DeHacker

Code Security Assessment

BlockchainFX

April 11th, 2025

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best prac tices. Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

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

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	BlockchainFX
Platform	Ethereum (ETH)
Website	https://lifebc.io
Type	DeSci
Language	Solidity
Codebase	0xd0d801eea2c2422df3e626b82ebbb618f4cc445e

Vulnerability Summary

Vulnerability Level	Total	Mitigated	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	3	1	0	1	0	1
Medium	3	0	0	0	0	3
Minor	1	0	0	0	0	1
Informational	1	0	0	0	0	1
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
BFX	BFX.sol	a3e7d8cc4d8c9e59acb986c306525885c29922267f94882b4cf2544e3460eea4
ERC	contracts/tokens/ERC20/ERC20Template.sol	075287d92e6a2e5a1ac252d34825734a6039e4251fd55e4f29c432e5aaae7c9f
SEC	contracts/utils/Secured.sol	4fc08f5aa39677f1c6806153ef6c344bfcef25f084807bc56a5eeda3cf093ee4
SHA	contracts/utils/Shallowed.sol	1668b7dc1505476e1c401173f1bf74cfa4267cfb33eba867d45eecefc8008f7b

Findings

ID	Title	Severity	Status
0XD-01	Centralization Related Risks	Major	Mitigated
ERC-04	Initial Token Distribution	Major	Acknowledged
ERC-05	Inconsistent Bot And Whale Registration In_update() Function	Major	Resolved
ERC-06	Pause And Unpause Functions Not Exposed	Medium	Resolved
ERC-07	setTaxAddress() Does Not RemovePrevious taxAddress From noTaxable List	Medium	Resolved
UTI-01	ERC20Template Contract Not Active InantiBot And antiWhale Contracts	Medium	Resolved
UTI-02	Third-Party Dependencies	Minor	Resolved
ERC-08	Potential Anti-Whale Check Inaccuracy InERC20Template Contract	Informational	Resolved

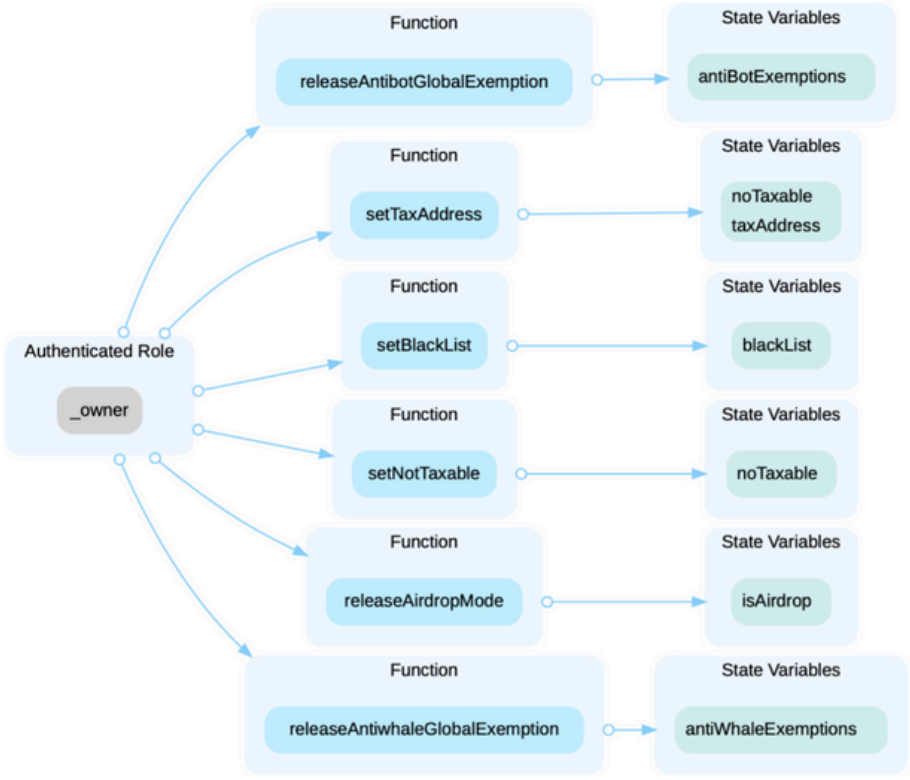
MAJOR

0XD-01 | Centralization Related Risks

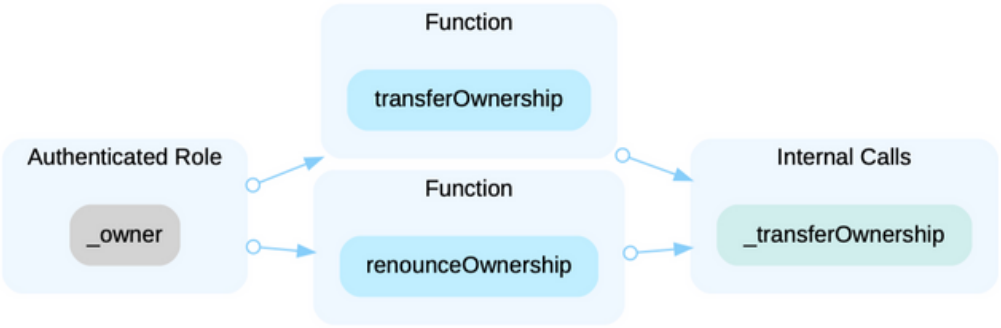
Issue	Severity	Location	Status
Centralization	Major	@openzeppelin/contracts/access/Ownable.sol (pre): 76,84; contracts/tokens/ERC20/ERC20Template.sol (pre): 109, 115, 123, 129, 134, 139	Mitigated

