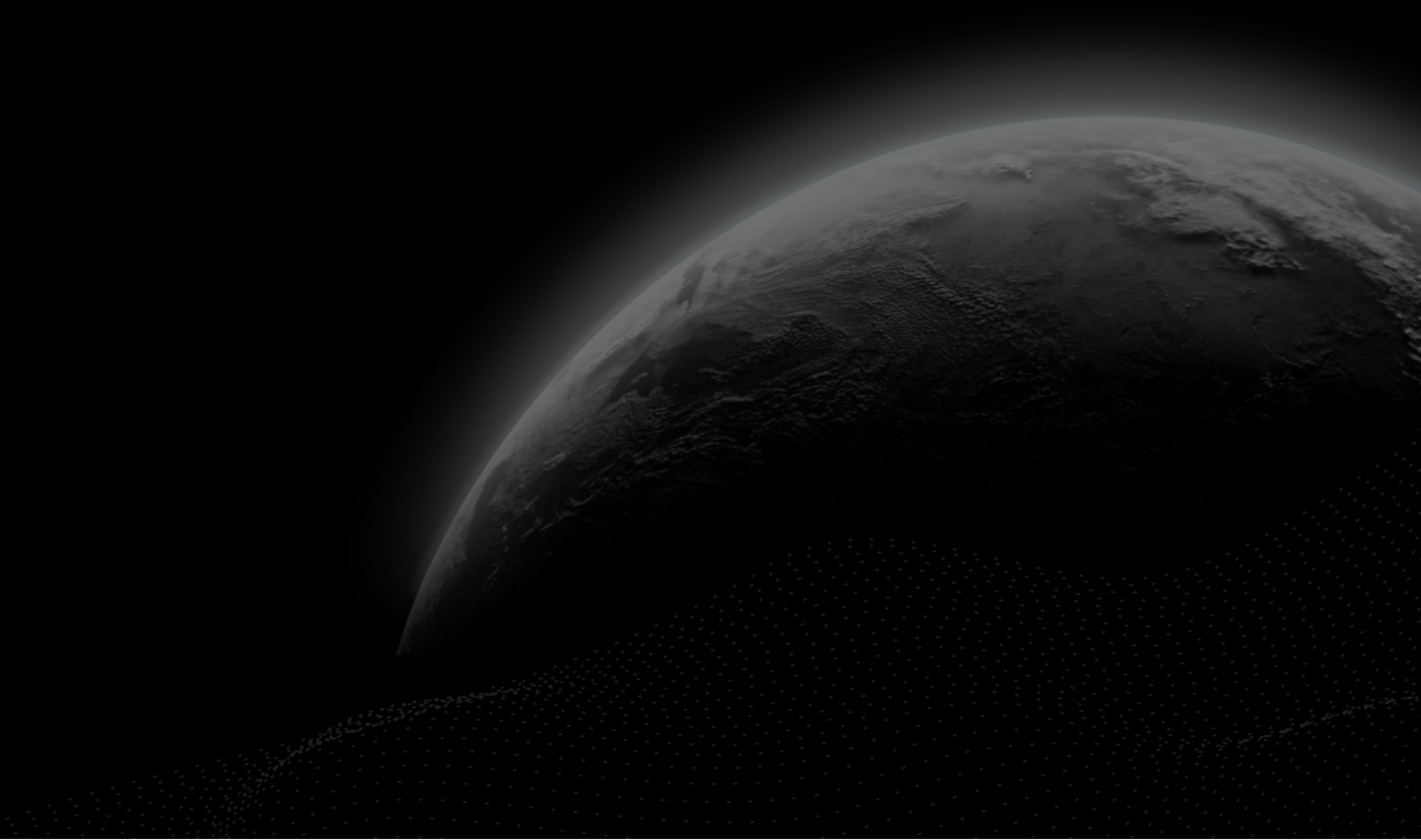
CERTIK

Security Assessment

Betfin - Events Contract

CertiK Assessed on Oct 18th, 2024



CertiK Assessed on Oct 18th, 2024

Betfin - Events Contract

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	Ethereum (ETH)	Formal Verification, Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Solidity	Delivered on 10/18/2024	N/A

