Forward Protocol

ForwardRootToken & ForwardProxy

Smart Contract Audit Report

November 23, 2021

Introduction

1. About Forward Protocol

Forward Protocol employs an easy-to-use WordPress-like model to facilitate a no-code environment that users can interact with even without technical knowledge.

Forward Protocol provides blockchain toolkits that connect the value-driven economy. The modular architecture and ready-to-deploy fully customizable smart contracts are designed for anyone to adopt blockchain technology, without any risk.

The protocol's modular design allows any organization to choose the module suited to their platform and modify it to be functional to their use case. Forward Protocol sets the stage for a value-driven economy, while the deploying organizations set the parameters.

Visit https://forwardprotocol.io/ 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 Forward Protocol team has provided the following doc for the purpose of audit:

1. https://whitepaper.forwardprotocol.io/

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: Forward Protocol
- Contracts Name: ForwardRootToken.sol, ForwardProxy.sol
- Languages: Solidity(Smart contract)
- Github commit/Smart Contract Address for audit: Null
- 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 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	-	1
Closed	-	-	-

Contract: FowardRootToken.sol

High Severity Issues

1. **refundTokens() function doesn't execute as intended**
   Line no - 37 to 39

   Explanation:
   The **refundTokens** function in the contract is actually intended to send any ERC20 token that is accidentally sent to the contract.

```
36
37      function refundTokens() external onlyOwner {
38          _transfer(address(this), owner(), balanceOf(address(this)));
39      }
```

   However, as per the current architecture of the function, the function initiates a transfer of **$FORWARD** token itself. It sets the **$FORWARD** token smart contract address as the sender of the tokens while the tokens being sent out to the owner aren't the accidentally sent tokens but the native token, i.e., the **$FORWARD** token itself.

   This symbolizes that the function doesn't even take into consideration the address of the accidentally forwarded ERC20 token but simply uses the address of the **$FORWARD** token itself for the refund, which is not intended behavior.

   Recommendation:
   It is highly recommended that the **refundTokens** function should be modified in a way that it effectively sends out the accidentally transferred token back to the owner and executes as expected.

2. **Contract locks Ether completely and fails to provide a way to withdraw Locked ether**

   Explanation:
   The **initialize()** function in the contract includes a **payable** keyword that allows ether to enter the contract through this function.

```
14
15      function initialize(address owner) external payable {
16          require(!initialized, "already initialized");
17
18          initializeERC20("Forward", "$FORWARD");
19          initializePausable();
20          initializeOwnable(owner);
21          initializeERC20Permit("Forward");
22
23          _mint(owner, 5000000000 * 1e18); // mint 5 bil $FORWARD tokens
24          initialized = true;
25      }
```

   However, during the audit procedure, no specific function to withdraw the locked ether was found in the contract. This will lead to an unexpected scenario where ether sent to the contract shall completely be lost.

   Recommendation:
   If the contract is supposed to receive ETHER, there should be functions that allow the withdrawal of the locked ether. Otherwise, the **payable** keyword shall be removed so that the contract doesn't receive any ether.

Medium Severity Issues

No issues were found.

Low Severity Issues

1. **Absence of Zero Address Validation**

   Explanation :
   The **ForwardRootToken** contract doesn't include any zero address validation check for the **owner** address argument being passed in the **initialize()** function of the contract.

   Recommendation:
   A **require** statement should be included in such functions to ensure no zero address is passed in the arguments.

Recommendations

1. **Excessive use of digits can be avoided**
   Line no - 26

   Explanation:
   The above-mentioned lines have a large number of digits that makes it difficult to review and reduces the readability of the code.

```
22
23          _mint(owner, 5000000000 * 1e18); // mint 5 bil $FORWARD tokens
24          initialized = true;
```

   Recommendation:
   Ether Suffix could be used to symbolize the 10^18 zeros.

Contract: ForwardProxy.sol

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

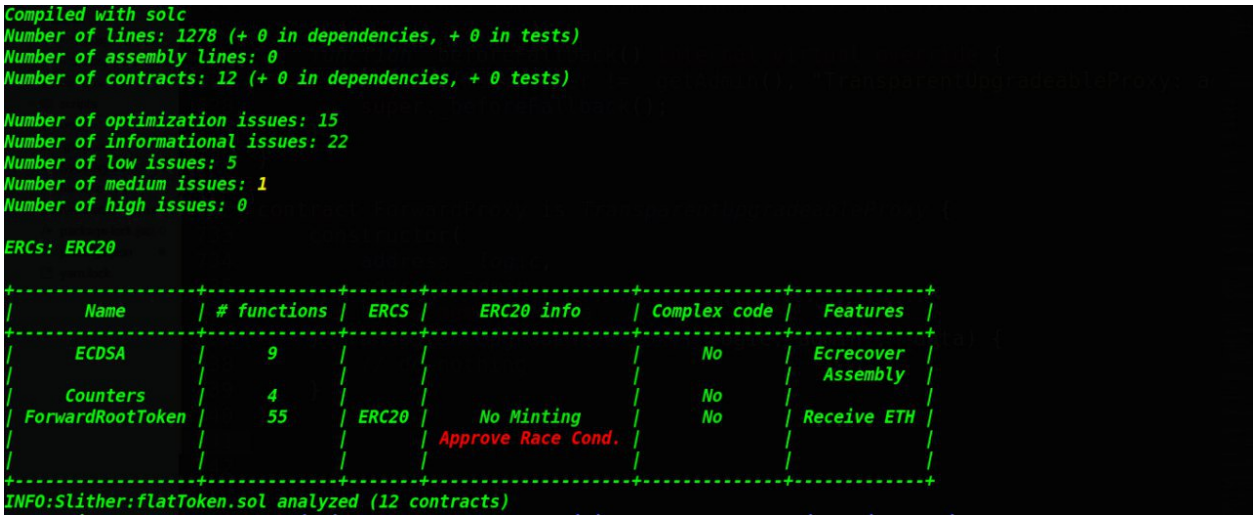
No issues were found.

