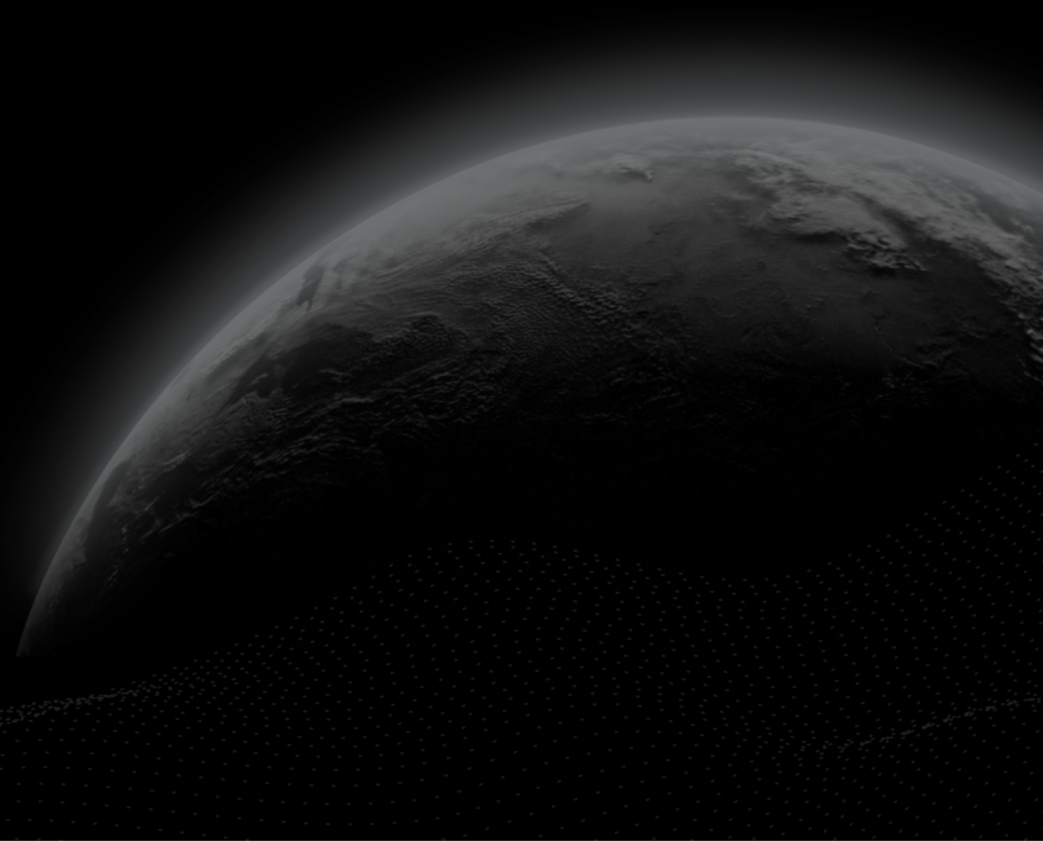
CERTIK

Security Assessment

Royal StableCoins - Audit

CertiK Assessed on Mar 27th, 2024



CertiK Assessed on Mar 27th, 2024

Royal StableCoins - Audit

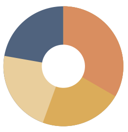
The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	EVM Compatible	Formal Verification, Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Solidity	Delivered on 03/27/2024	N/A

CODEBASE	COMMITS
4012eef2bb61b4448235eb22db7f89e5ad30aff4	4012eef2bb61b4448235eb22db7f89e5ad30aff4
View All in Codebase Page	View All in Codebase Page

Vulnerability Summary



9	2	0	0	7	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

🟥	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.
🟧	3	Major	3 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
🟨	2	Medium	1 Resolved, 1 Acknowledged	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	2	Minor	1 Resolved, 1 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
🟦	2	Informational	2 Acknowledged	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | ROYAL STABLECOINS - AUDIT

Summary

Executive Summary

Vulnerability Summary

Codebase

Audit Scope

Approach & Methods

Review Notes

Overview

External Dependencies

ReserveConsumerV3.sol

REUR.sol/ROYAL.sol/RXAU.sol

Privileged Functions

Findings

REU-01 : Centralization Risks in REUR.sol

ROY-01 : Centralization Risks in ROYAL.sol

RXA-01 : Centralization Risks in RXAU.sol

401-01 : Incompatibility with Deflationary Tokens

401-03 : Init Functions Are Susceptible To Front-running

401-06 : Third-Party Dependency Usage

RIB-01 : Missing Zero Address Validation

CRV-01 : Hardcode Address

REU-02 : Discussion On the `MAX_HEARTBEAT_DAYS`

Formal Verification

Considered Functions And Scope

Verification Results

Appendix

Disclaimer

CODEBASE | ROYAL STABLECOINS - AUDIT

Repository

4012eef2bb61b4448235eb22db7f89e5ad30aff4

Commit

4012eef2bb61b4448235eb22db7f89e5ad30aff4

AUDIT SCOPE | ROYAL STABLECOINS - AUDIT

4 files audited ● 4 files with Acknowledged findings

ID	Repo	File	SHA256 Checksum
● CRV	maalchain/ribg-stablecoin-contracts	ChainlinkReserveV3.sol	725696407c6719a7a46dd29c4eae440661a751ab9cf5540ede8d2954ea567fb2
● REU	maalchain/ribg-stablecoin-contracts	REUR.sol	bf334773dbe01967ae3956e5da01b7ad1e818227dca07dc08a4038c0a4dc7935
● ROY	maalchain/ribg-stablecoin-contracts	ROYAL.sol	144ab8cf4c6ec9932ab695d7a60f490eb219a6e05516fa12caec8507700513e6
● RXA	maalchain/ribg-stablecoin-contracts	RXAU.sol	520dfd9c0cad81fafc16eefe96ea30840b028760f344067f3700a7eb4adc0347

APPROACH & METHODS | ROYAL STABLECOINS - AUDIT

This report has been prepared for Royal StableCoins to discover issues and vulnerabilities in the source code of the Royal StableCoins - Audit project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Formal Verification, 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.

REVIEW NOTES | ROYAL STABLECOINS - AUDIT

Overview

The **Royal StableCoins** protocol implements three contracts that follow a standard ERC-20 interface for token functionality and include additional features such as vault, pausing/unpausing, blacklisting, and configuring minters. Additionally, it interacts with an external price feed to ensure that minting is performed based on certain conditions related to the availability of reserves.

External Dependencies

The following are external addresses used within the contracts:

ReserveConsumerV3.sol

- `reserveFeed` - the address of the Proof of Reserve Feed.

REUR.sol/ROYAL.sol/RXAU.sol

- `_owner` - the owner of the contract.
- `masterMinter` - who control the overall mint logic in the protocol.
- `pauser` - who has the right to pause/unpause the protocol.
- `_rescuer` - who can get out the extra tokens from the contract.

All the 3 tokens are interact with the `chainlink` Proof Of Reserve Feeds.

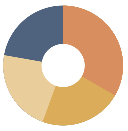
Privileged Functions

In the **Royal StableCoins** project, multiple roles are adopted to ensure the dynamic runtime updates of the project, which were specified in the centralization findings *REU-01*, *ROY-01*, *RXA-01*.

The advantage of this privileged role in the codebase is that the client reserves the ability to adjust the protocol according to the runtime required to best serve the community. It is also worth of note the potential drawbacks of these functions, which should be clearly stated through the client's action/plan. Additionally, if the private key of the privileged account is compromised, it could lead to devastating consequences for the project.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the `Timelock` contract.

FINDINGS | ROYAL STABLECOINS - AUDIT



9	0	3	2	2	2
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Royal StableCoins - Audit. Through this audit, we have uncovered 9 issues ranging from different severity levels. Utilizing the techniques of Formal Verification, Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

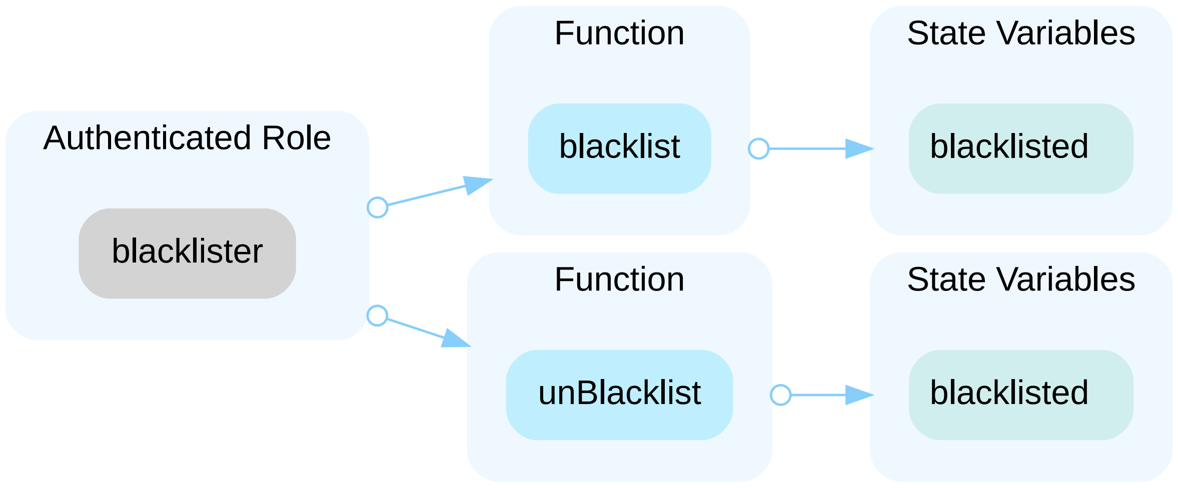
ID	Title	Category	Severity	Status
REU-01	**Centralization Risks In REUR.Sol**	**Centralization**	**Major**	● **Acknowledged**
ROY-01	**Centralization Risks In ROYAL.Sol**	**Centralization**	**Major**	● **Acknowledged**
RXA-01	**Centralization Risks In RXAU.Sol**	**Centralization**	**Major**	● **Acknowledged**
401-01	Incompatibility With Deflationary Tokens	Logical Issue	Medium	● Acknowledged
401-03	Init Functions Are Susceptible To Front-Running	Coding Issue	Medium	● Resolved
401-06	Third-Party Dependency Usage	Design Issue	Minor	● Acknowledged
RIB-01	Missing Zero Address Validation	Volatile Code	Minor	● Resolved
CRV-01	Hardcode Address	Coding Style	Informational	● Acknowledged
REU-02	Discussion On The `MAX_HEARTBEAT_DAYS`	Design Issue	Informational	● Acknowledged

REU-01 | CENTRALIZATION RISKS IN REUR.SOL

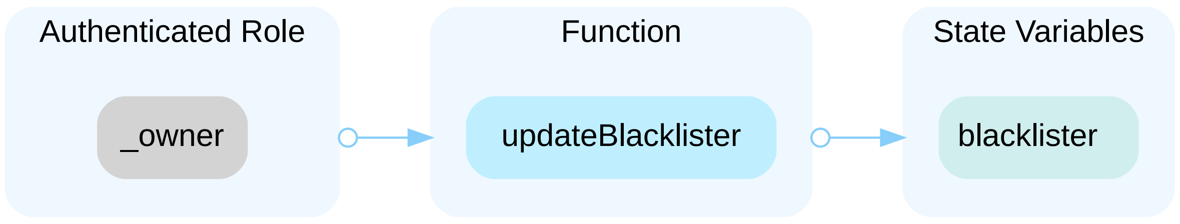
Category	Severity	Location	Status
Centralization	● **Major**	**REUR.sol (03/12): 290, 298, 393, 405, 410, 489, 498, 503, 813, 831, 981, 1030, 1223, 1240, 1257, 1274, 1291, 1296**	● **Acknowledged**

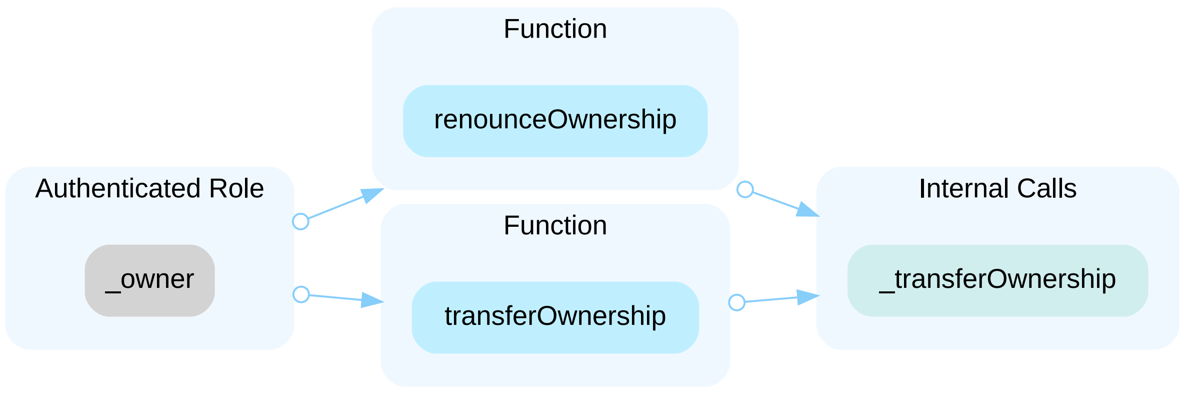
Description

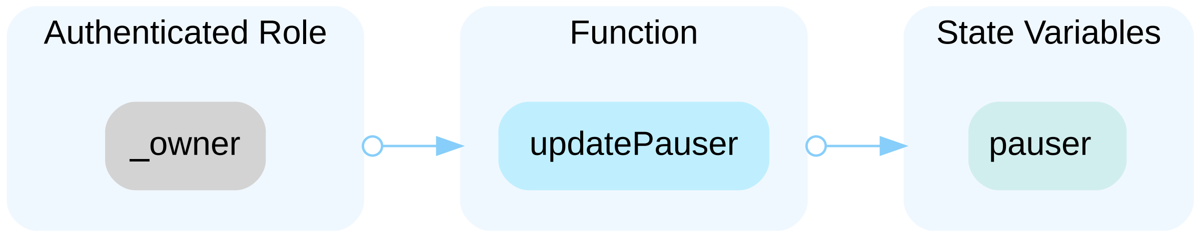
In the contract `Blacklistable` the role `blacklister` has authority over the functions shown in the diagram below. Any compromise to the `blacklister` account may allow the hacker to take advantage of this authority to add an address to the blacklist or remove from it.

