BlockPad

BPADToken & Blacklistable

Smart Contract Audit
Final Report

IMMUNEBYTES

Audits

BLOCKPAD

November 23, 2021

Introduction

1. About BlockPad

The BlockPad is a multichain network of decentralized applications that acts as a gateway to the Web 3.0 ecosystem. It predominantly solves the issue of investor confidence in Decentralized Finance (DeFi) by providing a suite of decentralized and intuitive products with compatibility of multiple blockchains.

It offers a decentralized way to safeguard investors from a range of projects that might pull out the liquidity or sell off vested tokens. On the other hand, projects can leverage its ecosystem from raising funds to support tokens by staking and liquidity mining applications.

Visit https://theblockpad.com/ to know more about.

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 105+ blockchain projects by providing security services on different frameworks. ImmuneBytes team helps start-up 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 BlockPad team has provided the following doc for the purpose of audit:

1. https://theblockpad.s3.amazonaws.com/pitchdeck.pdf

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: BlockPad
- Contracts Name: BPADToken.sol, Blacklistable.sol
- Languages: Solidity(Smart contract)
- Github commit/Smart Contract Address for audit: 2146cbc45b9615dc14aacd5aeb5c240a4a01c7b2
- Github commit/Smart Contract Address for audit: dd814448eb8885e9ea0ccdfde78748bfd1711375
- 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 References

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

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/Recommendations 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	-	-	-
Closed	-	-	3

Contract: Blacklistable.sol

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

1. **Absence of Error messages in Require Statements**
   Line no - 59, 28, 19

   Description:
   The **Blacklistable** contract includes a few functions(at the above-mentioned lines) that don't contain any error message in the **require** statement.

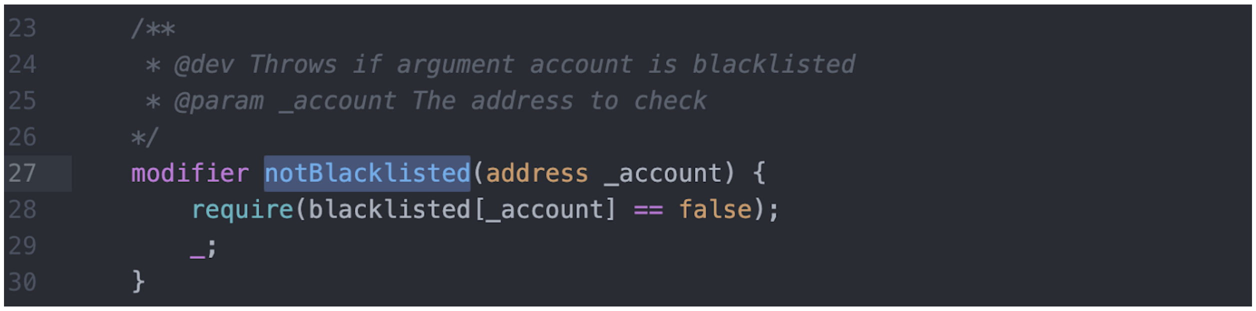
   While this makes it troublesome to detect the reason behind a particular function revert, it also reduces the readability of the code.

   Recommendation:
   Error Messages must be included in every **require** statement in the contract.

   Amended (November 23rd 2021): Issue was fixed by the **BlockPad** team and is no longer present in commit dd814448eb8885e9ea0ccdfde78748bfd1711375

2. **Unused modifier found in the contract**
   Line - 27

   Description:
   The contract includes a modifier called **nonBlacklisted** which is never used throughout the contract.



```
    /**
     * @dev Throws if argument account is blacklisted
     * @param _account The address to check
    */
    modifier notBlacklisted(address _account) {
        require(blacklisted[_account] == false);
        _;
    }
```

While this unnecessarily uses contract storage, it also reduces the code readability.

Recommendation:
Functions, state variables, or modifiers that are not supposed to be used in a contract should be removed.

Amended (November 23rd 2021): Issue was fixed by the **BlockPad** team and is no longer present in commit dd814448eb8885e9ea0ccdfde78748bfd1711375

Recommendations

1. **Inadequate input validations found in the Contract functions**
   Line no - 44, 53

   Description:
   The contract includes functions like **blackList()** & **unBlackList()** that do not contain adequate input validations within themselves.

   a. In the **blackList()** function, it is not checked whether the argument address being passed is already blacklisted.
   b. While, in the **unBlackList()** function, it is not checked if the argument address being passed is already a blacklisted address.

   Recommendation:
   It is quite important to include necessary input validations in the functions to ensure no invalid argument is passed and the redundant state variable update is controlled.

   Amended (November 23rd 2021): Issue was fixed by the **BlockPad** team and is no longer present in commit dd814448eb8885e9ea0ccdfde78748bfd1711375

2. **Blacklisting and Unblacklisting of addresses won't work unless blacklister is set.**

   Description:
   The **blackList()** & **unBlackList()** function has a modifier called **onlyBlacklister()** associated with them which ensures that only the blacklister address can **blacklist or unblacklist** a given address.

   However, the blacklister address is a **Zero Address** right after the contract deployment.

   Therefore, the **blackList() & unBlackList()** won't work unless the **blacklister** address is set as a valid address in the contract.

   Recommendation:
   It is recommended to trigger the **updateBlacklister()** function after the contract deployment, to ensure that the **blacklister** address is set and the blacklisting features can work as expected.

