TipViaCrypto

Token

Smart Contract Audit Report

June 17, 2022

Introduction

1. About TipViaCrypto

TipViaCrypto intends to create a unique ecosystem that allows cryptocurrencies to be used in everyday life. In recent times, it was common to hear about companies illegally withholding the funds of people who supported a particular creator or fundraiser. We are aware of the fact that people encountering unfair practices lose trust in the intermediaries involved in such transactions. TipViaCrypto wants to provide a solution in which the community can have full confidence. The public needs a guarantee that their funds will go to the right place. TipViaCrypto is able to guarantee this thanks to our unique solution the funds go directly from the supporter to the supported person excluding TipViaCrypto from the funds' circulation. This is the most transparent and fair solution there is.

Visit https://tipviacrypto.io/ to know more about it.

2. About ImmuneBytes

ImmuneBytes is a security start-up to provides professional services in the blockchain space. The team has hands-on experience in conducting smart contract audits, penetration testing, and security consulting. ImmuneBytes's security auditors have worked on various A-league projects and have a great understanding of DeFi projects like AAVE, Compound, 0x Protocol, Uniswap, and dydx.

The team has been able to secure 105+ blockchain projects by providing security services on different frameworks. ImmuneBytes team helps start-ups with a detailed analysis of the system ensuring security and managing the overall project.

Visit http://immunebytes.com/ to know more about the services.

Documentation Details

The team has provided the following doc for the purpose of audit:

1. https://tipviacrypto.io/wp-content/uploads/2022/05/Whitepaper_EN.pdf

Audit Process & Methodology

ImmuneBytes team has performed thorough testing of the project starting with analyzing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third-party smart contracts and libraries.

Our team then performed a formal line-by-line inspection of the Smart Contract in order to find any potential issues like Signature Replay Attacks, Unchecked External Calls, External Contract Referencing, Variable Shadowing, Race conditions, Transaction-ordering dependence, timestamp dependence, DoS attacks, and others.

In the Unit testing phase, we run unit tests written by the developer in order to verify the functions work as intended. In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was audited by a team of independent auditors which includes -

1. Testing the functionality of the Smart Contract to determine proper logic has been followed throughout.
2. Analyzing the complexity of the code by thorough, manual review of the code, line-by-line.
3. Deploying the code on testnet using multiple clients to run live tests.
4. Analyzing failure preparations to check how the Smart Contract performs in case of bugs and vulnerabilities.
5. Checking whether all the libraries used in the code are on the latest version.
6. Analyzing the security of the on-chain data.

Audit Details

- Project Name: TipViaCrypto
- Contracts Name: TipViaCryptoERC20.sol
- Languages: Solidity(Smart contract), Typescript (Unit Testing)
- Link for codebase for audit: https://gist.github.com/WojcikMM/2935513c7dcf59c3732d82e5420a24f4
- Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Contract Library, Slither, SmartCheck

Audit Goals

The focus of the audit was to verify that the smart contract system is secure, resilient, and working according to its specifications. The audit activities can be grouped into the following three categories:

1. Security: Identifying security-related issues within each contract and within the system of contracts.
2. Sound Architecture: Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.
3. Code Correctness and Quality: A full review of the contract source code. The primary areas of focus include
    a. Correctness
    b. Readability
    c. Sections of code with high complexity
    d. Quantity and quality of test coverage

Security Level Reference

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

High severity issues will bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Issues	High	Medium	Low
Open	-	-	-
Closed	-	-	-

