Hashbuzz

Security Assessment

CertiK Assessed on Nov 15th, 2024

CertiK Assessed on Nov 15th, 2024

Hashbuzz

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

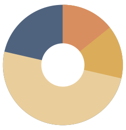
Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Hedera (HBAR)	Formal Verification, Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 11/15/2024	N/A

CODEBASE	COMMITS
smv201 View All in Codebase Page	base: d28bffc8f7e2f1c8f47e0876869579dd9afe02e8 final: 4f41ac7e561c5c688fd44c24cc9e81633dac3585 View All in Codebase Page

Vulnerability Summary



14	12	0	0	2	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

🟥	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.
🟧	2	Major	1 Resolved, 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.
🟨	2	Medium	2 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	7	Minor	6 Resolved, 1 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.
🟦	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 | HASHBUZZ

Summary

- Executive Summary
- Vulnerability Summary
- Codebase
- Audit Scope
- Approach & Methods

Review Notes

- Overview
- External Dependencies
  - Addresses
- Privileged Functions

Findings

- CLH-05 : Potential Dos Vulnerability Due to Lack of Assignment for `tokenBalances` of NFT Token Type
- HMB-02 : Centralization Related Risks
- CLH-04 : Inconsistent Update for `tokenCampaignBalances` of NFT
- CLH-06 : Potential Overwrite of Token Campaign Balances
- CLH-01 : Potential Timing Issue in Reward Adjustment Functions
- CLH-02 : Clarification on `expiryFungibleCampaign()` Function
- CLH-07 : Incorrect Token Type Handling in `expiryFungibleCampaign` Function
- HMB-01 : Lack of Sanity Checks
- HMB-03 : Inconsistent Data Type Usage in Storage and Function Parameters
- HVB-01 : Locked Blockchain Native Tokens
- UHM-01 : Invalid Use of Access Control Modifier
- CLH-03 : Clarification on Adjusting NFT Campaign Rewards
- HSH-01 : Hidden Role In the Contract May Arise Centralization Concerns
- THM-01 : Meaningless Parameter

Appendix

Disclaimer

CODEBASE | HASHBUZZ

Repository

smv201

Commit

base: d28bffc8f7e2f1c8f47e0876869579dd9afe02e8

final: 4f41ac7e561c5c688fd44c24cc9e81633dac3585

AUDIT SCOPE | HASHBUZZ

5 files audited ● 3 files with Acknowledged findings ● 2 files with Resolved findings

ID	Repo	File	SHA256 Checksum
● CLH	hashbuzz/smv201	contracts/HashbuzzModules/CampaignLifecycle.sol	02e4620eea553652cb225a7867efd7c8dd1a712d64e792924664eb02bb903a0e
● THM	hashbuzz/smv201	contracts/HashbuzzModules/Transactions.sol	8be1417fa52a2072b0978722956b54c6c700489d9472fd44ba81864226aa4bd8
● UHM	hashbuzz/smv201	contracts/HashbuzzModules/Utils.sol	8b2fd0c7b1116cd6873c0cd25f80fa738a167d7652ac20311ab7ada91cac9375
● HSH	hashbuzz/smv201	contracts/HashbuzzModules/HashbuzzStates.sol	aeeae45e3bb4727663dc0e73da7bfc5571946994fa7011c9fb5af92213bf19bf
● HVB	hashbuzz/smv201	contracts/HashbuzzV201.sol	e22704035f79de2ac6b9f1f2430816a4dce2ad7835c38b22420a0727aa1d935b

APPROACH & METHODS | HASHBUZZ

This report has been prepared for Hashbuzz to discover issues and vulnerabilities in the source code of the Hashbuzz project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Formal Verification, 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.

REVIEW NOTES | HASHBUZZ

Overview

The **Hashbuzz** project coordinates a series of smart contracts aimed at supporting campaign functionality. The files currently under review include:

- HashbuzzModules/CampaignLifecycle.sol
- HashbuzzModules/HashbuzzStates.sol
- HashbuzzModules/Transactions.sol
- HashbuzzModules/Utils.sol
- HashbuzzV201.sol

External Dependencies

In **Hashbuzz**, the module inherits or uses a few of the depending injection contracts or addresses to fulfill the need of its business logic. 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.

Addresses

The following addresses interact at some point with specified contracts, making them an external dependency. All of the following values are initialized either at deployment time or by specific functions in smart contracts.

- `owner` , `campaigners` .

We assume these contracts or addresses are valid and non-vulnerable actors and implementing proper logic to collaborate with the current project.

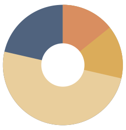
Privileged Functions

In the **Hashbuzz** project, the privileged roles are adopted to ensure the dynamic runtime updates of the project, which are specified in the **HMB-02: Centralization Related Risks** finding.

The advantage of those privileged roles in the codebase is that the client reserves the ability to adjust the protocol according to the runtime required to best serve the community. It is also worth noting the potential drawbacks of these functions, which should be clearly stated through the client's action/plan. Additionally, if the private keys of the privileged accounts are compromised, it could lead to devastating consequences for the project.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the `Timelock` contract.

