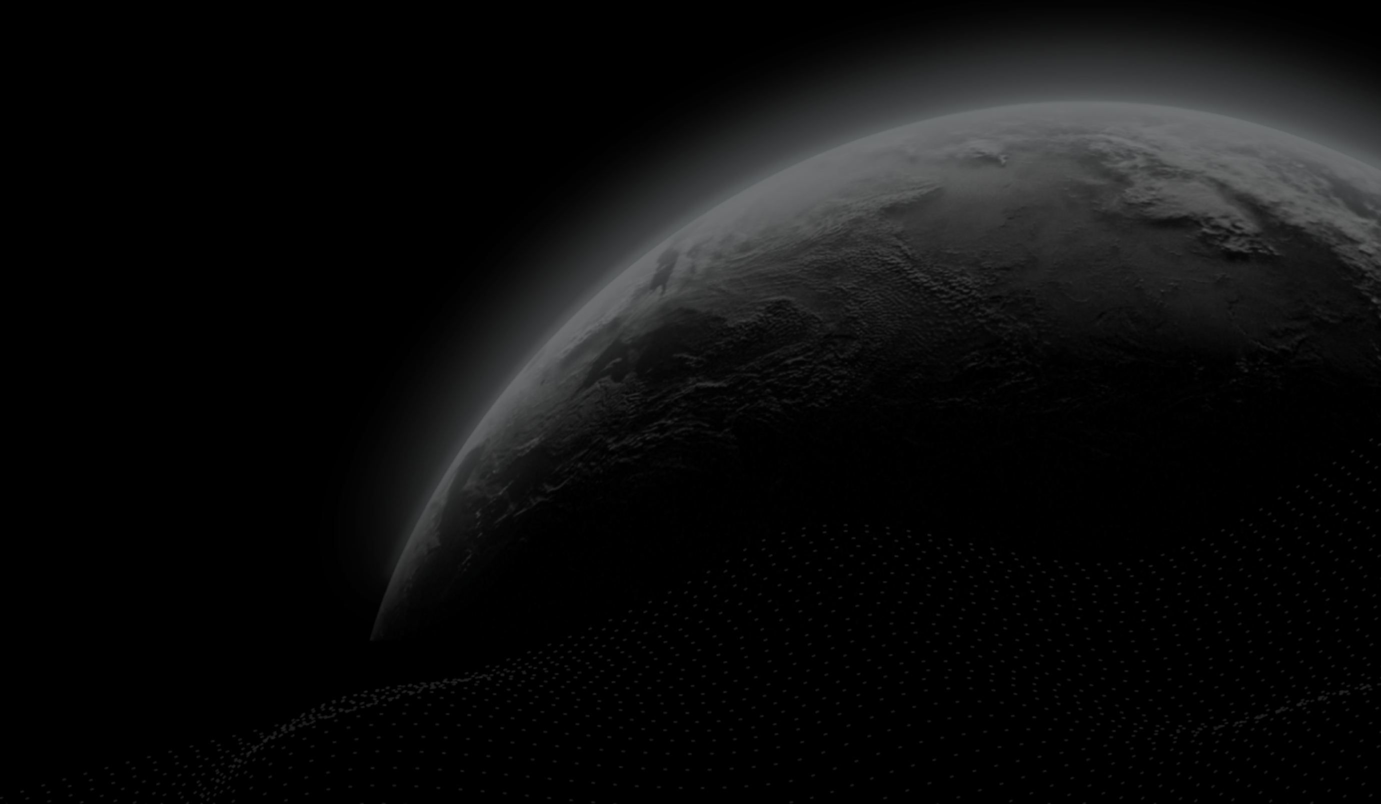
CERTIK

Security Assessment

Big Crypto Game - DUELS

CertiK Verified on Nov 4th, 2022



CertiK Verified on Nov 4th, 2022

Big Crypto Game - DUELS

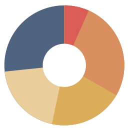
The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Ethereum	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Solidity	Delivered on 11/04/2022	N/A

CODEBASE	COMMITS
https://github.com/Crypto-Legions/Big-Crypto-Game-Contracts/tree/0a55e19734f45ece88f1f04d16b744a6825af00f ...View All	0a55e19734f45ece88f1f04d16b744a6825af00f ...View All

Vulnerability Summary



15	9	0	0	6	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

🟥 1	Critical	1 Resolved	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.
🟧 4	Major	4 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.
🟨 3	Medium	3 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨 3	Minor	1 Resolved, 2 Acknowledged	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.
🟦 4	Informational	4 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 | BIG CRYPTO GAME - DUELS

CODEBASE | BIG CRYPTO GAME - DUELS

Repository

https://github.com/Crypto-Legions/Big-Crypto-Game-Contracts/tree/0a55e19734f45ece88f1f04d16b744a6825af00f

Commit

0a55e19734f45ece88f1f04d16b744a6825af00f

AUDIT SCOPE | BIG CRYPTO GAME - DUELS

5 files audited ● 2 files with Acknowledged findings ● 3 files without findings

ID	File	SHA256 Checksum
● WNF	contracts/WarriorNFT.sol	b87a599bf73621f1618d807c9303b946aed8188d843f69537a0cd5c8645a812c
● DSB	contracts/DuelSystem.sol	7c5ba1496083944d97ab418b843ec285bb6cf0043d3592518f072fd0089e9b64
● RPB	contracts/RewardPool.sol	5b2d35b9d3eaf2d88f901e92dbee8ee4f2e3ae1e2c33b8fbd19d9580b4256c84
● IRP	contracts/interfaces/IRewardPool.sol	931080f4f23f6932148c3d5ad2e8c8aebc2d6b4da6fe49bd8acbc93ec2b8ff96
● LNF	contracts/LegionNFT.sol	f9816c4b1dcdb126390b95b0123ba94031654af27712a00acef2bd6a95c8c7cc

APPROACH & METHODS | BIG CRYPTO GAME - DUELS

This report has been prepared for Big Crypto Game to discover issues and vulnerabilities in the source code of the Big Crypto Game - DUELS 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.

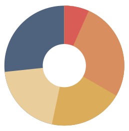
The auditing process pays special attention to the following considerations:

- 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.

