CERTIK

Security Assessment

MOBLAND Farming

Jul 20th, 2022

Table of Contents

Summary

This report has been prepared for MOBLAND Farming to discover issues and vulnerabilities in the source code of the MOBLAND Farming 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	MOBLAND Farming
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/superpowerlabs/synr-seed
Commit	af5a36b99f9339c1ba7b85e823934867f55362a0

Audit Summary

Delivery Date	Jul 20, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

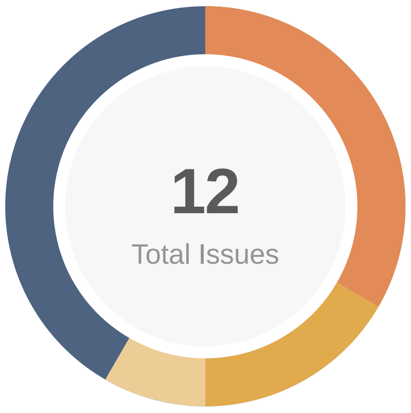
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0	0
● Major	4	0	0	0	4	0	0
● Medium	2	0	0	0	0	0	2
● Minor	1	0	0	0	0	0	1
● Optimization	1	0	0	0	0	0	1
● Informational	5	0	0	0	0	0	5
● Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
STP	token/SeedToken.sol	cef8e61413df3bd230568da0eac739a0e536648ba1aa8921c2a44e7024fbb2ca
SCK	token/SideToken.sol	fae192289a4e415b3d4715d3d3b17062e83a04018229b0b04864a730b5dd82d6
TRK	token/TokenReceiver.sol	86bf0db2623015d11d251bbb6c47866db607251b434f4f783a32982e30874400
WTK	token/WeedToken.sol	ab3fc8ada2aa6e47927ef1ee137d5dbcd56dfb78e384b72a8090a9ce80e8ca57
TCP	Tesseract.sol	98abbf4620e867070a9247b4510ac11af2c1c022783a223ba6a76e9bd5ef0476
MPK	pool/MainPool.sol	b38528fb56e5be63e8786e5ef22583191e47c4698e43d4bf73db9961aec526a0
SPP	pool/SeedPool.sol	42d1c69adab62e9addf513f52b7a754b7f48add6695d7c708695265cc9898b3d
SCP	pool/SidePool.sol	2c0c63981233536d5cf4515aee7d27c490cc69c12ecc68402fbad66c2479bcb6
SPV	pool/SidePoolViews.sol	f7140c8eac9edb70d2c5f0f3bca7a09abd1fc43d7d21f8630d75c435dd37c48d
MWC	bridge/MainWormholeBridge.sol	5723d4a128269fa457d44bc3276eede3ef5357911eb27241f449209d6796003a
SWC	bridge/SideWormholeBridge.sol	0b8849d1b2a5c65e504ebe06da3bb6bb3a2914e057d85a4663f923eb1aa9c72e
WBK	bridge/WormholeBridge.sol	f84a50e785503578cec0c51409d1f90d133979fcab7f21cfb607acdcd6f0dfeb

Findings



🟥	**Critical**	**0** (0.00%)
🟧	**Major**	**4** (33.33%)
🟨	**Medium**	**2** (16.67%)
🟨	**Minor**	**1** (8.33%)
🟦	**Informational**	**5** (41.67%)
🟩	**Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
MOB-01	Centralized Control Of Contract Upgrade	**Centralization / Privilege**	● **Major**	Mitigated
MOB-02	Missing Emit Events	Coding Style	● Informational	✓ Resolved
MPK-01	Discussion Of The `MainPool._unstake()`	Logical Issue	● Informational	✓ Resolved
POL-01	Centralization Risks In `Pool` Contracts	**Centralization / Privilege**	● **Major**	Mitigated
PUU-01	Logic Issue In `validateInput()`	Logical Issue	● Medium	✓ Resolved
SCP-01	Discussion On `SidePool._getStakedAndLockedAmount()`	Logical Issue	● Informational	✓ Resolved
SPC-01	Dead Code	Coding Style	● Informational	✓ Resolved
STP-01	Centralization Risks In `Token` Contracts	**Centralization / Privilege**	● **Major**	Mitigated
TCK-01	Contracts That Lock Ether	Control Flow	● Medium	✓ Resolved
WBC-01	Centralization Risks In `Bridge` Contracts	**Centralization / Privilege**	● **Major**	Mitigated
WBC-02	Assembly Usage	Language Specific	● Informational	✓ Resolved
WBK-01	Third Party Dependencies	Volatile Code	● Minor	✓ Resolved

