CERTIK

Preliminary Comments

Jax Network

Mar 12th, 2022

Table of Contents

Summary

This report has been prepared for Jax Network to discover issues and vulnerabilities in the source code of the Jax Network 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	Jax Network
Platform	BSC
Language	Solidity
Codebase	https://github.com/dream-well/jax-defi
Commit	ce1b509d29f668d551ef6fec4348ebdd83c54a01

Audit Summary

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

Vulnerability Summary

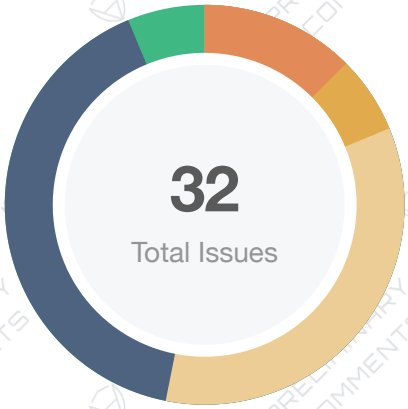
Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Mitigated	Resolved
Critical	0	0	0	0	0	0	0
Major	4	4	0	0	0	0	0
Medium	2	2	0	0	0	0	0
Minor	11	11	0	0	0	0	0
Informational	13	13	0	0	0	0	0
Discussion	2	2	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
LYJ	contracts/yield/LpYield.sol	9924d81d4756a4f0a1241b29d1c5bf8f45416dd8a7799ac2c9ee96f9fd766c37
TFW	contracts/yield/TxFeeWallet.sol	83a2703edcf268b7e352d85abc86c01b0d223d071af5ab0d7ee435fffd7224ee
UJN	contracts/yield/Ubi.sol	979528e65c22db424c5c5a534c54949663a36c2ff021b2fac9cc1ca2d32b412f
UTW	contracts/yield/UbiTaxWallet.sol	1e1b91a32b94787110dcc0ce4f9ca2db80f045b4087cc8ceb004e37ab67a8add
HSJ	contracts/HaberStornetta.sol	dc1414c7ac5e8c54ed758fd9160e498b689dcb0e5ecd22acccb13ebb88b654b8
JAX	contracts/JAXRE.sol	b9907c4b78362bc95284e68a8140c486e3a22f14aafcc8f6477e699fea6efe27
JAU	contracts/JAXUD.sol	7591673eee4a62869cf4b5c60b6ee042ae1d9cd6490698292678f5f2eeebba1c
JAJ	contracts/JaxAdmin.sol	683404173d2fbf776fea4e046421e5a8752b6a7fbcf136fb53bbf7b7d225210c
JLJ	contracts/JaxLibrary.sol	d59bd71e511aec697a3465fff8e4e730a95d9c8a78d02a97a764ee7b5abdeb20
JOJ	contracts/JaxOwnable.sol	279fdead3bfcde2e1eb937d6e1048bb1e436cffc216e766db01c0e3c9d22da5b
JPJ	contracts/JaxPlanet.sol	66faa9c4277e6367e03beb2d632ecb79a0aca4b3c20ab89dd89d6b1f81d0bec4
JSJ	contracts/JaxSwap.sol	d3002539ff69eb3a4bb8423aa1e799a7ae4e196dd7a1abb3fc49377851f17ba7
JTJ	contracts/JaxToken.sol	f6e17a1d0e8543d76b0cd8646945a975ea7fd5343af69c22eeb6ddd63cc025de
VRP	contracts/VRP.sol	65d6746afe6f98ab3a75252db794977f7d021ebc8cc394909fb4a854a7419cd4

ID	File	SHA256 Checksum
WJA	contracts/WJAX.sol	3bde08bed3c98d028dbdceeae6ddbad88b85200acfd0e7de2d7a4f699e9340b8

Findings



Critical	0 (0.00%)
Major	4 (12.50%)
Medium	2 (6.25%)
Minor	11 (34.38%)
Informational	13 (40.63%)
Discussion	2 (6.25%)

ID	Title	Category	Severity	Status
JAX-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Pending
JAX-02	Third Party Dependencies	Volatile Code	Minor	Pending
JAX-03	External Dependency	Logical Issue	Minor	Pending
JAJ-01	Pull-Over-Push Pattern in `JaxAdmin.updateGovernor()`	Logical Issue	Minor	Pending
JAJ-02	Privileged Ownership in `JaxAdmin.sol`	**Centralization / Privilege**	**Major**	Pending
JAJ-03	Naming Optimization	Coding Style	Informational	Pending
JAJ-04	Inconsistent Function Naming	Coding Style	Informational	Pending
JAJ-05	Lack of Sanity Check for Parameter `ratio`	Volatile Code	Minor	Pending
JNC-01	Divide Before Multiply	Mathematical Operations	Informational	Pending
JNC-02	Uninitialized Local Variable	Coding Style	Informational	Pending
JNC-03	Unused Return Value	Volatile Code	Informational	Pending
JNC-04	Locked Ether	Language Specific	Medium	Pending
JNC-05	Missing Zero Address Validation	Volatile Code	Minor	Pending

ID	Title	Category	Severity	Status
JNC-06	Redundant Code Components	Volatile Code	Informational	Pending
JNC-07	Missing Emit Events	Coding Style	Informational	Pending
JNC-08	Pull-Over-Push Pattern	Logical Issue	Minor	Pending
JNC-09	Inconsistent Comments and Code	Coding Style	Informational	Pending
JNC-10	Unused Return Value `jaxAdmin.system_status()`	Volatile Code	Informational	Pending
JNC-11	Potential Sandwich Attacks	Logical Issue	Minor	Pending
JNC-12	The Purpose of Function `withdrawByAdmin()`	Volatile Code	Discussion	Pending
JNC-13	Incorrect Error Message	Coding Style	Informational	Pending
JPJ-01	No Upper Limit for `min_transaction_tax`	**Centralization / Privilege**	**Major**	Pending
JSJ-01	Unused Local Variable in Modifirer `notContract`	Coding Style	Minor	Pending
JTJ-01	Unused `internal` Function	Volatile Code	Minor	Pending
LYJ-01	Uninitialized State Variable	Coding Style	Minor	Pending
LYJ-02	State Variable Should Be Declared Constant	Gas Optimization	Informational	Pending
LYJ-03	Potential Risk of Locked Funds	Volatile Code	Minor	Pending
LYJ-04	No Upper Limit for `liquidity_ratio_limit`	**Centralization / Privilege**	**Major**	Pending
UJN-01	Role Has The Same Naming But Different Meaning	Coding Style	Discussion	Pending
UTW-01	Incorrect `modifier` Naming	Coding Style	Informational	Pending
VRP-01	Dead Code	Coding Style	Informational	Pending
VRP-03	Function Call Will Never Succeed	Logical Issue	Medium	Pending

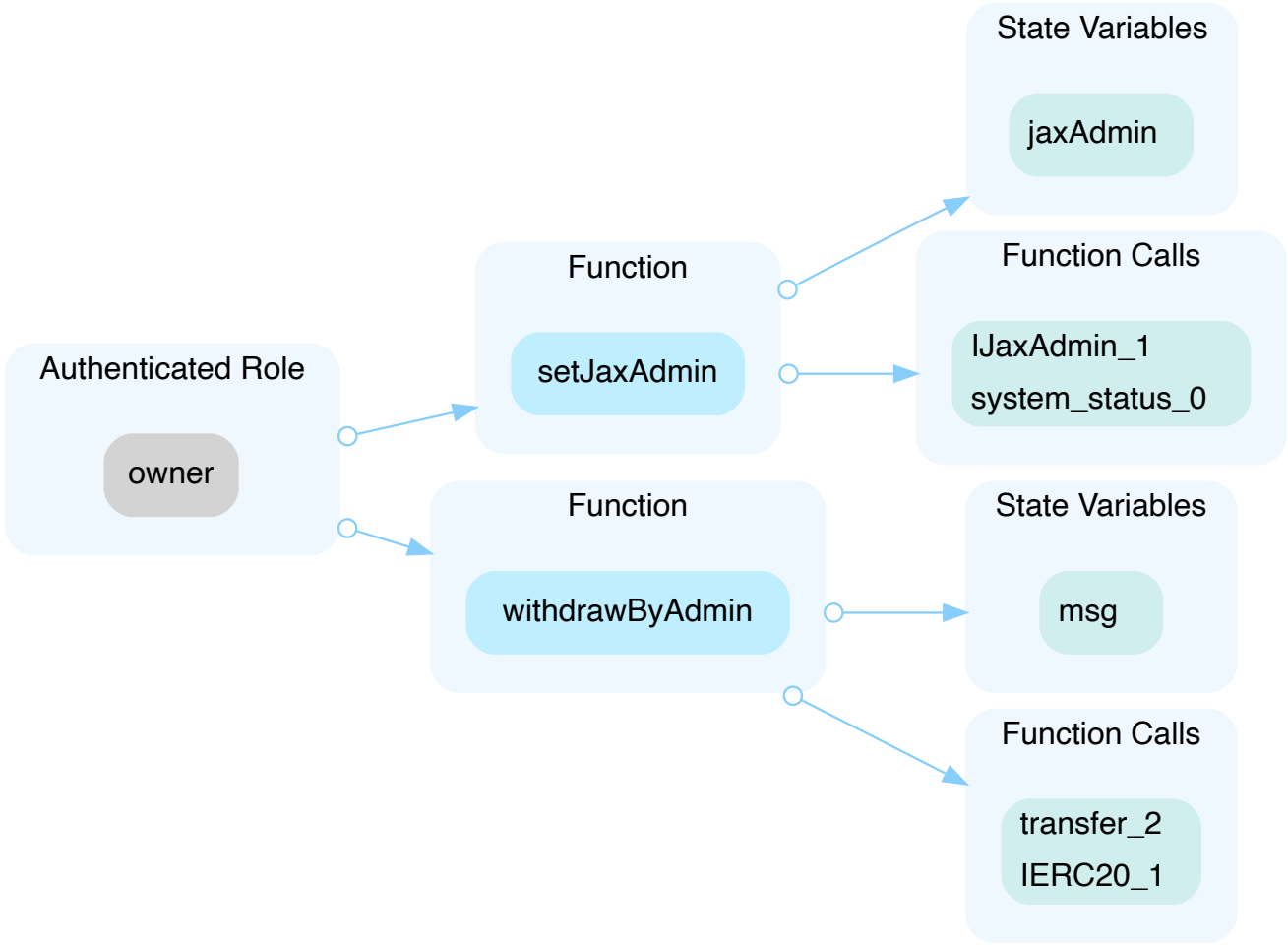
JAX-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	Major	Global	Pending

Description

In the contract `VRP` 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.



