Kaxaa

Token

Smart Contract Audit Report

August 04, 2022

Introduction

1. About Kaxaa

The Kaxaa Token is a utility token used to access proprietary real estate data on the Kaxaa Platform and used as currency in real estate transactions. The value of the Kaxaa Token is stabilized and determined by the proprietary Kaxaa Index algorithm based on real-world real estate market indicators. The Kaxaa Token may be used in real-world real estate transactions as a stabilized "currency" in and out of the Kaxaa platform.

Visit https://kaxaa.com/ to know more about it.

2. About ImmuneBytes

ImmuneBytes is a security start-up that 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 understand DeFi projects like AAVE, Compound, 0x Protocol, Uniswap, and dydx.

The team has assessed 175+ 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 Kaxaa team has provided the following doc for the purpose of audit:

1. KAXAA_White-Paper-2022.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 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 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. Check whether all the code libraries are on the latest version.
6. Analyzing the security of the on-chain data.

Audit Details

- Project Name: Kaxaa
- Contracts Name: KaxToken.sol
- Languages: Solidity(Smart contract), Typescript (Unit Testing)
- Link for codebase for audit: https://gitlab.com/KASA-Project/kaxaa-token
- Final Commit: b59b669089afa13a9f7898c35a9d44dcf75f555b
- 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 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:

Admin/Owner Privileges can be misused intentionally or unintentionally.

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	-	-	-

Admin/Owner Privileges

The **admin/owner** of the **Kaxaa** smart contract has a privilege over the smart contract. The privilege can be misused intentionally or unintentionally (in case the admin's private key gets hacked). We assume that these extra rights will always be used appropriately.

- *The owner of the contract can mint **n** number of tokens.*

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

No issues were found.

Recommendation / Informational

1. **Unlocked Pragma statements found in the contracts**

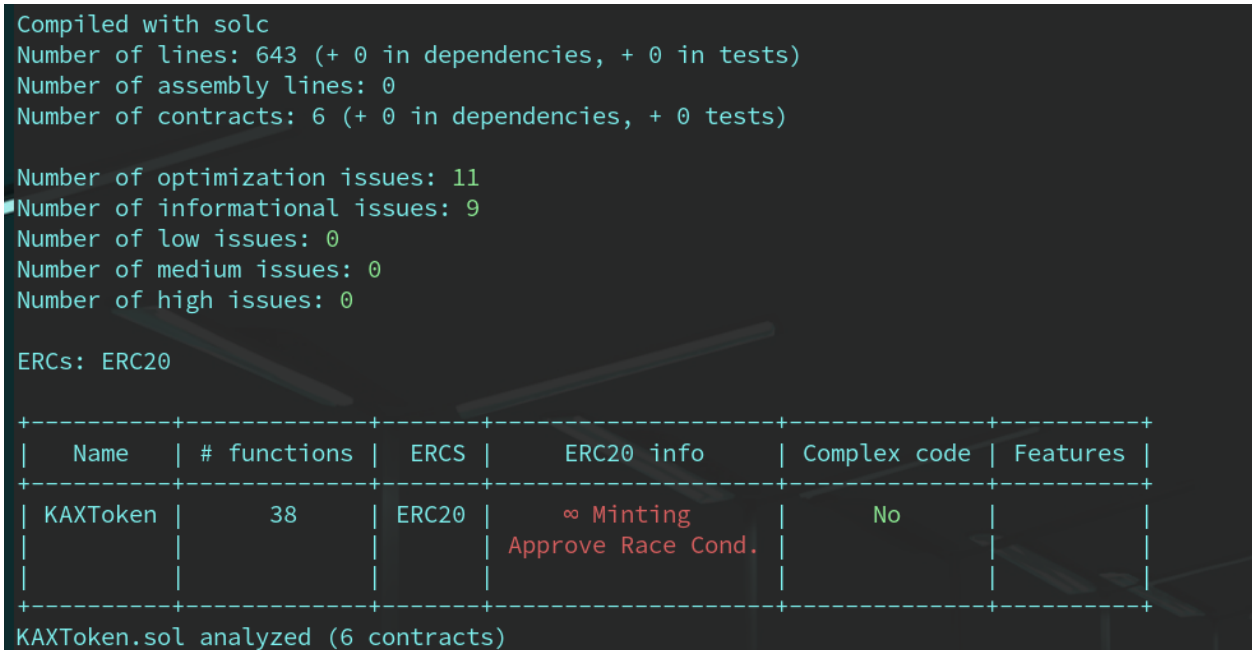
   Explanation:
   During the code review, it was found that the contracts included unlocked pragma solidity version statements.