MOB-01 | Centralized Control Of Contract Upgrade

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	Tesseract.sol: 18; bridge/WormholeBridge.sol: 13; pool/MainPool.sol: 22; pool/SidePool.sol: 21; pool/SidePoolViews.sol: 18; token/SeedToken.sol: 7; token/SideToken.sol: 11; token/TokenReceiver.sol: 9; token/WeedToken.sol: 7	Mitigated

Description

`PayloadUtilsUpgradeable`, `OwnableUpgradeable`, `UUPSUpgradeable`, `WormholeTunnelUpgradeable`, `ERC20Upgradeable`, `ERC20BurnableUpgradeable`, and `IERC721ReceiverUpgradeable` are upgradeable contracts, the owner can upgrade the contract without the community's commitment. If an attacker compromises the account, he can change the implementation of the contract and drain tokens from the contract.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

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

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

Long Term:

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

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

Permanent:

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

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

Alleviation

MOBLAND team: There is a strategy for it. Following OpenZeppelin best practices we will deploy the contracts, then we will transfer the ownership of the proxy-contract to a Gnosis safe multi-sig wallet. Then, any following upgrade will be performed according to that process. Here is the guide we will follow to transfer ownership to the multi-sig wallet, and later to deploy new implementations: https://docs.openzeppelin.com/defender/guide-upgrades

About the time lock. We are not implementing an explicit process because when a bug is discovered (which is the primary reason why we are using upgradeable contracts), the speed of the response is crucial to avoid the disaster. For example, the recent crash of the UST could have been mitigated if they did not have to wait the fixed lockup time before intervening.

MOB-02 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	● Informational	Tesseract.sol: 34; bridge/MainWormholeBridge.sol: 17; bridge/SideWormholeBridge.sol: 17; bridge/WormholeBridge.sol: 35, 40; pool/MainPool.sol: 61, 360, 377; pool/SeedPool.sol: 34, 36; pool/SidePool.sol: 78, 495; pool/SidePoolViews.sol: 29; token/SeedToken.sol: 15; token/SideToken.sol: 33, 38; token/WeedToken.sol: 15	✓ 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 MOBLAND team fixed this issue in the commit `c3583b88e1a4df34cdc5a9bc15aa0bd3152f26cd`.

MPK-01 | Discussion Of The `MainPool._unstake()`

Category	Severity	Location	Status
Logical Issue	● Informational	pool/MainPool.sol: 294~297	✓ Resolved

Description

In the `_unstake()` function, if `tokenType = SYNR_STAKE`, no need to verify `lockedUntil`?

In `deposit`, `tokenType=SYNR_STAKE` and `tokenType=SYNR_PASS_STAKE_FOR_SEEDS`, both set the `lockedUntil` attribute.

Is this consistent with the design intent?

Recommendation

Please ensure that this is done correctly and in accordance with the design intent.