CODEBASE

events-contract

View All in Codebase Page

COMMITS

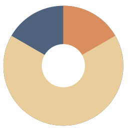
0b618136373d7ae459f5dce33d0584ec4c83c635

d98646e257ba3aa69d22b2e895943f4caeaf3bb9

6825b610c7effaf8eac7e6c37c96b7d02dd6d39c

View All in Codebase Page

Vulnerability Summary



6	5	0	0	1	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.
🟧	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.
🟨	0	Medium		Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	4	Minor	4 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.
🟦	1	Informational	1 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 | BETFIN - EVENTS CONTRACT

CODEBASE | BETFIN - EVENTS CONTRACT

Repository

events-contract

Commit

0b618136373d7ae459f5dce33d0584ec4c83c635

d98646e257ba3aa69d22b2e895943f4caeaf3bb9

6825b610c7effaf8eac7e6c37c96b7d02dd6d39c

AUDIT SCOPE | BETFIN - EVENTS CONTRACT

9 files audited ● 2 files with Acknowledged findings ● 2 files with Resolved findings ● 5 files without findings

ID	Repo	File	SHA256 Checksum
● EVE	betfinio/events-contract	src/Event.sol	40408726abc48316651d27270f8a363f7b7dad9b48eecc74ba6c6b7aaefbdec1
● EFB	betfinio/events-contract	src/EventFactory.sol	ac8aa8469e52eb34556a17aa24a16d47165dc3f63076d7c9eea509d595f503f1
● EBB	betfinio/events-contract	src/EventBet.sol	fbb6679c34e4e6aa535e3c342b6498b6411861bc036d6ada23ee6c1556d4eb7e
● EVN	betfinio/events-contract	src/Event.sol	28b53f17b30e5c33df752c7be66ab893396ec3010ebfd02cc9c416edd4550ba4
● EBU	betfinio/events-contract	src/EventBet.sol	f50ad5df2eaee79af77081de740bca3c25daa106efe809962ded7dbd330e3feb
● EFU	betfinio/events-contract	src/EventFactory.sol	99325f1b336621705efe1e75f183f834ff84f771d72bafb8fb363e3df5c5589e
● EVT	betfinio/events-contract	src/Event.sol	6a22b411308bb61989c8e4fb751ead514e825a912053a3be7a4f866d9b5cb9bd
● EBH	betfinio/events-contract	src/EventBet.sol	f50ad5df2eaee79af77081de740bca3c25daa106efe809962ded7dbd330e3feb
● EFH	betfinio/events-contract	src/EventFactory.sol	99325f1b336621705efe1e75f183f834ff84f771d72bafb8fb363e3df5c5589e

APPROACH & METHODS | BETFIN - EVENTS CONTRACT

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

The auditing process pays special attention to the following considerations:

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

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

- 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 | BETFIN - EVENTS CONTRACT

Overview

The `Event` contract and related contracts implement a betting system for specific events. They allows placing bets on different sides of an event during a specific time frame. Afterward, the owner determines the winning side, users can distribute winnings, settle losing bets, or refund depending on the event's outcome.

Privileged Functions

In the `Event` contract and related contracts, the admin roles are adopted to ensure the dynamic runtime updates of the project, which are specified in the finding `Centralization Related Risks` .

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.

External Dependencies

The `Event` contract and related contracts rely on a few external contracts or addresses to fulfill the needs of its business logic.

The following are third dependencies contracts used within the contracts:

- `IERC20` : From OpenZeppelin, this contract is the interface of the ERC-20 standard as defined in the ERC.
- `SafeERC20` : From Openzeppelin, this contract provides secure token transfer functions.
- `Ownable` : From Openzeppelin, this contract is foundational to owner permission implementation.
- `AccessControl` : From Openzeppelin, this contract implements role-based access control mechanisms.

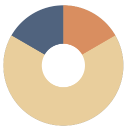
The following are external addresses used within the contracts:

- `_factory` : This contract is responsible for the user's placeBet and token transfer
- `_staking` : This contract belongs to the staking related part in **Betfin** project.

- `_core` : This contract coordinates interactions between the various contracts and maintains the state of the platform.

It is assumed that these contracts or addresses are trusted and implemented properly within the whole project.