FINDINGS | BIG CRYPTO GAME - DUELS



15	1	4	3	3	4
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Big Crypto Game - DUELS. Through this audit, we have uncovered 15 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
BCG-01	Missing Zero Address Validation	Volatile Code	Minor	● Acknowledged
DSB-01	Missing Check For `msg.sender`	Logical Issue	Critical	● Resolved
DSB-02	**Centralization Risks In DuelSystem.Sol**	**Centralization / Privilege**	**Major**	● **Acknowledged**
DSB-03	Missing Lower Limit Of `winnerPercent`	Logical Issue	Medium	● Resolved
DSB-04	Missing Check For New Price	Logical Issue	Medium	● Resolved
DSB-05	Potential Flash Loan Attack	Logical Issue, Language Specific	Minor	● Acknowledged
DSB-06	Missing Input Validation	Logical Issue	Minor	● Resolved
DSC-01	Principal Is Not Deducted When `standard` Flag Is False	Logical Issue	Medium	● Resolved
LNF-01	**Centralization Risks In LegionNFT.Sol**	**Centralization / Privilege**	**Major**	● **Acknowledged**
RPB-01	**Centralization Risks In RewardPool.Sol**	**Centralization / Privilege**	**Major**	● **Acknowledged**

ID	Title	Category	Severity	Status
WNF-01	**Centralization Risks In WarriorNFT.Sol**	**Centralization / Privilege**	**Major**	● **Acknowledged**
DSB-07	Incorrect `betAmounts` Maximum Limit	Logical Issue	Informational	● Resolved
DSB-08	Inconsistent Document And Codebase	Logical Issue	Informational	● Resolved
WNF-02	Potential Index Out-Of-Range Error	Logical Issue	Informational	● Resolved
WNF-03	Missing Removing Whitelist Feature	Logical Issue	Informational	● Resolved

BCG-01 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	LegionNFT.sol (d673ead): 691; RewardPool.sol (d673ead): 159	● Acknowledged

Description

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

```
691         duel = _addr;
```

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

```
159         duel = _duel;
```

- `_duel` 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

[`Big Crypto`]: Issue acknowledged.

DSB-01 | MISSING CHECK FOR `msg.sender`

Category	Severity	Location	Status
Logical Issue	● Critical	contracts/DuelSystem.sol: 66	● Resolved

Description

File: DuelSystem.sol

According to the logic of the function `cancelDuel()`, a duel that already exists can be cancelled if it has been created but not finished yet. If the `standard` flag is set to `true`, the `betAmount` `BLST` will be added to the `msg.sender` account. However, no relationship between `msg.sender` and the duel is checked. Therefore, a hacker can deploy a contract to loop all the duels that pass the `require` verification and get the whole `BLST`.

```
66     function cancelDuel(uint256 duelId) external {
67          require(duels[duelId].status==1, "Duel is already started or not
created");
68          require(doingDuels[duels[duelId].legion1], "it's not started duel");
69          if(duels[duelId].standard) {
70              rewardpool.addReward(msg.sender, duels[duelId].betAmount);
71          }
72          duels[duelId].status = 0;
73          doingDuels[duels[duelId].legion1] = false;
74      }
```

Recommendation

We recommend adding logic to check if the `legion1` of duel is owned by the `msg.sender`.

Alleviation

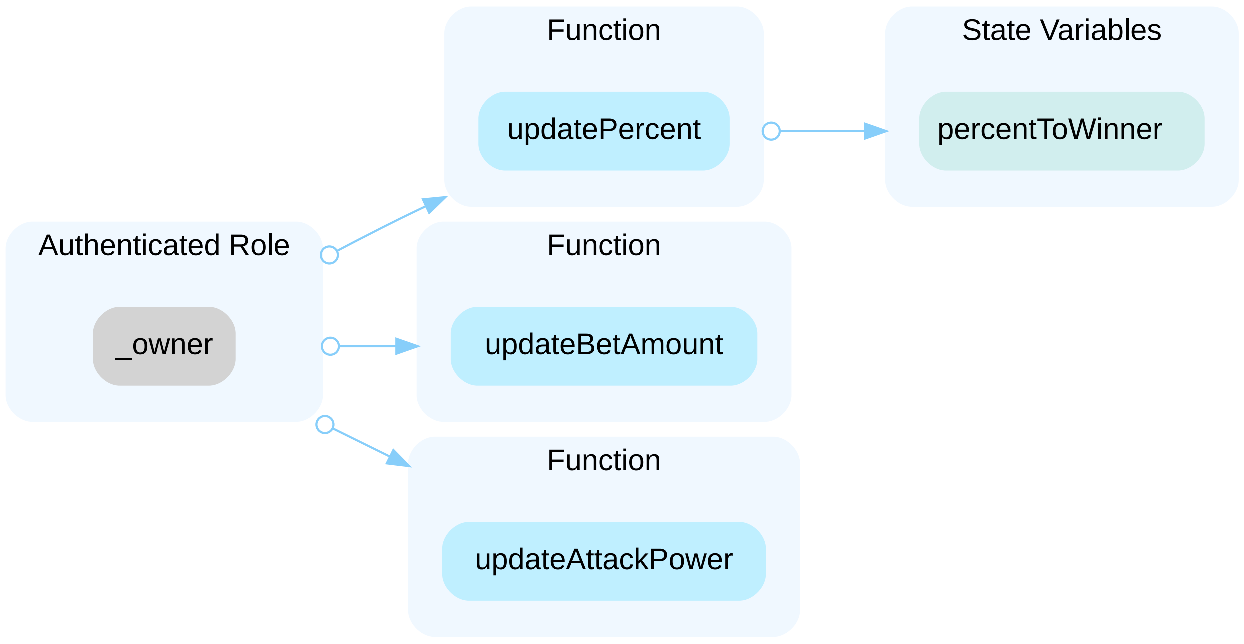
[`Certik`]: The team resolved this issue in the commit 1ffc38ca75d2745e0e33498fc89be7d1a4ea6f5c.

DSB-02 | CENTRALIZATION RISKS IN DUELSYSTEM.SOL

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**contracts/DuelSystem.sol: 105, 110, 115**	● **Acknowledged**

Description

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



Recommendation

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

Short Term:

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;

  AND

- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

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

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

Permanent:

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

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

Alleviation

[`Big Crypto Game`]: Issue acknowledged.

DSB-03 | MISSING LOWER LIMIT OF `winnerPercent`

Category	Severity	Location	Status
Logical Issue	● Medium	contracts/DuelSystem.sol: 116, 128, 133	● Resolved

Description

File: DuelSytem.sol

The bet amount of the two participants of a duel is the same, so each one's deposited amount is fifty percent of the whole bet amount. If the `Duel` is standard, the winner's reward should be more than fifty percent of the bet amount. Otherwise, the winner will gain nothing but lose a proportion of his/her bet amount even if he/she wins the duel.

```
115     function updatePercent(uint256 winnerPercent) external onlyOwner {
116         require(winnerPercent<=100, "bigger than 100");
117         percentToWinner = winnerPercent;
118     }
```

```
128     rewardpool.addReward(legion.ownerOf(duels[duelId].legion2),
duels[duelId].betAmount*percentToWinner/100);
```

```
133     rewardpool.addReward(legion.ownerOf(duels[duelId].legion1),
duels[duelId].betAmount*percentToWinner/100);
```

Recommendation

We recommend refactoring codes to ensure that the winner won't lose his/her bet amount.

Alleviation

[`Certik`]: The team resolved this issue in the commit 1ffc38ca75d2745e0e33498fc89be7d1a4ea6f5c.

