



Security Assessment

Yearn Together

CertiK Verified on May 7th, 2023





Certik Verified on May 7th, 2023

Yearn Together

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

Executive Summary

TYPES

DeFi

ECOSYSTEM

Binance Smart Chain
(BSC)

METHODS

Manual Review, Static Analysis

LANGUAGE

Solidity

TIMELINE

Delivered on 05/07/2023

KEY COMPONENTS

N/A

CODEBASE

<https://github.com/yearn/yearn-contracts>[...View All](#)

COMMITTS

[d1e20147a4cc5797a51baa0f966353e4bedc201a](#)[f69ba461ac71aec33cc3a7c944fea647358c1d00](#)[3758cfe5a019e3fe63a41b76460f5f09885643ec](#)[...View All](#)

Vulnerability Summary



7

Total Findings

5

Resolved

0

Mitigated

0

Partially Resolved

2

Acknowledged

0

Declined

0

Unresolved

0 Critical

Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.

1 Major

1 Acknowledged



Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.

1 Medium

1 Acknowledged



Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.

2 Minor

2 Resolved



Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.

3 Informational

3 Resolved



Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | YEARN TOGETHER

I **Summary**

[Executive Summary](#)

[Vulnerability Summary](#)

[Codebase](#)

[Audit Scope](#)

[Approach & Methods](#)

I **Third Party Dependencies**

[Description](#)

[Recommendation](#)

I **Findings**

[GLOBAL-01 : Centralization Risks](#)

[D1E-01 : Incompatibility with Deflationary Tokens](#)

[LAB-01 : Missing Zero Address Validation](#)

[LCB-03 : Unsafe Cast](#)

[LCB-02 : Missing Validation](#)

[LCB-04 : Lack Of self-reference Check in the ``setReferrer\(\)``](#)

[LCB-05 : User Can Favor USDT and Depreciate YTG token](#)

I **Optimizations**

[LAB-02 : Redundant Validation](#)

[LSB-01 : State Variable Should Be Declared Constant](#)

I **Appendix**

I **Disclaimer**

CODEBASE | YEARN TOGETHER

Repository

<https://github.com/yearn-together/platform-contracts>

Commit




[d1e20147a4cc5797a51baa0f966353e4bedc201a](#)

[f69ba461ac71aec33cc3a7c944fea647358c1d00](#)

[3758cfe5a019e3fe63a41b76460f5f09885643ec](#)

AUDIT SCOPE | YEARN TOGETHER

3 files audited ● 2 files with Acknowledged findings ● 1 file with Resolved findings

ID	File	SHA256 Checksum
● LAB	 LottoAdmin.sol	f9f81f066a9c6f3b919e9b826874654090fa474 7b9a3f3b403b2903308cfa18b
● LCB	 LottoCore.sol	18bb841c79da500da0a6012923b472f342903 5cf913de880a36e1d393d8aa3af
● LSB	 LottoStorage.sol	622d23fdca03e54f7761d5fb4870054e28bae6 101e5c053d5830f73549bb0c48

APPROACH & METHODS | YEARN TOGETHER

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

The auditing process pays special attention to the following considerations:

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

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

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

THIRD PARTY DEPENDENCIES | YEARN TOGETHER

Description

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

```
39      IERC20 public tokenAddress;           // token address (i.e. USDT)
```

- The contract `LottoBaseStorage` interacts with third party contract with `IERC20` interface via `tokenAddress`.

```
41      IERC20 public ytgTokenAddress;       // YearnTogether token address
```

- The contract `LottoBaseStorage` interacts with third party contract with `IERC20` interface via `ytgTokenAddress`.

```
97      VRFCoordinatorV2Interface COORDINATOR;
```

- The contract `LottoVrfStorage` interacts with third party contract with `VRFCoordinatorV2Interface` interface via `COORDINATOR`.

Recommendation

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

FINDINGS | YEARN TOGETHER



7

Total Findings

0

Critical

1

Major

1

Medium

2

Minor

3

Informational

This report has been prepared to discover issues and vulnerabilities for Yearn Together. Through this audit, we have uncovered 7 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
GLOBAL-01	Centralization Risks	Centralization / Privilege	Major	● Acknowledged
D1E-01	Incompatibility With Deflationary Tokens	Logical Issue	Medium	● Acknowledged
LAB-01	Missing Zero Address Validation	Volatile Code	Minor	● Resolved
LCB-03	Unsafe Cast	Logical Issue	Minor	● Resolved
LCB-02	Missing Validation	Logical Issue	Informational	● Resolved
LCB-04	Lack Of Self-Reference Check In The <code>_setReferrer()</code>	Logical Issue	Informational	● Resolved
LCB-05	User Can Favor USDT And Depreciate YTG Token	Logical Issue	Informational	● Resolved

GLOBAL-01 | CENTRALIZATION RISKS

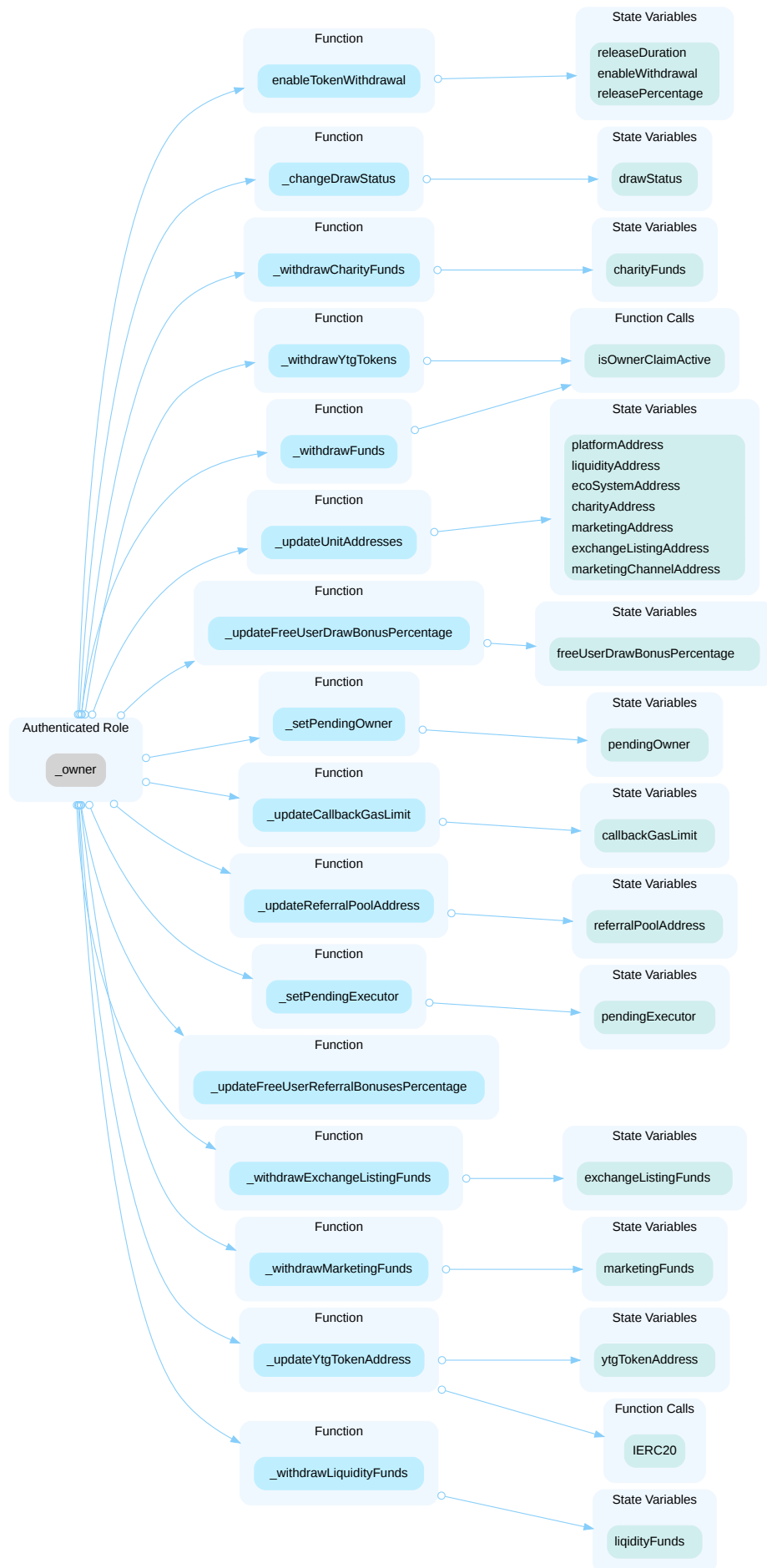
Category	Severity	Location	Status
Centralization / Privilege	● Major		● Acknowledged

