CERTIK

Preliminary Comments

ACENT Mainnet

Jul 26th, 2022

Table of Contents

Summary

This report has been prepared for ACENT Mainnet to discover issues and vulnerabilities in the source code of the ACENT Mainnet 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	ACENT Mainnet
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/triggah61/Acent-mainnet
Commit	c7c56d3de6f9ef96aefb132d1d5bfd9aee7d835a

Audit Summary

Delivery Date	Jul 26, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	1	1	0	0	0	0	0
Major	12	12	0	0	0	0	0
Medium	0	0	0	0	0	0	0
Minor	19	19	0	0	0	0	0
Informational	4	4	0	0	0	0	0
Discussion	6	6	0	0	0	0	0

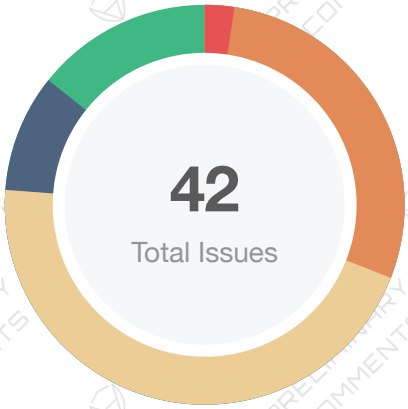
Audit Scope

ID	File	SHA256 Checksum
MAA	Acent Swap Smart Contract/Ade-Farm/contracts/libs/Multicall2.sol	efd4a023e908eb7fa5e211ea8cd4c16f6ab9172d0eb5702ce5519b2c89336dc9
WAC	Acent Swap Smart Contract/Ade-Farm/contracts/libs/WACE.sol	49532f25d353436f55f63036fdc3e9d538dae643c2066f9d6372f110b93965e8
ADE	Acent Swap Smart Contract/Ade-Farm/contracts/ADEInitMintable.sol	459523b3c9d708805c61114a5736ed12731fa84c7a08a5ca7acd10a2fffb3bc3
ADT	Acent Swap Smart Contract/Ade-Farm/contracts/ADEToken.sol	bda8d54b6a4a39dff56344ccc7a500c609f04288ce52244cdd38c8d4da7f8672
ADV	Acent Swap Smart Contract/Ade-Farm/contracts/ADEVault.sol	29d1610bc6624853d1bd41e28d6f66b8f6fcaf59f29ac1e51e50e2810a257f0e
ISA	Acent Swap Smart Contract/Ade-Farm/contracts/ISupercluster.sol	bdb4614c5616409ddef48c2b8483fda7f36127a38fcc060e20b172b44fb662ea
SAF	Acent Swap Smart Contract/Ade-Farm/contracts/Spica.sol	3353b98b0fb1b6af1f74d7cfb94c02adf7f580b023ac2e83955c6e3e7724c00d
SAA	Acent Swap Smart Contract/Ade-Farm/contracts/Supercluster.sol	ff1d0a0c6d8fcc3335ebc29fbab125b60644c655943c0826456933b5a1788a1b
TAF	Acent Swap Smart Contract/Ade-Farm/contracts/Timelock.sol	1dba5050448f8b8c9e8b7f79aee2b836aa3db0a5a27bebe1b1389d1417e23355
VAF	Acent Swap Smart Contract/Ade-Farm/contracts/Virgo.sol	3fec7da6b8a860773a135b6b157acbb18421c02c133cc2a55d0f3e77505e4f72
VAA	Acent Swap Smart Contract/Ade-Farm/contracts/VirgoAdmin.sol	4e46c4577f8c2aa6d9250cdf77523d1d95cef127ca47efd473d8f95256e22c41
IAC	Acent Swap Smart Contract/ace-swap-core-sh/contracts/interfaces/IAcentCallee.sol	aa4a0b491f55e5875b09c9b3ad03afd930672b89f9b2e0698a4f4e3e6cfb4d4f
IAE	Acent Swap Smart Contract/ace-swap-core-sh/contracts/interfaces/IAcentERC20.sol	1e8af03ee36e7b318e8ce7b7a74f7e7efb72ad1078275a70f6914b335a295c27
IAF	Acent Swap Smart Contract/ace-swap-core-sh/contracts/interfaces/IAcentFactory.sol	c772a42a74eade991d41e194eb458eb1d3e423d4abf4d521902d6f87f4a55821

ID	File	SHA256 Checksum
IAP	Acent Swap Smart Contract/ace-swap-core-sh/contracts/interfaces/IAcentPair.sol	88f8cc09f8b643b880cda14cfb298973b43f62533c95aaa5def5b1ddab93b0b2
IER	Acent Swap Smart Contract/ace-swap-core-sh/contracts/interfaces/IERC20.sol	9d68ff20353b1cfcc8e1a5530b7eaf79dea590df75b9a380efb7d12dcfe6f468
MAC	Acent Swap Smart Contract/ace-swap-core-sh/contracts/libraries/Math.sol	e4a9d451964a0689be2b244322a353de143ca4248d8736d91aca4ffadca4325f
SMA	Acent Swap Smart Contract/ace-swap-core-sh/contracts/libraries/SafeMath.sol	4b1c95ff75de7342e0fadff58064820a4eb7c2fcb422a75b4994980ce8e216ae
UQA	Acent Swap Smart Contract/ace-swap-core-sh/contracts/libraries/UQ112x112.sol	6633b57b0723b1d72e08cc3e8b29f0af838294e59863b6cdcce95a141ed02cdb
AER	Acent Swap Smart Contract/ace-swap-core-sh/contracts/AcentERC20.sol	0eb0815d68609aa1dd432e07c3aa5fa389dcde28bfe9ebfacba7d98b54aaa4db
AFC	Acent Swap Smart Contract/ace-swap-core-sh/contracts/AcentFactory.sol	2353a290012088b35aad501c12112549ece12bb219fc6ab4b51cb317259fb942
APA	Acent Swap Smart Contract/ace-swap-core-sh/contracts/AcentPair.sol	ffd9775def3985fedeed4bf29bf0cb0da4a86eb7076cb55ccc6c208a50e428b8
IAV	Acent Swap Smart Contract/deploy_periphery2/contracts/interfaces/V1/IADEV1Exchange.sol	923c26a68cc7efd9101e2eba3d67adbae1fd26f57609732335805f169447bbbc
IDE	Acent Swap Smart Contract/deploy_periphery2/contracts/interfaces/V1/IADEV1Factory.sol	8c4eaf66eca13d2984e20707390c9190bf2a9b8ed080b35a5560237153e53f6d
IAD	Acent Swap Smart Contract/deploy_periphery2/contracts/interfaces/IADEFactory.sol	dd10a7f73da44912dd66e15a77e02990710bfe7bc53d058d6df941730579b894
IAM	Acent Swap Smart Contract/deploy_periphery2/contracts/interfaces/IADEMigrator.sol	d85f852578d762818d476cce43e314ac8cc9d0bc2dd25effadc77c77abd8ee39
IAA	Acent Swap Smart Contract/deploy_periphery2/contracts/interfaces/IADEPair.sol	c6f5e91c47340d82c7176a8b086617daabc973d5765f78515b89baa5278494e2
IAR	Acent Swap Smart Contract/deploy_periphery2/contracts/interfaces/IADERouter01.sol	c868129a9820942561bc13de16f4704347e7ef2ae9b1c2a7a8f65797f14167ab
IAS	Acent Swap Smart Contract/deploy_periphery2/contracts/interfaces/IADERouter02.sol	0c009e9743d476be58bb436c8209038c87b223060430555a344a4ec24921b9a4

ID	File	SHA256 Checksum
IEC	Acent Swap Smart Contract/deploy_periphery2/contracts/interfaces/IERC20.sol	2b63f199f838028184efefbcfd6cf2b9192624c3dae5dc1116ecbb15c36a67e8
IWE	Acent Swap Smart Contract/deploy_periphery2/contracts/interfaces/IWETH.sol	60760053849b916b72386fef1126ab69c7482c2aa6b5fb836368eff42905cb30
ADL	Acent Swap Smart Contract/deploy_periphery2/contracts/libraries/ADELibrary.sol	854a9a1e8c9ff27456664439cfeec6a14896a34052c8c1ac9b7ff47a97c431b5
ADO	Acent Swap Smart Contract/deploy_periphery2/contracts/libraries/ADEOracleLibrary.sol	5d350231587bb555394b79c5d26105904fa9a633bbc692c3be120509527cbfd3
SMS	Acent Swap Smart Contract/deploy_periphery2/contracts/libraries/SafeMath.sol	39e5b4c0bc19b72fa59c2d2177ba5ed6cfadcd76f3f804563011e579367530fb
ADM	Acent Swap Smart Contract/deploy_periphery2/contracts/ADEMigrator.sol	3205372f2e1c5b47675b81ca8db2954f87640c7a55c091c835cc4a11bc168ec4
ADR	Acent Swap Smart Contract/deploy_periphery2/contracts/ADERouter.sol	c1ccf7876b1a36fe34983a8241a2bbdf7ca05a76c9a799be7fd720e9acdcd38e
ADA	Acent Swap Smart Contract/deploy_periphery2/contracts/ADERouter01.sol	90fb46b9fe66d36ab07d3b630008662671dc4853dd99cebc1157120c8c5b4a17
RAR	Acent Swap Smart Contract/deploy_periphery2/contracts/RealAdeRouter.sol	f68c7c1ce1ef0efd60fcc706c5522c4721eba18b3262a54a43a40bd7322e34cf
WAE	Acent Swap Smart Contract/deploy_periphery2/contracts/WACE.sol	65980a786998ef3556cff4420b63fc2483e7115e184f3a3b0feeab23c01aa1b3
FDP	POSA Smart Contract/contracts/FundDistributor.sol	e217a9dd918e79c31c57031ed5f0b22b5f936c7a33bd421ec5c30d608378a87e

Findings



Critical	1 (2.38%)
Major	12 (28.57%)
Medium	0 (0.00%)
Minor	19 (45.24%)
Informational	4 (9.52%)
Discussion	6 (14.29%)

