Throne

ERC20

Smart Contract Audit Report

IMMUNEBYTES

Audits

September 3, 2021

Introduction

1. About Throne

Throne has developed an exciting and disruptive blockchain technology that will transform our relationship to content and empower content creators like never before.

An NFT marketplace for the next generation of creators and collectors.

Focusing on the defining NFT revolution to bring everyday users on-chain to transact NFTs and build a trusted collection. The importance of technology and contemporary culture is reflected in everything we do, and we have clearly set ourselves apart as a vibrant force in the market because of it.

Visit https://www.thr.one/ 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 75+ 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 Throne team has provided the following doc for the purpose of audit:

1. https://www.thr.one/about

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: Throne
- Contracts Name: ThroneERC20.sol
- Mainnet Address: https://etherscan.io/token/0x2e95cea14dd384429eb3c4331b776c4cfbb6fcd9
- Languages: Solidity(Smart contract)
- Github commit for initial audit: 4cc2879ee4dfe465202515df1cb1bb7330c365d2
- Platforms and Tools: Remix IDE, Truffle, Ganache, Solhint, 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 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	-	-	-

High Severity Issues

No issues were found.

Medium severity issues

No issues were found.

Low Severity Issues

1. **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 must be marked as **external** within the contract:
   - **mint()** function at Line 39
   - **pause()** function at Line 53
   - **unpause()** function at Line 67

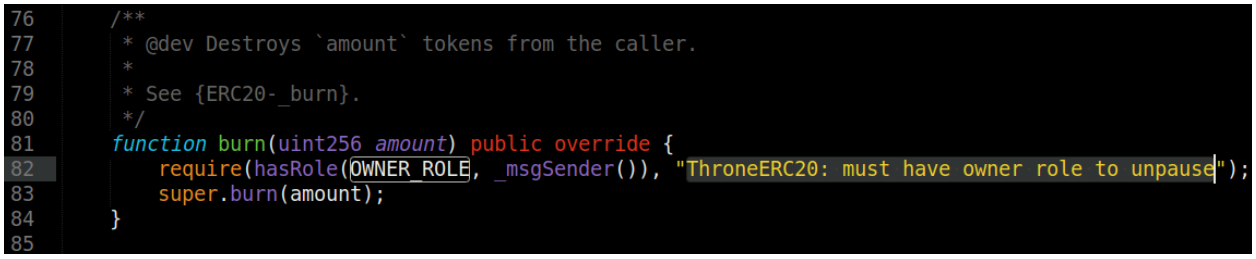
   Recommendation:
   If the PUBLIC visibility of the above-mentioned functions is not intended, then the EXTERNAL Visibility keyword should be preferred.

2. **Invalid Error Messages found in Require Statements**
   Line no - 82, 98

   Explanation:
   The **ThroneERC20** contract includes a few require statements with misleading error messages in require statements.

   The following functions, for instance, include error messages of **Unpause** while the functionality actually represents a **Burn** feature:
   a. **burn()** function at Line 82

b. burnFrom() function at Line 98



```
    /**
     * @dev Destroys `amount` tokens from the caller.
     *
     * See {ERC20-_burn}.
     */
    function burn(uint256 amount) public override {
        require(hasRole(OWNER_ROLE, _msgSender()), "ThroneERC20: must have owner role to unpause");
        super.burn(amount);
    }
```

While this badly affects the readability of the code, it also leads to a scenario where wrong information is provided to the user during a function revert.

Recommendation:
It is recommended to provide adequate error messages in require statement to avoid the above-mentioned scenarios.

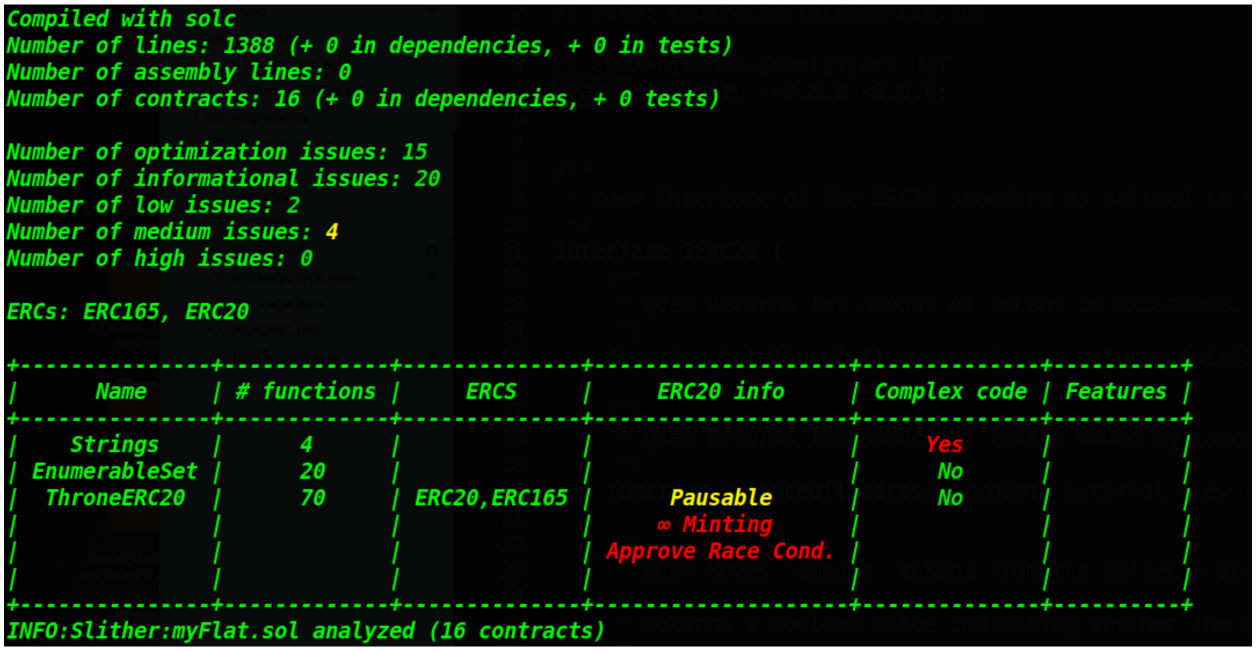
Recommendations

1. **Test cases can be improved**

Explanation:
Test cases related to the access control and other imperative functionalities of the contract were not found in the current test files.

Recommendation:
It's recommended to write extensive test cases to ensure the contract executes as per expectation.

Automated Test Results



```
Compiled with solc
Number of lines: 1388 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 16 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 15
Number of informational issues: 20
Number of low issues: 2
Number of medium issues: 4
Number of high issues: 0

ERCs: ERC165, ERC20

+---------------+-------------+-------------+--------------------+--------------+----------+
|      Name     | # functions |     ERCS    |     ERC20 info     | Complex code | Features |
+---------------+-------------+-------------+--------------------+--------------+----------+
|    Strings    |      4      |             |                    |     Yes      |          |
| EnumerableSet |      20     |             |                    |      No      |          |
|  ThroneERC20  |      70     | ERC20,ERC165 |     Pausable      |      No      |          |
|               |             |             |     ∞ Minting      |              |          |
|               |             |             | Approve Race Cond. |              |          |
|               |             |             |                    |              |          |
+---------------+-------------+-------------+--------------------+--------------+----------+
INFO:Slither:myFlat.sol analyzed (16 contracts)
```

Concluding Remarks

While conducting the audits of the Throne smart contract, it was observed that the contract contained only Low severity and a few areas of recommendations. No High or Medium severity are found.

Our auditors suggest that recommendations should be resolved by Throne 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 Throne 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