Alleviation

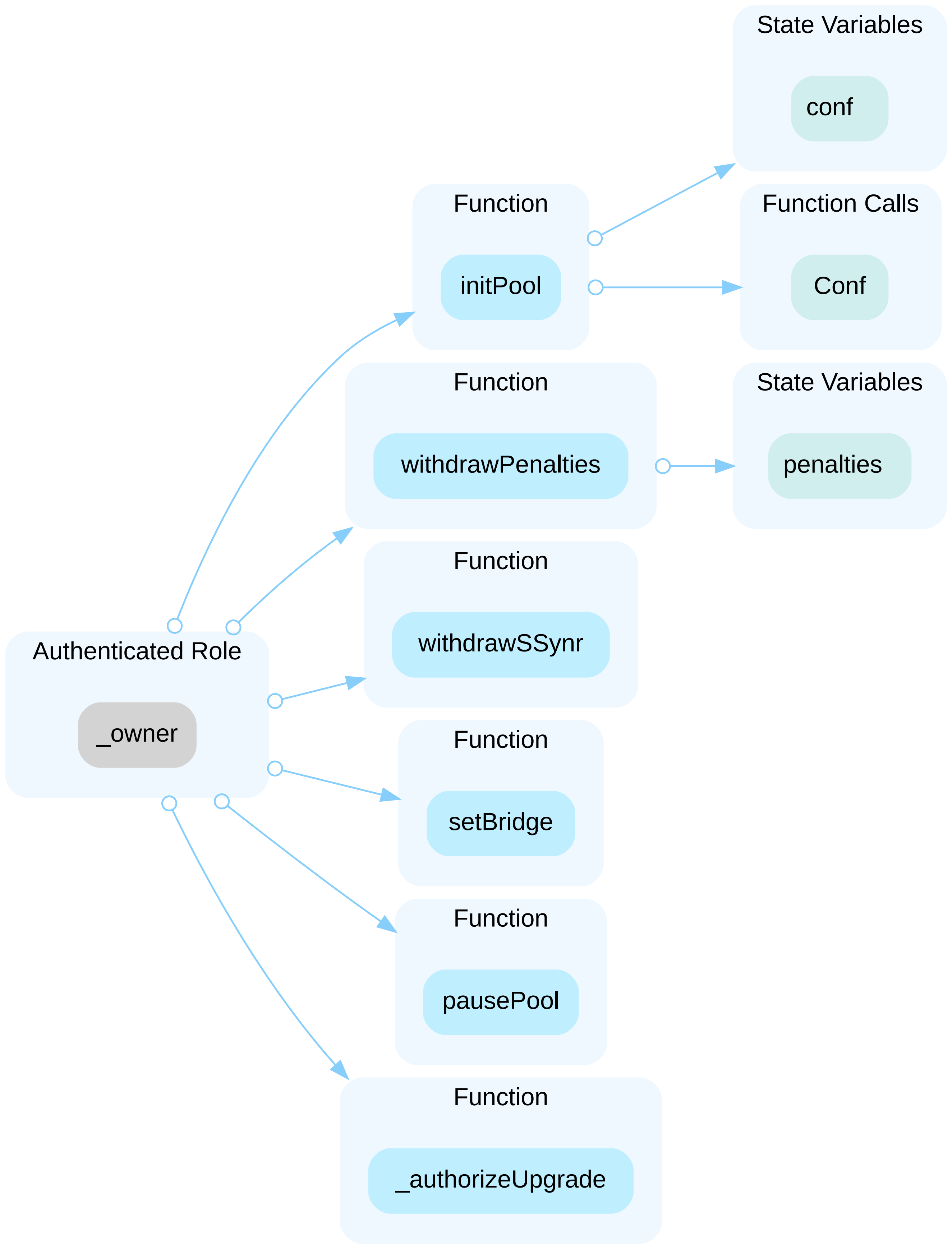
MOBLAND team: The lockupUntil parameter is set during the staking, but the token can be unstaked earlier paying a penalty. That is why that parameter is not checked during the unstake.

POL-01 | Centralization Risks In `Pool` Contracts

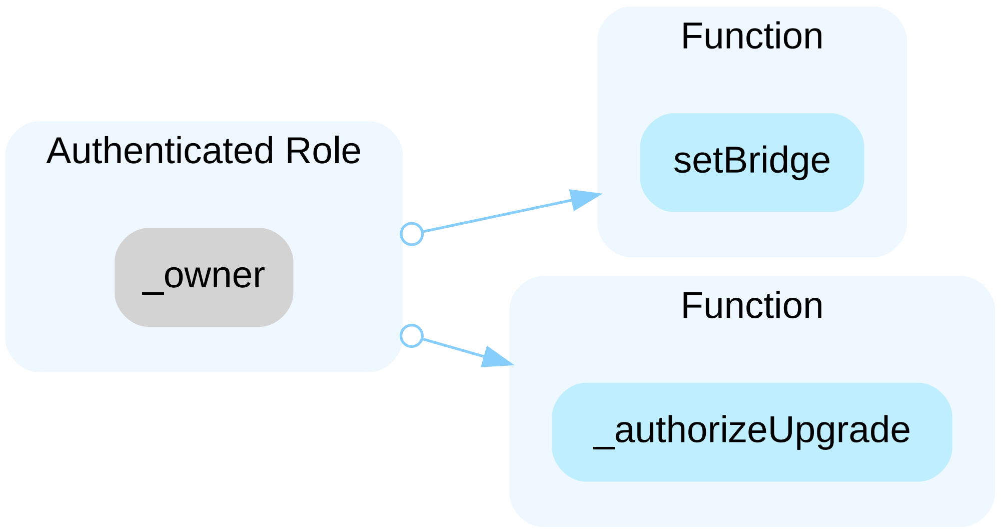
Category	Severity	Location	Status
Centralization / Privilege	● **Major**	pool/MainPool.sol (06.07): 87, 89, 100, 116, 361, 378; pool/SeedPool.sol (06.07): 30, 32; pool/SidePool.sol (06.07): 72, 74, 166, 173, 203, 687	Mitigated