DSB-04 | MISSING CHECK FOR NEW PRICE

Category	Severity	Location	Status
Logical Issue	● Medium	contracts/DuelSystem.sol: 96, 120	● Resolved

Description

File: DuelSystem.sol

Function `updatePrediction()` is used to update the prediction price for `price1` or `price2` .

```
96      function updatePrediction(uint256 duelId, uint256 price) external {
97          require(block.timestamp < duels[duelId].startTime+invitePeriod,
"Invitation is expired");
98          if(legion.ownerOf(duels[duelId].legion1) == msg.sender) {
99              duels[duelId].price1 = price;
100         } else if(legion.ownerOf(duels[duelId].legion2) == msg.sender) {
101             duels[duelId].price2 = price;
102
```

According to the logic of function `finishDuel()` , the owner of `legion1` will be the winner if the difference between `price1` and `price2` is equal. But this is not reasonable and inconsistent with the document provided by the client.

```
126     if(diff1>diff2) {
127               if(duels[duelId].standard) {
128                   rewardpool.addReward(legion.ownerOf(duels[duelId].legion2),
duels[duelId].betAmount*percentToWinner/100);
129               }
130               legion.updateApAfterDuel(duels[duelId].legion2,
duels[duelId].legion1, duels[duelId].standard);
131           } else {
132               if(duels[duelId].standard) {
133                   rewardpool.addReward(legion.ownerOf(duels[duelId].legion1),
duels[duelId].betAmount*percentToWinner/100);
134               }
135               legion.updateApAfterDuel(duels[duelId].legion1,
duels[duelId].legion2, duels[duelId].standard);
136           }
```

Recommendation

We recommend adding a check to the function `updatePrediction()` that `price1` cannot equal `price2` , just like function `joinDuel()` .

Alleviation

[`Certik`]: The team resolved this issue in the commit 1ffc38ca75d2745e0e33498fc89be7d1a4ea6f5c.

DSB-05 | POTENTIAL FLASH LOAN ATTACK

Category	Severity	Location	Status
Logical Issue, Language Specific	● Minor	contracts/DuelSystem.sol: 120	● Acknowledged

Description

File: DuelSystem.sol

The price of `BLST` simply equals the amount of `BUSD` tokens swapped out from DEX by using one `BLST` . The swapped-out amount is determined by the reserves of `BLST` and `BUSD` in the `DEX` pair. The change of reserves would impact the swapped-out amount of `BUSD` , the price of `BLST` .
As an example, in the following scenario, the attacker can win the duel by manipulating the price of `BLST` . Let's assume that the price of `BLST` is `$0.02` now in pancake dex.

1. Alice creates a duel and set the `price1` to `$0.05` .
2. The attacker joins the duel created at step 1 and set the `price2` to `$0.1` .
3. When the duel ends, the attacker can use `Flash Loan` to gain a large amount of `BUSD` and raise the price of `BLST` in the `DEX` to be greater than `$10` , then call the `finishDuel()` function.
4. In this case, the attacker surely wins the duel and increases his/her legion's attacker power.
5. At the end, the `resultPrice` is set to the manipulated price.

During the above process, the attacker needs to pay for the fees of the flash loan, the swap from `BUSD` to `BLST` , and the swap from `BLST` to `BUSD` .

Recommendation

Considering the give and take, such attacks don't always happen. However, we recommend the client to consider this scenario. A potential solution to protect the price of the `BLST` from manipulation is the use of a price oracle.

Alleviation

[`Big Crypto Game`]: Issue acknowledged. I do not think we need to protect attackers to manipulate the `BLST` price to win the duel, because the risk/cost for a flash loan is much higher than the potential money a hacker can earn with a duel win.

DSB-06 | MISSING INPUT VALIDATION

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/DuelSystem.sol: 105, 110	● Resolved

Description

File: DuelSystem.sol

The state variables `attackPowers` and `betAmounts` have items in ascending order. To keep the correct order, it is necessary to check that the value of the parameter `ap` / `amount` is greater than the previous item and less than the latter item. Ignoring the insertion operation if the `ap` / `amount` already exists.

```
    function updateAttackPower(uint8 index, uint256 ap) external onlyOwner {
         require(index<10, "Index is out of range");
         attackPowers[index] = ap;
     }
```

```
    function updateBetAmount(uint8 index, uint256 amount) external onlyOwner {
         require(index<10, "Index is out of range");
         betAmounts[index] = amount;
     }
```

Recommendation

We recommend refactoring codes to ensure that the ascending order is maintained after the insertion operation.

Alleviation

[`Certik`]: The team resolved this issue in the commit 1ffc38ca75d2745e0e33498fc89be7d1a4ea6f5c.

DSC-01 | PRINCIPAL IS NOT DEDUCTED WHEN `standard` FLAG IS FALSE

Category	Severity	Location	Status
Logical Issue	● Medium	DuelSystem.sol (d673ead): 57, 85	● Resolved

Description

File: DuelSystem.sol

According to the logic of functions `createDuel()/joinDuel()` . No `betAmount` `USD` will be deducted from the players if the `standard` flag is `false` . Could you please confirm is this your design?

In addition, we think that `standard` flag is also used to determine if the mode is all in. Could you please introduce the function of the `standard` flag?

```
57      if(standard) {
58         rewardpool.subReward(msg.sender, betAmount*10**18);
59      }
```

Recommendation

We recommend refactoring code if above logic does not conform to the design.

