CERTIK

Security Assessment

Flying Cash

May 12th, 2022

Table of Contents

Summary

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

The auditing process pays special attention to the following considerations:

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

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

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

Overview

Project Summary

Project Name	Flying Cash
Platform	Ethereum
Language	Solidity
Codebase	1. https://github.com/flyingcash/flyingcash-contracts 2. https://github.com/flyingcash/tokbridge-contracts
Commit	1. 82f57ab1482ccfaccb226fbba18951db5c5c7b65 2. f91c3c921c148838fd81674cb279b9713d114092

Audit Summary

Delivery Date	May 12, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	3	0	0	0	0	0	3
Major	5	0	0	3	0	0	2
Medium	6	0	0	5	0	0	1
Minor	13	0	0	9	0	0	4
Informational	10	0	0	7	0	0	3
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
FCN	contracts/FlyingCashAdapterNoAsset.sol	e356606cbc9710e04a5bc8d17d8e238b222b562584bf42b17e2e40cad185176f
ERF	contracts/upgradeable_contracts/ERC677BridgeForBurnableMintableToken.sol	70411ae02cbb01ff7a95fd77ae9b53cef45981a6226255882f157be4a8a0e15f
BOC	contracts/BoringOwnable.sol	8d299b7c8ae5ee7227ace1bc23b2d7cc355ba6e14fb27414cbb16143243a6993
BBC	contracts/upgradeable_contracts/BasicBridge.sol	411fef90c618b33655ca3b2b2342c6c68d344696f19ff297976f882c2d0e74fa
HMA	contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/HomeMultiAMBErc20ToErc677.sol	720ca079adaef96e2c9331121edbacaf305f531857c8bcb75ad93a221795c6c9
BBV	contracts/upgradeable_contracts/BaseBridgeValidators.sol	7c804c0343f8199de472e179ca3cdd2176cd5e452e85b777376f705bb55d74ec
BRA	contracts/upgradeable_contracts/BaseRewardAddressList.sol	ad6908fe3dbcba2030cf5b22b8d002814aac6f64de2d3819ba0c5d42826e1031
ERS	contracts/upgradeable_contracts/ERC677Storage.sol	5247f2d0100d741ac1009656647cacf52fe04e971cf34c6ea792118eb83769e0
CCC	contracts/upgradeable_contracts/ChaiConnector.sol	0bbfab45cdd9a49314b8803175df9cafb8257c7b11b0ae46db93996cc320c0ea
HFM	contracts/upgradeable_contracts/amb_native_to_erc20/HomeFeeManagerAMBNativeToErc20.sol	d1d3e0d740e9fb512303474ed7cfc9216f3ac6da4933dbe9cfd92a0d5c5c359b
OMC	contracts/upgradeable_contracts/OverdrawManagement.sol	4a62bad21b3825f26f6e956e7de645728a5c9e9737da030fd2788fe34c5ea71f
BER	contracts/upgradeable_contracts/BaseERC677Bridge.sol	5d9e8a5c6fedd332a4d5ce008ec6180f2d5b069bc73ada092887891f1e47d2c5
ERC	contracts/upgradeable_contracts/ERC20Bridge.sol	980f8b246a67c42057200782f67d7b432382720eae87b9d2ae7e2f384353b1a9
VCK	contracts/Voucher.sol	79e7d9cf64d86b19534143abee605d6bd9e3a45a23a5cf97ac53bbb0d329d15e

ID	File	SHA256 Checksum
FAB	contracts/upgradeable_contracts/arbitrary_message/ForeignAMB.sol	eed670f5a9a3561f9c521ea5aedde5f17139cbe31c55d5a2337508d93c12f05b
ERB	contracts/upgradeable_contracts/ERC677Bridge.sol	844d471a9c2387fa634734ce6b694359fe72799ee9594f3264104a272aee31cb
BAC	contracts/upgradeable_contracts/BasicAMBMediator.sol	b9edc98cf5f0b363ff1ea67a2270f63fae7bc029b66af7f928120959df1d00d9
DSB	contracts/upgradeable_contracts/DecimalShiftBridge.sol	c80542b92f5453a7f2d737c2638f55604d151a84220f2860693389b979c59e3f
BRB	contracts/upgradeable_contracts/BlockRewardBridge.sol	2265f6d1b349455e11ceae1edc029d6303d0a769d18cd2110f34f769408e3d20
TIS	contracts/upgradeable_contracts/TransferInfoStorage.sol	b394a164de3252a074b1755da83a0a7630bb273ebd80c331d07c5bb9539211c3
BMA	contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/BasicMultiAMBErc20ToErc677.sol	2899b4968924d944ceefac5bdeeb37c48c0115f0ec9ac6332b155f386d5fd768
ERT	contracts/ERC677BridgeToken.sol	526ada570713c5c5506ddfcb380e61b98d381272a6d0fa8e99b1249d65bdde5f
BOM	contracts/upgradeable_contracts/BaseOverdrawManagement.sol	a936ffc60eeb5ebdde2a272ccc8bfdbe6f0f5169732db9843fd76470abfe112c
TSC	contracts/upgradeable_contracts/TokenSwapper.sol	9a43d66482ae25be79a96024325b5648e3274d5da230a81eaa9b92c4223e4aac
GTC	contracts/upgradeable_contracts/GasTokenConnector.sol	5aae0bdbd1af2e7a315a548c0c2ef51a7304263eca5b6b6e704425e5b33f8f45
BVC	contracts/upgradeable_contracts/BridgeValidators.sol	60ab1cdd2fda228eaf94b46757e62bdea737a3da42c6694a1868b00d04d746f7
BRF	contracts/upgradeable_contracts/BlockRewardFeeManager.sol	95110ed77b2b0c2847cfd84a48f72d0e984683302c8cee7ccda87b6f678168b2
BMF	contracts/upgradeable_contracts/BaseMediatorFeeManager.sol	0d3d93a55245fc312c4a38ca7baebf5b39ff16e6f0b3b93c199a57b91e03799f
BHA	contracts/upgradeable_contracts/arbitrary_message/BasicHomeAMB.sol	af647f637a210543d7be3a0192e7d8ed739dbc28a453bab7fbb602a77a67878b

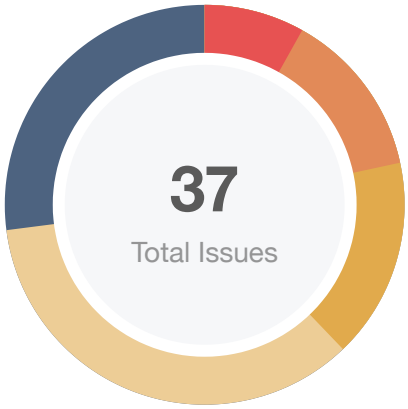
ID	File	SHA256 Checksum
MRC	contracts/upgradeable_contracts/MessageRelay.sol	576ec281e15c6e8e9c24801c81e68eb48ed0fffc7e1c725ee03c9c731b7481ec
BTB	contracts/upgradeable_contracts/BasicTokenBridge.sol	873cebf1226bdf2721f2661fd098a722b289b4b74f654c4efacb10cbfacb2091
FMA	contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/ForeignMultiAMBErc20ToErc677.sol	4849f7597ea073b14594525847e0e2ba164b902fc0ab10f11ea652b4c67eecbb
HAB	contracts/upgradeable_contracts/arbitrary_message/HomeAMB.sol	d4cbc4cc21ad3e4b086b37b125e85837b75f73ca84ec4d7f5675ac042c7697bf
VAM	contracts/upgradeable_contracts/arbitrary_message/VersionableAMB.sol	a6e3c668c258b4c13b9585ae4d1459a3d3f230ac823ff2730161ef4dc69cdda7
FCC	contracts/FlyingCash.sol	2981ee089bd36464495b9c9e8bac56c8f11e66d1cff953ffd6af093e5e73433b
HFA	contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/HomeFeeManagerMultiAMBErc20ToErc677.sol	b03587e467ae44e077bfbe7882c9010fe3f98baea5248ec8ed0ff8c8c9c942da
FAW	contracts/upgradeable_contracts/arbitrary_message/ForeignAMBWithGasToken.sol	2bc1c9308e287646b3b4c9167feaf3001278c4205573f00fc4b43a058f69eee7
RMC	contracts/upgradeable_contracts/RewardableMediator.sol	84e18369bd4101af6cfae75a94aeec22b76e90911e3cc44b2e1a45e5bc9e64f4
FMD	contracts/FeeManagerDefault.sol	e5eadcfb307b33a75718469441517973159a70e98bbb644647eb88d2aa5d85e5
VSC	contracts/upgradeable_contracts/ValidatorStorage.sol	7b3d62ec7ec5ed1ecebe83a0c9b2155d8bca52a1bd7e420bea51577e85836f7a
VBC	contracts/upgradeable_contracts/VersionableBridge.sol	f655be52ed253aede05ade14a8ab995095e02d437e1cb72db5e3286ff024b822
FCA	contracts/FlyingCashAdapterFilda.sol	2db361d9eddbfd9c2c7a2b8f43b376bbd480b076bbe2b15f5c30d8a3d7b1b1fc
RGC	contracts/upgradeable_contracts/ReentrancyGuard.sol	3376a33dd88f7fa06319717e03f284817a554968b89be27796e9dcdacacd4dbf
MPC	contracts/upgradeable_contracts/arbitrary_message/MessageProcessor.sol	8cf49071dc5dced1c02e1f51a3df8a00545f80ac2b8c5394355f79da5a13b8f3

ID	File	SHA256 Checksum
RVC	contracts/upgradeable_contracts/RewardableValidators.sol	cd29cc8b3ddcd69aa9d1c4daa007c0938a84958c172552300e6f7c04ad04f37a
ERR	contracts/ERC677BridgeTokenRewardable.sol	a433772b617607117b55e3ee1899857ec7433fbe6dfcc27930c1c7d8de7ac9af
FFM	contracts/upgradeable_contracts/amb_native_to_erc20/ForeignFeeManagerAMBNativeToErc20.sol	4d1438e840cf914a43f7e00b473c26c576e6af7b30efc316ee1968694e44170d
BMT	contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/BasicMultiTokenBridge.sol	effef23478d97c3e263a766e78c45668901f7cfadffb7a6e4fe9a576926adcbe
BAM	contracts/upgradeable_contracts/amb_native_to_erc20/BasicAMBNativeToErc20.sol	1457dafdcfec201edf64e249e1afc0ca9013dc4213dab067bcf8e9b888a98aad
HAM	contracts/upgradeable_contracts/amb_native_to_erc20/HomeAMBNativeToErc20.sol	db99f19c35410911148692963dae38ceb2a9bfddd8fbdacdf1becc1398f3a35b
RBC	contracts/upgradeable_contracts/RewardableBridge.sol	54963c9f7d6a8f4377e34aa4e8dcff96acad0e33bc95989535ac3f1f7df46948
IBC	contracts/upgradeable_contracts/InitializableBridge.sol	1121c87b80b43ffa345fa609bc407b9d920a21d455859c741e2967471d629af9
SCK	contracts/upgradeable_contracts/Sacrifice.sol	286a2966e5bd4377f5f5130afb2dc1602cefb388c9b7f4690a8c717abe99bfb8
FCT	contracts/FlyingCashToken.sol	ebaec48f2d4654784b3b718bebe3b4851101f9406ddd088fd811891dda2acf5c
ERM	contracts/ERC677MultiBridgeToken.sol	bd42a25010d46d4db719bde2a9e0a58d093804b744b86bfc251a1f84064a647e
OSB	contracts/upgradeable_contracts/OtherSideBridgeStorage.sol	7abe690fb824a9fda8dd7b12efc9cbac0ed64cc6c0ca56f22577c0b80ba14e8c
VFM	contracts/upgradeable_contracts/ValidatorsFeeManager.sol	53b862c390b9f6802a04dd52658448f0bd524ac6011cf07f382b92993d2c28cc
FAM	contracts/upgradeable_contracts/amb_native_to_erc20/ForeignAMBNativeToErc20.sol	a6fbdac97ba1ef075a4422a951113a85672cba71bc84890c370273ded8f745af
ICK	contracts/upgradeable_contracts/Initializable.sol	973a28a487bbb142ff0a35c06ae261815329113500738332523af69ce8719471

ID	File	SHA256 Checksum
FMN	contracts/FeeManagerNoAsset.sol	ca42b9b430190513cc18ff73fddf5c4dcaf3451900995bdbafe70b48de9b9b07
CRH	contracts/upgradeable_contracts/ChooseReceiverHelper.sol	aa285b4a017aa00e195e0948495850c9cec9df8cb014c4422ed81901b492f940
FTC	contracts/upgradeable_contracts/FeeTypes.sol	adea3fdef59299bcb3da4dc750f7eb5de99179f02faba416426c9f0437abb70a
BFC	contracts/BaseFlyingCash.sol	4c5f26adf4092e9395f1e0c06e6f18a267dde4b320925218169bb63936d6ed64
BFM	contracts/upgradeable_contracts/BaseFeeManager.sol	0df1671c6c07a2e3eb107d68aa71b98f41a9dd3716db9a0555abb1b74ff3f140
BFA	contracts/upgradeable_contracts/arbitrary_message/BasicForeignAMB.sol	7db11d58c609d26496d969f2cdca4f5968e862f25f8f079a762e8a45b0e3946f
MTB	contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/MultiTokenBridgeMediator.sol	389bdf78a4ec4294357d7d28a0b7de61b416b0a31cd30871b0e93fab76474c79
MDC	contracts/upgradeable_contracts/arbitrary_message/MessageDelivery.sol	5612c73d90431dbf29e10f38e6b432557a8b926f1f5124fdd6aa2e68ff1abc9b
GIC	contracts/GovernableInitiable.sol	5e666d50f7f5fd05943bb0d8221d400b3772c32e7d04e561d7e1a3180e80d689
CCK	contracts/upgradeable_contracts/Claimable.sol	c3d71c18e4a1873eb4af872b825f867dcbacae7c0d073a90e28a590960aa512b
BHB	contracts/upgradeable_contracts/BasicHomeBridge.sol	9a4c68322877d28f5939625a5f5c2e86aa26fb983967680b19e70e5bc4faaeb2
TBC	contracts/TokenBridge.sol	51dabc026dbb1fd45ffac472ec4b8472997dad52fdd2aab5ac41977f11c71d38
FMC	contracts/FeeManager.sol	8778eac12befe620748e77bf90bc962c1c13aebb9d10148364e671fd0629300b
BFB	contracts/upgradeable_contracts/BasicForeignBridge.sol	3fce3378e41e78cdfa78ca1d6f4783191bb61ec4ce1bae61d4b8f379c9d4fdda
VCP	contracts/upgradeable_contracts/Validatable.sol	155b308fcc0d6d5f12ce0fd85e26b19736c34e29cce670fe619f07fb6bdab1ad

ID	File	SHA256 Checksum
BAB	contracts/upgradeable_contracts/arbitrary_message/BasicAMB.sol	2610dba4cd24b4d11dc1af1c1b5c308f76eecbe036a396eb0a5508545f836425
TBM	contracts/upgradeable_contracts/TokenBridgeMediator.sol	7093097da9c152c428d457057c3fbf94524798737d96081d55c13184bd96b8e7

Findings



Critical	**3** (8.11%)
Major	**5** (13.51%)
Medium	**6** (16.22%)
Minor	**13** (35.14%)
Informational	**10** (27.03%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
FlyingCash-01	Centralization Risks In `flyingcash`	**Centralization / Privilege**	**Major**	Acknowledged
FlyingCash-02	Centralization Risk In `tokbridge`	**Centralization / Privilege**	**Major**	Acknowledged
FlyingCash-03	External Dependency In `flyingcash/contracts`	Volatile Code	Minor	Acknowledged
FlyingCash-04	Third Party Dependencies	Volatile Code	Minor	Acknowledged
FlyingCash-05	Unlocked Compiler Version In Project `flyingcash`	Language Specific	Informational	Acknowledged
BAB-01	Inconsistent Function Naming And Functionality	Coding Style	Informational	Resolved
BFM-01	No Upper Limit For FeeRate	Volatile Code	Minor	Acknowledged
BMF-01	Inconsistent `require()` Condition	Logical Issue	Medium	Acknowledged
BMF-02	Potential Duplicate Accounts In Passed In `_rewardAccountList`	Volatile Code	Medium	Acknowledged
BMF-03	`distributeFee()` Will `revert`	Logical Issue	Minor	Acknowledged
BOC-01	Incorrect Exception Handling	Logical Issue	Minor	Resolved
CKP-01	Privileged Ownership	**Centralization / Privilege**	**Major**	Resolved

ID	Title	Category	Severity	Status
CKP-02	Missing Emit Events	Coding Style	Minor	Resolved
CKP-03	Lack Of Reasonable Fee Limitation	Logical Issue	Minor	Resolved
CKP-04	Unnecessary Inheritance In `flyingcash/contracts`	Logical Issue	Informational	Resolved
CON-01	Missing Emit Events	Coding Style	Minor	Acknowledged
CON-02	Pull-Over-Push Pattern	Logical Issue	Minor	Acknowledged
CON-03	Unused State Variable	Gas Optimization	Informational	Acknowledged
CON-04	Missing Error Messages	Coding Style	Informational	Acknowledged
ERT-01	Incorrect ERC677 Implementation	Logical Issue	Medium	Acknowledged
ESP-01	Unnecessary Inheritance In `tokbridge/contracts`	Logical Issue	Informational	Acknowledged
FCC-01	Potential Rug-pull Privileged	**Centralization / Privilege**	**Critical**	Resolved
FCC-02	Potential Revert	Logical Issue	Major	Resolved
FCC-03	Incorrect Event Emit	Language Specific	Medium	Resolved
FCC-04	Lack Of `return` Statement	Volatile Code	Minor	Resolved
FCC-05	Finance Model	Volatile Code	Informational	Acknowledged
FCN-01	Potential Over Mint	Logical Issue	Critical	Resolved
FMN-01	Missing Access Control	Logical Issue	Critical	Resolved
PCK-01	Locked Ether	Language Specific	Medium	Acknowledged
PRO-01	Redundant Statement	Volatile Code	Informational	Resolved
TBC-01	Permission Check And Unknown Implementation Of Interface	Volatile Code	Major	Acknowledged

ID	Title	Category	Severity	Status
UPG-01	Unchecked Low-level Call	Control Flow	Medium	Acknowledged
UPG-02	Missing Input Validation For Type Of Fee	Volatile Code	Minor	Acknowledged
UPG-03	Risk For Weak Randomness	Logical Issue	Minor	Acknowledged
UPG-04	External Dependency In `tokbridge/contracts`	Volatile Code	Minor	Acknowledged
UPG-05	Dead Code | Redundant Code In `tokbridge/contracts`	Volatile Code	Informational	Acknowledged
UPG-06	Unimplemented Function	Compiler Error	Informational	Acknowledged

FlyingCash-01 | Centralization Risks In `flyingcash`

Category	Severity	Location	Status
Centralization / Privilege	Major		Acknowledged

Description

In the contract `BaseFlyingCash` the role `Governance` has authority over the functions listed below.