   It's not considered a better practice in Smart contract development to do so as it might lead to accidental deployment to a version with unfixed bugs.

   Recommendation:
   It's always recommended to lock pragma statements to a specific version while writing contracts.

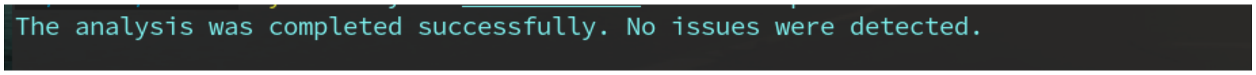
Automated Audit Result

Slither



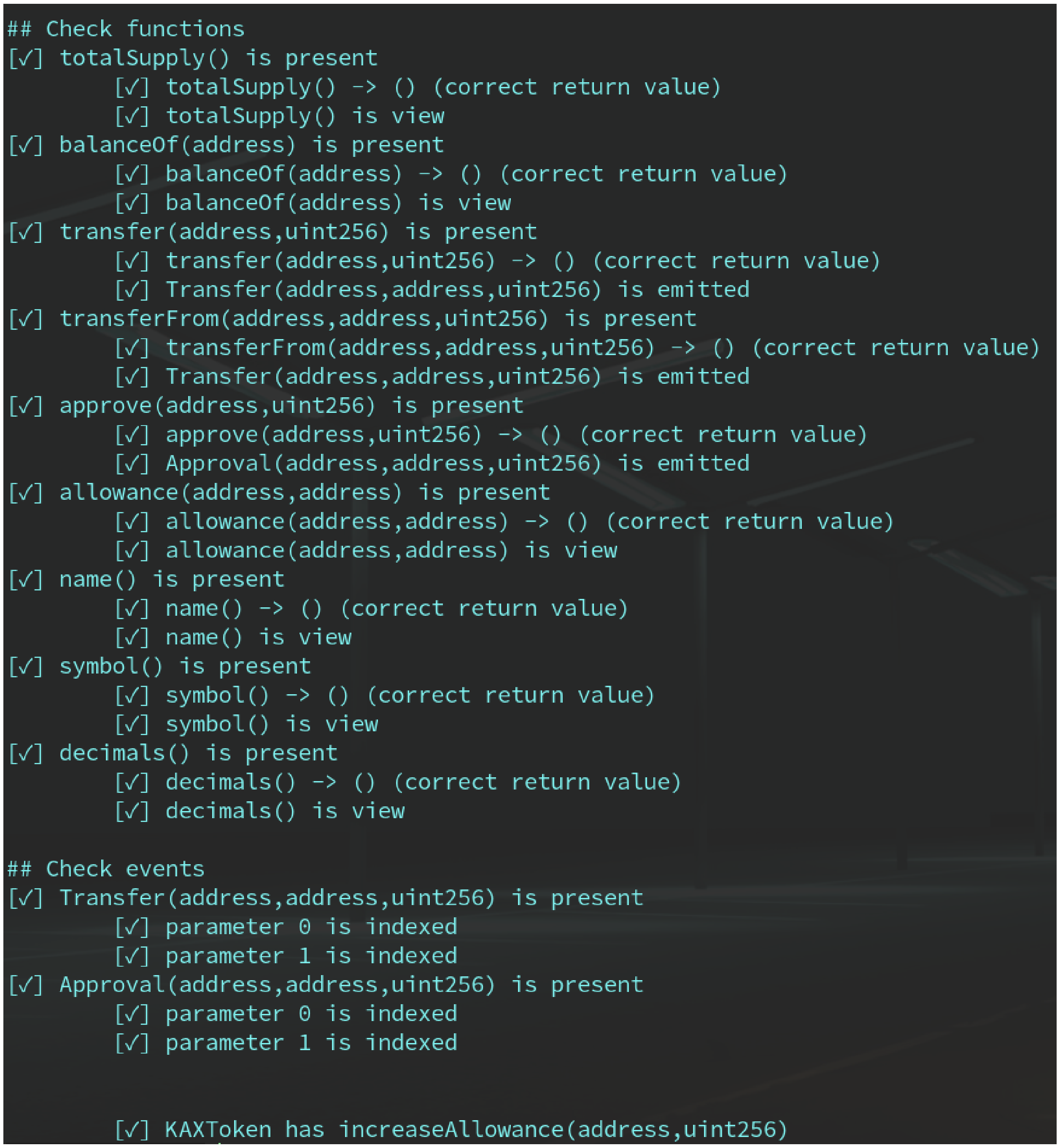
```
Compiled with solc
Number of lines: 643 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 6 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 11
Number of informational issues: 9
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 |
+----------+-------------+-------+-------------------+-------------+----------+
| KAXToken |      38     | ERC20 |     ∞ Minting     |      No      |          |
|          |             |       | Approve Race Cond. |             |          |
|          |             |       |                   |              |          |
+----------+-------------+-------+-------------------+-------------+----------+
KAXToken.sol analyzed (6 contracts)
```