FINDINGS | HASHBUZZ



14	0	2	2	7	3
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Hashbuzz. Through this audit, we have uncovered 14 issues ranging from different severity levels. Utilizing the techniques of Formal Verification, Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
CLH-05	Potential Dos Vulnerability Due To Lack Of Assignment For `tokenBalances` Of NFT Token Type	Coding Issue	Major	● Resolved
HMB-02	**Centralization Related Risks**	**Centralization**	**Major**	● **Acknowledged**
CLH-04	Inconsistent Update For `tokenCampaignBalances` Of NFT	Logical Issue	Medium	● Resolved
CLH-06	Potential Overwrite Of Token Campaign Balances	Logical Issue	Medium	● Resolved
CLH-01	Potential Timing Issue In Reward Adjustment Functions	Logical Issue	Minor	● Acknowledged
CLH-02	Clarification On `expiryFungibleCampaign()` Function	Design Issue	Minor	● Resolved
CLH-07	Incorrect Token Type Handling In `expiryFungibleCampaign` Function	Volatile Code	Minor	● Resolved
HMB-01	Lack Of Sanity Checks	Inconsistency, Logical Issue	Minor	● Resolved
HMB-03	Inconsistent Data Type Usage In Storage And Function Parameters	Inconsistency	Minor	● Resolved
HVB-01	Locked Blockchain Native Tokens	Volatile Code	Minor	● Resolved

ID	Title	Category	Severity	Status
UHM-01	Invalid Use Of Access Control Modifier	Logical Issue	Minor	● Resolved
CLH-03	Clarification On Adjusting NFT Campaign Rewards	Design Issue, Inconsistency	Informational	● Resolved
HSH-01	Hidden Role In The Contract May Arise Centralization Concerns	Coding Issue	Informational	● Resolved
THM-01	Meaningless Parameter	Coding Issue	Informational	● Resolved

CLH-05 | POTENTIAL DOS VULNERABILITY DUE TO LACK OF ASSIGNMENT FOR `tokenBalances` OF NFT TOKEN TYPE

Category	Severity	Location	Status
Coding Issue	● Major	contracts/HashbuzzModules/CampaignLifecycle.sol (11/7-d28bffc): 109	● Resolved

Description

The `addNFTCampaign()` function allows the contract owner to add a campaign's balance to a specified `campaignAddress`. This function increases the balance of `tokenCampaignBalances` at the given `campaignAddress` by the `tokenAmount`. The `tokenAmount` must be within the range of 1 to `tokenBalances[campaigner][tokenId][NFT]`. However, within the audit scope, there is no mechanism to set a value for `tokenBalances[campaigner][tokenId][NFT]`. Consequently, the `addNFTCampaign()` function will consistently fail to execute. It is worth noting that functions exist to set `tokenBalances` for `FUNGIBLE` tokens.

```
    require(
        tokenBalances[campaigner][tokenId][NFT] >= uint64(tokenAmount),
        ERR_INSUFFICIENT_BALANCE
    );
```

Proof of Concept

The following PoC show the issue mentioned.

```
   // SPDX-License-Identifier: UNLICENSED
pragma solidity 0.8.20;

import {Test, console2} from "forge-std/Test.sol";
import "../src/contracts/HashbuzzV201.sol";

contract HashbuzzTest is Test {

    address private Verifier;
    address private Bob = address(0x456);
    address private Alice = address(0x123);
    address private tokenId = address(0xadb);
    string campaignAddress = "ca";

    HashbuzzV201 hb;

    function setUp() public {
        hb = new HashbuzzV201();
    }

    function testProcess() public {
        hb.addCampaigner(Bob);
        hb.associateToken(tokenId, 2, true);
        hb.addNFTCampaign(tokenId, campaignAddress, Bob, 10000);
    }

}
```

Output text:

```
forge test -vv
[⠘] Compiling...
[⠰] Compiling 1 files with 0.8.20
[⠒] Solc 0.8.20 finished in 4.65s
Compiler run successful!

Running 1 test for test/Hashbuzz.t.sol:HashbuzzTest
[FAIL. Reason: E008] testProcess() (gas: 114057)
Test result: FAILED. 0 passed; 1 failed; 0 skipped; finished in 2.03ms
Ran 1 test suites: 0 tests passed, 1 failed, 0 skipped (1 total tests)

Failing tests:
Encountered 1 failing test in test/Hashbuzz.t.sol:HashbuzzTest
[FAIL. Reason: E008] testProcess() (gas: 114057)

Encountered a total of 1 failing tests, 0 tests succeeded
```

Recommendation

Recommend implementing logic to assign a value to the `tokenBalances` for `NFT` type.

Alleviation

[Hashbuzz, 11/13/2024]:

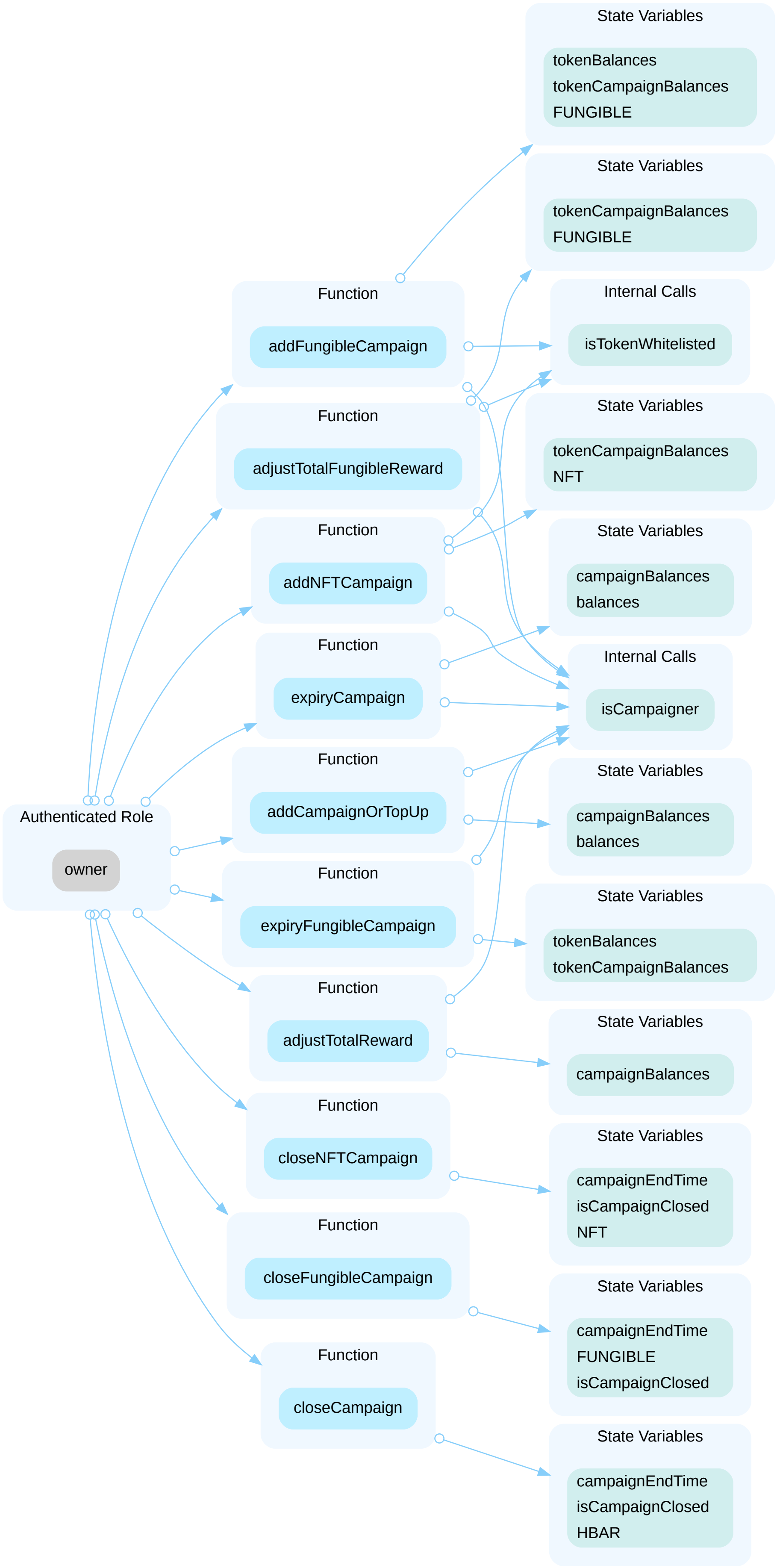
The team resolved this issue by excluding all NFT-related methods from this audit scope in the updated version 7094693fcd6885cff783a44effa355d68df60e3c.

HMB-02 | CENTRALIZATION RELATED RISKS

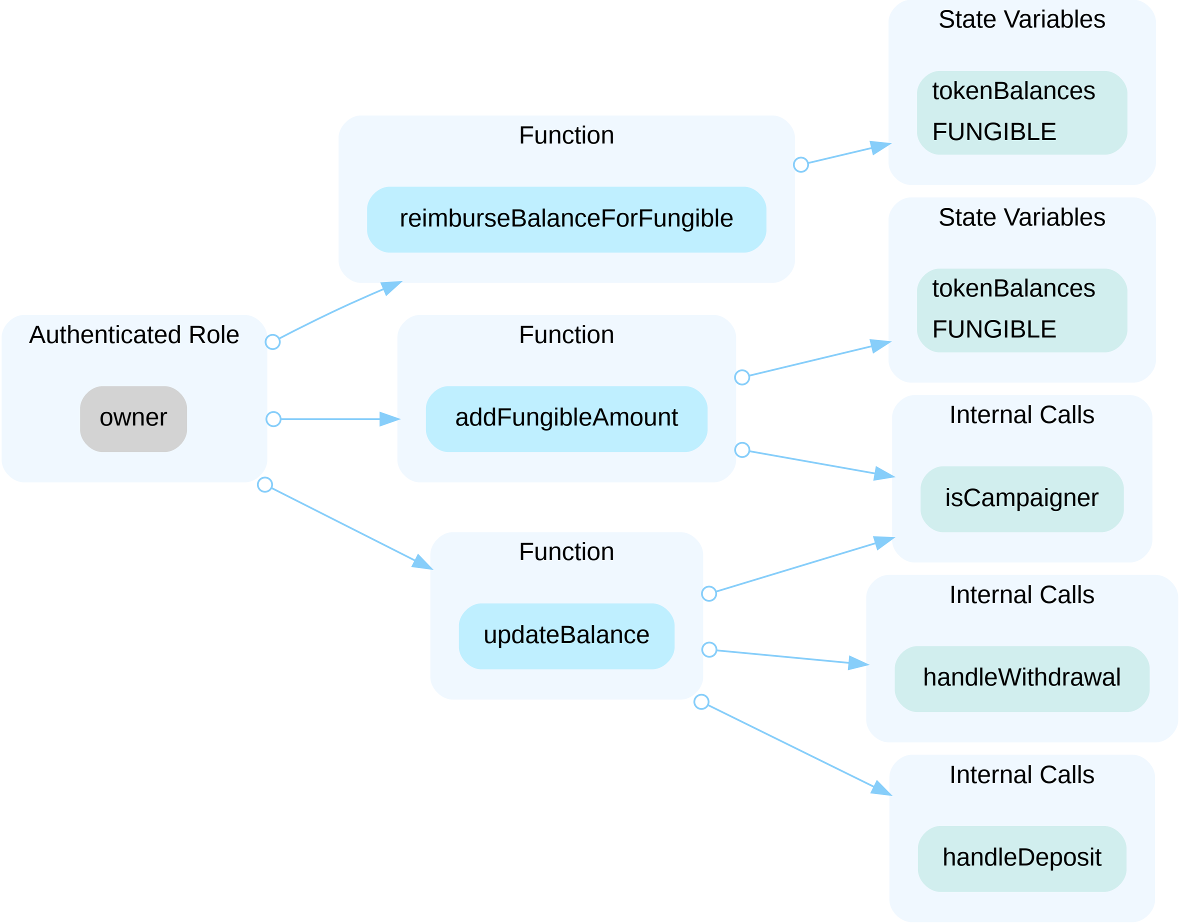
Category	Severity	Location	Status
Centralization	● **Major**	**contracts/HashbuzzModules/CampaignLifecycle.sol (11/7-d28bffc): 18, 47, 94, 127, 153, 180, 208, 242, 290, 327; contracts/HashbuzzModules/Transactions.sol (11/7-d28bffc): 33, 58, 81; contracts/HashbuzzModules/Utils.sol (11/7-d28bffc): 19, 85, 99, 118, 132, 146**	● **Acknowledged**