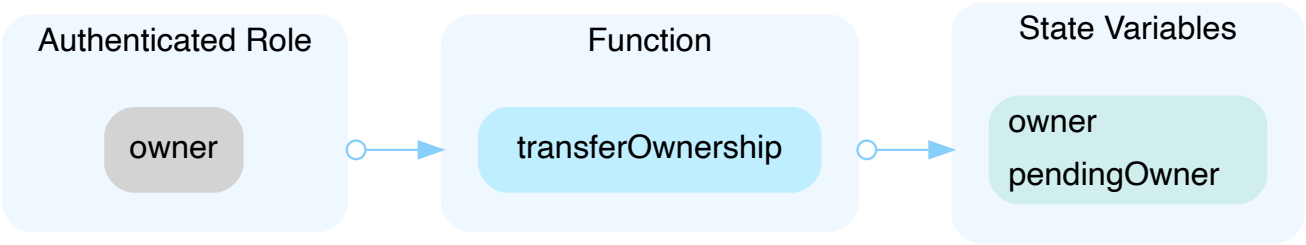
- function `setLockToken()`
- function `setVoucher()`
- function `setFeeManager()`
- function `setAdapter()`
- function `setNetworkBridge()`
- function `setAcceptVouchers()`
- function `pause()`
- function `unpause()`
- function `applyWithdraw()`

Any compromise to the `Governance` account may allow the hacker to take advantage of this authority.

Role `Governance` means account stored in the storage at slot `_GAVERNANCE_SLOT`

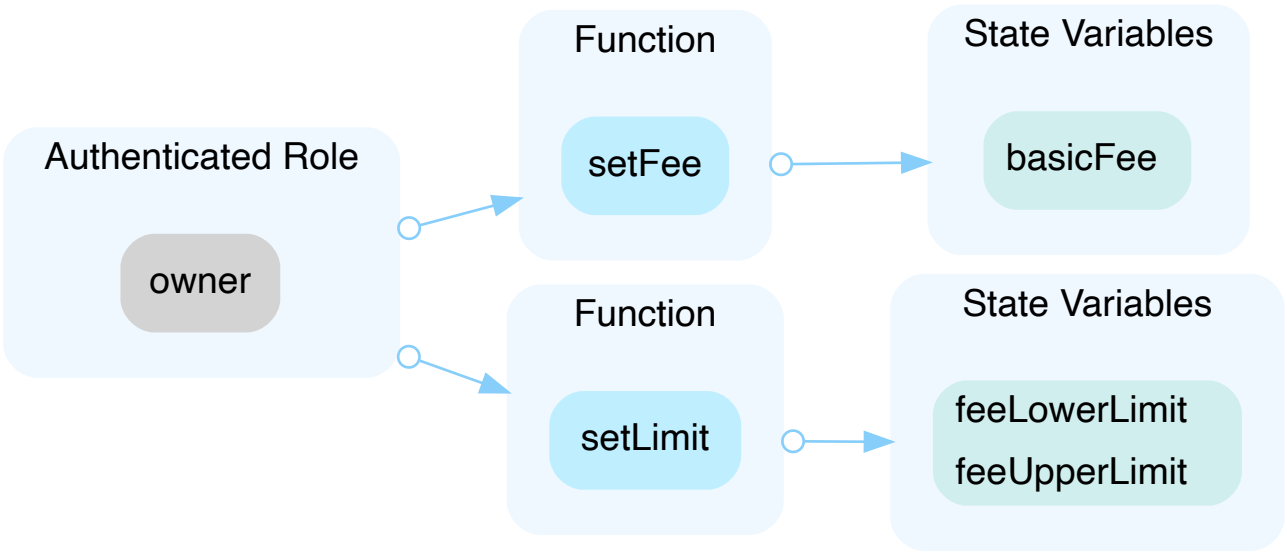
In the contract `BoringOwnable` the role `owner` has authority over the functions shown in the diagram below.

Any compromise to the `owner` account may allow the hacker to take advantage of this authority.



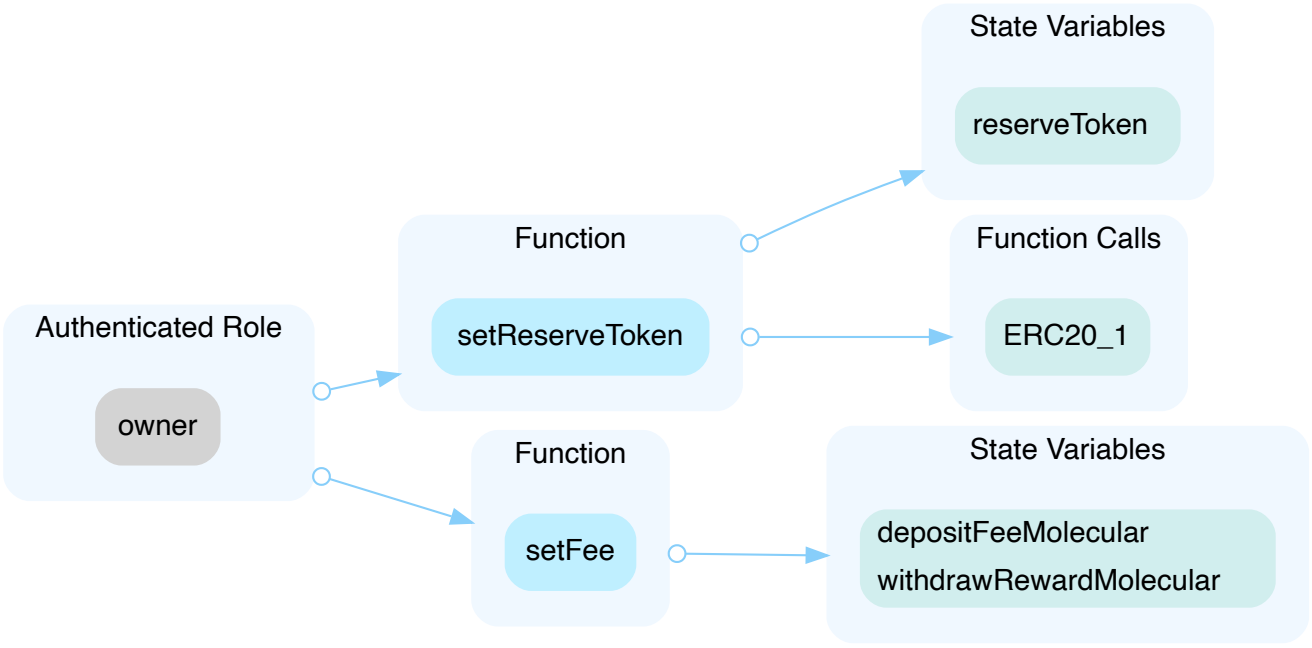
In the contract `FeeManager` the role `owner` has authority over the functions shown in the diagram below.

Any compromise to the `owner` account may allow the hacker to take advantage of this authority。



In the contract `FeeManagerNoAsset` 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.



In the contract `FlyingCash` the role `` has authority over the functions listed below.

Any compromise to the `` account may allow the hacker to take advantage of this authority.

In the contract `FlyingCashAdapterFilda` the role `Governance` has authority over the functions listed below.

- function `setWhitelist()`
- function `openLoan()`
- function `setLiquity()`
- function `claimComp()`

Any compromise to the `Governance` account may allow the hacker to take advantage of this authority.

Role `Governance` means account stored in the storage at slot `_GAVERNANCE_SLOT`

In the contract `FlyingCashAdapterNoAsset` the role `Governance` has authority over the functions listed below.

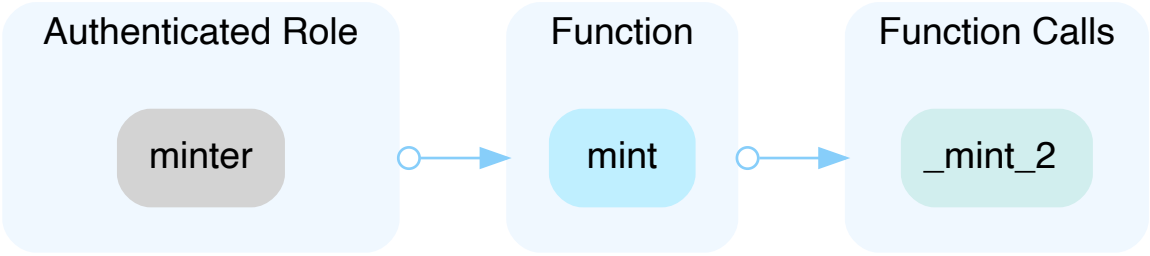
- function `setWhitelist()`
- function `withdraw()`

Any compromise to the `Governance` account may allow the hacker to take advantage of this authority.

Role `Governance` means account stored in the storage at slot `_GAVERNANCE_SLOT`

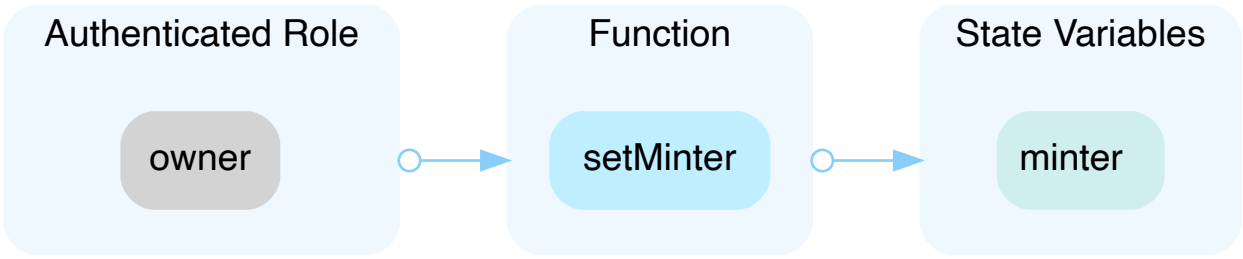
In the contract `FlyingCashToken` the role `minter` has authority over the functions shown in the diagram below.

Any compromise to the `minter` account may allow the hacker to take advantage of this authority.



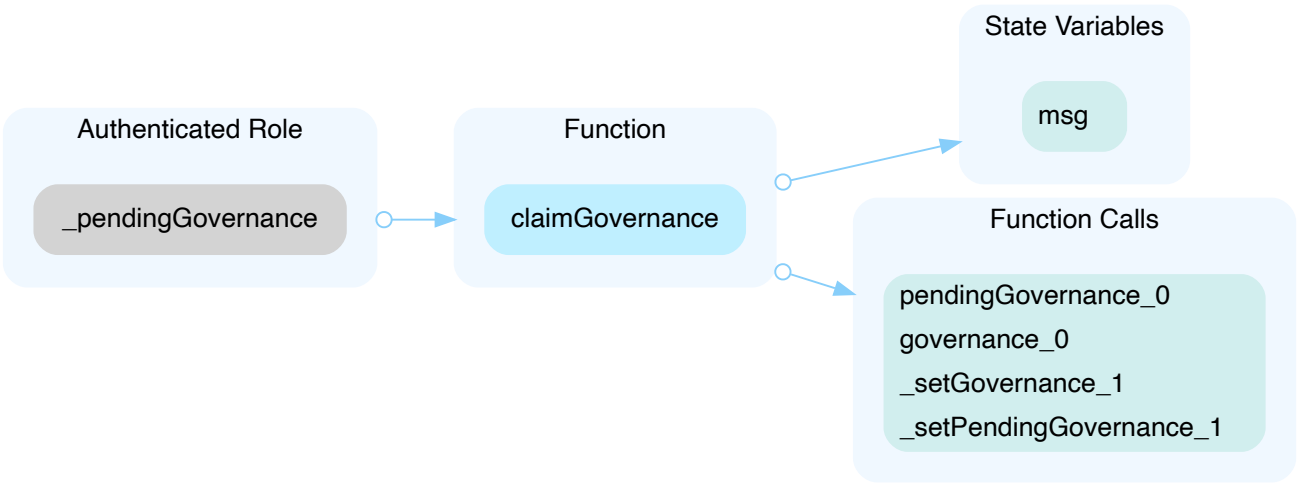
In the contract `FlyingCashToken` the role `owner` has authority over the functions shown in the diagram below.

Any compromise to the `owner` account may allow the hacker to take advantage of this authority.



In the contract `GovernableInitiable` the role `_pendingGovernance` has authority over the functions shown in the diagram below.

Any compromise to the `_pendingGovernance` account may allow the hacker to take advantage of this authority.



In the contract `GovernableInitiable` the role `Governance` has authority over the functions listed below.

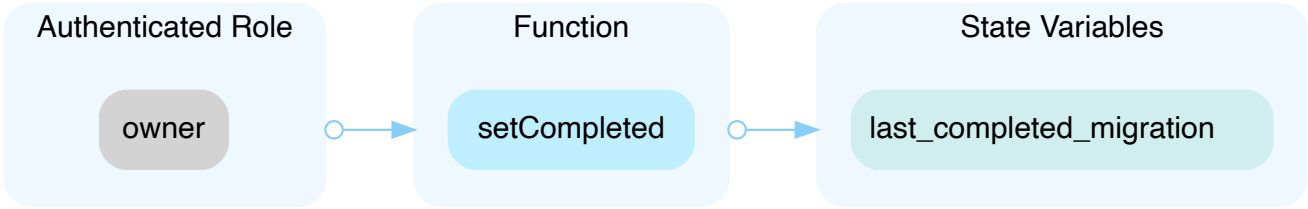
- function `setGovernance()`

Any compromise to the `Governance` account may allow the hacker to take advantage of this authority.

Role `Governance` means account stored in the storage at slot `_GAVERNANCE_SLOT`

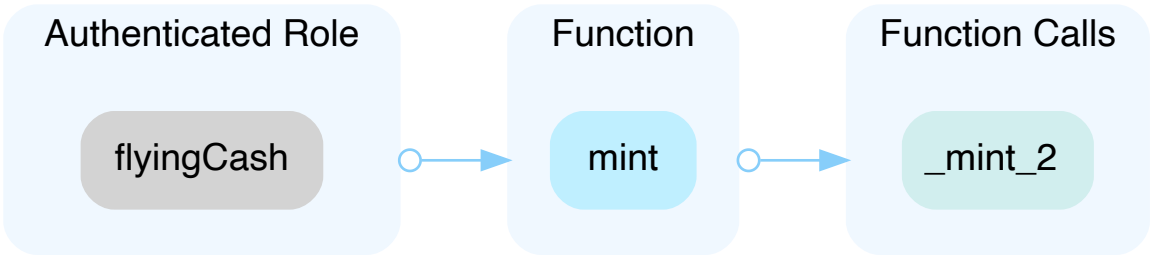
In the contract `Migrations` the role `owner` has authority over the functions shown in the diagram below.

Any compromise to the `owner` account may allow the hacker to take advantage of this authority.



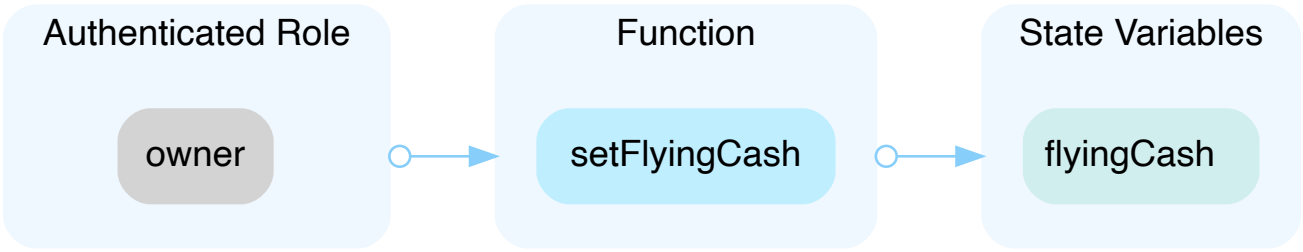
In the contract `Voucher` the role `flyingCash` has authority over the functions shown in the diagram below.

Any compromise to the `flyingCash` account may allow the hacker to take advantage of this authority.



In the contract `Voucher` 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.



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

[FlyingCash] : We have removed some functions that are controlled by centralized privileges or roles. We will use Time-lock and Multi sign when we deploy contracts.