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

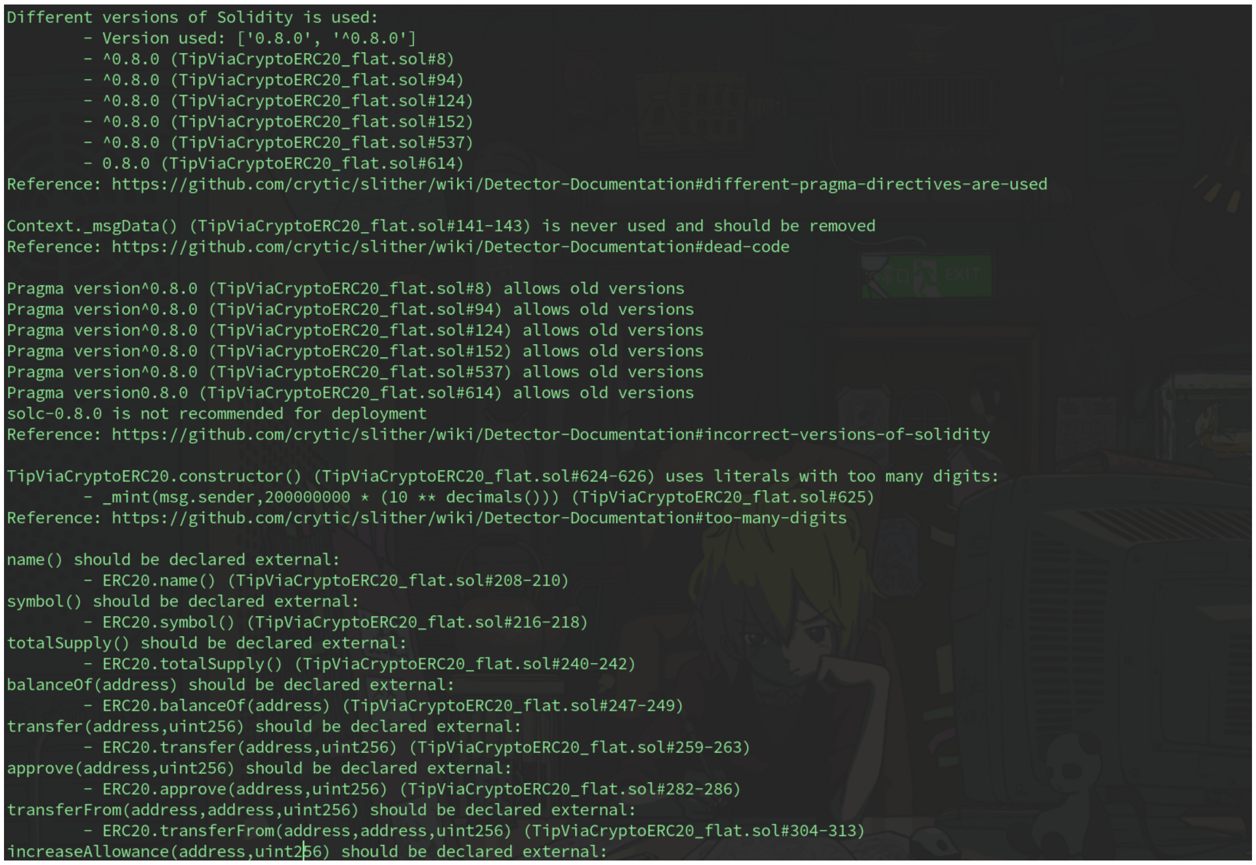
No issues were found.

Recommendation / Informational

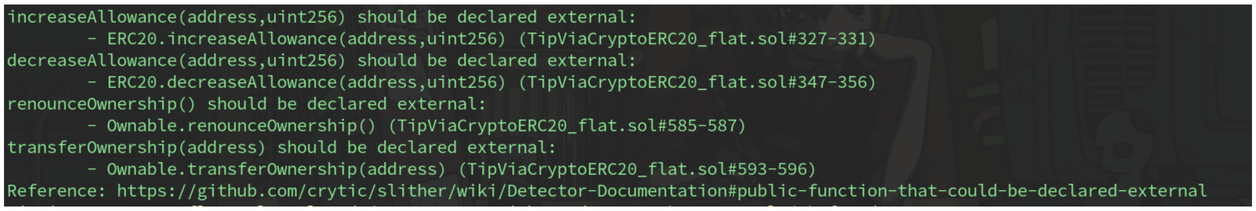
No issues were found.