Recommendations

--

Automated Audit Result

1. ForwardRootToken.sol



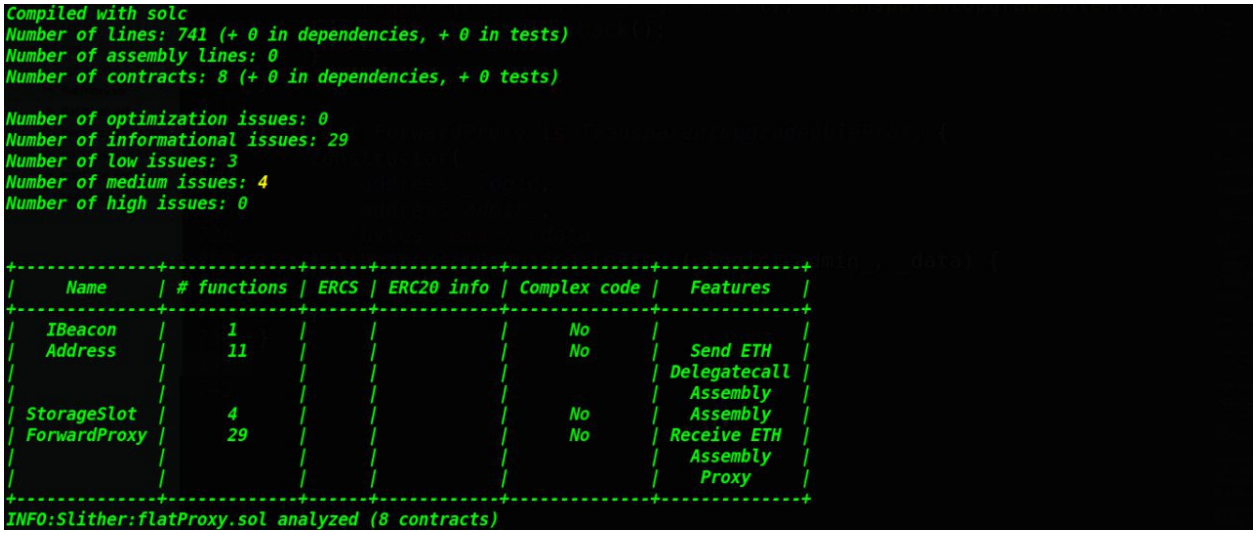
```
Compiled with solc
Number of lines: 1278 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 12 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 15
Number of informational issues: 22
Number of low issues: 5
Number of medium issues: 1
Number of high issues: 0

ERCs: ERC20

+------------------+-------------+-------+--------------------+--------------+--------------+
|       Name       | # functions | ERCS  |     ERC20 info     | Complex code |   Features   |
+------------------+-------------+-------+--------------------+--------------+--------------+
|      ECDSA       |      9      |       |                    |      No      |  Ecrecover   |
|                  |             |       |                    |              |   Assembly   |
|     Counters     |      4      |       |                    |      No      |              |
| ForwardRootToken |      55     | ERC20 |     No Minting     |      No      | Receive ETH  |
|                  |             |       | Approve Race Cond. |              |              |
|                  |             |       |                    |              |              |
+------------------+-------------+-------+--------------------+--------------+--------------+
INFO:Slither:flatToken.sol analyzed (12 contracts)
```

2. ForwardProxy.sol



```
Compiled with solc
Number of lines: 741 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 8 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 0
Number of informational issues: 29
Number of low issues: 3
Number of medium issues: 4
Number of high issues: 0


+--------------+-------------+------+------------+--------------+--------------+
|     Name     | # functions | ERCS | ERC20 info | Complex code |   Features   |
+--------------+-------------+------+------------+--------------+--------------+
|   IBeacon    |      1      |      |            |      No      |              |
|   Address    |      11     |      |            |      No      |   Send ETH   |
|              |             |      |            |              | Delegatecall |
|              |             |      |            |              |   Assembly   |
| StorageSlot  |      4      |      |            |      No      |   Assembly   |
| ForwardProxy |      29     |      |            |      No      | Receive ETH  |
|              |             |      |            |              |   Assembly   |
|              |             |      |            |              |    Proxy     |
+--------------+-------------+------+------------+--------------+--------------+
INFO:Slither:flatProxy.sol analyzed (8 contracts)
```

Concluding Remarks

While conducting the audits of the Forward Protocol smart contract, it was observed that the contracts contain High and Low severity issues.

Our auditors suggest that High and 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 Forward Protocol 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