FlyingCash-02 | Centralization Risk In `tokbridge`

Category	Severity	Location	Status
Centralization / Privilege	Major		Acknowledged

Description

1. In `tokbridge/contracts/upgradeability/`:

In the contract `OwnedUpgradeabilityProxy` the role `_upgradeabilityOwner` has authority over the functions listed below.

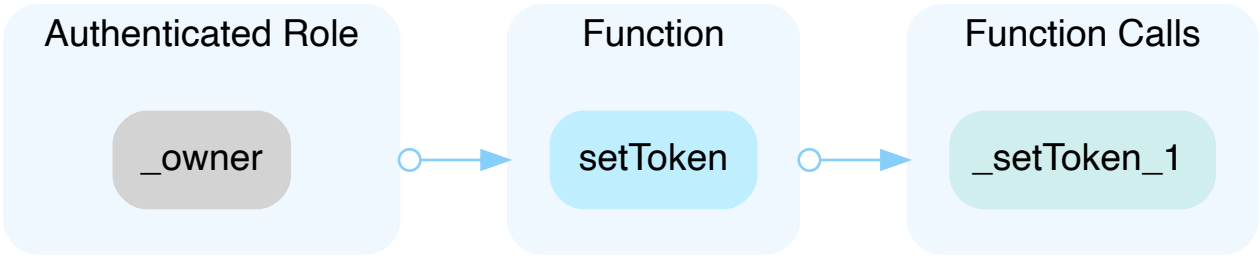
- funciton `transferProxyOwnership()`
- function `upgradeTo()`
- function `upgradeToAndCall()`

Any compromise to the `_upgradeabilityOwner` account may allow the hacker to take advantage of this authority.

2. In `tokbridge/contracts/upgradeable_contracts/amb_native_to_erc20/`:

In the contract `ForeignFeeManagerAMBNativeToErc20` 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.



In the contract `BasicAMBNativeToErc20` the role `_upgradeabilityOwner` has authority over the functions listed below.

- Function `claimTokens()`

Any compromise to the `_upgradeabilityOwner` account may allow the hacker to take advantage of this authority.

In the contract `ForeignAMBNativeToErc20` the role `_upgradeabilityOwner` has authority over the functions listed below.

- Function `claimTokensFromErc677()`

Any compromise to the `_upgradeabilityOwner` account may allow the hacker to take advantage of this authority.

In the contract `HomeAMBNativeToErc20` the role `_upgradeabilityOwner` has authority over the functions listed below.

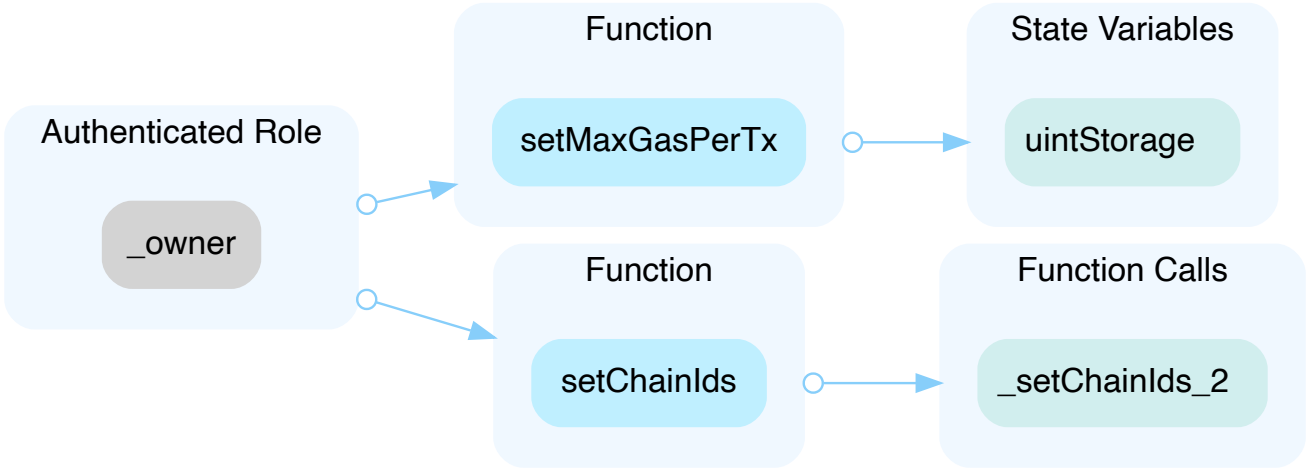
- Function `fixMediatorBalance()`

Any compromise to the `_upgradeabilityOwner` account may allow the hacker to take advantage of this authority.

3. In `tokbridge/contracts/upgradeable_contracts/arbitrary_message/`:

In the contract `BasicAMB` the role `addressStorage[OWNER]` has authority over the functions shown in the diagram below.

Any compromise to the `addressStorage[OWNER]` account may allow the hacker to take advantage of this authority.



In the contract `BasicHomeAMB` the role `Validator` has authority over the functions listed below.

- Function `executeAffirmation()`

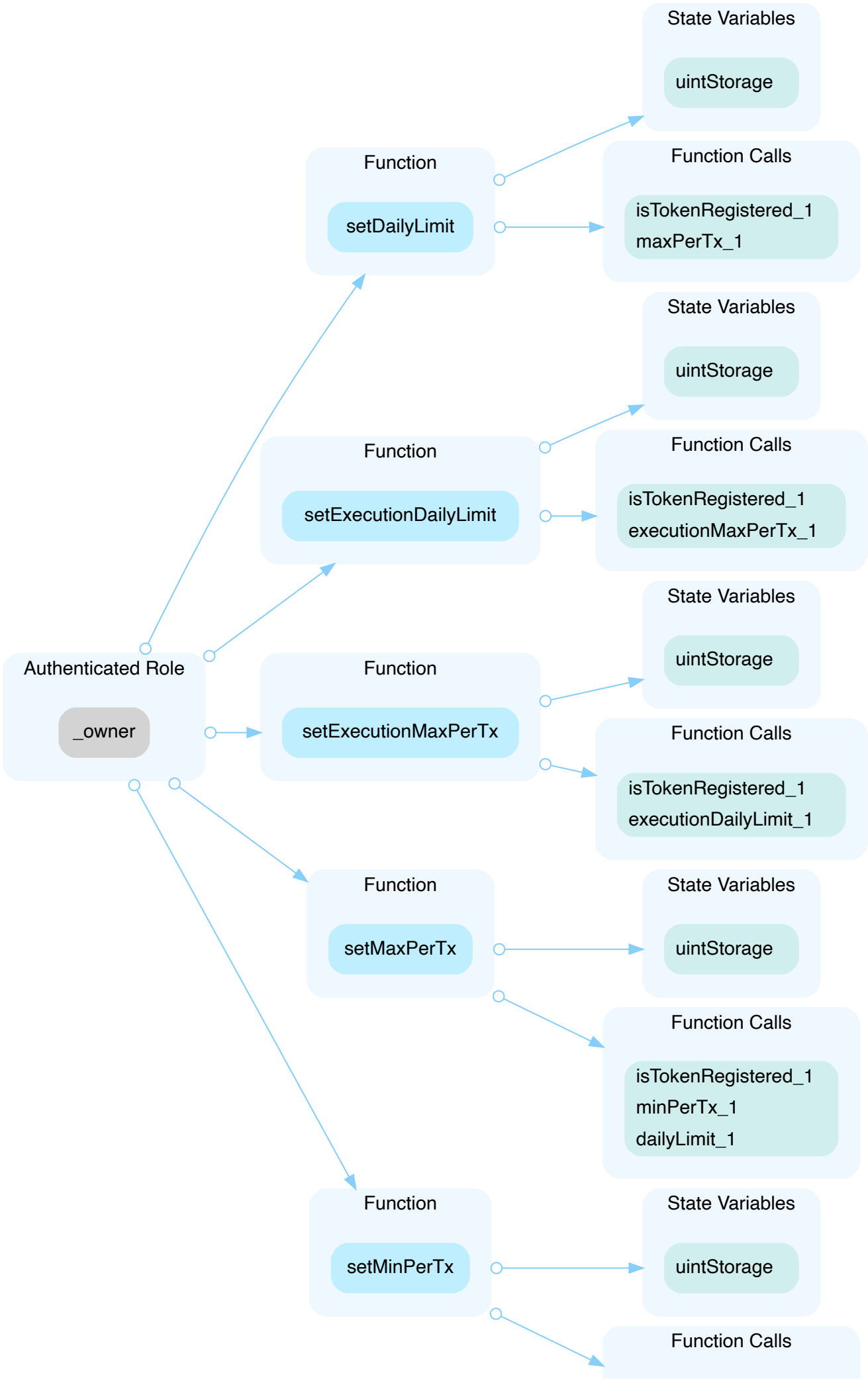
- Function `submitSignature()`

Any compromise to the `Validator` account may allow the hacker to take advantage of this authority.

4. In `tokbridge/contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/`:

In the contract `BasicMultiTokenBridge` the role `addressStorage[OWNER]` has authority over the functions shown in the diagram below.

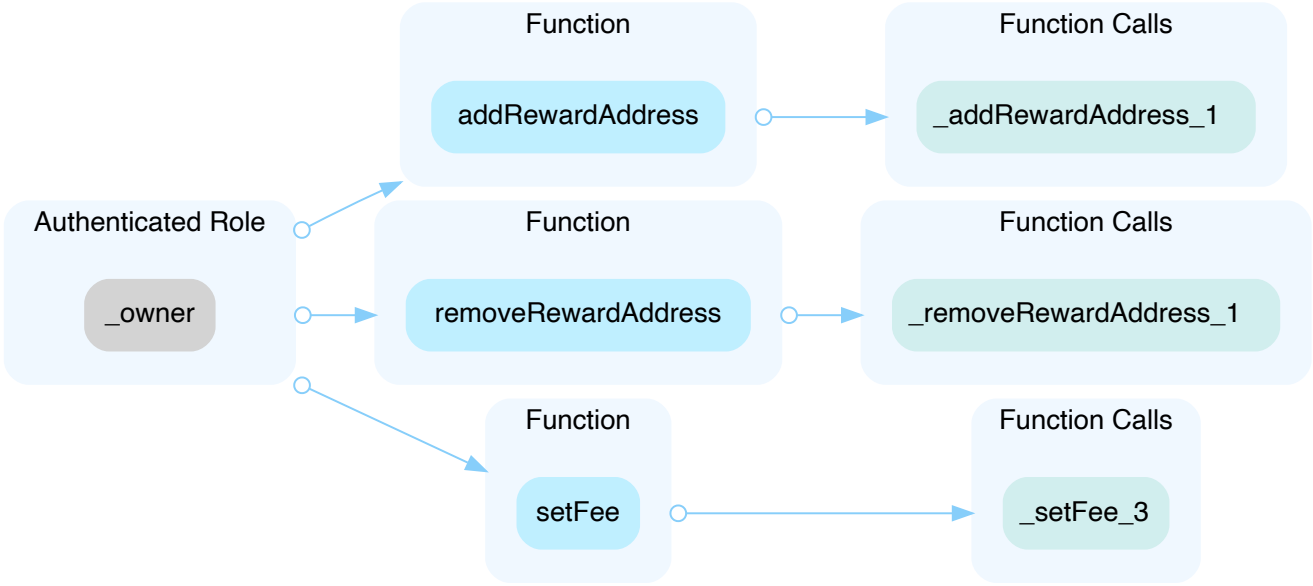
Any compromise to the `addressStorage[OWNER]` account may allow the hacker to take advantage of this authority.


State Variables
uintStorage
Function
setDailyLimit
Function Calls
isTokenRegistered_1
maxPerTx_1
State Variables
uintStorage
Function
setExecutionDailyLimit
Function Calls
isTokenRegistered_1
executionMaxPerTx_1
State Variables
uintStorage
Authenticated Role
_owner
Function
setExecutionMaxPerTx
Function Calls
isTokenRegistered_1
executionDailyLimit_1
Function
setMaxPerTx
State Variables
uintStorage
Function Calls
isTokenRegistered_1
minPerTx_1
dailyLimit_1
Function
setMinPerTx
State Variables
uintStorage
Function Calls

isTokenRegistered_1
dailyLimit_1
maxPerTx_1

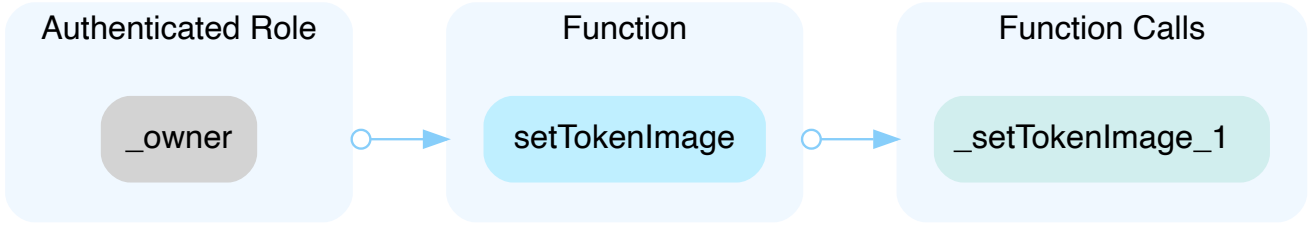
In the contract `HomeFeeManagerMultiAMBErc20ToErc677` the role `addressStorage[OWNER]` has authority over the functions shown in the diagram below.

Any compromise to the `addressStorage[OWNER]` account may allow the hacker to take advantage of this authority.



In the contract `HomeMultiAMBErc20ToErc677` the role `addressStorage[OWNER]` has authority over the functions shown in the diagram below.

Any compromise to the `addressStorage[OWNER]` account may allow the hacker to take advantage of this authority.



In the contract `HomeMultiAMBErc20ToErc677` the contract `bridgeContract` has authority over the functions listed below.

- Function `deployAndHandleBridgedTokens()`
- Function `handleBridgedTokens()`

Any compromise to the `bridgeContract` contract may allow the hacker to take advantage of this authority.

In the contract `ForeignMultiAMBErc20ToErc677` the role `_upgradeabilityOwner` has authority over the functions listed below.

- Function `fixMediatorBalance()`

Any compromise to the `_upgradeabilityOwner` account may allow the hacker to take advantage of this authority.

In the contract `ForeignMultiAMBErc20ToErc677` the contract `bridgeContract` has authority over the functions listed below.

- Function `handleBridgedTokens()`

Any compromise to the `bridgeContract` contract may allow the hacker to take advantage of this authority.

In the contract `BasicMultiAMBErc20ToErc677` the role `_upgradeabilityOwner` has authority over the functions listed below.

- Function `claimTokens()`

Any compromise to the `_upgradeabilityOwner` account may allow the hacker to take advantage of this authority.

In the contract `MultiTokenBridgeMediator` the contract `bridgeContract` has authority over the functions listed below.

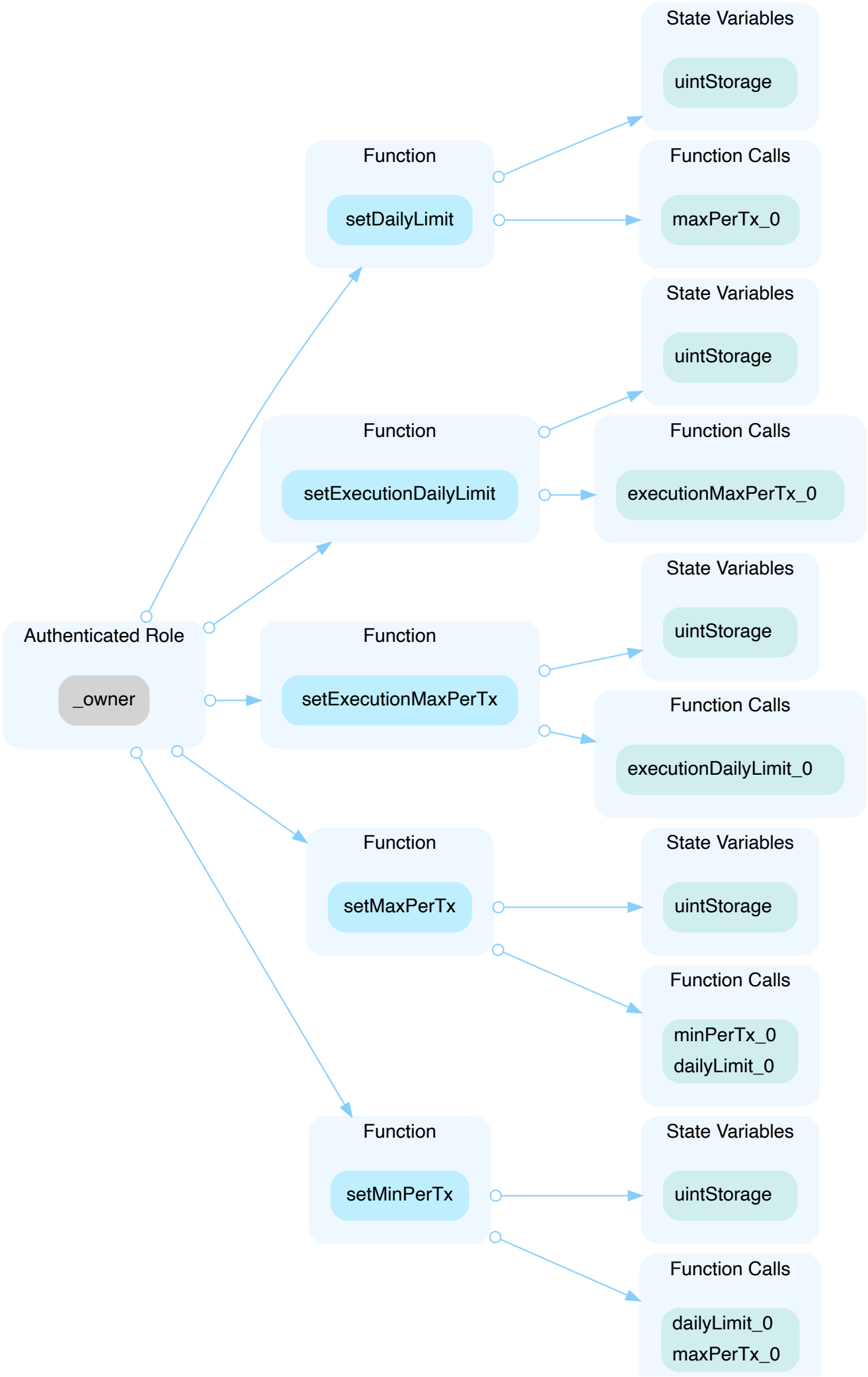
- Function `fixFailedMessage()`

Any compromise to the `bridgeContract` contract may allow the hacker to take advantage of this authority.