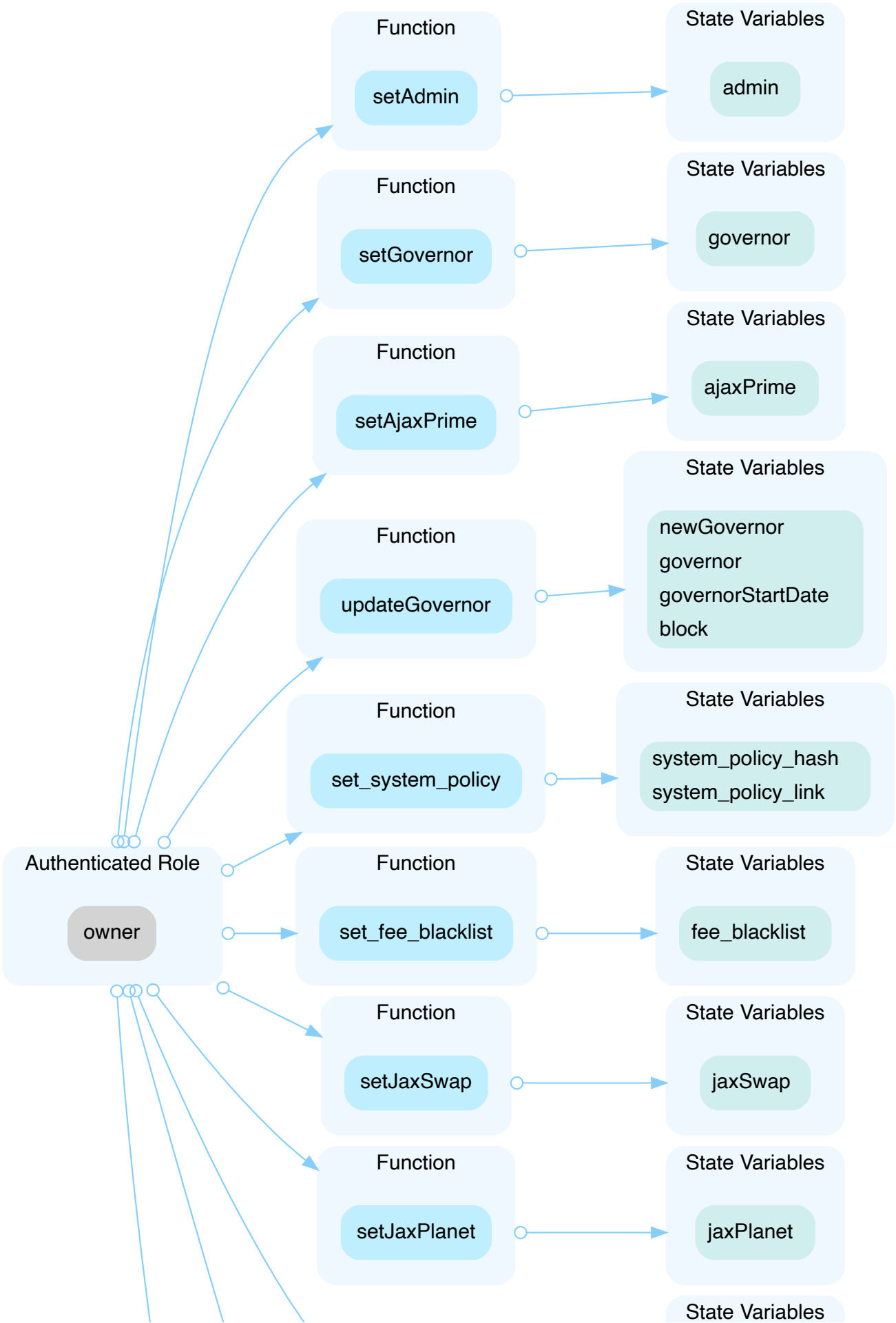
In the contract `VRP` the role `JaxAdmin.governor` has authority over the functions shown below.

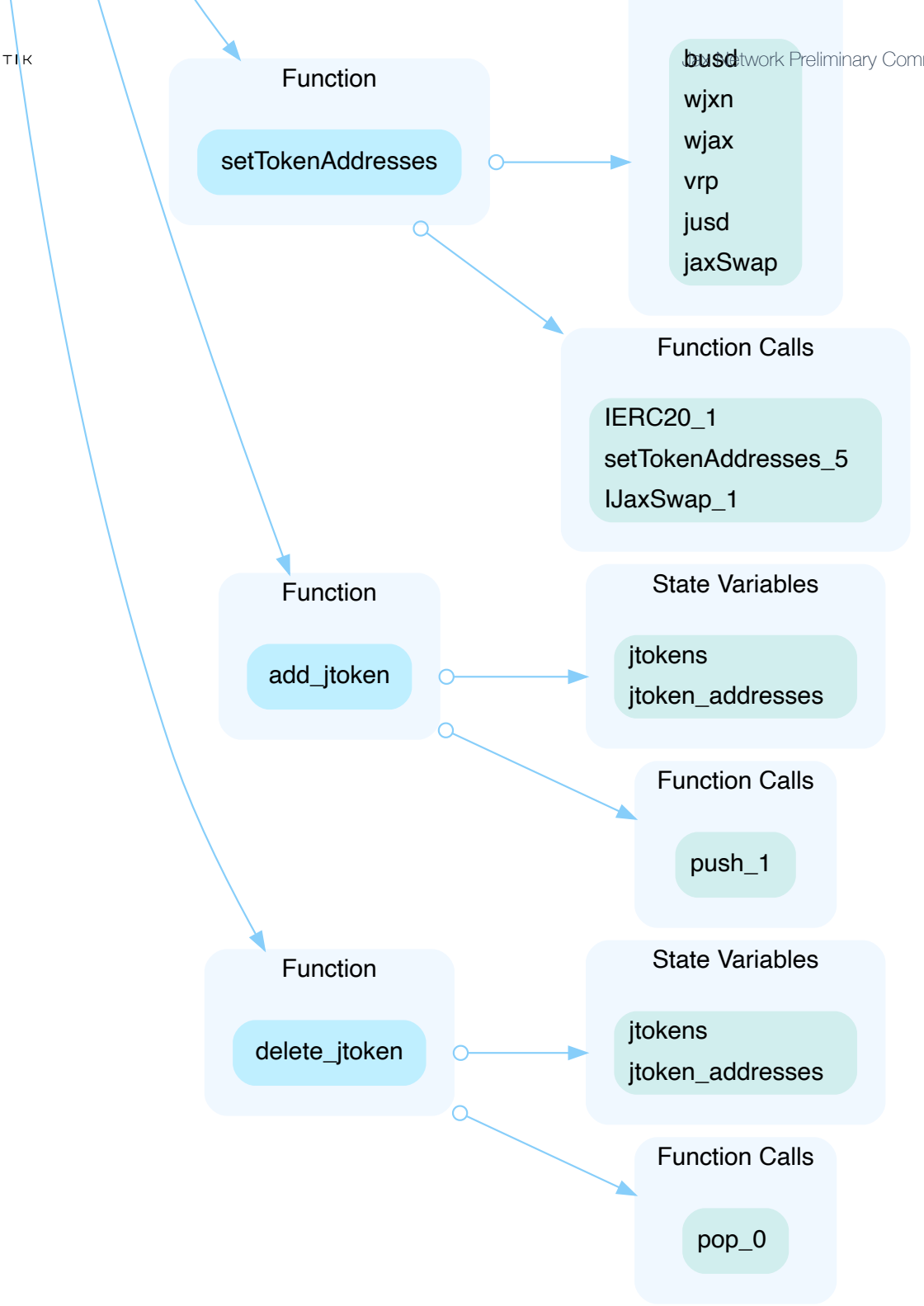
- function `deposit_reward()`
- function `set_reward_token()`

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

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

- function `setAdmin()`
- function `set_system_policy()`
- function `setJaxSwap()`
- function `setJaxPlanet()`
- function `setTokenAddresses()`

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

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

- function `setSystemStatus()`
- function `setOperators()`
- function `setWhitelistForOperator()`
- function `set_readme()`
- function `set_governor_policy()`
- function `set_blacklist()`
- function `setTransactionFee()`
- function `setReferralFee()`
- function `setCashback()`
- function `set_use_wjxn_usd_dex_pair()`
- function `set_use_wjax_usd_dex_pair()`
- function `set_freeze_vrp_wjxn_swap()`
- function `set_wjxn_wjax_collateralization_ratio()`
- function `set_wjax_collateralization_ratio()`
- function `set_wjax_jusd_markup_fee()`
- function `setPriceImpactLimit()`

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

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

- function `setGovernor()`
- function `setAjaxPrime()`
- function `updateGovernor()`
- function `set_fee_blacklist()`
- function `add_jtoken()`
- function `delete_jtoken()`

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

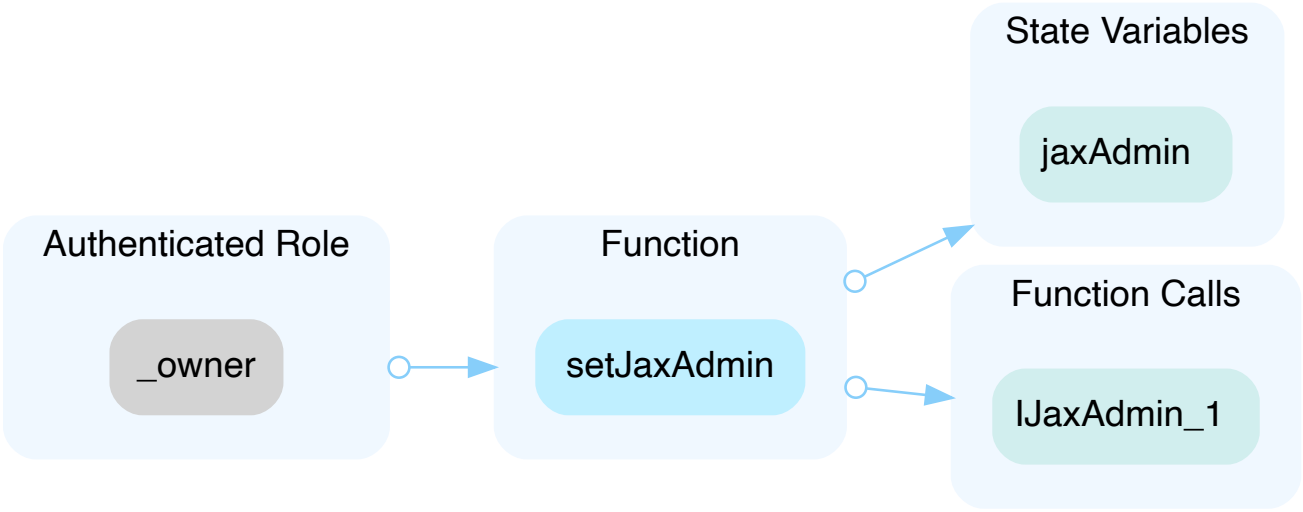
In the contract `JaxAdmin` the roles `operators` have authority over the functions listed below.

- function `set_jusd_jtoken_ratio()`
- function `set_wjxn_usd_ratio()`
- function `set_wjax_usd_ratio()`