I Description

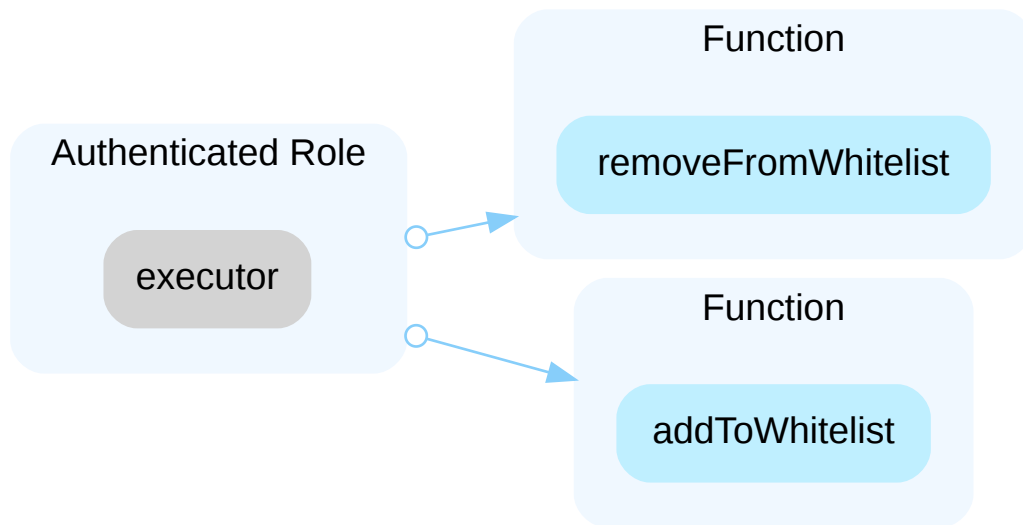
Decentralization Efforts

I Description

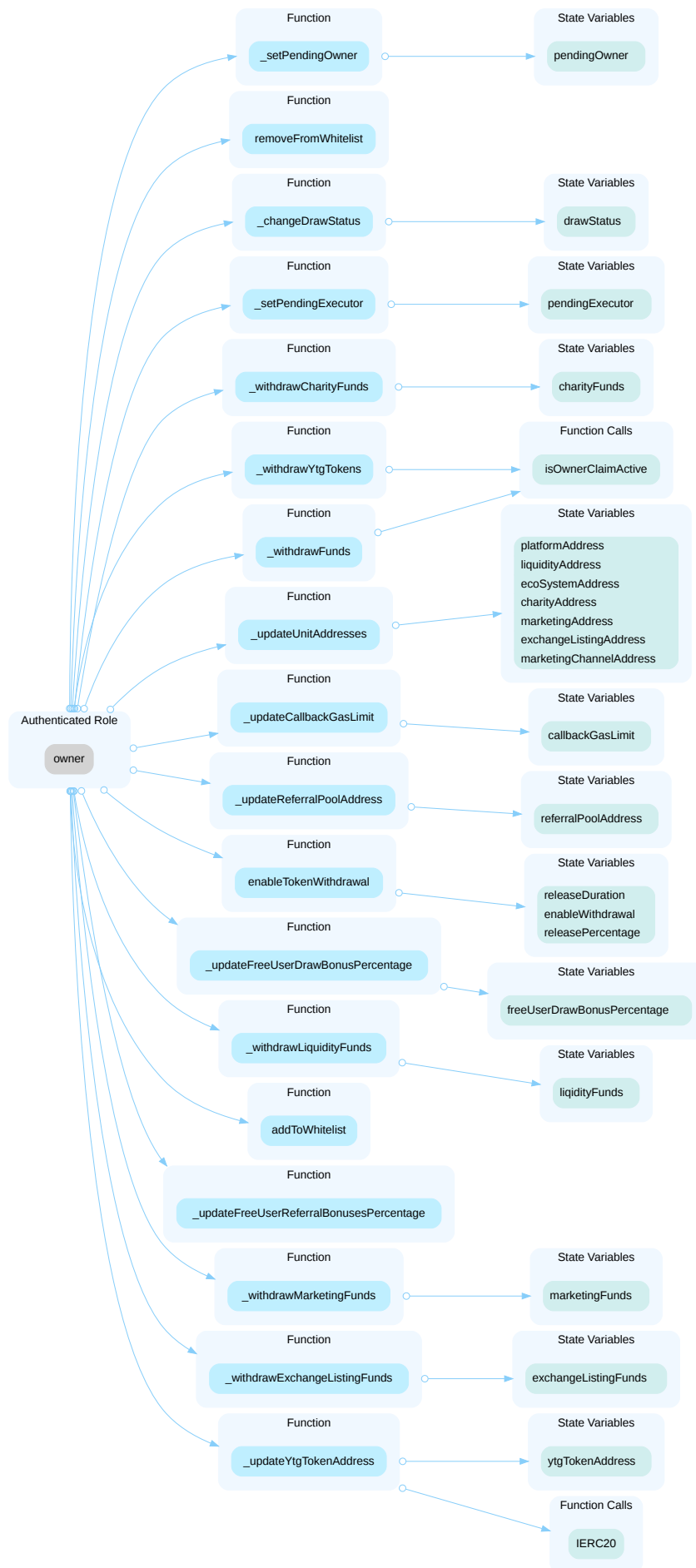
In the contract `LottoAdmin` the role `_owner` has authority over the functions shown in the diagram below.



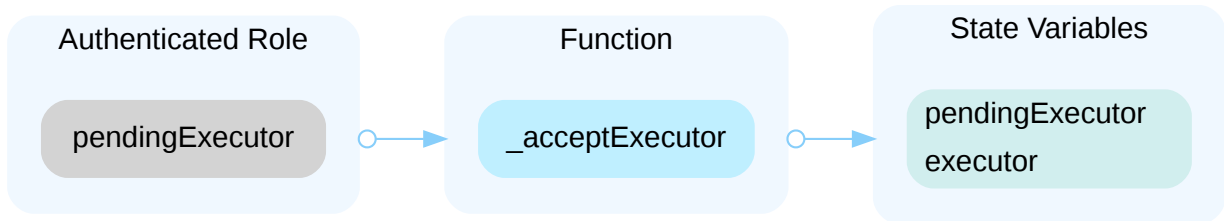
In the contract `LottoAdmin` the role `executor` has authority over the functions shown in the diagram below.