5. In `tokbridge/contracts/upgradeable_contracts/`:

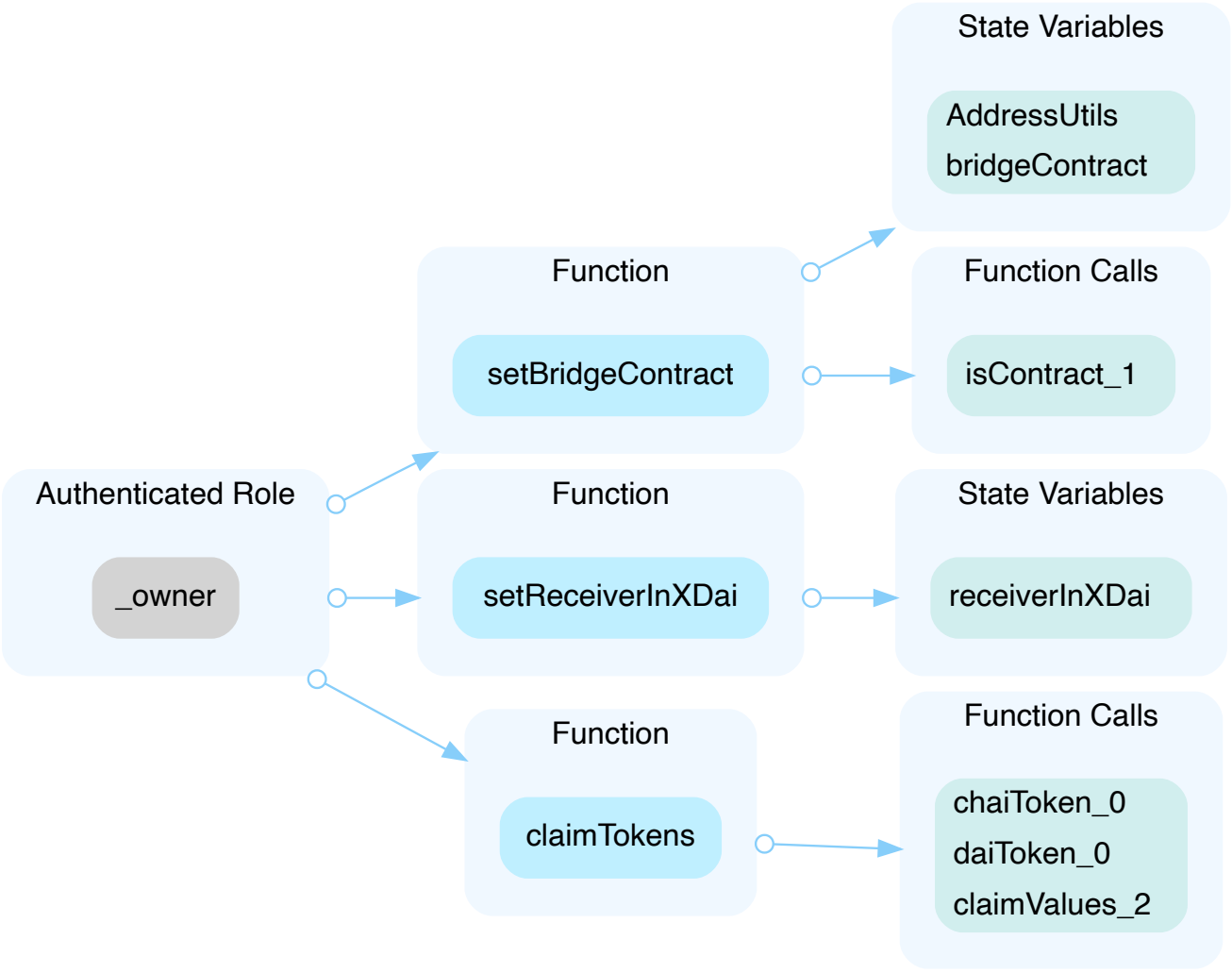
In the contract `BasicTokenBridge` the role `addressStorage[OWNER]` has authority over the functions shown in the diagram below.

Any compromise to the `addressStorage[OWNER]` account may allow the hacker to take advantage of this authority.


State Variables
uintStorage
Function
setDailyLimit
Function Calls
maxPerTx_0
State Variables
uintStorage
Function
setExecutionDailyLimit
Function Calls
executionMaxPerTx_0
State Variables
uintStorage
Authenticated Role
_owner
Function
setExecutionMaxPerTx
Function Calls
executionDailyLimit_0
Function
setMaxPerTx
State Variables
uintStorage
Function Calls
minPerTx_0
dailyLimit_0
Function
setMinPerTx
State Variables
uintStorage
Function Calls
dailyLimit_0
maxPerTx_0

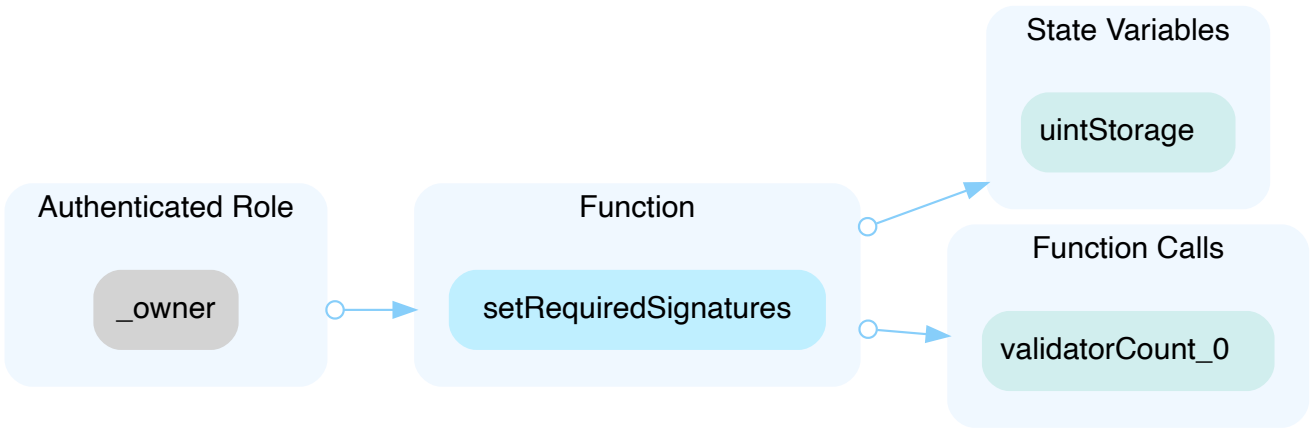
In the contract `InterestReceiver` the role `_owner` has authority over the functions shown in the diagram below.

Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



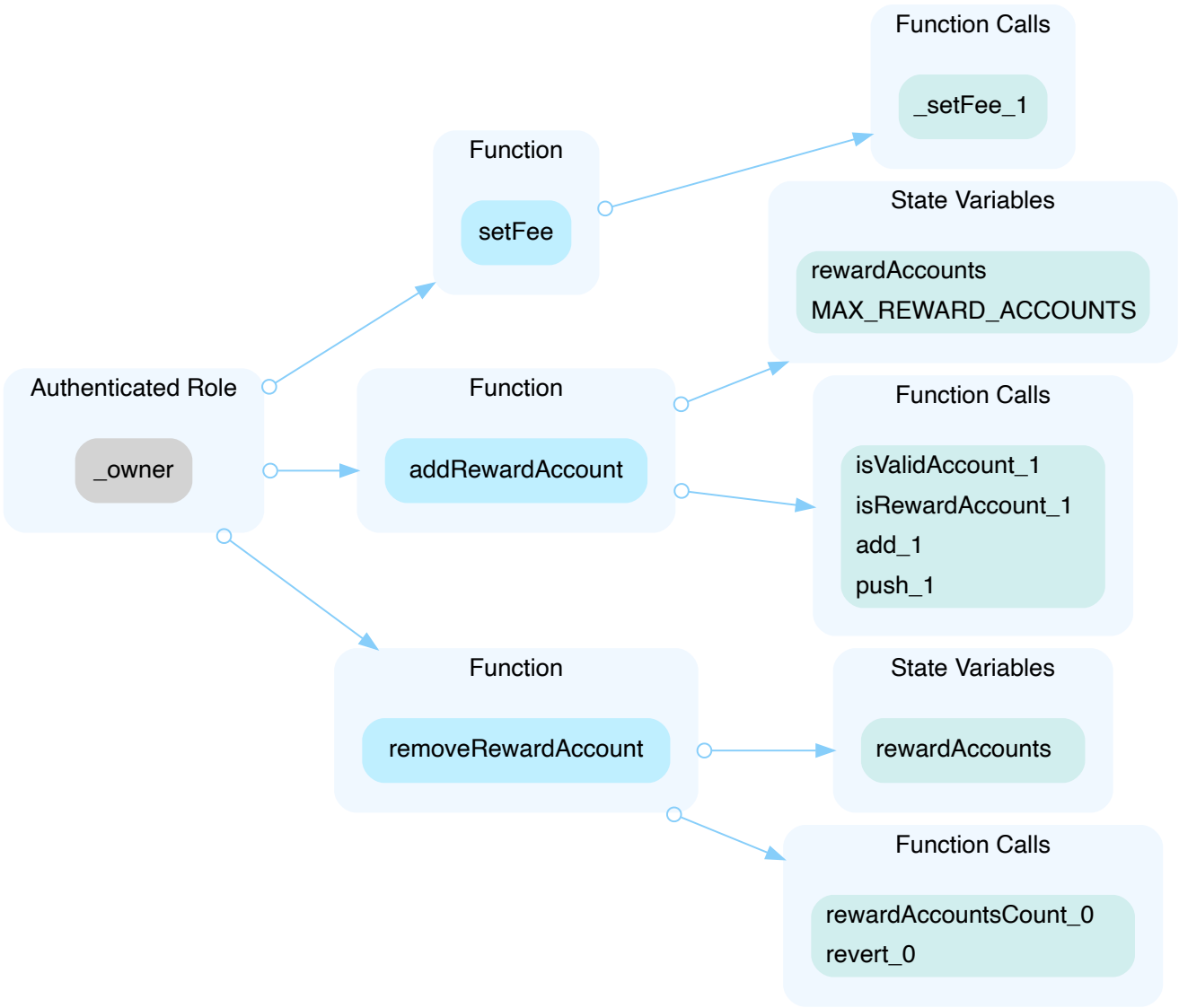
In the contract `BaseBridgeValidators` the role `addressStorage[OWNER]` has authority over the functions shown in the diagram below.

Any compromise to the `addressStorage[OWNER]` account may allow the hacker to take advantage of this authority.



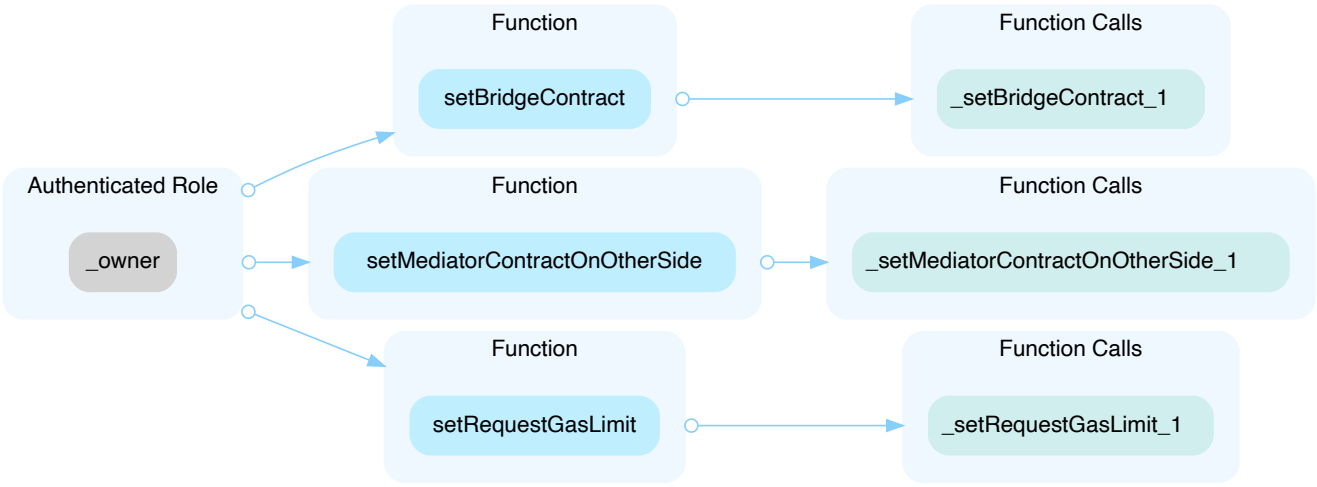
In the contract `BaseMediatorFeeManager` the role `_owner` has authority over the functions shown in the diagram below.

Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



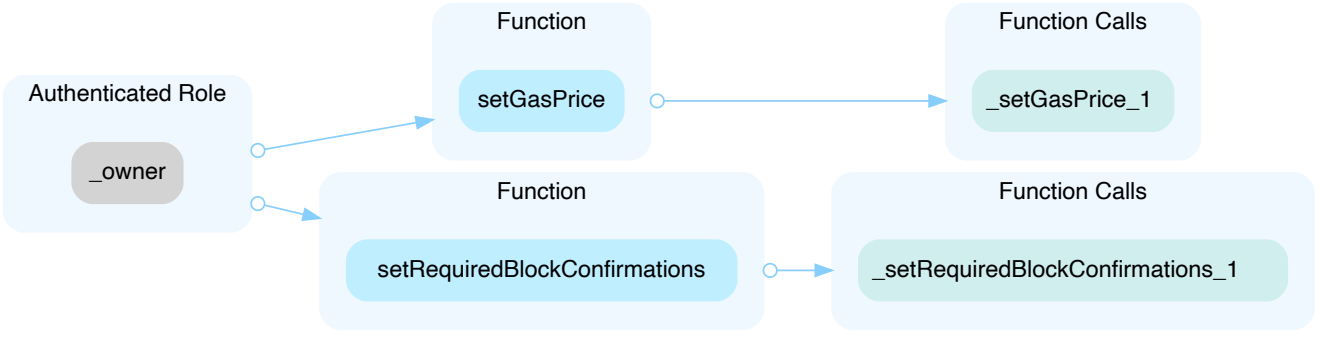
In the contract `BasicAMBMediator` the role `addressStorage[OWNER]` has authority over the functions shown in the diagram below.

Any compromise to the `addressStorage[OWNER]` account may allow the hacker to take advantage of this authority.



In the contract `BasicBridge` the role `addressStorage[OWNER]` has authority over the functions shown in the diagram below.

Any compromise to the `addressStorage[OWNER]` account may allow the hacker to take advantage of this authority.



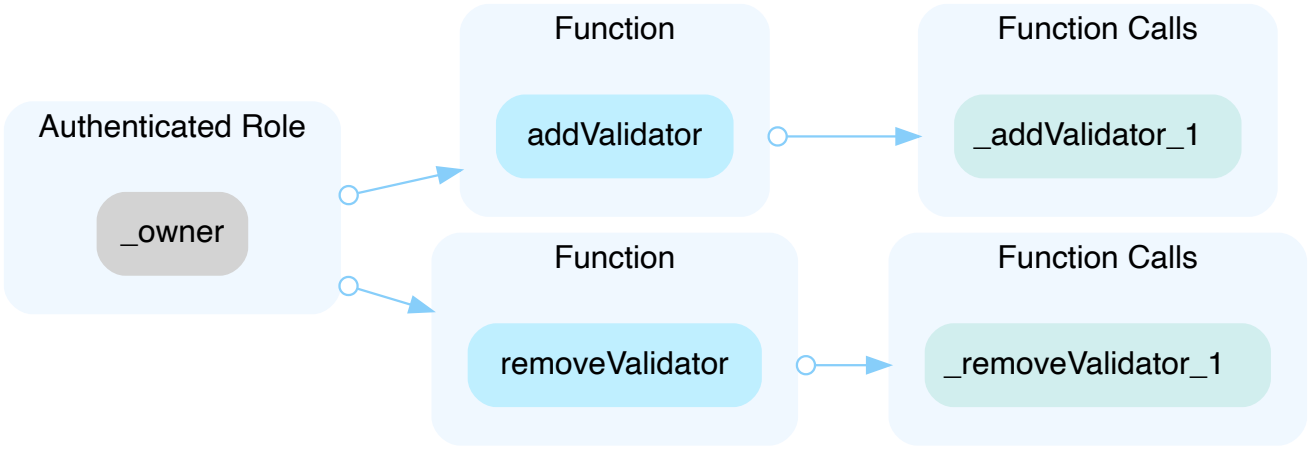
In the contract `BasicBridge` the role `_upgradeabilityOwner` has authority over the functions listed below.