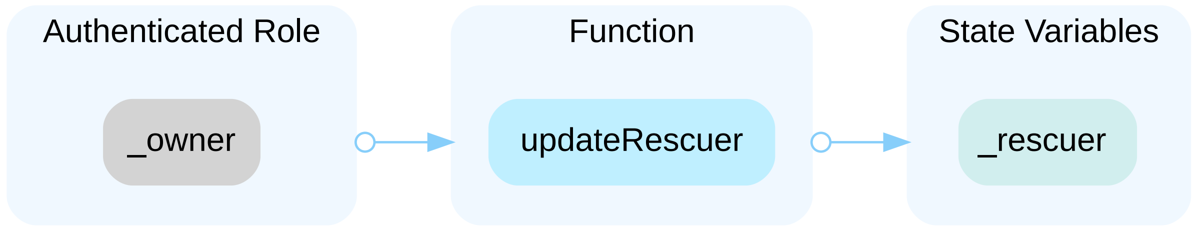


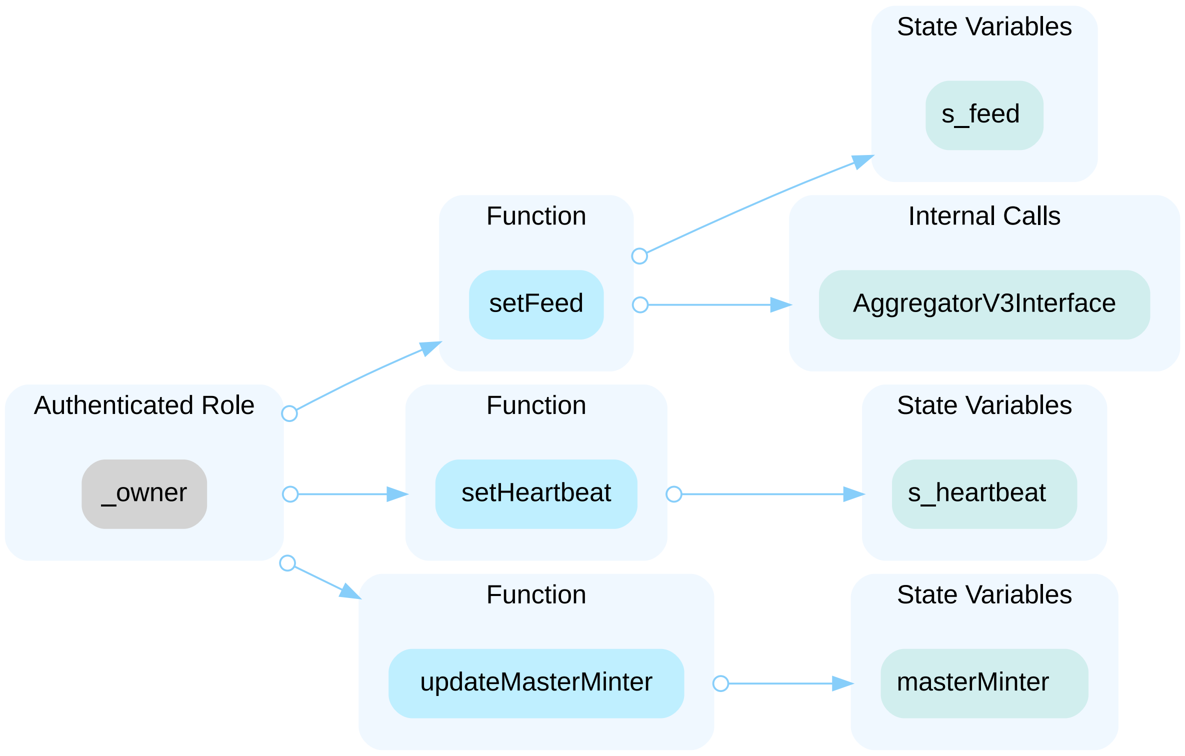
In the contract `RoyalEURO` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority to transfer ownership, change `blacklister` / `pauser` / `_rescuer` / `masterMinter` , set the heartbeat, set the address of the `s_feed` .



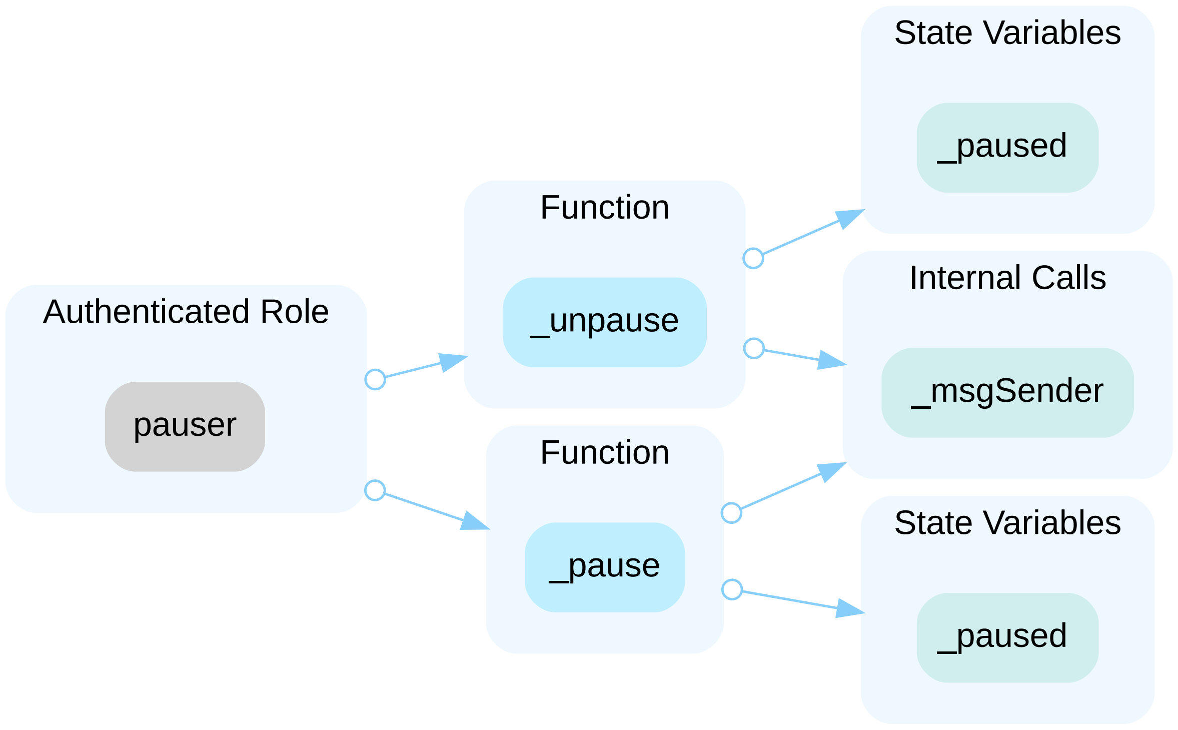




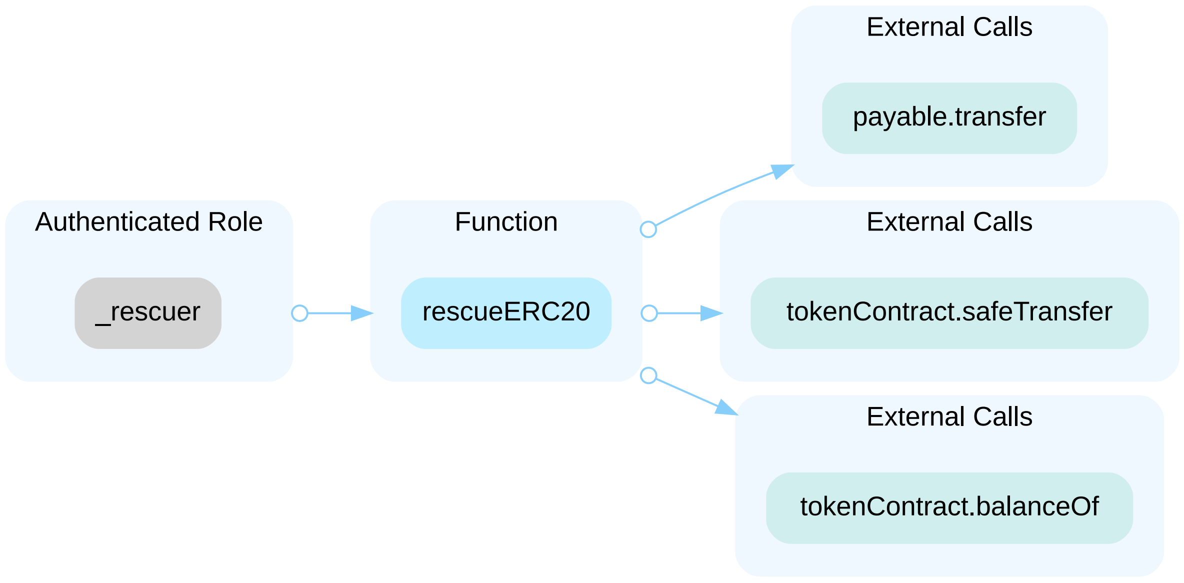




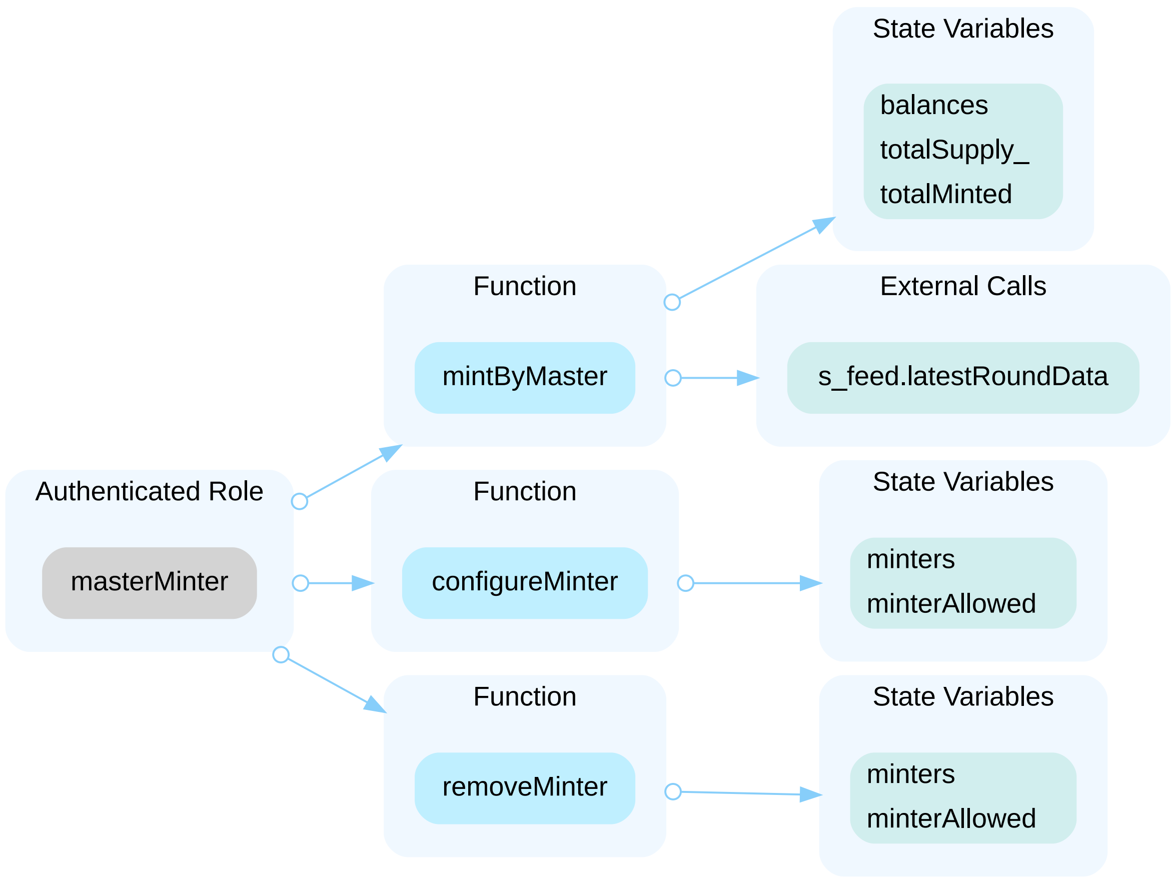
In the contract `Pausable` the role `pauser` has authority over the functions shown in the diagram below. Any compromise to the `pauser` account may allow the hacker to take advantage of this authority to `pause` / `unpause` the protocol.