Description

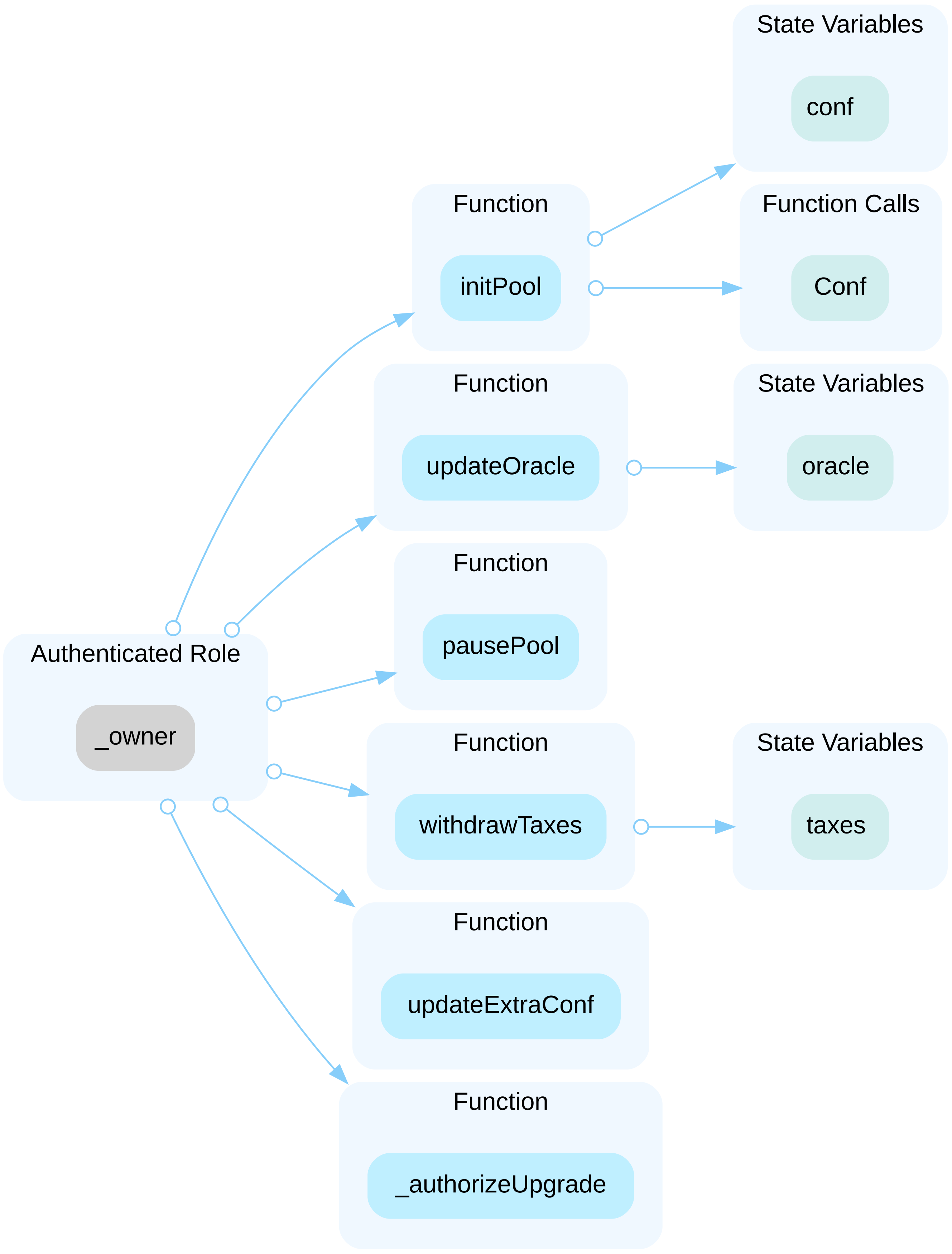
In the contract `MainPool` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.


State Variables
conf
Function
initPool
Function Calls
Conf
Function
withdrawPenalties
State Variables
penalties
Function
withdrawSSynr
Authenticated Role
_owner
Function
setBridge
Function
pausePool
Function
_authorizeUpgrade

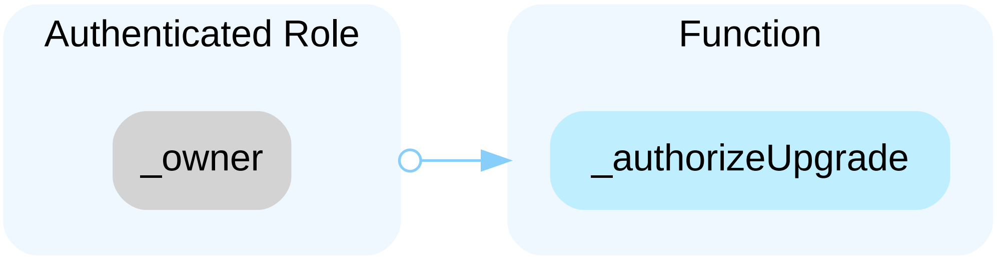
In the contract `SeedPool` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



In the contract `SidePool` 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.