Automated Audit Result

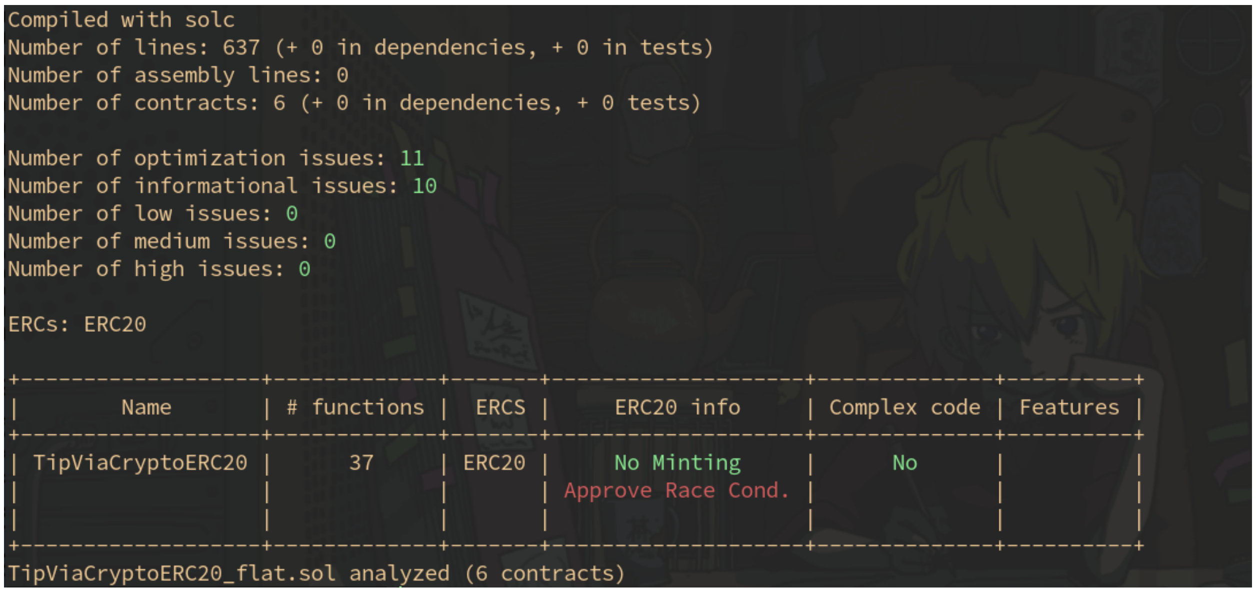
Slither



```
Different versions of Solidity is used:
        - Version used: ['0.8.0', '^0.8.0']
        - ^0.8.0 (TipViaCryptoERC20_flat.sol#8)
        - ^0.8.0 (TipViaCryptoERC20_flat.sol#94)
        - ^0.8.0 (TipViaCryptoERC20_flat.sol#124)
        - ^0.8.0 (TipViaCryptoERC20_flat.sol#152)
        - ^0.8.0 (TipViaCryptoERC20_flat.sol#537)
        - 0.8.0 (TipViaCryptoERC20_flat.sol#614)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

Context._msgData() (TipViaCryptoERC20_flat.sol#141-143) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.8.0 (TipViaCryptoERC20_flat.sol#8) allows old versions
Pragma version^0.8.0 (TipViaCryptoERC20_flat.sol#94) allows old versions
Pragma version^0.8.0 (TipViaCryptoERC20_flat.sol#124) allows old versions
Pragma version^0.8.0 (TipViaCryptoERC20_flat.sol#152) allows old versions
Pragma version^0.8.0 (TipViaCryptoERC20_flat.sol#537) allows old versions
Pragma version0.8.0 (TipViaCryptoERC20_flat.sol#614) allows old versions
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

TipViaCryptoERC20.constructor() (TipViaCryptoERC20_flat.sol#624-626) uses literals with too many digits:
        - _mint(msg.sender,200000000 * (10 ** decimals())) (TipViaCryptoERC20_flat.sol#625)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

name() should be declared external:
        - ERC20.name() (TipViaCryptoERC20_flat.sol#208-210)
symbol() should be declared external:
        - ERC20.symbol() (TipViaCryptoERC20_flat.sol#216-218)
totalSupply() should be declared external:
        - ERC20.totalSupply() (TipViaCryptoERC20_flat.sol#240-242)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (TipViaCryptoERC20_flat.sol#247-249)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (TipViaCryptoERC20_flat.sol#259-263)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (TipViaCryptoERC20_flat.sol#282-286)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (TipViaCryptoERC20_flat.sol#304-313)
increaseAllowance(address,uint256) should be declared external:
```