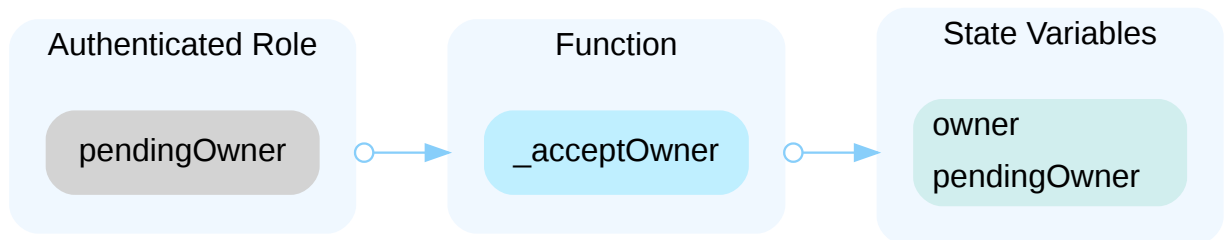
In the contract `LottoAdmin` the role `owner` has authority over the functions shown in the diagram below.



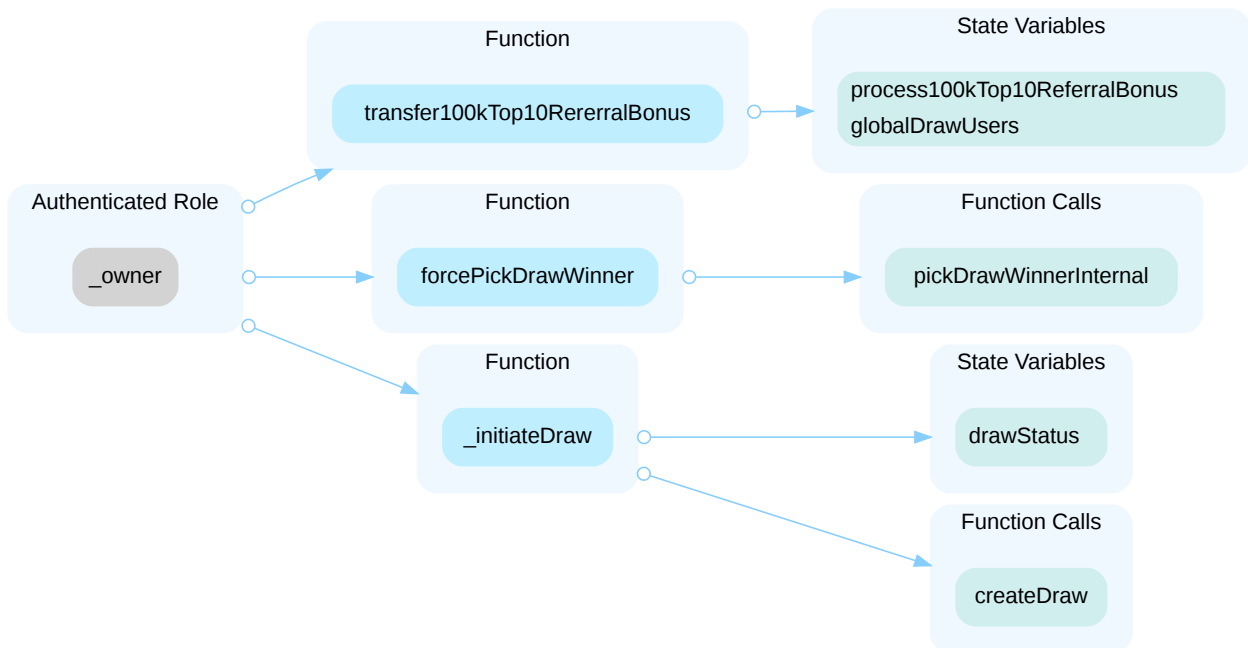
In the contract `LottoAdmin` the role `pendingExecutor` has authority over the functions shown in the diagram below.



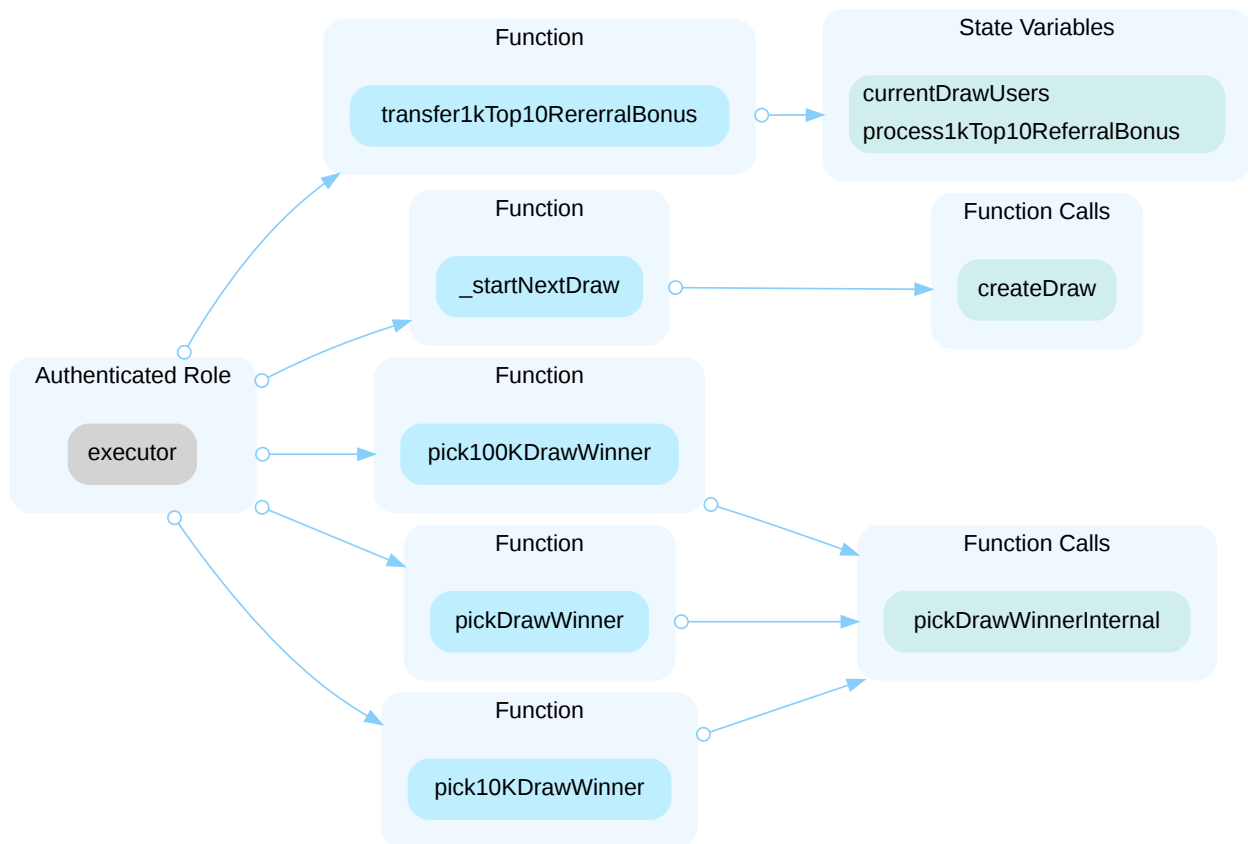
In the contract `LottoAdmin` the role `pendingOwner` has authority over the functions shown in the diagram below.



