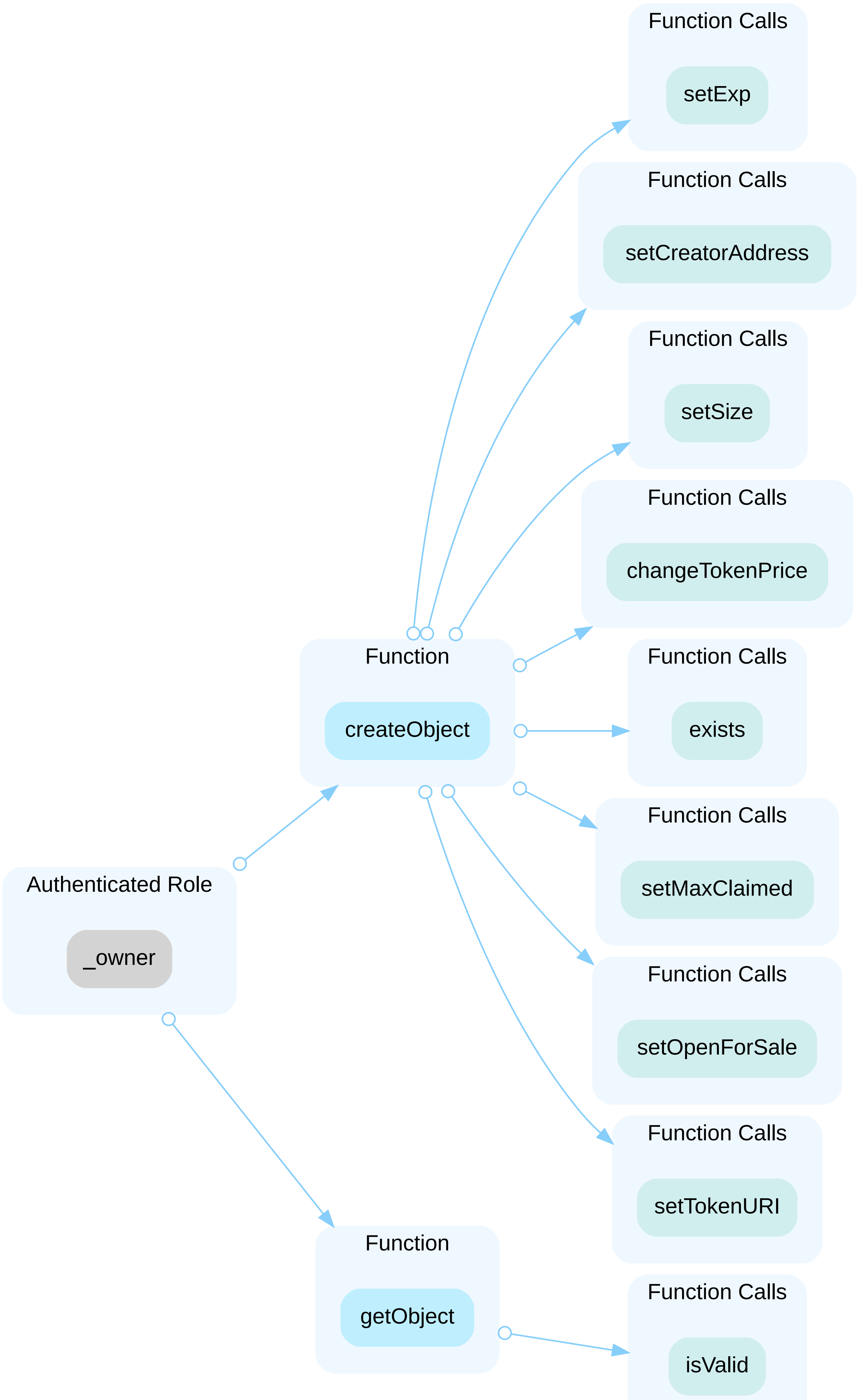
The information above is posted in this link: https://docs.philand.xyz/user-guide/contract-info.

QOM-01 | CENTRALIZATION RISKS IN QUESTOBJECT.SOL

Category	Severity	Location	Status
Centralization / Privilege	● Major	contracts/object/QuestObject.sol: 100, 132	● Mitigated

Description

In the contract `QuestObject` 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 create an arbitrary amount of quest objects to be claimed via the function `createObject()` .


Function Calls
setExp
Function Calls
setCreatorAddress
Function Calls
setSize
Function Calls
changeTokenPrice
Function
createObject
Function Calls
exists
Function Calls
setMaxClaimed
Authenticated Role
_owner
Function Calls
setOpenForSale
Function Calls
setTokenURI
Function
getObject
Function Calls
isValid

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

`[CertiK]` : QuestObject is deployed at 0x3D8C06e65ebf06A9d40F313a35353be06BD46038

The timelock contract address is 0xf8a6d37ab13700c317d5011cc6bd88bdd726e13f

This transaction proves that the previous owner `0xb7caa0ed757bbfaa208342752c9b1c541e36a4b9` has been revoked of the owner role by the timelock and that the timelock is the owner of the contract tx

The multisig address is deployed at this address 0x913A8aa07B5Ed9e5ba73ee193696A27638ECD742 and it has the executor role in the timelock contract.

