IMMUNEBYTES
Audits

The Winkies

SMART CONTRACT AUDIT
FINAL REPORT

February 14, 2022

IMMUNEBYTES

TOC

TABLE OF CONTENTS

Introduction

1. About The Winkies

Born at Ecole Polytechnique (France's TOP Engineering School) in 2017, Mainbot was created with a goal in mind: Leveraging new technology to enhance kids' education and prepare them for the future.

The first product is an educational and evolutive robot called Winky, which is already used daily by thousands of families across Europe to teach 5 to 12 year old about robotics, programming and artificial intelligence.

The team is now ready to take education to the next level with the launch of The WinkyVerse: a Global Education Games Metaverse powered by its own digital currency, The Winkies.
The WinkyVerse is the first educational ecosystem who manages to combine no less than six of the most promising technologies: Robotics, Artificial Intelligence, Programming, Gaming, Augmented Reality and Blockchain.

Visit https://getwinkies.com/ to learn more about it.

2. About ImmuneBytes

ImmuneBytes is a security start-up to provide professional services in the blockchain space. The team has hands-on experience in conducting smart contract audits, penetration testing, and security consulting. ImmuneBytes's security auditors have worked on various A-league projects and have a great understanding of DeFi projects like AAVE, Compound, 0x Protocol, Uniswap, dydx.

The team has been able to secure 125+ blockchain projects by providing security services on different frameworks. ImmuneBytes team helps start-ups with a detailed analysis of the system ensuring security and managing the overall project.

Visit http://immunebytes.com/ to know more about the services.

Documentation Details

The Winkies team has provided the following doc for the purpose of audit:

1. https://drive.google.com/file/d/1JUVMHTNB2X6POYuJB3PHMWICZJm3dsqd/view

Audit Process & Methodology

ImmuneBytes team has performed thorough testing of the project starting with analyzing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third-party smart contracts and libraries.

Our team then performed a formal line-by-line inspection of the Smart Contract in order to find any potential issues like Signature Replay Attacks, Unchecked External Calls, External Contract Referencing, Variable Shadowing, Race conditions, Transaction-ordering dependence, timestamp dependence, DoS attacks, and others.

In the Unit testing phase, we run unit tests written by the developer in order to verify the functions work as intended. In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was audited by a team of independent auditors which includes -

1. Testing the functionality of the Smart Contract to determine proper logic has been followed throughout.
2. Analyzing the complexity of the code by thorough, manual review of the code, line-by-line.
3. Deploying the code on testnet using multiple clients to run live tests.
4. Analyzing failure preparations to check how the Smart Contract performs in case of bugs and vulnerabilities.
5. Checking whether all the libraries used in the code are on the latest version.
6. Analyzing the security of the on-chain data.

Audit Details

- Project Name: The Winkies
- Contracts Name: IStakingPlatform.sol, StakingPlatform.sol, TesterStakingPlatform.sol
- Languages: Solidity(Smart contract), Typescript (Unit Testing)
- Github commits for the audit: 9d8689176435bcec5091ea5c7ea7425b7a93891d
- Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Contract Library, Slither, SmartCheck

Audit Goals

The focus of the audit was to verify that the smart contract system is secure, resilient, and working according to its specifications. The audit activities can be grouped into the following three categories:

1. Security: Identifying security-related issues within each contract and within the system of contracts.
2. Sound Architecture: Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.
3. Code Correctness and Quality: A full review of the contract source code. The primary areas of focus include
    a. Correctness
    b. Readability
    c. Sections of code with high complexity
    d. Quantity and quality of test coverage

Security Level Reference

Every issue in this report were assigned a severity level from the following:

Admin/Owner Privileges can be misused either intentionally or unintentionally.
High severity issues will bring problems and should be fixed.
Medium severity issues could potentially bring problems and should eventually be fixed.
Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Issues	High	Medium	Low
Open	-	-	2
Closed	-	-	-

Contract Name: StakingPlatform, StakingPlatformTester

High Severity Issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

1. **Constant declaration should be preferred**
   Line no-27

   Explanation:
   State variables that are not supposed to change throughout the contract should be declared as **constant.**

   The state variable **_precision** is not updated or modified throughout the contract. Therefore, a constant keyword can be assigned to the variable.

   Recommendation:
   Constant keyword should be attached to variables unless the current design is intended.