Description

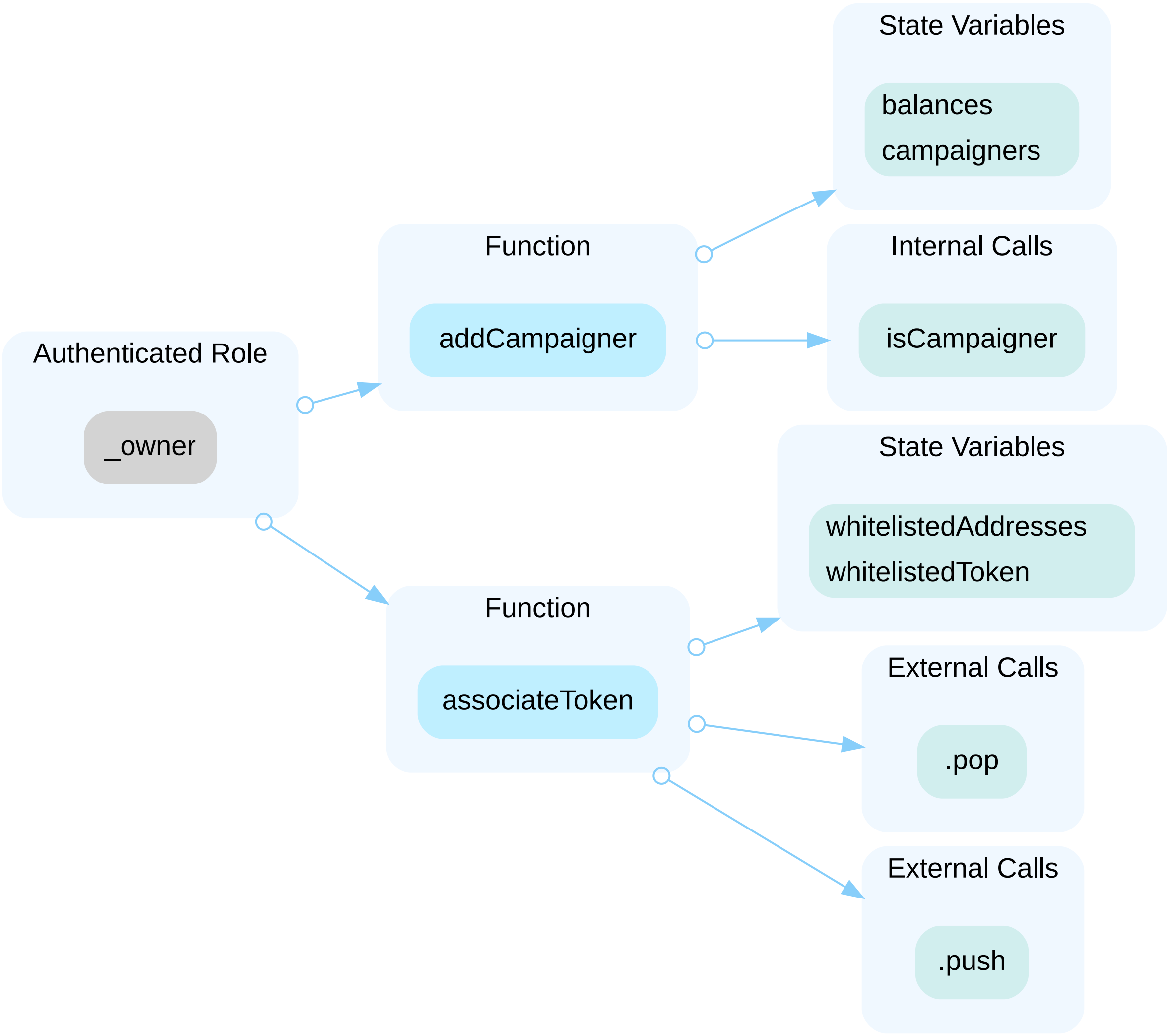
In the contract `Lifecycle` , the role `owner` has authority over the functions shown in the diagram below. Any compromise to the `owner` account may allow the hacker to take advantage of this authority and add fungible token campaigns for campaigners, adjust total fungible reward balances, close NFT campaigns, expire campaigns and update campaigner balances, close fungible campaigns, expire fungible token campaigns, add NFT campaigns and update campaign balances, add or top up campaign balances, adjust total rewards for a campaigner's campaign, and close the specified campaigns.