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

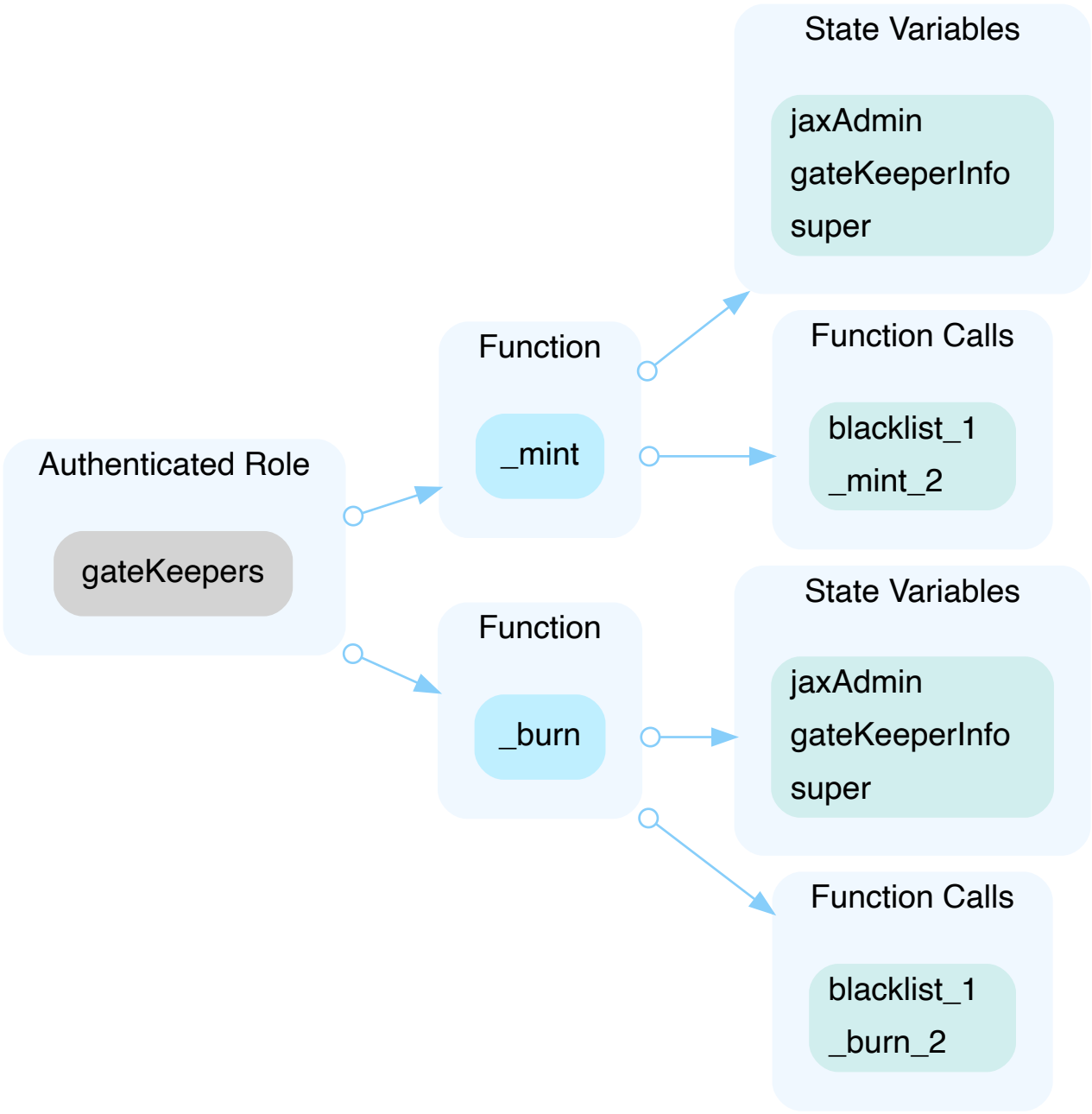
In the contract `WJAX` the role `_owner` has authority over the functions shown in the diagram below.

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



In the contract `WJAX` the role `gateKeepers` has authority over the functions shown in the diagram below.

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



In the contract `WJAX` the contract `JaxAdmin` has authority over the functions shown below.

- function `setTransactionFee()`
- function `setReferralFee()`
- function `setCashback()`

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

In the contract `WJAX` the role `JaxAdmin.ajaxPrime` has authority over the functions shown below.

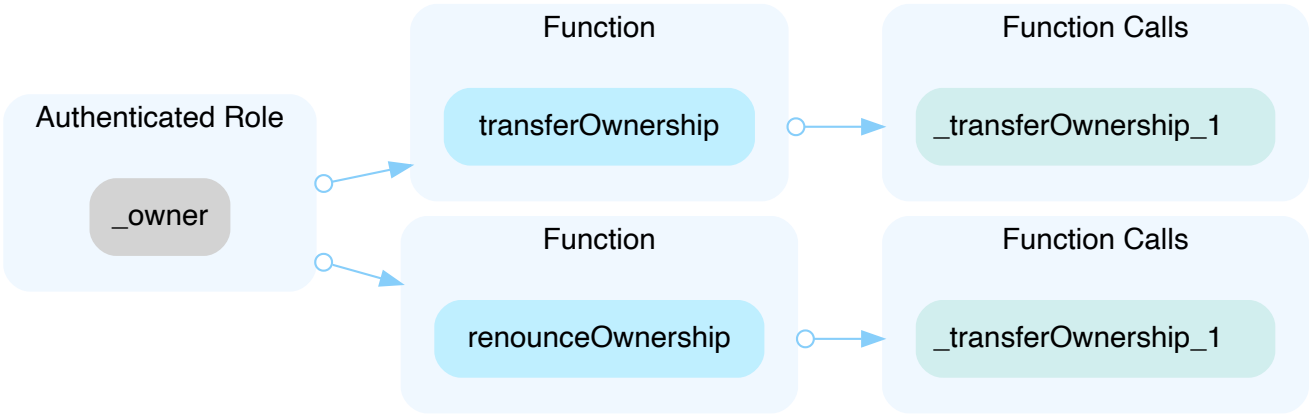
- function `setGateKeepers()`

- function `setMintBurnLimit()`

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

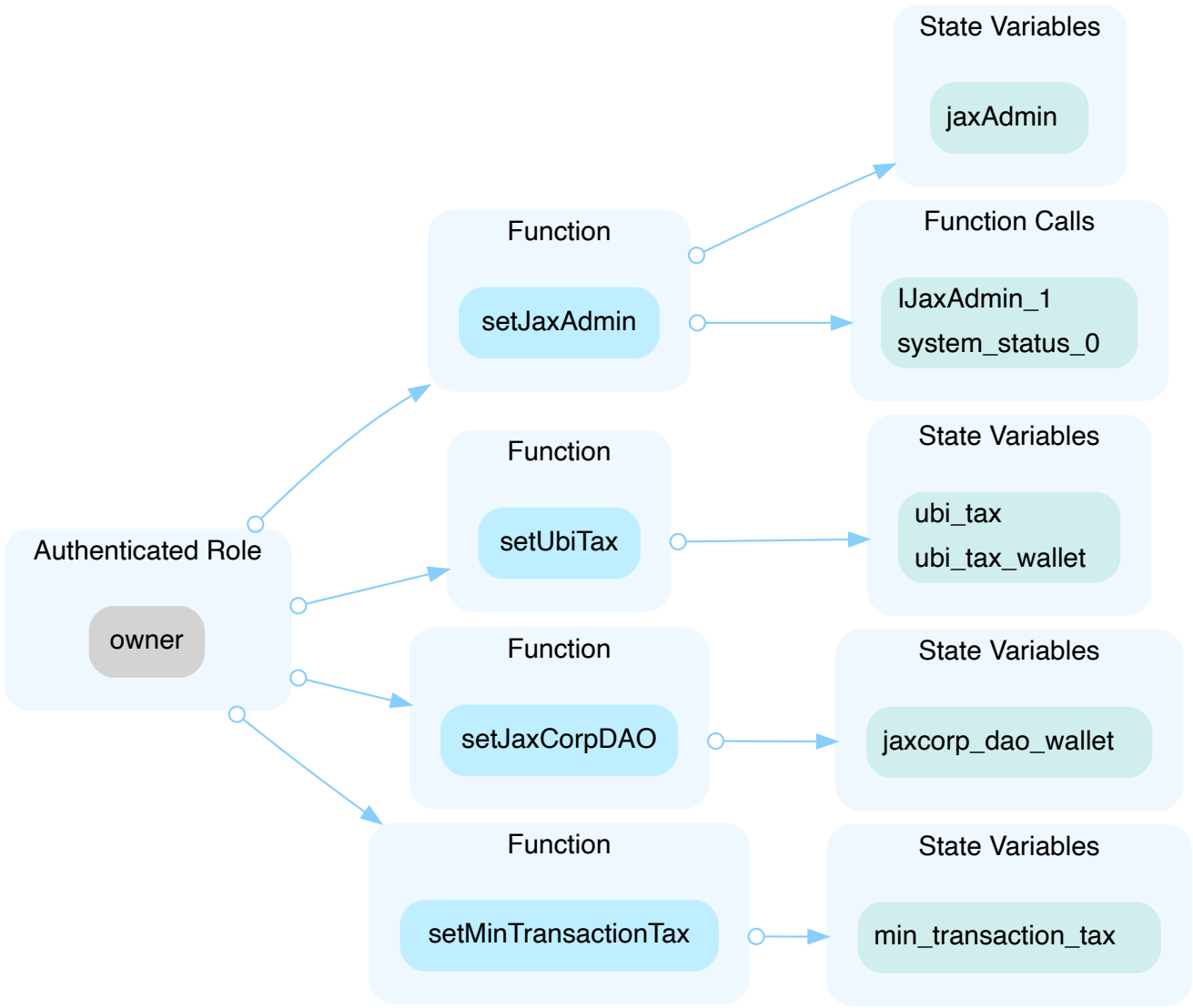
In the contract `JaxOwnable` the role `_owner` has authority over the functions shown in the diagram below.

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



In the contract `JaxPlanet` 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 `JaxPlanet` the role `JaxAdmin.admin` has authority over the functions listed below.

- function `setJaxAdmin()`

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

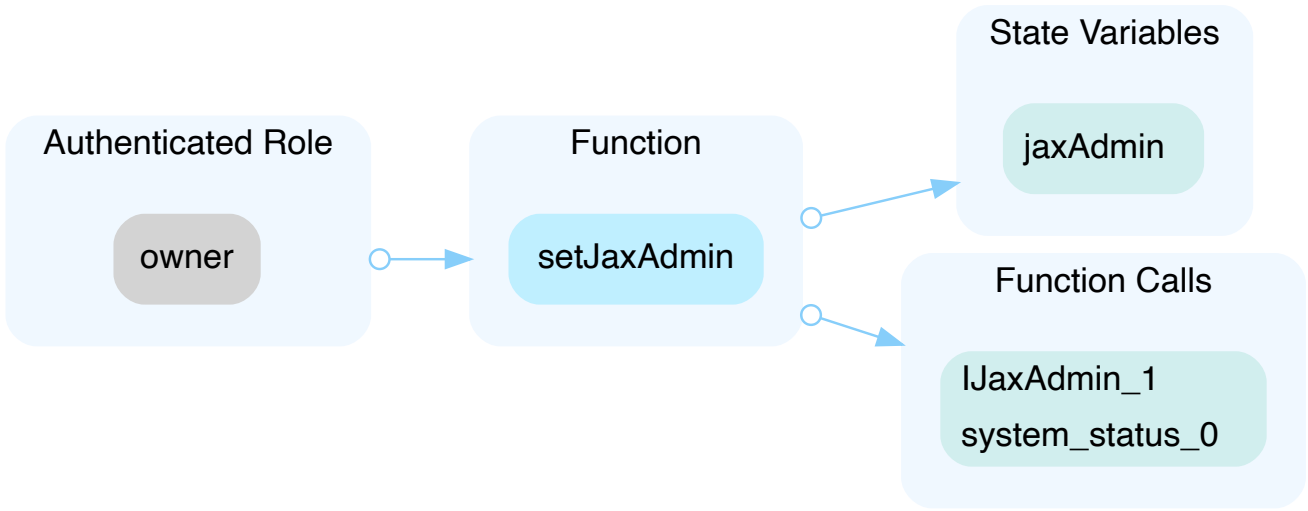
In the contract `JaxPlanet` the role `JaxAdmin.ajaxPrime` has authority over the functions listed below.

- function `setUbiTax()`
- function `setJaxCorpDAO()`
- function `setMinTransactionTax()`

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

In the contract `JaxSwap` 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 `JaxSwap` the role `JaxAdmin.admin` has authority over the functions listed below.

- function `setJaxAdmin()`

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

In the contract `JaxSwap` the role `JaxAdmin.governor` has authority over the functions listed below.