Alleviation

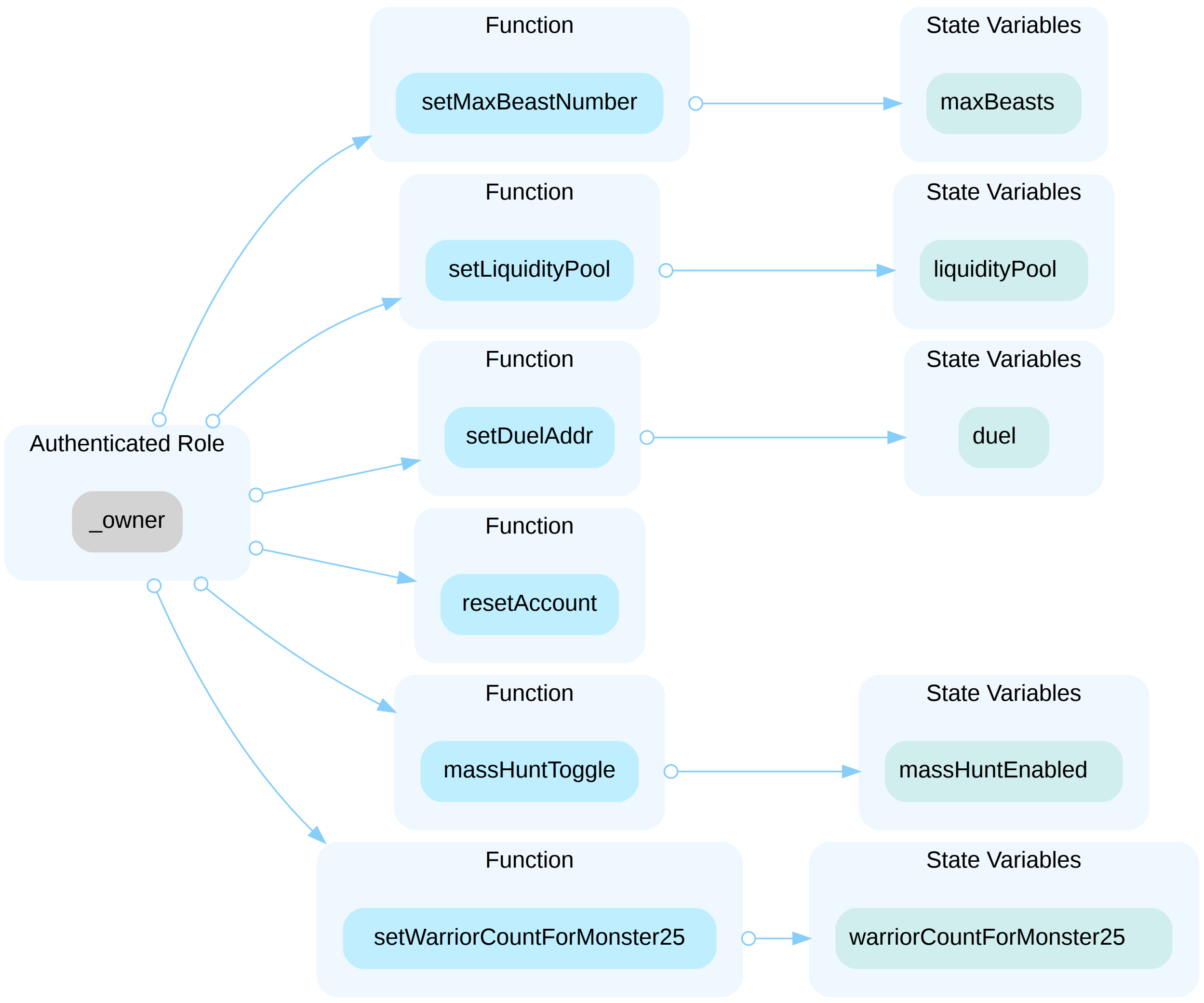
[`Big Crypto Game`]: Indeed, all-in duels bet amounts should also be deducted from the Unclaimed Wallet. [`Certik`]: The team resolved this finding in the commit 1ffc38ca75d2745e0e33498fc89be7d1a4ea6f5c.

LNF-01 | CENTRALIZATION RISKS IN LEGIONNFT.SOL

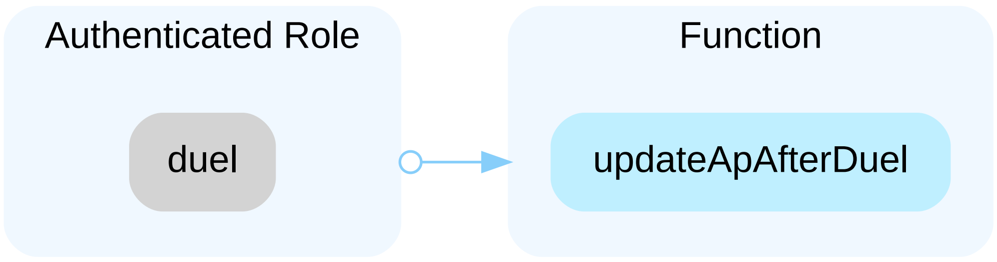
Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**LegionNFT.sol (d673ead): 229, 266, 295, 376, 630, 638, 646, 683, 690**	● **Acknowledged**

Description

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



Recommendation

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

Short Term:

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

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

Long Term:

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

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

Permanent:

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

- Renounce the ownership and never claim back the privileged roles.

  OR
- Remove the risky functionality.

Alleviation

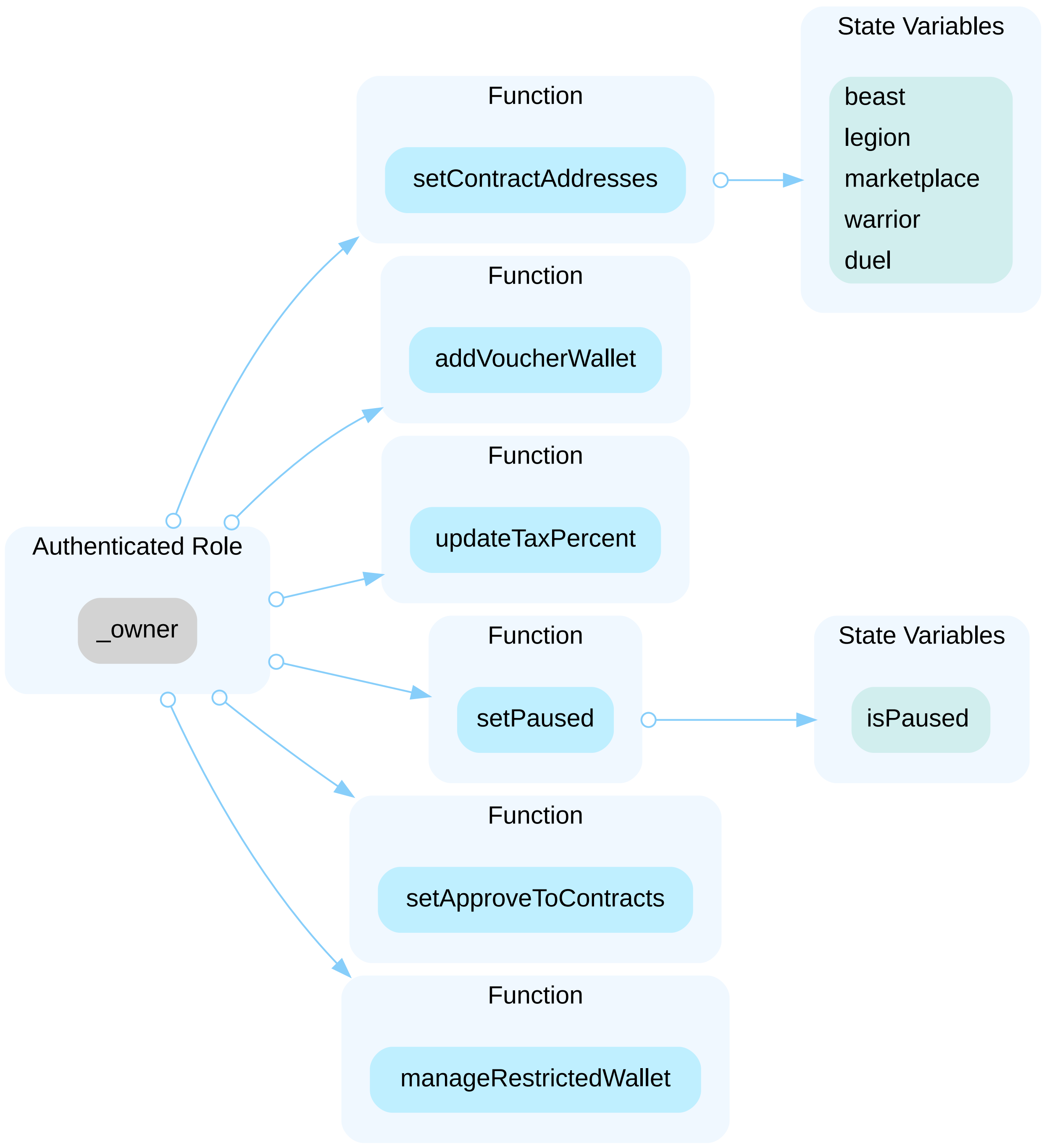
[`Big Crypto Game`]: Issue acknowledged.