The multsig uses 4 signers that we list below :

Owner 1 : 0xFe3DdB7883c3f09e1fdc9908B570C6C79fB25f7C

Owner 2 : 0xE95330D7CDcd37bf0Ad875C29e2a2871FeFa3De8

Owner 3 : 0xA756641a137eb3F85C75Cb20D3fc4F69D4d3FD2f

Owner 4 : 0x1f9eFa166EE8Da21332DC5A55F7e267456cfd099

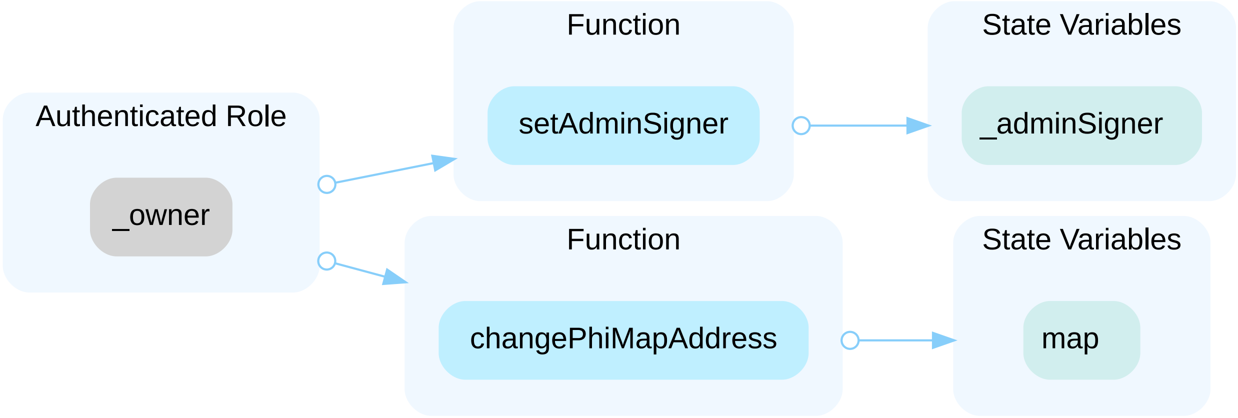
The information above is posted in this link: https://docs.philand.xyz/user-guide/contract-info.

RMP-01 | CENTRALIZATION RISKS IN REGISTRY.SOL

Category	Severity	Location	Status
Centralization / Privilege	● Major	contracts/Registry.sol: 123, 182	● Mitigated

Description

In the contract `Registry` 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 change the Phi Map address and the Phi land owner. The owner may also change the admin signer name which will prevent any coupon from verifying.



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

`[CertiK]` : `Registry` is deployed at 0x6532B97295a0728880AE81D5CD7248f32E24e39a

The timelock contract address is 0xf8a6d37ab13700c317d5011cc6bd88bdd726e13f

This transaction proves that the previous owner `0xb7caa0ed757bbfaa208342752c9b1c541e36a4b9` has been revoked of the owner role by the timelock and that the timelock is the owner of the contract tx

The multisig address is deployed at this address 0x913A8aa07B5Ed9e5ba73ee193696A27638ECD742 and it has the executor role in the timelock contract.

The multsig uses 4 signers that we list below :

Owner 1 : 0xFe3DdB7883c3f09e1fdc9908B570C6C79fB25f7C

Owner 2 : 0xE95330D7CDcd37bf0Ad875C29e2a2871FeFa3De8

Owner 3 : 0xA756641a137eb3F85C75Cb20D3fc4F69D4d3FD2f

Owner 4 : 0x1f9eFa166EE8Da21332DC5A55F7e267456cfd099

The information above is posted in this link: https://docs.philand.xyz/user-guide/contract-info.

MPH-04 | MISSING ERROR MESSAGE

Category	Severity	Location	Status
Coding Style	● Informational	contracts/object/BasePlate.sol: 135, 137, 141, 143, 145, 147, 157, 166, 190, 194, 196, 198, 200; contracts/object/FreeObject.sol: 117, 127, 142; contracts/object/PremiumObject.sol: 136, 138, 142, 144, 146, 148, 158, 167, 191, 195, 197, 199, 201; contracts/object/QuestObject.sol: 134, 136, 140; contracts/object/WallPaper.sol: 135, 137, 141, 143, 145, 147, 157, 166, 190, 194, 196, 198, 200	● Resolved

Description

The **require** command can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We recommend including an error message in the linked statements.

Alleviation

`[CertiK]` : The Philand team resolved the issue by following our recommendation above. The linked statements now include an error message. The changes can be seen at the commit ae809771b700a6c767911c726bccb210b57c7ffc.

OPTIMIZATIONS | PHI

ID	Title	Category	Severity	Status
MPH-05	User-Defined Getters	Gas Optimization	Optimization	● Partially Resolved

MPH-05 | USER-DEFINED GETTERS

Category	Severity	Location	Status
Gas Optimization	● Optimization	contracts/PhiMap.sol: 302~304, 313~315, 400~402, 410~412, 418~420; contracts/Registry.sol: 113~115; contracts/utils/BaseObject.sol: 142~144, 175~177, 179~181, 205~207	● Partially Resolved

Description

The linked functions are equivalent to the compiler-generated getter functions for the respective variables.

Recommendation

We recommend that the linked variables are instead declared as `public` as compiler-generated getter functions are less prone to error and much more maintainable than manually written ones.

Alleviation

[CertiK] - The Phi team removed some of the redundant functions. The changes can be seen at the following commit bbd83bf58fc24a437f59548fb993e35a65edb580.

APPENDIX | PHI

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.
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.
Compiler Error	Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

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.