State Variables
conf
Function
initPool
Function Calls
Conf
Function
updateOracle
State Variables
oracle
Function
pausePool
Authenticated Role
_owner
Function
withdrawTaxes
State Variables
taxes
Function
updateExtraConf
Function
_authorizeUpgrade

In the contract `SidePoolViews` 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

MOBLAND team: We will transfer all contracts' ownership to a set of multi-sig wallets after deploying the contracts. Since we are focusing on the safety of the contract, similar to what we say about MOB-01, we are not going to set a time-lock to be able to intervene promptly if necessary. However, as soon as we think that the contracts are safe, we will renounce to the ownership.

PUU-01 | Logic Issue In `validateInput()`

Category	Severity	Location	Status
Logical Issue	● Medium	utils/PayloadUtilsUpgradeable.sol (06.07): 21~36	✓ Resolved

Description

1. The maximum value of `tokenType` for enumerated types is 6, so the following require statement is more appropriate:

```
require(tokenType <= 6, "PayloadUtilsUpgradeable: invalid token type");
```

2. When the `tokenType` is `BLUEPRINT_STAKE_FOR_BOOST` or `BLUEPRINT_STAKE_FOR_SEEDS`, the input parameter `tokenAmountOrID` must be less than 8001 for both.

Recommendation

It is suggested to modify the two checks mentioned above.

Alleviation

MOBLAND team: That value was there in consideration of the maximum number of token types we plan to support in the future. Also, that check about the blueprint for boost or for seed was a remaining from a previous version. In fact, validateInput is only called by the MainPool which cannot receive blueprints (being on Ethereum, while blueprints tokens are on BNB Chain). So, that line has been removed in https://github.com/superpowerlabs/synr-seed/pull/116

The entire contract PayloadUtils has been removed as well because the contract initially was used by all the pool, but the contracts evolved so that only MainPool uses 3 of the 4 functions in PayloadUtils, while the bridges use the 4th one. So, it was better to integrate those functions in MainPool and WormholeBridge and delete PayloadUtils.

SCP-01 | Discussion On `SidePool._getStakedAndLockedAmount()`

Category	Severity	Location	Status
Logical Issue	● Informational	pool/SidePool.sol: 334~336	✓ Resolved

Description

```
} else if (tokenType != BLUEPRINT_STAKE_FOR_BOOST && tokenType !=
SYNR_PASS_STAKE_FOR_BOOST) {
  revert("SidePool: invalid tokenType");
}
```

In the function `SidePool._getStakedAndLockedAmount()`, according to the `else if` condition, when `tokenType` is `BLUEPRINT_STAKE_FOR_BOOST` or `SYNR_PASS_STAKE_FOR_BOOST`, all the `else-if` conditions are missed, and the call will return `stakedAmount=0` and `generator=0` directly.

Please review this logic and ensure it meets the design intent.

Recommendation

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

MOBLAND team: When staking a SYNR Pass or a Blueprint for boost, there is no locked amount because the two NFTs are used to boost the existent locked SEED. That is why that condition is used only to exclude invalid tokens (which is an extra condition needed only to get a more precise reason for the revert).

SPC-01 | Dead Code

Category	Severity	Location	Status
Coding Style	● Informational	pool/SidePool.sol (06.07): 212	✓ Resolved

Description

One or more internal functions are not used.

File: mobland/pool/SidePool.sol (Line 212, Contract `SidePool`)

```
function _updateLastRatioUpdateAt() internal {
```

Recommendation

We recommend removing those unused functions.

Alleviation

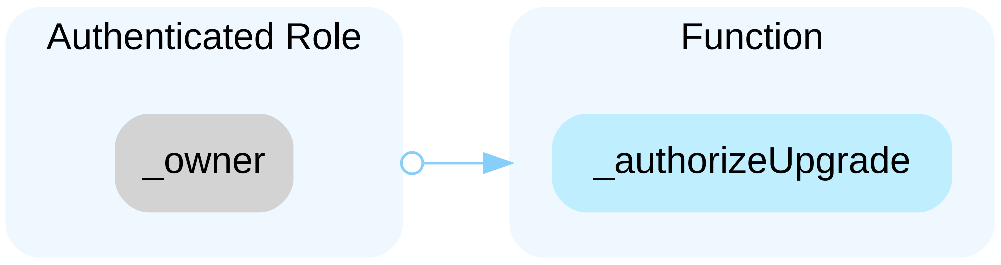
The MOBLAND team fixed this issue in the commit `407d704546c626e2f320a0164e0a3fee2ea1169d`.