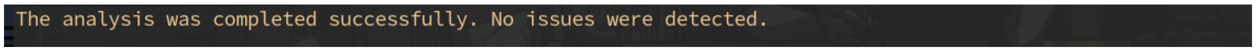


```
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (TipViaCryptoERC20_flat.sol#327-331)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (TipViaCryptoERC20_flat.sol#347-356)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (TipViaCryptoERC20_flat.sol#585-587)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (TipViaCryptoERC20_flat.sol#593-596)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```



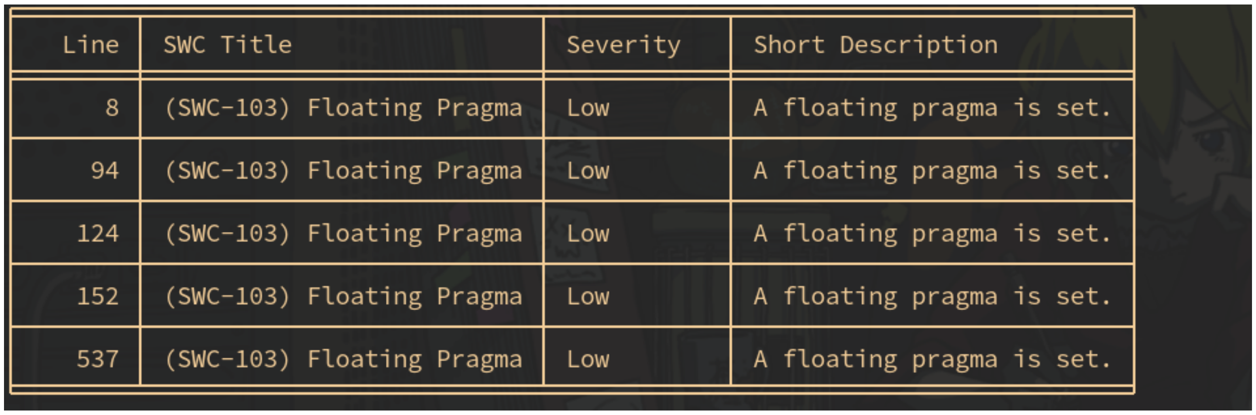
```
Compiled with solc
Number of lines: 637 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 6 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 11
Number of informational issues: 10
Number of low issues: 0
Number of medium issues: 0
Number of high issues: 0

ERCs: ERC20

+--------------------+-------------+-------+---------------------+--------------+----------+
|        Name        | # functions |  ERCS |      ERC20 info     | Complex code | Features |
+--------------------+-------------+-------+---------------------+--------------+----------+
| TipViaCryptoERC20 |      37     | ERC20 |      No Minting     |      No      |          |
|                    |             |       | Approve Race Cond. |              |          |
|                    |             |       |                     |              |          |
+--------------------+-------------+-------+---------------------+--------------+----------+
TipViaCryptoERC20_flat.sol analyzed (6 contracts)
```

Mythril