2. **External Visibility should be preferred**

   Explanation:
   Functions that are never called throughout the contract should be marked as **external** visibility instead of **public** visibility.

   This will effectively result in Gas Optimization as well.

   Therefore, the following function in the StakingPlatformTester contract could be marked as **external** within the contract:

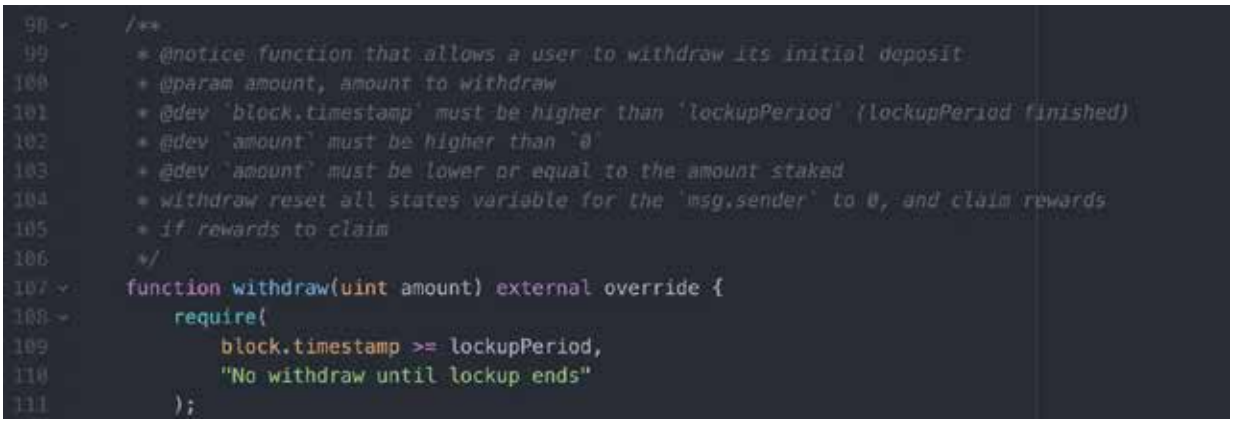
   - setPrecision(uint)

   Recommendation:
   If the **PUBLIC** visibility of the above-mentioned functions is not intended, then the **EXTERNAL** Visibility keyword should be preferred.

Recommendations/Informational

1. **Inadequate Natspec Annotations found for withdraw() function**
   Line no: 104

   Explanation:
   The Natspec annotations assigned to the withdraw function states - "withdraw reset all states variable for the `msg.sender` to 0, and claim rewards"



```
    /**
     * @notice function that allows a user to withdraw its initial deposit
     * @param amount, amount to withdraw
     * @dev `block.timestamp` must be higher than `lockupPeriod` (lockupPeriod finished)
     * @dev `amount` must be higher than `0`
     * @dev `amount` must be lower or equal to the amount staked
     * withdraw reset all states variable for the `msg.sender` to 0, and claim rewards
     * if rewards to claim
     */
    function withdraw(uint amount) external override {
        require(
            block.timestamp >= lockupPeriod,
            "No withdraw until lockup ends"
        );
```

   However, the same is not true as withdraw() function simply deducts the withdraw amount staked mapping for the caller, instead of reverting them back to zero.

   Recommendation:
   It is recommended to provide accurate natspec annotations for every function to avoid confusion and increase readability.

Automated Audit Result

1. StakingPlatform.sol

```
Compiled with solc
Number of lines: 891 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 7 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 3
Number of informational issues: 27
Number of low issues: 14
Number of medium issues: 2
Number of high issues: 2
ERCs: ERC20

+-----------------+-------------+-------+--------------------+--------------+--------------------+
|      Name       | # functions | ERCS  |     ERC20 info     | Complex code |      Features      |
+-----------------+-------------+-------+--------------------+--------------+--------------------+
| StakingPlatform |      30     |       |                    |      No      |     Send ETH       |
|                 |             |       |                    |              | Tokens interaction |
|      IERC20     |      6      | ERC20 |     No Minting     |      No      |                    |
|                 |             |       | Approve Race Cond. |              |                    |
|    SafeERC20    |      6      |       |                    |      No      |     Send ETH       |
|                 |             |       |                    |              | Tokens interaction |
|     Address     |      11     |       |                    |      No      |     Send ETH       |
|                 |             |       |                    |              |    Delegatecall    |
|                 |             |       |                    |              |      Assembly      |
+-----------------+-------------+-------+--------------------+--------------+--------------------+
contracts/staking/StakingPlatform.sol analyzed (7 contracts)
```