ID	Title	Category	Severity	Status
GLOBAL-01	Unlocked Compiler Version	Language Specific	Informational	Pending
ADE-01	Centralization Risks In ADEInitMintable.sol	**Centralization / Privilege**	**Major**	Pending
ADR-01	Incompatibility With Deflationary Tokens	Logical Issue	Minor	Pending
ADT-01	Centralization Risks In ADEToken.sol	**Centralization / Privilege**	**Major**	Pending
ADV-01	Centralization Risks In ADEVault.sol	**Centralization / Privilege**	**Major**	Pending
ADV-02	Centralization Risks In `treasury` Address	Logical Issue	Major	Pending
ADV-03	Unreasonable Charges	Logical Issue	Discussion	Pending
AFC-01	Centralization Risks In AcentFactory.sol	**Centralization / Privilege**	**Major**	Pending
APA-01	Centralization Risks In AcentPair.sol	**Centralization / Privilege**	**Major**	Pending
APA-02	Divide By Zero	Logical Issue	Minor	Pending
APA-03	Inconsistent Comment And Code	Inconsistency	Discussion	Pending
ASS-02	Discussion About Swap Fees	Logical Issue	Discussion	Pending
CAB-01	Missing Zero Address Validation	Volatile Code	Minor	Pending

ID	Title	Category	Severity	Status
FDP-01	Centralization Risks In FundDistributor.sol	**Centralization / Privilege**	**Major**	Pending
FDP-02	Missing Input Validation	Volatile Code	Minor	Pending
MAA-01	Potential Risk Of Low-level Call	Logical Issue, Language Specific	Minor	Pending
SAA-01	Centralization Risks In Supercluster.sol	**Centralization / Privilege**	**Major**	Pending
SAA-02	Missing Input Validation	Volatile Code	Minor	Pending
SAA-03	Missing Input Validation	Volatile Code	Informational	Pending
SAA-04	The Source Of The `offeringToken`	Logical Issue	Discussion	Pending
SAA-05	Contract Usage Scenarios	Logical Issue	Discussion	Pending
SAF-01	Centralization Risks In Spica.sol	**Centralization / Privilege**	**Major**	Pending
TAF-01	Centralization Risks In Timelock.sol	**Centralization / Privilege**	**Major**	Pending
VAA-01	Centralization Risks In VirgoAdmin.sol	**Centralization / Privilege**	**Major**	Pending
VAA-03	Function Cannot Call After Ownership Transfer	Logical Issue	Discussion	Pending
VAF-01	Logic Flaw In `emergencyWithdraw()`	Logical Issue	Critical	Pending
VAF-02	Centralization Risks In Virgo.sol	**Centralization / Privilege**	**Major**	Pending
VAF-03	Unknown Implementation Of `migrator.migrate()`	Logical Issue	Minor	Pending
VAF-04	`add()` Function Not Restricted	Logical Issue	Minor	Pending
VAF-07	Inefficient Code Execution Path	Logical Issue	Informational	Pending
VAF-08	Lack Of Pool Validity Checks	Logical Issue	Informational	Pending

ID	Title	Category	Severity	Status
WAC-01	Function `transfer` Prevents Transfers To The Zero Address	Logical Issue	Minor	Pending
WAC-02	Function `transfer` Prevents Overflows In The Recipient's Balance	Logical Issue	Minor	Pending
WAC-03	Function `transferFrom` Fails For Transfers From The Zero Address	Logical Issue	Minor	Pending
WAC-04	Function `transferFrom` Fails For Transfers To The Zero Address	Logical Issue	Minor	Pending
WAC-05	Function `transferFrom` Prevents Overflows In The Recipient's Balance	Logical Issue	Minor	Pending
WAC-06	Missing `emit` Keyword	Language Specific	Minor	Pending
WAE-01	Function `transfer` Prevents Transfers To The Zero Address	Logical Issue	Minor	Pending
WAE-02	Function `transfer` Prevents Overflows In The Recipient's Balance	Logical Issue	Minor	Pending
WAE-03	Function `transferFrom` Fails For Transfers From The Zero Address	Logical Issue	Minor	Pending
WAE-04	Function `transferFrom` Fails For Transfers To The Zero Address	Logical Issue	Minor	Pending
WAE-05	Function `transferFrom` Prevents Overflows In The Recipient's Balance	Logical Issue	Minor	Pending

GLOBAL-01 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational		Pending

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 differing compiler version numbers. 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 `v0.6.12` the contract should contain the following line:

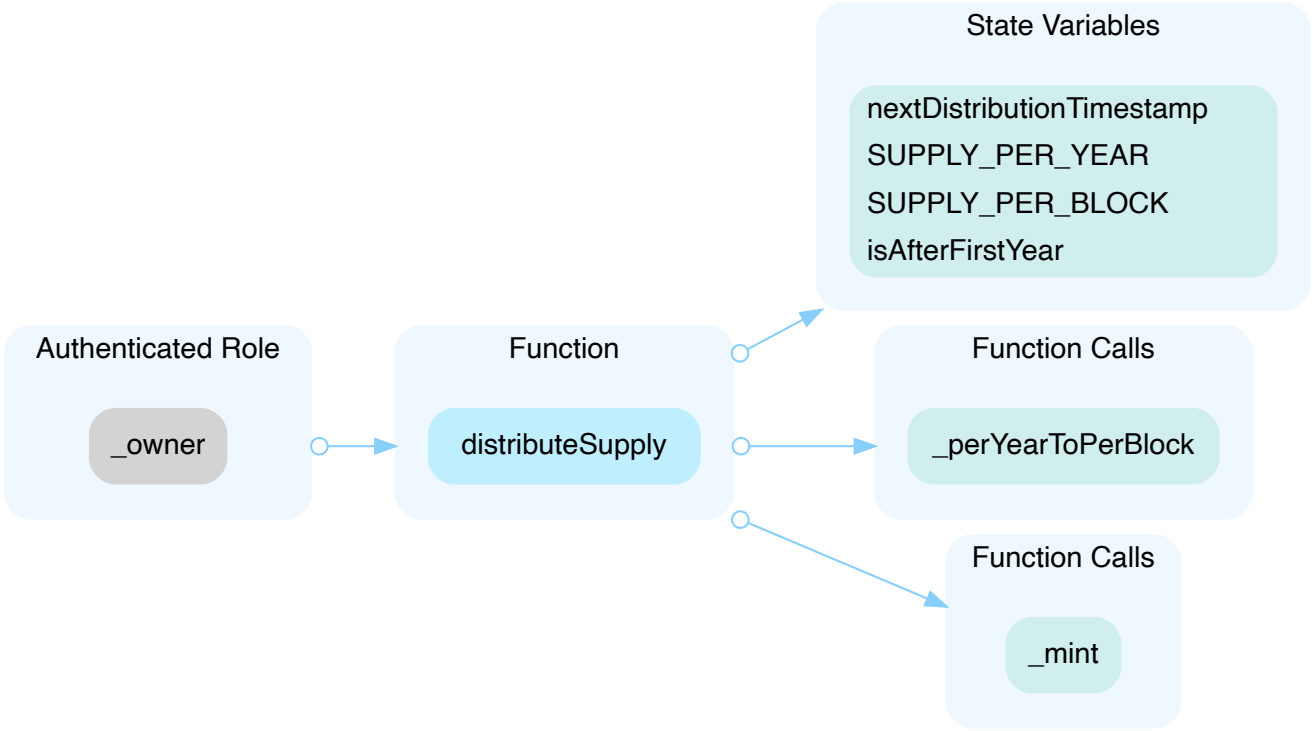
```
pragma solidity 0.6.12;
```

ADE-01 | Centralization Risks In ADEInitMintable.sol

Category	Severity	Location	Status
Centralization / Privilege	Major	Acent Swap Smart Contract/Ade-Farm/contracts/ADEInitMintable.sol: 27	Pending

Description

In the contract `ADEInitMintable` 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.

ADR-01 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	Minor	Acent Swap Smart Contract/deploy_periphery2/contracts/ADERouter.sol: 63	Pending

Description

When users add or remove LP tokens into the router, and the `mint` and `burn` operations are performed. When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. As a result, the amount inconsistency will occur and the transaction may fail due to the validation checks.

Recommendation

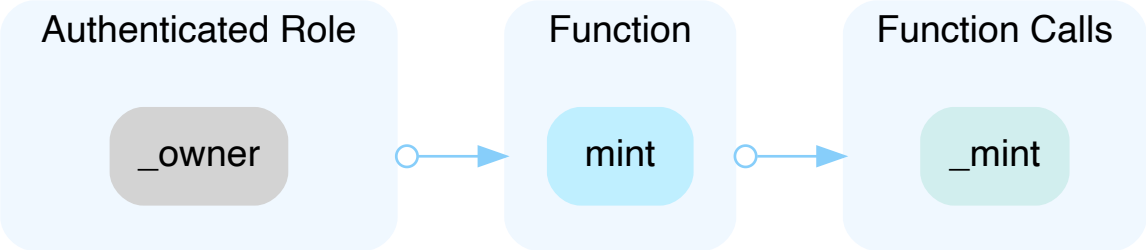
We advise the client to regulate the set of LP tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

ADT-01 | Centralization Risks In ADEToken.sol

Category	Severity	Location	Status
Centralization / Privilege	Major	Acent Swap Smart Contract/Ade-Farm/contracts/ADEToken.sol: 14	Pending

Description

In the contract `ADEToken` 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*.