Description

In the contract ERC20Template , the role _owner has authority over the functions shown in the diagram below. Any compromise to the _owner account may allow the hacker to take advantage of this authority and release antibot global exemption, set the tax address and mark as non-taxable, set blacklist status for an address, set address taxable status, disable airdrop mode, and remove global antiwhale exemption.



In the contract Ownable , the role _owner has authority over the functions shown in the diagram below. Any compromise tothe _owner account may allow the hacker to take advantage of this authority and transfer or renounce contract ownership.



Recommendation

It is recommended that the team be transparent regarding the initial token distribution process. The token distribution planshould be published in a public location that the community can access. The team should make efforts to restrict access tothe private keys of the deployer account or EOAs. A multi-signature (⅔, ⅗) wallet can be used to prevent a single point offailure due to a private key compromise. Additionally, the team can lock up a portion of tokens, release them with a vestingschedule for long-term success, and deanonymize the project team with a third-party KYC provider to create greateraccountability.

Short Term:
Timelock and Multi sign (⅔, ⅗) combination mitigate by delaying the sensitive operation and avoiding a single point of keymanagement 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:
Time lock 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.

MAJOR

ERC-04 | Initial Token Distribution

Issue	Severity	Location	Status
Centralization	Major	BFX.sol (final): 175	Acknowledged

Description

All of the BFX tokens are sent to the deployer address. This is a centralization risk because the address can distribute tokenswithout obtaining the consensus of the community. Any compromise to the address may allow a hacker to steal and sell tokens on the market, resulting in severe damage to the project.

Recommendation

It is recommended that the team be transparent regarding the initial token distribution process. The token distribution planshould be published in a public location that the community can access. The team should make efforts to restrict access tothe private keys of the deployer account or EOAs. A multi-signature (⅔, ⅗) wallet can be used to prevent a single point offailure due to a private key compromise. Additionally, the team can lock up a portion of tokens, release them with a vesting schedule for long-term success, and deanonymize the project team with a third-party KYC provider to create greater accountability.

MAJOR

ERC-05 | Inconsistent Bot And Whale Registration In _update() Function

Issue	Severity	Location	Status
Logical Issue	Major	contracts/tokens/ERC20/ERC20Template.sol (pre): 166	Resolved

Description

The _update() function is designed to enforce "anti-bot" and "anti-whale" checks before updating token balances. However, the function misplaces the bot and whale tracking logic, leading to inconsistent behavior:

- noBots(sender) → Ensures sender is not a bot.
- noWhales(recipient, amount) → Ensures recipient is not a whale.
- Inconsistency in tracking:
  - registerBlock(recipient); → Registers the recipient for future bot checks.
  - registerBlockTimeStamp(sender); → Registers the sender for future whale checks.
  - However, bot checks should apply to senders, and whale checks should apply to recipients.

It is important to note that antiBot and antiWhale contracts are treated as black boxes since they are not in the audit scope.

```
153     function _update(
154         address sender,
155         address recipient,
156         uint256 amount
157     )
158         internal
159         virtual
160         override
161         whenNotPaused
162         noBots(sender)
163         noWhales(recipient, amount)
164     {
165         registerBlock(recipient);
166         registerBlockTimeStamp(sender);
167         if (isAirdrop) {
168             if (!noTaxable[sender]) revert("Airdrop mode is enabled");
169         }
170         /// @dev the tx is charged based on the sender
171         if (blackList[sender]) revert BlacklistedAddress(sender);
172         if (blackList[recipient]) revert BlacklistedAddress(recipient);
173         uint tax = 0;
174         if (!noTaxable[sender]) {
175             tax = (amount / 100) * taxFee; // % tax
176             super._update(sender, taxAddress, tax);
177         }
178         super._update(sender, recipient, amount - tax);
179     }
```

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

import {Test, console} from "forge-std/Test.sol";
import {ERC20Template} from "../src/tokens/ERC20/ERC20Template.sol";
import {IERC20AntiBot} from "../src/interfaces/services/IERC20AntiBot.sol";

contract BFXTest is Test {
    ERC20Template public bfx;
    IERC20AntiBot public antiBot;
    address user = vm.addr(1);

    function setUp() public {
        vm.createSelectFork(vm.envString("ETH_MAINNET"));
        bfx = ERC20Template(0xD0d801eEa2c2422dF3e626b82EBBb618f4Cc445e);
        antiBot = IERC20AntiBot(bfx.antiBot());
    }

    function testIssue() public {
        // activate bfx in antiBot
        require(antiBot.isActive(address(bfx)) == false, "error0");
        vm.startPrank(0xb3B4A563A1679E43b24b781b9232e7408Db64aB8);
        antiBot.setCanUseAntiBot{value: 0.005 ether}
(0xe75ab27fb16f03b1754b0a9d28b83820171f94916fdf7433955d9a16380d0fca, address(bfx));
        antiBot.setActive(address(bfx), true);
        vm.stopPrank();
        require(antiBot.isActive(address(bfx)) == true, "error1");