FINDINGS | BETFIN - EVENTS CONTRACT



6	0	1	0	4	1
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Betfin - Events Contract. Through this audit, we have uncovered 6 issues ranging from different severity levels. Utilizing the techniques of Static Analysis, Formal Verification & Manual Review to complement rigorous manual code reviews, we discovered the following findings:

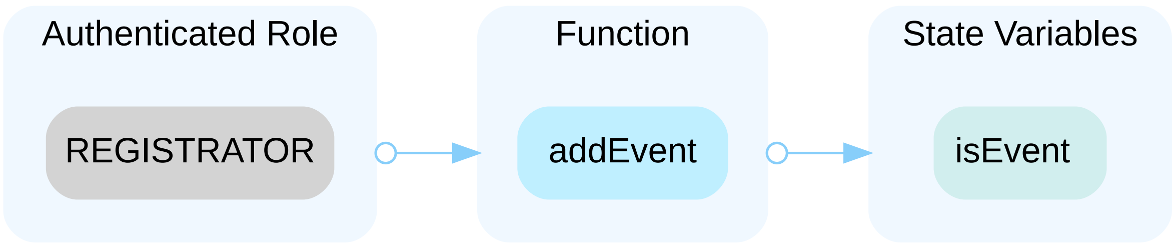
ID	Title	Category	Severity	Status
SRC-02	**Centralization Related Risks**	**Centralization**	**Major**	● **Acknowledged**
EFB-01	Bet Recipient Not Checked	Logical Issue	Minor	● Resolved
EVE-01	Potential Revert In `refundNext` Function	Logical Issue	Minor	● Resolved
EVN-01	A Minor Portion Of Tokens May Be Locked In The Contract	Incorrect Calculation	Minor	● Resolved
SRC-03	Missing Zero Address Validation	Volatile Code	Minor	● Resolved
EVE-02	Incorrect Error Message	Coding Style	Informational	● Resolved

SRC-02 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	● **Major**	**src/Event.sol (0b618136373d7ae459f5dce33d0584ec4c83c635): 125; src/EventFactory.sol (0b618136373d7ae459f5dce33d0584ec4c83c635): 77**	● **Acknowledged**

Description

In the contract `EventFactory` , the role `REGISTRATOR` has authority over the functions shown in the diagram below. Any compromise to the `REGISTRATOR` account may allow the hacker to take advantage of this authority and add an event address.



Additionally, the `EventFactory` contract inherits the `AccessControl` contract from OpenZeppelin, the `DEFAULT_ADMIN_ROLE` role has the following authorities within the contract:

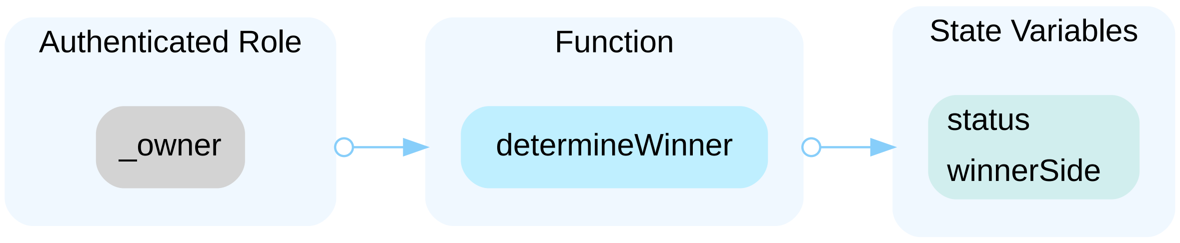
- `grantRole()` : Grants specified roles to an account, allowing it to perform actions associated with that role.
- `revokeRole()` : Removes specified roles from an account, restricting it from performing certain actions.

If the `DEFAULT_ADMIN ROLE` is compromised, an attacker could grant critical roles to unauthorized addresses, effectively allowing them to manipulate the contract. The attacker could also revoke roles from legitimate addresses, disrupting the normal operation and administration of the contract.

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

It's noted that the `Event` game utilizes the centralized winner bet results which are controlled by the contract owner.

Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and determine the winner side and update the status.