State Variables
tokenBalances
tokenCampaignBalances
FUNGIBLE
State Variables
tokenCampaignBalances
FUNGIBLE
Function
addFungibleCampaign
Internal Calls
isTokenWhitelisted
Function
adjustTotalFungibleReward
State Variables
tokenCampaignBalances
NFT
Function
addNFTCampaign
State Variables
campaignBalances
balances
Function
expiryCampaign
Internal Calls
isCampaigner
Function
addCampaignOrTopUp
State Variables
campaignBalances
balances
Authenticated Role
owner
Function
expiryFungibleCampaign
State Variables
tokenBalances
tokenCampaignBalances
Function
adjustTotalReward
State Variables
campaignBalances
Function
closeNFTCampaign
State Variables
campaignEndTime
isCampaignClosed
NFT
Function
closeFungibleCampaign
State Variables
campaignEndTime
FUNGIBLE
isCampaignClosed
Function
closeCampaign
State Variables
campaignEndTime
isCampaignClosed
HBAR

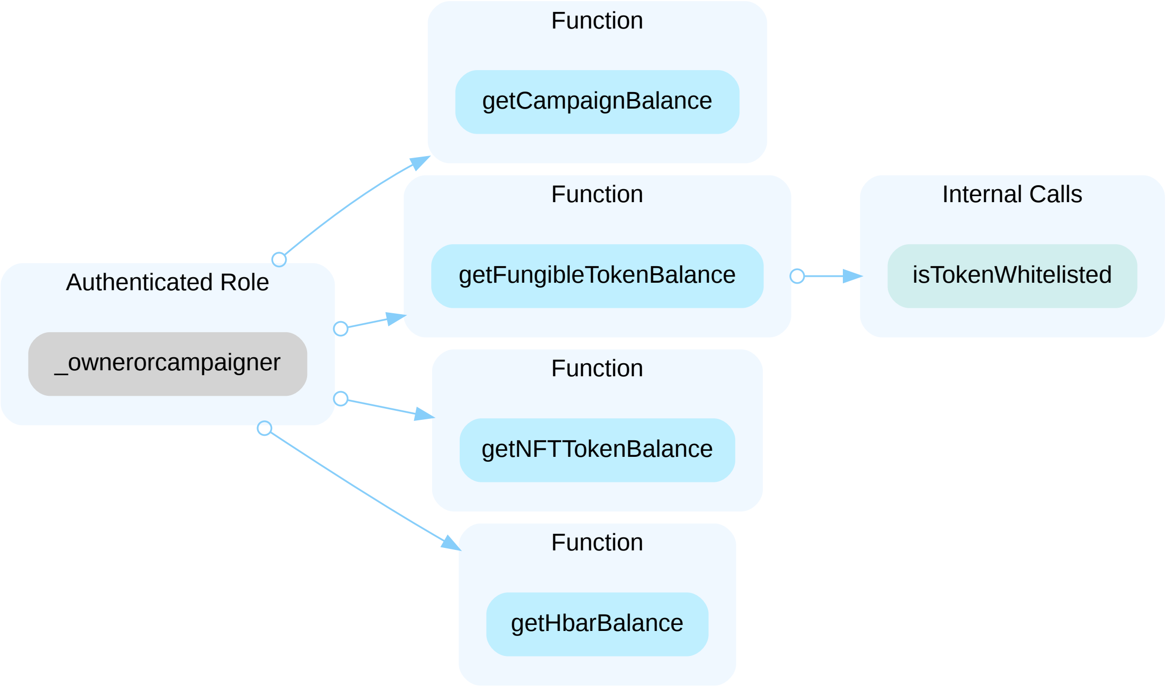
In the contract `Transactions` , the role `owner` has authority over the functions shown in the diagram below. Any compromise to the `owner` account may allow the hacker to take advantage of this authority and reimburse fungible token balance for a campaigner, add fungible token amount to a campaigner balance, and update the balance for a campaigner.



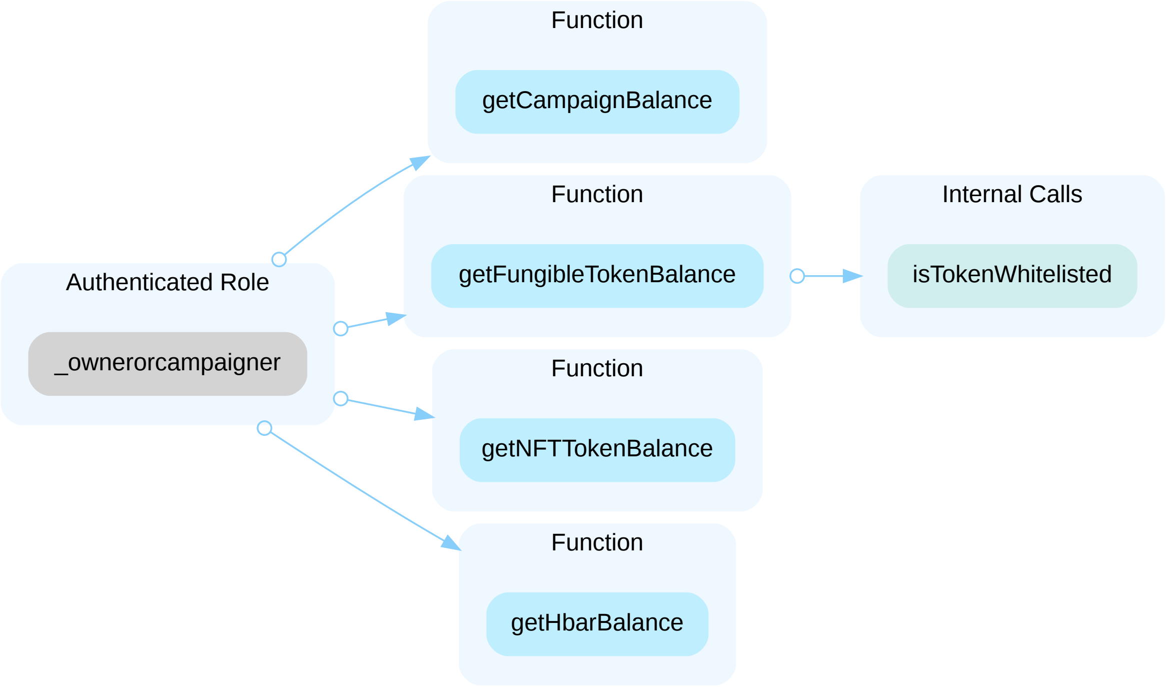
In the contract `Utils` , the role `owner` has authority over the functions shown in the diagram below. Any compromise to the `owner` account may allow the hacker to take advantage of this authority and add a new campaigner, associate or disassociate a token with the whitelist.