2. TesterStakingPlatform.sol

```
Compiled with solc
Number of lines: 933 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 8 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 3
Number of informational issues: 28
Number of low issues: 14
Number of medium issues: 2
Number of high issues: 2
ERCs: ERC20

+----------------------+-------------+-------+--------------------+--------------+--------------------+
|         Name         | # functions | ERCS  |     ERC20 info     | Complex code |      Features      |
+----------------------+-------------+-------+--------------------+--------------+--------------------+
| StakingPlatformTester|      34     |       |                    |      No      | Tokens interaction |
|        IERC20        |      6      | ERC20 |     No Minting     |      No      |                    |
|                      |             |       | Approve Race Cond. |              |                    |
|       SafeERC20      |      6      |       |                    |      No      |     Send ETH       |
|                      |             |       |                    |              | Tokens interaction |
|       Address        |      11     |       |                    |      No      |     Send ETH       |
|                      |             |       |                    |              |    Delegatecall    |
|                      |             |       |                    |              |      Assembly      |
+----------------------+-------------+-------+--------------------+--------------+--------------------+
contracts/staking/TesterStakingPlatform.sol analyzed (8 contracts)
```

3. Token.sol

```
Compiled with solc
Number of lines: 500 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 5 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 11
Number of informational issues: 9
Number of low issues: 0
Number of medium issues: 0
Number of high issues: 0

ERCs: ERC20

+-------+-------------+-------+--------------------+--------------+----------+
|  Name | # functions |  ERCS |     ERC20 info     | Complex code | Features |
+-------+-------------+-------+--------------------+--------------+----------+
| Token |      30     | ERC20 |     No Minting     |      No      |          |
|       |             |       | Approve Race Cond. |              |          |
|       |             |       |                    |              |          |
+-------+-------------+-------+--------------------+--------------+----------+
contracts/token/Token.sol analyzed (5 contracts)
```

Unit Tests

1. Staking Platform for DEEP POOL

```
StakingPlatform - Deep Pool
  ✓ Should deploy the new Token (437ms)
  ✓ should distribute tokens among users (91ms)
  ✓ Should deploy the new staking platform
  ✓ Should send tokens to staking platform
  ✓ Should deposit to staking platform (140ms)
  ✓ Should return the amount staked
  ✓ Should start Staking and ending period should last 1 year
  ✓ Should fail if trying to start Staking twice
  ✓ Should return the amount staked
  ✓ Should return and claim rewards staked after 1 day (93ms)
  ✓ Should return the amount staked after 1 day (85ms)
  ✓ Should return the amount of rewards for a specific user after 1 day
  ✓ Should revert if exceed the max staking amount
  ✓ Should deposit 100 000 tokens
  ✓ Should deposit 900 000 tokens
  ✓ Should fail deposit tokens
  ✓ Should fail withdraw residual before ending period
  ✓ Should fail withdraw tokens before ending period
  ✓ Should fail claiming tokens
  ✓ Should not withdraw tokens before lockup period
  ✓ Should not withdraw tokens after 200days lockup still active
  ✓ Should fail claiming tokens
  ✓ Should withdraw tokens after ending period
  ✓ Should return the amount staked after 1000 days (94ms)
  ✓ Should withdraw initial deposit (85ms)
  ✓ Should withdraw residual balances
  ✓ Should fail withdraw residual if nothing to withdraw after increasing 1000days
  ✓ Should return the amount staked once staking finished and withdrew
  ✓ Should fail deposit after staking ended
  ✓ Should return the amount staked


  30 passing (1s)
```

2. Staking Platform for MID POOL

```
StakingPlatform - Mid Pool
  ✓ Should deploy the new Token (491ms)
  ✓ should distribute tokens among users
  ✓ Should deploy the new staking platform
  ✓ Shoud increase precision
  ✓ Should send tokens to staking platform
  ✓ Should deposit to staking platform (50ms)
  ✓ Should return the amount staked
  ✓ Should start Staking and ending period should last 1 year
  ✓ Should fail if trying to start Staking twice
  ✓ Should return the amount staked
  ✓ Should revert if exceed the max staking amount
  ✓ Should claim rewards and stake for 183 days (2463ms)
  ✓ Should claim rewards and stake for 182 (total 1year) days (2221ms)
  ✓ Should not withdraw residual balances before endingperiod + 1 year
  ✓ Should withdraw residual balances
  ✓ Should fail withdraw initial deposit after withdrawResidualBalance
  ✓ Should withdraw initial deposit
  ✓ Should withdraw residual after tokens sent to contract
  ✓ Should fail withdraw residual if no residual balance


  19 passing (5s)
```