```
The analysis was completed successfully. No issues were detected.
```

Mythx

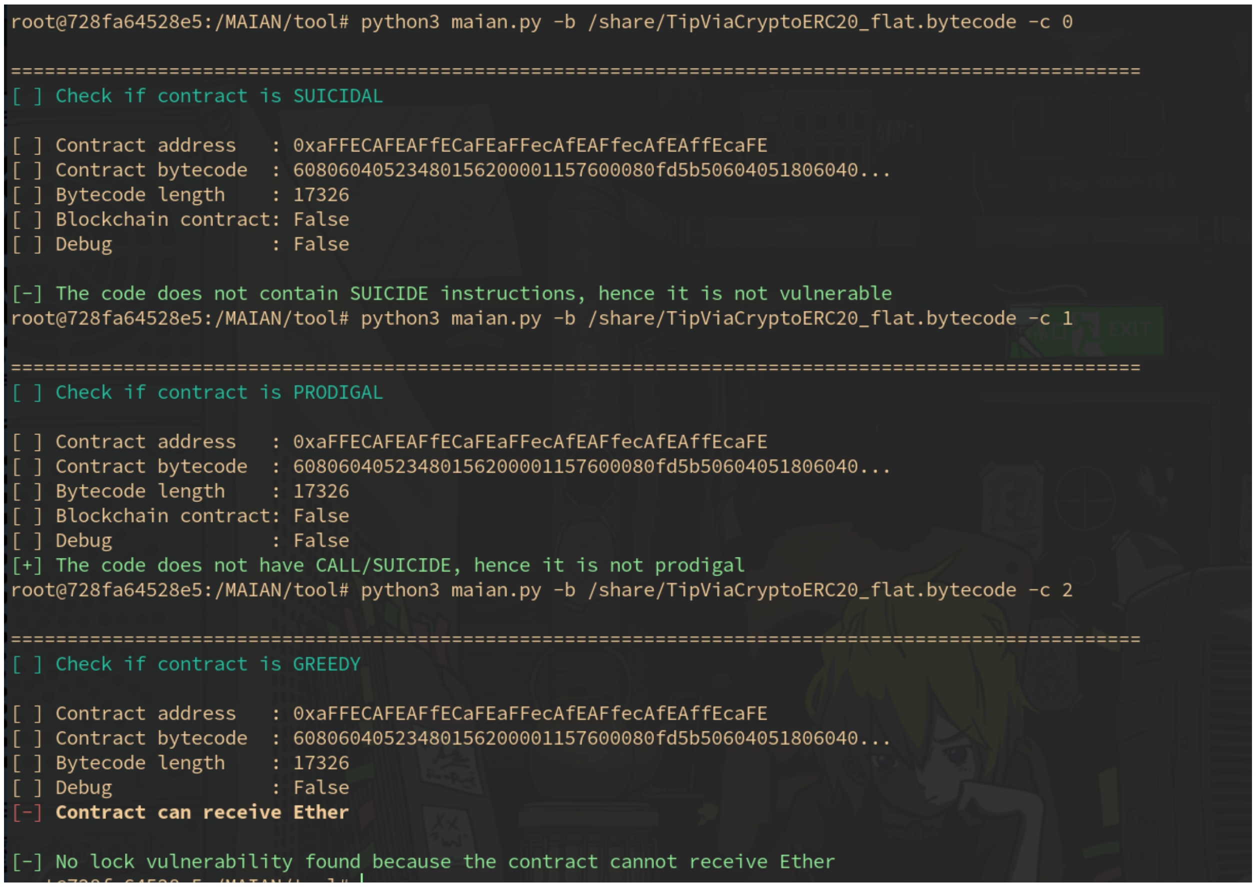


Line	SWC Title	Severity	Short Description
8	(SWC-103) Floating Pragma	Low	A floating pragma is set.
94	(SWC-103) Floating Pragma	Low	A floating pragma is set.
124	(SWC-103) Floating Pragma	Low	A floating pragma is set.
152	(SWC-103) Floating Pragma	Low	A floating pragma is set.
537	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Testnet

Deployment	https://rinkeby.etherscan.io/address/0xb543109cd73356f444cd945c3244fa32663ae3e2	
announceWithdraw	Pass	https://rinkeby.etherscan.io/tx/0xe2f424378932dd11cebd818ab48cfc82543b7f5517caa29511c56ff61e458623
cancelWithdraw	Pass	https://rinkeby.etherscan.io/tx/0x8ee0d88237696afaecff1fcee6a050016cb4887378327a73103a6ef0f29e95b7
claimReward	Pass	https://rinkeby.etherscan.io/tx/0x4df6e63d98474d0930bd53b621b0e0af9d120be3ee61abeb44df6b5bbcbfb3db
grantRole	Pass	https://rinkeby.etherscan.io/tx/0xebf4cfe2dff72da8fc181f99d7b3b479ce2daf6428297680cb86ee121e697cc0
renounceRole	Pass	https://rinkeby.etherscan.io/tx/0xd2173c8287fcd0a168e8f10af8ce876c8337a783f3f1f4180ac4ec329ab60e60
revokeRole	Pass	https://rinkeby.etherscan.io/tx/0x8cf53549ab63eb441378eb62da293aa1761f9fd047887e99b4119eb73a9030a5
stake	Pass	https://rinkeby.etherscan.io/tx/0xa6594de3d2eb508429e61173c1f347ba0e1f87ff72360b2ef8aec18073bf726e
unstake	Pass	https://rinkeby.etherscan.io/tx/0x5edd0669be6082229845be5aefa29f2b9630a2313ff6b65a5fd5b5409aa6418d
whitelist	Pass	https://rinkeby.etherscan.io/tx/0x97e8d8d2b47e2cffde80133fac5eb440a8e634de050bf1137a18f25231848696

Maian Bytecode Analysis