        // increase "bot"'s BFX balance by 1 ether
        address bot = vm.addr(1);
        deal(address(bfx), bot, 1 ether);

        address user1 = vm.addr(2);
        address user2 = vm.addr(3);

        // bot can send tokens twice in the same block number
        vm.startPrank(bot);
        bfx.transfer(user1, 1000);
        bfx.transfer(user2, 1000);
        vm.stopPrank();
    }
}
```

Recommendation

Correct the registration logic:

- Register senders for bot checks (to prevent bots from sending tokens).
- Register recipients for whale checks (to prevent large token holdings).

MEDIUM

ERC-06 | Pause And Unpause Functions Not Exposed

Issue	Severity	Location	Status
Design Issue	Medium	contracts/tokens/ERC20/ERC20Template.sol (pre): 50	Resolved

Description

The contract ERC20Template extends the Pausable contract and contains a function with whenNotPaused modifier.However, there is no public or external pause() and unpause() functions that extend the internal _pause() and_unpause() functions, and as a result, the contract is always unpaused.

Recommendation

We recommend considering exposing the internal _pause() and _unpause() functions to the public or external withproper access control mechanisms.

MEDIUM

ERC-07 | setTaxAddress() Does Not Remove Previous taxAddress From noTaxable List

Issue	Severity	Location	Status
Logical Issue	Medium	contracts/tokens/ERC20/ERC20Template.sol (pre): 50	Resolved

Description

The setTaxAddress() function allows the owner to update the taxAddress and marks the new address as noTaxable. However, it **does not remove the previous** taxAddress from the noTaxable list, which could lead to **unintended tax exemptions** for multiple addresses.

```
123     function setTaxAddress(address _taxAddress) external onlyOwner {
124         taxAddress = _taxAddress;
125         noTaxable[_taxAddress] = true;
126     }
```

Recommendation

Removed previous taxAddress from noTaxable

MEDIUM

UTI-01 | ETC20Templete Contract Not Active In antiBot And antiWhale Contracts

Issue	Severity	Location	Status
Design Issue	Medium	contracts/utils/Secured.sol (pre): 16; contracts/utils/Shallowed.sol (pre): 17	Resolved

Description

The ERC20Template contract integrates with antiBot and antiWhale contracts to enforce anti-bot and anti-whale protections. However, as of March 12, 2025, the ERC20Template contract is not active in either of these contracts. As a result, the intended protections against bots and large transactions are not enabled.Additionally, it is important to note that the antiBot and antiWhale contracts are outside the audit scope.

Recommendation

Verify and activate the ERC20Template in the antiBot and antiWhale contracts.

MINOR

UTI-02 | Third-Party Dependencies

Issue	Severity	Location	Status
Volatile Code	Minor	contracts/utils/Secured.sol (pre): 13; contracts/utils/Shallowed.sol (pre):17	Resolved

Description

The contract is serving as the underlying entity to interact with out-of-scope antiBot and antiWhale contracts. The scopeof the audit treats out-of-scope entities as black boxes and assumes their functional correctness. However, in the real world,these contracts can be compromised and this may lead to lost or stolen assets.

Recommendation

We recommend that the project team constantly monitor the functionality of these contracts to mitigate any side effects thatmay occur when unexpected changes are introduced.

INFORMATIONAL

ERC-08 | Potential Anti-Whale Check Inaccuracy In ERC20Templete Contract

Issue	Severity	Location	Status
Design Code	Informational	contracts/tokens/ERC20/ERC20Template.sol (pre): 163	Resolved

Description

The _update() function in the ERC20Template contract enforces an anti-whale mechanism through the noWhales(recipient, amount) modifier. However, the check applies to the deducted sender amount, rather than the actual received amount by the recipient.Since the taxFee is currently set to zero in contract 0xd0d801eea2c2422df3e626b82ebbb618f4cc445e , this issue does not cause any direct impact in the current deployment. However, if the contract is used elsewhere with a non-zero taxFee , the check could become inaccurate. It is important to note that the antiWhale contract is treated as a black box since it is not in the audit scope.

Recommendation

We recommend that the project team constantly monitor the functionality of these contracts to mitigate any side effects thatmay occur when unexpected changes are introduced.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Appendix

Finding Categories

Centralization / Privilege
Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Coding Style
Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code
Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Logical Issue
Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block. timestamp works.

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.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

Ethereum
Cosmos
Eos
Substrate

TECH STACK

Python
Solidity
Rust
c++

CONTACTS

https://dehacker.io
https://twitter.com/dehackerio
https://github.com/dehacker/audits_public
https://t.me/dehackerio
https://blog.dehacker.io/

DeHacker

April 11th 2025