3. Staking Platform for QUICK POOL

```
StakingPlatform - Quick Pool
  ✓ Should deploy the new Token (379ms)
  ✓ should distribute tokens among users (113ms)
  ✓ Should deploy the new staking platform
  ✓ Shoud increase precision
  ✓ Should send tokens to staking platform
  ✓ Should deposit to staking platform (141ms)
  ✓ Should return the amount staked
  ✓ Should start Staking and ending period should last 1 year
  ✓ Should fail if trying to start Staking twice
  ✓ Should return the amount staked
  ✓ Should return and claim rewards staked after 1 day (89ms)
  ✓ Should revert if exceed the max staking amount
  ✓ Should deposit 100 000 tokens
  ✓ Should deposit 900 000 tokens
  ✓ Should fail deposit tokens
  ✓ Should fail withdraw tokens before ending period
  ✓ Should withdraw tokens after lockup period
  ✓ Should withdraw tokens after lockup period
  ✓ Should return the amount staked after 185 days (total 366 passed days) (92ms)
  ✓ Should withdraw initial deposit (80ms)
  ✓ Should return the amount staked once staking finished and withdrew
  ✓ Should not withdraw residual balances before endingperiod + 1 year
  ✓ Should withdraw residual balances
  ✓ Should fail withdraw residual if nothing to withdraw
  ✓ Should fail deposit after staking ended
  ✓ Should return the amount staked


26 passing (1s)
```

4. Pool Test

```
  StakingPlatform - Pool
    ✓ Should deploy the new Token (309ms)
    ✓ should distribute tokens among users (66ms)
    ✓ Should deploy the new staking platform
    ✓ Should set precision to 2
    ✓ Should send tokens to staking platform
    ✓ Should deposit to staking platform for user1 and user2 (51ms)
    ✓ Should return the amount staked
    ✓ Should withdraw tokens before staking starts
    ✓ Should re-deposit to staking platform for user1 and user2
    ✓ Should start Staking and ending period should last 1 year
    ✓ Should fail if trying to start Staking twice
    ✓ Should return the amount staked
    ✓ Should return and claim rewards staked after 1 day (40ms)
    ✓ Should return the amount staked after 1 day (2days passed)
    ✓ Should return the amount staked after 10 days (12days passed) (65ms)
    ✓ Should deposit balance for user1 & user2
    ✓ Should return the amount staked after increasing staked for 10days (22days passed) (80ms)
    ✓ Should deposit balance for user1 & user2
    ✓ Should deposit balance for user3 & user4 (122days passed)
    ✓ Should return the amount staked after 10 days (132days passed) (57ms)
    ✓ Should deposit balance for user1 & user2
    ✓ Should return the amount staked after 10 days (142 days passed) (55ms)
    ✓ Should return the amount of rewards for a specific user after 1 day (143days passed)
    ✓ Should revert if exceed the max staking amount
    ✓ Should deposit 100 000 tokens
    ✓ Should deposit 900 000 tokens
    ✓ Should fail deposit tokens
    ✓ Should fail withdraw residual before ending period
    ✓ Should fail withdraw tokens before ending period
    ✓ Should fail claiming tokens (144 days passed)
    ✓ Should not withdraw tokens after 200days lockup still active (344 days passed)
    ✓ Should deposit to staking platform for user3 and user4
    ✓ Should return the total amount staked
    ✓ Should return and claim rewards staked after 1 day (345 days passed) (44ms)
    ✓ Should withdraw tokens after ending period (845days passed)
    ✓ Should withdraw tokens after ending period
    ✓ Should return the amount staked after 1000 days
    ✓ Should withdraw initial deposit (70ms)
```

5. Withdraw Tests

```
StakingPlatform - Withdraw Amount
  ✓ Should return rewards at start
  ✓ Should return rewards after one day
  ✓ Should return rewards after 50 days
  ✓ Should return rewards after 100 days
  ✓ Should withdraw after 50 days
  ✓ Should withdraw 90% after 50 days and returns rewards after endPeriod (66ms)


6 passing (1s)
```