RPB-01 | CENTRALIZATION RISKS IN REWARDPOOL.SOL

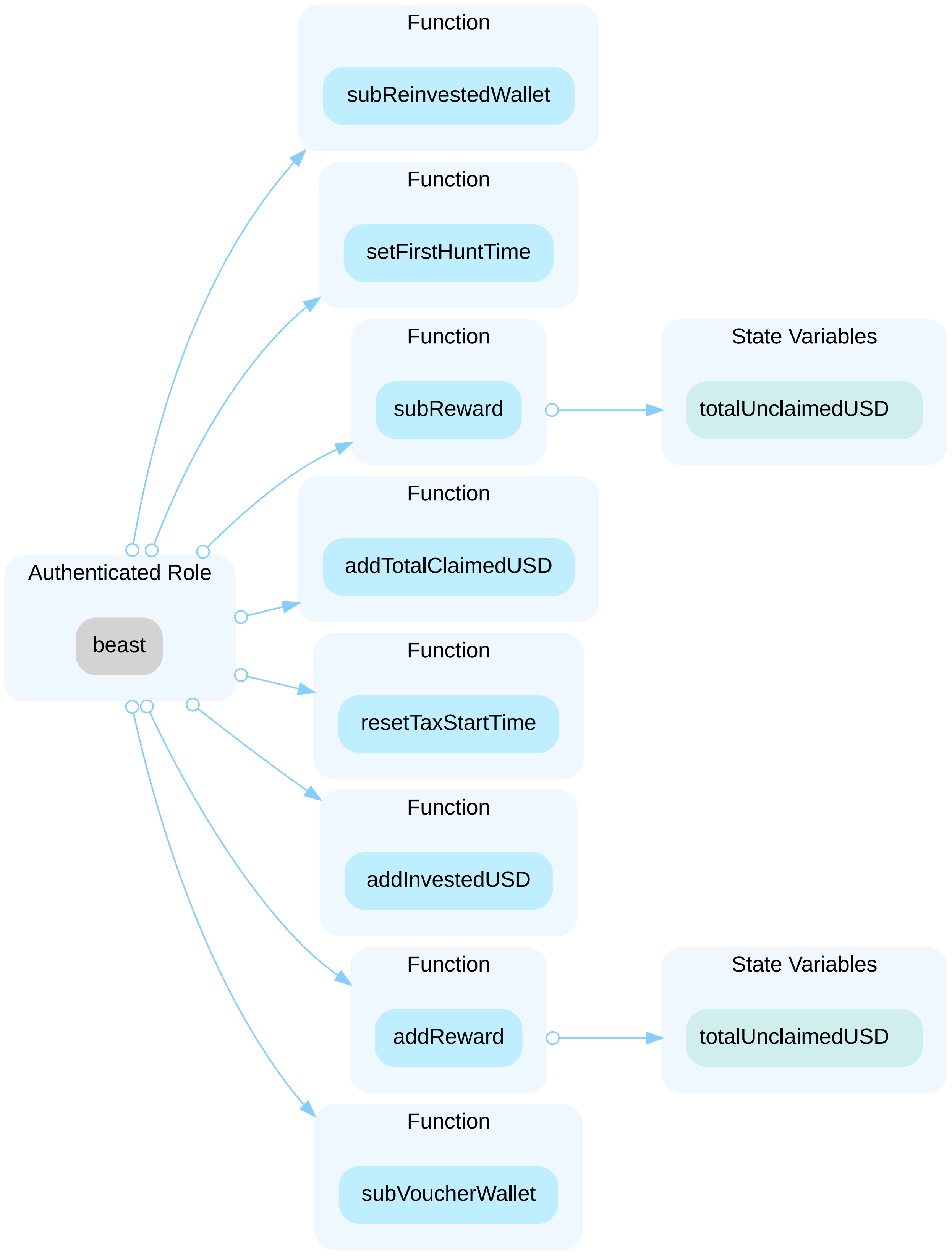
Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**RewardPool.sol (d673ead): 129, 148, 179, 189, 201, 214, 238, 250, 403, 534, 549, 557, 567, 578**	● **Acknowledged**