- Function `claimTokens()`

Any compromise to the `_upgradeabilityOwner` account may allow the hacker to take advantage of this authority.

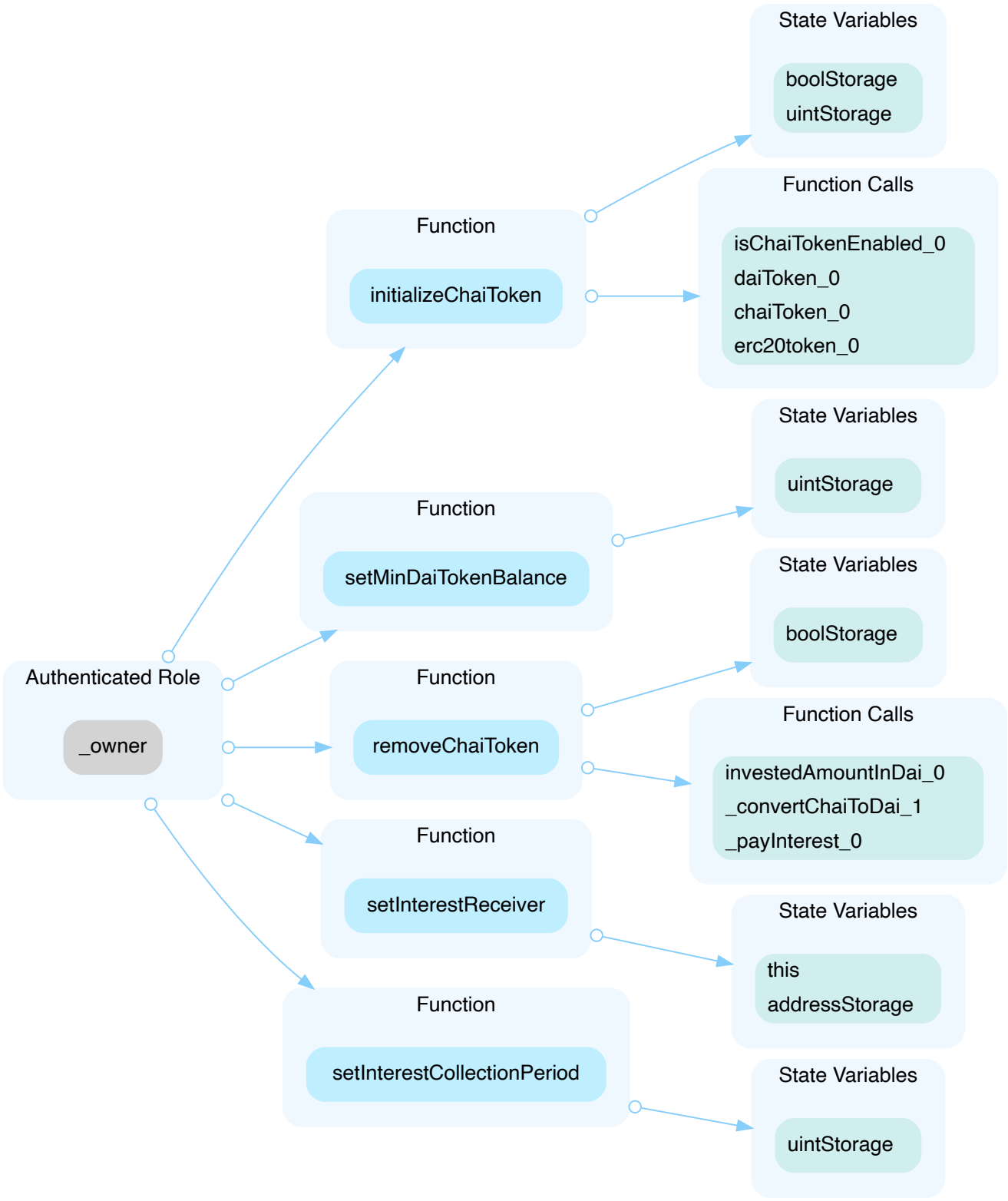
In the contract `BridgeValidators` the role `addressStorage[OWNER]` has authority over the functions shown in the diagram below.

Any compromise to the `addressStorage[OWNER]` account may allow the hacker to take advantage of this authority.



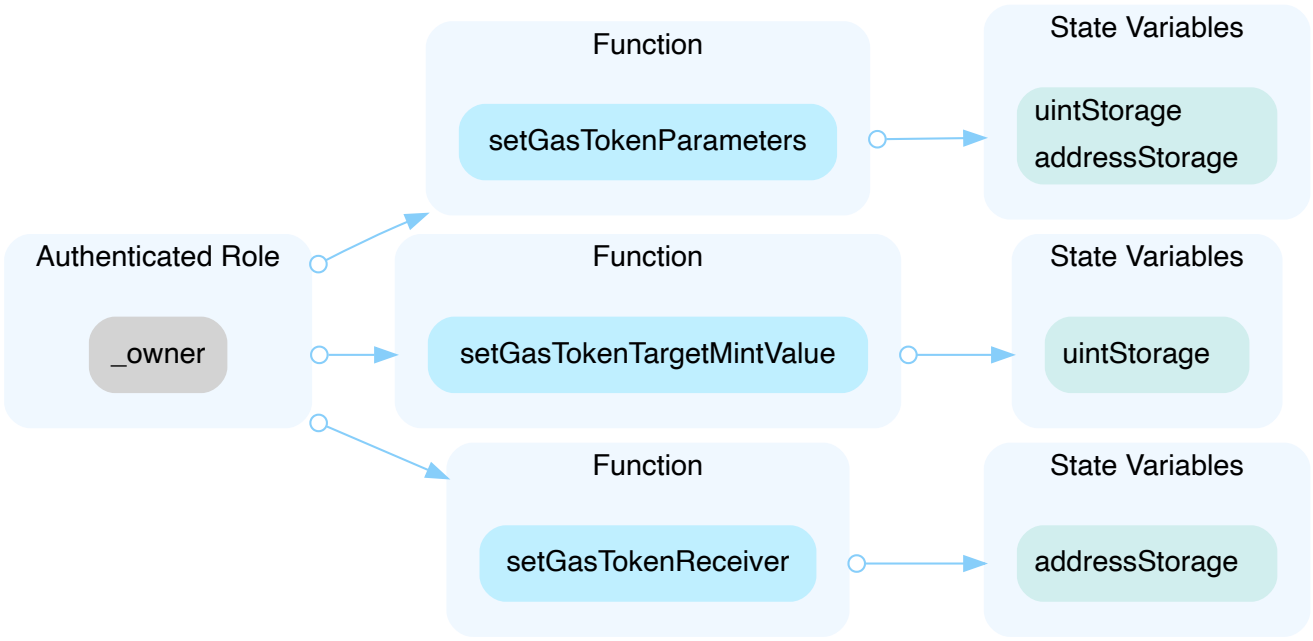
In the contract `ChaiConnector` the role `addressStorage[OWNER]` has authority over the functions shown in the diagram below.

Any compromise to the `addressStorage[OWNER]` account may allow the hacker to take advantage of this authority.



In the contract `GasTokenConnector` the role `addressStorage[OWNER]` has authority over the functions shown in the diagram below.

Any compromise to the `addressStorage[OWNER]` account may allow the hacker to take advantage of this authority.

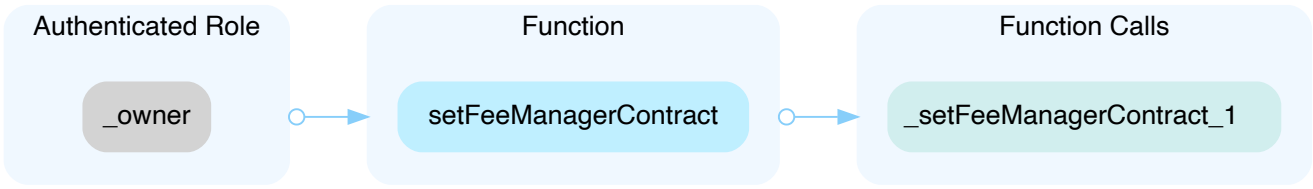


In the contract `RewardableBridge` the role `addressStorage[OWNER]` has authority over the functions shown in the diagram below.

Any compromise to the `addressStorage[OWNER]` account may allow the hacker to take advantage of this authority.

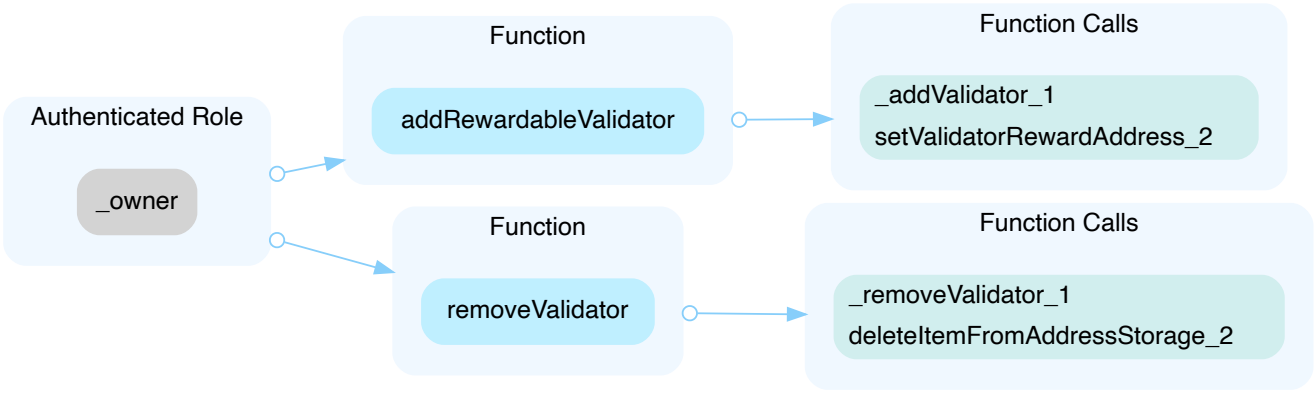
In the contract `RewardableMediator` the role `addressStorage[OWNER]` has authority over the functions shown in the diagram below.

Any compromise to the `addressStorage[OWNER]` account may allow the hacker to take advantage of this authority.



In the contract `RewardableValidators` the role `addressStorage[OWNER]` has authority over the functions shown in the diagram below.

Any compromise to the `addressStorage[OWNER]` account may allow the hacker to take advantage of this authority.



In the contract `OverdrawManagement` the role `_upgradeabilityOwner` has authority over the functions listed below.

- Function `fixAssetsAboveLimits()`

Any compromise to the `_upgradeabilityOwner` account may allow the hacker to take advantage of this authority.

In the contract `TokenBridgeMediator` the contract `bridgeContract` has authority over the functions listed below.

- Function `handleBridgedTokens()`
- Function `fixFailedMessage()`

Any compromise to the `bridgeContract` contract may allow the hacker to take advantage of this authority.

In the contract `Ownable` the contract `addressStorage[OWNER]` has authority over the functions listed below.

- Function `transferOwnership()`

Any compromise to the `addressStorage[OWNER]` contract may allow the hacker to take advantage of this authority.

In the contract `BasicHomeBridge` the role `Validator` has authority over the functions listed below.

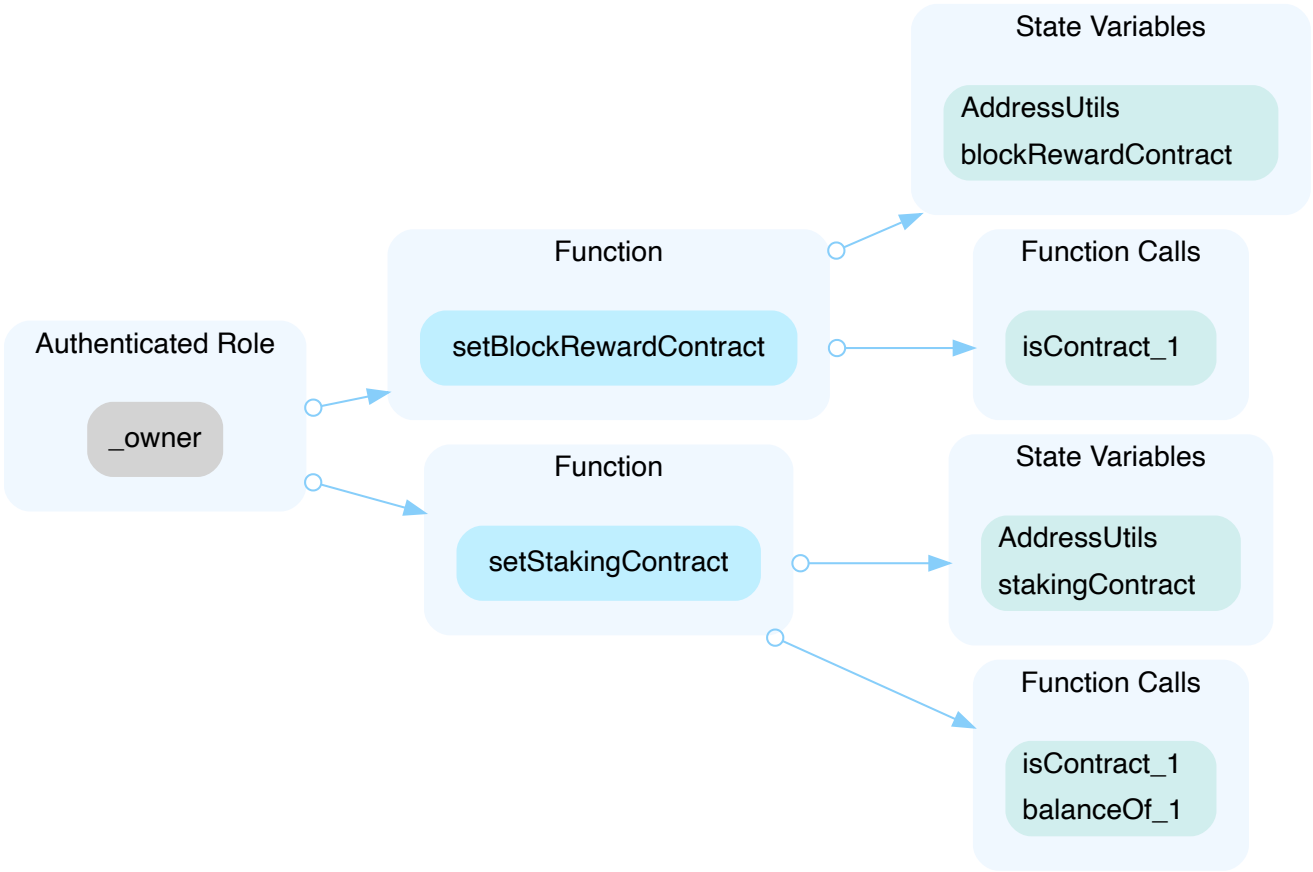
- Function `executeAffirmation()`
- Function `submitSignature()`

Any compromise to the `Validator` account may allow the hacker to take advantage of this authority.

6. In `tokbridge/contracts/`:

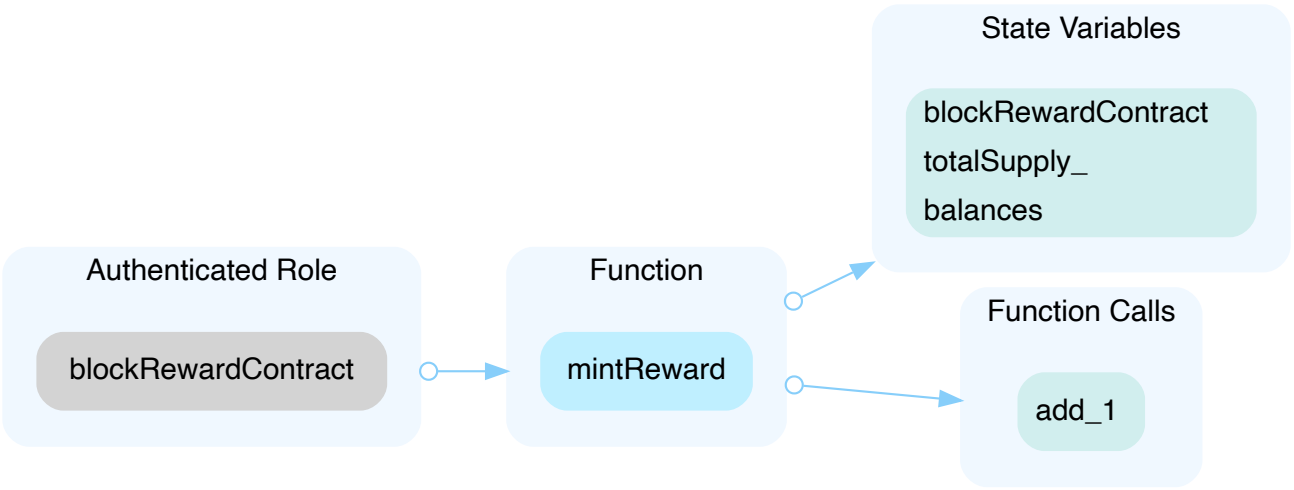
In the contract `ERC677BridgeTokenRewardable` the role `addressStorage[OWNER]` has authority over the functions shown in the diagram below.