6. Pool Restake Tests

```
StakingPlatform - Restake
  ✓ Should deploy the new Token (369ms)
  ✓ should distribute tokens among users (61ms)
  ✓ Should deploy the new staking platform
  ✓ Should send tokens to staking platform
  ✓ Should deposit to staking platform for user1 and user2 (51ms)
  ✓ Should return the amount staked
  ✓ Should start Staking and ending period should last 1 year
  ✓ Should fail if trying to start Staking twice
  ✓ Should return the amount staked
  ✓ Should return and claim rewards staked after 1 day (41ms)
  ✓ Should return the amount staked after 1 day (2days passed)
  ✓ Should return the amount staked after 10 days (12days passed) (69ms)
  ✓ Should deposit balance for user1 & user2
  ✓ Should return the amount staked after increasing staked for 10days (22days passed) (68ms)
  ✓ Should return the amount staked after increasing staked for 10days (32days passed) (59ms)
  ✓ Should deposit a second time balance for user1 & user2
  ✓ Should withdraw after 100days for user1 and user2
  ✓ Should have 0 rewards for user1 & user2 after 10days
  ✓ Should deposit balance for user3 & user4 (122days passed) (50ms)
  ✓ Should return the amount staked after 10 days (132days passed) (101ms)
  ✓ Should deposit balance for user3 & user4 (47ms)
  ✓ Should return the amount staked after 30 days (213ms)
  ✓ Should re-deposit balance for user3 & user4
  ✓ Should return the amount staked after 30 more days (190ms)
  ✓ Should return the amount of rewards for a specific user after 1 day
  ✓ Should revert if exceed the max staking amount
  ✓ Should deposit 1 000 000 tokens with user5
  ✓ Should deposit 100 000 tokens
  ✓ Should deposit 900 000 tokens
  ✓ Should fail deposit tokens
  ✓ Should fail withdraw residual before ending period
  ✓ Should return 0 rewards after 0 days
  ✓ Should revert if nothing to claim after 0 days
  ✓ Should loop and try to withdraw / deposit (469ms)
  ✓ Should return 0 rewards after 0 days
  ✓ Should return rewards after 1 day
  ✓ Should deposit 1 000 000 tokens with user6
  ✓ Should deposit 1 000 000 tokens with user7
  ✓ Should return rewards after 100 day
  ✓ Should return rewards for user7 (119ms)
```

7. Initial Tests

```
StakingPlatform - PoolTests
  ✓ Should return rewards at start
  ✓ Should return rewards after one day
  ✓ Should return rewards after 50 days
  ✓ Should return rewards after 100 days
  ✓ Should return rewards after 200 days (endPeriod + 100days)
  ✓ Should deposit after 50 days and farm until endPeriod (51ms)
  ✓ Should deposit after 50 days and farm until endPeriod: precision(20) (59ms)
  ✓ Should withdraw and deposit & farm all together at the same time until endPeriod (82ms)
  ✓ Should withdraw after 99 days and re-deposit and farm until endPeriod, withdraw scenario (116ms)
  ✓ Should withdraw after 99 days and re-deposit and farm until endPeriod, claimRewards scenario (135ms)
  ✓ Should withdraw after 99 and farm until endPeriod (1day) (73ms)
  ✓ Should withdraw after 99 and farm half a day and then withdraw before ending (102ms)
  ✓ Should test with 0 (70ms)
  ✓ Should test with low value (52ms)
  ✓ Should fail with very low value
  ✓ Should withdraw residual if nobody claimedRewards
  ✓ Should withdraw residual if rewards claimed
  ✓ Should withdraw residual if rewards claimed (with low values) (65ms)
  ✓ Should withdraw residual if nobody claimedRewards (with low values)


19 passing (3s)
```

Concluding Remarks

While conducting the audits of The Winkies smart contract, it was observed that the contracts contain only Low severity issues.

Our auditors suggest that Low severity issues should be resolved by the developers. The recommendations given will improve the operations of the smart contract.

Disclaimer

ImmuneBytes's audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

Our team does not endorse The Winkies platform or its product nor this audit is investment advice.
Notes:

- Please make sure contracts deployed on the mainnet are the ones audited.
- Check for the code refactor by the team on critical issues.

IMMUNEBYTES

Audits