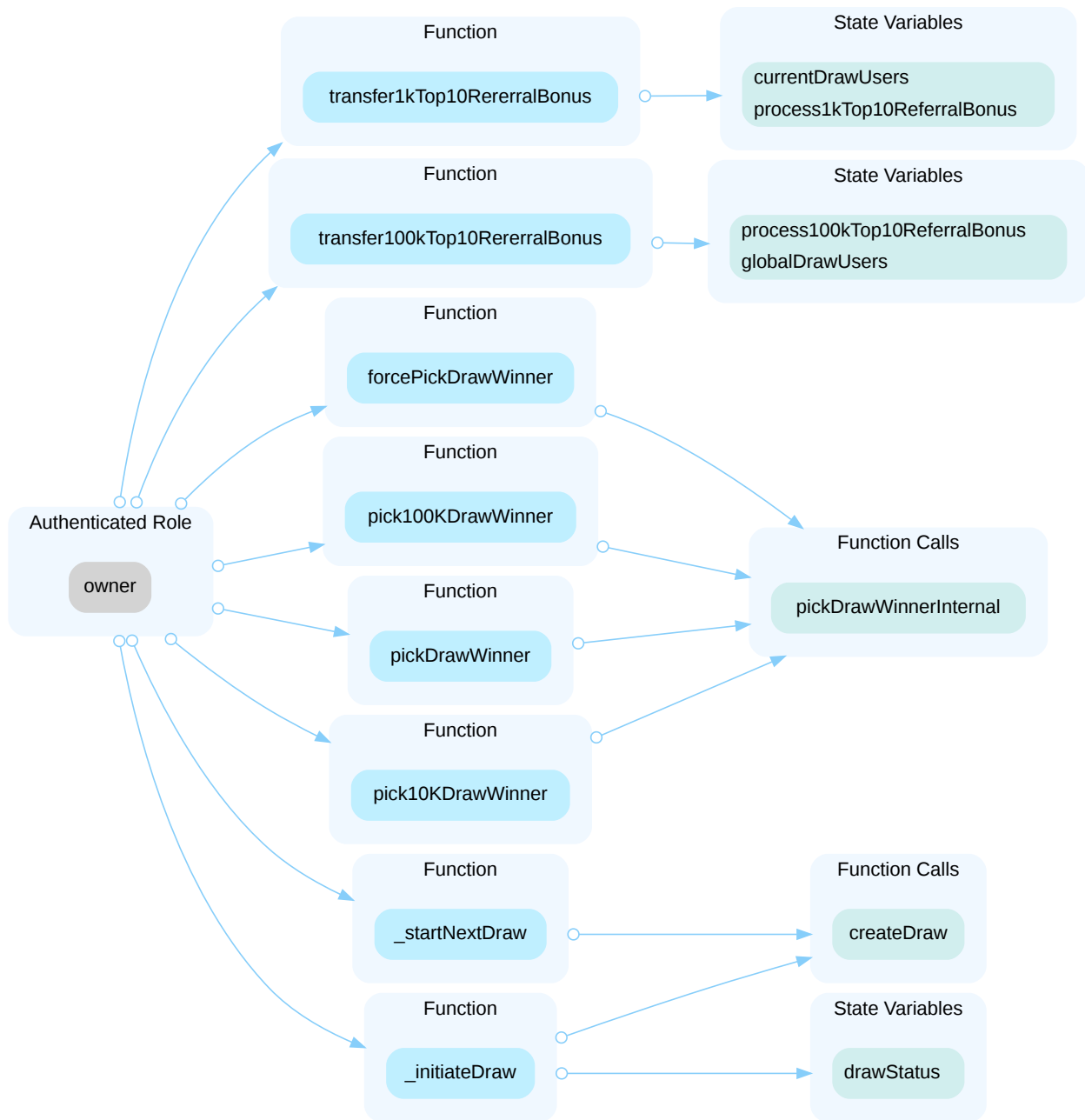
In the contract `LottoCore` the role `_owner` has authority over the functions shown in the diagram below.



In the contract `LottoCore` the role `executor` has authority over the functions shown in the diagram below.

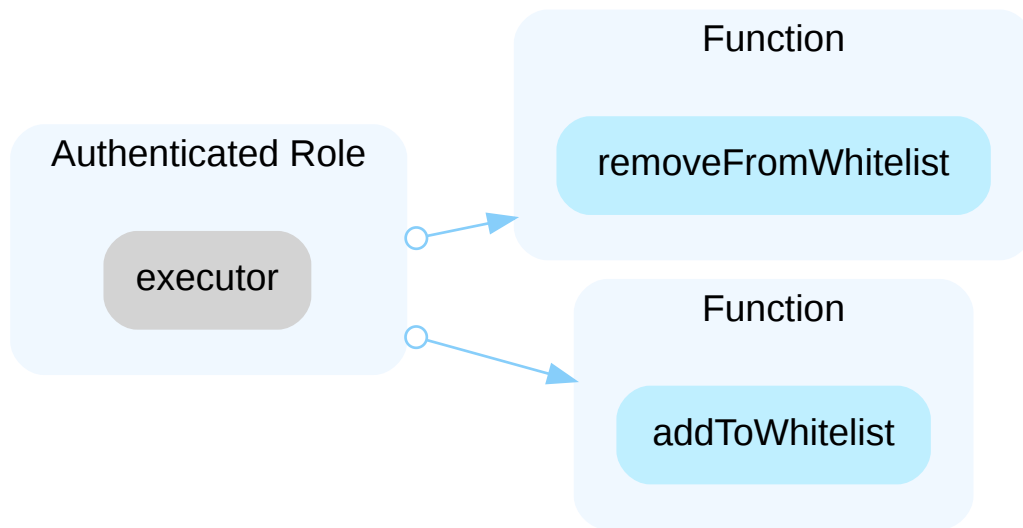


In the contract `LottoCore` the role `owner` has authority over the functions shown in the diagram below.

