A picture containing vessel

Description automatically generated

it@blockstars.com.au

**BLOCKSTARS TECHNOLOGY**

17/09/2021

Smart Contracts Code Review and Security Analysis Report

Handle-forex

**Table of Contents**

[Introduction 2](#_Toc82687089)

[Project Scope and Specifications 4](#_Toc82687090)

[Analysis Result and Suggestions 5](#_Toc82687091)

[Smart Contract: TGE.sol 5](#_Toc82687092)

[Smart Contract: ForexVesting.sol 7](#_Toc82687093)

[Smart Contract: Forex.sol 7](#_Toc82687094)

[Smart Contract: Signature.sol 9](#_Toc82687095)

[Executive Summary 10](#_Toc82687096)

# Introduction

This report contains confidential information of audit summary of Forex tokens related smart contract programs running in Handle.fi project. It analyses security vulnerabilities, smart contract best practices, and possible attacks, using popular analysis tools and linters. We outlined our systematic approach to evaluate potential security issues in core smart contracts in Handle-forex project and provide additional suggestions for improvement.

**The basic information of Handle.fi Forex Tokens project**

|  |  |
| --- | --- |
| **Item** | **Description** |
| **Issuer** | Handle.fi |
| **Website** | <https://handle.fi/> |
| **Source** | Smart contract programs (Handle-forex) |
| **Language** | Solidity |
| **Blockchain** | Ethereum |
| **Git Repository** | <https://github.com/handle-fi/handle-forex> |
| **Audit method** | Static analysis using symbolic execution tools |

# Project Scope and Specifications

Scope of this project is to identify smart contract vulnerabilities to improve the coding practice in Handle forex token related smart contracts implemented for Handle.fi project. We classified the security vulnerabilities of smart contracts in three categories according to their impact level, such as critical, medium, and low class of vulnerabilities. Moreover, we used the impact level definitions from the recent smart contract security analysis research works [[ref1](https://arxiv.org/pdf/1905.01467.pdf), [ref2](https://arxiv.org/pdf/1908.08605.pdf)]. and assigned proper impact values for the identified smart contract vulnerabilities in Handle.fi Forex tokens project.

**IP1**: It raises critical behaviours and attackers can make benefit by using this vulnerability.

**IP2**: It raises critical behaviours and attackers cannot make benefit using this vulnerability.

**IP3**: It raises critical behaviours and attackers cannot trigger them externally (If they trigger, they cannot make benefit.

**IP4**: Contract works normally, and it leads to potential risks of errors when external programs call the contract.

**IP5**: It works normally, and it will not lead risks for external callers. But there is no re-usability, and it leads to gas wastage.

|  |  |
| --- | --- |
| **Impact Levels** | **Severity** |
| IP1 | High Critical |
| IP2 | Moderate Critical |
| IP3 | Average Critical |
| IP4 | Less Critical |
| IP5 | Normal Risk |

# Analysis Result and Suggestions

We used different tools and linters to analyse given smart contracts from Handle.fi Forex tokens project. The major tools we used to analyse smart contracts are Mythx (Consensys), and Solhint linter. The platforms we integrated to test the contracts are Remix, Visual Studio Code, Truffle, and Openzepplin Test Environment.

Smart Contract: TGE.sol

##### Finding 1

**Issue: Unchecked returned value from low-level external call**

**Severity**: **IP3 (Average Critical)**

**Contract: TGE.sol**

**Function name:** call()

**Description**:

* Low-level external calls return a boolean value. If the callee halts with an exception, 'false' is returned and execution continues in the caller. The caller should check whether an exception happened and react accordingly to avoid unexpected behavior.
* For example it is often desirable to wrap low-level external calls in require() so the transaction is reverted if the call fails.
* Eg: Return value check is recommended to use as below in Line 136

(bool success, )= msg.sender.call{value: currentDeposit + deposit - userCap}(""); require(success, “Failed to send”); // check return value before write logics

deposit = userCap - currentDeposit;

* Further it is recommended to use openzeppelin SafeMath utils for integer variables to avoid integer arithmetic issues.

**Exact Place of usage:**

msg.sender.call{value: currentDeposit + deposit - userCap}(""); - Line 136

msg.sender.call{value: ethDeposited + deposit - depositCap}(""); - Line 141

if (eth > 0) msg.sender.call{value: eth}(""); - Line 187

if (balance > 0) msg.sender.call{value: self.balance}(""); - Line 364

**Smart Contract Weakness Classification:** [SWC-104](https://swcregistry.io/docs/SWC-104)

##### Finding 2

**Issue: Dependence on predictable environment variable**

**Severity**: **IP4 (Less Critical)**

**Contract: TGE.sol**

**Variable name:** block.timestamp

**Description**:

* A control flow decision is made based on The block.timestamp environment variable. The block.timestamp environment variable is used to determine a control flow decision.
* Avoid using any of those environment variables and be aware that use of these variables introduces a certain level of trust into miners.
* But, It is more critical if you use the block.timestamp values in calculations related to funds.