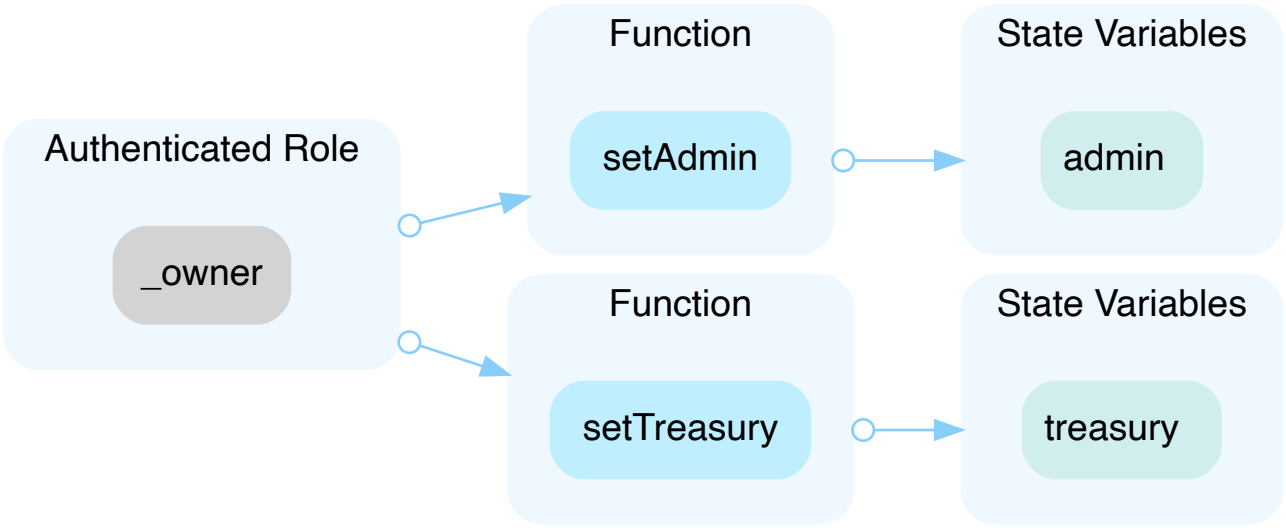
- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

ADV-01 | Centralization Risks In ADEVault.sol

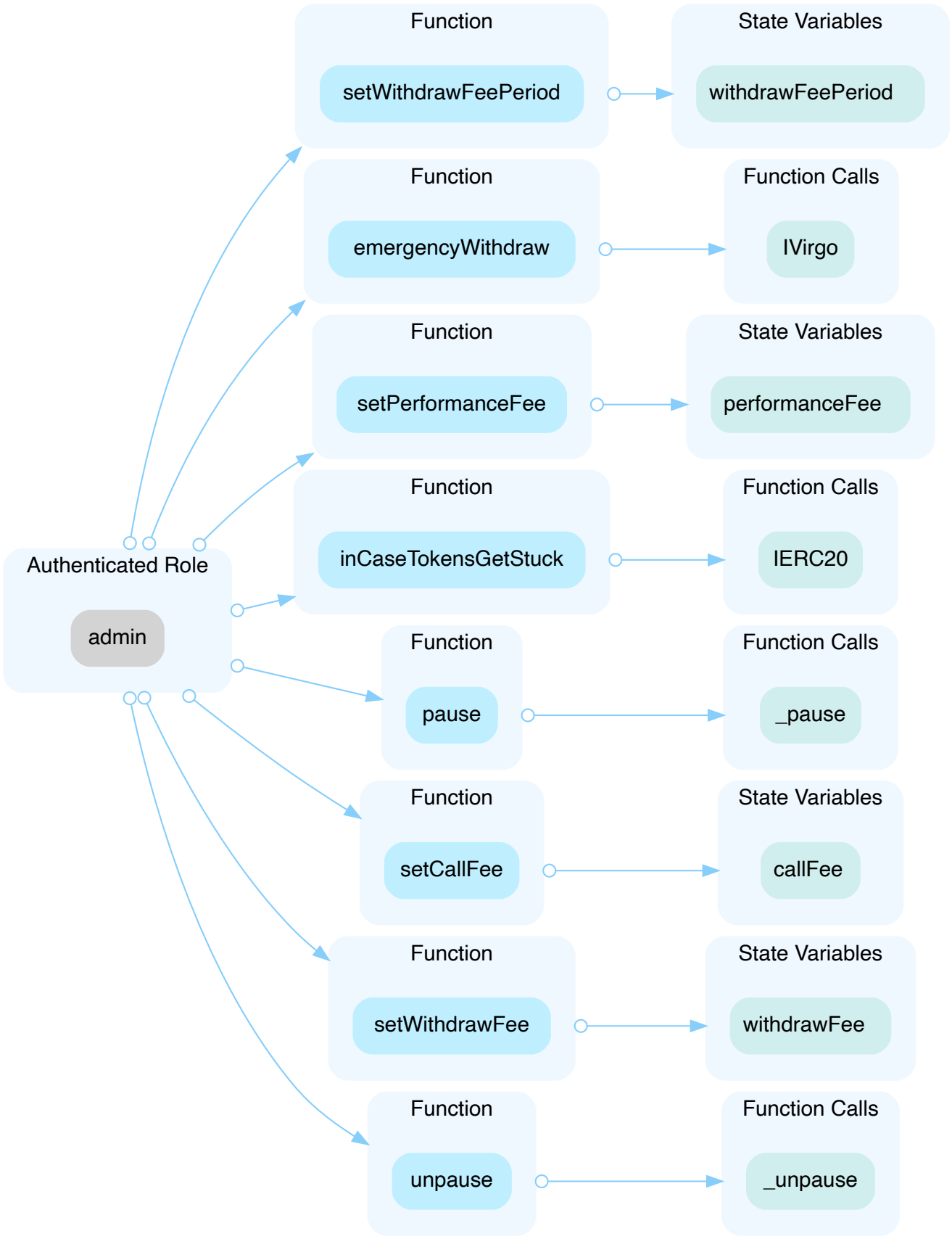
Category	Severity	Location	Status
Centralization / Privilege	Major	Acent Swap Smart Contract/Ade-Farm/contracts/ADEVault.sol: 175, 185, 195, 205, 215, 225, 238, 245, 257, 266	Pending

Description

In the contract `ADEVault` 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 `ADEVault` the role `admin` has authority over the functions shown in the diagram below. Any compromise to the `admin` 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.

ADV-02 | Centralization Risks In `treasury` Address

Category	Severity	Location	Status
Logical Issue	Major	Acent Swap Smart Contract/Ade-Farm/contracts/ADEVault.sol: 159, 327	Pending

Description

The linked statement transfers the withdrawal/performance fee valued in the `ADE` token to the `treasury`. As a result, over time the `treasury` will accumulate a significant portion of `ADE` tokens. If the `treasury` is an EOA (Externally Owned Account), mishandling its private key can have devastating consequences for the project as a whole.

Recommendation

In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

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

ADV-03 | Unreasonable Charges

Category	Severity	Location	Status
Logical Issue	Discussion	Acent Swap Smart Contract/Ade-Farm/contracts/ADEVault.sol: 325~329	Pending

Description

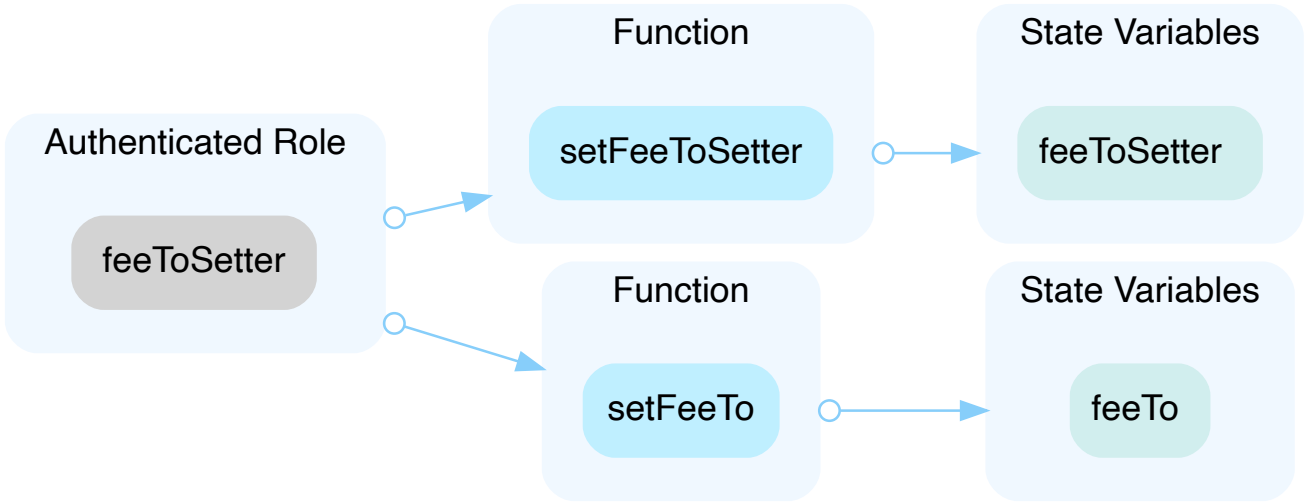
According to the logic of the linked boolean expression, there is a penalty if the time difference between redemption and the last pledge is within 3 days. However, the fee is still charged even though the withdrawal shares contain multiple parts which are deposited 3 days ago and shouldn't be punished. We think that is unreasonable. Is it in line with the original design logic?

AFC-01 | Centralization Risks In AcentFactory.sol

Category	Severity	Location	Status
Centralization / Privilege	Major	Acent Swap Smart Contract/ace-swap-core-sh/contracts/AcentFactory.sol: 52, 57	Pending

Description

In the contract `AcentFactory` the role `feeToSetter` has authority over the functions shown in the diagram below. Any compromise to the `feeToSetter` 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*.

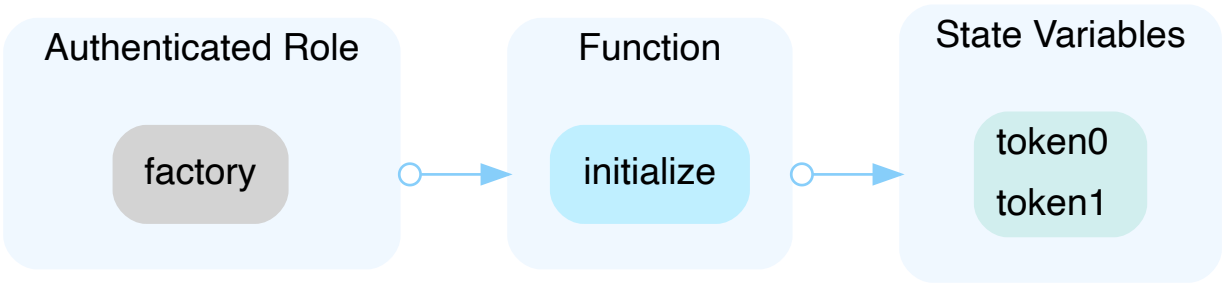
- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

APA-01 | Centralization Risks In AcentPair.sol

Category	Severity	Location	Status
Centralization / Privilege	Major	Acent Swap Smart Contract/ace-swap-core-sh/contracts/AcentPair.sol: 76	Pending

Description

In the contract `AcentPair` the role `factory` has authority over the functions shown in the diagram below. Any compromise to the `factory` 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.