Additionally, `Event` contract inherits the `Ownable` contract from OpenZeppelin, the owner has the following authorities within the contract:

- `renounceOwnership()` : Leaves the contract without owner;
- `transferOwnership()` : Transfers ownership of the contract to a new account. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority, disrupt the normal access control.

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

[Betfin Team, 10/10/2024]:

Issue acknowledged. I won't make any changes for the current version.

[CertiK, 10/10/2024]

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

EFB-01 | BET RECIPIENT NOT CHECKED

Category	Severity	Location	Status
Logical Issue	● Minor	src/EventFactory.sol (0b618136373d7ae459f5dce33d0584ec4c83c635): 62	● Resolved

Description

In the `EventFactory` contract, the `placeBet` function fails to verify that the third address extracted from the `data` parameter through `abi.decode` matches the `player` parameter . This discrepancy could allow transactions where the bet recipient address does not align with `player` , potentially leading to situations where players might not receive their due rewards or the bet is wrongly attributed.

Recommendation

It's recommended to amend the `placeBet` function to include a validation check the bet recipient address extracted from `data` matches `player` .

Alleviation

[Betfin Team, 10/10/2024]:

From now on, we allow users to create bets for another user, by passing player recipient in data argument.

Additionally, the team added a check in the `EventBet` constructor to ensure the recipient address is not zero.

Changes have been reflected in the commit hash: https://github.com/betfinio/events-contract/commit/b89442fe8b63ec04333e810fddaa91430e341436.

EVE-01 | POTENTIAL REVERT IN `refundNext` FUNCTION

Category	Severity	Location	Status
Logical Issue	● Minor	src/Event.sol (0b618136373d7ae459f5dce33d0584ec4c83c635): 230	● Resolved

Description

The `refundNext` function in the `Event` contract is designed to process refunds for a predefined number of bets, set currently at 100. It accomplishes this by invoking the `refundNextByStep` function. However, there exists a safeguard within `refundNextByStep` that prevents the `step` parameter from exceeding the total number of bets. Consequently, if the total number of bets is fewer than 100, such as 10, directly invoking `refundNext` will trigger a transaction revert due to this limitation.

Recommendation

It's recommended to adjust the predefined step according to the remaining steps. For example:

```
    function refundNext() external {
        // execute refund with predefined step
        uint256 pending = bets.length - offset;
        if (pending == 0) return;
        refundNextByStep(CALC_STEP >= pending ? pending : CALC_STEP);
    }
```

Alleviation

[Betfin Team, 10/10/2024]:
Issue acknowledged. Changes have been reflected in the commit hash: https://github.com/betfinio/events-contract/commit/1eac825041bb0fd0fe67a7b1da7a2139b1afc6c8.

EVN-01 A MINOR PORTION OF TOKENS MAY BE LOCKED IN THE CONTRACT

Category	Severity	Location	Status
Incorrect Calculation	● Minor	src/Event.sol (10/10-d98646): 148	● Resolved

Description

In the `placeBet` function, the factory contract transfers a specified amount of tokens to the contract on behalf of the user. However, aside from the non-zero check on the amount in the `core` contract, no further checks are conducted during the subsequent token transfer process. If the user sets a smaller token amount, precision loss when calculating the fee in the core could result in a fee of 0.

Unlike the separate calculation of fee and amount in the `core` contract, the `_distribute()` function in the `Event` contract first calculates the fee based on the total amount from all participants and the bank, then distributes the reward proportionally to the winner based on this value. If the bank is large enough, the fee will not be 0. This creates an inconsistency between the total amount of tokens transferred by the factory to the `Event` contract and the actual amount used for reward distribution, with the excess becoming locked in the contract.

Proof of Concept

```
    function testLockToken() public {
        getTokens(alice, 1000);
        for (uint256 i=0; i<30; i++){
            placeBet(alice, 27, 1);
        }

        vm.warp(block.timestamp+3 days);

        _event.determineWinner(1);

        vm.startPrank(alice);
        console.log("event balance before:", token.balanceOf(address(_event)));
        _event.distribute(0, 30);
        // locked amount may increase if similar placeBets increase
        console.log("event balance after:", token.balanceOf(address(_event)));
        vm.stopPrank();
    }
```

```
[PASS] testLockToken() (gas: 16561120)
Logs:
  event balance before: 810
  event balance after: 30

Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 89.19s (62.44s CPU
time)
```

Recommendation

It is recommended to establish a minimum limit for the amount to ensure it is sufficiently large, minimizing the issue of precision loss.