- function `swap_wjxn_wjax()`
- function `swap_wjax_wjxn()`

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

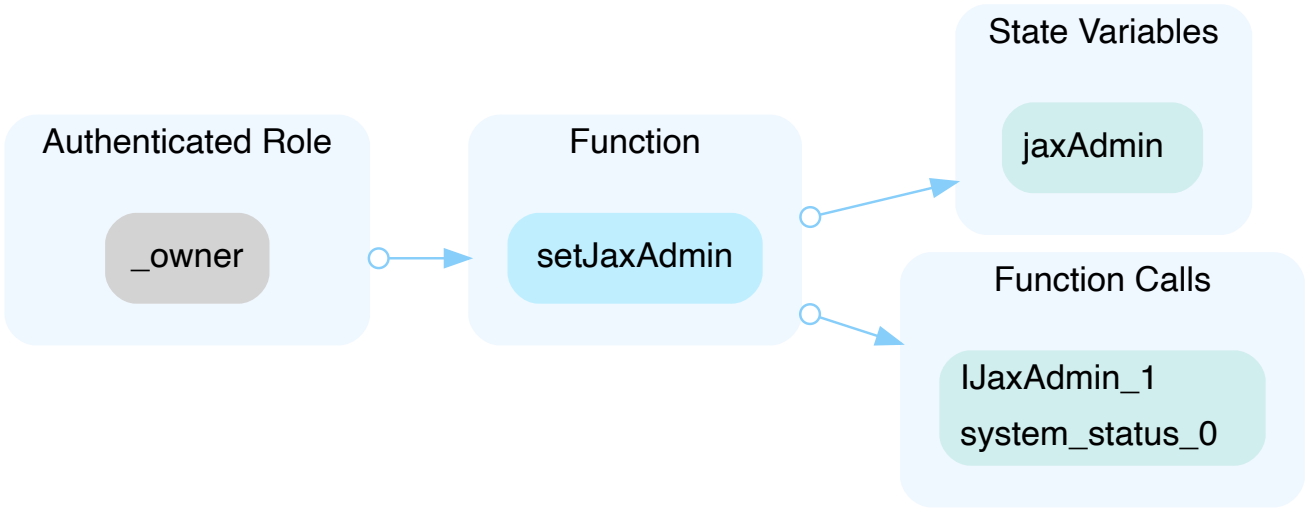
In the contract `JaxSwap` the contract `JaxAdmin` has authority over the functions listed below.

- function `setTokenAddresses()`

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

In the contract `JaxToken` 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.



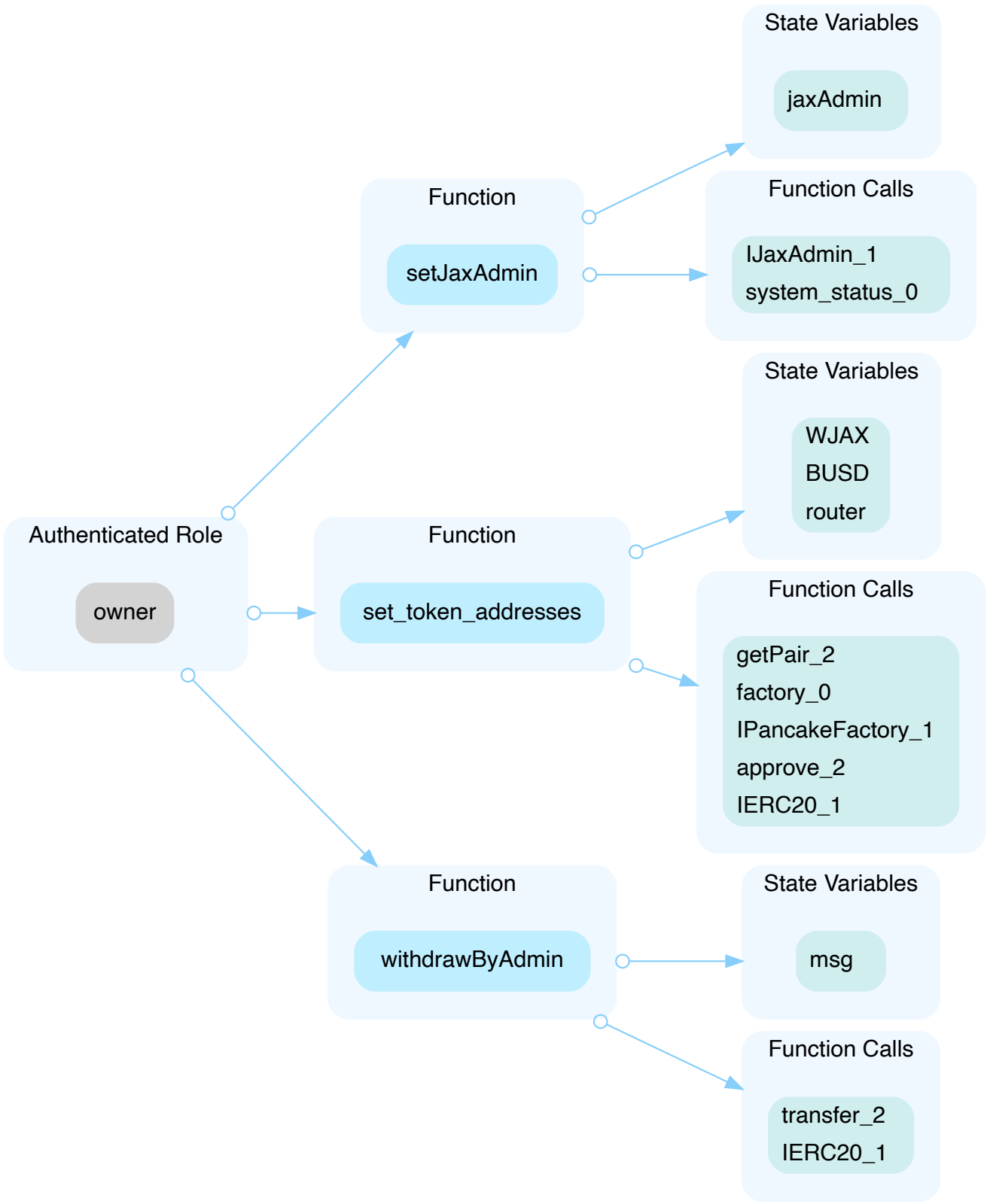
In the contract `JaxToken` the contract `JaxAdmin` has authority over the functions shown below.

- function `setTransactionFee()`
- function `setReferralFee()`
- function `setCashback()`

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

In the contract `LpYield` 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 `LpYield` the role `JaxAdmin.admin` has authority over the functions listed below.

- function `setJaxAdmin()`
- function `set_token_addresses()`

- function `withdrawByAdmin()`

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

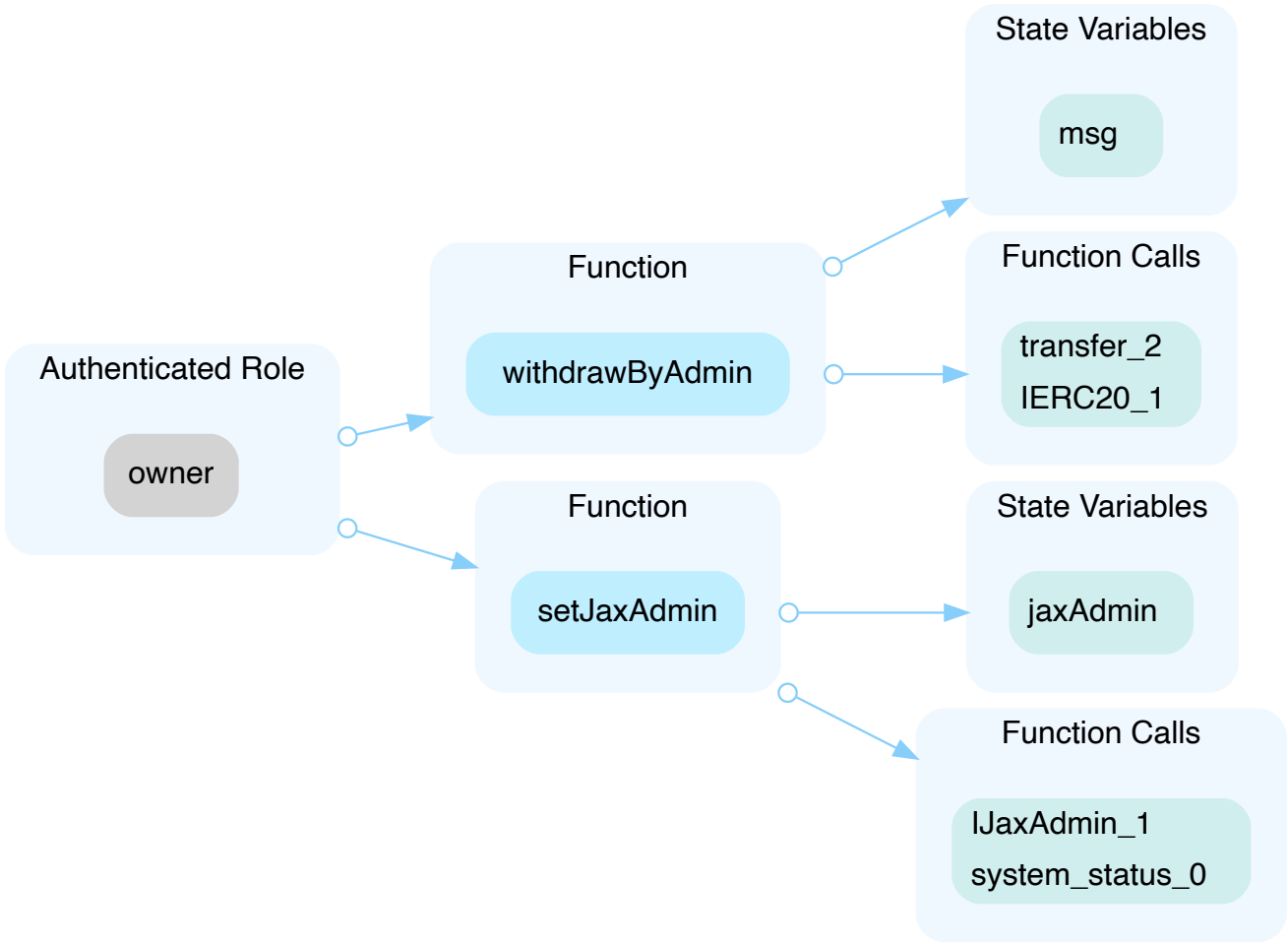
In the contract `LpYield` the role `JaxAdmin.governor` has authority over the functions listed below.

- function `set_reward_token()`
- function `set_busd_deposit_range()`
- function `set_deposit_fair_price_range()`
- function `set_withdraw_fair_price_range()`
- function `set_check_fair_price_deposit()`
- function `set_check_fair_price_withdraw()`
- function `set_liquidity_ratio_limit()`

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

In the contract `TxFeeWallet` 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 `LpYield` the role `JaxAdmin.admin` has authority over the functions listed below.