Any compromise to the `addressStorage[OWNER]` account may allow the hacker to take advantage of this authority.



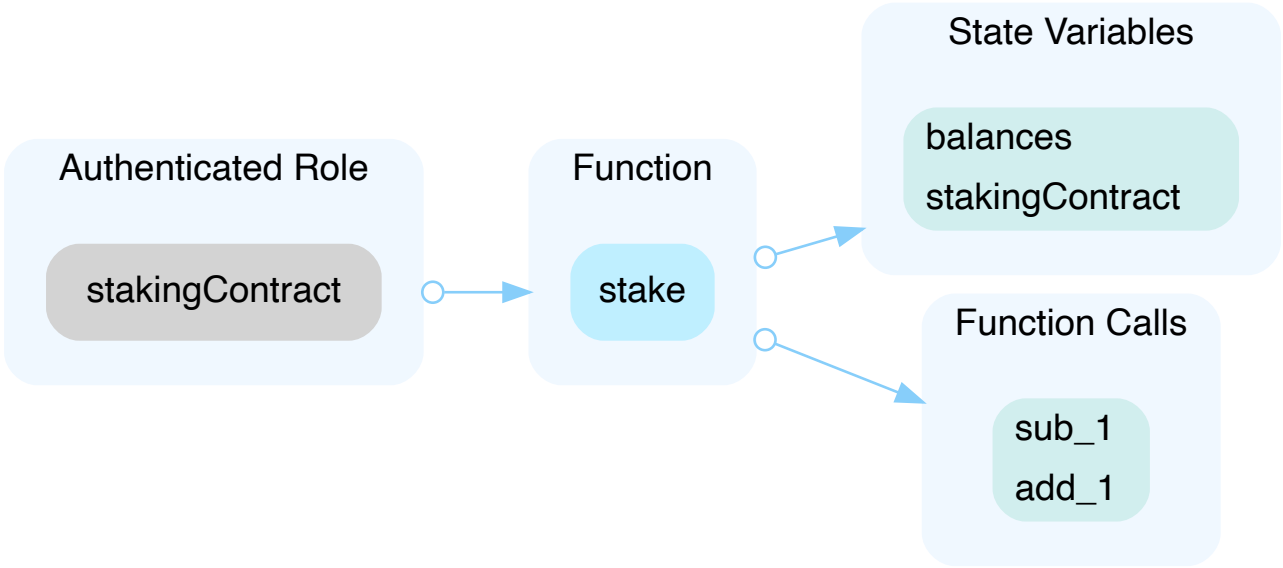
In the contract `ERC677BridgeTokenRewardable` the role `blockRewardContract` has authority over the functions shown in the diagram below.

Any compromise to the `blockRewardContract` account may allow the hacker to take advantage of this authority.



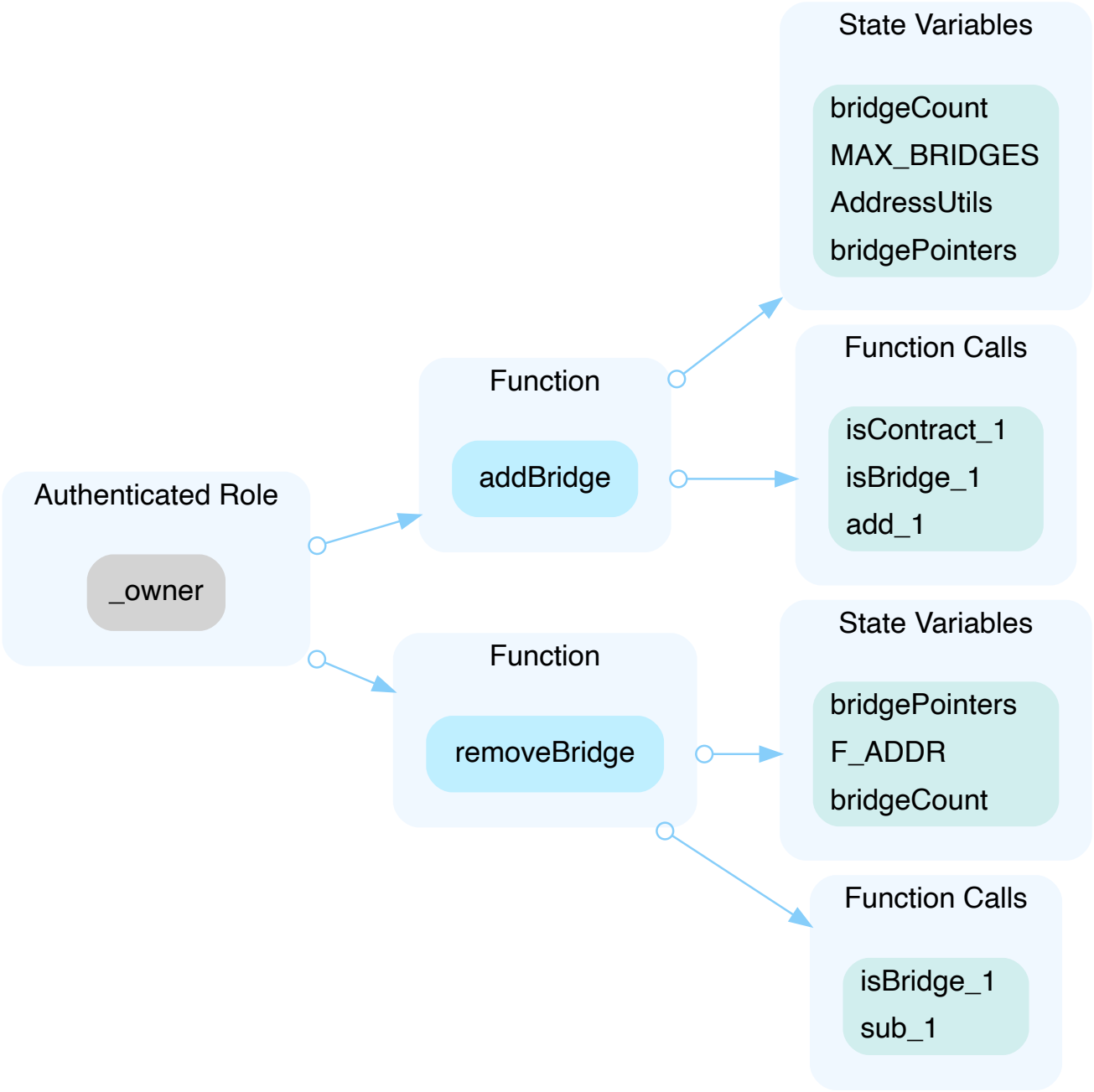
In the contract `ERC677BridgeTokenRewardable` the role `stakingContract` has authority over the functions shown in the diagram below.

Any compromise to the `stakingContract` account may allow the hacker to take advantage of this authority.



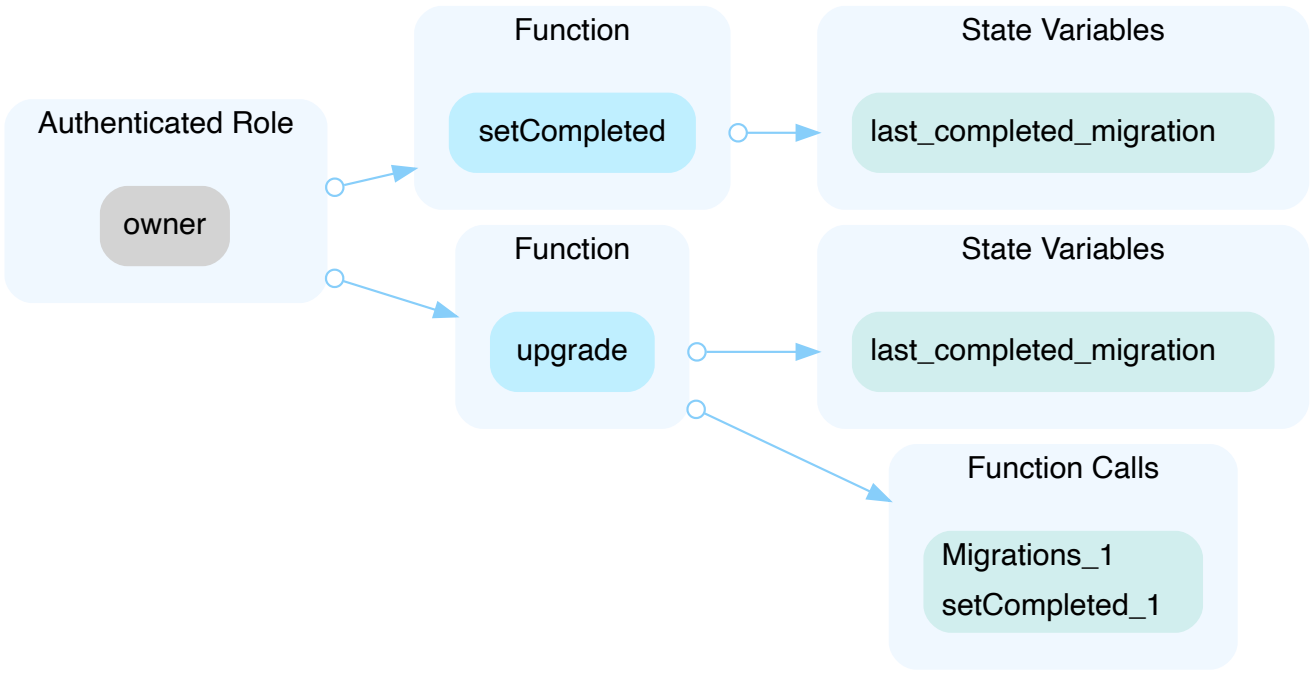
In the contract `ERC677MultiBridgeToken` the role `addressStorage[OWNER]` has authority over the functions shown in the diagram below.

Any compromise to the `addressStorage[OWNER]` account may allow the hacker to take advantage of this authority.



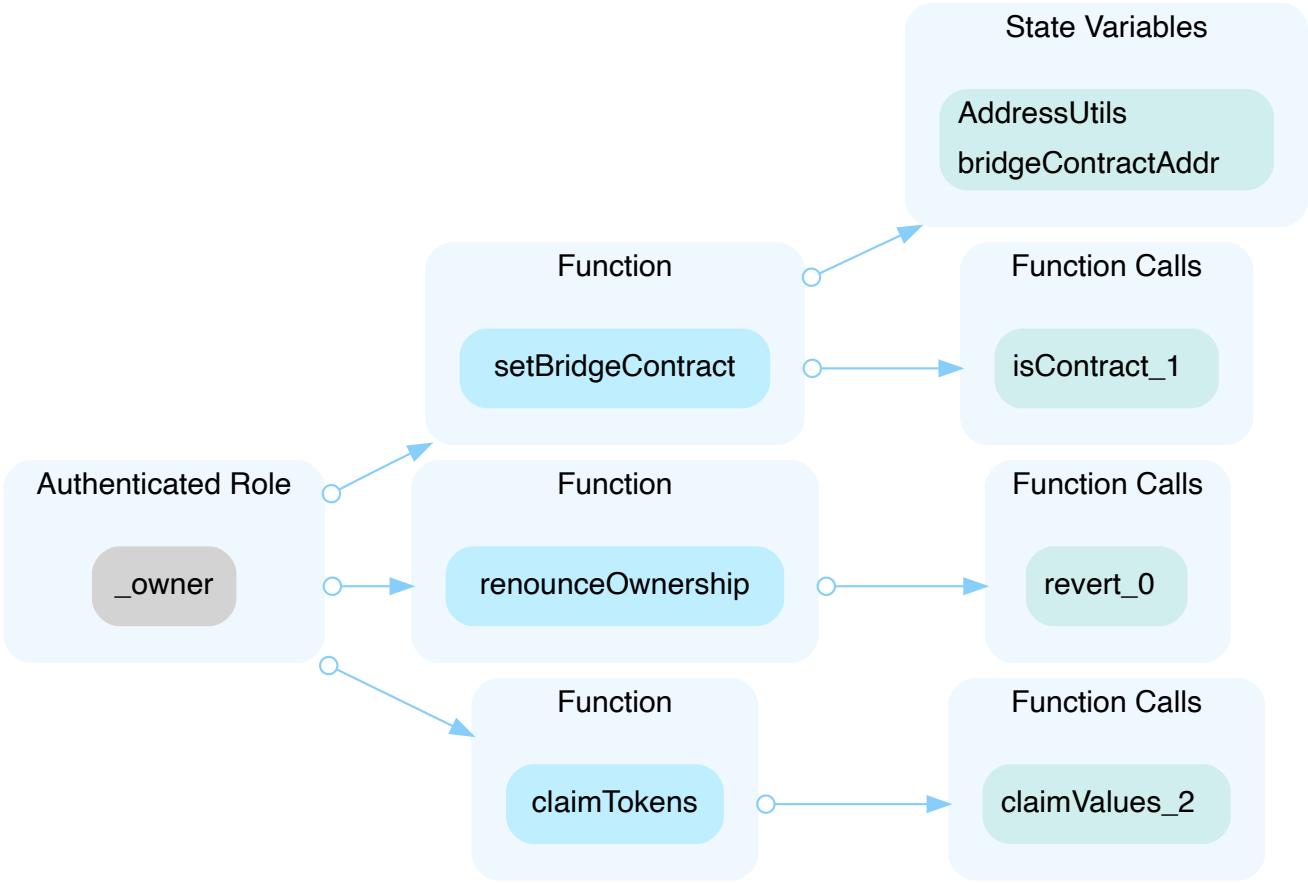
In the contract `Migrations` the role `owner` has authority over the functions shown in the diagram below.

Any compromise to the `owner` account may allow the hacker to take advantage of this authority.



In the contract `ERC677BridgeToken` the role `addressStorage[OWNER]` has authority over the functions shown in the diagram below.

Any compromise to the `addressStorage[OWNER]` account may allow the hacker to take advantage of this authority.



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

[JaxNetwork]: We are not prepared to modify the code. We will use `Timelock` and Multi sign when we deploy the contracts.

FlyingCash-03 | External Dependency In `flyingcash/contracts`

Category	Severity	Location	Status
Volatile Code	Minor		Acknowledged

Description

The implementations of following external contracts are unknown and not in the audit scope.

- In `BaseFlyingCash.sol`
  - `adapter`
  - `lockToken`
  - `feeManager`
- In `FeeManagerNoAsset.sol`
  - `reserveToken`

The scope of these audit would treat the external dependency entities as black boxes and assume functional correctness.

Recommendation

We recommend ensuring all of the external contracts are correct.

Alleviation

The FlyingCash team acknowledged this finding.

FlyingCash-04 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Minor		Acknowledged

Description

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

Recommendation

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

Alleviation

[FlyingCash]: We will constantly monitor the statuses of `compound` to mitigate the side effects when unexpected activities are observed.

FlyingCash-05 | Unlocked Compiler Version In Project `flyingcash`

Category	Severity	Location	Status
Language Specific	Informational		Acknowledged

Description

The contract has unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to different compiler versions. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.6.2` the contract should contain the following line:

```
pragma solidity 0.6.2;
```

Alleviation

The FlyingCash team acknowledged this finding.

BAB-01 | Inconsistent Function Naming And Functionality

Category	Severity	Location	Status
Coding Style	Informational	contracts/upgradeable_contracts/arbitrary_message/BasicAMB.sol (0312_1): 158~160	Resolved

Description

Function `sourceChainId` is executed in function `_isDestinationChainIdValid()`.

By function naming "_isDestinationChainIdValid", it seem function `destinationChainId()` should be executed here.

Recommendation

We recommend client to make sure whether the correct function is used here or not.

Alleviation

The FlyingCash team has confirmed that the function used here is correct.

BFM-01 | No Upper Limit For FeeRate

Category	Severity	Location	Status
Volatile Code	Minor	contracts/upgradeable_contracts/BaseFeeManager.sol (0312_1): 28	Acknowledged

Description

In the current implementation, there is no reasonable upper limit for the fee rate.

Recommendation

We recommend setting a reasonable upper limit of `FeeRate` such as 10**17(10%).

Alleviation

[FlyingCash]: No rate cap will not be set here, the rates are used for the balance of funds in the chain.

BMF-01 | Inconsistent `require()` Condition

Category	Severity	Location	Status
Logical Issue	Medium	contracts/upgradeable_contracts/BaseMediatorFeeManager.sol (0312_1): 37, 85	Acknowledged

Description

In `constructor()`, the maximum amount of reward account is `MAX_REWARD_ACCOUNTS` :

```
37    require(_rewardAccountList.length > 0 && _rewardAccountList.length <=
MAX_REWARD_ACCOUNTS);
```

But in the function `addRewardAccount`, the maximum amount of reward account will be `MAX_REWARD_ACCOUNTS - 1` :

```
85    require(rewardAccounts.length.add(1) < MAX_REWARD_ACCOUNTS);
86    rewardAccounts.push(_account);
```

Recommendation

We recommend client to unify the require condition used in `constructor()` and function `addRewardAccount`.

Alleviation

The FlyingCash team acknowledged this finding.

BMF-02 | Potential Duplicate Accounts In Passed In `_rewardAccountList`

Category	Severity	Location	Status
Volatile Code	Medium	contracts/upgradeable_contracts/BaseMediatorFeeManager.sol (0312_1): 36	Acknowledged

Description

The `constructor()` does not check if an address is duplicate in the passed in parameter `_rewardAccountList`. Some address will be able to receive additional reward.

It is not sure if this functionality should be allowed.

Recommendation

We recommend client to add a check if this issue is not allowed.

Alleviation

The FlyingCash team acknowledged this finding.