Alleviation

[Betfin Team, 10/15/2024]:

Issue acknowledged. The team resolved this issue in the commit hash 6825b610c7effaf8eac7e6c37c96b7d02dd6d39c by adding a minimum value check to amount.

SRC-03 | MISSING ZERO ADDRESS VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	src/EventBet.sol (0b618136373d7ae459f5dce33d0584ec4c83c635): 26; src/EventFactory.sol (0b618136373d7ae459f5dce33d0584ec4c83c635): 34	● Resolved

Description

Addresses are not validated before assignment or external calls, potentially allowing the use of zero addresses and leading to unexpected behavior or vulnerabilities. For example, transferring tokens to a zero address can result in a permanent loss of those tokens.

The following provided addresses lack zero address validation:

```
    constructor(address _player, uint256 _amount, address _game, uint256 _side)
Ownable(_msgSender()) {
        ...
    }
```

`_player` and `_game` in the `EventBet` contract.

```
    constructor(address _staking, address _core) {
        ...
    }
```

`_staking` and `_core` in the `EventFactory` contract.

Recommendation

It is recommended to add a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

[Betfin Team, 10/10/2024]:

Issue Acknowledged. The team resolved this issue in the commit hash b89442fe8b63ec04333e810fddaa91430e341436 by adding the zero-check for provided address.

EVE-02 | INCORRECT ERROR MESSAGE

Category	Severity	Location	Status
Coding Style	● Informational	src/Event.sol (0b618136373d7ae459f5dce33d0584ec4c83c635): 23, 223	● Resolved

Description

The error message in the code `require(step > 0, "E09");` is incorrect according to the contract comment "E09 - To big step".

```
219     function refundNextByStep(uint256 step) public {
220         // Ensure the contract is in a refundable state
221         require(status == 31 || status == 32, "E11");
222         // Ensure the step is not zero or negative
223     @>  require(step > 0, "E09");
224         // Ensure the step is not larger than the total number of bets
225         require(step <= bets.length, "E09");
```

Recommendation

It's recommended to update the error message to reflect the correct meaning.

Alleviation

[Betfin Team, 10/10/2024]:
Issue acknowledged. Changes have been reflected in the commit hash: https://github.com/betfinio/events-contract/commit/54cfc2e2b14c5e8c24c9f817740e87ef22a7ee6c.

OPTIMIZATIONS | BETFIN - EVENTS CONTRACT

ID	Title	Category	Severity	Status
SRC-01	Variables That Could Be Declared As Immutable	Gas Optimization	Optimization	● Resolved

SRC-01 | VARIABLES THAT COULD BE DECLARED AS IMMUTABLE

Category	Severity	Location	Status
Gas Optimization	● Optimization	src/Event.sol (0b618136373d7ae459f5dce33d0584ec4c83c635): 36, 37, 38; src/EventBet.sol (0b618136373d7ae459f5dce33d0584ec4c83c635): 24	● Resolved

Description

The linked variables assigned in the constructor can be declared as `immutable`. Immutable state variables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. A big advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variables since they will not be stored in storage.

Recommendation

We recommend declaring these variables as immutable. Please note that the `immutable` keyword only works in Solidity version `v0.6.5` and up.

Alleviation

[Betfin Team, 10/10/2024]:
Issue acknowledged. Changes have been reflected in the commit hash: https://github.com/betfinio/events-contract/commit/54cfc2e2b14c5e8c24c9f817740e87ef22a7ee6c.

FORMAL VERIFICATION | BETFIN - EVENTS CONTRACT

Formal guarantees about the behavior of smart contracts can be obtained by reasoning about properties relating to the entire contract (e.g. contract invariants) or to specific functions of the contract. Once such properties are proven to be valid, they guarantee that the contract behaves as specified by the property. As part of this audit, we applied formal verification to prove that important functions in the smart contracts adhere to their expected behaviors.

Considered Functions And Scope

In the following, we provide a description of the properties that have been used in this audit. They are grouped according to the type of contract they apply to.