APA-02 | Divide By Zero

Category	Severity	Location	Status
Logical Issue	Minor	Acent Swap Smart Contract/ace-swap-core-sh/contracts/AcentPair.sol: 153~155	Pending

Description

The call to `burn()` function will fail if the value of `totalSupply` is 0.

Recommendation

We advise the client to add the following validation in the function `burn()`

```
1 require(totalSupply != 0, "The value of totalSupply must not be 0");
```

APA-03 | Inconsistent Comment And Code

Category	Severity	Location	Status
Inconsistency	Discussion	Acent Swap Smart Contract/ace-swap-core-sh/contracts/AcentPair.sol: 98, 109	Pending

Description

According to the comment, the linked statement should be `uint denominator = rootK.mul(5).add(rootKLast);`. The current calculation let the mint liquidity be bigger than 1/6th of the growth in sqrt(k). Is it in line with the original design logic?

ASS-02 | Discussion About Swap Fees

Category	Severity	Location	Status
Logical Issue	Discussion	Acent Swap Smart Contract/ace-swap-core-sh/contracts/AcentPair.sol: 190~191; Acent Swap Smart Contract/deploy_periphery2/contracts/libraries/ADELibrary.sol: 43~59	Pending

Description

Currently, when swap tokens through the acent routers, the swap fee is 3‰, but in the acent pairs it uses 2‰ to check the balance. They are not the same. Is this as expected? If so, users can write their own router and set the swap fee as 2‰ to avoid paying more fees.

Recommendation

Please provide more information about the design logic.

CAB-01 | Missing Zero Address Validation

Category	Severity	Location	Status
Volatile Code	Minor	Acent Swap Smart Contract/ace-swap-core-sh/contracts/AcentFactory.sol: 28, 54, 59; Acent Swap Smart Contract/ace-swap-core-sh/contracts/AcentPair.sol: 78, 79; POSA Smart Contract/contracts/FundDistributor.sol: 19, 29	Pending

Description

Addresses should be checked before assignment or external call to make sure they are not zero addresses.

```
18          (bool success, bytes memory returnData) =
router.delegatecall(abi.encodeWithSelector(
19              IADERouter01(router).swapExactTokensForTokens.selector, amountIn,
amountOutMin, path, to, deadline
20          ));
```

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

```
19          ubi = _ubi;
```

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

```
28          feeToSetter = _feeToSetter;
```

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

```
29          ubi = _newUBI;
```

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

```
33          (bool success, bytes memory returnData) =
router.delegatecall(abi.encodeWithSelector(
34              IADERouter01(router).swapTokensForExactTokens.selector, amountOut,
amountInMax, path, to, deadline
35          ));
```

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

```
47          (bool success, bytes memory returnData) =
router.delegatecall(abi.encodeWithSelector(
48              IADERouter01(router).swapExactETHForTokens.selector, amountOutMin, path,
to, deadline
49          ));
```

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

```
48          disliked = _validator;
```

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

```
54          feeTo = _feeTo;
```

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

```
59          feeToSetter = _feeToSetter;
```

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

```
 62          (bool success, bytes memory returnData) =
router.delegatecall(abi.encodeWithSelector(
 63              IADERouter01(router).swapTokensForExactETH.selector, amountOut,
amountInMax, path, to, deadline
 64          ));
```

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

```
 77          (bool success, bytes memory returnData) =
router.delegatecall(abi.encodeWithSelector(
 78              IADERouter01(router).swapExactTokensForETH.selector, amountIn,
amountOutMin, path, to, deadline
 79          ));
```

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

```
78          token0 = _token0;
```

- `_token0` is not zero-checked before being used.

```
79          token1 = _token1;
```

- `_token1` is not zero-checked before being used.

```
 91          (bool success, bytes memory returnData) =
router.delegatecall(abi.encodeWithSelector(
 92              IADERouter01(router).swapETHForExactTokens.selector, amountOut, path,
to, deadline
 93          ));
```

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

Recommendation

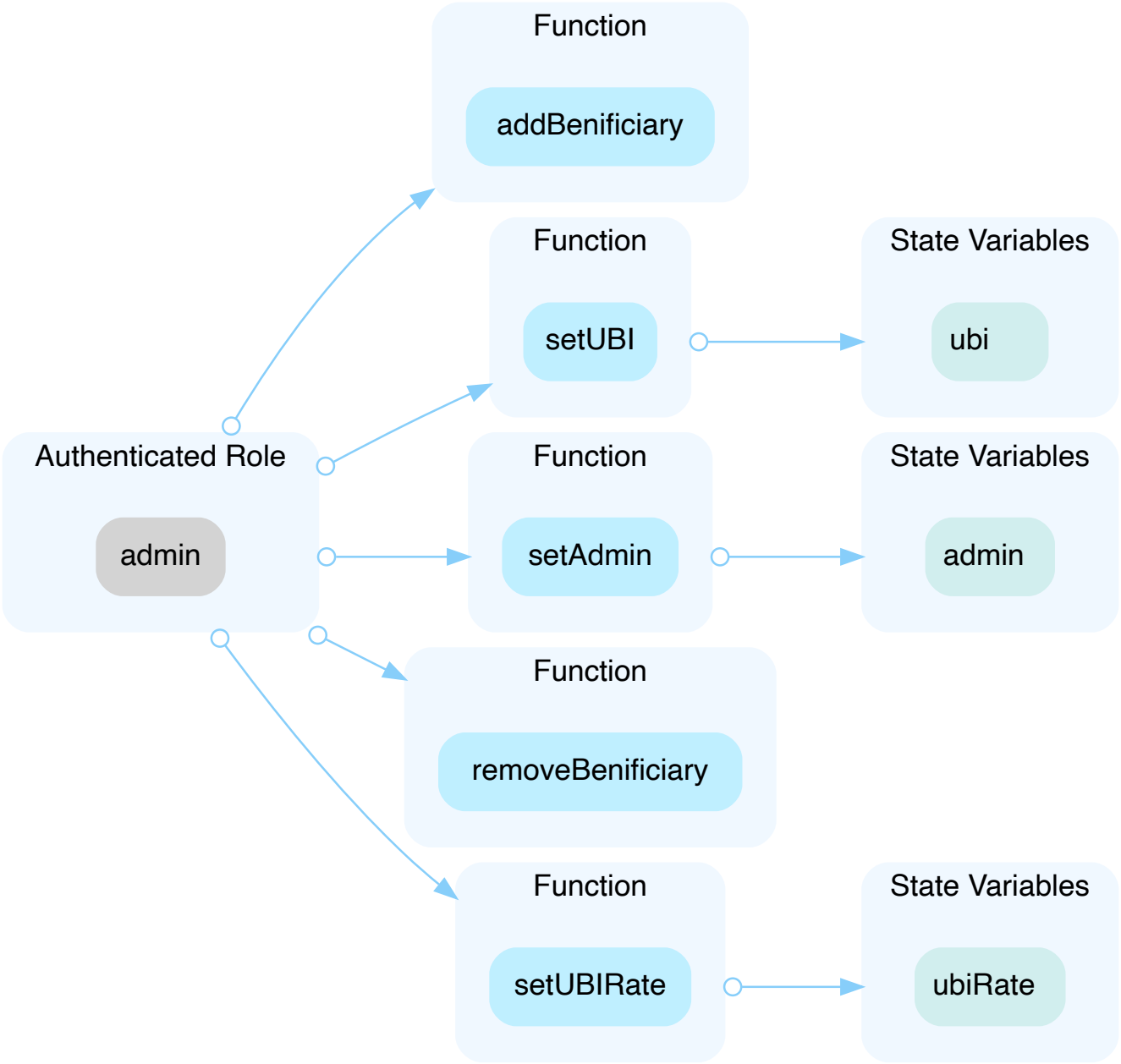
We advise adding a zero-check for the passed-in address value to prevent unexpected errors.

FDP-01 | Centralization Risks In FundDistributor.sol

Category	Severity	Location	Status
Centralization / Privilege	Major	POSA Smart Contract/contracts/FundDistributor.sol: 28, 32, 37, 42 , 49	Pending

Description

In the contract `FundDistributor` the role `admin` has authority over the functions shown in the diagram below. Any compromise to the `admin` 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.

FDP-02 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	POSA Smart Contract/contracts/FundDistributor.sol: 42~60	Pending

Description

The duplicated benificiaries possibly appear in the `benificiaries` because there is no limitation for adding two same addresses by calling the `addBenificiary()` function. The benificiary may not be cleanly removed if the duplicated benificiaries really exist.

Recommendation

We advise refactoring the linked statements as below:

```
42 mapping(address => bool) private existed;
43
44 function addBenificiary(address payable _benificiary) public onlyAdmin {
45     require(_benificiary != address(0), "Invalid benificiary address");
46     require(!existed[_benificiary], "benificiary is already added");
47
48     benificiaries.push(_benificiary);
49     existed[_benificiary] = true;
50
51     emit AddBenificiary(_benificiary);
52 }
53
54 function removeBenificiary(address payable _benificiary) public onlyAdmin {
55     require(_benificiary != address(0), "Invalid benificiary address");
56     require(existed[_benificiary], "benificiary is not added");
57
58     for (uint256 i = 0; i < benificiaries.length; ++i) {
59         if (benificiaries[i] == _benificiary) {
60             benificiaries[i] = benificiaries[benificiaries.length - 1];
61             benificiaries.pop();
62
63             existed[_benificiary] = false;
64
65             emit RemoveBenificiary(_benificiary);
66             break;
67         }
68     }
69 }
```

MAA-01 | Potential Risk Of Low-level Call

Category	Severity	Location	Status
Logical Issue, Language Specific	Minor	Acent Swap Smart Contract/Ade-Farm/contracts/libs/Multicall2.sol: 20~28, 56~67	Pending

Description

Anyone can call the two functions, `aggregate()` and `tryAggregate()` because they don't have access limitations. The attackers may successfully access such the `onlyOwner/onlyAdmin` functions of the target contract with the malicious `callData` if the `Muticall2` contract is set to be the `owner/admin` of the `target` contract. In this case, the contract is attacked.