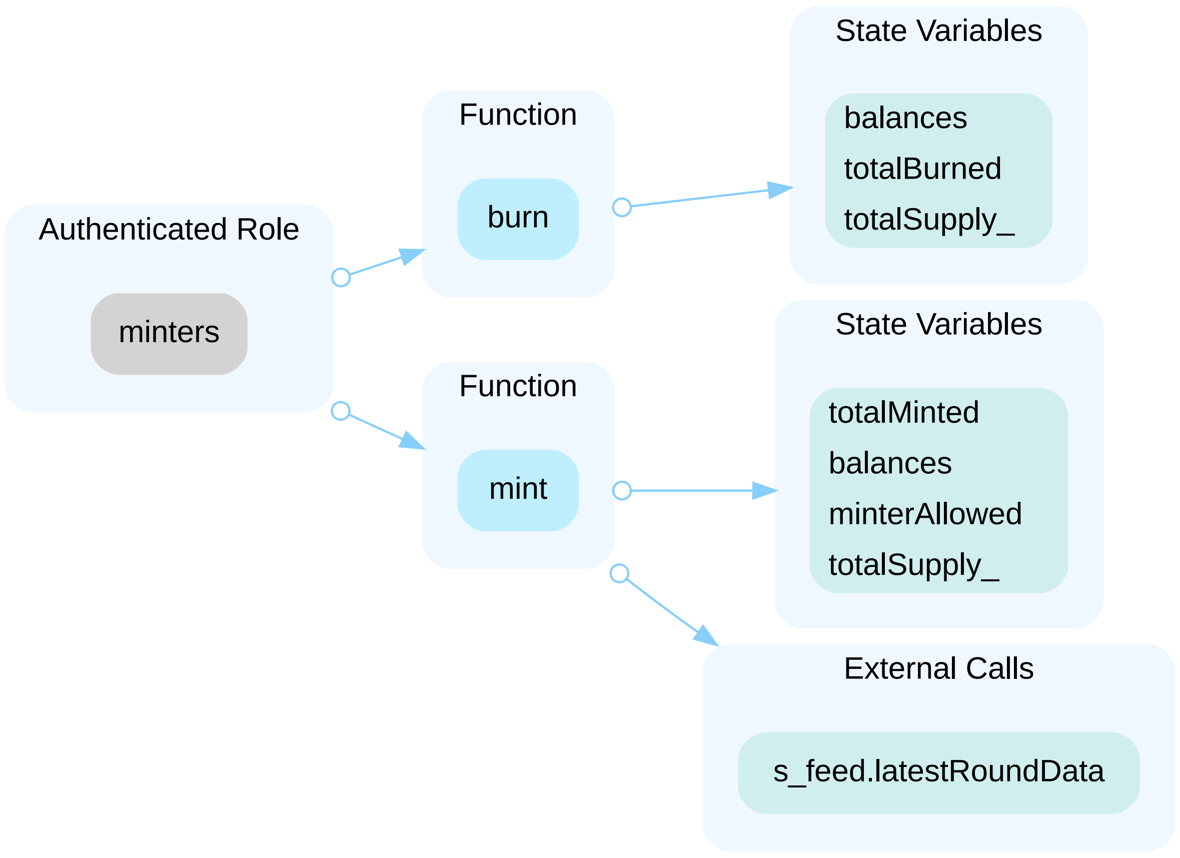
In the contract `Rescuable` the role `_rescuer` has authority over the functions shown in the diagram below. Any compromise to the `_rescuer` account may allow the hacker to take advantage of this authority to rescue extra tokens from the contract.



In the contract `RoyalEURO` the role `masterMinter` has authority over the functions shown in the diagram below. Any compromise to the `masterMinter` account may allow the hacker to take advantage of this authority to grant/revoke the role `minters` and mint tokens to an arbitrary address that is not on the blacklist.



In the contract `RoyalEURO` the role `minters` has authority over the functions shown in the diagram below. Any compromise to the `minters` account may allow the hacker to take advantage of this authority to mint tokens to an arbitrary address that is not on the blacklist.



In commit: 64e3df33d1f58561657bee0fdc85086e748378a8, the team introduced centralized control into the `initialize` function, enabling the `_owner` to execute the contract's `initialize` method.

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

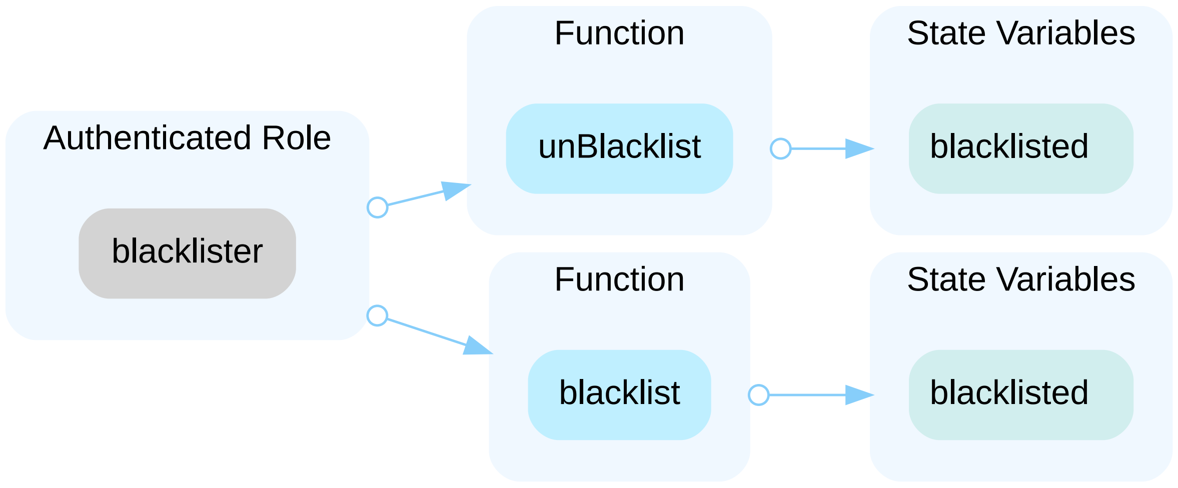
[Royal StableCoin 03/25/2024]: Issue acknowledged. We will implement multisig wallet on the mainnet.

ROY-01 | CENTRALIZATION RISKS IN ROYAL.SOL

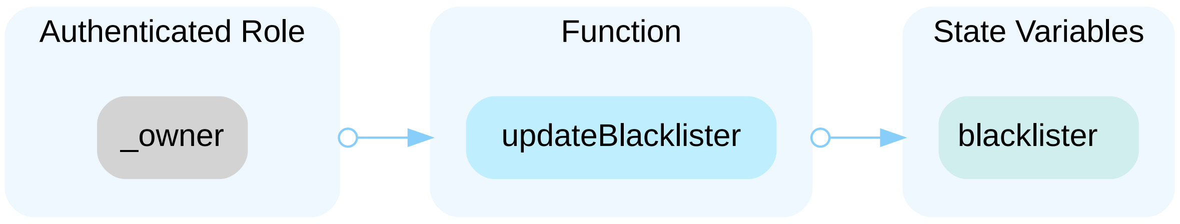
Category	Severity	Location	Status
Centralization	● Major	ROYAL.sol (03/12): 290, 298, 393, 405, 410, 489, 498, 503, 813, 831, 980, 1029, 1222, 1239, 1256, 1273, 1290, 1295	● Acknowledged

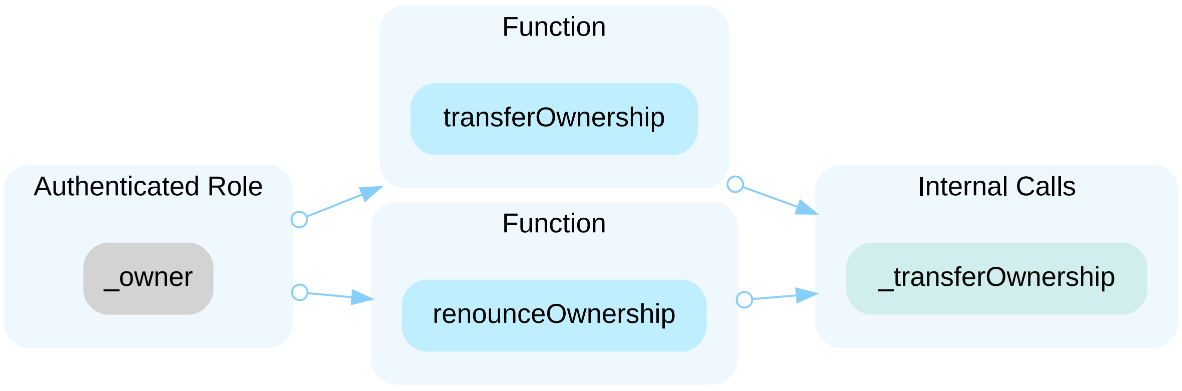
Description

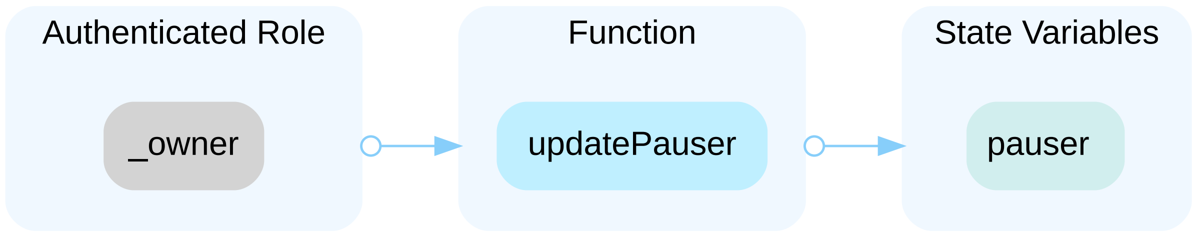
In the contract `Blacklistable` the role `blacklister` has authority over the functions shown in the diagram below. Any compromise to the `blacklister` account may allow the hacker to take advantage of this authority to add an address to the blacklist or remove from it.

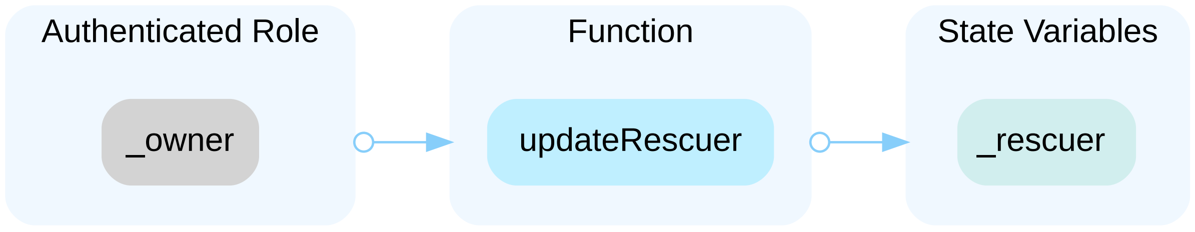


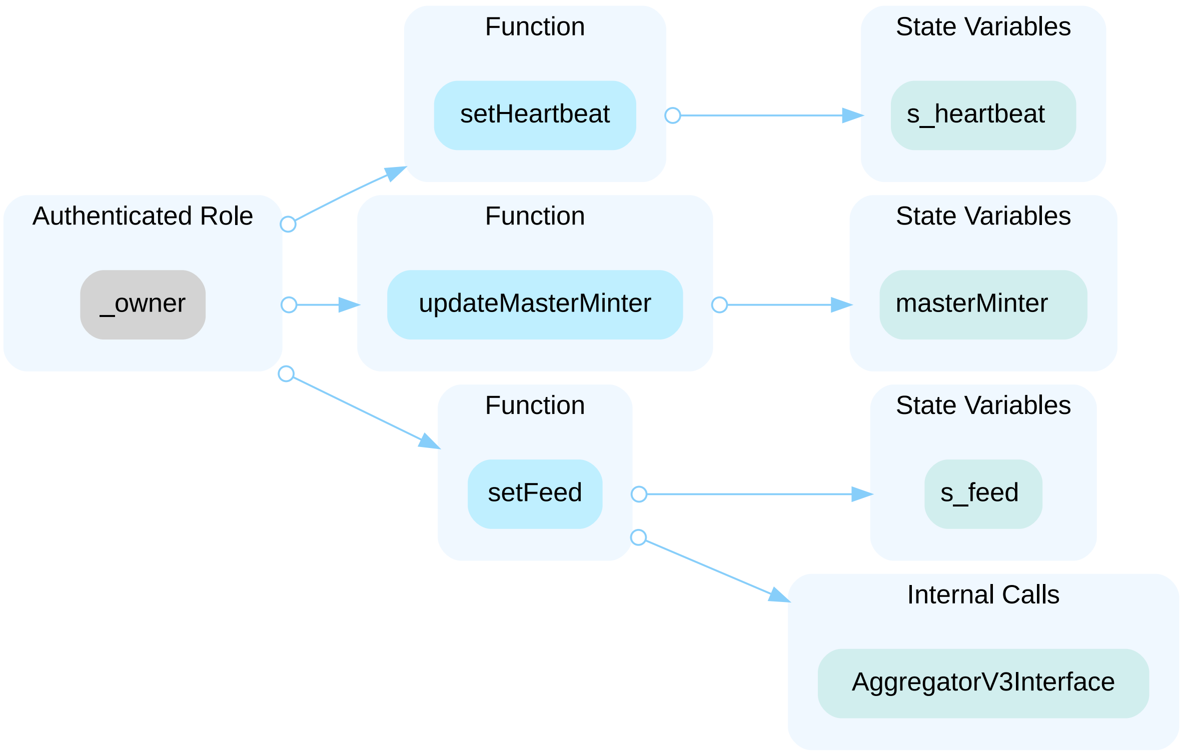
In the contract `RoyalEURO` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority to transfer ownership, change `blacklister` / `pauser` / `_rescuer` / `masterMinter` , set the heartbeat, set the address of the `s_feed` .



