GORDIAN AGENCY

SMART CONTRACT SECURITY AUDIT

September 7th, 2021

Professional Auditing Agency

Website https://gordian.agency/

AUDIT DETAILS

GORDIAN AGENCY

Project

Timechain Swap Token (TCS)

Deployer Address

0xfbfae0dd49882e503982f8eb4b8b1e464eca0b91

Client Contacts

_____

Blockchain

Fantom

Project Website

_____

BACKGROUND

GORDIAN AGENCY

Gordian Agency was commissioned by Timechain Swap Token (TCS) to perform an audit of smart contracts:

· Timechain Swap Token (TCS)

The purpose of this audit was to achieve the following:

- Ensure that the token contract functions as intended
- Identify potential security issues with the token contracts

The information in this report should be used to understand the risk exposure of the smart contracts, and as a guide to improve the security posture of the smart contracts by remediating the issues that were identified.

Timechain Swap Token (TCS) Smart Contract Details

Smart Contract Testing and Audit

Timechain Swap Token (TCS) contract is deployed on the Fantom network as the TimeChain Swap Token. It is tested on the local ganache network to check its usage and security vulnerabilities with manual checks and automated analysis tools.
Contract is already deployed on Fantom Mainnet: here

Methodology

1. **Analysis with automated tools**
   It is very important to understand the contract structure before testing. Automated tools like slither are used for the static analysis of the contract and get the overview of contract structure.
2. **White Box Testing / Source Code Review**
   Understanding programming language used and coding practices followed is a key in performing an efficient source code review solution. It is performed by Understand Program Specification, Obtain and Review Source Code and identifying flaws.

3. **Black Box Testing**
   Understanding how your system works is a key in providing you the right security Solution. Multiple Exploitations and garbage collection data is introduced to the contract to check its response.

Contract Summary

1. **Contract Context**
    - From Context
        - _msgData() (internal)
        - _msgSender() (internal)
        - constructor() (internal)
2. **Contract Ownable**
    - From Context
        - _msgData() (internal)
        - _msgSender() (internal)
    - From Ownable
        - _transferOwnership(address) (internal)
        - constructor() (internal)
        - owner() (public)
        - renounceOwnership() (public)
        - transferOwnership(address) (public)
3. **Contract IBEP20**
    - From IBEP20
        - allowance(address, address) (external)
        - approve(address, uint256) (external)
        - balanceOf(address) (external)
        - decimals() (external)
        - getOwner() (external)
        - name() (external)
        - totalSupply() (external)
        - transfer(address, uint256) (external)
        - transferFrom(address, address, uint256) (external)