**Exact place of usage:**

require(

\_generationStartDate > block.timestamp, Line - 83

"Start date must be in the future"

);

return block.timestamp >= generationStartDate; Line – 267

return block.timestamp > generationStartDate + generationDuration; Line – 274

return claimDate != 0 && block.timestamp >= claimDate; - Line – 281

claimDate = block.timestamp + 1; - Line – 377

**Smart Contract Weakness Classification:** [SWC-116](https://swcregistry.io/docs/SWC-116)

##### Finding 3

**Issue: A floating pragma is set**

**Severity**: **IP5 (Normal Risk)**

**Contract: TGE.sol**

**Description**:

* The current pragma Solidity directive is ""^0.8.0"".
* It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds.

Eg: pragma solidity 0.8.0;

* This is especially important if you rely on bytecode-level verification of the code.

**Exact place of usage:**

pragma solidity ^0.8.0; - Line 2

**Smart Contract Weakness Classification:** [SWC-103](https://swcregistry.io/docs/SWC-103)

Smart Contract: ForexVesting.sol

##### Finding 1

**Issue: Dependence on predictable environment variable**

**Severity**: **IP4 (Less Critical)**

**Contract: ForexVesting.sol**

**Variable name:** block.timestamp

**Description**:

* A control flow decision is made based on The block.timestamp environment variable. The block.timestamp environment variable is used to determine a control flow decision.
* Avoid using any of those environment variables and be aware that use of these variables introduces a certain level of trust into miners.
* It is more critical if you use the block.timestamp values in calculations related to funds

**Exact place of usage:**

require(

block.timestamp >= participant.lastClaimDate + minimumClaimDelay, - Line 84

"Must wait before next claim"

);

uint256 elapsed = block.timestamp - lastClaimDate; - Line 111

return \_balanceOf(account, block.timestamp); - Line 145

claimStartDate = block.timestamp + 1; - Line 227

return claimStartDate != 0 && block.timestamp >= claimStartDate; - Line 234

**Smart Contract Weakness Classification:** [SWC-116](https://swcregistry.io/docs/SWC-116)

##### Finding 2

**Issue: A floating pragma is set**

**Severity**: **IP5 (Normal Risk)**

**Contract: ForexVesting.sol**

**Description**:

* The current pragma Solidity directive is ""^0.8.0"".
* It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds.

Eg: pragma solidity 0.8.0;

* This is especially important if you rely on bytecode-level verification of the code.

**Exact place of usage:**

pragma solidity ^0.8.0; - Line 2

**Smart Contract Weakness Classification:** [SWC-103](https://swcregistry.io/docs/SWC-103)

Smart Contract: Forex.sol

##### Finding 1

**Issue: A floating pragma is set**

**Severity**: **IP5 (Normal Risk)**

**Contract: Forex.sol**

**Description**:

* The current pragma Solidity directive is ""^0.8.0"".
* It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds.

Eg: pragma solidity 0.8.0;

* This is especially important if you rely on bytecode-level verification of the code.

**Exact place of usage:**

pragma solidity ^0.8.0; - Line 2

**Smart Contract Weakness Classification:** [SWC-103](https://swcregistry.io/docs/SWC-103)

Smart Contract: Signature.sol

##### Finding 1

**Issue: A floating pragma is set**

**Severity**: **IP5 (Normal Risk)**

**Contract: Signature.sol**

**Description**:

* The current pragma Solidity directive is ""^0.8.0"".
* It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds.

Eg: pragma solidity 0.8.0;

**Exact place of usage:**

pragma solidity ^0.8.0; - Line 2

**Smart Contract Weakness Classification:** [SWC-103](https://swcregistry.io/docs/SWC-103)

# Executive Summary

|  |  |  |  |
| --- | --- | --- | --- |
| **Contracts** | Findings & SWC | **Status** | **Action** |
| TGE.sol | Unchecked returned value from low-level external call - [SWC-104](https://swcregistry.io/docs/SWC-104) | **IP3 (Average Critical)** | **Recommended to correct** |
| Dependence on predictable environment variable - [SWC-116](https://swcregistry.io/docs/SWC-116) | **IP4 (Less Critical)** | **Better to avoid the use of timestamp value. Can be ignored since the timestamp value is not used in fund related calculations.** |
| A floating pragma is set - [SWC-103](https://swcregistry.io/docs/SWC-103) | **IP5 (Normal Risk)** | **Can be ignored** |
| ForexVesting.sol | Dependence on predictable environment variable - [SWC-116](https://swcregistry.io/docs/SWC-116) | **IP4 (Less Critical)** | **Better to avoid the use of timestamp value. Can be ignored since the timestamp value is not used in fund related calculations.** |
| A floating pragma is set - [SWC-103](https://swcregistry.io/docs/SWC-103) | **IP5 (Normal Risk)** | **Can be ignored** |
| Forex.sol | A floating pragma is set - [SWC-103](https://swcregistry.io/docs/SWC-103) | **IP5 (Normal Risk)** | **Can be ignored** |
| Signature.sol | A floating pragma is set - [SWC-103](https://swcregistry.io/docs/SWC-103) | **IP5 (Normal Risk)** | **Can be ignored** |