Approved

SMART CONTRACT SECURITY AUDIT

FRGX Token

Scan and check this report
was posted at Approved Github

December, 2023

Website: Approved.ltd

Table of Contents

Disclaimer

This is a standard report based on our automated and manual examination of cybersecurity vulnerabilities and framework flaws of the project's smart contract.

Reading the full analysis report is essential to build your understanding of project's security level. It is crucial to take note, though we have done our best to perform this analysis and report, that you should not rely on the our research and cannot claim what it states or how we created it.

Before making any judgments, you have to conduct your own independent research.

We will discuss this in more depth in the following disclaimer - please read it fully.

DISCLAIMER: You agree to the terms of this disclaimer by reading this report or any portion thereof. Please stop reading this report and remove and delete any copies of this report that you download and/or print if you do not agree to these conditions. Scan and verify report's presence in the GitHub repository by a qr-code at the title page. This report is for non-reliability information only and does not represent investment advice. No one shall be entitled to depend on the report or its contents, and Approved and its affiliates shall not be held responsible to you or anyone else, nor shall Approved provide any guarantee or representation to any person with regard to the accuracy or integrity of the report.

Without any terms, warranties or other conditions other than as set forth in that exclusion and Approved excludes hereby all representations, warrants, conditions and other terms (including, without limitation, guarantees implied by the law of satisfactory quality, fitness for purposes and the use of reasonable care and skills).

The report is provided as "as is" and does not contain any terms and conditions. Except as legally banned, Approved disclaims all responsibility and responsibilities and no claim against Approved is made to any amount or type of loss or damages (without limitation, direct, indirect, special, punitive, consequential or pure economic loses or losses) that may be caused by you or any other person, or any damages or damages, including without limitations (whether innocent or negligent).

Security analysis is based only on the smart contracts. No applications or operations were reviewed for security. No product code has been reviewed.

Procedure

Our analysis contains following steps:

1. Project Analysis;

2. Unit Testing:
   - Smart contract functions will be unit tested on multiple parameters and under multiple conditions to ensure that all paths of functions are functioning as intended.
   - In this phase intended behaviour of smart contract is verified.
   - In this phase, we would also ensure that smart contract functions are not consuming unnecessary gas.
   - Gas limits of functions will be verified in this stage.

3. Automated Testing:
   - Mythril
   - Oyente
   - Manticore
   - Solgraph

Terminology

We categorize the finding into 4 categories based on their vulnerability:

- Low-severity issue — less important, must be analyzed
- Medium-severity issue — important, needs to be analyzed and fixed
- High-severity issue —important, might cause vulnerabilities, must be analyzed and fixed
- Critical-severity issue —serious bug causes, must be analyzed and fixed.

Limitations

The security audit of Smart Contract cannot cover all vulnerabilities. Even if no vulnerabilities are detected in the audit, there is no guarantee that future smart contracts are safe. Smart contracts are in most cases safeguarded against specific sorts of attacks. In order to find as many flaws as possible, we carried out a comprehensive smart contract audit. Audit is a document that is not legally binding and guarantees nothing.

Basic Security Recommendation

Unlike hardware and paper wallets, hot wallets are connected to the internet and store private keys online, which exposes them to greater risk. If a company or an individual holds significant amounts of cryptocurrency in a hot wallet, they should consider using MultiSig addresses. Wallet security is enhanced when private keys are stored in different locations and are not controlled by a single entity.