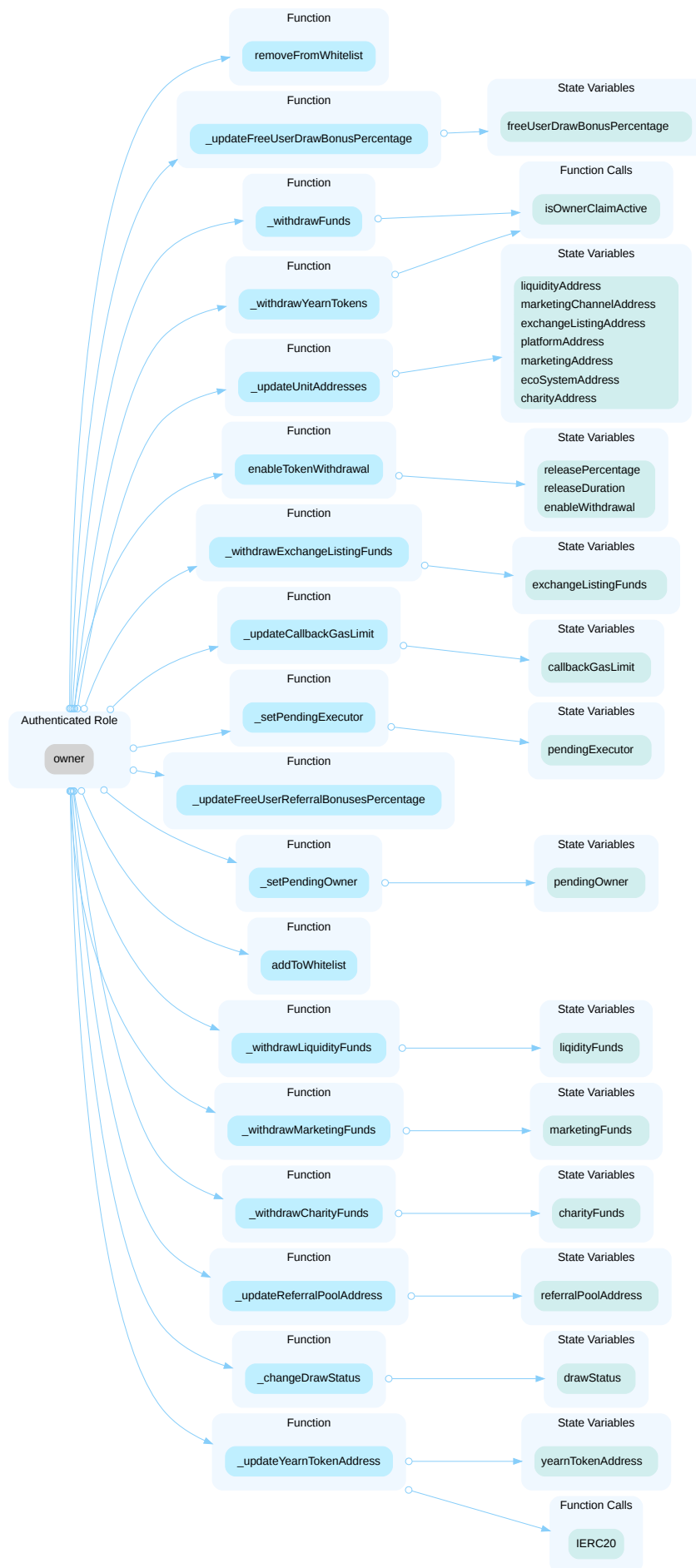


In the latest commit of the codebase [f69ba461ac71aec33cc3a7c944fea647358c1d00](#), the above privileged roles and functions have been updated as below:

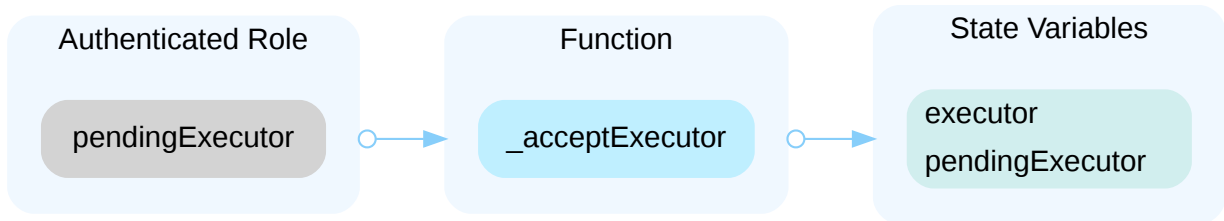
In the contract `PlatformAdmin` the role `executor` has authority over the functions shown in the diagram below.



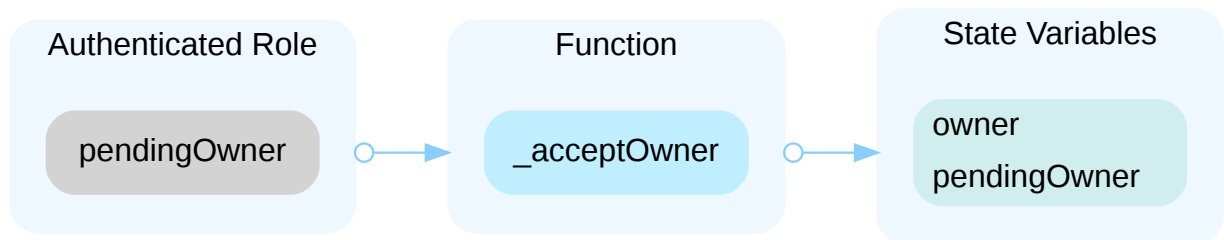
In the contract `PlatformAdmin` the role `owner` has authority over the functions shown in the diagram below.



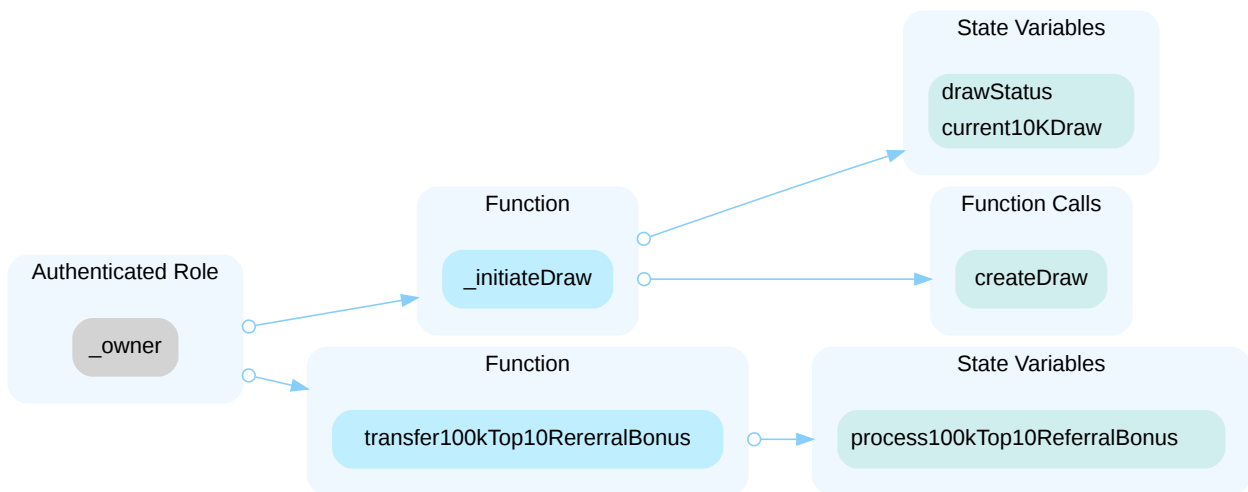
In the contract `PlatformAdmin` the role `pendingExecutor` has authority over the functions shown in the diagram below.