Recommendation

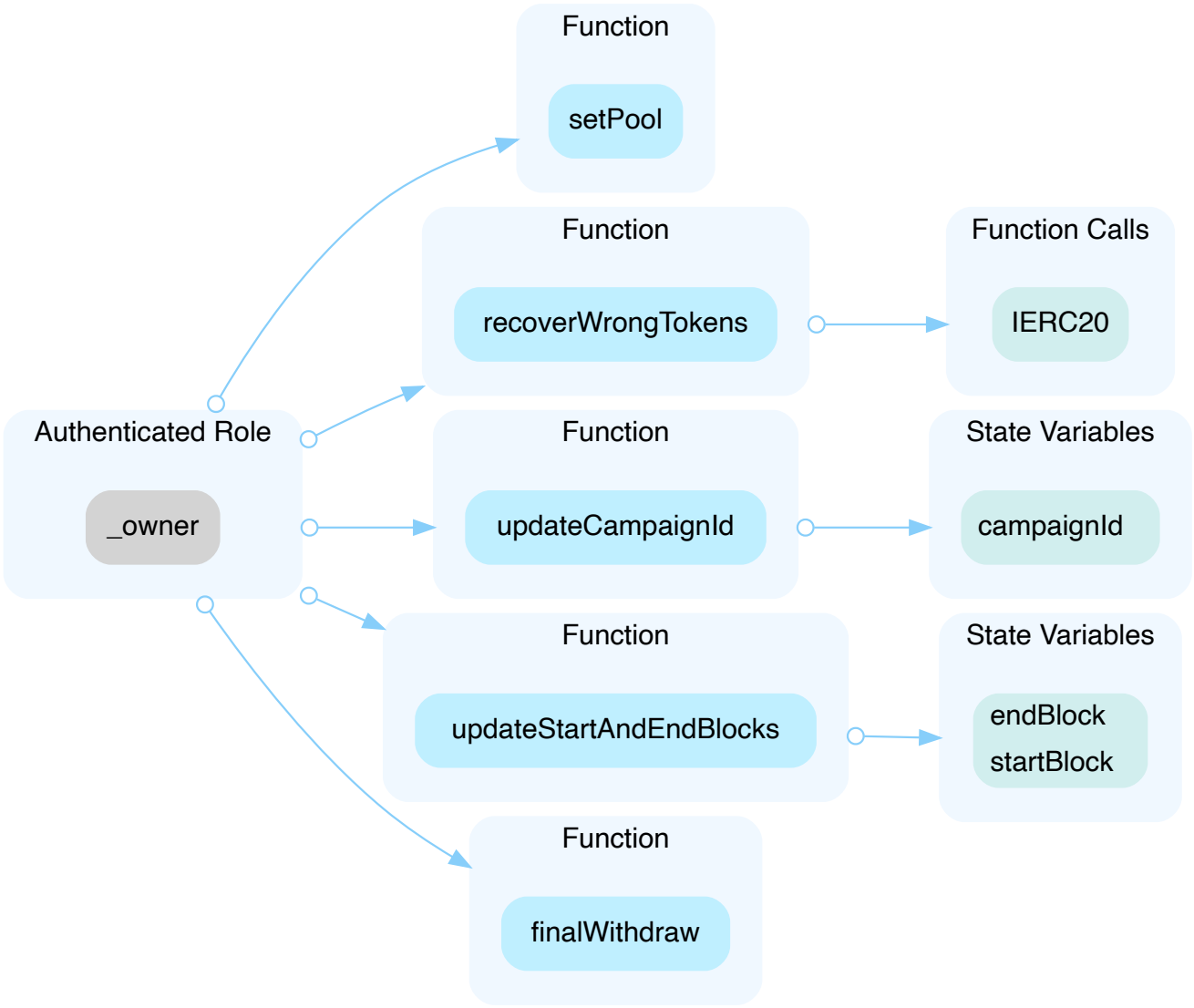
We advise not to set the `Multicall2` contract as the roles of other contracts who have special privileges.

SAA-01 | Centralization Risks In Supercluster.sol

Category	Severity	Location	Status
Centralization / Privilege	Major	Acent Swap Smart Contract/Ade-Farm/contracts/Supercluster.sol: 214 , 235, 254, 277, 292	Pending

Description

In the contract `Supercluster` 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.

SAA-02 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	Acent Swap Smart Contract/Ade-Farm/contracts/Supercluster.sol: 111~113	Pending

Description

The given input is missing the check for the validation and the zero-address.

Recommendation

We advise adding the check for the passed-in values to prevent unexpected errors as below:

```
111 require(block.number < _startBlock, "startBlock must be higher than current block");
112 require(_startBlock < _endBlock, "endBlock is less than or equals startBlock");
113 require(address(0) != _adminAddress, "set admin address to the zero address");
114 startBlock = _startBlock;
115 endBlock = _endBlock;
```

SAA-03 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Informational	Acent Swap Smart Contract/Ade-Farm/contracts/Supercluster.sol: 363~367, 381~388, 402~419	Pending

Description

The given input is missing the check for the array size.

Recommendation

We advise adding the check for the passed-in values to prevent unexpected errors as below:

```
363 require(_pids.length < numberPools, "only 2 pools exist");
364 uint256[] memory allocationPools = new uint256[](_pids.length);
365 for (uint8 i = 0; i < _pids.length; i++) {
366     require(_pids[i] < numberPools, "Non valid pool id");
367     allocationPools[i] = _getUserAllocationPool(_user, _pids[i]);
368 }
369 return allocationPools;
```

```
381 require(_pids.length < numberPools, "only 2 pools exist");
382 uint256[] memory amountPools = new uint256[](_pids.length);
383 bool[] memory statusPools = new bool[](_pids.length);
384
385 for (uint8 i = 0; i < _pids.length; i++) {
386   require(_pids[i] < numberPools, "Non valid pool id");
387   amountPools[i] = _userInfo[_user][_pids[i]].amountPool;
388   statusPools[i] = _userInfo[_user][_pids[i]].claimedPool;
389 }
390 return (amountPools, statusPools);
```

```
402 require(_pids.length < numberPools, "only 2 pools exist");
403 uint256[3][] memory amountPools = new uint256[3][](_pids.length);
404
405 for (uint8 i = 0; i < _pids.length; i++) {
406     require(_pids[i] < numberPools, "Non valid pool id");
407     uint256 userOfferingAmountPool;
408     uint256 userRefundingAmountPool;
409     uint256 userTaxAmountPool;
410
411     if (_poolInformation[_pids[i]].raisingAmountPool > 0) {
412         (
```

```
413             userOfferingAmountPool,
414             userRefundingAmountPool,
415             userTaxAmountPool
416         ) = _calculateOfferingAndRefundingAmountsPool(_user, _pids[i]);
417     }
418
419     amountPools[i] = [userOfferingAmountPool, userRefundingAmountPool,
userTaxAmountPool];
420 }
421 return amountPools;
```

SAA-04 | The Source Of The `offeringToken`

Category	Severity	Location	Status
Logical Issue	Discussion	Acent Swap Smart Contract/Ade-Farm/contracts/Supercluster.sol: 198, 223	Pending

Description

We only see the operations for transferring out `offeringToken` from the `Supercluster` contract. However, the balance of `offeringToken` in the `Supercluster` contract is zero. So what's the source of the `offeringToken` in the `Supercluster` contract.

Recommendation

Please provide more information about the design logic.

SAA-05 | Contract Usage Scenarios

Category	Severity	Location	Status
Logical Issue	Discussion	Acent Swap Smart Contract/Ade-Farm/contracts/Supercluster.sol: 11	Pending

Description

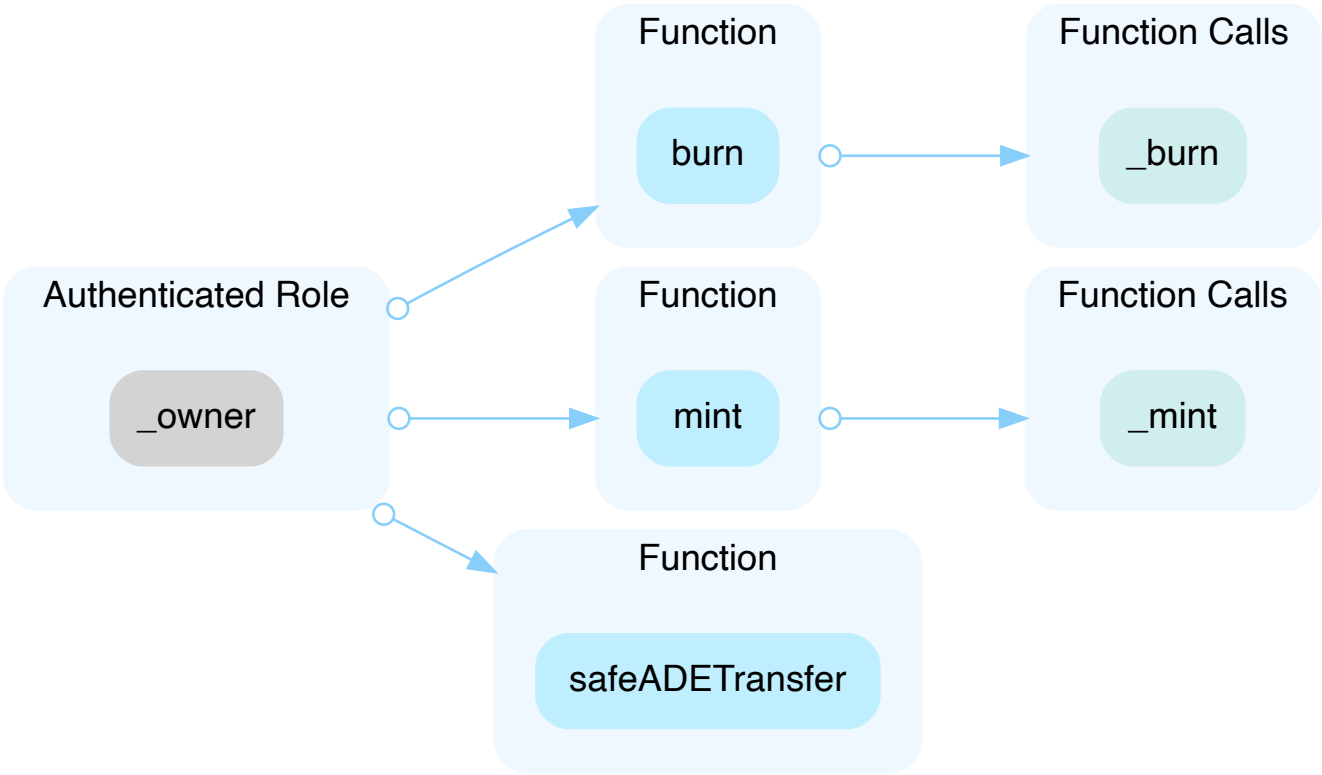
The contract attracts the users to deposit `LP` tokens and gives the `offeringToken` to the users as reward when the end block is reached. The users cannot get their all or a part of the deposited `LP` tokens back. What kind of scenarios would use this contract ?