- function `withdrawByAdmin()`
- function `setJaxAdmin()`

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

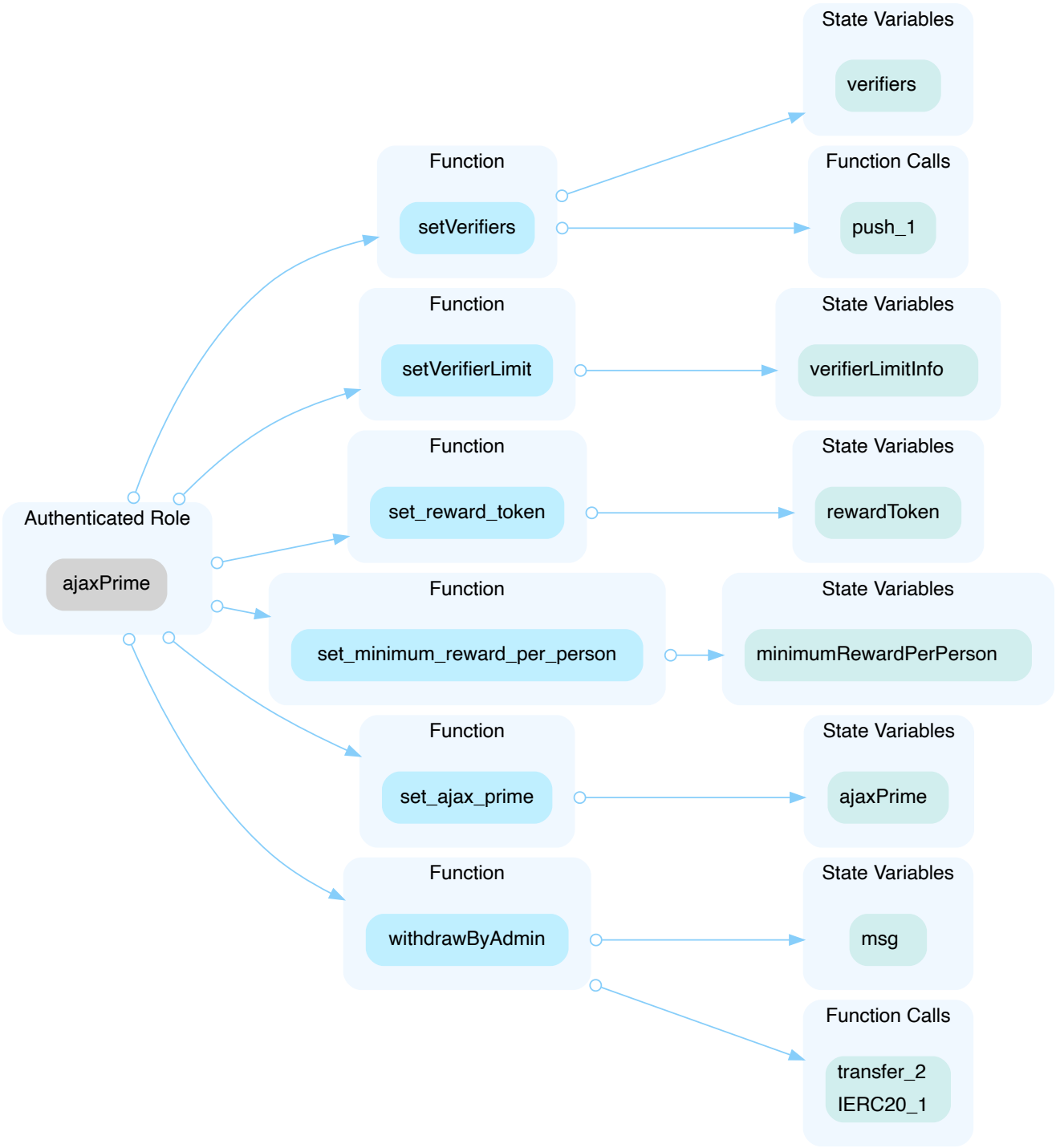
In the contract `LpYield` the role `JaxAdmin.governor` has authority over the functions listed below.

- function `set_yield_info()`
- function `set_yield_tokens()`
- function `set_reward_token()`
- function `pay_yield()`

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

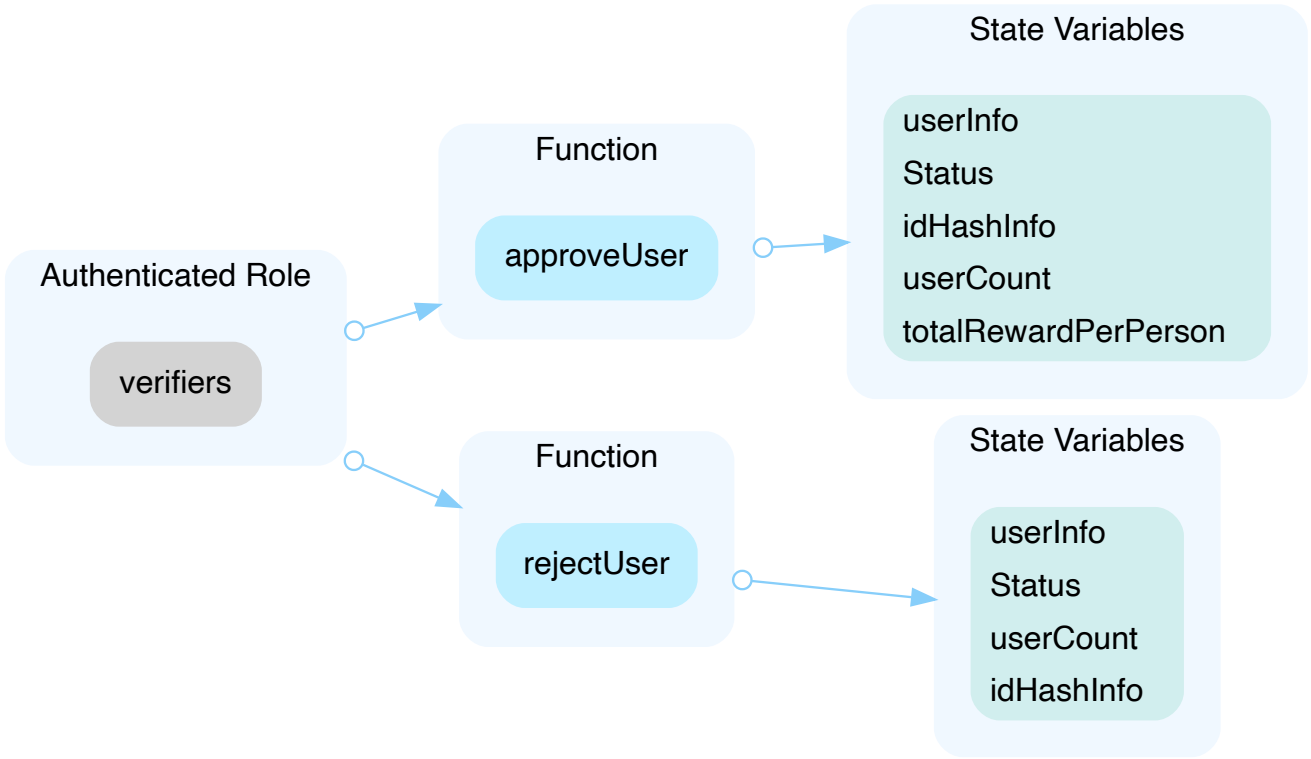
In the contract `Ubi` the role `ajaxPrime` has authority over the functions shown in the diagram below.

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



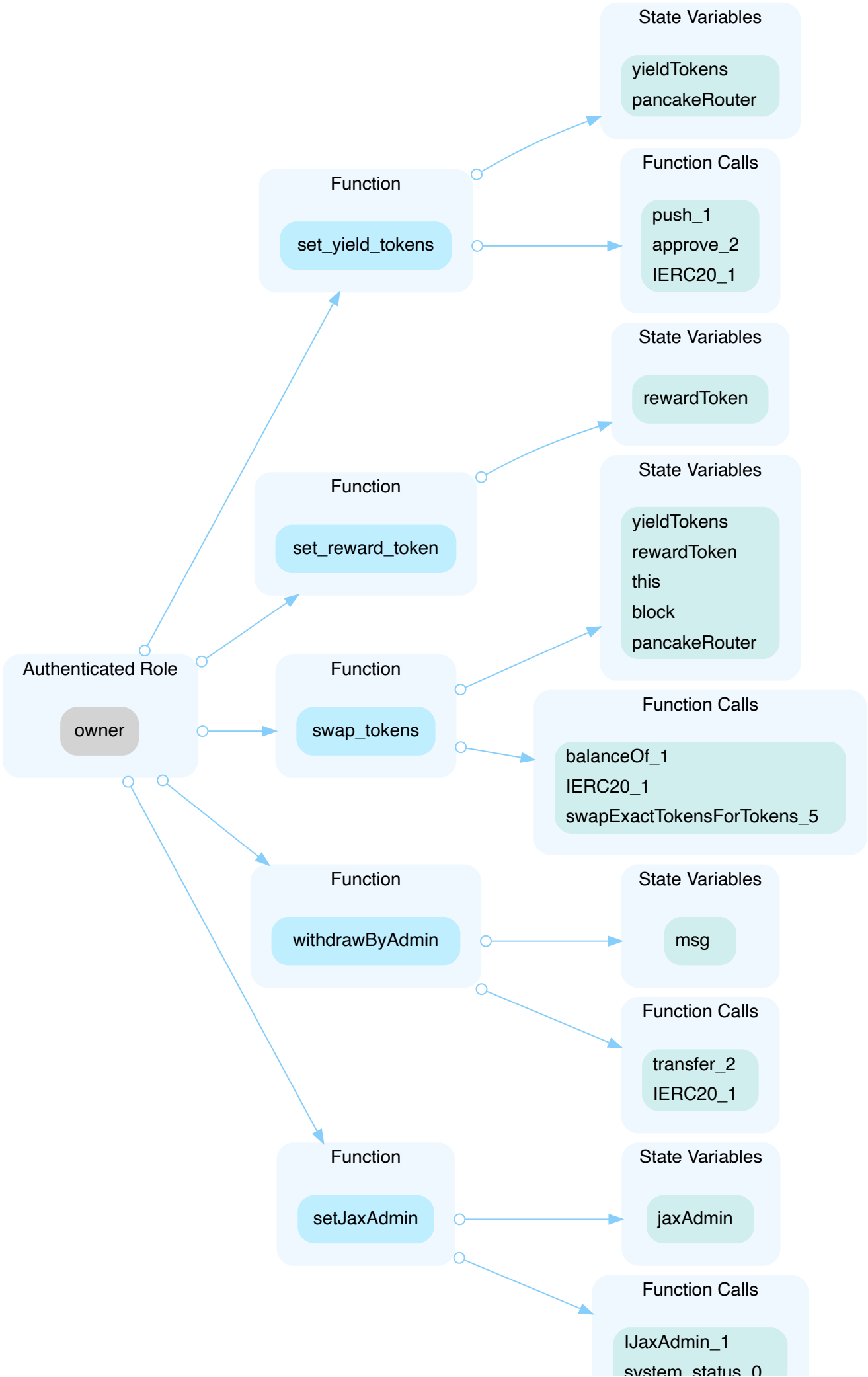
In the contract `Ubi` the role `verifiers` has authority over the functions shown in the diagram below.

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



In the contract `UbiTaxWallet` 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
yieldTokens
pancakeRouter
Function
set_yield_tokens
Function Calls
push_1
approve_2
IERC20_1
State Variables
rewardToken
Function
set_reward_token
State Variables
yieldTokens
rewardToken
this
block
pancakeRouter
Authenticated Role
owner
Function
swap_tokens
Function Calls
balanceOf_1
IERC20_1
swapExactTokensForTokens_5
Function
withdrawByAdmin
State Variables
msg
Function Calls
transfer_2
IERC20_1
Function
setJaxAdmin
State Variables
jaxAdmin
Function Calls
IJaxAdmin_1

In the contract `UbiTaxWallet` the role `JaxAdmin.ajaxPrime` has authority over the functions shown in the diagram below.

- Function `set_yield_tokens()`
- Function `set_reward_token()`
- Function `swap_tokens()`
- Function `withdrawByAdmin()`
- Function `setJaxAdmin()`

Any compromise to the `JaxAdmin.ajaxPrime` 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.

JAX-02 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Minor	Global	Pending

Description

The contract is serving as the underlying entity to interact with third party `PancakeSwap` protocols. The scope of the audit treats 3rd party entities as black boxes and assume these 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 `JaxNetwork` requires interaction with `PancakeSwap`, etc. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

JAX-03 | External Dependency

Category	Severity	Location	Status
Logical Issue	Minor	Global	Pending

Description

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

- `BUSD`
- `rewardToken`
- `WJNX`
- `JUSD`

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

Recommendation

We recommend ensuring the external dependency contracts are correct.

Alleviation

Collect Coin team acknowledged this finding.