   Amended (November 23rd 2021): Issue was fixed by the **BlockPad** team and is no longer present in commit dd814448eb8885e9ea0ccdfde78748bfd1711375

Contract: BPADToken.sol

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

1. **External Visibility should be preferred**

   Description:
   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 must be marked as **external** within the contract:
   - **togglePause()** function at **Line 31**

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

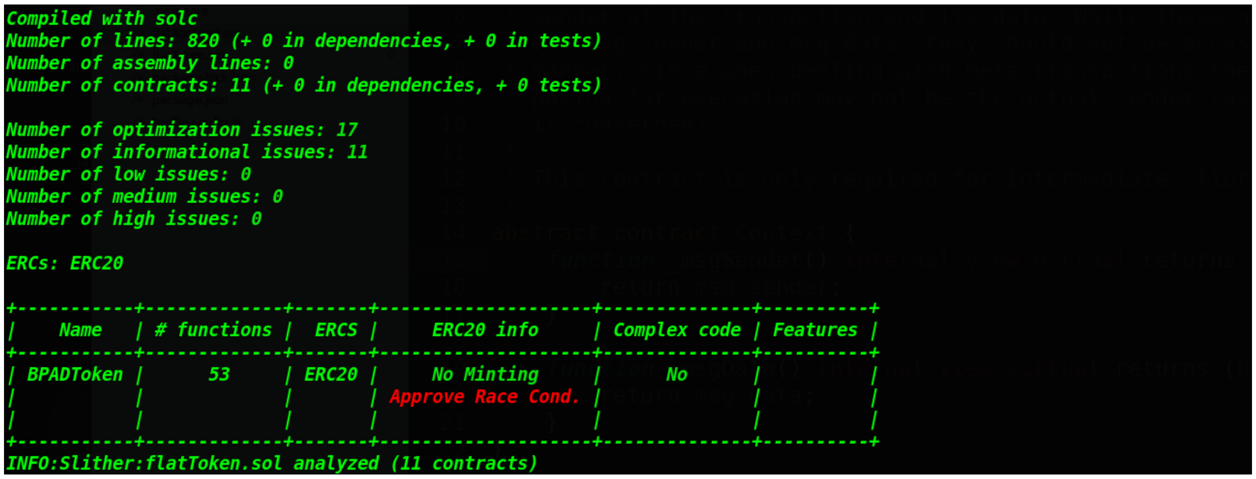
Amended (November 23rd 2021): Issue was fixed by the **BlockPad** team and is no longer present in commit dd814448eb8885e9ea0ccdfde78748bfd1711375

Recommendations

--

Automated Audit Result

1. BPADToken.sol



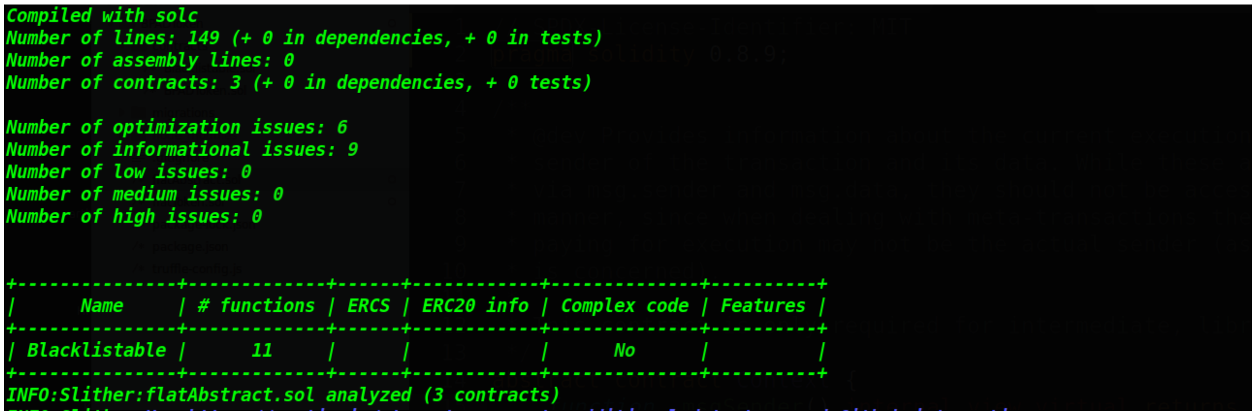
```
Compiled with solc
Number of lines: 820 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 11 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 17
Number of informational issues: 11
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 |
+-----------+-------------+-------+--------------------+--------------+----------+
| BPADToken |      53     | ERC20 |     No Minting     |      No      |          |
|           |             |       | Approve Race Cond. |              |          |
|           |             |       |                    |              |          |
+-----------+-------------+-------+--------------------+--------------+----------+
INFO:Slither:flatToken.sol analyzed (11 contracts)
```

2. Blacklistable.sol



```
Compiled with solc
Number of lines: 149 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 3 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 6
Number of informational issues: 9
Number of low issues: 0
Number of medium issues: 0
Number of high issues: 0


+---------------+-------------+------+------------+--------------+----------+
|      Name     | # functions | ERCS | ERC20 info | Complex code | Features |
+---------------+-------------+------+------------+--------------+----------+
| Blacklistable |      11     |      |            |      No      |          |
+---------------+-------------+------+------------+--------------+----------+
INFO:Slither:flatAbstract.sol analyzed (3 contracts)
```

Concluding Remarks

While conducting the audits of the BlockPad smart contract, it was observed that the contracts contain Medium and Low severity issues.

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

- ***Note - 23rd November 2021***
    - ***The BlockPad team has fixed the issues based on the auditor's recommendation.***

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 BlockPad 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