STP-01 | Centralization Risks In Token Contracts

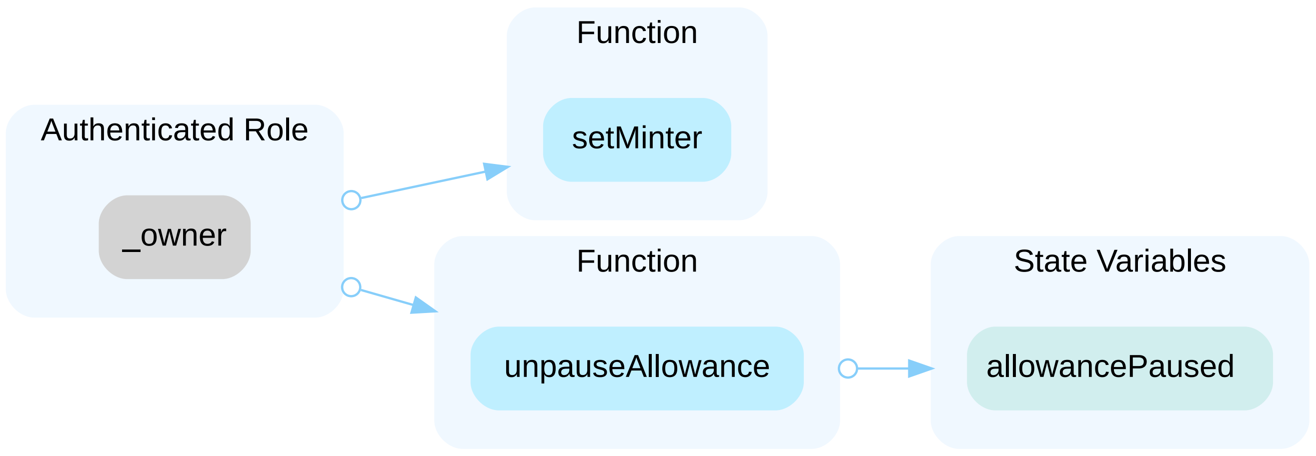
Category	Severity	Location	Status
Centralization / Privilege	● **Major**	token/SeedToken.sol: 15	Mitigated

Description

In the contract `SeedToken` 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.



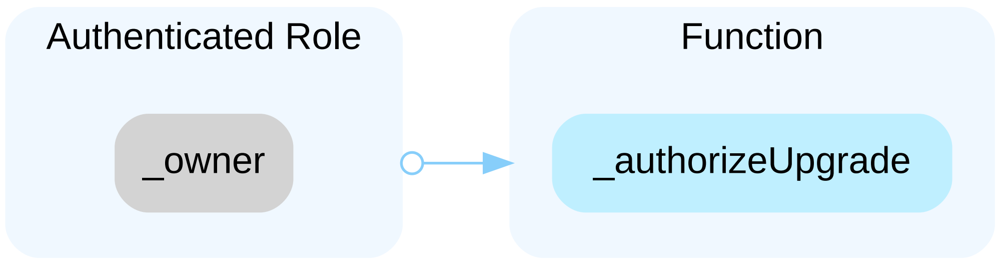
In the contract `SideToken` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



In the contract `SideToken`, the role `minter` has authority over the following functions:

- mint()

In the contract `WeedToken` 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

MOBLAND team: We will transfer all contracts’ ownership to a set of multi-sig wallets after deploying the contracts. Since we are focusing on the safety of the contract, similar to what we say about MOB-01, we are not going to set a time-lock to be able to intervene promptly if necessary. However, as soon as we think that the contracts are safe, we will renounce to the ownership.

TCK-01 | Contracts That Lock Ether

Category	Severity	Location	Status
Control Flow	● Medium	Tesseract.sol (06.07): 50	✓ Resolved

Description

Contract with a payable function, but without a withdrawal capacity.

```
function crossChainTransfer(
  uint8 bridgeType,
  uint256 payload,
  uint16 recipientChain,
  uint32 nonce
) external payable virtual override returns (uint64 sequence) {
  ...
}
```

Every Ether sent to `Tesseract` will be lost.