JAJ-01 | Pull-Over-Push Pattern In `JaxAdmin.updateGovernor()`

Category	Severity	Location	Status
Logical Issue	Minor	contracts/JaxAdmin.sol: 228~234	Pending

Description

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

Recommendation

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

The following code snippet can be taken as a reference:

```
228    function updateGovernor () external {
229      require(newGovernor != governor && newGovernor != address(0x0), "New governor
hasn't been elected");
230      require(governorStartDate >= block.timestamp, "New governor is not ready");
231      require(msg.sender == newGovernor, "You are not nominated as potential
governor");
232      address old_governor = governor;
233      governor = newGovernor;
234      newGovernor = address(0x0);
235      emit Update_Governor(old_governor, newGovernor);
236    }
```

JAJ-02 | Privileged Ownership In `JaxAdmin.sol`

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/JaxAdmin.sol: 189~192	Pending

Description

In contract `VRP.sol`, users are able to vote a candidate address to be new `governor` for contract `JaxAdmin` by calling function `VRP.vote_governor()`. After 1 week, the candidate address can be set as new `governor`.

- VPR.sol

```
226 function vote_governor(address candidate) public {
227   require(balanceOf(msg.sender) > 0, "Only VRP holders can participate voting.");
228   require(vote[msg.sender] != candidate, "Already Voted");
229   if(vote[msg.sender] != address(0x0)) {
230     voteShare[vote[msg.sender]] -= balanceOf(msg.sender);
231   }
232   vote[msg.sender] = candidate;
233   voteShare[candidate] += balanceOf(msg.sender);
234   emit Vote_Governor(msg.sender, candidate);
235   check_candidate(candidate);
236 }
237
238 function check_candidate(address candidate) internal {
239   if(candidate == address(0x0)) return;
240   if(voteShare[candidate] >= totalSupply() * 51 / 100) {
241     jaxAdmin.electGovernor(candidate);
242   }
243 }
```

- JaxAdmin.sol

```
216 function electGovernor (address _governor) external {
217   require(msg.sender == address(vrp), "Only VRP contract can perform this
operation.");
218   newGovernor = _governor;
219   governorStartDate = block.timestamp + 7 * 24 * 3600;
220   emit Elect_Governor(_governor);
221 }
```

```
228 function updateGovernor () external onlyAjaxPrime {
229   require(newGovernor != governor && newGovernor != address(0x0), "New governor
hasn't been elected");
230   require(governorStartDate >= block.timestamp, "New governor is not ready");
231   address old_governor = governor;
232   governor = newGovernor;
233   emit Update_Governor(old_governor, newGovernor);
234 }
```

But the role `ajaxPrime` in contract `JaxAdmin` has the super privilege to set `governor` directly by executing function `setGovernor()`.

```
189 function setGovernor (address _governor) external onlyAjaxPrime {
190   governor = _governor;
191   emit Set_Governor(_governor);
192 }
```

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.

JAJ-03 | Naming Optimization

Category	Severity	Location	Status
Coding Style	Informational	contracts/JaxAdmin.sol: 50	Pending

Description

According to the use of mapping `fee_blacklist` in functions `JaxToken.transfer()` and `WJAX.transfer()`, we recommend renaming this mapping to `fee_FreeList`.

"fee_blacklist" sounds like "The users who can not cat reward from transaction fee."

Recommendation

We recommend renaming the `mapping` to **"fee_FreeList"** for improving the coding readability.

JAJ-04 | Inconsistent Function Naming

Category	Severity	Location	Status
Coding Style	Informational	contracts/JaxAdmin.sol: 421, 415, 389, 382	Pending

Description

The functions `get_wjxn_jusd_ratio()` and `get_wjax_jusd_ratio()` within the contract `JaxAdmin` get `ratio` in **Exchange of WJXN and BUSD** and **Exchange of WJAX and BUSD**.

Therefore it should be better to change "jusd" to "busd" or "usd" in functions naming.

Recommendation

We recommend modifying the word "jusd" in functions name to "busd".

JAJ-05 | Lack Of Sanity Check For Parameter `ratio`

Category	Severity	Location	Status
Volatile Code	Minor	contracts/JaxAdmin.sol: 439, 434	Pending

Description

In functions `set_wjxn_usd_ratio()` and `set_wjax_usd_ratio()`, the result of function `check_price_bound()` is used to make sure the new `ratio` is within a reasonable range of variation.

```
361   function set_wjxn_usd_ratio(uint ratio) external onlyOperator callLimit(0x1, 180){
362     require(wjxn_usd_ratio == 0 || check_price_bound(wjxn_usd_ratio, ratio, 10),
363         "Out of 10% ratio change");
364     wjxn_usd_ratio = ratio;
365     emit Set_Wjxn_Usd_Ratio(ratio);
366   }
367
368   function set_wjax_usd_ratio(uint ratio) external onlyOperator callLimit(0x2, 180)
{
369     require(wjax_usd_ratio == 0 || check_price_bound(wjax_usd_ratio, ratio, 5),
370       "Out of 5% ratio change");
371     wjax_usd_ratio = ratio;
372     emit Set_Wjax_Usd_Ratio(ratio);
373   }
```

It should be necessary to call function `check_price_bound()` in functions `set_wjxn_wjax_collateralization_ratio()` and `set_wjax_collateralization_ratio` to check the passed parameter `ratio` like functions above.

Recommendation

We recommend adding a `require` statement to call function `check_price_bound()` for checking the value of parameter `ratio`.

JNC-01 | Divide Before Multiply

Category	Severity	Location	Status
Mathematical Operations	Informational	contracts/JaxAdmin.sol: 460, 470 contracts/JaxLibrary.sol: 24 contracts/VRP.sol: 161, 168, 213, 216 contracts/yield/LpYield.sol: 306, 314, 363, 365	Pending

Description

Performing integer division before multiplication truncates the low bits, losing the precision of calculation.

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 460, Function `JaxAdmin.show_reserves`)

```
    uint wjax_usd_value = wjax_reserves * get_wjax_jusd_ratio() * (10 ** jusd.decimals())
/ 1e8 / (10 ** wjax.decimals());
```

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 470, Function `JaxAdmin.show_reserves`)

```
        wjax_lsc_ratio = wjax_usd_value * 1e8 / lsc_usd_value;
```

File: projects/JaxNetwork/contracts/JaxLibrary.sol (Line 24, Function `JaxLibrary.swapWithPriceImpactLimit`)

```
    require((reserveOut * 1e18 / reserveIn) * (1e8 - limit) / 1e8 <= amountOut * 1e18 /
amountIn, "Price Impact too high");
```

File: projects/JaxNetwork/contracts/VRP.sol (Line 161, Function `VRP.deposit_reward`)

```
        uint rewardPerShare = amount * 1e36 / epochShare; // multiplied by 1e36
```

File: projects/JaxNetwork/contracts/VRP.sol (Line 168, Function `VRP.deposit_reward`)

```
        newEpoch.totalRewardPerBalance = epochInfo[currentEpoch-1].totalRewardPerBalance
+ rewardPerShare * (block.number - lastEpochBlock);
```

File: projects/JaxNetwork/contracts/VRP.sol (Line 213, Function `VRP.harvest`)

```
        uint reward = (user.totalReward - user.rewardPaid)/1e36;
```

File: projects/JaxNetwork/contracts/VRP.sol (Line 216, Function `VRP.harvest`)

```
        user.rewardPaid += reward * 1e36;
```

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 306, Function `LpYield.deposit_reward`)

```
        uint rewardPerShare = amount * 1e36 / epochShare; // multiplied by 1e36
```

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 314, Function `LpYield.deposit_reward`)

```
        newEpoch.totalRewardPerBalance = epochInfo[currentEpoch-1].totalRewardPerBalance
+ rewardPerShare * (block.number - lastEpochBlock);
```

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 363, Function `LpYield._harvest`)

```
        reward = (user.totalReward - user.rewardPaid)/1e36;
```

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 365, Function `LpYield._harvest`)

```
        user.rewardPaid += reward * 1e36;
```

Recommendation

We recommend applying multiplication before division to avoid loss of precision.

JNC-02 | Uninitialized Local Variable

Category	Severity	Location	Status
Coding Style	Informational	contracts/JaxAdmin.sol: 137, 143, 197, 206, 323 contracts/WJAX.sol: 142 contracts/yield/Ubi.sol: 65	Pending

Description

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 137, Function `JaxAdmin.userIsOperator`)

```
    uint index;
```

- Local variable `index` is never initialized.

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 143, Function `JaxAdmin.userIsOperator`)

```
        for(uint j; j < functions_whitelisted.length; j+=1) {
```

- Local variable `j` is never initialized.

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 197, Function `JaxAdmin.setOperators`)

```
    for(uint index; index < operatorsCnt; index += 1 ) {
```

- Local variable `index` is never initialized.

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 206, Function `JaxAdmin.setWhitelistForOperator`)

```
    uint i;
```

- Local variable `i` is never initialized.

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 323, Function `JaxAdmin.delete_jtoken`)

```
uint jtoken_index;
```

- Local variable `jtoken_index` is never initialized.

File: projects/JaxNetwork/contracts/WJAX.sol (Line 142, Function `WJAX.setGateKeepers`)

```
for(uint index; index < cnt; index += 1) {
```

- Local variable `index` is never initialized.

File: projects/JaxNetwork/contracts/yield/Ubi.sol (Line 65, Function `Ubi.setVerifiers`)

```
        for(uint index; index < verifiersCnt; index += 1 ) {
```

- Local variable `index` is never initialized.

Recommendation

We recommend initializing all local variables. If a variable is meant to be initialized to zero, explicitly set it to zero to improve code readability.

JNC-03 | Unused Return Value

Category	Severity	Location	Status
Volatile Code	Informational	contracts/yield/LpYield.sol: 102, 137, 145, 98, 99 contracts/yield/TxFeeWallet.sol: 123~129, 72, 84 contracts/yield/UbiTaxWallet.sol: 45, 68~74	Pending

Description

The return value of an external call is not stored in a local or state variable.

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 102, Function `LpYield.initialize`)

```
        IERC20(lpToken).approve(address(router), type(uint256).max);
```

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 137, Function `LpYield.setJaxAdmin`)

```
        jaxAdmin.system_status();
```

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 145, Function `LpYield.set_token_addresses`)

```
        IERC20(lpToken).approve(address(router), type(uint256).max);
```

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 98, Function `LpYield.initialize`)

```
        IERC20(BUSD).approve(address(router), type(uint256).max);
```

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 99, Function `LpYield.initialize`)

```
        IERC20(WJAX).approve(address(router), type(uint256).max);
```

File: projects/JaxNetwork/contracts/yield/TxFeeWallet.sol (Line 123-129, Function `TxFeeWallet.swap_tokens`)

```
            pancakeRouter.swapExactTokensForTokens(
                amountIn,
                0,
                path,
                address(this),
                block.timestamp
            );
```

File: projects/JaxNetwork/contracts/yield/TxFeeWallet.sol (Line 72, Function `TxFeeWallet.set_yield_info`)

```
                IERC20(rewardToken).approve(yield.yield_address, type(uint).max);
```

File: projects/JaxNetwork/contracts/yield/TxFeeWallet.sol (Line 84, Function `TxFeeWallet.set_yield_tokens`)

```
            IERC20(newYieldTokens[i]).approve(address(pancakeRouter), type(uint256).max);
```

File: projects/JaxNetwork/contracts/yield/UbiTaxWallet.sol (Line 45, Function `UbiTaxWallet.set_yield_tokens`)

```
            IERC20(newYieldTokens[i]).approve(address(pancakeRouter), type(uint256).max);
```

File: projects/JaxNetwork/contracts/yield/UbiTaxWallet.sol (Line 68-74, Function `UbiTaxWallet.swap_tokens`)

```
            pancakeRouter.swapExactTokensForTokens(
                amountIn,
                0,
                path,
                address(this),
                block.timestamp
            );
```

Recommendation

We recommend checking or using the return values of all external function calls.