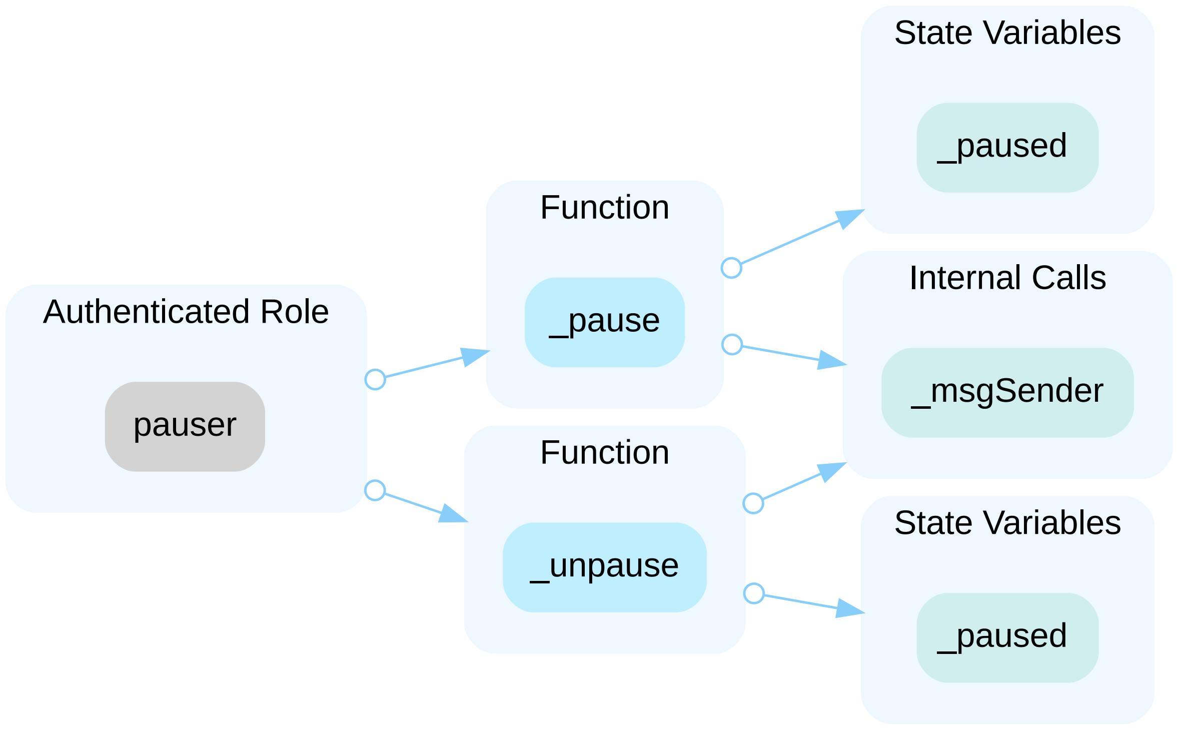




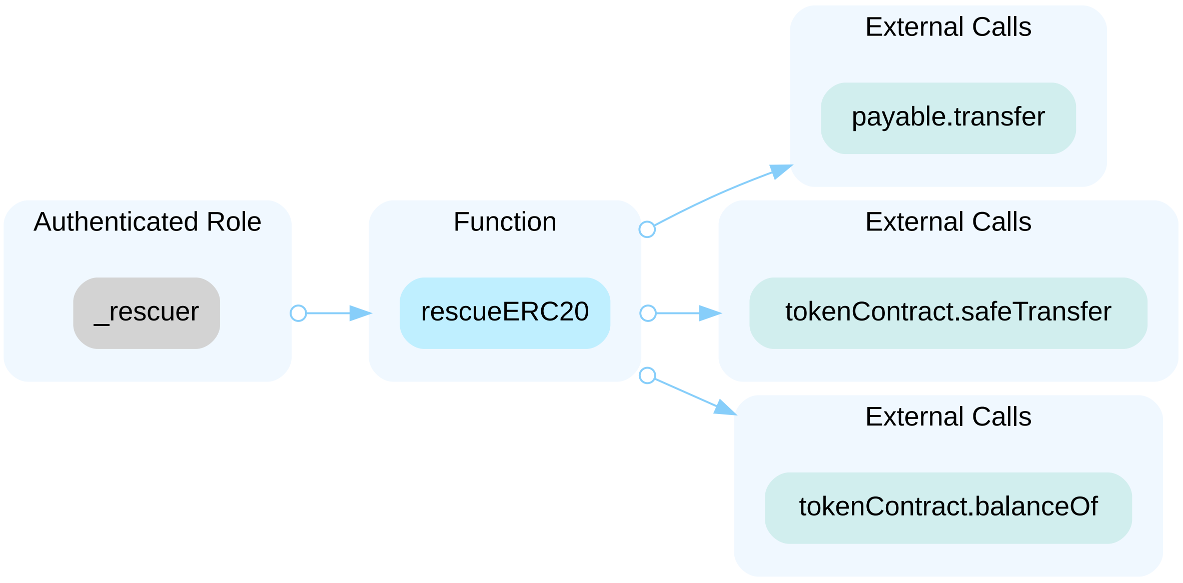




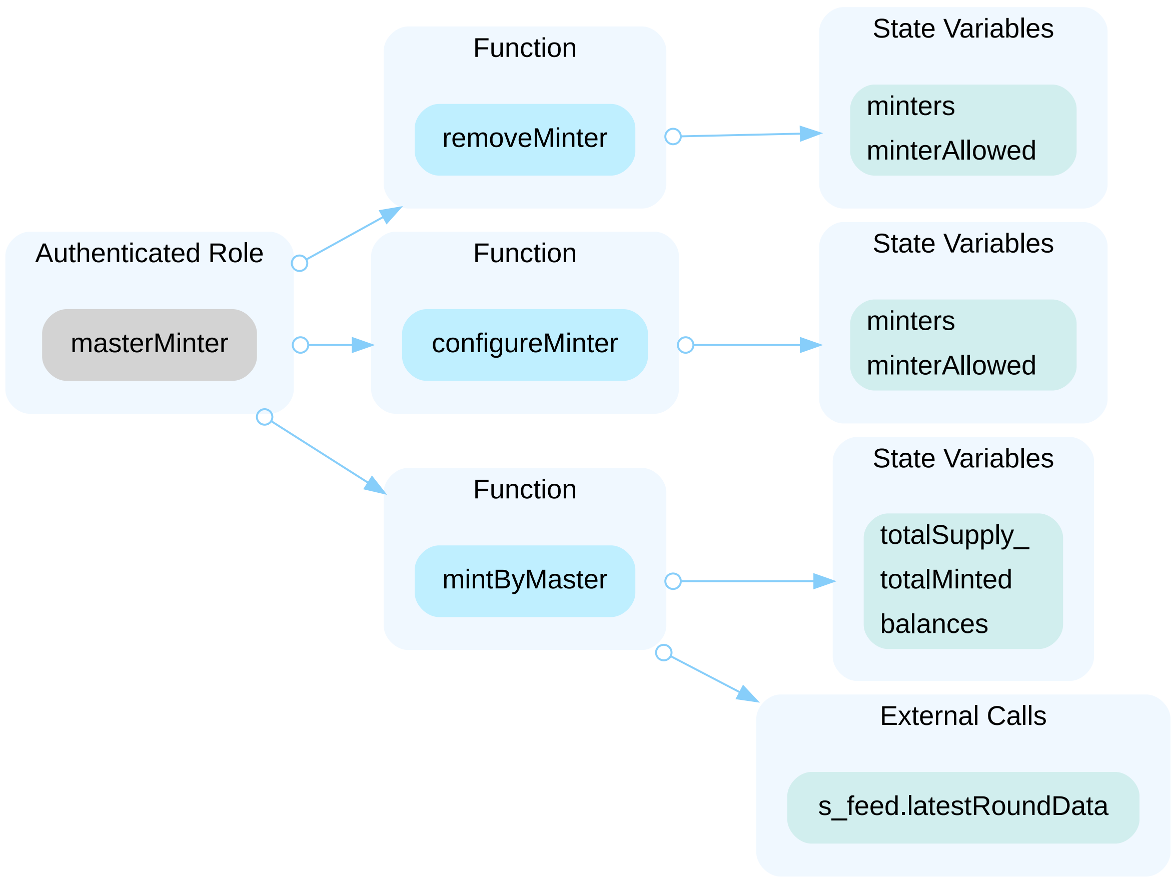
In the contract `Pausable` the role `pauser` has authority over the functions shown in the diagram below. Any compromise to the `pauser` account may allow the hacker to take advantage of this authority to `pause` / `unpause` the protocol.



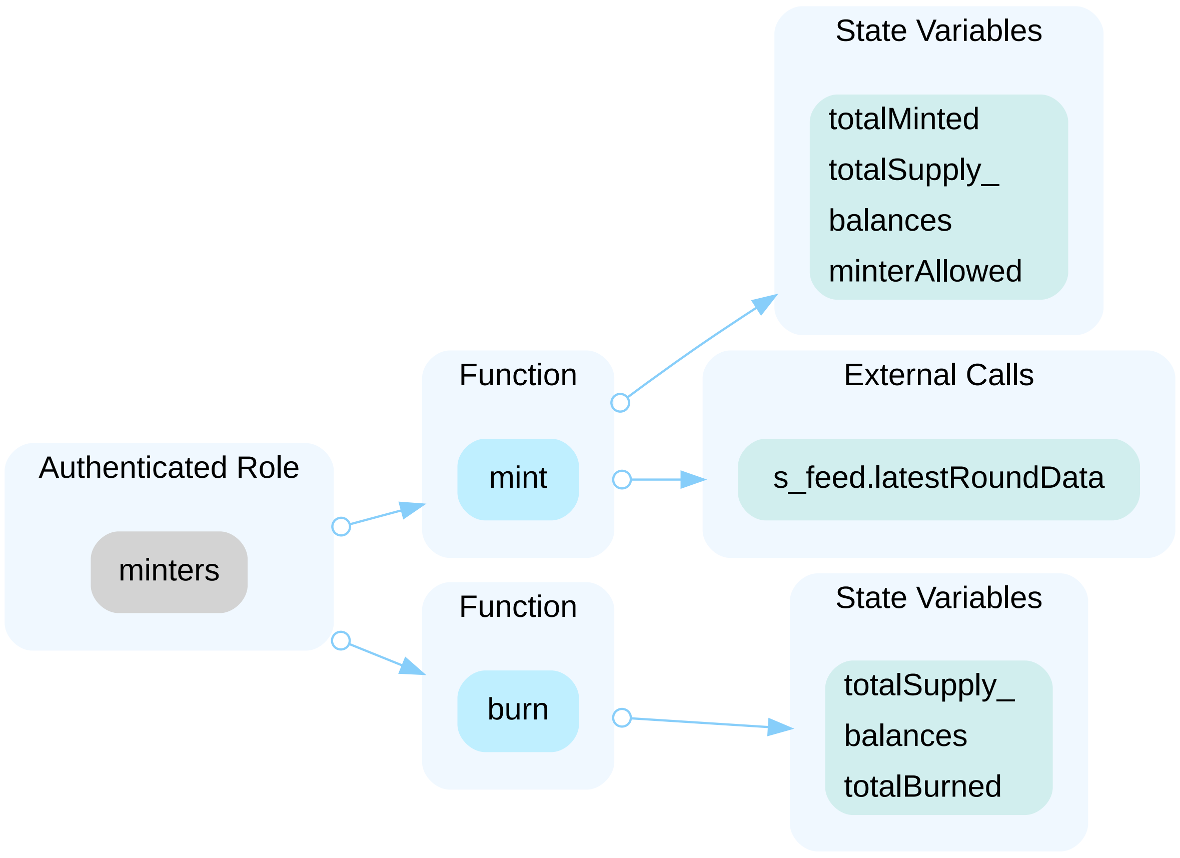
In the contract `Rescuable` the role `_rescuer` has authority over the functions shown in the diagram below. Any compromise to the `_rescuer` account may allow the hacker to take advantage of this authority to rescue extra tokens from the contract.