In the contract `Utils`, 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 and retrieve campaign balance, retrieve fungible token balance, retrieve NFT token balance for campaigner, and retrieve campaigner's Hbar balance.



In the contract `Utils` , the role `Campaigner` has authority over the functions shown in the diagram below. Any compromise to the `_ownerorcampaigner` account may allow the hacker to take advantage of this authority and retrieve campaign balance, retrieve fungible token balance, retrieve NFT token balance for campaigner, and retrieve campaigner's Hbar balance.



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

[Hashbuzz, 11/13/2024]:

Our team has reviewed the concerns and is planning to shift towards decentralisation in the upcoming development phase. This will involve the introduction of smart-node architecture, multi-signature wallets, and a DAO. Currently, our objective is to deploy the existing smart contract on the mainnet promptly to gain traction, even though access for new campaigners will be restricted initially. We acknowledge the need for a more permanent resolution in this smart contract and anticipate removing the admin role privilege as we transition to the new architecture.

[CertiK, 11/13/2024]:

The team has eliminated privileged functions related to NFTs. Additionally, the previous privileged function `reimburseBalanceForFungible()` has been renamed to `reimburseCampaigner()`.

It is suggested to implement the aforementioned methods to avoid centralized failure. Also, it strongly encourages the project team to periodically revisit the private key security management of all addresses related to centralized roles.

CLH-04 INCONSISTENT UPDATE FOR `tokenCampaignBalances` OF NFT

Category	Severity	Location	Status
Logical Issue	● Medium	contracts/HashbuzzModules/CampaignLifecycle.sol (11/7-d28bffc): 112	● Resolved

Description

The `addNFTCampaign()` function, unlike the `addFungibleCampaign()` function, increases the `tokenAmount` balance in `tokenCampaignBalances` without deducting it from `tokenBalances`. This appears to be intentional, as indicated by the emission of the `NewCampaignIsAdded` event.

```
...
require(
    tokenBalances[campaigner][tokenId][NFT] >= uint64(tokenAmount),
    ERR_INSUFFICIENT_BALANCE
);
tokenCampaignBalances[campaignAddress][tokenId][NFT] = uint64(
    tokenAmount
);

...
emit NewCampaignIsAdded(campaignAddress, uint64(tokenAmount), NFT);
...
```

However, when the owner expires a campaign with NFT tokens, the balance from `tokenCampaignBalances` is transferred back to `tokenBalances` directly. Consequently, `tokenBalances` receives an entirely new balance.

```
tokenBalances[campaigner][tokenId][tokenType] += tokenCampaignBalances[
        campaignAddress
    ][tokenId][tokenType];
tokenCampaignBalances[campaignAddress][tokenId][tokenType] = 0;
```

Recommendation

Recommend the client to verify the related logic thoroughly and make code adjustments if necessary.

Alleviation

[Hashbuzz, 11/13/2024]:

The team resolved this issue by excluding all NFT-related methods from this audit scope in the updated version 7094693fcd6885cff783a44effa355d68df60e3c.

CLH-06 | POTENTIAL OVERWRITE OF TOKEN CAMPAIGN BALANCES

Category	Severity	Location	Status
Logical Issue	● Medium	contracts/HashbuzzModules/CampaignLifecycle.sol (11/7-d28bffc): 112~114	● Resolved

Description

The `addNFTCampaign()` function allows adding a new campaign for NFT tokens. However, the assignment of `tokenCampaignBalances` with `tokenAmount` directly overwrites any existing value, as shown in the code snippet below:

```
tokenCampaignBalances[campaignAddress][tokenId][NFT] = uint64(tokenAmount);
```

This can lead to issues if the function is called multiple times for the same `campaignAddress` and `tokenId`, as it will overwrite the previous balance instead of accumulating it.

Recommendation

To ensure that the token balances are accumulated rather than overwritten, modify the assignment operation to use the `+=` operator.

Alleviation

[Hashbuzz, 11/13/2024]:

The team resolved this issue by excluding all NFT-related methods from this audit scope in the updated version 7094693fcd6885cff783a44effa355d68df60e3c.

CLH-01 | POTENTIAL TIMING ISSUE IN REWARD ADJUSTMENT FUNCTIONS

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/HashbuzzModules/CampaignLifecycle.sol (11/7-d28bffc): 208, 242	● Acknowledged

Description

The `adjustTotalReward()` and `adjustTotalFungibleReward()` functions in the `Lifecycle` contract allow the owner to distribute tokens after a campaign closes. However, there is no validation to ensure these functions are executed before the `campaignEndTime`. This could lead to potential concurrency issues with the `expiryCampaign()` and `expiryFungibleCampaign()` functions, which can be invoked after the `campaignEndTime`. Without this validation, the owner could adjust rewards concurrently with campaign expiry operations, potentially resulting in inconsistencies in token distribution.