Recommendation

Remove the payable attribute or add a withdraw function.

Alleviation

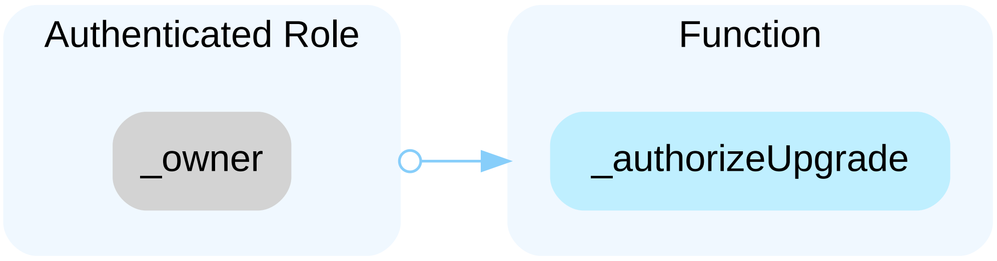
MOBLAND team: The Wormhole protocol requires that the function starting the transfer is payable because in the future Wormhole can decide to apply a fee. Right now the function crossChainTransfer is not transferring any value to the bridge contract, but we think it is better to keep it ready, in case that has to be done, without changing the signature of the function. Thus said, the risk that some money are locked forever in the contract is not a real issue because the contract is upgradeable, but to mitigate that we added a withdraw function in https://github.com/superpowerlabs/synr-seed/pull/118

WBC-01 | Centralization Risks In `Bridge` Contracts

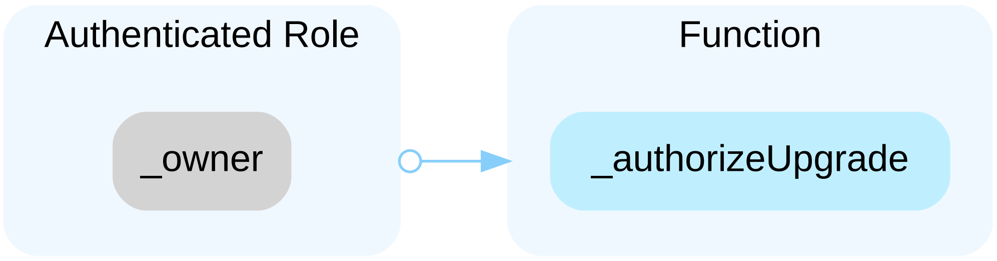
Category	Severity	Location	Status
Centralization / Privilege	● **Major**	bridge/WormholeBridge.sol (06.07): 35	Mitigated

Description

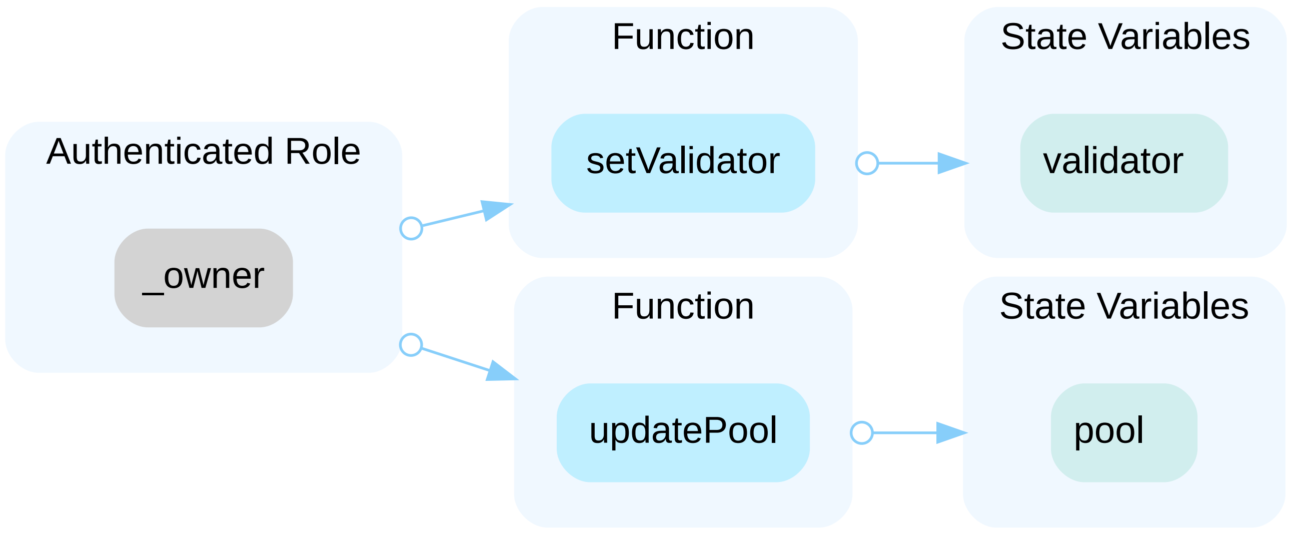
In the contract `MainWormholeBridge` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