SAF-01 | Centralization Risks In Spica.sol

Category	Severity	Location	Status
Centralization / Privilege	Major	Acent Swap Smart Contract/Ade-Farm/contracts/Spica.sol: 11, 15, 29	Pending

Description

In the contract `Spica` 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*.

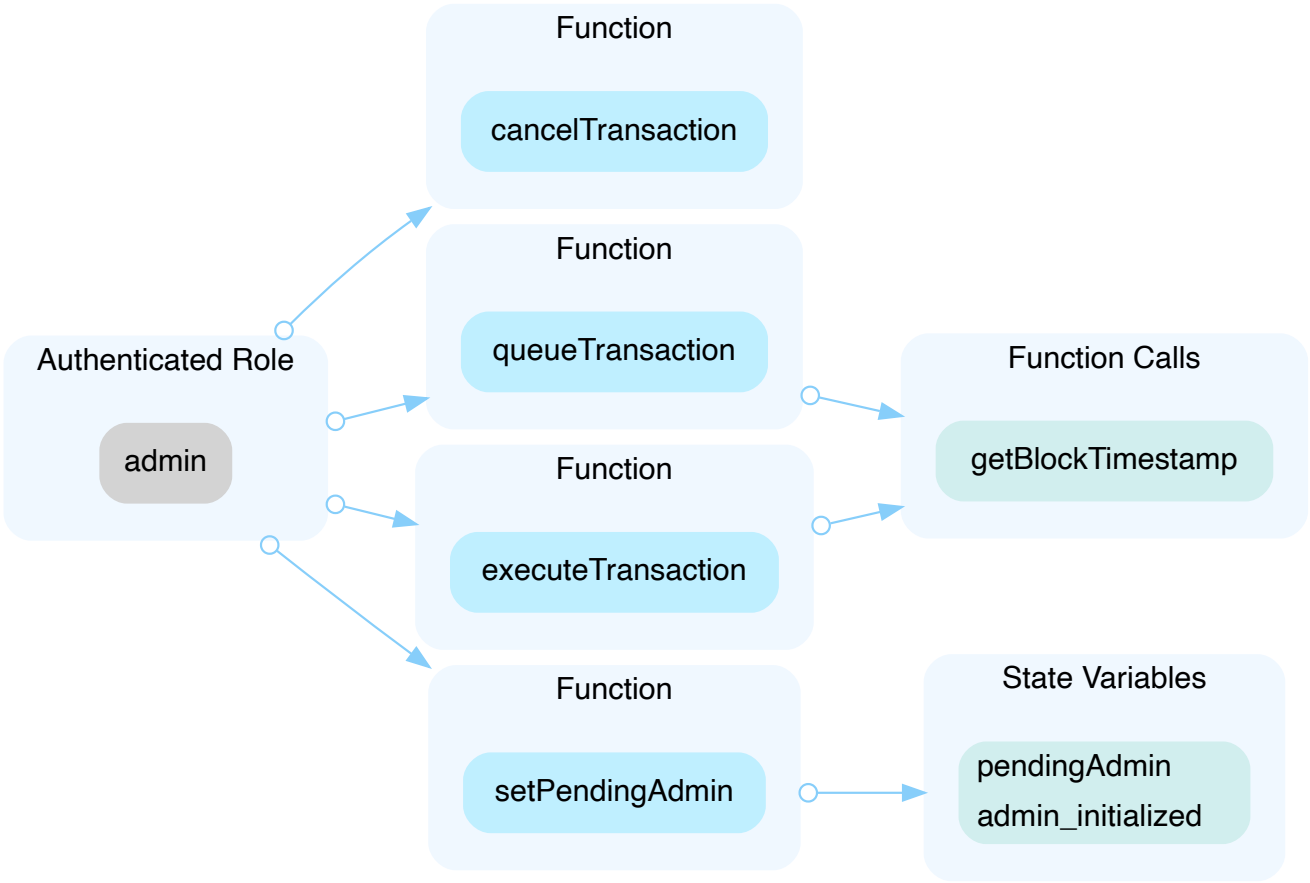
- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

TAF-01 | Centralization Risks In Timelock.sol

Category	Severity	Location	Status
Centralization / Privilege	Major	Acent Swap Smart Contract/Ade-Farm/contracts/Timelock.sol: 60, 68 , 81, 92, 101	Pending

Description

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



In the contract `Timelock` the role `pendingAdmin` has authority over the functions shown in the diagram below. Any compromise to the `pendingAdmin` 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

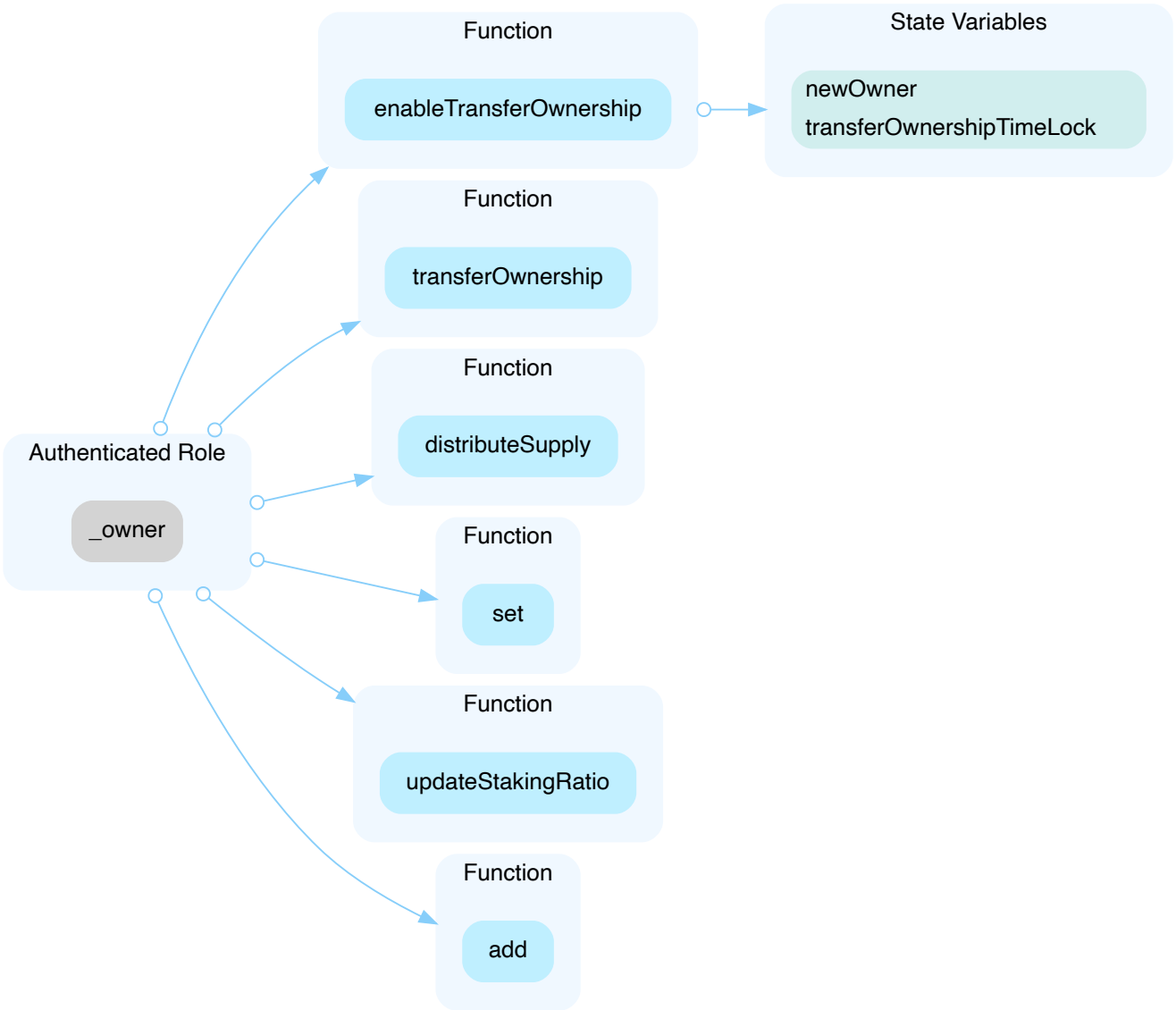
- Remove the risky functionality.

VAA-01 | Centralization Risks In VirgoAdmin.sol

Category	Severity	Location	Status
Centralization / Privilege	Major	Acent Swap Smart Contract/Ade-Farm/contracts/VirgoAdmin.sol: 34, 42, 50, 57, 63, 72	Pending

Description

In the contract `VirgoAdmin` 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.