Verification of Standard Ownable Properties

We verified *partial* properties of the public interfaces of those token contracts that implement the Ownable interface. This involves:

- function `owner` that returns the current owner,
- functions `renounceOwnership` that removes ownership,
- function `transferOwnership` that transfers the ownership to a new owner.

The properties that were considered within the scope of this audit are as follows:

Property Name	Title
ownable-renounceownership-correct	Ownership is Removed
ownable-owner-succeed-normal	`owner` Always Succeeds
ownable-transferownership-correct	Ownership is Transferred
ownable-renounce-ownership-is-permanent	Once Renounced, Ownership Cannot be Regained

Verification of contracts derived from AccessControl v4.4

We verified properties of the public interface of contracts that provide an AccessControl-v4.4 compatible API. This involves:

- The `hasRole` function, which returns `true` if an account has been granted a specific `role`.
- The `getRoleAdmin` function, which returns the admin role that controls a specific `role`.
- The `grantRole` and `revokeRole` functions, which are used for granting a `role` to an account and revoking a `role` from an `account`, respectively.
- The `renounceRole` function, which allows the calling account to revoke a `role` from itself.

The properties that were considered within the scope of this audit are as follows:

Property Name	Title
accesscontrol-hasrole-change-state	`hasRole` Function Does Not Change State
accesscontrol-getroleadmin-succeed-always	`getRoleAdmin` Function Always Succeeds
accesscontrol-renouncerole-revert-not-sender	`renounceRole` Reverts When Caller Is Not the Confirmation Address
accesscontrol-renouncerole-succeed-role-renouncing	`renounceRole` Successfully Renounces Role
accesscontrol-hasrole-succeed-always	`hasRole` Function Always Succeeds
accesscontrol-getroleadmin-change-state	`getRoleAdmin` Function Does Not Change State
accesscontrol-revokerole-correct-role-revoking	`revokeRole` Correctly Revokes Role
accesscontrol-grantrole-correct-role-granting	`grantRole` Correctly Grants Role
accesscontrol-default-admin-role	AccessControl Default Admin Role Invariance

Verification Results

For the following contracts, formal verification established that each of the properties that were in scope of this audit (see scope) are valid:

Detailed Results For Contract Event (src/Event.sol) In Commit 6825b610c7effaf8eac7e6c37c96b7d02dd6d39c

Verification of Standard Ownable Properties

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results For Contract EventBet (src/EventBet.sol) In Commit 6825b610c7effaf8eac7e6c37c96b7d02dd6d39c

Verification of Standard Ownable Properties

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	
ownable-renounce-ownership-is-permanent	● True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results For Contract EventFactory (src/EventFactory.sol) In Commit 6825b610c7effaf8eac7e6c37c96b7d02dd6d39c

Verification of contracts derived from AccessControl v4.4

Detailed Results for Function `hasRole`

Property Name	Final Result	Remarks
accesscontrol-hasrole-change-state	● True	
accesscontrol-hasrole-succeed-always	● True	

Detailed Results for Function `getRoleAdmin`

Property Name	Final Result	Remarks
accesscontrol-getroleadmin-succeed-always	● True	
accesscontrol-getroleadmin-change-state	● True	

Detailed Results for Function `renounceRole`

Property Name	Final Result	Remarks
accesscontrol-renouncerole-revert-not-sender	● True	
accesscontrol-renouncerole-succeed-role-renouncing	● True	

Detailed Results for Function `revokeRole`

Property Name	Final Result	Remarks
accesscontrol-revokerole-correct-role-revoking	● True	

Detailed Results for Function `grantRole`

Property Name	Final Result	Remarks
accesscontrol-grantrole-correct-role-granting	● True	

Detailed Results for Function `DEFAULT_ADMIN_ROLE`

Property Name	Final Result	Remarks
accesscontrol-default-admin-role	● True	

Detailed Results For Contract Event (src/Event.sol) In Commit d98646e257ba3aa69d22b2e895943f4caeaf3bb9

Verification of Standard Ownable Properties

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results For Contract EventBet (src/EventBet.sol) In Commit d98646e257ba3aa69d22b2e895943f4caeaf3bb9

