GVISP 1
your space

2018.03.12.

COIN PAYMENT PROCESSOR.ORG

#OPEN CONSORTIUM cPRO

development@coinpaymentprocessor.org

GVISP1 LTD

#CP PROCESSOR GOLD PARTNER

office@gvisp.com

FOREX

ETHEREUM SMART CONTRACT AUDIT

GVISP1 LTD.

GVISP1 Ltd. **** Your space**** Bul.Vojvode Misica 17, 11000 Belgrade, Serbia, EU

1. Audit description

One contract was checked: **Forex.sol.**

The purpose of this audit is to check all functionalities of **Forex.sol** contract, and to determine level of security and probability of adverse outcomes.

Forex.sol is a contract that converts between stable tokens. It burns sent stable tokens and mints other ones at current price.

In the following lines, the contract is being referenced by the name of the file where it was written in. The file contains exactly one contract, so there is no room for confusion.

2. Quick review

Forex.sol is a contract that converts between Stable Tokens.

- ✓ Constructor parameters for **Forex.sol** are: **Main.sol** contract address and addresses of all stable tokens (DoTo, EuTo, KrTo, YeTo, YuTo).
- ✓ All functions and state variables are well commented which is good in order to understand quickly how everything is supposed to work.
- ✓ This contract must be deployed after **Main.sol** and all five StableToken.sol contracts as it takes their addresses as parameters.

3. A brief review of contract's functionalities

Contract contains many functions of which only `withdrawFee`, `makeAvailable` and `makeUnavailable` need (standard) authorization. Other functions can be called by anyone if requirements are satisfied.

- ✓ `makeAvailable` and `makeUnavailable` functions regulate which tokens are available for conversion inside this contract.
- ✓ `withdrawFee` function transfers fee (in defined stable tokens) to msg.sender's address. Only authorized addresses can call it.
- ✓ `exchange` is the contract's main function. When calling, one must specify the address of stable tokens he is sending and address of stable tokens he is buying, and the amount of first tokens.
- ✓ Fee equals 0.2% and is stored inside the contract.

4. Functionalities test

- ✓ makeAvailable: ✓
- ✓ makeUnavailable: ✓
- ✓ exchange: ✓
- ✓ withdrawFee: ✓

5. Detailed code check (line-by-line)

- ✓ After deployment, only Main.sol and each StableToken.sol contract addresses are set, which means all of these contracts must be deployed before Forex.sol.
- ✓ Forex.sol contract “reads authorizations” from Main.sol contract in order to secure some function are called only by authorized addresses. After deployment, “mint authorization” must be set for its address inside Main.sol so it could actually call `mint` function within StableToken.sol.

State variables of the contract:

- ➢ address **mainContractAddress** - address of Main.sol contract
- ➢ address **dotoAddress** - DoTo stable token address
- ➢ address **eutoAddress** - EuTo stable token address
- ➢ address **krtoAddress** - KrTo stable token address
- ➢ address **yetoAddress** - YeTo stable token address
- ➢ address **yutoAddress** - YuTo stable token address

Mappings:

- ➢ (address => uint256) fees - mapps each token address to fee expressed in that token
- ➢ (address => bool) availableForConversion - mapps each stable token address to “true” if token is available for forex conversion, to “false” otherwise

Modifiers:

- ➢ onlyAuthorized - checks if msg.sender is authorized

Function:

- ➢ `makeAvailable` - function that makes specific token available for conversion; can be called only by authorized addresses
- ➢ `makeUnavailable` - function that makes specific token unavailable for conversion; can only be called by authorized addresses
- ➢ `exchange` - function that exchanges one stable token for another; it burns sent token and mints others to msg.sender’s address. Defined stable tokens that are supposed to be exchanged must be available for conversion in order for function to work.

6. Static analysis test, vulnerabilities and outcomes

- ✓ Static analysis of the code was conducted and no security flows were found.

https://oyente.melon.fund
browser/stable.sol:Forex
EVM Code Coverage :
Callstack Depth Attack Vulnerability : False
Re-Entrancy Vulnerability : False
Assertion Failure: False
Parity Multisig Bug 2 : False
Transaction-Ordering Dependence (TOD) : False

- ✓ Over and underflows are not possible in this contract.

7. Final comments

Function “setStableAddress” should probably be added, so not all (not even one) stable token must be deployed before **Forex.sol.**

Generally, the code is well commented. Contract imports two interfaces: **StableToken.sol** and **Main.sol** in order to call functions from those contracts.