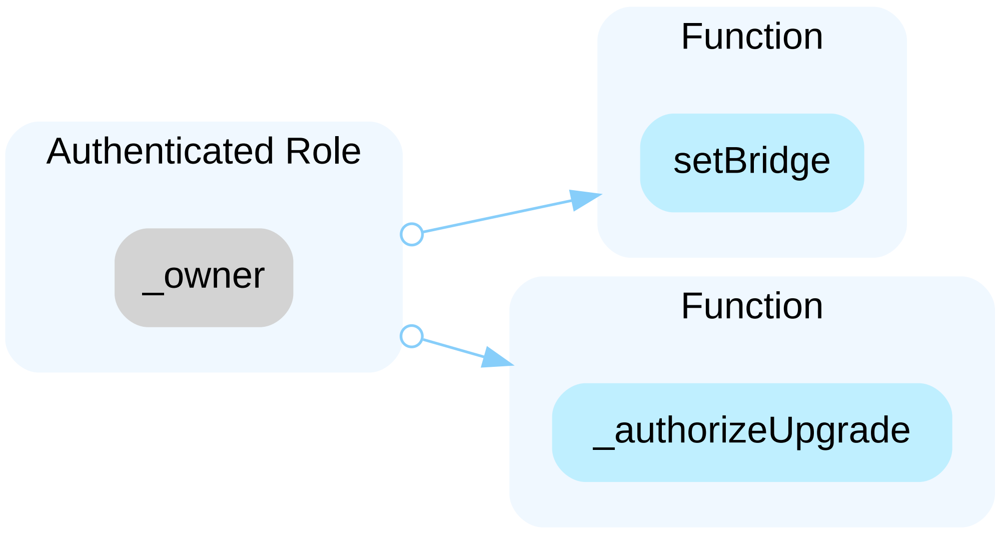
In the contract `SideWormholeBridge` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



In the contract `WormholeBridge` 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 `Tesseract` 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

MOBLAND team: We will transfer all contracts' ownership to a set of multi-sig wallets after deploying the contracts. Since we are focusing on the safety of the contract, similar to what we say about MOB-01, we are not going to set a time-lock to be able to intervene promptly if necessary. However, as soon as we think that the contracts are safe, we will renounce to the ownership.

WBC-02 | Assembly Usage

Category	Severity	Location	Status
Language Specific	● Informational	bridge/WormholeBridge.sol (06.07): 106~108	✓ Resolved

Description

File: mobland/bridge/WormholeBridge.sol (Line 106-108, Function `WormholeBridge.getChainId`)

```
assembly {
  id := chainid()
}
```

Recommendation

We advise against using EVM assembly, as it is error-prone.

Alleviation

The MOBLAND team fixed this issue in the commit `fcc98ec265d88b3742851b17326c826624fd533c`.

WBK-01 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	● Minor	bridge/WormholeBridge.sol: 13	✓ Resolved

Description

The `WormholeBridge` contract is inherited from third-party `WormholeTunnelUpgradeable` protocol. The scope of the audit treats 3rd party entities as black boxes and assumes 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 `WormholeBridge` requires interaction with `WormholeTunnelUpgradeable`. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

MOBLAND team: The WormholeTunnel package is not a third party package. It has been co-written, and is currently managed by the author of the synr-seed contracts. Look at the README of the repo at https://github.com/ndujaLabs/wormhole-tunnel for more details.

Optimizations

ID	Title	Category	Severity	Status
PUC-01	Redundant Contract	Gas Optimization	● Optimization	✓ Resolved

PUC-01 | Redundant Contract

Category	Severity	Location	Status
Gas Optimization	● Optimization	utils/PayloadUtils.sol (06.07)	✓ Resolved

Description

The `PayloadUtils` contract is basically the same as `PayloadUtilsUpgradeable` contract, just keep one.

Recommendation

Consider removing this contract.

Alleviation

The MOBLAND team fixed this issue in https://github.com/superpowerlabs/synr-seed/pull/116.

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.

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.