In the contract `RoyalEURO` the role `masterMinter` has authority over the functions shown in the diagram below. Any compromise to the `masterMinter` account may allow the hacker to take advantage of this authority to grant/revoke the role `minters` and mint tokens to an arbitrary address that is not on the blacklist.



In the contract `RoyalEURO` the role `minters` has authority over the functions shown in the diagram below. Any compromise to the `minters` account may allow the hacker to take advantage of this authority to mint tokens to an arbitrary address that is not on the blacklist.



In commit: 64e3df33d1f58561657bee0fdc85086e748378a8, the team introduced centralized control into the `initialize` function, enabling the `_owner` to execute the contract's `initialize` method.

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

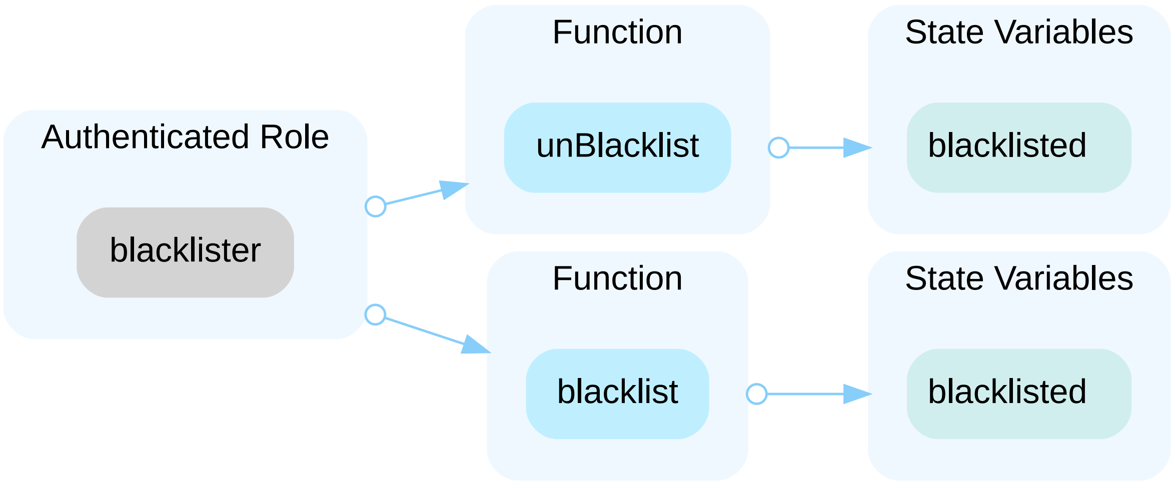
[Royal StableCoin 03/25/2024]: Issue acknowledged. We will implement multisig wallet on the mainnet.

RXA-01 | CENTRALIZATION RISKS IN RXAU.SOL

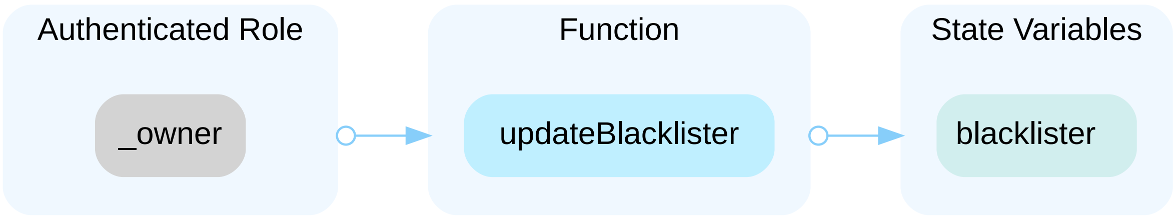
Category	Severity	Location	Status
Centralization	● **Major**	**RXAU.sol (03/12): 290, 298, 393, 405, 410, 489, 498, 503, 813, 831, 980, 1029, 1222, 1239, 1256, 1273, 1290, 1295**	● **Acknowledged**

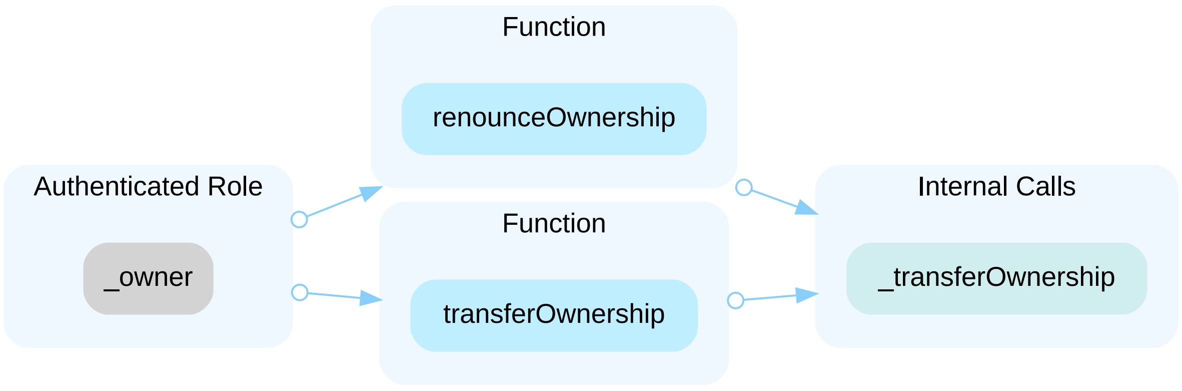
Description

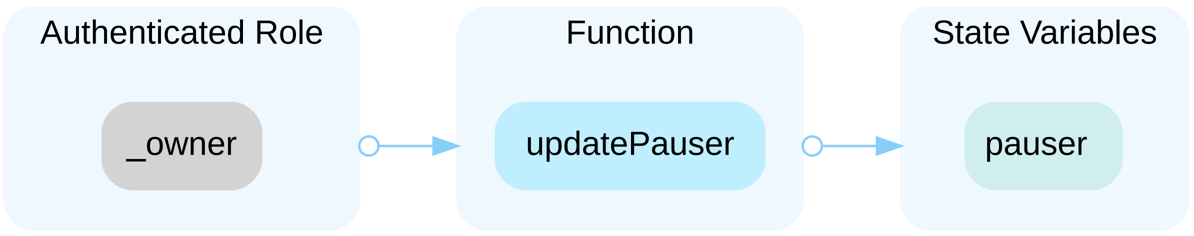
In the contract `Blacklistable` the role `blacklister` has authority over the functions shown in the diagram below. Any compromise to the `blacklister` account may allow the hacker to take advantage of this authority to add an address to the blacklist or remove from it.

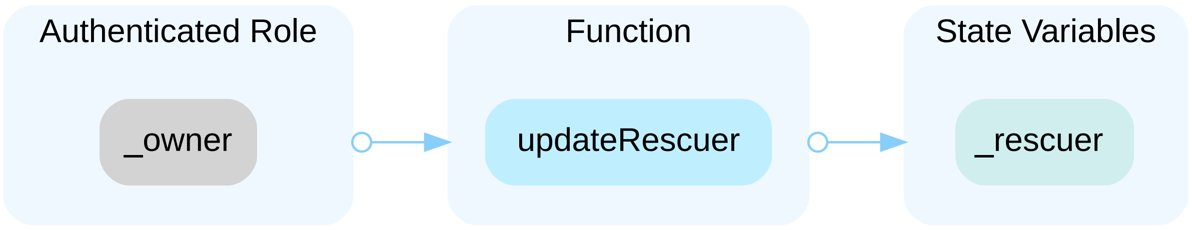


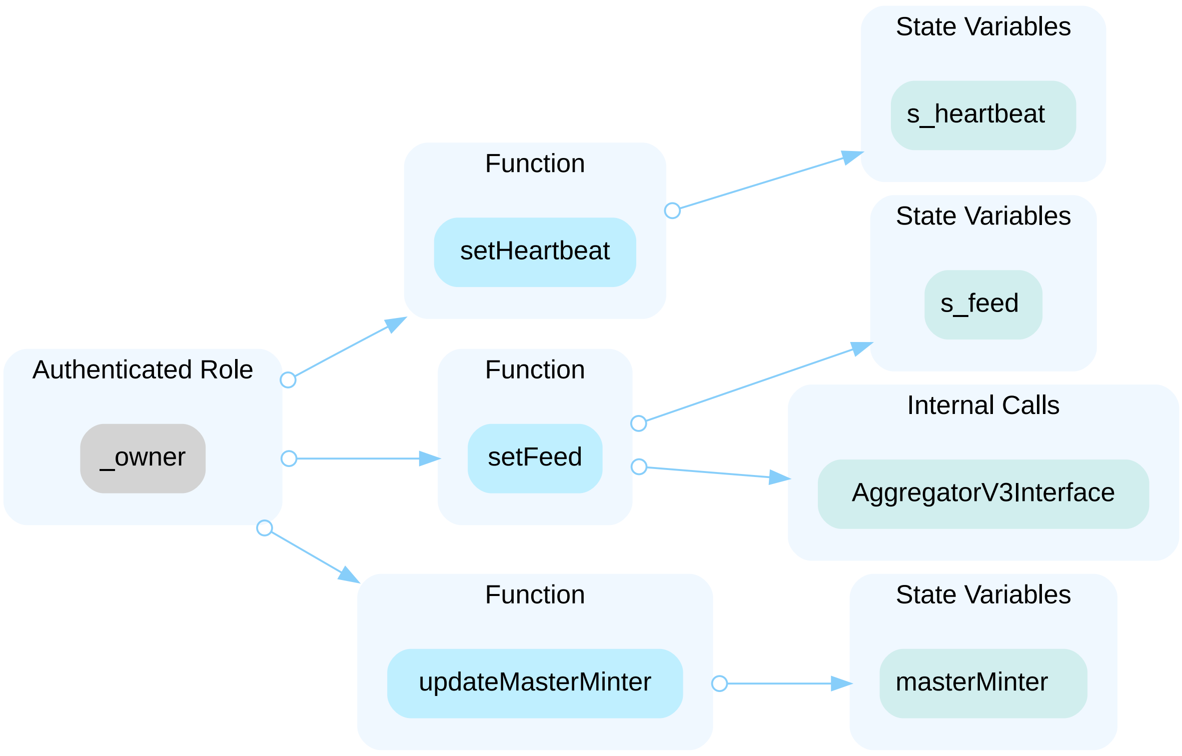
In the contract `RoyalEURO` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority to transfer ownership, change `blacklister` / `pauser` / `_rescuer` / `masterMinter` , set the heartbeat, set the address of the `s_feed` .