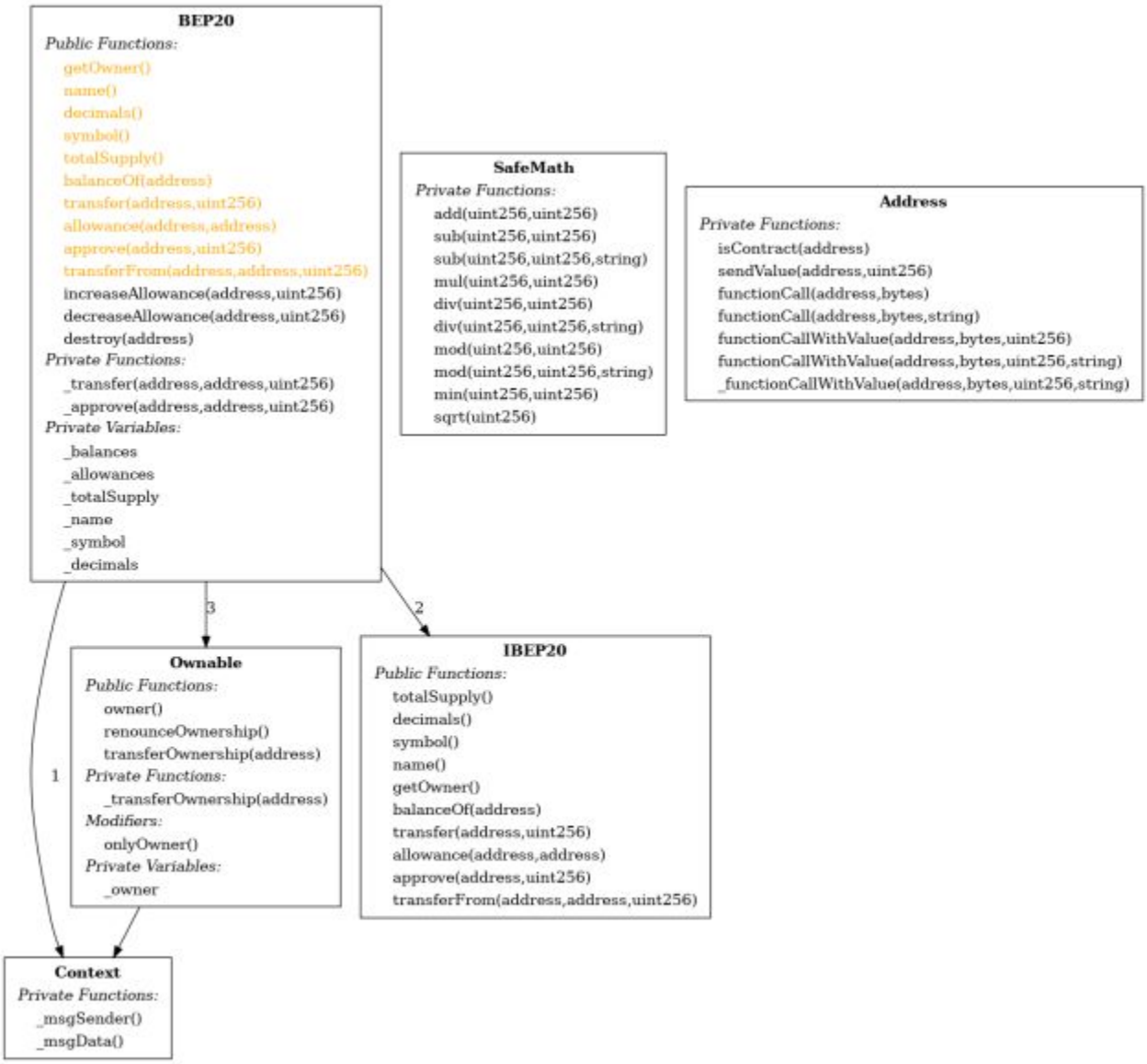
Contract Summary

4. **Contract SafeMath (Most derived contract)**
    - From SafeMath
        - add(uint256,uint256) (internal)
        - div(uint256,uint256) (internal)
        - div(uint256,uint256,string) (internal)
        - min(uint256,uint256) (internal)
        - mod(uint256,uint256) (internal)
        - mod(uint256,uint256,string) (internal)
        - mul(uint256,uint256) (internal)
        - sqrt(uint256) (internal)
        - sub(uint256,uint256) (internal)
        - sub(uint256,uint256,string) (internal)
5. **Contract Address (Most derived contract)**
    - - From Address
        - _functionCallWithValue(address,bytes,uint256,string) (private)
        - functionCall(address,bytes) (internal)
        - functionCall(address,bytes,string) (internal)
        - functionCallWithValue(address,bytes,uint256) (internal)
        - functionCallWithValue(address,bytes,uint256,string) (internal)
        - isContract(address) (internal)
        - sendValue(address,uint256) (internal)

Contract Summary

6. **Contract BEP20 (Most derived contract)**
    - From Ownable
        - _transferOwnership(address) (internal)
        - constructor() (internal)
        - owner() (public)
        - renounceOwnership() (public)
        - transferOwnership(address) (public)
    - From Context
        - _msgData() (internal)
        - _msgSender() (internal)
    - From BEP20
        - _approve(address,address,uint256) (internal)
        - _transfer(address,address,uint256) (internal)
        - allowance(address,address) (public)
        - approve(address,uint256) (public)
        - balanceOf(address) (public)
        - constructor(string,string,uint256) (public) - decimals() (public)
        - decreaseAllowance(address,uint256) (public)
        - destroy(address) (public)
        - getOwner() (external)
        - increaseAllowance(address,uint256) (public)
        - name() (public)
        - symbol() (public)
        - totalSupply() (public)
        - transfer(address,uint256) (public)
        - transferFrom(address,address,uint256) (public)

Contract Inheritance Graph



Findings

Manual Checks

BEP20 Contract

1. Tokens can only be minted while deploying the contract.
2. All the minted tokens are transferred to the deployer during deployment, i.e deployer owns the totalSupply initially.
3. Tokens can be transferred using transfer() method.
4. Users can set the allowance for others to spend their token using increaseAllowance() method.
5. Users can increase or decrease the allowance.
6. Users can transfer allowance tokens using transferFrom() method.
7. Allowance amount is adjusted after execution of transferFrom() method.
8. Owner can destroy the contract.
9. All the ether stored in contract is transferred to a given address during destruction.

ISSUES DETAILS

Vulnerability Analysis

BEP20 Contract

1. Guard check using modifiers to verify the owner in the destroy() method.
2. SafeMath library is implemented on uint256 datatype to prevent integer overflow/underflow.

CONCLUSION

Contract is not prone to any known attacks and vulnerabilities

NOTES:

Please check the disclaimer below and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is solely provided for the contracts mentioned in the report and does not include any other potential contracts deployed by the Owner.

DISCLAIMER

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Gordian and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Gordian) owe no duty of care towards you or any other person, nor does Gordian make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Gordian hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Gordian hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Gordian, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.