VAA-03 | Function Cannot Call After Ownership Transfer

Category	Severity	Location	Status
Logical Issue	Discussion	Acent Swap Smart Contract/Ade-Farm/contracts/VirgoAdmin.sol: 39, 47, 54, 72~75	Pending

Description

The `transferOwnership()`/`add()`/`set()`/`distributeSupply()`/`updateStakingRatio()` functions called at L74/L39/L47/L54/L60 requires that the `msg.sender` must be the `owner` of the `Virgo` contract. We ensure that the `msg.sender` is the `VirgoAdmin` contract. So the aforementioned functions can be successfully called only if the `VirgoAdmin` contract is the `owner` of the `Virgo` contract. If the `owner` of the `Virgo` contract is changed to another address not the `VirgoAdmin` contract, the aforementioned functions will never be called. Is this design in line with the original design?

VAF-01 | Logic Flaw In `emergencyWithdraw()`

Category	Severity	Location	Status
Logical Issue	Critical	Acent Swap Smart Contract/Ade-Farm/contracts/Virgo.sol: 306	Pending

Description

When `msg.sender` calls `enterStaking()` , `spica` token will be minted to `msg.sender` when `pool.lpToken` is staked in the contract. However, if the `msg.sender` calls `emergencyWithdraw()`, the `pool.lpToken` can be transferred back to the `msg.sender` but the `spica` token that has been minted to the `msg.sender` will not be burnt. Therefore, `msg.sender` can call `enterStaking()` and `emergencyWithdraw()` repeatedly to ultimately mint a huge amount of `spica` token, with just the same amount of pool.lpToken.

Recommendation

We advise the client to burn the same amount of `spica` along with the withdraw of `pool.lpToken` when calling the `emergencyWithdraw()`. i.e:

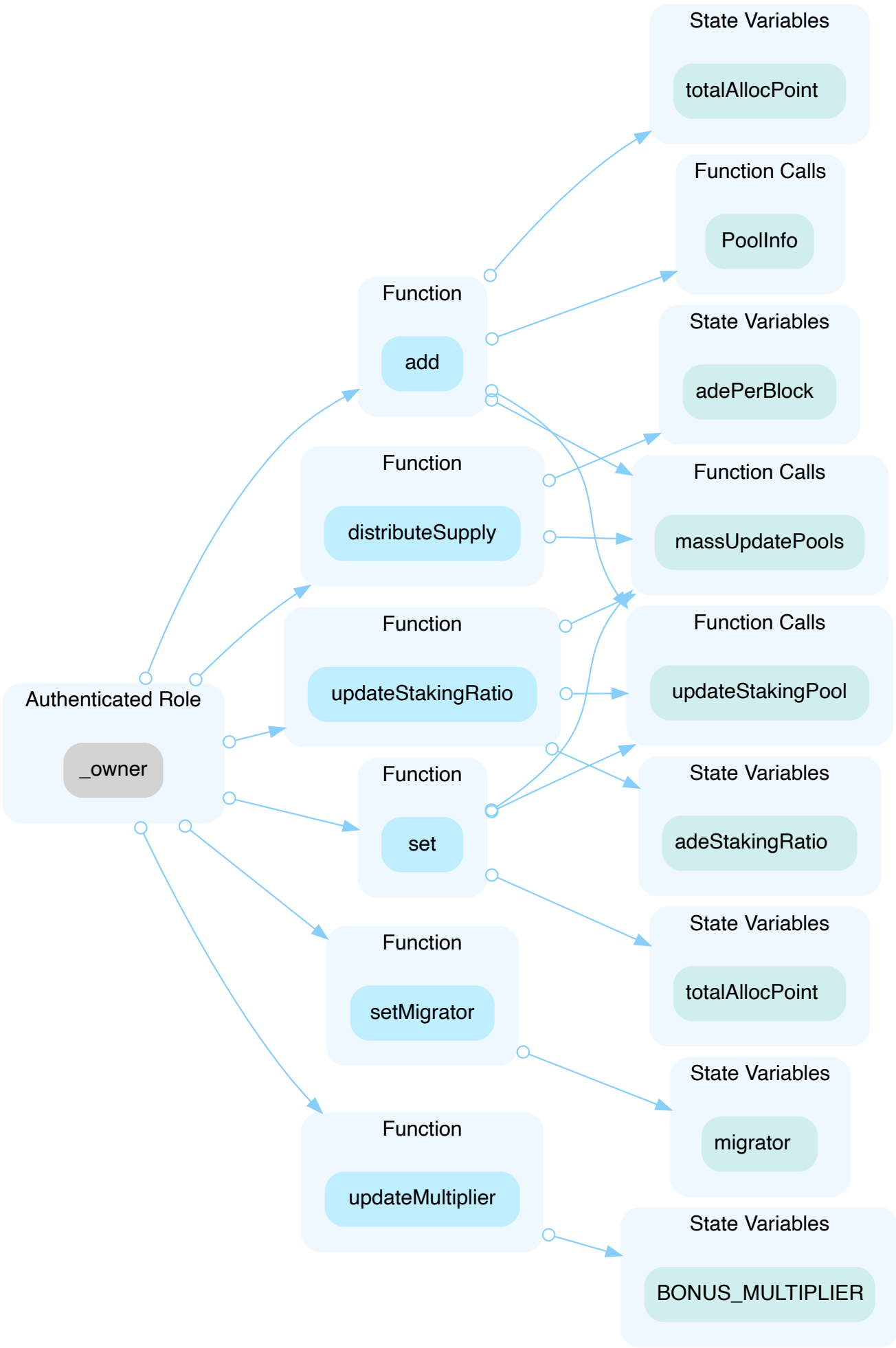
```
function emergencyWithdraw(uint256 _pid) public {
    PoolInfo storage pool = poolInfo[_pid];
    UserInfo storage user = userInfo[_pid][msg.sender];
    if(_pid == 0) {
        spica.burn(msg.sender, user.amount);
    }
    uint256 amount = user.amount;
    user.amount = 0;
    user.rewardDebt = 0;
    pool.lpToken.safeTransfer(address(msg.sender), amount);
    emit EmergencyWithdraw(msg.sender, _pid, amount);
}
```

VAF-02 | Centralization Risks In Virgo.sol

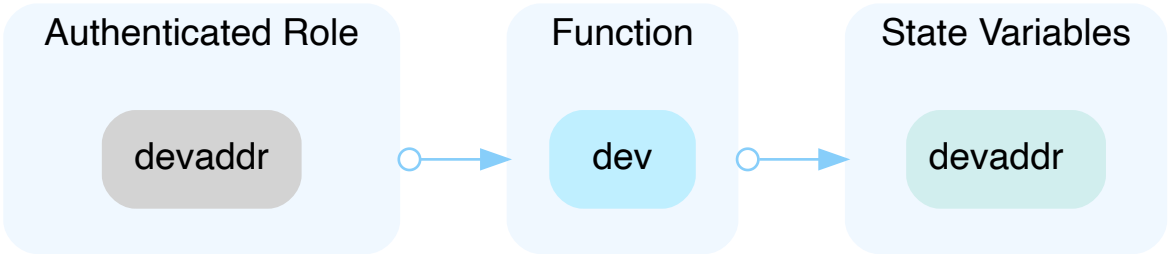
Category	Severity	Location	Status
Centralization / Privilege	**Major**	Acent Swap Smart Contract/Ade-Farm/contracts/Virgo.sol: 104, 114, 130, 156, 319, 324, 333	Pending

Description

In the contract `Virgo` 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
totalAllocPoint
Function Calls
PoolInfo
Function
add
State Variables
adePerBlock
Function
distributeSupply
Function Calls
massUpdatePools
Function
updateStakingRatio
Function Calls
updateStakingPool
Authenticated Role
_owner
Function
set
State Variables
adeStakingRatio
State Variables
totalAllocPoint
Function
setMigrator
State Variables
migrator
Function
updateMultiplier
State Variables
BONUS_MULTIPLIER

In the contract `Virgo` the role `devaddr` has authority over the functions shown in the diagram below. Any compromise to the `devaddr` 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.

VAF-03 | Unknown Implementation Of `migrator.migrate()`

Category	Severity	Location	Status
Logical Issue	Minor	Acent Swap Smart Contract/Ade-Farm/contracts/Virgo.sol: 167	Pending

Description

`setMigrator()` function can set `migrator` contract to any contract that is implemented from `IMigratorChef` interface by `owner`. As result, invocation of `migrator.migrate()` in function `migrate()` may bring dangerous effects as it is unknown to the user. However, the project may lose the ability to upgrade and migrate if `setMigrator()` and `migrate()` are removed.

Recommendation

Please provide more details on how to prevent abusage of the migrate functionality.

VAF-04 | `add()` Function Not Restricted

Category	Severity	Location	Status
Logical Issue	Minor	Acent Swap Smart Contract/Ade-Farm/contracts/Virgo.sol: 114	Pending

Description

When the same LP token is added into a pool more than once in function `add()`, the total amount of reward in function `updatePool()` will be incorrectly calculated. The current implementation is relying on the operation correctness to avoid repeatedly adding the same LP token to the pool, as the function will only be called by the owner.

Recommendation

We recommend adding the check for ensuring whether the given pool for addition is a duplicate of an existing pool so that the pool addition is only successful when there is no duplicate. This can be done by using a mapping of `addresses` -> `booleans`, which can restrict the same address from being added twice.

VAF-07 | Inefficient Code Execution Path

Category	Severity	Location	Status
Logical Issue	Informational	Acent Swap Smart Contract/Ade-Farm/contracts/Virgo.sol: 135~139	Pending

Description

The linked statement at L135 will cause data copy operation from `memory` to `storage` space and need more gas fee to execute, however, it is unnecessarily executed if the values of `preAllocPoint` and `_allocPoint` are the same.

Recommendation

We advise refactoring the linked statements as below:

```
134 if (prevAllocPoint != _allocPoint) {
135     poolInfo[_pid].allocPoint = _allocPoint;
136     totalAllocPoint = totalAllocPoint.sub(prevAllocPoint).add(_allocPoint);
137     updateStakingPool();
138 }
```

VAF-08 | Lack Of Pool Validity Checks

Category	Severity	Location	Status
Logical Issue	Informational	Acent Swap Smart Contract/Ade-Farm/contracts/Virgo.sol	Pending

Description

There's no sanity check to validate if a pool exists.