Additionally, all `FUNGIBLE` tokens share the same close flag. `isCampaignClosed` and `campaignEndTime` are not linked to individual token.

Recommendation

We would like to confirm if the current implementation aligns with the intended design and recommend add the check for `campaignEndTime` in the adjust functions.

Alleviation

[CertiK, 11/13/2024]:
We initially believed the process followed the order of adding a campaign, closing it, adjusting the reward, and then expiring the campaign. However, the current setup allows the owner to adjust the reward even after the `campaignEndTime`, as long as the campaign hasn't been expired yet. If the adjustment isn't done before the campaign expires, it can't be performed. We'd like to verify if this behavior matches the intended design. Is it necessary to require campaign expires before `campaignEndTime`?

[Hashbuzz, 11/14/2024]:
Yes, this conforms with the intended design process we have setup.

[CertiK, 11/14/2024]:
Since there's no timestamp check in the function, we would like to remind the team to carefully execute related functions according to the intended design process.

CLH-02 | CLARIFICATION ON `expiryFungibleCampaign()` FUNCTION

Category	Severity	Location	Status
Design Issue	● Minor	contracts/HashbuzzModules/CampaignLifecycle.sol (11/7-d28bffc): 290	● Resolved

Description

The `expiryFungibleCampaign()` function, as per its comments and logic, is designed to handle both `FUNGIBLE` and `NFT` types. However, its name suggests it only handles the `FUNGIBLE` type, which is misleading.

Recommendation

We would like to clarify the usage of the `expiryFungibleCampaign()` function with the client.

Alleviation

[Hashbuzz, 11/13/2024]:
The team resolved this issue by excluding all NFT-related methods from this audit scope in the updated version 7094693fcd6885cff783a44effa355d68df60e3c.

CLH-07 | INCORRECT TOKEN TYPE HANDLING IN `expiryFungibleCampaign` FUNCTION

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/HashbuzzModules/CampaignLifecycle.sol (11/7-d28bffc): 290	● Resolved

Description

The `expiryFungibleCampaign()` function is designed to handle the expiry of campaigns with `Fungible` and `NFT` tokens. However, it lacks a proper check for the input `tokenType`. If the `tokenType` provided is `HBAR`, the function may proceed with incorrect logic, potentially emitting an incorrect event and causing unintended side effects.

Recommendation

Recommend adding a validation check at the beginning of the `expiryFungibleCampaign()` function to ensure that the `tokenType` is the correct value.

Alleviation

[Hashbuzz, 11/13/2024]:

The team resolved this issue by removing the input `tokenType` in the updated version 7094693fcd6885cff783a44effa355d68df60e3c.

HMB-01 | LACK OF SANITY CHECKS

Category	Severity	Location	Status
Inconsistency, Logical Issue	● Minor	contracts/HashbuzzModules/CampaignLifecycle.sol (11/7-d28bffc): 19, 21, 291; contracts/HashbuzzModules/Transactions.sol (11/7-d28bffc): 35, 60, 81, 82~84; contracts/HashbuzzModules/Utils.sol (11/7-d28bffc): 100, 119, 133~134, 147	● Resolved

Description

The linked parameters are missing necessary validation checks. For example:

1. `campaignAddress` must not be empty.
2. `amount` should be greater than zero.
3. `tokenId` needs to be whitelisted.
4. `campaigner` must exist.

Recommendation

Recommend incorporating appropriate validations for the input parameters.

Alleviation

[Hashbuzz, 11/14/2024]:

The team resolved this issue by using `uint256` to store the token balances in the updated version 78f65c8ba53d6640e1b12e4747eb036f21b47441.

[CertiK, 11/14/2024]:

Some parameters still lack the necessary checks. Additionally, there are several duplicate checks(in `addCampaignOrTopUp()` , `addFungibleCampaign()`) added in the commit 78f65c8ba53d6640e1b12e4747eb036f21b47441.

[Hashbuzz, 11/15/2024]:

The team resolved this issue by heeding the advice in the updated version 4f41ac7e561c5c688fd44c24cc9e81633dac3585.

HMB-03 INCONSISTENT DATA TYPE USAGE IN STORAGE AND FUNCTION PARAMETERS

Category	Severity	Location	Status
Inconsistency	● Minor	contracts/HashbuzzModules/CampaignLifecycle.sol (11/7-d28bffc): 51~52, 84, 98~99, 119; contracts/HashbuzzModules/Transactions.sol (11/7-d28bffc): 61~62, 65, 67, 71, 84, 86, 96	● Resolved

Description

The contract under review exhibits an inconsistency in data types between its storage variables and function parameters. Specifically, the contract declares `tokenBalances` and `tokenCampaignBalances` mappings with a data type of `uint64` to store token amounts. However, several functions in the contract utilize `int64` as their input and output data types for handling these amounts. This discrepancy between `uint64` and `int64` could lead to incorrect return values and unexpected behaviors due to type conversion issues. Below are the relevant code snippets:

```
...
mapping(address => mapping(address => mapping(uint256 => uint64)))
        public tokenBalances;
...
mapping(string => mapping(address => mapping(uint256 => uint64)))
        public tokenCampaignBalances;
...
```

Recommendation

To prevent type conversion errors and ensure consistent data integrity across the contract, it is recommended to standardize the data type used throughout the contract.