BMF-03 | `distributeFee()` Will `revert`

Category	Severity	Location	Status
Logical Issue	Minor	contracts/upgradeable_contracts/BaseMediatorFeeManager.sol (0312_1): 157	Acknowledged

Description

Function `distributeFee()` will revert if there is no reward account.

Recommendation

We recommend adding an `if` branch that if the there is no reward account, exit the function immediately.

Alleviation

The FlyingCash team acknowledged this finding.

BOC-01 | Incorrect Exception Handling

Category	Severity	Location	Status
Logical Issue	Minor	contracts/BoringOwnable.sol (0312_2): 28~40	Resolved

Description

- When `newOwner != address(0)` and `renounce` statements are both **true**, the role `owner` of `BoringOwnable` contract can't be actually renounced.
- Three independent and functionally different operations appear inside a single function at the same time. It is hard for code reviewing, debugging, or refactoring.

Recommendation

We recommend refactoring the linked codes as below:

```
21 function renounceOwnership() public onlyOwner {
22     emit OwnershipTransferred(owner, address(0));
23     owner = address(0);
24     pendingOwner = address(0);
25 }
26
27 function transferOwnership(address newOwner) public onlyOwner {
28     require(address(0) != newOwner, "pendingOwner set to the zero address.");
29     pendingOwner = newOwner;
30 }
31
32 function transferOwnershipDirectly(address newOwner) public onlyOwner {
33     require(address(0)!=newOwner,"not allowed to transfer owner to address(0)");
34     owner = newOwner;
35     emit OwnershipTransferred(owner, newOwner);
36     pendingOwner = address(0);
37 }
```

Alleviation

This finding has been resolved in commit 53b8c15f8ab3013a18375e82ed10976a45c79c6a.

CKP-01 | Privileged Ownership

Category	Severity	Location	Status
Centralization / Privilege	**Major**	contracts/FlyingCashAdapterFilda.sol (0312_2): 34~38; contracts/FlyingCashAdapterNoAsset.sol (0312_2): 22~26	Resolved

Description

Governance has the privilege to modify the `FlyingCashAdapterStorage.whitelist` at any time, which means any user is at risk of being banned from trading.

Recommendation

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

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

Short Term:

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

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

Long Term:

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

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

Permanent:

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

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

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

The FlyingCash team removed the function `setWhitelist()` and used the modifier `onlyFlyingCash` to verify the caller. This change was supplied in commit 301dbae0798bb6a2b77f76efc469c4cdafa744d0.

CKP-02 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Minor	contracts/BaseFlyingCash.sol (0312_2): 72~75, 94~103, 128; contracts/Voucher.sol (0312_2): 32~34; contracts/FlyingCashAdapterNoAsset.sol (0312_2): 28~32, 34~37; contracts/FeeManagerNoAsset.sol (0312_2): 26~29, 31~34; contracts/FeeManager.sol (0312_2): 22~24, 26~29; contracts/FlyingCashToken.sol (0312_2): 32~34	Resolved

Description

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

Recommendation

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

Alleviation

The FlyingCash team emitted events for the sensitive functions in commit 53b8c15f8ab3013a18375e82ed10976a45c79c6a.

CKP-03 | Lack Of Reasonable Fee Limitation

Category	Severity	Location	Status
Logical Issue	Minor	contracts/FeeManager.sol (0312_2): 23; contracts/FeeManagerNoAsset.sol (0312_2): 32, 33	Resolved

Description

The sate variables can be set up to **1000** which means transaction fee rate can be **100%**.

Recommendation

We recommend setting a reasonable upper limit for passed in parameters.

Alleviation

The FlyingCash team added a reasonable upper limit to the functions in commit 53b8c15f8ab3013a18375e82ed10976a45c79c6a. And the variables `basicFee`, `_depositeFee`, and `_withdrawReward` of this protocol are capped at 30%, and users should be aware of this rule before using it.

CKP-04 | Unnecessary Inheritance In `flyingcash/contracts`

Category	Severity	Location	Status
Logical Issue	Informational	contracts/FlyingCashAdapterNoAsset.sol (0312_2): 5; contracts/FeeManagerDefault.sol (0312_2): 5, 7	Resolved

Description

In `FeeManagerDefault.sol`

There is no need for contract `FeeManagerDefault` to inherit `BoringOwnable`, the former has nothing to do with the later.

In `FlyingCashAdapterNoAsset.sol`

There is no need for contract `FlyingCashAdapterNoAsset` to inherit `BoringOwnable`, the former has nothing to do with the later.

Recommendation

We recommend removing the unnecessary inheritance.

Alleviation

The FlyingCash team removed the unnecessary inheritance in commit 53b8c15f8ab3013a18375e82ed10976a45c79c6a.

CON-01 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Minor	contracts/ERC677BridgeTokenRewardable.sol (0312_1): 16~19, 21~25; contracts/upgradeable_contracts/BasicTokenBridge.sol (0312_1): 83~86, 88~91, 93~96; contracts/upgradeable_contracts/arbitrary_message/BasicAMB.sol (0312_1): 56~58, 81~83; contracts/upgradeable_contracts/BasicAMBMediator.sol (0312_1): 30~32, 38~40, 48~50; contracts/upgradeable_contracts/RewardableBridge.sol (0312_1): 59~62; contracts/upgradeable_contracts/BaseMediatorFeeManager.sol (0312_1): 69~71, 82~87, 96~108; contracts/upgradeable_contracts/GasTokenConnector.sol (0312_1): 20~23, 29~31, 37~39; contracts/upgradeable_contracts/BasicBridge.sol (0312_1): 31~33, 39~41; contracts/upgradeable_contracts/amb_native_to_erc20/ForeignFeeManagerAMBNativeToErc20.sol (0312_1): 34~36; contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/HomeFeeManagerMultiAMBErc20ToErc677.sol (0312_1): 48~50, 57~59, 68~70; contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/HomeMultiAMBErc20ToErc677.sol (0312_1): 69~71; contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/BasicMultiTokenBridge.sol (0312_1): 158~162, 170~174, 181~185; contracts/upgradeable_contracts/ChaiConnector.sol (0312_1): 64~70, 86~88, 101~105, 118~128, 148~150; contracts/upgradeable_contracts/RewardableMediator.sol (0312_1): 21~23; contracts/upgradeable_contracts/Ownable.sol (0312_1): 57~60; contracts/ERC677BridgeToken.sol (0312_1): 29~32, 96~98, 100~102	Acknowledged

Description

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

Recommendation

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

Alleviation

The FlyingCash team acknowledged this finding.

CON-02 | Pull-Over-Push Pattern

Category	Severity	Location	Status
Logical Issue	Minor	contracts/upgradeable_contracts/Ownable.sol (0312_1): 57~68; contracts/upgradeability/OwnedUpgradeabilityProxy.sol (0312_1): 38~42	Acknowledged

Description

The change of `_upgradeabilityOwner` by function `transferProxyOwnership()` overrides the previously set `_upgradeabilityOwner` with the new one without guaranteeing the new `_upgradeabilityOwner` is able to actuate transactions on-chain.

The same issue is happened in contract `upgradeable_contracts/Ownable.sol`.

Recommendation

We advise the pull-over-push pattern to be applied here whereby a new `_upgradeabilityOwner` is first proposed and consequently needs to accept the `_upgradeabilityOwner` status ensuring that the account can actuate transactions on-chain.

The following code snippet can be taken as a reference:

```
address public potentialUpgradeabilityOwner;

event UpgradeabilityOwnerNominated(address);

function transferProxyOwnership(address newOwner) external onlyUpgradeabilityOwner {
    require(newOwner != address(0), "potential upgradeabilityOwner can not be the zero address.");
    emit UpgradeabilityOwnerNominated(newOwner);
    potentialUpgradeabilityOwner = newOwner;
}

function acceptProxyOwnership() external {
    require(msg.sender == potentialUpgradeabilityOwner, 'You must be nominated as potential upgradeabilityOwner before you can accept owner role');
    emit ProxyOwnershipTransferred(upgradeabilityOwner(), potentialUpgradeabilityOwner);
    setUpgradeabilityOwner(potentialUpgradeabilityOwner);
    potentialUpgradeabilityOwner = address(0);
}
```

Alleviation

The FlyingCash team acknowledged this finding.

CON-03 | Unused State Variable

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/TokenProxy.sol (0312_1): 18, 19, 20, 22, 27, 28; contracts/upgradeability/EternalStorage.sol (0312_1): 9, 13	Acknowledged

Description

One or more state variables are never used in the codebase.

Variable `allowed` in `TokenProxy` is never used in `TokenProxy`.

File: projects/tokbridge/contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/TokenProxy.sol (Line 20, Contract `TokenProxy`)

```
    mapping(address => mapping(address => uint256)) internal allowed;
```

File: projects/tokbridge/contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/TokenProxy.sol (Line 13)

```
contract TokenProxy is Proxy {
```

Variable `balances` in `TokenProxy` is never used in `TokenProxy`.

File: projects/tokbridge/contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/TokenProxy.sol (Line 18, Contract `TokenProxy`)

```
    mapping(address => uint256) internal balances;
```

File: projects/tokbridge/contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/TokenProxy.sol (Line 13)

```
contract TokenProxy is Proxy {
```

Variable `expirations` in `TokenProxy` is never used in `TokenProxy`.

File: projects/tokbridge/contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/TokenProxy.sol (Line 28, Contract `TokenProxy`)

```
    mapping(address => mapping(address => uint256)) internal expirations;
```

File: projects/tokbridge/contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/TokenProxy.sol (Line 13)

```
contract TokenProxy is Proxy {
```

Variable `intStorage` in `EternalStorage` is never used in `BaseOverdrawManagement`.

File: projects/tokbridge/contracts/upgradeability/EternalStorage.sol (Line 13, Contract `EternalStorage`)

```
    mapping(bytes32 => int256) internal intStorage;
```

File: projects/tokbridge/contracts/upgradeable_contracts/BaseOverdrawManagement.sol (Line 5)

```
contract BaseOverdrawManagement is EternalStorage {
```

Variable `mintingFinished` in `TokenProxy` is never used in `TokenProxy`.

File: projects/tokbridge/contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/TokenProxy.sol (Line 22, Contract `TokenProxy`)

```
    bool internal mintingFinished;
```

File: projects/tokbridge/contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/TokenProxy.sol (Line 13)

```
contract TokenProxy is Proxy {
```

Variable `nonces` in `TokenProxy` is never used in `TokenProxy`.

File: projects/tokbridge/contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/TokenProxy.sol (Line 27, Contract `TokenProxy`)

```
    mapping(address => uint256) internal nonces;
```

File: projects/tokbridge/contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/TokenProxy.sol (Line 13)

```
contract TokenProxy is Proxy {
```

Variable `stringStorage` in `EternalStorage` is never used in `BaseOverdrawManagement`.

File: projects/tokbridge/contracts/upgradeability/EternalStorage.sol (Line 9, Contract `EternalStorage`)

```
    mapping(bytes32 => string) internal stringStorage;
```

File: projects/tokbridge/contracts/upgradeable_contracts/BaseOverdrawManagement.sol (Line 5)

```
contract BaseOverdrawManagement is EternalStorage {
```

Variable `totalSupply` in `TokenProxy` is never used in `TokenProxy`.

File: projects/tokbridge/contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/TokenProxy.sol (Line 19, Contract `TokenProxy`)

```
    uint256 internal totalSupply;
```

File: projects/tokbridge/contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/TokenProxy.sol (Line 13)

```
contract TokenProxy is Proxy {
```

Recommendation

We advise removing the unused variables.

Alleviation

The FlyingCash team acknowledged this finding.

CON-04 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	contracts/ERC677BridgeTokenRewardable.sol (0312_1): 17, 22, 23, 28, 34, 57, 58, 63, 64; contracts/upgradeable_contracts/BaseRewardAddressList.sol (0312_1): 30, 71, 72, 76, 88, 94, 97, 110, 115, 116, 135; contracts/upgradeable_contracts/BaseERC677Bridge.sol (0312_1): 16, 22, 23; contracts/upgradeable_contracts/BasicTokenBridge.sol (0312_1): 72, 78, 84, 89, 94, 111~115, 125; contracts/upgradeable_contracts/ChooseReceiverHelper.sol (0312_1): 15, 17, 18; contracts/upgradeable_contracts/arbitrary_message/BasicAMB.sol (0312_1): 33, 34, 107, 108; contracts/upgradeable_contracts/arbitrary_message/BasicHomeAMB.sol (0312_1): 22, 26, 56, 57, 63, 68, 74; contracts/upgradeable_contracts/arbitrary_message/MessageDelivery.sol (0312_1): 20, 22; contracts/upgradeable_contracts/arbitrary_message/BasicForeignAMB.sol (0312_1): 30, 31, 32, 42; contracts/upgradeable_contracts/BasicAMBMediator.sol (0312_1): 21, 22, 81, 98; contracts/upgradeable_contracts/RewardableBridge.sol (0312_1): 60, 66, 89, 94; contracts/upgradeable_contracts/BaseMediatorFeeManager.sol (0312_1): 25, 37, 43, 83, 84, 85; contracts/upgradeable_contracts/Claimable.sol (0312_1): 10, 51; contracts/upgradeable_contracts/BasicForeignBridge.sol (0312_1): 31, 32, 34, 46; contracts/upgradeable_contracts/Validatable.sol (0312_1): 13; contracts/upgradeable_contracts/GasTokenConnector.sol (0312_1): 81, 84, 88; contracts/upgradeable_contracts/BasicBridge.sol (0312_1): 44; contracts/upgradeable_contracts/ERC20Bridge.sol (0312_1): 15, 20, 21, 22, 23; contracts/upgradeable_contracts/ValidatorsFeeManager.sol (0312_1): 38, 51; contracts/upgradeable_contracts/BridgeValidators.sol (0312_1): 11, 12, 14, 15, 18, 19; contracts/upgradeable_contracts/BlockRewardBridge.sol (0312_1): 17, 28; contracts/upgradeable_contracts/TokenBridgeMediator.sol (0312_1): 58, 59, 60, 73; contracts/upgradeable_contracts/RewardableValidators.sol (0312_1): 12, 13, 15, 16, 17, 20, 21, 22, 50; contracts/upgradeable_contracts/BasicHomeBridge.sol (0312_1): 29, 33, 46, 57, 58, 63, 68; contracts/upgradeable_contracts/amb_native_to_erc20/BasicAMBNativeToErc20.sol (0312_1): 45, 46; contracts/upgradeable_contracts/amb_native_to_erc20/HomeAMBNativeToErc20.sol (0312_1): 55, 73, 74, 146, 158, 161; contracts/upgradeable_contracts/amb_native_to_erc20/ForeignAMBNativeToErc20.sol (0312_1): 104, 107, 127, 129; contracts/upgradeable_contracts/amb_native_to_erc20/ForeignFeeManagerAMBNativeToErc20.sol (0312_1): 43; contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/MultiTokenBridgeMediator.sol (0312_1): 61, 62, 63, 76; contracts/upgradeable_contra	Acknowledged

Category	Severity	Location	Status
		cts/multi_amb_erc20_to_erc677/HomeFeeManagerMultiAMBErc20ToErc677.sol (0312_1): 29, 38, 101, 136, 152; contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/BasicMultiAMBErc20ToErc677.sol (0312_1): 79; contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/HomeMultiAMBErc20ToErc677.sol (0312_1): 44, 45, 128, 145, 164, 169, 238; contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/BasicMultiTokenBridge.sol (0312_1): 133, 134, 146, 147, 159, 160, 171, 172, 182, 183, 232~236, 251; contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/ForeignMultiAMBErc20ToErc677.sol (0312_1): 37, 38, 87, 103, 127, 132, 205, 208, 211; contracts/upgradeable_contracts/Upgradeable.sol (0312_1): 8; contracts/upgradeable_contracts/OverdrawManagement.sol (0312_1): 18, 19, 23; contracts/upgradeable_contracts/ChaiConnector.sol (0312_1): 34, 65, 66, 77, 125, 172, 191, 268, 283, 306, 312; contracts/upgradeable_contracts/RewardableMediator.sol (0312_1): 30; contracts/upgradeable_contracts/BaseBridgeValidators.sol (0312_1): 20, 21, 34, 41, 48, 49, 52, 59, 60, 64, 70, 99, 110, 120; contracts/upgradeable_contracts/Ownable.sol (0312_1): 24, 34~38, 58; contracts/upgradeable_contracts/DecimalShiftBridge.sol (0312_1): 19; contracts/upgradeable_contracts/BaseFeeManager.sol (0312_1): 28; contracts/upgradeability/Proxy.sol (0312_1): 22; contracts/upgradeability/OwnedUpgradeabilityProxy.sol (0312_1): 29, 39, 68; contracts/upgradeability/UpgradeabilityProxy.sol (0312_1): 25, 28, 32; contracts/ERC677BridgeToken.sol (0312_1): 30, 35, 41, 45, 59, 65, 72; contracts/ERC677MultiBridgeToken.sol (0312_1): 44, 45, 46, 49, 62, 67, 73, 91, 98	

Description

The **require** 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 advise adding error messages to the linked **require** statements.

Alleviation

The FlyingCash team acknowledged this finding.

ERT-01 | Incorrect ERC677 Implementation

Category	Severity	Location	Status
Logical Issue	Medium	contracts/ERC677BridgeToken.sol (0312_1): 58~62, 64~68	Acknowledged

Description

In ERC677 token `ERC677BridgeToken`, function `callAfterTransfer()` is used inside `transfer()` and `transferFrom()`, this is not an intended behavior by the ERC667 standard.

Recommendation

We recommend only executing function `callAfterTransfer()` in function `ERC677BridgeToken()`.