Verification of Standard Ownable Properties

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	
ownable-renounce-ownership-is-permanent	● True	

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results For Contract EventFactory (src/EventFactory.sol) In Commit d98646e257ba3aa69d22b2e895943f4caeaf3bb9

Verification of contracts derived from AccessControl v4.4

Detailed Results for Function `getRoleAdmin`

Property Name	Final Result	Remarks
accesscontrol-getroleadmin-succeed-always	● True	
accesscontrol-getroleadmin-change-state	● True	

Detailed Results for Function `renounceRole`

Property Name	Final Result	Remarks
accesscontrol-renouncerole-revert-not-sender	● True	
accesscontrol-renouncerole-succeed-role-renouncing	● True	

Detailed Results for Function `hasRole`

Property Name	Final Result	Remarks
accesscontrol-hasrole-succeed-always	● True	
accesscontrol-hasrole-change-state	● True	

Detailed Results for Function `DEFAULT_ADMIN_ROLE`

Property Name	Final Result	Remarks
accesscontrol-default-admin-role	● True	

Detailed Results for Function `grantRole`

Property Name	Final Result	Remarks
accesscontrol-grantrole-correct-role-granting	● True	

Detailed Results for Function `revokeRole`

Property Name	Final Result	Remarks
accesscontrol-revokerole-correct-role-revoking	● True	

Detailed Results For Contract Event (src/Event.sol) In Commit 0b618136373d7ae459f5dce33d0584ec4c83c635

Verification of Standard Ownable Properties

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	

Detailed Results For Contract EventBet (src/EventBet.sol) In Commit 0b618136373d7ae459f5dce33d0584ec4c83c635

Verification of Standard Ownable Properties

Detailed Results for Function `transferOwnership`

Property Name	Final Result	Remarks
ownable-transferownership-correct	● True	

Detailed Results for Function `owner`

Property Name	Final Result	Remarks
ownable-owner-succeed-normal	● True	

Detailed Results for Function `renounceOwnership`

Property Name	Final Result	Remarks
ownable-renounceownership-correct	● True	
ownable-renounce-ownership-is-permanent	● True	

Detailed Results For Contract EventFactory (src/EventFactory.sol) In Commit 0b618136373d7ae459f5dce33d0584ec4c83c635

Verification of contracts derived from AccessControl v4.4

Detailed Results for Function `renounceRole`

Property Name	Final Result	Remarks
accesscontrol-renouncerole-succeed-role-renouncing	● True	
accesscontrol-renouncerole-revert-not-sender	● True	

Detailed Results for Function `revokeRole`

Property Name	Final Result	Remarks
accesscontrol-revokerole-correct-role-revoking	● True	

Detailed Results for Function `grantRole`

Property Name	Final Result	Remarks
accesscontrol-grantrole-correct-role-granting	● True	

Detailed Results for Function `DEFAULT_ADMIN_ROLE`

Property Name	Final Result	Remarks
accesscontrol-default-admin-role	● True	

Detailed Results for Function `getRoleAdmin`

Property Name	Final Result	Remarks
accesscontrol-getroleadmin-change-state	● True	
accesscontrol-getroleadmin-succeed-always	● True	

Detailed Results for Function `hasRole`

Property Name	Final Result	Remarks
accesscontrol-hasrole-change-state	● True	
accesscontrol-hasrole-succeed-always	● True	

APPENDIX | BETFIN - EVENTS CONTRACT

Finding Categories

Categories	Description
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.
Coding Style	Coding Style findings may not affect code behavior, but indicate areas where coding practices can be improved to make the code more understandable and maintainable.
Incorrect Calculation	Incorrect Calculation findings are about issues in numeric computation such as rounding errors, overflows, out-of-bounds and any computation that is not intended.
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.

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.

Details on Formal Verification

Some Solidity smart contracts from this project have been formally verified. Each such contract was compiled into a mathematical model that reflects all its possible behaviors with respect to the property. The model takes into account the semantics of the Solidity instructions found in the contract. All verification results that we report are based on that model.

The following assumptions and simplifications apply to our model:

- Certain low-level calls and inline assembly are not supported and may lead to a contract not being formally verified.
- We model the semantics of the Solidity source code and not the semantics of the EVM bytecode in a compiled contract.