Alleviation

[Hashbuzz, 11/13/2024]:
The team resolved this issue by heeding the advice in the updated version 7094693fcd6885cff783a44effa355d68df60e3c.

[CertiK, 11/13/2024]:
We would like to confirm with the client whether the uint64 range is sufficient for token balances, especially since tokens often have up to 18 decimal places.

[Hashbuzz, 11/14/2024]:
The team resolved this issue by using `uint256` to store the token balances in the updated version 78f65c8ba53d6640e1b12e4747eb036f21b47441.

HVB-01 | LOCKED BLOCKCHAIN NATIVE TOKENS

Category	Severity	Location	Status
Volatile Code	● Minor	contracts/HashbuzzV201.sol (11/7-d28bffc): 15~16	● Resolved

Description

In the `HashbuzzV201` contract, there are two payable functions that can be used to receive native tokens. However, the contract lacks a mechanism to withdraw native tokens. As a result, any native tokens sent to these contracts may become permanently trapped.

```
    receive() external payable {}
    fallback() external payable {}
```

Recommendation

Consider adding a withdraw/sweep function to contracts that are capable of receiving native tokens.

Alleviation

[Hashbuzz, 11/13/2024]:

The team resolved this issue by removing the payable functions in the updated version 7094693fcd6885cff783a44effa355d68df60e3c.

UHM-01 | INVALID USE OF ACCESS CONTROL MODIFIER

Category	Severity	Location	Status
Logical Issue	● Minor	contracts/HashbuzzModules/Utils.sol (11/7-d28bffc): 99~104, 118~121, 132~137, 146~149	● Resolved

Description

The functions marked as 'view' or 'pure' are unnecessarily restricted by the `onlyOwnerOrCampaigner` modifier. These functions are designed to be read-only, meaning they do not modify the state on the blockchain. However, they are restricted so that only a specific address can call them.

It's important to note that private state variables can be read off-chain, rendering the access restriction on these functions ineffective.

Recommendation

We recommend that access restrictions are not used on `view` or `pure` functions, as they do not improve security for read-only operations. Instead, these getter functions should be made public to allow transparency and follow best practice. If there is sensitive information that should not be disclosed, the way in which this data is managed and stored should be reconsidered, as restricting access in this way does not provide effective security.

Alleviation

[Hashbuzz, 11/15/2024]:

The team resolved this issue by heeding the advice in the updated version 4f41ac7e561c5c688fd44c24cc9e81633dac3585.

CLH-03 | CLARIFICATION ON ADJUSTING NFT CAMPAIGN REWARDS

Category	Severity	Location	Status
Design Issue, Inconsistency	● Informational	contracts/HashbuzzModules/CampaignLifecycle.sol (11/7-d28bffc): 281	● Resolved

Description

The `Lifecycle` contract handles various campaign types, including `HBAR`, fungible tokens, and NFTs, offering capabilities to add, close, adjust, and expire campaigns. However, it lacks functionality for adjusting NFT campaign rewards. While the contract includes `adjustTotalReward()` and `adjustTotalFungibleReward()` functions for modifying rewards in `HBAR` and fungible token campaigns, there is no corresponding function for NFT campaigns. This absence could hinder the ability to properly adjust rewards for NFT campaigns.

Recommendation

We would like to confirm if the current implementation aligns with the intended design and recommend implementing an `adjustTotalNFTReward()` function similar to the existing reward adjustment functions for `HBAR` and fungible tokens.

Alleviation

[Hashbuzz, 11/13/2024]:

The team resolved this issue by excluding all NFT-related methods from this audit scope in the updated version 7094693fcd6885cff783a44effa355d68df60e3c.

HSH-01 HIDDEN ROLE IN THE CONTRACT MAY ARISE CENTRALIZATION CONCERNS

Category	Severity	Location	Status
Coding Issue	● Informational	contracts/HashbuzzModules/HashbuzzStates.sol (11/7-d28bffc): 9, 15	● Resolved

Description

The contract performs access control check over the certain roles `owner` and `campaigners`. However, the roles are currently unavailable via `getter` function. This makes it hard for normal user to get transparent information of the contract and may arise potential confusion.

Recommendation

Recommend changing the visibility of the private role to clarify the transparency.

Alleviation

[Hashbuzz, 11/13/2024]:

The team resolved this issue by heeding the advice in the updated version 7094693fcd6885cff783a44effa355d68df60e3c.

THM-01 | MEANINGLESS PARAMETER

Category	Severity	Location	Status
Coding Issue	● Informational	contracts/HashbuzzModules/Transactions.sol (11/7-d28bffc): 85	● Resolved

Description

The `reimburseBalanceForFungible()` function accepts a parameter `tokenType` , intended to verify if the type is `FUNGIBLE` . However, the function's logic uses a hardcoded token type `FUNGIBLE` , ignoring the input `tokenType` parameter in its operations.

Recommendation

Recommend removing the meaningless parameter.

Alleviation

[Hashbuzz, 11/13/2024]:

The team resolved this issue by heeding the advice in the updated version 7094693fcd6885cff783a44effa355d68df60e3c.

APPENDIX | HASHBUZZ

Finding Categories

Categories	Description
Coding Issue	Coding Issue findings are about general code quality including, but not limited to, coding mistakes, compile errors, and performance issues.
Inconsistency	Inconsistency findings refer to different parts of code that are not consistent or code that does not behave according to its specification.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.
Design Issue	Design Issue findings indicate general issues at the design level beyond program logic that are not covered by other finding categories.

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.

Elevating Your Entire Web3 Journey

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.