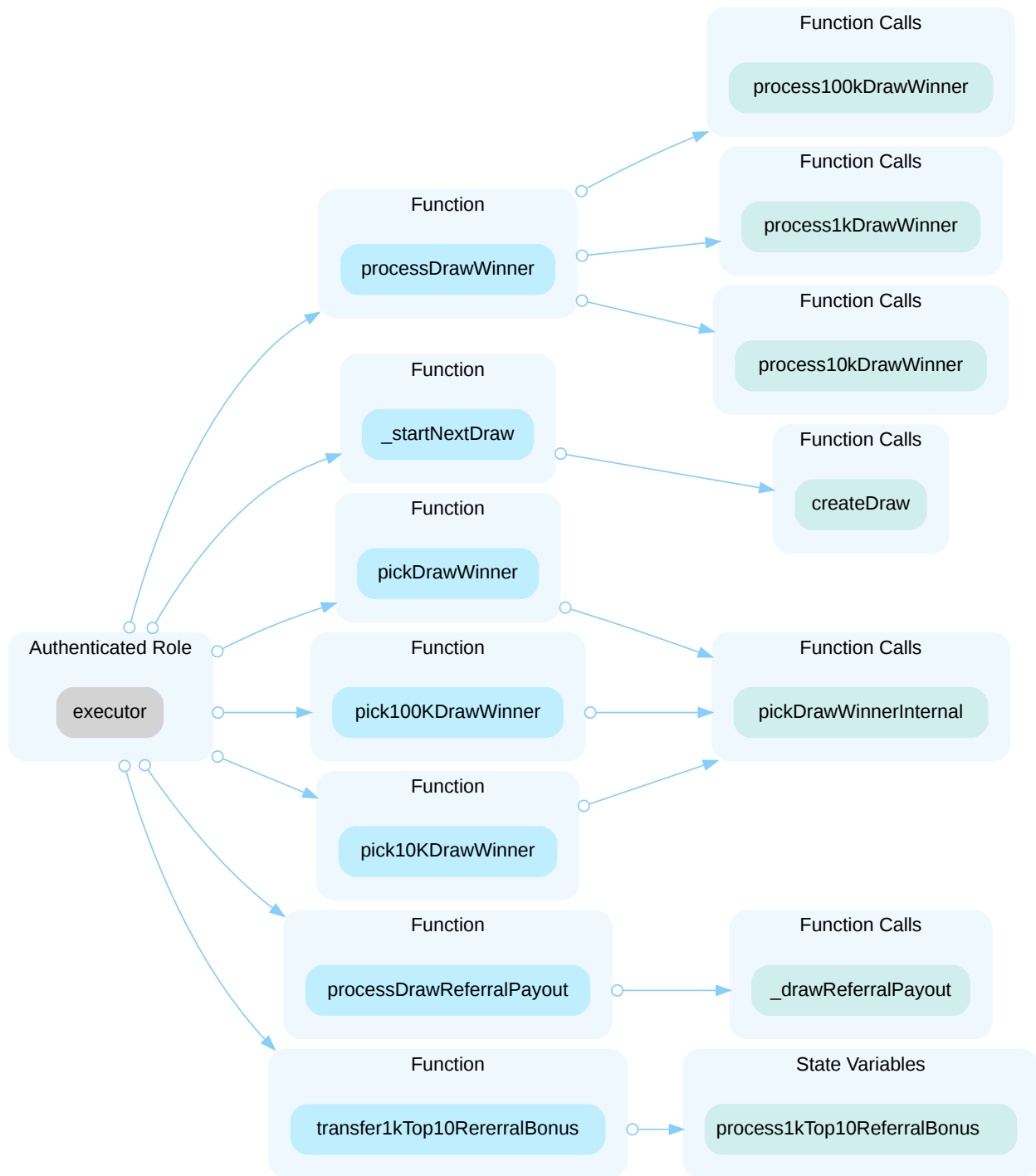
In the contract `PlatformAdmin` the role `pendingOwner` has authority over the functions shown in the diagram below.



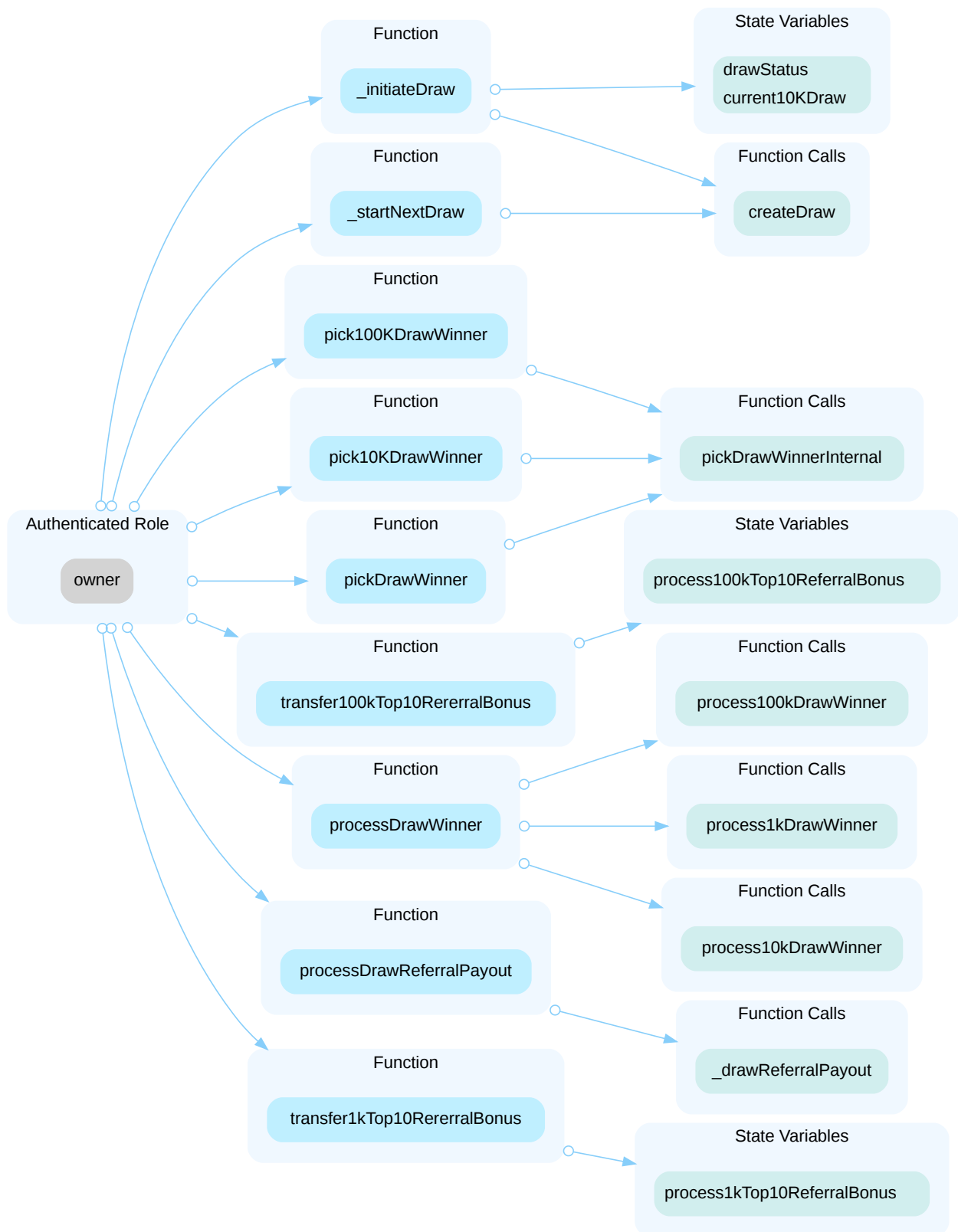
In the contract `PlatformCore` the role `_owner` has authority over the functions shown in the diagram below.