JNC-04 | Locked Ether

Category	Severity	Location	Status
Language Specific	Medium	contracts/WJAX.sol: 90~96 contracts/JaxToken.sol: 80~86	Pending

Description

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

File: projects/JaxNetwork/contracts/JaxToken.sol (Line 80, Contract `JaxToken`)

```
80 constructor (
81   string memory name,
82   string memory symbol,
83   uint8 decimals
84 )
85   BEP20(name, symbol)
86   payable
87 {
88   _setupDecimals(decimals);
89   tx_fee_wallet = msg.sender;
90 }
91
```

File: projects/JaxNetwork/contracts/WJAX.sol (Line 90, Contract `WJAX`)

```
constructor (
```

Recommendation

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

JNC-05 | Missing Zero Address Validation

Category	Severity	Location	Status
Volatile Code	Minor	contracts/JaxAdmin.sol: 185, 190, 218, 224, 285, 291, 446, 494, 495, 496, 499 contracts/JaxPlanet.sol: 133, 82 contracts/JaxToken.sol: 122 contracts/VRP.sol: 221 contracts/WJAX.sol: 159 contracts/yield/LpYield.sol: 142, 143, 150, 96, 97 contracts/yield/TxFeeWallet.sol: 59, 90 contracts/yield/Ubi.sol: 144, 145, 150, 77 contracts/yield/UbiTaxWallet.sol: 36, 51	Pending

Description

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

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 185, Function `JaxAdmin.setAdmin`)

```
    admin = _admin;
```

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

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 190, Function `JaxAdmin.setGovernor`)

```
    governor = _governor;
```

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

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 218, Function `JaxAdmin.electGovernor`)

```
    newGovernor = _governor;
```

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

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 224, Function `JaxAdmin.setAjaxPrime`)

```
    ajaxPrime = _ajaxPrime;
```

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

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 285, Function `JaxAdmin.setJaxSwap`)

```
    jaxSwap = _jaxSwap;
```

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

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 291, Function `JaxAdmin.setJaxPlanet`)

```
    jaxPlanet = _jaxPlanet;
```

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

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 446, Function `JaxAdmin.set_wjax_jusd_markup_fee`)

```
    wjax_jusd_markup_fee_wallet = _wallet;
```

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

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 494, Function `JaxAdmin.initialize`)

```
    admin = sender;
```

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 495, Function `JaxAdmin.initialize`)

```
    governor = sender;
```

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 496, Function `JaxAdmin.initialize`)

```
    ajaxPrime = sender;
```

File: projects/JaxNetwork/contracts/JaxAdmin.sol (Line 499, Function `JaxAdmin.initialize`)

```
    owner = sender;
```

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

File: projects/JaxNetwork/contracts/JaxPlanet.sol (Line 133, Function `JaxPlanet.setJaxCorpDAO`)

```
      jaxcorp_dao_wallet = jaxCorpDao_wallet;
```

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

File: projects/JaxNetwork/contracts/JaxPlanet.sol (Line 82, Function `JaxPlanet.setUbiTax`)

```
      ubi_tax_wallet = wallet;
```

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

File: projects/JaxNetwork/contracts/JaxToken.sol (Line 122, Function `JaxToken.setTransactionFee`)

```
      tx_fee_wallet = wallet;
```

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

File: projects/JaxNetwork/contracts/VRP.sol (Line 221, Function `VRP.set_reward_token`)

```
        rewardToken = _rewardToken;
```

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

File: projects/JaxNetwork/contracts/WJAX.sol (Line 159, Function `WJAX.setTransactionFee`)

```
      tx_fee_wallet = wallet;
```

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

File: projects/JaxNetwork/contracts/pancake-for-test/PancakeFactory.sol (Line 274, Function `PancakePair.initialize`)

```
        token0 = _token0;
```

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

File: projects/JaxNetwork/contracts/pancake-for-test/PancakeFactory.sol (Line 275, Function `PancakePair.initialize`)

```
        token1 = _token1;
```

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

File: projects/JaxNetwork/contracts/pancake-for-test/PancakeFactory.sol (Line 421, Function `PancakeFactory.constructor`)

```
        feeToSetter = _feeToSetter;
```

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

File: projects/JaxNetwork/contracts/pancake-for-test/PancakeFactory.sol (Line 447, Function `PancakeFactory.setFeeTo`)

```
        feeTo = _feeTo;
```

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

File: projects/JaxNetwork/contracts/pancake-for-test/PancakeFactory.sol (Line 452, Function `PancakeFactory.setFeeToSetter`)

```
        feeToSetter = _feeToSetter;
```

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

File: projects/JaxNetwork/contracts/pancake-for-test/PancakeRouter.sol (Line 402, Function `PancakeRouter.constructor`)

```
        factory = _factory;
```

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

File: projects/JaxNetwork/contracts/pancake-for-test/PancakeRouter.sol (Line 403, Function `PancakeRouter.constructor`)

```
        WETH = _WETH;
```

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

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 142, Function `LpYield.set_token_addresses`)

```
        WJAX = _WJAX;
```

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

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 143, Function `LpYield.set_token_addresses`)

```
        BUSD = _BUSD;
```

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

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 150, Function `LpYield.set_reward_token`)

```
        rewardToken = _rewardToken;
```

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

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 96, Function `LpYield.initialize`)

```
        BUSD = _BUSD;
```

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

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 97, Function `LpYield.initialize`)

```
        WJAX = _WJAX;
```

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

File: projects/JaxNetwork/contracts/yield/TxFeeWallet.sol (Line 59, Function `TxFeeWallet.initialize`)

```
        rewardToken = _rewardToken;
```

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

File: projects/JaxNetwork/contracts/yield/TxFeeWallet.sol (Line 90, Function `TxFeeWallet.set_reward_token`)

```
        rewardToken = _rewardToken;
```

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

File: projects/JaxNetwork/contracts/yield/Ubi.sol (Line 144, Function `Ubi.initialize`)

```
        ajaxPrime = _ajaxPrime;
```

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

File: projects/JaxNetwork/contracts/yield/Ubi.sol (Line 145, Function `Ubi.initialize`)

```
        rewardToken = _rewardToken;
```

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

File: projects/JaxNetwork/contracts/yield/Ubi.sol (Line 150, Function `Ubi.set_ajax_prime`)

```
        ajaxPrime = newAjaxPrime;
```

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

File: projects/JaxNetwork/contracts/yield/Ubi.sol (Line 77, Function `Ubi.set_reward_token`)

```
        rewardToken = _rewardToken;
```

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

File: projects/JaxNetwork/contracts/yield/UbiTaxWallet.sol (Line 36, Function `UbiTaxWallet.initialize`)

```
        rewardToken = _rewardToken;
```

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

File: projects/JaxNetwork/contracts/yield/UbiTaxWallet.sol (Line 51, Function `UbiTaxWallet.set_reward_token`)

```
        rewardToken = _rewardToken;
```

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

Recommendation

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

JNC-06 | Redundant Code Components

Category	Severity	Location	Status
Volatile Code	Informational	contracts/WJAX.sol: 107~110 contracts/VRP.sol: 152~155, 147~150 contracts/JaxToken.sol: 97~100 contracts/HaberStornetta.sol: 5 contracts/JaxAdmin.sol: 119~122 contracts/JaxPlanet.sol: 57~60, 67~70	Pending

Description

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

Recommendation

We advise to remove the redundant statements for production environments.

JNC-07 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	contracts/yield/LpYield.sol: 369~371 contracts/yield/UbiTaxWallet.sol: 79~81 contracts/yield/TxFeeWallet.sol: 134~136 contracts/yield/Ubi.sol: 154~156 contracts/VRP.sol: 328~330 contracts/WJAX.sol: 300~306, 308~314	Pending

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.

JNC-08 | Pull-Over-Push Pattern

Category	Severity	Location	Status
Logical Issue	Minor	contracts/yield/Ubi.sol: 148~152 contracts/JaxAdmin.sol: 189~192, 223~226, 184~187 contracts/JaxOwnable.sol: 16~34	Pending

Description

The change of following privileged roles by functions overrides the previously set address with the new one without guaranteeing the new address is able to actuate transactions on-chain.

- `JaxOwnable.sol` :
  - Modifying `JaxOwnable.owner` by function `_transferOwnership()`
- `JaxAdmin.sol`
  - Modifying `JaxAdmin.admin` by function `setAdmin()`
  - Modifying `JaxAdmin.governor` by function `setGovernor()`
  - Modifying `JaxAdmin.ajaxPrime` by function `setAjaxPrime()`
- `yield/Ubi.sol`
  - Modifying `ajaxPrime` by function `set_ajax_prime()`

Recommendation

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

The following code snippet for changing `JaxOwnable.owner` can be taken as a reference:

```
address public potentialOwner;

event OwnerNominated(address indexed newOwner);

function transferOwnership(address newOwner) external onlyOwner {
  require(newOwner != address(0), "new owner is the zero address");
  potentialOwner = newOwner;
  emit OwnerNominated(potentialOwner);
}

function acceptOwnership() external {
```

```
  require(msg.sender == potentialOwner, 'You must be nominated as potential owner before
you can accept owner role');
  address oldOwner = owner;
  owner = potentialOwner;
  potentialOwner = address(0);
  emit OwnershipTransferred(oldOwner, owner);
}

function renounceOwnership() external onlyOwner {
  address oldOwner = owner;
  owner = address(0);
  potentialOwner = address(0);
  emit OwnershipTransferred(oldOwner, owner);

}
```

JNC-09 | Inconsistent Comments And Code

Category	Severity	Location	Status
Coding Style	Informational	contracts/yield/LpYield.sol: 71, 280, 25 contracts/WJAX.sol: 40~44, 178, 56 contracts/VRP.sol: 35~40 contracts/JaxToken.sol: 141, 55	Pending

Description

In `JaxToken.sol` and `WJAX.sol` every assignment statement for state variable `cashback`, assigned value is clearly not `1e8`.

```
55    uint public cashback = 0; //1e8
```

```
139    function setCashback(uint cashback_percent) external onlyJaxAdmin {
140        require(cashback_percent <= 1e8 * 30 / 100 , "Cashback percent can't be more
than 30.");
141        cashback = cashback_percent; //1e8
142        emit Set_Cashback(cashback_percent);
143    }
```

In contract `VRP`, the title in comment is recorded as "WJAX".

```
35 /**
36  * @title WJAX
37  * @dev Implementation of the WJAX
38  */
39 //, Initializable
40 contract VRP is IVRP, Initializable, JaxOwnable {
```

In contract `WJAX`, the title in comment is recorded as "JaxToken".

```
35 /**
36 * @title JaxToken
37 * @dev Implementation of the JaxToken. Extension of {BEP20} that adds a fee
transaction behaviour.
38 */
39 contract WJAX is BEP20 {
```

In contract `yield/LpYield.sol`, the comment of variable `rewardPerShare` is documented as "1e36". For a variable of type `uint`, this comment can be interpreted as “The variable `rewardPerShare` has a fixed value of 1e36.”

```
25          uint rewardPerShare; // 1e36
```

But this variable doesn't have a fixed value according to the usage of this variable in the function `depodit_reward()`.

```
306         uint rewardPerShare = amount * 1e36 / epochShare; // multiplied by 1e36
307         ... ...
308         newEpoch.rewardPerShare = rewardPerShare;
```

The same issue is happened in following position :

- Comment of state variable `liquidity_ratio_limit` (line #71)
- Comment of function `get_liquidity_ratio()` (line #280)

Recommendation

We recommend modifying the comments to improve the code readability.

JNC-10 | Unused Return Value `jaxAdmin.system_status()`

Category	Severity	Location	Status
Volatile Code	Informational	contracts/JaxPlanet.sol: 75 contracts/JaxSwap.sol: 100 contracts/JaxToken.sol: 114 contracts/VRP.sol: 120 contracts/yield/LpYield.sol: 137 contracts/yield/TxFeeWallet.sol: 141 contracts/yield/UbiTaxWallet.sol: 86	Pending

Description

The return value of `jaxAdmin.system_status()` is not used in function.