Formalism for property specifications

All properties are expressed in a behavioral interface specification language that CertiK has developed for Solidity, which allows us to specify the behavior of each function in terms of the contract state and its parameters and return values, as well as contract properties that are maintained by every observable state transition. Observable state transitions occur when the contract's external interface is invoked and the invocation does not revert, and when the contract's Ether balance is changed by the EVM due to another contract's "self-destruct" invocation. The specification language has the usual Boolean connectives, as well as the operator `\old` (used to denote the state of a variable before a state transition), and several types of specification clause:

Apart from the Boolean connectives and the modal operators "always" (written `[]`) and "eventually" (written `<>`), we use the following predicates to reason about the validity of atomic propositions. They are evaluated on the contract's state whenever a discrete time step occurs:

- `requires [cond]` - the condition `cond`, which refers to a function's parameters, return values, and contract state variables, must hold when a function is invoked in order for it to exhibit a specified behavior.
- `ensures [cond]` - the condition `cond`, which refers to a function's parameters, return values, and both `\old` and current contract state variables, is guaranteed to hold when a function returns if the corresponding requires condition held when it was invoked.
- `invariant [cond]` - the condition `cond`, which refers only to contract state variables, is guaranteed to hold at every observable contract state.
- `constraint [cond]` - the condition `cond`, which refers to both `\old` and current contract state variables, is guaranteed to hold at every observable contract state except for the initial state after construction (because there is no previous state); constraints are used to restrict how contract state can change over time.

Description of the Analyzed AccessControl-v4.4 Properties

Properties related to function `hasRole`

accesscontrol-hasrole-change-state

The `hasRole` function must not change any state variables.

Specification:

```
assignable \nothing;
```

accesscontrol-hasrole-succeed-always

The `hasRole` function must always succeed, assuming that its execution does not run out of gas.

Specification:

```
reverts_only_when false;
```

Properties related to function `getRoleAdmin`

accesscontrol-getroleadmin-change-state

The `getRoleAdmin` function must not change any state variables.

Specification:

```
assignable \nothing;
```

accesscontrol-getroleadmin-succeed-always

The `getRoleAdmin` function must always succeed, assuming that its execution does not run out of gas.

Specification:

```
reverts_only_when false;
```

Properties related to function `renounceRole`

accesscontrol-renouncerole-revert-not-sender

The `renounceRole` function must revert if the caller is not the same as `account`.

Specification:

```
reverts_when account != msg.sender;
```

accesscontrol-renouncerole-succeed-role-renouncing

After execution, `renounceRole` must ensure the caller no longer has the renounced role.

Specification:

```
ensures !hasRole(role, account);
```

Properties related to function `revokeRole`

accesscontrol-revokerole-correct-role-revoking

After execution, `revokeRole` must ensure the specified account no longer has the revoked role.

Specification:

```
ensures !hasRole(role, account);
```

Properties related to function `grantRole`

accesscontrol-grantrole-correct-role-granting

After execution, `grantRole` must ensure the specified account has the granted role.

Specification:

```
ensures hasRole(role, account);
```

Properties related to function `DEFAULT_ADMIN_ROLE`

accesscontrol-default-admin-role

The default admin role must be invariant, ensuring consistent access control management.

Specification:

```
invariant DEFAULT_ADMIN_ROLE() == 0x00;
```

Description of the Analyzed Ownable Properties

Properties related to function `renounceOwnership`

ownable-renounce-ownership-is-permanent

The contract must prohibit regaining of ownership once it has been renounced.

Specification:

```
constraint \old(owner()) == address(0) ==> owner() == address(0);
```

ownable-renounceownership-correct

Invocations of `renounceOwnership()` must set ownership to address(0).

Specification:

```
ensures this.owner() == address(0);
```

Properties related to function `owner`

ownable-owner-succeed-normal

Function `owner` must always succeed if it does not run out of gas.

Specification:

```
reverts_only_when false;
```

Properties related to function `transferOwnership`

ownable-transferownership-correct

Invocations of `transferOwnership(newOwner)` must transfer the ownership to the `newOwner`.

Specification:

```
ensures this.owner() == newOwner;
```

DISCLAIMER | CERTIK

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.