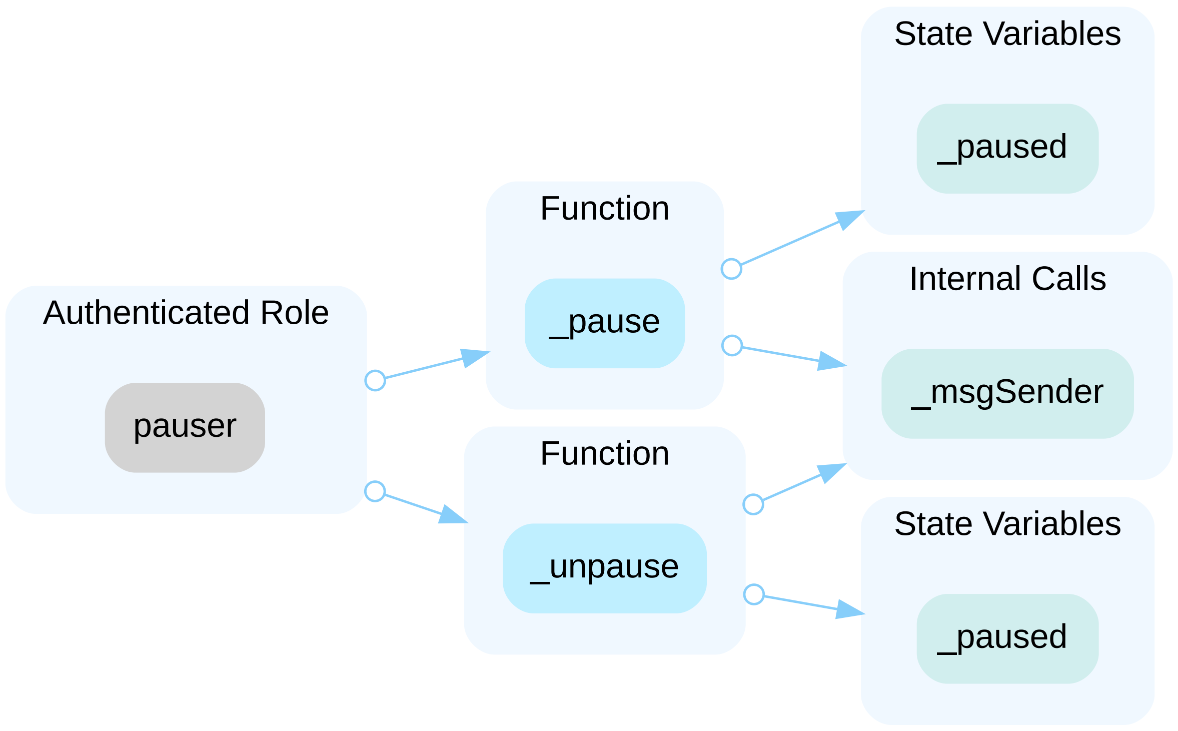




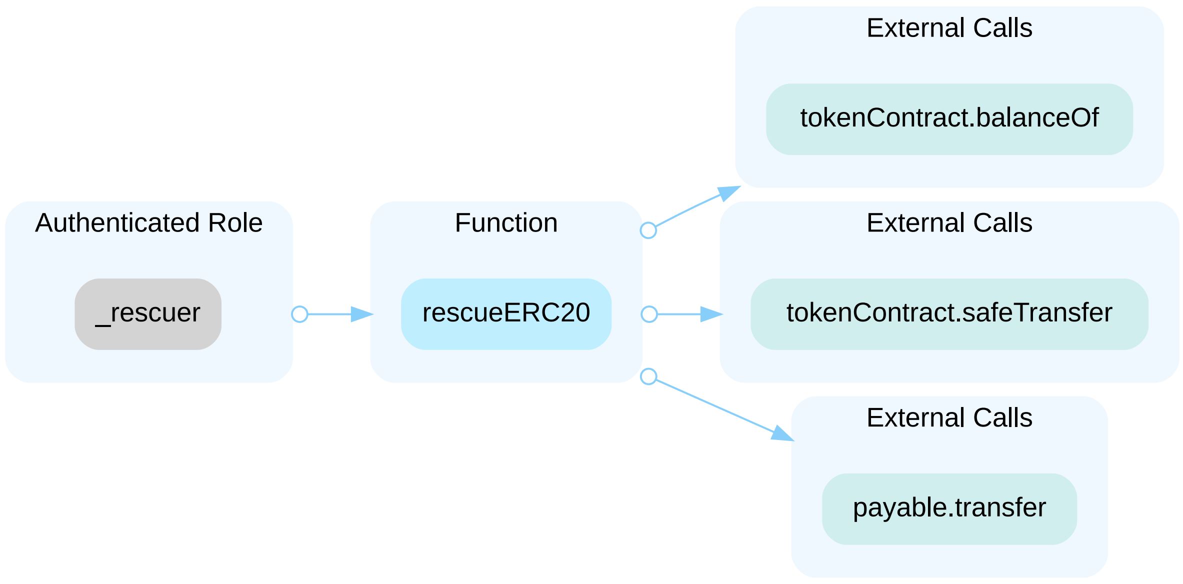




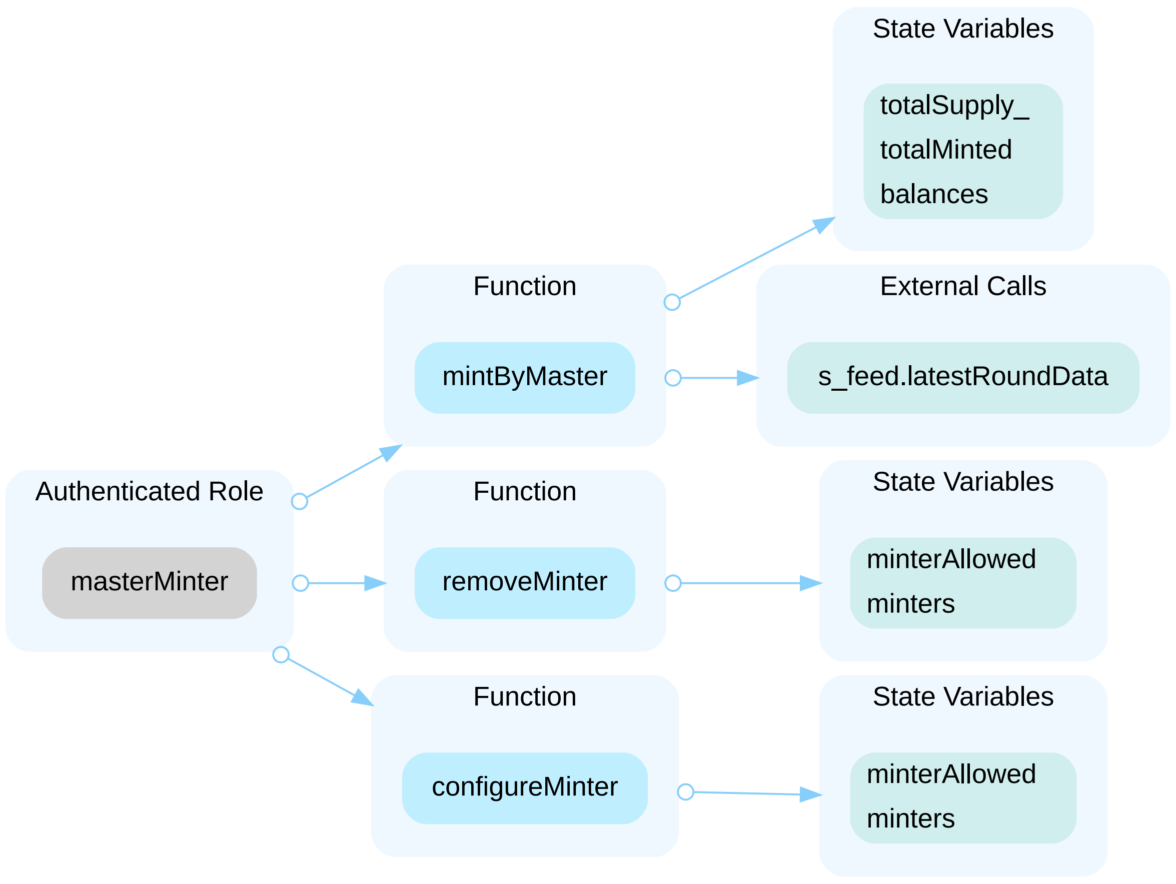
In the contract `Pausable` the role `pauser` has authority over the functions shown in the diagram below. Any compromise to the `pauser` account may allow the hacker to take advantage of this authority to `pause` / `unpause` the protocol.



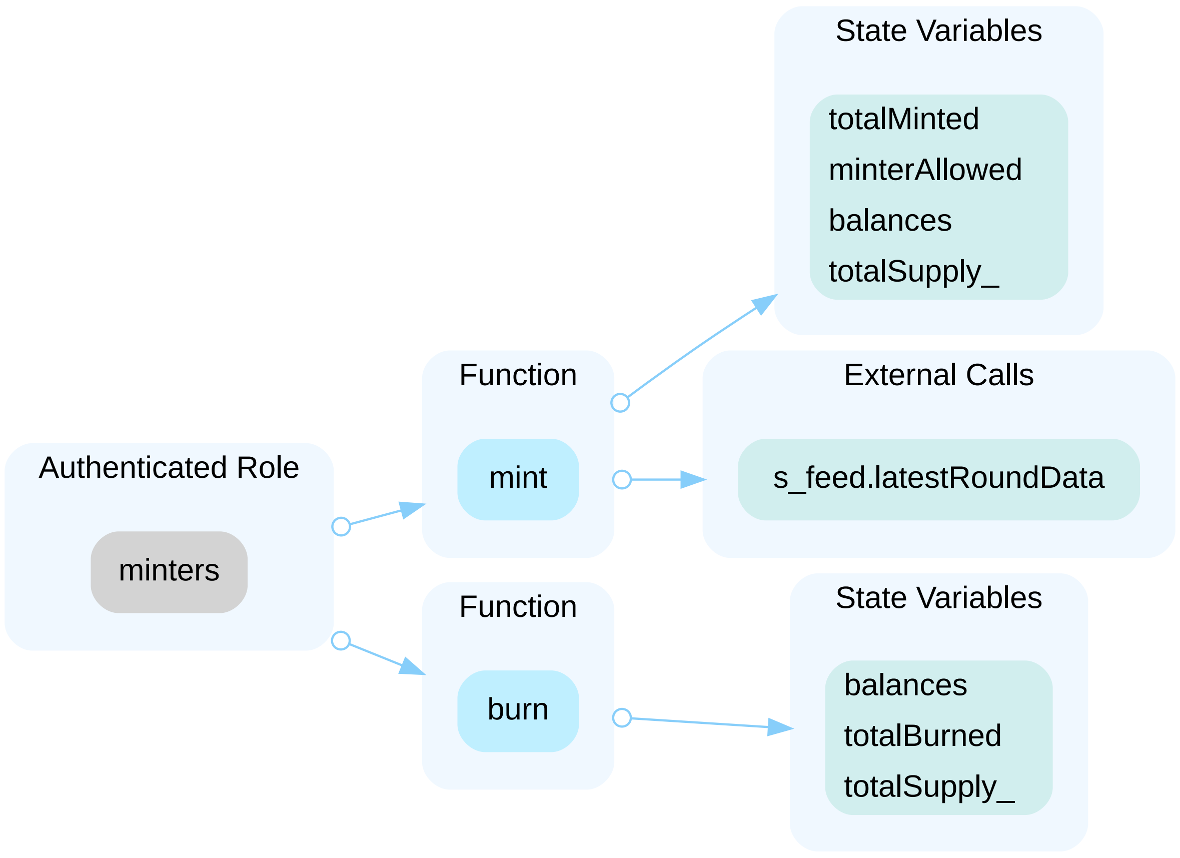
In the contract `Rescuable` the role `_rescuer` has authority over the functions shown in the diagram below. Any compromise to the `_rescuer` account may allow the hacker to take advantage of this authority to rescue extra tokens from the contract.



In the contract `RoyalEURO` the role `masterMinter` has authority over the functions shown in the diagram below. Any compromise to the `masterMinter` account may allow the hacker to take advantage of this authority to grant/revoke the role `minters` and mint tokens to an arbitrary address that is not on the blacklist.



In the contract `RoyalEURO` the role `minters` has authority over the functions shown in the diagram below. Any compromise to the `minters` account may allow the hacker to take advantage of this authority to mint tokens to an arbitrary address that is not on the blacklist.



In commit: 64e3df33d1f58561657bee0fdc85086e748378a8, the team introduced centralized control into the `initialize` function, enabling the `_owner` to execute the contract's `initialize` method.

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

[Royal StableCoin 03/25/2024]: Issue acknowledged. We will implement multisig wallet on the mainnet.

401-01 | INCOMPATIBILITY WITH DEFLATIONARY TOKENS

Category	Severity	Location	Status
Logical Issue	● Medium	REUR.sol (03/12): 708, 719; ROYAL.sol (03/12): 708, 719; RXAU.sol (03/12): 708, 719	● Acknowledged

Description

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

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

Recommendation

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

Alleviation

[Royal StableCoin 03/25/2024]: It is a stablecoin with no transaction fees and hence no deflationary mechanism is required.

401-03 | INIT FUNCTIONS ARE SUSCEPTIBLE TO FRONT-RUNNING

Category	Severity	Location	Status
Coding Issue	● Medium	REUR.sol (03/12): 925~964; ROYAL.sol (03/12): 924~963; RXAU.sol (03/12): 924~963	● Resolved

Description

The `initialize()` functions below are not called by another contract atomically after the contract is deployed, so it's possible for a malicious user to call `initialize()` which, if it's noticed in time, would require the project to re-deploy the contract in order to properly initialize.

Recommendation

We recommend creating a factory contract, which will `new` and `initialize()` each contract atomically. Or consider replacing the `initialize` function with a `constructor` since these contracts are not designed to be upgradable contracts.

Alleviation

[CertiK 03/25/2025]: The team resolved this issue at the commit 64e3df33d1f58561657bee0fdc85086e748378a8.

401-06 | THIRD-PARTY DEPENDENCY USAGE

Category	Severity	Location	Status
Design Issue	● Minor	ChainlinkReserveV3.sol (03/12): 7; REUR.sol (03/12): 898; ROYAL.sol (03/12): 897; RXAU.sol (03/12): 897	● Acknowledged

Description

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

- The contract `ReserveConsumerV3` interacts with third party contract with `AggregatorV3Interface` interface via `reserveFeed`.

- The contract `RoyalEURO` interacts with third party contract with `AggregatorV3Interface` interface via `s_feed`.

- The contract `RoyalDollar` interacts with third party contract with `AggregatorV3Interface` interface via `s_feed`.

- The contract `RoyalGold` interacts with third party contract with `AggregatorV3Interface` interface via `s_feed`.

Recommendation

The auditors understood that the business logic requires interaction with third parties. It is recommended for the team to constantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[Royal StableCoin 03/25/2024]: It is unavoidable and as the stablecoin uses PoR from chainlink, it has to depend upon this third-party.

[CertiK 03/25/2024]: We recognize that the contract relies on the `chainlink` POR. This observation is meant to prompt the team to keep an eye on the updates regarding `chainlink`.