Description

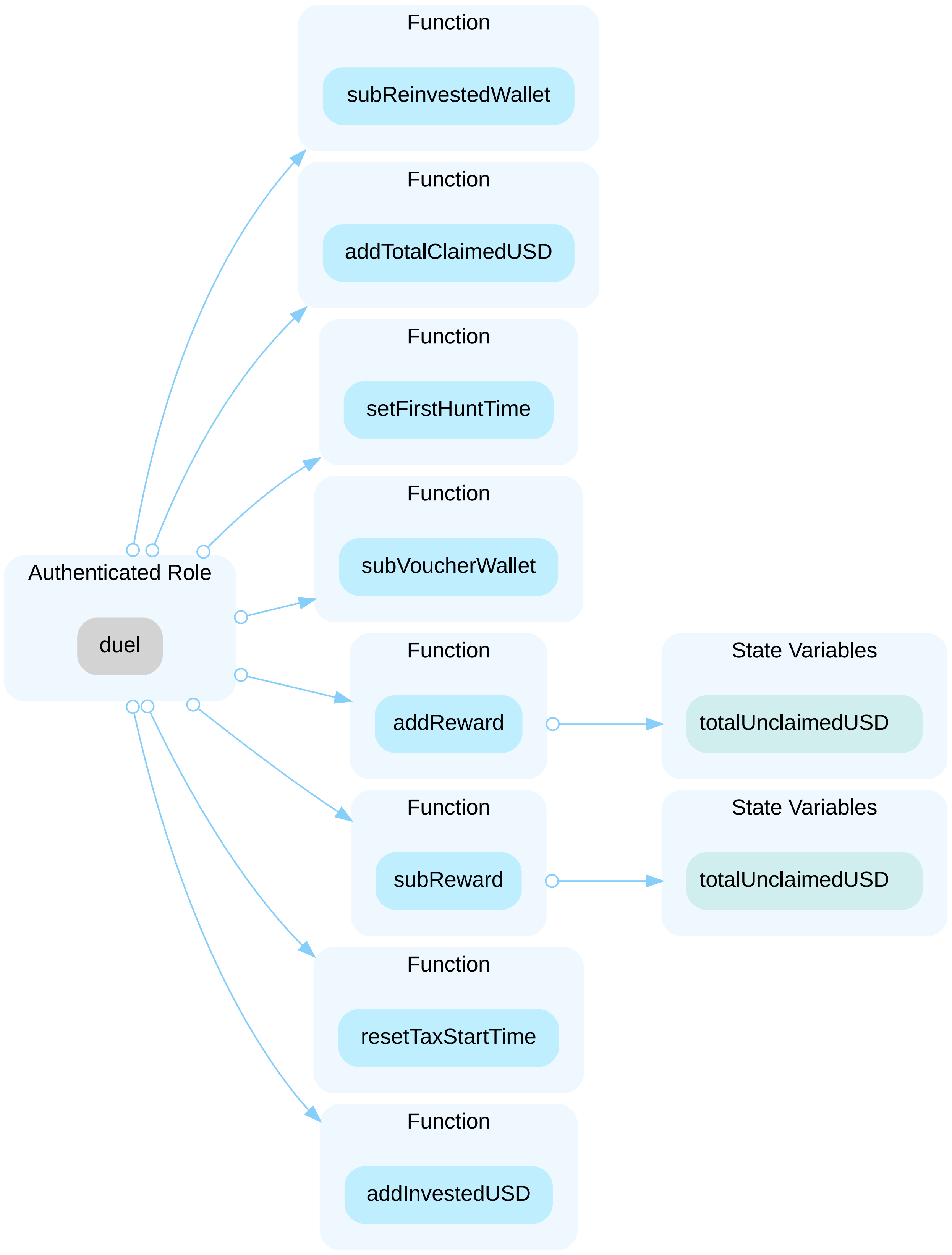
In the contract `RewardPool` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



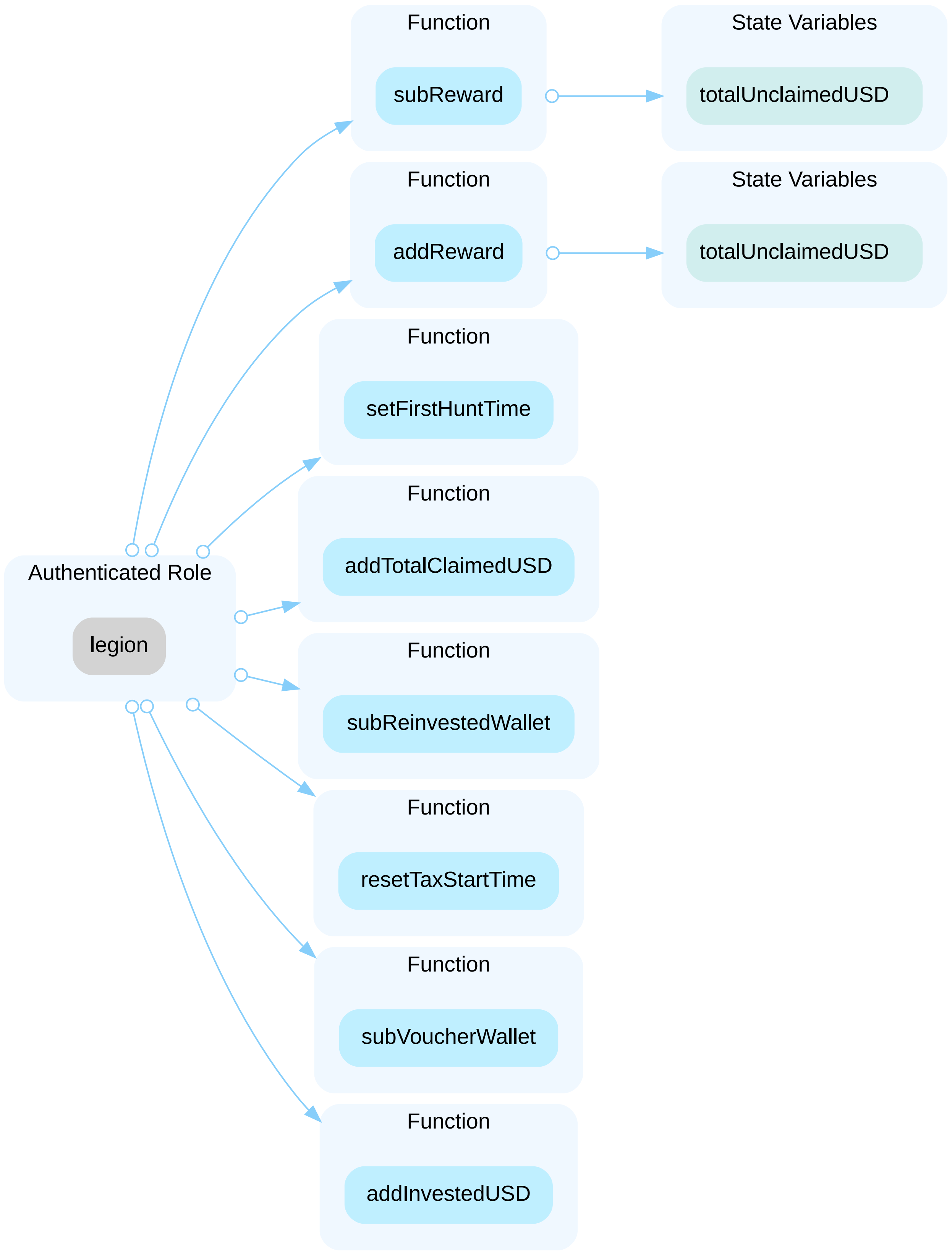
In the contract `RewardPool` the role `beast` has authority over the functions shown in the diagram below. Any compromise to the `beast` account may allow the hacker to take advantage of this authority.