Recommendation

We advise the client to adopt following modifier `validatePoolByPid` to functions `set()`, `deposit()`, `withdraw()`, `emergencyWithdraw()`, `migrate()`, `pendingADE()`, and `updatePool()`.

```
modifier validatePoolByPid(uint256 _pid) {
  require (_pid < poolInfo.length , "Pool does not exist") ;
  _;
}
```

WAC-01 | Function `transfer` Prevents Transfers To The Zero Address

Category	Severity	Location	Status
Logical Issue	Minor	Acent Swap Smart Contract/Ade-Farm/contracts/libs/WACE.sol: 40~42	Pending

Description

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

Recommendation

Ensure that the `transfer` function in contract `WCRO` reverts if invoked with a recipient address of zero.

WAC-02 | Function `transfer` Prevents Overflows In The Recipient's Balance

Category	Severity	Location	Status
Logical Issue	Minor	Acent Swap Smart Contract/Ade-Farm/contracts/libs/WACE.sol: 40~42	Pending

Description

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

Recommendation

Ensure that all calls of the form `transfer(recipient, amount)` fail if adding the value in `amount` to the token balance of address `recipient` leads to an overflow. One possibility to implement this is the use of `SafeMath.add` for updating the recipient's balance, as it reverts upon overflows.

WAC-03 | Function `transferFrom` Fails For Transfers From The Zero Address

Category	Severity	Location	Status
Logical Issue	Minor	Acent Swap Smart Contract/Ade-Farm/contracts/libs/WACE.sol: 44~61	Pending

Description

All calls of the form `transferFrom(from, dest, amount)` where the `from` address is zero, must fail.

Recommendation

Ensure that all calls of the form `transferFrom(from, dest, amount)` in contract `WCRO` revert if invoked with the zero address in its `from` parameter.

WAC-04 | Function `transferFrom` Fails For Transfers To The Zero Address

Category	Severity	Location	Status
Logical Issue	Minor	Acent Swap Smart Contract/Ade-Farm/contracts/libs/WACE.sol: 44~61	Pending

Description

All calls of the form `transferFrom(from, dest, amount)` where the `dest` address is zero, must fail.

Recommendation

Ensure that all calls of the form `transferFrom(from, dest, amount)` in contract `WCRO` revert if invoked with the zero address in its `dest` parameter.

WAC-05 | Function `transferFrom` Prevents Overflows In The Recipient's Balance

Category	Severity	Location	Status
Logical Issue	Minor	Acent Swap Smart Contract/Ade-Farm/contracts/libs/WACE.sol: 44~61	Pending

Description

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

Recommendation

Ensure that the `transferFrom` function reverts and thereby fails for invocations of the form `transferFrom(from, dest, amount)`, where transferring the value in `amount` would result in an overflow of the balance entry for the `dest` address.

WAC-06 | Missing `emit` Keyword

Category	Severity	Location	Status
Language Specific	Minor	Acent Swap Smart Contract/Ade-Farm/contracts/libs/WACE.sol: 58	Pending

Description

The `emit` keyword should be used when an event is triggered to print program log information.

Recommendation

We advise refactoring the linked statement as below:

```
58  emit Transfer(src, dst, wad);
```

WAE-01 | Function `transfer` Prevents Transfers To The Zero Address

Category	Severity	Location	Status
Logical Issue	Minor	Acent Swap Smart Contract/deploy_periphery2/contracts/WACE.sol: 56~58	Pending

Description

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

Recommendation

Ensure that the `transfer` function in contract `WACE` reverts if invoked with a recipient address of zero.

WAE-02 | Function `transfer` Prevents Overflows In The Recipient's Balance

Category	Severity	Location	Status
Logical Issue	Minor	Acent Swap Smart Contract/deploy_periphery2/contracts/WACE.sol: 56~58	Pending

Description

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

Recommendation

Ensure that all calls of the form `transfer(recipient, amount)` fail if adding the value in `amount` to the token balance of address `recipient` leads to an overflow. One possibility to implement this is the use of `SafeMath.add` for updating the recipient's balance, as it reverts upon overflows.

WAE-03 | Function `transferFrom` Fails For Transfers From The Zero Address

Category	Severity	Location	Status
Logical Issue	Minor	Acent Swap Smart Contract/deploy_periphery2/contracts/WACE.sol: 60~77	Pending

Description

All calls of the form `transferFrom(from, dest, amount)` where the `from` address is zero, must fail.

Recommendation

Ensure that all calls of the form `transferFrom(from, dest, amount)` in contract `WACE` revert if invoked with the zero address in its `from` parameter.

WAE-04 | Function `transferFrom` Fails For Transfers To The Zero Address

Category	Severity	Location	Status
Logical Issue	Minor	Acent Swap Smart Contract/deploy_periphery2/contracts/WACE.sol: 60~77	Pending

Description

All calls of the form `transferFrom(from, dest, amount)` where the `dest` address is zero, must fail.

Recommendation

Ensure that all calls of the form `transferFrom(from, dest, amount)` in contract `WACE` revert if invoked with the zero address in its `dest` parameter.

WAE-05 | Function `transferFrom` Prevents Overflows In The Recipient's Balance

Category	Severity	Location	Status
Logical Issue	Minor	Acent Swap Smart Contract/deploy_periphery2/contracts/WACE.sol: 60~77	Pending

Description

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

Recommendation

Ensure that the `transferFrom` function reverts and thereby fails for invocations of the form `transferFrom(from, dest, amount)`, where transferring the value in `amount` would result in an overflow of the balance entry for the `dest` address.

Optimizations

ID	Title	Category	Severity	Status
ASS-01	Variables That Could Be Declared As `constant`	Gas Optimization	Optimization	Pending
CAB-02	Improper Usage Of `public` And `external` Type	Gas Optimization	Optimization	Pending
SAF-02	Mutability Specifiers Missing	Gas Optimization	Optimization	Pending
VAA-02	Mutability Specifiers Missing	Gas Optimization	Optimization	Pending
VAF-05	Unused State Variable	Gas Optimization	Optimization	Pending
VAF-06	Mutability Specifiers Missing	Gas Optimization	Optimization	Pending

ASS-01 | Variables That Could Be Declared As `constant`

Category	Severity	Location	Status
Gas Optimization	Optimization	Acent Swap Smart Contract/Ade-Farm/contracts/libs/WACE.sol: 4, 5, 6; Acent Swap Smart Contract/deploy_periphery2/contracts/WACE.sol: 19, 20, 21	Pending

Description

The linked variables could be declared as `constant` since these state variables are never modified.

Recommendation

We recommend to declare these variables as `constant`.

CAB-02 | Improper Usage Of `public` And `external` Type

Category	Severity	Location	Status
Gas Optimization	Optimization	Acent Swap Smart Contract/Ade-Farm/contracts/ADEToken.sol: 14; Acent Swap Smart Contract/Ade-Farm/contracts/Spica.sol: 11, 15, 29; Acent Swap Smart Contract/Ade-Farm/contracts/Timelock.sol: 51, 68; Acent Swap Smart Contract/Ade-Farm/contracts/Virgo.sol: 104, 114, 130, 156, 161, 220, 242, 263, 284, 304, 319, 333; Acent Swap Smart Contract/Ade-Farm/contracts/VirgoAdmin.sol: 34, 42, 57, 63; Acent Swap Smart Contract/Ade-Farm/contracts/libs/Multicall2.sol: 32, 50; Acent Swap Smart Contract/Ade-Farm/contracts/libs/WACE.sol: 23, 34, 40; Acent Swap Smart Contract/deploy_periphery2/contracts/ADERouter.sol: 403, 407, 417; Acent Swap Smart Contract/deploy_periphery2/contracts/ADERouter01.sol: 261, 265, 269; Acent Swap Smart Contract/deploy_periphery2/contracts/RealAdeRouter.sol: 794, 798, 808; Acent Swap Smart Contract/deploy_periphery2/contracts/WACE.sol: 39, 50, 56; POSA Smart Contract/contracts/FundDistributor.sol: 37, 42, 49	Pending

Description

`public` functions that are never called by the contract could be declared as `external`. `external` functions are more efficient than `public` functions.

Recommendation

Consider using the external attribute for public functions that are never called within the contract.

SAF-02 | Mutability Specifiers Missing

Category	Severity	Location	Status
Gas Optimization	Optimization	Acent Swap Smart Contract/Ade-Farm/contracts/Spica.sol: 20	Pending

Description

The linked variables are assigned only once, either during their contract-level declaration or during the `constructor`'s execution.

Recommendation

For the former, we advise that the `constant` keyword is introduced in the variable declaration to greatly optimize the gas cost involved in utilizing the variable. For the latter, we advise that the `immutable` mutability specifier is set at the variable's contract-level declaration to greatly optimize the gas cost of utilizing the variables.

VAA-02 | Mutability Specifiers Missing

Category	Severity	Location	Status
Gas Optimization	Optimization	Acent Swap Smart Contract/Ade-Farm/contracts/VirgoAdmin.sol: 20	Pending

Description

The linked variables are assigned only once, either during their contract-level declaration or during the `constructor`'s execution.

Recommendation

For the former, we advise that the `constant` keyword is introduced in the variable declaration to greatly optimize the gas cost involved in utilizing the variable. For the latter, we advise that the `immutable` mutability specifier is set at the variable's contract-level declaration to greatly optimize the gas cost of utilizing the variables.

VAF-05 | Unused State Variable

Category	Severity	Location	Status
Gas Optimization	Optimization	Acent Swap Smart Contract/Ade-Farm/contracts/Virgo.sol: 56, 88, 213	Pending

Description

The linked state variable is never used within the `Virgo` contract.

Recommendation

We advise removing the unused variables.

VAF-06 | Mutability Specifiers Missing

Category	Severity	Location	Status
Gas Optimization	Optimization	Acent Swap Smart Contract/Ade-Farm/contracts/Virgo.sol: 52~54, 86~87	Pending

Description

The linked variables are assigned only once, either during their contract-level declaration or during the `constructor`'s execution.

Recommendation

For the former, we advise that the `constant` keyword is introduced in the variable declaration to greatly optimize the gas cost involved in utilizing the variable. For the latter, we advise that the `immutable` mutability specifier is set at the variable's contract-level declaration to greatly optimize the gas cost of utilizing the variables.

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.

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.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

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.