RIB-01 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	REUR.sol (03/12): 961; ROYAL.sol (03/12): 818, 960; RXAU.sol (03/12): 818, 960; REUR.sol (03/25-64e3df33): 818	● Resolved

Description

Addresses are not validated before assignment or external calls, potentially allowing the use of zero addresses and leading to unexpected behavior or vulnerabilities. For example, transferring tokens to a zero address can result in a permanent loss of those tokens.

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

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

Recommendation

It is recommended to add a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

[CertiK 03/25/2024]: The team partially resolved this issue at commit: 64e3df33d1f58561657bee0fdc85086e748378a8. The `to` address passed to the `rescueERC20` function lacks a check for the zero address.

[Royal StableCoin 03/26/2024]: The "to" address is not being checked for zero address validation as there might be an intention to burn the tokens as well.

CRV-01 | HARDCODE ADDRESS

Category	Severity	Location	Status
Coding Style	● Informational	ChainlinkReserveV3.sol (03/12): 16	● Acknowledged

Description

The address of the POR Feed is hardcoded as `0xa81FE04086865e63E12dD3776978E49DEEa2ea4e` .

```
    constructor() {
        reserveFeed = AggregatorV3Interface(
            0xa81FE04086865e63E12dD3776978E49DEEa2ea4e
// will be changed to REUR PoR later
        );
    }
```

Recommendation

Consider change it to the correct address before deploying on the main chain.

Alleviation

[Roayl StableCoin 03/25/2024]: The address was hardcoded to run tests on the Ethereum testnet. It will be changed to the contract address of ROYAL PoR CA once Chainlink deploys it on mainnet and testnet.

REU-02 | DISCUSSION ON THE `MAX_HEARTBEAT_DAYS`

Category	Severity	Location	Status
Design Issue	● Informational	REUR.sol (03/12): 895	● Acknowledged

Description

The variable `MAX_HEARTBEAT_DAYS` has been assigned a value but is not being used. It appears that `MAX_HEARTBEAT_DAYS` has not been integrated into the business logic. The audit team would like to verify with the team the original purpose behind the `MAX_HEARTBEAT_DAYS` variable.

Recommendation

Consider deleting it once you've reviewed the initial design.

Alleviation

[Royal StableCoin 03/25/2024]: It was in the recommendation of the chainlink docs. Will review and remove it if not required at all or will make use of it in the business logic of the contract.

FORMAL VERIFICATION | ROYAL STABLECOINS - AUDIT

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

Considered Functions And Scope

In the following, we provide a description of the properties that have been used in this audit. They are grouped according to the type of contract they apply to.

Verification of Standard Ownable Properties

We verified *partial* properties of the public interfaces of those token contracts that implement the Ownable interface. This involves:

- function `owner` that returns the current owner,
- functions `renounceOwnership` that removes ownership,
- function `transferOwnership` that transfers the ownership to a new owner.

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

Property Name	Title
ownable-owner-succeed-normal	`owner` Always Succeeds
ownable-renounce-ownership-is-permanent	Once Renounced, Ownership Cannot be Regained
ownable-transferownership-correct	Ownership is Transferred.
ownable-renounceownership-correct	Ownership is Removed.

Verification of Pausable ERC-20 Compliance

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

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

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

Property Name	Title
erc20-transferfrom-fail-exceed-balance	`transferFrom` Fails if the Requested Amount Exceeds the Available Balance
erc20-approve-succeed-normal	`approve` Succeeds for Valid Inputs
erc20-transferfrom-fail-exceed-allowance	`transferFrom` Fails if the Requested Amount Exceeds the Available Allowance
erc20-approve-correct-amount	`approve` Updates the Approval Mapping Correctly
erc20-transfer-never-return-false	`transfer` Never Returns `false`
erc20pausable-transfer-succeed-normal	`transfer` Succeeds on Valid Transfers
erc20pausable-transferfrom-succeed-normal	`transferFrom` Succeeds on Valid Transfers
erc20-transfer-false	If `transfer` Returns `false` , the Contract State Is Not Changed
erc20-transferfrom-fail-recipient-overflow	`transferFrom` Prevents Overflows in the Recipient's Balance
erc20-transfer-recipient-overflow	`transfer` Prevents Overflows in the Recipient's Balance
erc20-transfer-revert-zero	`transfer` Prevents Transfers to the Zero Address
erc20-transfer-exceed-balance	`transfer` Fails if Requested Amount Exceeds Available Balance
erc20-allowance-change-state	`allowance` Does Not Change the Contract's State
erc20-transfer-correct-amount	`transfer` Transfers the Correct Amount in Transfers
erc20-balanceof-change-state	`balanceOf` Does Not Change the Contract's State
erc20-totalsupply-change-state	`totalSupply` Does Not Change the Contract's State
erc20-transferfrom-correct-allowance	`transferFrom` Updated the Allowance Correctly
erc20-transferfrom-correct-amount	`transferFrom` Transfers the Correct Amount in Transfers
erc20-totalsupply-correct-value	`totalSupply` Returns the Value of the Corresponding State Variable
erc20-approve-never-return-false	`approve` Never Returns `false`
erc20pausable-transferfrom-revert-paused	`transferFrom` Fails for a Paused Contract
erc20pausable-transfer-revert-paused	`transfer` Fails for a Paused Contract

Property Name	Title
erc20-totalsupply-succeed-always	`totalSupply` Always Succeeds
erc20-allowance-correct-value	`allowance` Returns Correct Value
erc20-allowance-succeed-always	`allowance` Always Succeeds
erc20-approve-false	If `approve` Returns `false`, the Contract's State Is Unchanged
erc20-approve-revert-zero	`approve` Prevents Approvals For the Zero Address
erc20-balanceof-succeed-always	`balanceOf` Always Succeeds
erc20-transferfrom-never-return-false	`transferFrom` Never Returns `false`
erc20-transferfrom-revert-zero-argument	`transferFrom` Fails for Transfers with Zero Address Arguments
erc20-transferfrom-false	If `transferFrom` Returns `false`, the Contract's State Is Unchanged
erc20-balanceof-correct-value	`balanceOf` Returns the Correct Value

Verification Results

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

Detailed Results For Contract Rescuable (REUR.sol) In Commit 4012eef2bb61b4448235eb22db7f89e5ad30aff4

Verification of Standard Ownable Properties

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounce-ownership-is-permanent	● True	
ownable-renounceownership-correct	● True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results For Contract Blacklistable (REUR.sol) In Commit 4012eef2bb61b4448235eb22db7f89e5ad30aff4

Verification of Standard Ownable Properties

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	
ownable-renounce-ownership-is-permanent	● True	

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results For Contract Vault (REUR.sol) In Commit 4012eef2bb61b4448235eb22db7f89e5ad30aff4

Verification of Standard Ownable Properties

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	
ownable-renounce-ownership-is-permanent	● True	

Detailed Results For Contract Vault (RXAU.sol) In Commit 4012eef2bb61b4448235eb22db7f89e5ad30aff4

Verification of Standard Ownable Properties

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	
ownable-renounce-ownership-is-permanent	● True	

Detailed Results For Contract Blacklistable (RXAU.sol) In Commit 4012eef2bb61b4448235eb22db7f89e5ad30aff4

Verification of Standard Ownable Properties

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	
ownable-renounce-ownership-is-permanent	● True	

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results For Contract Rescuable (RXAU.sol) In Commit 4012eef2bb61b4448235eb22db7f89e5ad30aff4

Verification of Standard Ownable Properties

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	
ownable-renounce-ownership-is-permanent	● True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results For Contract Rescuable (ROYAL.sol) In Commit 4012eef2bb61b4448235eb22db7f89e5ad30aff4

Verification of Standard Ownable Properties

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	
ownable-renounce-ownership-is-permanent	● True	

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results For Contract Blacklistable (ROYAL.sol) In Commit 4012eef2bb61b4448235eb22db7f89e5ad30aff4

Verification of Standard Ownable Properties

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	
ownable-renounce-ownership-is-permanent	● True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results For Contract Vault (ROYAL.sol) In Commit 4012eef2bb61b4448235eb22db7f89e5ad30aff4

Verification of Standard Ownable Properties

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	
ownable-renounce-ownership-is-permanent	● True	

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

In the remainder of this section, we list all contracts where formal verification of at least one property was not successful. There are several reasons why this could happen:

- False: The property is violated by the project.

- Inconclusive: The proof engine cannot prove or disprove the property due to timeouts or exceptions.
- Inapplicable: The property does not apply to the project.

Detailed Results For Contract RoyalEURO (REUR.sol) In Commit 4012eef2bb61b4448235eb22db7f89e5ad30aff4

Verification of Pausable ERC-20 Compliance

Detailed Results for Function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-fail-exceed-balance	● True	
erc20-transferfrom-fail-exceed-allowance	● True	
erc20pausable-transferfrom-succeed-normal	● Inapplicable	The property does not apply to the contract
erc20-transferfrom-fail-recipient-overflow	● True	
erc20-transferfrom-correct-allowance	● True	
erc20-transferfrom-correct-amount	● True	
erc20pausable-transferfrom-revert-paused	● True	
erc20-transferfrom-never-return-false	● True	
erc20-transferfrom-revert-zero-argument	● True	
erc20-transferfrom-false	● True	

Detailed Results for Function `approve`

Property Name	Final Result	Remarks
erc20-approve-succeed-normal	● Inapplicable	The property does not apply to the contract
erc20-approve-correct-amount	● True	
erc20-approve-never-return-false	● True	
erc20-approve-false	● True	
erc20-approve-revert-zero	● True	

Detailed Results for Function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-never-return-false	● True	
erc20pausable-transfer-succeed-normal	● Inapplicable	The property does not apply to the contract
erc20-transfer-false	● True	
erc20-transfer-recipient-overflow	● True	
erc20-transfer-revert-zero	● True	
erc20-transfer-exceed-balance	● True	
erc20-transfer-correct-amount	● True	
erc20pausable-transfer-revert-paused	● True	

Detailed Results for Function `allowance`

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

Detailed Results for Function `balanceOf`

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

Detailed Results for Function `totalSupply`

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

Detailed Results For Contract RoyalGold (RXAU.sol) In Commit 4012eef2bb61b4448235eb22db7f89e5ad30aff4

Verification of Pausable ERC-20 Compliance

Detailed Results for Function `totalSupply`

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

Detailed Results for Function `allowance`

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

Detailed Results for Function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-correct-amount	● True	
erc20pausable-transfer-revert-paused	● True	
erc20-transfer-revert-zero	● True	
erc20-transfer-never-return-false	● True	
erc20pausable-transfer-succeed-normal	● Inapplicable	The property does not apply to the contract
erc20-transfer-false	● True	
erc20-transfer-recipient-overflow	● True	
erc20-transfer-exceed-balance	● True	

Detailed Results for Function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-correct-allowance	● True	
erc20-transferfrom-correct-amount	● True	
erc20pausable-transferfrom-revert-paused	● True	
erc20-transferfrom-false	● True	
erc20-transferfrom-never-return-false	● True	
erc20pausable-transferfrom-succeed-normal	● Inapplicable	The property does not apply to the contract
erc20-transferfrom-revert-zero-argument	● True	
erc20-transferfrom-fail-exceed-balance	● True	
erc20-transferfrom-fail-recipient-overflow	● True	
erc20-transferfrom-fail-exceed-allowance	● True	

Detailed Results for Function `approve`

Property Name	Final Result	Remarks
erc20-approve-never-return-false	● True	
erc20-approve-succeed-normal	● Inapplicable	The property does not apply to the contract
erc20-approve-false	● True	
erc20-approve-revert-zero	● True	
erc20-approve-correct-amount	● True	

Detailed Results for Function `balanceOf`

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

Detailed Results For Contract RoyalDollar (ROYAL.sol) In Commit 4012eef2bb61b4448235eb22db7f89e5ad30aff4

Verification of Pausable ERC-20 Compliance

Detailed Results for Function `transfer`

Property Name	Final Result	Remarks
erc20-transfer-exceed-balance	● True	
erc20-transfer-correct-amount	● True	
erc20pausable-transfer-revert-paused	● True	
erc20-transfer-false	● True	
erc20-transfer-recipient-overflow	● True	
erc20-transfer-never-return-false	● True	
erc20-transfer-revert-zero	● True	
erc20pausable-transfer-succeed-normal	● Inapplicable	The property does not apply to the contract

Detailed Results for Function `allowance`

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

Detailed Results for Function `balanceOf`

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

Detailed Results for Function `totalSupply`

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

Detailed Results for Function `transferFrom`

Property Name	Final Result	Remarks
erc20-transferfrom-correct-allowance	● True	
erc20-transferfrom-correct-amount	● True	
erc20pausable-transferfrom-revert-paused	● True	
erc20-transferfrom-never-return-false	● True	
erc20-transferfrom-false	● True	
erc20-transferfrom-revert-zero-argument	● True	
erc20pausable-transferfrom-succeed-normal	● Inapplicable	The property does not apply to the contract
erc20-transferfrom-fail-recipient-overflow	● True	
erc20-transferfrom-fail-exceed-balance	● True	
erc20-transferfrom-fail-exceed-allowance	● True	

Detailed Results for Function `approve`

Property Name	Final Result	Remarks
erc20-approve-never-return-false	● True	
erc20-approve-false	● True	
erc20-approve-revert-zero	● True	
erc20-approve-succeed-normal	● Inapplicable	The property does not apply to the contract
erc20-approve-correct-amount	● True	

APPENDIX | ROYAL STABLECOINS - AUDIT

Finding Categories

Categories	Description
Coding Style	Coding Style findings may not affect code behavior, but indicate areas where coding practices can be improved to make the code more understandable and maintainable.
Coding Issue	Coding Issue findings are about general code quality including, but not limited to, coding mistakes, compile errors, and performance issues.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.
Design Issue	Design Issue findings indicate general issues at the design level beyond program logic that are not covered by other finding categories.