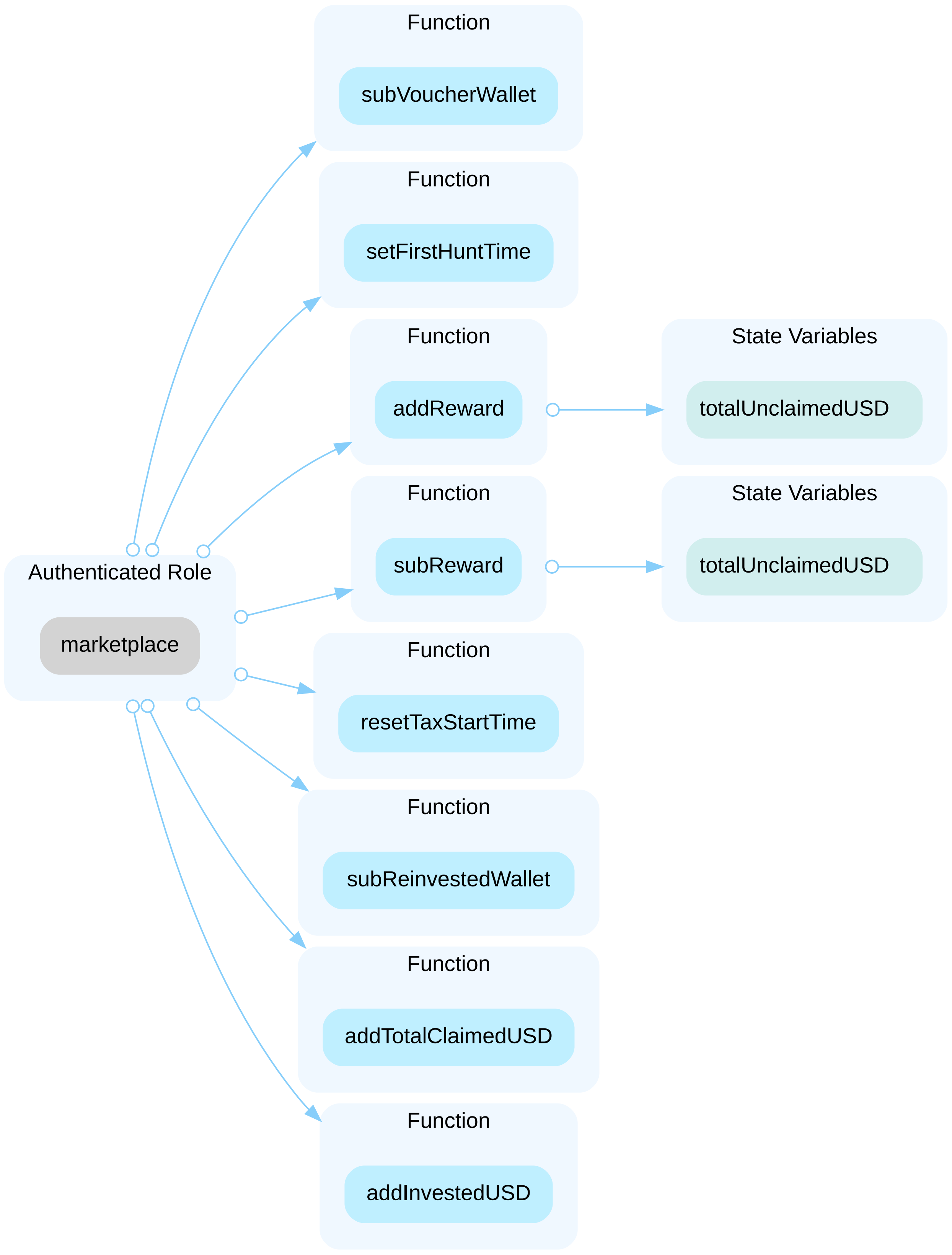
In the contract `RewardPool` the role `duel` has authority over the functions shown in the diagram below. Any compromise to the `duel` account may allow the hacker to take advantage of this authority.



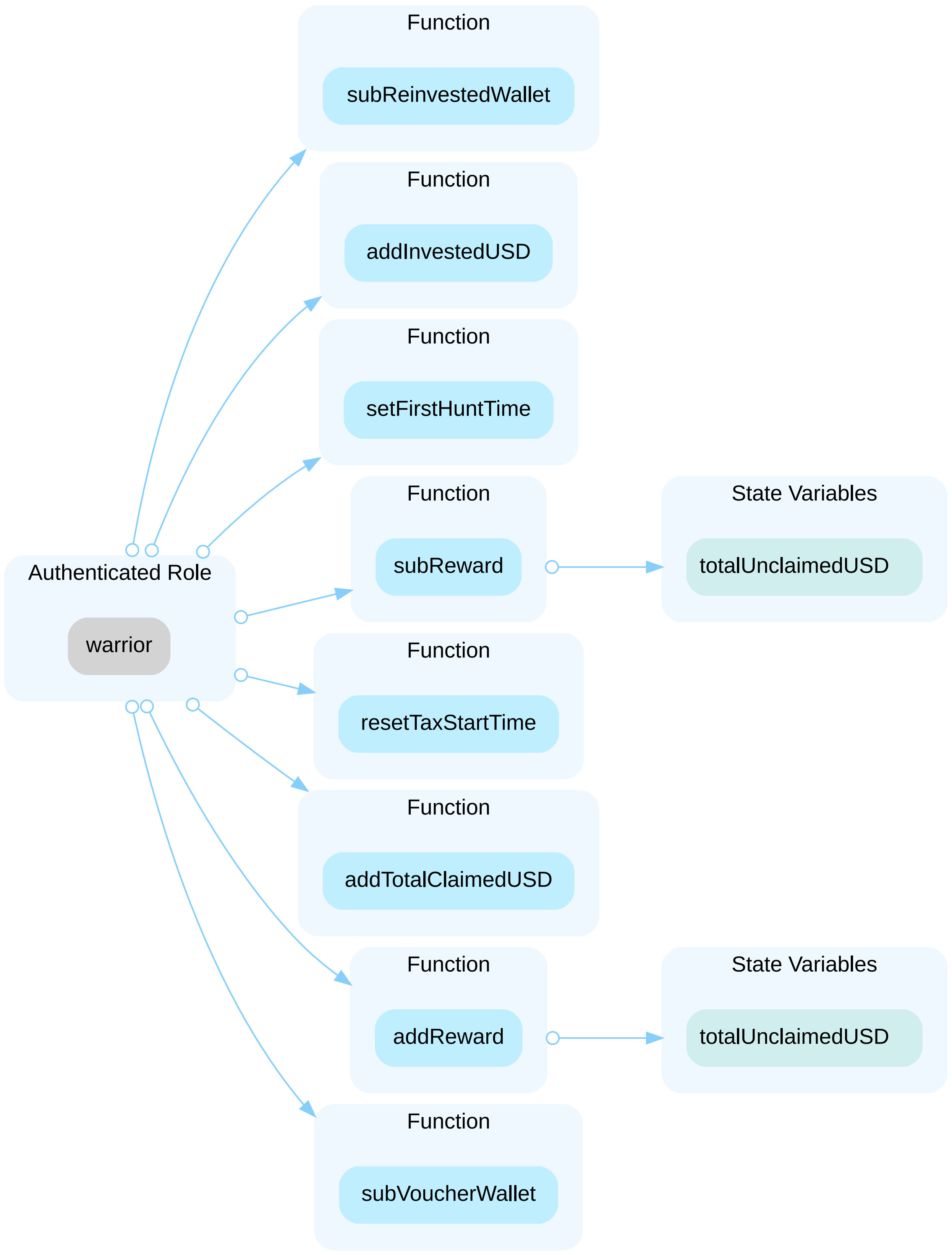
In the contract `RewardPool` the role `legion` has authority over the functions shown in the diagram below. Any compromise to the `legion` account may allow the hacker to take advantage of this authority.