```
root@728fa64528e5:/MAIAN/tool# python3 maian.py -b /share/TipViaCryptoERC20_flat.bytecode -c 0

====================================================================================================
[ ] Check if contract is SUICIDAL

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 6080604052348015620001157600080fd5b50604051806040...
[ ] Bytecode length    : 17326
[ ] Blockchain contract: False
[ ] Debug              : False

[-] The code does not contain SUICIDE instructions, hence it is not vulnerable
root@728fa64528e5:/MAIAN/tool# python3 maian.py -b /share/TipViaCryptoERC20_flat.bytecode -c 1

====================================================================================================
[ ] Check if contract is PRODIGAL

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 6080604052348015620001157600080fd5b50604051806040...
[ ] Bytecode length    : 17326
[ ] Blockchain contract: False
[ ] Debug              : False
[+] The code does not have CALL/SUICIDE, hence it is not prodigal
root@728fa64528e5:/MAIAN/tool# python3 maian.py -b /share/TipViaCryptoERC20_flat.bytecode -c 2

====================================================================================================
[ ] Check if contract is GREEDY

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 6080604052348015620001157600080fd5b50604051806040...
[ ] Bytecode length    : 17326
[ ] Debug              : False
[-] Contract can receive Ether

[-] No lock vulnerability found because the contract cannot receive Ether
```

This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

Testnet

Deployment	https://rinkeby.etherscan.io/address/0xc5faa420592377e7a9676f318fe039420191ecbb	
Approve()	Pass	https://rinkeby.etherscan.io/tx/0x55494fc7a4ef815b9bacb925c6286f3d367dcabb87452f98db2275f16186b882
burn()	Pass	https://rinkeby.etherscan.io/tx/0xa530abb3dc4f6c2d87275d26e5d2e7f20eff6011e65a8520fe5de5beac4f6813
decreaseAllowance()	Pass	https://rinkeby.etherscan.io/tx/0x8266593af943217b465333a6a1cc034079fcba3b7c473708cdccf3197e459e06
increaseAllowance()	Pass	https://rinkeby.etherscan.io/tx/0x739360e260c70cb0d6b3759e2126ac4be84fcb24a50ead92c31a8364257b0ad9
transfer()	Pass	https://rinkeby.etherscan.io/tx/0xd78802c8649388377719735d343455ae42aeb493383cacccac338ee43fe84c03
transferOwnership()	Pass	https://rinkeby.etherscan.io/tx/0x5fa0f2fb2074075be7cad44e8f76dadbed78c100e8eed32d1a5be917f97150fb

Concluding Remarks

While conducting the audits of the TipViaCrypto smart contracts, it was observed that the contracts contain no severity issues.

Disclaimer

ImmuneBytes's audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

Our team does not endorse the TipViaCrypto platform or its product nor this audit is investment advice.
Notes:

- Please make sure contracts deployed on the mainnet are the ones audited.
- Check for the code refactor by the team on critical issues.

ImmuneBytes