In the contract `PlatformCore` the role `executor` has authority over the functions shown in the diagram below.



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



Any compromise to the privileged account may allow the hacker to take advantage of this authority:

In `PlatformAdmin` contract:

- `_withdrawLiquidityFunds` : allows the owner to withdraw liquidity funds from the contract, with `onlyOwner` privilege

- `_withdrawExchangeListingFunds` : allows the owner to withdraw exchange listing funds from the contract, with `onlyOwner` privilege
- `_withdrawCharityFunds` : allows the owner to withdraw charity funds from the contract, with `onlyOwner` privilege
- `_withdrawMarketingFunds` : allows the owner to withdraw marketing funds from the contract, with `onlyOwner` privilege
- `isOwnerClaimActive` : returns whether the owner claim is active, with public visibility
- `_withdrawFunds` : allows the owner to withdraw the remaining funds from the contract after all draws are ended, with `onlyOwner` privilege
- `_withdrawYearnTokens` : allows the owner to withdraw the remaining yearn tokens from the contract after all draws are ended, with `onlyOwner` privilege
- `_updateFreeUserReferralBonusesPercentage` : allows the owner to update free user referral bonus percentages, with `onlyOwner` privilege
- `_updateFreeUserDrawBonusPercentage` : allows the owner to update the free user draw bonus percentage, with `onlyOwner` privilege
- `addToWhitelist` : allows the executor to add users to the whitelist, with `onlyExecutor` privilege
- `removeFromWhitelist` : allows the executor to remove users from the whitelist, with `onlyExecutor` privilege
- `_changeDrawStatus` : allows the owner to pause or resume the draw to update administrator features, with `onlyOwner` privilege
- `_updateYearnTokenAddress` : allows the owner to change the ytg token address, with `onlyOwner` privilege
- `_updateCallbackGasLimit` : allows the owner to update the callback gas limit, with `onlyOwner` privilege
- `enableTokenWithdrawal` : allows the owner to enable/disable token withdrawal with specified percentage and duration, with `onlyOwner` privilege
- `_updateUnitAddresses` : allows the owner to update all unit addresses, with `onlyOwner` privilege
- `_updateReferralPoolAddress` : allows the owner to update the referral pool address, with `onlyOwner` privilege
- `_setPendingOwner` : allows the owner to set a new pending owner, with `onlyOwner` privilege
- `_acceptOwner` : allows the new pending owner to accept the transfer of ownership, with public visibility
- `_setPendingExecutor` : allows the owner to set a new pending executor, with `onlyOwner` privilege
- `_acceptExecutor` : allows the new pending executor to accept the transfer of executorship, with public visibility
-

In `PlatformCore` contract:

- `_initiateDraw` : Owner start the initial the draw, with `onlyOwner` privilege
- `_startNextDraw` : Executor start the next draw, with `onlyExecutor` privilege
- `pickDrawWinner` : Executor pick the draw winner, and based on the draw, 10k and 100k draw are calculated, with `onlyExecutor` privilege
- `pick10KDrawWinner` : Calculate 10k draw winner, with `onlyExecutor` privilege
- `pick100KDrawWinner` : Calculate 100k draw winner, with `onlyExecutor` privilege
- `processDrawWinner` : Executor process the draw winner of 1k, 10k and 100k draw, with `onlyExecutor` privilege
- `processDrawReferralPayout` : Executor process the draw referral payout of 1k, 10k and 100k draw, with `onlyExecutor` privilege
- `transfer1kTop10ReferralBonus` : Owner process top 10 1K draw referral bonus, with `onlyExecutor` privilege
- `transfer100kTop10ReferralBonus` : Owner process top 10 global 100K draw referral bonus, with `onlyOwner` privilege

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 ($\frac{2}{3}$, $\frac{3}{5}$) 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.

I Alleviation

[Certik]: The team deployed the codebase at <https://bscscan.com/address/0xe61aB07FE287D85FdBC2Be0b3a5a9F1b4D6BFF45> and will verify the codebase in the future.

D1E-01 | INCOMPATIBILITY WITH DEFLATIONARY TOKENS

Category	Severity	Location	Status
Logical Issue	● Medium	LottoAdmin.sol (d1e20147a4cc5797a51baa0f966353e4bedc201a): 21, 23, 33, 35, 45, 47, 57, 59; LottoCore.sol (d1e20147a4cc5797a51baa0f966353e4bedc201a): 230, 237, 292, 308	● Acknowledged

Description

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

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

```
23      tokenAddress.safeTransfer(liquidityAddress, amount);
```

- Transferring tokens by `amount`.

```
21      liquidityFunds -= amount;
```

- The `amount` appears to be used for bookkeeping purposes without compensating the potential transfer fees.

```
35      tokenAddress.safeTransfer(exchangeListingAddress, amount);
```

- Transferring tokens by `amount`.

```
33      exchangeListingFunds -= amount;
```

- The `amount` appears to be used for bookkeeping purposes without compensating the potential transfer fees.

```
47      tokenAddress.safeTransfer(charityAddress, amount);
```

- Transferring tokens by `amount`.


```
45      charityFunds -= amount;
```

- The `amount` appears to be used for bookkeeping purposes without compensating the potential transfer fees.

```
59      tokenAddress.safeTransfer(marketingChannelAddress, amount);
```

- Transferring tokens by `amount`.

```
57      marketingFunds -= amount;
```

- The `amount` appears to be used for bookkeeping purposes without compensating the potential transfer fees.

```
230      tokenAddress.safeTransferFrom(msg.sender, address(this), amount);
```

- Transferring tokens by `amount`.

```
237      uint256 drawAmount = amount - (draws[_draw].fee * numberOfTickets);
```

- The `amount` appears to be used for bookkeeping purposes without compensating the potential transfer fees.

```
292      depositInternal(currentDraw, _referrer, amount);
```

- Transferring tokens by `amount`.
- This function call executes the following operation.
- In `LottoCore.depositInternal`,
 - `tokenAddress.safeTransferFrom(msg.sender, address(this), amount);`

```
292      depositInternal(currentDraw, _referrer, amount);
```

- This function call executes the following operation.
- In `LottoCore.depositInternal`,
 - `uint256 drawAmount = amount - (draws[_draw].fee * numberOfTickets);`

- The `amount` appears to be used for bookkeeping purposes without compensating the potential transfer fees.

```
308         depositInternal(currentDraw, _referrer, amount);
```

- Transferring tokens by `amount`.
- This function call executes the following operation.
- In `LottoCore.depositInternal`,
 - `tokenAddress.safeTransferFrom(msg.sender, address(this), amount);`

```
308         depositInternal(currentDraw, _referrer, amount);
```

- This function call executes the following operation.
- In `LottoCore.depositInternal`,
 - `uint256 drawAmount = amount - (draws[_draw].fee * numberOfTickets);`
- The `amount` appears to be used for bookkeeping purposes without compensating the potential transfer fees.

Recommendation

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

Alleviation

[Yearn Together]: In our platform, as per design we collect the fee when they deposit the amount in the beginning itself.

We don't collect any fee while withdrawal.