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.

Details on Formal Verification

Some Solidity smart contracts from this project have been formally verified. Each such contract was compiled into a mathematical model that reflects all its possible behaviors with respect to the property. The model takes into account the semantics of the Solidity instructions found in the contract. All verification results that we report are based on that model.

The following assumptions and simplifications apply to our model:

- Certain low-level calls and inline assembly are not supported and may lead to a contract not being formally verified.
- We model the semantics of the Solidity source code and not the semantics of the EVM bytecode in a compiled contract.

Formalism for property specifications

All properties are expressed in a behavioral interface specification language that CertiK has developed for Solidity, which allows us to specify the behavior of each function in terms of the contract state and its parameters and return values, as well as contract properties that are maintained by every observable state transition. Observable state transitions occur when the contract's external interface is invoked and the invocation does not revert, and when the contract's Ether balance is changed by the EVM due to another contract's "self-destruct" invocation. The specification language has the usual Boolean connectives, as well as the operator `\old` (used to denote the state of a variable before a state transition), and several types of specification clause:

Apart from the Boolean connectives and the modal operators "always" (written `[]`) and "eventually" (written `<>`), we use the following predicates to reason about the validity of atomic propositions. They are evaluated on the contract's state whenever a discrete time step occurs:

- `requires [cond]` - the condition `cond`, which refers to a function's parameters, return values, and contract state variables, must hold when a function is invoked in order for it to exhibit a specified behavior.
- `ensures [cond]` - the condition `cond`, which refers to a function's parameters, return values, and both `\old` and current contract state variables, is guaranteed to hold when a function returns if the corresponding requires condition held when it was invoked.
- `invariant [cond]` - the condition `cond`, which refers only to contract state variables, is guaranteed to hold at every observable contract state.
- `constraint [cond]` - the condition `cond`, which refers to both `\old` and current contract state variables, is guaranteed to hold at every observable contract state except for the initial state after construction (because there is no previous state); constraints are used to restrict how contract state can change over time.

Description of the Analyzed ERC-20-Pausable Properties

Properties related to function `transferFrom`

erc20-transferfrom-correct-allowance

All non-reverting invocations of `transferFrom(from, dest, amount)` that return `true` must decrease the allowance for address `msg.sender` over address `from` by the value in `amount`.

Specification:

```
ensures \result ==> allowance(\old(sender), msg.sender) == \old(allowance(sender,
msg.sender)) - \old(amount)
                  || (allowance(\old(sender), msg.sender) == \old(allowance(sender,
msg.sender)) && \old(allowance(sender, msg.sender)) == type(uint256).max);
```

erc20-transferfrom-correct-amount

All invocations of `transferFrom(from, dest, amount)` that succeed and that return `true` subtract the value in `amount` from the balance of address `from` and add the same value to the balance of address `dest`.

Specification:

```
requires recipient != sender;
requires balanceOf(recipient) + amount <= type(uint256).max;
ensures \result ==> balanceOf(\old(recipient)) == \old(balanceOf(recipient) +
amount)
                  && balanceOf(\old(sender)) == \old(balanceOf(sender) - amount);
 also
requires recipient == sender;
ensures \result ==> balanceOf(\old(recipient)) == \old(balanceOf(recipient));
```

erc20-transferfrom-fail-exceed-allowance

Any call of the form `transferFrom(from, dest, amount)` with a value for `amount` that exceeds the allowance of address `msg.sender` must fail.

Specification:

```
requires msg.sender != sender;
requires amount > allowance(sender, msg.sender);
ensures !\result;
```

erc20-transferfrom-fail-exceed-balance

Any call of the form `transferFrom(from, dest, amount)` with a value for `amount` that exceeds the balance of address `from` must fail.

Specification:

```
requires amount > balanceOf(sender);
ensures !\result;
```

erc20-transferfrom-fail-recipient-overflow

Any call of `transferFrom(from, dest, amount)` with a value in `amount` whose transfer would cause an overflow of the balance of address `dest` must fail.

Specification:

```
requires recipient != sender;
requires balanceOf(recipient) + amount > type(uint256).max;
ensures !\result;
```

erc20-transferfrom-false

If `transferFrom` returns `false` to signal a failure, it must undo all incurred state changes before returning to the caller.

Specification:

```
ensures !\result ==> \assigned (\nothing);
```

erc20-transferfrom-never-return-false

The `transferFrom` function must never return `false`.

Specification:

```
ensures \result;
```

erc20-transferfrom-revert-zero-argument

All calls of the form `transferFrom(from, dest, amount)` must fail for transfers from or to the zero address.

Specification:

```
ensures \old(sender) == address(0) ==> !\result;
also
ensures \old(recipient) == address(0) ==> !\result;
```

erc20pausable-transferfrom-revert-paused

Any call of the form `transferFrom(from, dest, amount)` must fail for a paused contract.

Specification:

```
reverts_when paused();
```

erc20pausable-transferfrom-succeed-normal

All invocations of `transferFrom(from, dest, amount)` must succeed and return `true` if

- the value of `amount` does not exceed the balance of address `from`,
- the value of `amount` does not exceed the allowance of `msg.sender` for address `from`,
- transferring a value of `amount` to the address in `dest` does not lead to an overflow of the recipient's balance,
- the contract is not paused, and
- the supplied gas suffices to complete the call.

Specification:

```
requires recipient != address(0) && sender != address(0) && recipient != sender;
requires !paused();
requires amount <= balanceOf(sender);
requires amount <= allowance(sender, msg.sender);
requires balanceOf(recipient) + amount <= type(uint256).max;
ensures \result;
reverts_only_when false;
```

Properties related to function `approve`

erc20-approve-correct-amount

All non-reverting calls of the form `approve(spender, amount)` that return `true` must correctly update the allowance mapping according to the address `msg.sender` and the values of `spender` and `amount`.

Specification:

```
requires spender != address(0);
ensures \result ==> allowance(msg.sender, \old(spender)) == \old(amount);
```

erc20-approve-false

If function `approve` returns `false` to signal a failure, it must undo all state changes that it incurred before returning to the caller.

Specification:

```
ensures !\result ==> \assigned (\nothing);
```

erc20-approve-never-return-false

The function `approve` must never returns `false`.

Specification:

```
ensures \result;
```

erc20-approve-revert-zero

All calls of the form `approve(spender, amount)` must fail if the address in `spender` is the zero address.

Specification:

```
ensures \old(spender) == address(0) ==> !\result;
```

erc20-approve-succeed-normal

All calls of the form `approve(spender, amount)` must succeed, if

- the address in `spender` is not the zero address and
- the execution does not run out of gas.

Specification:

```
requires spender != address(0);
ensures \result;
reverts_only_when false;
```

Properties related to function `transfer`

erc20-transfer-correct-amount

All non-reverting invocations of `transfer(recipient, amount)` that return `true` must subtract the value in `amount` from the balance of `msg.sender` and add the same value to the balance of the `recipient` address.

Specification:

```
requires recipient != msg.sender;
requires balanceOf(recipient) + amount <= type(uint256).max;
ensures \result ==> balanceOf(recipient) == \old(balanceOf(recipient) + amount)
&& balanceOf(msg.sender) == \old(balanceOf(msg.sender) - amount);
  also
requires recipient == msg.sender;
ensures \result ==> balanceOf(msg.sender) == \old(balanceOf(msg.sender));
```

erc20-transfer-exceed-balance

Any transfer of an amount of tokens that exceeds the balance of `msg.sender` must fail.

Specification:

```
requires amount > balanceOf(msg.sender);
ensures !\result;
```

erc20-transfer-false

If the `transfer` function in contract `RoyalEURO` fails by returning `false`, it must undo all state changes it incurred before returning to the caller.

Specification:

```
ensures !\result ==> \assigned (\nothing);
```

erc20-transfer-false

If the `transfer` function in contract `RoyalGold` fails by returning `false`, it must undo all state changes it incurred before returning to the caller.

Specification:

```
ensures !\result ==> \assigned (\nothing);
```

erc20-transfer-false

If the `transfer` function in contract `RoyalDollar` fails by returning `false`, it must undo all state changes it incurred before returning to the caller.

Specification:

```
ensures !\result ==> \assigned (\nothing);
```

erc20-transfer-never-return-false

The transfer function must never return `false` to signal a failure.

Specification:

```
ensures \result;
```

erc20-transfer-recipient-overflow

Any invocation of `transfer(recipient, amount)` must fail if it causes the balance of the `recipient` address to overflow.

Specification:

```
requires recipient != msg.sender;
requires balanceOf(recipient) + amount > type(uint256).max;
ensures !\result;
```

erc20-transfer-revert-zero

Any call of the form `transfer(recipient, amount)` must fail if the recipient address is the zero address.

Specification:

```
ensures \old(recipient) == address(0) ==> !\result;
```

erc20pausable-transfer-revert-paused

Any invocation of `transfer(recipient, amount)` must fail if the contract is paused.

Specification:

```
reverts_when paused();
```

erc20pausable-transfer-succeed-normal

All invocations of the form `transfer(recipient, amount)` must succeed and return `true` if

- the `recipient` address is not the zero address,
- the contract is not paused,
- `amount` does not exceed the balance of address `msg.sender` ,
- transferring `amount` to the `recipient` address does not lead to an overflow of the recipient's balance, and
- the supplied gas suffices to complete the call.

Specification:

```
requires recipient != address(0) && recipient != msg.sender;
requires !paused();
requires amount <= balanceOf(msg.sender);
requires balanceOf(recipient) + amount <= type(uint256).max;
ensures \result;
reverts_only_when false;
```

Properties related to function `allowance`

erc20-allowance-change-state

Function `allowance` must not change any of the contract's state variables.

Specification:

```
assignable \nothing;
```

erc20-allowance-correct-value

Invocations of `allowance(owner, spender)` must return the allowance that address `spender` has over tokens held by address `owner` .

Specification:

```
ensures \result == allowance(\old(owner), \old(spender));
```

erc20-allowance-succeed-always

Function `allowance` must always succeed, assuming that its execution does not run out of gas.

Specification:

```
reverts_only_when false;
```

Properties related to function `balanceOf`

erc20-balanceof-change-state

Function `balanceOf` must not change any of the contract's state variables.

Specification:

```
assignable \nothing;
```

erc20-balanceof-correct-value

Invocations of `balanceOf(owner)` must return the value that is held in the contract's balance mapping for address `owner` .

Specification:

```
ensures \result == balanceOf(\old(account));
```

erc20-balanceof-succeed-always

Function `balanceOf` must always succeed if it does not run out of gas.

Specification:

```
reverts_only_when false;
```

Properties related to function `totalSupply`

erc20-totalsupply-change-state

The `totalSupply` function in contract RoyalEURO must not change any state variables.

Specification:

```
assignable \nothing;
```

erc20-totalsupply-change-state

The `totalSupply` function in contract RoyalGold must not change any state variables.

Specification:

```
assignable \nothing;
```

erc20-totalsupply-change-state

The `totalSupply` function in contract RoyalDollar must not change any state variables.

Specification:

```
assignable \nothing;
```

erc20-totalsupply-correct-value

The `totalSupply` function must return the value that is held in the corresponding state variable of contract RoyalEURO.

Specification:

```
ensures \result == totalSupply();
```

erc20-totalsupply-correct-value

The `totalSupply` function must return the value that is held in the corresponding state variable of contract RoyalGold.

Specification:

```
ensures \result == totalSupply();
```

erc20-totalsupply-correct-value

The `totalSupply` function must return the value that is held in the corresponding state variable of contract RoyalDollar.

Specification:

```
ensures \result == totalSupply();
```

erc20-totalsupply-succeed-always

The function `totalSupply` must always succeeds, assuming that its execution does not run out of gas.

Specification:

```
reverts_only_when false;
```

Description of the Analyzed Ownable Properties

Properties related to function `owner`

ownable-owner-succeed-normal

Function `owner` must always succeed if it does not run out of gas.

Specification:

```
reverts_only_when false;
```

Properties related to function `renounceOwnership`

ownable-renounce-ownership-is-permanent

The contract must prohibit regaining of ownership once it has been renounced.

Specification:

```
constraint \old(owner()) == address(0) ==> owner() == address(0);
```

ownable-renounceownership-correct

Invocations of `renounceOwnership()` must set ownership to address(0).

Specification:

```
ensures this.owner() == address(0);
```

Properties related to function `transferOwnership`

ownable-transferownership-correct

Invocations of `transferOwnership(newOwner)` must transfer the ownership to the `newOwner`.

Specification:

```
ensures this.owner() == newOwner;
```

DISCLAIMER | CERTIK

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.