Mythril



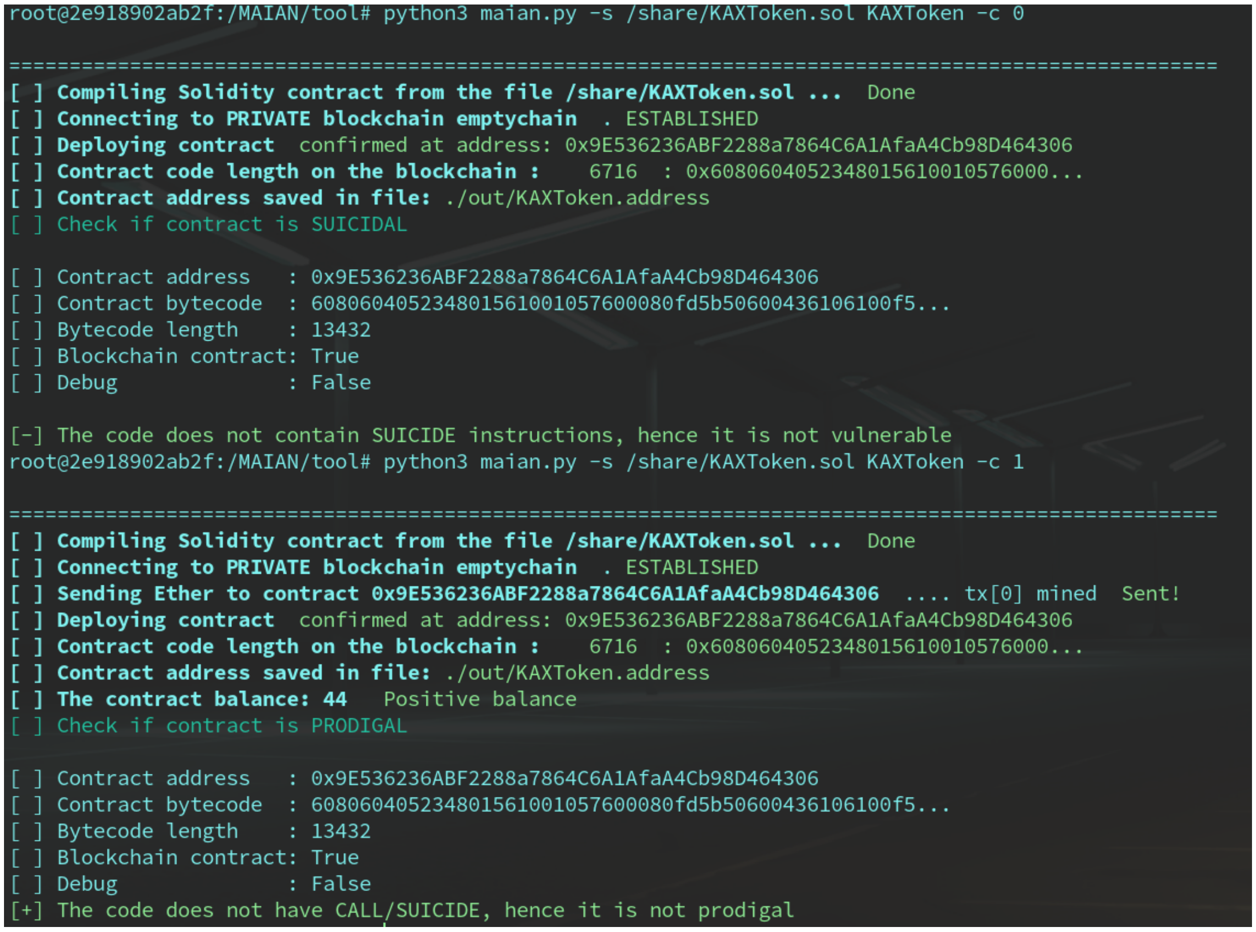
```
The analysis was completed successfully. No issues were detected.
```