Alleviation

The FlyingCash team acknowledged this finding.

ESP-01 | Unnecessary Inheritance In `tokbridge/contracts`

Category	Severity	Location	Status
Logical Issue	Informational	contracts/upgradeability/EternalStorageProxy.sol (0312_1): 3, 13	Acknowledged

Description

In `upgradeability/EternalStorageProxy.sol`

All of the state variables declared in `EternalStorage` are specified as being `internal` and never used, there is no need for contract `EternalStorageProxy` to inherit `EternalStorage`.

Recommendation

We recommend removing the unnecessary inheritance.

Alleviation

The FlyingCash team acknowledged this finding.

FCC-01 | Potential Rug-pull Privileged

Category	Severity	Location	Status
Centralization / Privilege	**Critical**	contracts/FlyingCash.sol (0312_2): 123	Resolved

Description

The `governance` of contract `FlyingCash` can get all the assets in `voucherSet` and `lockToken` without any restriction.

Recommendation

The team should provide a mechanism that can protect users from losing their assets.

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

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

Short Term:

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

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

Long Term:

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

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

Permanent:

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

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

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

The FlyingCash team removed the function `withdraw()` in commit 4b3c1bf7029b040c9151fdf149d4828d8f8585e2.

FCC-02 | Potential Revert

Category	Severity	Location	Status
Logical Issue	Major	contracts/FlyingCash.sol (0312_2): 136	Resolved

Description

In the contract `FlyingCashAdapterNoAsset`, the implementation of function `repayBorrow()` as follow statement.

```
39 function repayBorrow(uint _amount) external override {
40   _amount;
41   require(false, "FlyingCashAdapterNoAsset: repayBorrow not implemented");
42 }
```

According to the above statement, if the value of variable `adapter` is the address of `FlyingCashAdapterNoAsset`, all calls to `withdraw()` will be reverted.

Recommendation

Review the code logic to ensure it meets the design intent.

Alleviation

The FlyingCash team removed the function `repayBorrow()` in commit 4b3c1bf7029b040c9151fdf149d4828d8f8585e2.

FCC-03 | Incorrect Event Emit

Category	Severity	Location	Status
Language Specific	Medium	contracts/FlyingCash.sol (0312_2): 117	Resolved

Description

According to the action in function `withdrawReserve()` and the naming of events declared in interface `IFlyingCash`, event `ReserveAdded` should not be emitted here.

Recommendation

We recommend emitting event `ReserveWithdrawn` here.

Alleviation

The FlyingCash team changed the event name in commit e37c7ecbc26662b3c6d58c0808afaed7ca6d8305.

FCC-04 | Lack Of `return` Statement

Category	Severity	Location	Status
Volatile Code	Minor	contracts/FlyingCash.sol (0312_2): 35, 68	Resolved

Description

Functions `deposit(uint, string, address)` and `withdraw(address, uint)` are declared a return value of type `uint`, but there are no `return` statement.

This issue does not cause compilation warning or compilation error.

Recommendation

Review the code logic to ensure it matches the design intent.

Alleviation

This finding has resolved in commit 4b3c1bf7029b040c9151fdf149d4828d8f8585e2.

FCC-05 | Finance Model

Category	Severity	Location	Status
Volatile Code	Informational	contracts/FlyingCash.sol (0312_2): 87	Acknowledged

Description

In the function `deposit()`, the user can deposit `_amount` quantity of `lockToken` to this contract. And the user will get `_amount - depositFee` quantity of `voucher` tokens in return.

In the function `withdraw()`, the user can retrieve the `lockToken` he deposited from this contract. The user will burn `_amount` quantity of `voucher` tokens to get `_amount` quantity of `lockToken`.

What is the purpose of this contract? The user does not seem to get any reward from this process.

Recommendation

The team should ensure that the rules of this protocol are transparent to the community and that users should know the rules before using it.

Financial models of blockchain protocols need to be resilient to attacks. They need to pass simulations and verifications to guarantee the security of the overall protocol.

The financial model of this protocol is not in the scope of this audit.

Alleviation

[FlyingCash]: Regarding the fee, our design intention is to balance the assets between the chains by the fee rate, and we will make different settings for different situations, and generally charge only one end, not both.

[CertiK]. This is the rule of this protocol and the team will make it transparent to the community. Users should know it before using this protocol.

FCN-01 | Potential Over Mint

Category	Severity	Location	Status
Logical Issue	Critical	contracts/FlyingCashAdapterNoAsset.sol (0312_2): 34	Resolved

Description

```
34 function withdraw(uint _amount) external override {
35   require(whitelist[msg.sender], "FlyingCashAdapterNoAsset: sender is not in
whitelist");
36   token.mint(msg.sender, _amount);
37 }
```

Users in `whitelist[]` maintained by `governance` can mint any number of `tokens` with the function `withdraw()`.

Recommendation

We recommend reviewing the design intent and making the rule transparent to the community.

Alleviation

The FlyingCash team has modified the modifier of the function `withdraw()` to `onlyFlyingCash` to allow only `flyingCash` addresses to call it. This change was committed at 301dbae0798bb6a2b77f76efc469c4cdafa744d0.

FMN-01 | Missing Access Control

Category	Severity	Location	Status
Logical Issue	Critical	contracts/FeeManagerNoAsset.sol (0312_2): 47	Resolved

Description

The function `getWithdrawFee()` will transfer assets from the reserve to any given account. However, this function does not have any access control.

Recommendation

We recommend adding access controls for this function.

Alleviation

This finding has resolved in commit e37c7ecbc26662b3c6d58c0808afaed7ca6d8305.

PCK-01 | Locked Ether

Category	Severity	Location	Status
Language Specific	Medium	contracts/upgradeability/Proxy.sol (0312_1): 19	Acknowledged

Description

The contract has one or more payable functions, but does not have a function to withdraw the fund.

File: projects/tokbridge/contracts/upgradeability/Proxy.sol (Line 19, Contract `Proxy`)

```
    function() public payable {
```

Recommendation

We recommend removing the `payable` attribute or adding a withdraw function.

Alleviation

[FlyingCash]: The contract `Proxy` will not be used and we will not change the code now.

PRO-01 | Redundant Statement

Category	Severity	Location	Status
Volatile Code	Informational	contracts/FlyingCash.sol (0312_2): 6, 100; contracts/upgradeable_contracts/BaseFeeManager.sol (0312_1): 5; contracts/FeeManager.sol (0312_2): 32, 32, 32, 37, 37; contracts/FeeManagerDefault.sol (0312_2): 10, 10, 10, 15, 15, 15; contracts/FeeManagerNoAsset.sol (0312_2): 37; contracts/FlyingCashAdapterNoAsset.sol (0312_2): 40	Resolved

Description

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

File: projects/flyingcash/contracts/FeeManager.sol (Line 32, Function `FeeManager.getDepositeFee`)

```
        account;network;amount;
```

File: projects/flyingcash/contracts/FeeManager.sol (Line 37, Function `FeeManager.getWithdrawFee`)

```
        account;network;
```

File: projects/flyingcash/contracts/FeeManagerDefault.sol (Line 10, Function `FeeManagerDefault.getDepositeFee`)

```
        account;network;amount;
```

File: projects/flyingcash/contracts/FeeManagerDefault.sol (Line 15, Function `FeeManagerDefault.getWithdrawFee`)

```
        account;network;amount;
```

File: projects/flyingcash/contracts/FeeManagerNoAsset.sol (Line 37, Function `FeeManagerNoAsset.getDepositeFee`)

```
        account;
```

File: projects/flyingcash/contracts/FlyingCashAdapterNoAsset.sol (Line 40, Function `FlyingCashAdapterNoAsset.repayBorrow`)

```
        _amount;
```

File: projects/flyingcash/contracts/FlyingCash.sol (Line 6, `import` statement)

```
import "./compound/CErc20.sol";
```

File: projects/flyingcash/contracts/FlyingCash.sol (Line 100, `if` statement)

```
        if (token == address(voucher)) continue;
```

File: projects/tokbridge/contracts/upgradeable_contracts/BaseFeeManager.sol (Line 5, `import` statement)

```
import "../interfaces/IRewardableValidators.sol";
```

Recommendation

We advise that they are removed to better prepare the code for production environments.

Alleviation

The redundant `import` statement has been removed in commit e37c7ecbc26662b3c6d58c0808afaed7ca6d8305.

TBC-01 | Permission Check And Unknown Implementation Of Interface

Category	Severity	Location	Status
Volatile Code	Major	contracts/TokenBridge.sol (0312_2): 15~17	Acknowledged

Description

The function `relayTokens()` of the contract `TokenBridge` is declared as `external` without any restrictions. And the logic is based on the implementation of the `IMultiAMBErc2ToErc677` interface. This could be a risk for this protocol.

Recommendation

The team should ensure that every unrestricted "public" or "external" function is secure and transparent.

Alleviation

[FlyingCash]: We will constantly monitor the statuses to mitigate the side effects when unexpected activities are observed.

UPG-01 | Unchecked Low-level Call

Category	Severity	Location	Status
Control Flow	Medium	contracts/upgradeable_contracts/RewardableMediator.sol (0312_1): 50; contracts/upgradeable_contracts/GasTokenConnector.sol (0312_1): 91	Acknowledged

Description

The low-level `call` function returns the status of the call as first variable in the returned tuple. The status of the `call` is not asserted to be `true`, which would treat the low-level call as a success even when it reverted.

File: projects/tokbridge/contracts/upgradeable_contracts/GasTokenConnector.sol (Line 91, Function `GasTokenConnector._collectGasTokens`)

```
receiver.call(abi.encodeWithSelector(ON_TOKEN_TRANSFER, address(this), target, ""));
```

File: projects/tokbridge/contracts/upgradeable_contracts/RewardableMediator.sol (Line 50, Function `RewardableMediator.distributeFee`)

```
_feeManager.call(abi.encodeWithSelector(ON_TOKEN_TRANSFER, address(this), _fee, ""));
```

Recommendation

We advise to check the return value of a low-level call or log it.

Alleviation

The FlyingCash team acknowledged this finding.

UPG-02 | Missing Input Validation For Type Of Fee

Category	Severity	Location	Status
Volatile Code	Minor	contracts/upgradeable_contracts/RewardableBridge.sol (0312_1): 24, 65; contracts/upgradeable_contracts/BaseFeeManager.sol (0312_1): 20	Acknowledged

Description

Any error in the input of byte32 type parameter `_feeType` will consider the fee type as `FOREIGN_FEE`, even the input is not equal to `FOREIGN_FEE`.

Recommendation

We recommend adding a require statement to make sure `_feeType` is equal to `HOME_FEE` or `FOREIGN_FEE`.

Alleviation

The FlyingCash team acknowledged this finding.

UPG-03 | Risk For Weak Randomness

Category	Severity	Location	Status
Logical Issue	Minor	contracts/upgradeable_contracts/BaseMediatorFeeManager.sol (0312_1): 178~180; contracts/upgradeable_contracts/BaseFeeManager.sol (0312_1): 60~62; contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/HomeFeeManagerMultiAMBErc20ToErc677.sol (0312_1): 111~113	Acknowledged

Description

Weak pseudorandom number generator due to blockhash which can be influenced by miners to some extent so they should be avoided.

Recommendation

We advise the client to consider mixing a seed value based on the chainlink random service.

Alleviation

The FlyingCash team acknowledged this finding.

UPG-04 | External Dependency In `tokbridge/contracts`

Category	Severity	Location	Status
Volatile Code	Minor	contracts/upgradeable_contracts/GasTokenConnector.sol (0312_1): 59; contracts/upgradeable_contracts/ChaiConnector.sol (0312_1): 58	Acknowledged

Description

The implementations of following external contracts are unknown and not in the audit scope.

- `IChai(0x06AF07097C9Eeb7fD685c692751D5C66dB49c215)`
- `ERC20(0x6B175474E89094C44Da98b954EedeAC495271d0F)`
- `IGasToken(0x0000000000b3F879cb30FE243b4Dfee438691c04)`

Recommendation

We recommend ensuring all of the external contracts are correct.

Alleviation

[FlyingCash]: We will constantly monitor the statuses of these external contracts to mitigate the side effects when unexpected activities are observed.

UPG-05 | Dead Code | Redundant Code In `tokbridge/contracts`

Category	Severity	Location	Status
Volatile Code	Informational	contracts/upgradeable_contracts/OtherSideBridgeStorage.sol (0312_1): 8, 12; contracts/upgradeable_contracts/ERC677Bridge.sol (0312_1): 11~18; contracts/upgradeable_contracts/arbitrary_message/HomeAMB.sol (0312_1): 14~16, 18~20; contracts/upgradeable_contracts/arbitrary_message/MessageProcessor.sol (0312_1): 156~175; contracts/upgradeable_contracts/arbitrary_message/ForeignAMB.sol (0312_1): 9~11, 13~15; contracts/upgradeable_contracts/RewardableBridge.sol (0312_1): 21~37, 64~67, 88~91, 93~96; contracts/upgradeable_contracts/BaseOverdrawManagement.sol (0312_1): 28~31; contracts/upgradeable_contracts/ERC20Bridge.sol (0312_1): 14~17; contracts/upgradeable_contracts/ERC677BridgeForBurnableMintableToken.sol (0312_1): 7~10; contracts/upgradeable_contracts/BlockRewardBridge.sol (0312_1): 12~14, 16~30; contracts/upgradeable_contracts/amb_native_to_erc20/BasicAMBNativeToErc20.sol (0312_1): 79~84; contracts/upgradeable_contracts/amb_native_to_erc20/HomeFeeManagerAMBNativeToErc20.sol (0312_1): 37~39; contracts/upgradeable_contracts/amb_native_to_erc20/HomeAMBNativeToErc20.sol (0312_1): 86~102, 109~112, 119~121; contracts/upgradeable_contracts/amb_native_to_erc20/ForeignAMBNativeToErc20.sol (0312_1): 68~83, 90~92, 156~158, 174~176; contracts/upgradeable_contracts/amb_native_to_erc20/ForeignFeeManagerAMBNativeToErc20.sol (0312_1): 52~54; contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/BasicMultiAMBErc20ToErc677.sol (0312_1): 29~31; contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/HomeMultiAMBErc20ToErc677.sol (0312_1): 160~176, 183~193, 201~203; contracts/upgradeable_contracts/multi_amb_erc20_to_erc677/ForeignMultiAMBErc20ToErc677.sol (0312_1): 58~63, 99~109, 193~196	ⓘ Acknowledged

Description

One or more internal functions are not used.

Recommendation

We recommend removing those unused functions.

Alleviation

The FlyingCash team acknowledged this finding.

UPG-06 | Unimplemented Function

Category	Severity	Location	Status
Compiler Error	Informational	contracts/upgradeable_contracts/BlockRewardFeeManager.sol (0312_1): 16; contracts/upgradeable_contracts/BasicHomeBridge.sol (0312_1): 95, 98, 158; contracts/upgradeable_contracts/ValidatorsFeeManager.sol (0312_1): 65, 68; contracts/upgradeable_contracts/ERC677Bridge.sol (0312_1): 21; contracts/upgradeable_contracts/BaseFeeManager.sol (0312_1): 58; contracts/upgradeable_contracts/arbitrary_message/MessageProcessor.sol (0312_1): 205	Acknowledged

Description

These functions were not implemented within the scope of the contract for this audit.

File: projects/tokbridge/contracts/upgradeable_contracts/arbitrary_message/MessageProcessor.sol (Line 205, Contract `MessageProcessor`)

```
    function emitEventOnMessageProcessed(address sender, address executor, bytes32 messageId, bool status) internal;
```

File: projects/tokbridge/contracts/upgradeable_contracts/BaseFeeManager.sol (Line 58, Contract `BaseFeeManager`)

```
    function getFeeManagerMode() external pure returns (bytes4);
```

File: projects/tokbridge/contracts/upgradeable_contracts/BlockRewardFeeManager.sol (Line 16, Contract `BlockRewardFeeManager`)

```
    function distributeFeeFromBlockReward(uint256 _fee) internal;
```

File: projects/tokbridge/contracts/upgradeable_contracts/ERC677Bridge.sol (Line 21, Contract `ERC677Bridge`)

```
    function fireEventOnTokenTransfer(address _from, uint256 _value) internal;
```

File: projects/tokbridge/contracts/upgradeable_contracts/ValidatorsFeeManager.sol (Line 65, Contract `ValidatorsFeeManager`)

```
    function onAffirmationFeeDistribution(address _rewardAddress, uint256 _fee) internal;
```

File: projects/tokbridge/contracts/upgradeable_contracts/ValidatorsFeeManager.sol (Line 68, Contract `ValidatorsFeeManager`)

```
    function onSignatureFeeDistribution(address _rewardAddress, uint256 _fee) internal;
```

File: projects/tokbridge/contracts/upgradeable_contracts/BasicHomeBridge.sol (Line 95, Contract `BasicHomeBridge`)

```
    function onExecuteAffirmation(address, uint256, bytes32) internal returns (bool);
```

File: projects/tokbridge/contracts/upgradeable_contracts/BasicHomeBridge.sol (Line 98, Contract `BasicHomeBridge`)

```
    function onSignaturesCollected(bytes) internal;
```

File: projects/tokbridge/contracts/upgradeable_contracts/BasicHomeBridge.sol (Line 158, Contract `BasicHomeBridge`)

```
    function onFailedAffirmation(address, uint256, bytes32) internal;
```

Recommendation

Please implement all unimplemented functions in any contract you intend to use directly (not simply inherit from).

Alleviation

The FlyingCash team acknowledged this finding.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

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.

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

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.

About

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