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



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



Recommendation

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

Short Term:

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

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

Long Term:

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

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

Permanent:

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

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

Alleviation

[`Big Crypto Game`]: Issue acknowledged.

[`Certik`]: Privileged function `addVoucherWallet()` was replaced by new privileged function `setVoucherWallet()`. Code change was applied in the commit 1ffc38ca75d2745e0e33498fc89be7d1a4ea6f5c.

WNF-01 | CENTRALIZATION RISKS IN WARRIORNFT.SOL

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**contracts/WarriorNFT.sol: 199, 211**	● **Acknowledged**

Description

In the contract `WarriorNFT` , the role `owner` has authority over the following functions:

- function whitelist()
- function setPublicWhitelist()

Any compromise to the `owner` account may allow a 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., multi-signature wallets.

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

Short Term:

Timelock and Multi sign ($2/3$, $3/5$) 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.

Alleviation

[`Big Crypto Game`]: Issue acknowledged.

DSB-07 | INCORRECT `betAmounts` MAXIMUM LIMIT

Category	Severity	Location	Status
Logical Issue	● Informational	contracts/DuelSystem.sol: 32, 111	● Resolved

Description

File: DuelSystem.sol

The biggest index number of the state variable `betAmounts` is `8`. If the given index number `9` is greater than `8` it can pass the check of statement at #L111 because the maximum index limit is `10`. In this case, an index out of range error occurs which means that the maximum index limit should be `9`.

```
32      uint256[9] public betAmounts = [40, 60, 70, 100, 140, 190, 250, 400, 500];
```

```
110     function updateBetAmount(uint8 index, uint256 amount) external onlyOwner {
111          require(index<10, "Index is out of range");
112          betAmounts[index] = amount;
113      }
```

Recommendation

We recommend refactoring the codes to correct the maximum index limit.

Alleviation

[`Certik`]: The team resolved this issue in the commit 1ffc38ca75d2745e0e33498fc89be7d1a4ea6f5c.

DSB-08 | INCONSISTENT DOCUMENT AND CODEBASE

Category	Severity	Location	Status
Logical Issue	● Informational	contracts/DuelSystem.sol: 79, 97~102, 126, 131	● Resolved

Description

1. The result of a `Duel` will be three different outcomes: 1. the creator is the winner; 2. the joiner is the winner; 3. the duel is a draw. The document says that if the `Duel` is a draw, both players will get 100% of the bet amount back to their `Unclaimed Wallet`. The code to refund the bet amount doesn't exist.
2. The document says that once another player accepts your `Duel` invitation, then you will not be able to change your price prediction anymore. However, according to the code logic in the linked statement, the `Duel` creator still has a chance to change his/her price prediction if the invitation has not expired and another player already has joined the `Duel`. The logic between the document and the codebase is inconsistent.

Recommendation

We recommend maintaining consistency between the document and the codebase.

Alleviation

[`Certik`]: The team resolved this issue in the commit 1ffc38ca75d2745e0e33498fc89be7d1a4ea6f5c.

WNF-02 | POTENTIAL INDEX OUT-OF-RANGE ERROR

Category	Severity	Location	Status
Logical Issue	● Informational	contracts/WarriorNFT.sol: 23, 27, 63, 92, 199~204	● Resolved

Description

File: WarriorNFT.sol

The whitelist level is used to index the max minted amount stored in the state variable `maxAmountForWhitelist` . So the range of the index of the array state variable `maxAmountForWhitelist` should be the same as the whitelist level's range. They all range from `0` to `9` . If the given level's value is higher than `9` , an index out-of-range error will occur at #L63 and #L92 statements.

```
199     function whitelist(address[] memory wallets, uint8 level) external onlyOwner
{
200         for (uint256 i = 0; i < wallets.length; i++) {
201             whitelistLevel[wallets[i]] = level;
202             whitelisted[wallets[i]] = true;
203         }
204     }
```

Recommendation

We recommend adding a check for the whitelist level to avoid the index out-of-range error.

Alleviation

[`CertiK`]: The team resolved this issue in the commit 1ffc38ca75d2745e0e33498fc89be7d1a4ea6f5c.

WNF-03 | MISSING REMOVING WHITELIST FEATURE

Category	Severity	Location	Status
Logical Issue	● Informational	contracts/WarriorNFT.sol: 202	● Resolved

Description

File: WarriorNFT.sol

Adding users to and removing users from the whitelist is required to maintain the integrity of the whitelist functionality. We only find the code logic to add users to the whitelist in the code base. If these users are never needed, they need to be removed from the whitelist.

```
    function whitelist(address[] memory wallets, uint8 level) external onlyOwner
{
        for (uint256 i = 0; i < wallets.length; i++) {
            whitelistLevel[wallets[i]] = level;
            whitelisted[wallets[i]] = true;
        }
    }
```

Recommendation

We recommend adding codes to remove users from the whitelist.

Alleviation

[`CertiK`]: The team resolved this issue in the commit 1ffc38ca75d2745e0e33498fc89be7d1a4ea6f5c.

OPTIMIZATIONS | BIG CRYPTO GAME - DUELS

ID	Title	Category	Severity	Status
DSC-02	State Variable Should Be Declared Constant	Gas Optimization	Optimization	● Resolved
DSC-03	Function Should Be Declared External	Gas Optimization	Optimization	● Resolved

DSC-02 | STATE VARIABLE SHOULD BE DECLARED CONSTANT

Category	Severity	Location	Status
Gas Optimization	● Optimization	DuelSystem.sol (d673ead): 33, 34	● Resolved

Description

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

```
33      uint256 invitePeriod = 10 minutes; // this should be changed when deploying
```

- `invitePeriod` should be declared `constant` .

```
34      uint256 duelPeriod = 20 minutes; // this should be changed when deploying
```

- `duelPeriod` should be declared `constant` .

Recommendation

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

Alleviation

[`Certik`]: The team resolved this issue in the commit 1ffc38ca75d2745e0e33498fc89be7d1a4ea6f5c.

DSC-03 | FUNCTION SHOULD BE DECLARED EXTERNAL

Category	Severity	Location	Status
Gas Optimization	● Optimization	DuelSystem.sol (d673ead): 152	● Resolved

Description

The functions which are never called internally within the contract should have external visibility for gas optimization.

```
152     function getAllDuels() public view returns (Duel[] memory) {
```

Recommendation

We advise to change the visibility of the aforementioned functions to `external`.

Alleviation

[`Certik`]: The team resolved this issue in the commit 1ffc38ca75d2745e0e33498fc89be7d1a4ea6f5c.

APPENDIX | BIG CRYPTO GAME - DUELS

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

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

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.