So can be accepted without any coding corrections

LAB-01 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	LottoAdmin.sol (d1e20147a4cc5797a51baa0f966353e4bedc201a): 159, 160, 161, 162, 163, 164, 165, 169, 180, 206	Resolved

Description

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

```
159         platformAddress = _platformAddress;
```

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

```
160         liquidityAddress = _liquidityAddress;
```

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

```
161         marketingAddress = _marketingAddress;
```

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

```
162         ecoSystemAddress = _ecoSystemAddress;
```

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

```
163         exchangeListingAddress = _exchangeListingAddress;
```

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

```
164         charityAddress = _charityAddress;
```

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

```
165      marketingChannelAddress = _marketingChannelAddress;
```

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

```
169      referralPoolAddress = _referralPoolAddress;
```

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

```
180      pendingOwner = newPendingOwner;
```

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

```
206      pendingExecutor = newPendingExecutor;
```

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

Recommendation

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

Alleviation

[Yearn Together]: Issue acknowledged. Changes have been reflected in the commit hash:

<https://github.com/yearn/yearn-lottopad/commit/30a9757a25b84516062c2272b2e4e65b4a4fa11e>

LCB-03 | UNSAFE CAST

Category	Severity	Location	Status
Logical Issue	● Minor	LottoCore.sol (d1e20147a4cc5797a51baa0f966353e4bedc201a): 69	● Resolved

Description

In the statement `uint40(block.timestamp);`, it casts a `uint256` value to an `uint40` without evaluating its bounds.

Recommendation

We advise a safe casting operation to be performed by ensuring the result is still positive as high numbers will cause an underflow to occur here, thereby causing the system to misbehave.

Alleviation

[Yearn Together]: Issue acknowledged. Changes have been reflected in the commit hash:
<https://github.com/yearn/yearn-protocol/commit/30a9757a25b84516062c2272b2e4e65b4a4fa11e>

LCB-02 | MISSING VALIDATION

Category	Severity	Location	Status
Logical Issue	● Informational	LottoCore.sol (d1e20147a4cc5797a51baa0f966353e4bedc201a): 405	● Resolved

Description

```
405 tokenAddress.safeTransfer(referrer, bonus);
```

In the function `_drawReferralPayout()`, a certain amount of token will be transferred to `referrer` as a bonus. However there's validation to check if `bonus` is 0. Bonus with a value of 0 will still be transferred to the referrer, which may result in unnecessary gas fees and token transfers.

Recommendation

We recommend adding validation to check if `bonus` is 0 before the transfer

```
405 if (bonus > 0) {  
406   tokenAddress.safeTransfer(referrer, bonus);  
407 }
```

Alleviation

[Yearn Together]: Issue acknowledged. Changes have been reflected in the commit hash:
<https://github.com/yearn/yearncontracts/commit/30a9757a25b84516062c2272b2e4e65b4a4fa11e>

LCB-04 | LACK OF SELF-REFERENCE CHECK IN THE `_setReferrer()`

Category	Severity	Location	Status
Logical Issue	● Informational	LottoCore.sol (d1e20147a4cc5797a51baa0f966353e4bedc201a): 133, 291	● Resolved

Description

In function `_setReferrer()` there's no validation to check if `_referrer` could be `msg.sender` address itself. This may potentially violate the design.

Recommendation

We recommend team check if the design allows any participant use same participant address as referrer

Alleviation

[Yearn Together]: Issue acknowledged. Changes have been reflected in the commit hash:
<https://github.com/yearn/yearn-protocol/commit/30a9757a25b84516062c2272b2e4e65b4a4fa11e>

LCB-05 | USER CAN FAVOR USDT AND DEPRECIATE YTG TOKEN

Category	Severity	Location	Status
Logical Issue	● Informational	LottoCore.sol (d1e20147a4cc5797a51baa0f966353e4bedc201a): 311, 325	● Resolved

Description

There are two functions, `claimToken()` and `claimAmount()` allows the user to withdraw `YTG` token and `USDT` token under certain conditions. The user will always select the most beneficial token to withdraw. If at the early stage, the price of `YTG` token is less than the price of `USDT`, then the user will withdraw `USDT` and depreciate the `YTG` token, which will also potentially drain the deposited `USDT` tokens in the LottoCore deployment

Meanwhile, `claimAmount()` has less validations comparing to `claimToken()`, which leads the user could favor withdrawing USDT compared to waiting `claimToken()` to be enabled.

Recommendation

We recommend the team double confirm the design and check if the current `claimAmount()` implementation aligns with the design.

Alleviation

[Yearn Together]: `claimAmount()` and `claimToken()` are performing 2 different activities.

In our platform the user can claim both our token (YTG) and the USDT.

USDT - distributed in our platform who ever perform in our game play elements

YTG - distributed even for participation

OPTIMIZATIONS | YEARN TOGETHER

ID	Title	Category	Severity	Status
LAB-02	Redundant Validation	Gas Optimization	Optimization	● Acknowledged
LSB-01	State Variable Should Be Declared Constant	Gas Optimization	Optimization	● Resolved

LAB-02 | REDUNDANT VALIDATION

Category	Severity	Location	Status
Gas Optimization	● Optimization	LottoAdmin.sol (d1e20147a4cc5797a51baa0f966353e4bedc201a): 190, 216	● Acknowledged

Description

```
187 function _acceptOwner() external {
188     // Check caller is pendingOwner and pendingOwner ≠ address(0)
189     require(msg.sender == pendingOwner);
190     require(msg.sender != address(0));
191
192     // Store owner with value pendingOwner
193     owner = pendingOwner;
194
195     // Clear the pending value
196     pendingOwner = address(0);
197 }
```

In above code snippet, the line `require(msg.sender == pendingOwner);` already checks if `msg.sender` is equal to `pendingOwner`, which means that `msg.sender` cannot be equal to `address(0)` (which represents the null address) because `pendingOwner` is also not equal to `address(0)`.

In other words, the line `require(msg.sender == pendingOwner);` already ensures that `msg.sender` is a valid address and not equal to the null address. Therefore, there is no need to include the additional check `require(msg.sender != address(0));` in the code.

A similar optimization chance happens in `_acceptExecutor()` function

Recommendation

We advise the team omitting the aforementioned `require(msg.sender != address(0));` validation from `_acceptOwner()` function, and `require(msg.sender != address(0));` in `_acceptExecutor()` function

LSB-01 | STATE VARIABLE SHOULD BE DECLARED CONSTANT

Category	Severity	Location	Status
Gas Optimization	● Optimization	LottoStorage.sol (d1e20147a4cc5797a51baa0f966353e4bedc201a): 109, 120, 124, 198	● Resolved

Description

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

```
109     bytes32 keyHash =  
0x4b09e658ed251bcafeebbc69400383d49f344ace09b9576fe248bb02c003fe9f;
```

- `keyHash` should be declared `constant`.

```
120     uint16 requestConfirmations = 3;
```

- `requestConfirmations` should be declared `constant`.

```
124     uint32 numWords = 1;
```

- `numWords` should be declared `constant`.

```
198     uint8 public EXTRA_BONUS_PURCHASE_REWARDS = 5;           // whitelisted user  
get extra 5% bonus token on top of purchase bonus
```

- `EXTRA_BONUS_PURCHASE_REWARDS` should be declared `constant`.

Recommendation

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

Alleviation

[Yearn Together]: Issue acknowledged. Changes have been reflected in the commit hash:

<https://github.com/yearn-together/lottopad/commit/30a9757a25b84516062c2272b2e4e65b4a4fa11e>

APPENDIX | YEARN TOGETHER

Finding Categories

Categories	Description
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.

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 | CERTIK

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.

CertiK | Securing the Web3 World

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