File: projects/JaxNetwork/contracts/JaxPlanet.sol (Line 75, Function `JaxPlanet.setJaxAdmin`)

```
    jaxAdmin.system_status();
```

File: projects/JaxNetwork/contracts/JaxSwap.sol (Line 100, Function `JaxSwap.setJaxAdmin`)

```
    jaxAdmin.system_status(); // check if jaxAdmin is correct contract.
```

File: projects/JaxNetwork/contracts/JaxToken.sol (Line 114, Function `JaxToken.setJaxAdmin`)

```
    jaxAdmin.system_status();
```

File: projects/JaxNetwork/contracts/VRP.sol (Line 120, Function `VRP.setJaxAdmin`)

```
        jaxAdmin.system_status();
```

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 137, Function `LpYield.setJaxAdmin`)

```
        jaxAdmin.system_status();
```

File: projects/JaxNetwork/contracts/yield/TxFeeWallet.sol (Line 141, Function `TxFeeWallet.setJaxAdmin`)

```
        jaxAdmin.system_status();
```

File: projects/JaxNetwork/contracts/yield/UbiTaxWallet.sol (Line 86, Function `UbiTaxWallet.setJaxAdmin`)

```
        jaxAdmin.system_status();
```

Recommendation

We recommend adding a `require` statement for checking the return value of `jaxAdmin.system_status()` to make sure the contract deployed on new address `jaxAdmin` is correct.

JNC-11 | Potential Sandwich Attacks

Category	Severity	Location	Status
Logical Issue	Minor	contracts/yield/UbiTaxWallet.sol: 68~75 contracts/yield/TxFeeWallet.sol: 123~129	Pending

Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by front-running (before the transaction being attacked) a transaction to purchase one of the assets and make profits by back-running (after the transaction being attacked) a transaction to sell the asset.

The functions listed blew call `pancakeRouter.swapExactTokensForTokens()` without setting restrictions on slippage or minimum output amount, so transaction triggering in function is vulnerable to sandwich attacks, especially when the input amount is large.

- function `swap_tokens()` (yield/TxFeeWallet.sol)
- function `swap_tokens()` (yield/UbiTaxWallet.sol)

Recommendation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

JNC-12 | The Purpose Of Function `withdrawByAdmin()`

Category	Severity	Location	Status
Volatile Code	Discussion	contracts/yield/Ubi.sol: 154~156 contracts/yield/LpYield.sol: 369~371	Pending

Description

In the contract `LPYield`, the roles `JaxAdmin.admin` and `owner` have authority over the function `withdrawByAdmin()`, which will allow the roles to withdraw all of the funds from this contract.

In our understanding of this contract, users can deposit `BUSD` to add liquidity of `pair(BUSD, WJAX)` for earning `rewardToken`, and the role `JaxAdmin.governor` will deposit `rewardToken` to this contract for increasing the revenue of users. Therefore all of tokens deposited in this contract belong to users.

Function `withdrawByAdmin()` in contract `Ubi.sol` is similar to issue.

Recommendation

We hope that client would teach us which token will withdraw to `JaxAdmin.admin` and tell us the why they want to withdraw it. Or provide us the white paper about logic here.

JNC-13 | Incorrect Error Message

Category	Severity	Location	Status
Coding Style	Informational	contracts/yield/UbiTaxWallet.sol: 26 contracts/yield/Ubi.sol: 44	Pending

Description

The error message in `require(msg.sender == ajaxPrime, "Only Admin");` is not described correctly.

The error message in `require(jaxAdmin.userIsAjaxPrime(msg.sender) || msg.sender == owner, "Only Admin can perform this operation.");` is not described correctly.

Recommendation

The word `Admin` in message can be changed to `AjaxPrime`.

JPJ-01 | No Upper Limit For `min_transaction_tax`

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/JaxPlanet.sol: 144	Pending

Description

State variable `min_transaction_tax` is used to store the lowest value for `Colony.transaction_tax`.

According to the `require` statement in function `registerColony()` and function `setJaxCorpDAO()`, the value for variable `transaction_tax` in struct `Colony` can not be more than 20%, but this upper limit is not checked in function `setMinTransactionTax()`.

Without this upper limit check, The value of `transaction_tax` obtained by function `getColony()` may exceed 20%.

```
108 function getColony(address addr) external view returns(Colony memory) {
109   Colony memory colony = colonies[addr];
110   if(colony.transaction_tax < min_transaction_tax)
111     colony.transaction_tax = min_transaction_tax;
112   return colony;
113 }
```

Recommendation

We recommend adding a reasonable upper limit check for parameter `min_tx_tax`.

JSJ-01 | Unused Local Variable In Modifirer `notContract`

Category	Severity	Location	Status
Coding Style	Minor	contracts/JaxSwap.sol: 87~96	Pending

Description

In modifier `notContract()`, local variable `size` is declared and assigned a value, but the obtained value is never used in `require` statement.

Recommendation

We recommend adding a check of `size` in `require` statement as following :

```
93      require((msg.sender == tx.origin) && (size == 0),
94           "Contract_Call_Not_Allowed"); //Only non-contract/eoa can perform this
operation
```

JTJ-01 | Unused `internal` Function

Category	Severity	Location	Status
Volatile Code	Minor	contracts/JaxToken.sol: 102~105, 268~271, 263~266	Pending

Description

Functions `_mint()` and `_burn()` are specified as `internal`, these functions can only be accessed from within the current contract or contracts deriving from `JaxToken`.

Due to the modifier `onlyJaxSwap` used for these functions, only contract `JaxSwap` can call these functions.

```
102    modifier onlyJaxSwap() {
103    require(msg.sender == jaxAdmin.jaxSwap(), "Only JaxSwap can perform this
operation.");
104      _;
105    }
```

```
263    function _mint(address account, uint amount) internal override(BEP20) notFrozen
onlyJaxSwap {
264        require(!jaxAdmin.blacklist(account), "account is blacklisted");
265        super._mint(account, amount);
266    }
267
268    function _burn(address account, uint amount) internal override(BEP20) notFrozen
onlyJaxSwap {
269      require(!jaxAdmin.blacklist(account), "account is blacklisted");
270      super._burn(account, amount);
271    }
```

But contract `JaxSwap` is not deriving from `JaxToken`, which means functions `_mint()` and `_burn()` will never be executed.

```
56  contract JaxSwap is IJaxSwap, Initializable, JaxOwnable {
```

Recommendation

We advise the client to review the type of these functions and the justifiability for using modifier `onlyJaxSwap`.

LYJ-01 | Uninitialized State Variable

Category	Severity	Location	Status
Coding Style	Minor	contracts/yield/LpYield.sol: 304, 310, 48, 49	Pending

Description

One or more state variables are used without being initialized in the constructor.

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 48, Contract `LpYield`)

```
    uint public totalLpAmount;
```

- `totalLpAmount` is never initialized, but used in `LpYield.deposit_reward`.

```
304         uint epochShare = (block.number - lastEpochBlock) * totalLpAmount +
epochSharePlus - epochShareMinus;
```

```
310         newEpoch.rewardTokenPrice = getPrice(rewardToken, BUSD) * 1e18 *
totalLpAmount / totalBusdStaked;
```

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 49, Contract `LpYield`)

```
    uint public totalBusdStaked;
```

- `totalBusdStaked` is never initialized, but used in `LpYield.deposit_reward`.

```
310         newEpoch.rewardTokenPrice = getPrice(rewardToken, BUSD) * 1e18 *
totalLpAmount / totalBusdStaked;
```

Recommendation

We recommend initializing the state variables at declaration or in the constructor. If a variable is meant to be initialized to zero, explicitly set it to zero to improve code readability.

LYJ-02 | State Variable Should Be Declared Constant

Category	Severity	Location	Status
Gas Optimization	Informational	contracts/yield/LpYield.sol: 48, 49	Pending

Description

State variables that never change should be declared as `constant` to save gas.

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 48, Contract `LpYield`)

```
    uint public totalLpAmount;
```

- `totalLpAmount` should be declared `constant`.

File: projects/JaxNetwork/contracts/yield/LpYield.sol (Line 49, Contract `LpYield`)

```
    uint public totalBusdStaked;
```

- `totalBusdStaked` should be declared `constant`.

Recommendation

We recommend adding the `constant` attribute to state variables that never change.

LYJ-03 | Potential Risk Of Locked Funds

Category	Severity	Location	Status
Volatile Code	Minor	contracts/yield/LpYield.sol: 251	Pending

Description

Since the `require` statement checks the lower limit of the ratio of WJAX used to provide liquidity to `WJAX.totalSupply()`, it will be possible that the user who finally called `withdraw()` will not be able to withdraw deposited funds.

Recommendation

We recommend client to make sure that users are informed.

LYJ-04 | No Upper Limit For `liquidity_ratio_limit`

Category	Severity	Location	Status
Centralization / Privilege	Major	contracts/yield/LpYield.sol: 298	Pending

Description

State variable `liquidity_ratio_limit` is used to store the lowest ratio of `WJAX` deposited for adding liquidity and `WJAX.totalSupply()`.

If the value of `liquidity_ratio_limit` is `100% * 1e8`, every user can't withdraw funds.

Recommendation

We recommend adding an upper limit check for parameter `liquidity_ratio_limit`.

UJN-01 | Role Has The Same Naming But Different Meaning

Category	Severity	Location	Status
Coding Style	Discussion	contracts/yield/Ubi.sol: 43~46	Pending

Description

Role `ajaxPrime` means `JaxAdmin.ajaxPrime` in contract `JaxAdmin.sol`, `JaxPlanet.sol` and `WJAX.sol` but means `Ubi.ajaxPrime` in `Ubi.sol`.

Recommendation

We recommend client to make sure whether the privileged role `ajaxPrime` used here is `JaxAdmin.ajaxPrime` or not. It not, we recommend renaming this role for improving the coding readability.

UTW-01 | Incorrect `modifier` Naming

Category	Severity	Location	Status
Coding Style	Informational	contracts/yield/UbiTaxWallet.sol: 25~28	Pending

Description

The naming of `modifier` is "onlyAdmin", but the account used to check in `require` statement is `JaxAdmin.ajaxAdmin`.

```
25      modifier onlyAdmin() {
26          require(jaxAdmin.userIsAjaxPrime(msg.sender) || msg.sender == owner, "Only
Admin can perform this operation.");
27          _;
28      }
```

Recommendation

We recommend modifying the naming from "onlyAdmin" to "onlyAjaxPrime".

VRP-01 | Dead Code

Category	Severity	Location	Status
Coding Style	Informational	contracts/VRP.sol: 324~326	Pending

Description

One or more internal functions are not used.

File: projects/JaxNetwork/contracts/VRP.sol (Line 324, Contract `VRP`)

```
    function _setupDecimals(uint8 decimals_) internal {
```

Recommendation

We recommend removing those unused functions.

VRP-03 | Function Call Will Never Succeed

Category	Severity	Location	Status
Logical Issue	Medium	contracts/VRP.sol: 144	Pending

Description

Function `burnFrom()` calls in function `burn()` in line #144, but the function call never succeeds because of the modifier `onlyJaxSwap`, which means function `burn()` can only be called in contract `JaxSwap`.

```
132 function burn(address account, uint256 amount) public onlyJaxSwap {
133   updateReward(account);
134   UserInfo storage info = userInfo[account];
135   info.sharePlus += amount * (block.number - lastEpochBlock);
136   epochSharePlus += amount * (block.number - lastEpochBlock);
137   _burn(account, amount);
138 }
```

```
140 function burnFrom(address account, uint256 amount) public {
141   uint256 currentAllowance = allowance(account, msg.sender);
142   require(currentAllowance >= amount, "BEP20: burn amount exceeds allowance");
143   _approve(account, msg.sender, currentAllowance - amount);
144   burn(account, amount);
145 }
```

Recommendation

We recommend client to unify the permissions for `burn()` and `burnFrom()`, or removing the execution of function `burn()` in `burnFrom()`.

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.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

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.

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.