Token Audit Details for 18.12.2023

Project Name: **FRGX Token**

Language: **Solidity**

Compiler Version: **v0.8.18**

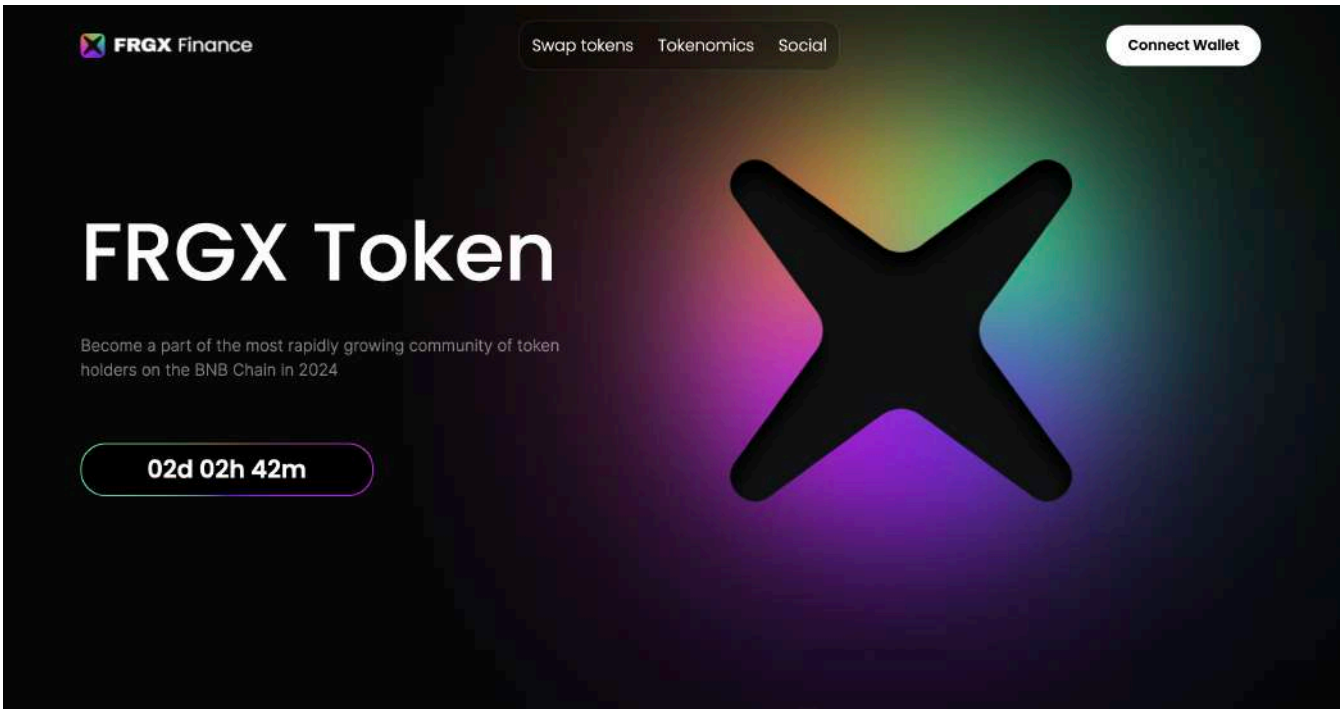
Social Profiles

Project Website: **https://frgx.finance/**

Project Twitter: **https://twitter.com/FRGXfinance**

Project Telegram: **https://t.me/frgxfinance**

Project Website Overview



✓JavaScript errors hasn’t been found.
✓Malware pop-up windows hasn’t been detected.
✓No issues with loading elements, code, or stylesheets.

Vulnerabilities checking

Issue Description	Checking Status
Compiler Errors	Completed
Delays in Data Delivery	Completed
Re-entrancy	Completed
Transaction-Ordering Dependence	Completed
Timestamp Dependence	Completed
Shadowing State Variables	Completed
DoS with Failed Call	Completed
DoS with Block Gas Limit	Completed
Outdated Complier Version	Completed
Assert Violation	Completed
Use of Deprecated Solidity Functions	Completed
Integer Overflow and Underflow	Completed
Function Default Visibility	Completed
Malicious Event Log	Completed
Math Accuracy	Completed
Design Logic	Completed
Fallback Function Security	Completed
Cross-function Race Conditions	Completed
Safe Zeppelin Module	Completed

Security Issues

1) Unchecked transfer:

Check: unchecked transfer

Severity: High

Confidence: **Medium**

```
function withdraw(IERC20 token↑, address to↑, uint amount↑) public {
    require(msg.sender == owner, "TokenStorage: onlyOwner");

    token↑.transfer(to↑, amount↑);
}
```

```
IERC20(tokenToSwap).transfer(poolAddress, IERC20(tokenToSwap).balanceOf(address(this)));
_transfer(address(this), poolAddress, amount↑/2);
```

```
function _updateLiquidityPool(address callerPoolAddress↑, uint amount↑) private {
    address poolAddress = pankakeFactory.getPair(wBNB, address(this));
    address tokenToSwap = wBNB;
    if(poolAddress == callerPoolAddress↑) {
        poolAddress = pankakeFactory.getPair(busd, address(this));
        tokenToSwap = busd;
    }

    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = tokenToSwap;

    pancakeV2Router.swapExactTokensForTokens(
        amount↑ / 2,
        1,
        path,
        address(tokenStorage),
        block.timestamp + 100
    );

    tokenStorage.withdraw(IERC20(tokenToSwap), address(this), IERC20(tokenToSwap).balanceOf(address(tokenStorage)));

    IERC20(tokenToSwap).transfer(poolAddress, IERC20(tokenToSwap).balanceOf(address(this)));
    _transfer(address(this), poolAddress, amount↑/2);

    IV2Pool(poolAddress).mint(address(this));

    // lock liquidity
    IERC20(poolAddress).transfer(locker, IERC20(poolAddress).balanceOf(address(this)));
}
```

L844-848, L1204, L1182-1211

Description:

The return value of an external transfer/transferFrom call is not checked.

Recommendation:

Use safeERC20, or ensure the transfer/transferFrom return value is checked.

2) Divide before multiply:

Check: divide-before-multiply

Severity: Medium

Confidence: **Medium**

```
if(hasFId(to)) {
    if(!accountsWithFId[to]) {
        totalShares+= super.balanceOf(to);
        accountsWithFId[to] = true;
        rewardDebt[to] = pointsPerShare * super.balanceOf(to) / 1e18;
    } else {
        totalShares+= amount;
    }
}
```

```
    totalReward+= amount*3/100;
    if(totalShares > 0) {
        pointsPerShare+= amount*3/100 * 1e18 / totalShares;
    }
```

L1084, L1017

Description:

Solidity's integer division truncates. Thus, performing division before multiplication can lead to precision loss.

Recommendation:

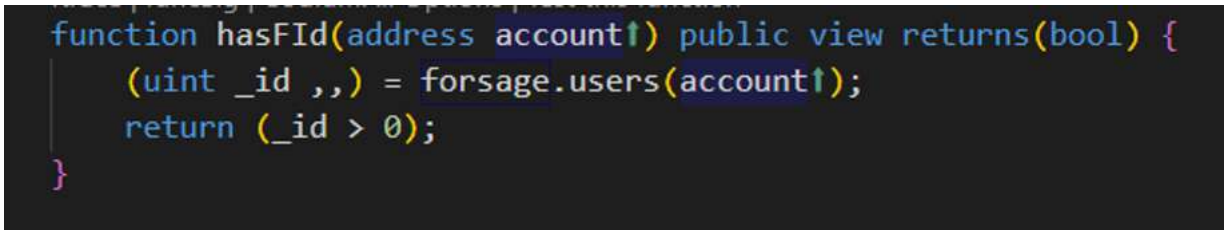
Consider ordering multiplication before division.

3) Unused return:

Check: unused-return

Severity: Medium

Confidence: **Medium**



```
function hasFId(address account) public view returns(bool) {
    (uint _id ,,) = forsage.users(account);
    return (_id > 0);
}
```

```
function _refReward(address account1, uint256 amount1) private {
    address id1 = forsage.id1();

    if(account1 == id1) {
        // _transfer(address(this), account, amount * 15 / 1000);
        _sendRefRewards(account1, account1, amount1 * 15 / 1000);
        // _transfer(address(this), account, amount * 10 / 1000);
        _sendRefRewards(account1, account1, amount1 * 10 / 1000);
        // _transfer(address(this), account, amount * 5 / 1000);
        _sendRefRewards(account1, account1, amount1 * 5 / 1000);
        return;
    }

    (,address referrerAddress, ) = forsage.users(account1);
    // _transfer(address(this), referrerAddress, amount * 15 / 1000);
    _sendRefRewards(account1, referrerAddress, amount1 * 15 / 1000);

    if(referrerAddress == id1) {
        // _transfer(address(this), referrerAddress, amount * 10 / 1000);
        _sendRefRewards(account1, referrerAddress, amount1 * 10 / 1000);
        // _transfer(address(this), referrerAddress, amount * 5 / 1000);
        _sendRefRewards(account1, referrerAddress, amount1 * 5 / 1000);
        return;
    }

    (,referrerAddress, ) = forsage.users(referrerAddress);
    // _transfer(address(this), referrerAddress, amount * 10 / 1000);
    _sendRefRewards(account1, referrerAddress, amount1 * 10 / 1000);

    if(referrerAddress == id1) {
        // _transfer(address(this), referrerAddress, amount * 5 / 1000);
        _sendRefRewards(account1, referrerAddress, amount1 * 5 / 1000);
        return;
    }

    (,referrerAddress, ) = forsage.users(referrerAddress);
    // _transfer(address(this), referrerAddress, amount * 5 / 1000);
    _sendRefRewards(account1, referrerAddress, amount1 * 5 / 1000);
}
```

L964-967, L1137-1175

Description:

The return value of an external call is not stored in a local or state variable.

Recommendation:

Ensure that all the return values of the function calls are used.

4) Missing zero address validation:

Check: missing-zero-check

Severity: Low

Confidence: **Medium**

```
function update(address newImpl) public onlyOwner {
    impl = newImpl;
}
```

L-885

Description:

Detect missing zero address validation.

Recommendation:

Check that the address is not zero.

5) Assembly usage:

Configuration

Check: assembly

Severity: Low

Confidence: **High**

```
function _delegate(address implementation) internal {
    // solhint-disable-next-line no-inline-assembly
    assembly {
        // Copy msg.data. We take full control of memory in this inline assembly
        // block because it will not return to Solidity code. We overwrite the
        // Solidity scratch pad at memory position 0.
        calldatacopy(0, 0, calldatasize())

        // Call the implementation.
        // out and outsize are 0 because we don't know the size yet.
        let result := delegatecall(gas(), implementation, 0, calldatasize(), 0, 0)

        // Copy the returned data.
        returndatacopy(0, 0, returndatasize())

        switch result
        // delegatecall returns 0 on error.
        case 0 { revert(0, returndatasize()) }
        default { return(0, returndatasize()) }
    }
}
```

L732-752

Description:

The use of assembly is error-prone and should be avoided.

Recommendation:

Do not use evm assembly.

6) Dead-code:

Check: dead-code

Severity: Low

Confidence: **Medium**

```
function _contextSuffixLength() internal view virtual returns (uint256) {
    return 0;
}
```

```
function _transfer(address from, address to, uint256 value) internal virtual {
    if (from == address(0)) {
        revert ERC20InvalidSender(address(0));
    }
    if (to == address(0)) {
        revert ERC20InvalidReceiver(address(0));
    }
    _update(from, to, value);
}
```

```
function update(uint storedAmount) internal {
    totalReward+= storedAmount;
    if(totalShares > 0) {
        pointsPerShare+= storedAmount * 1e18 / totalShares;
    }
}
```

L-33-35, L569-577, L1224-1229

Description:

Functions that are not used.

Recommendation:

Remove unused functions.

7) State variables that could be declared constant:

Check: immutable-states

Severity: Low

Confidence: **High**

```
address public owner;
```

L-838

Description:

State variables that are not updated following deployment should be declared immutable to save gas.

Recommendation:

Add the immutable attribute to state variables that never change or are set only in the constructor.

Conclusion for project owner

High, Medium, and Low-severity issues exist within smart contracts.

NOTE: Please check the disclaimer above and note, that the audit makes no statements or warranties on the business model, investment attractiveness, or code sustainability. Contract security report for community

Approved Contact Info

Website: https://approved.ltd

Telegram: @team_approved

GitHub: https://github.com/Approved-Audits/smart_contracts