ERC20 Check

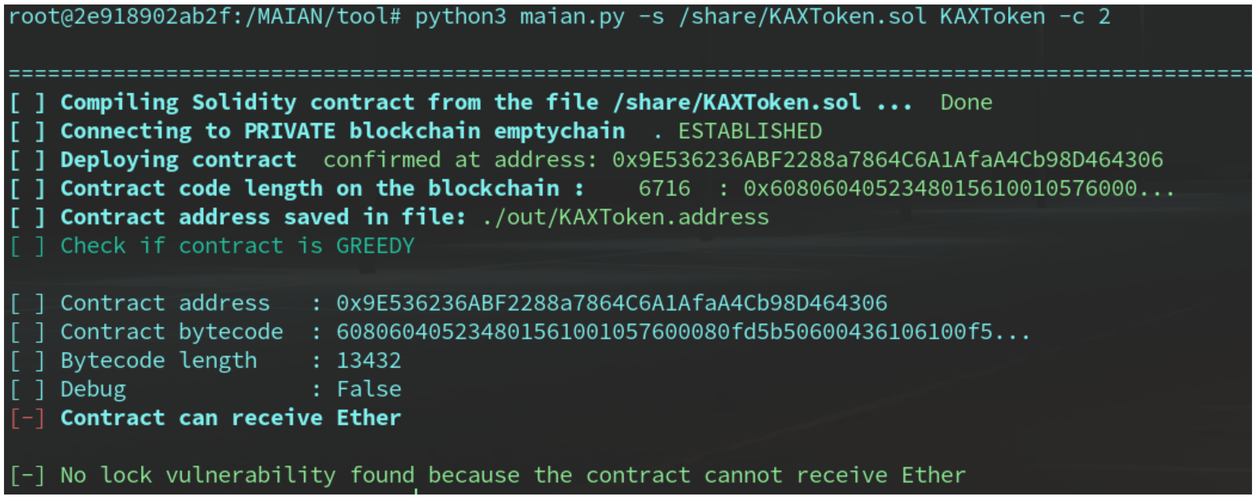


```
## Check functions
[✓] totalSupply() is present
        [✓] totalSupply() -> () (correct return value)
        [✓] totalSupply() is view
[✓] balanceOf(address) is present
        [✓] balanceOf(address) -> () (correct return value)
        [✓] balanceOf(address) is view
[✓] transfer(address,uint256) is present
        [✓] transfer(address,uint256) -> () (correct return value)
        [✓] Transfer(address,address,uint256) is emitted
[✓] transferFrom(address,address,uint256) is present
        [✓] transferFrom(address,address,uint256) -> () (correct return value)
        [✓] Transfer(address,address,uint256) is emitted
[✓] approve(address,uint256) is present
        [✓] approve(address,uint256) -> () (correct return value)
        [✓] Approval(address,address,uint256) is emitted
[✓] allowance(address,address) is present
        [✓] allowance(address,address) -> () (correct return value)
        [✓] allowance(address,address) is view
[✓] name() is present
        [✓] name() -> () (correct return value)
        [✓] name() is view
[✓] symbol() is present
        [✓] symbol() -> () (correct return value)
        [✓] symbol() is view
[✓] decimals() is present
        [✓] decimals() -> () (correct return value)
        [✓] decimals() is view

## Check events
[✓] Transfer(address,address,uint256) is present
        [✓] parameter 0 is indexed
        [✓] parameter 1 is indexed
[✓] Approval(address,address,uint256) is present
        [✓] parameter 0 is indexed
        [✓] parameter 1 is indexed


        [✓] KAXToken has increaseAllowance(address,uint256)
```

Maian Solidity Analysis

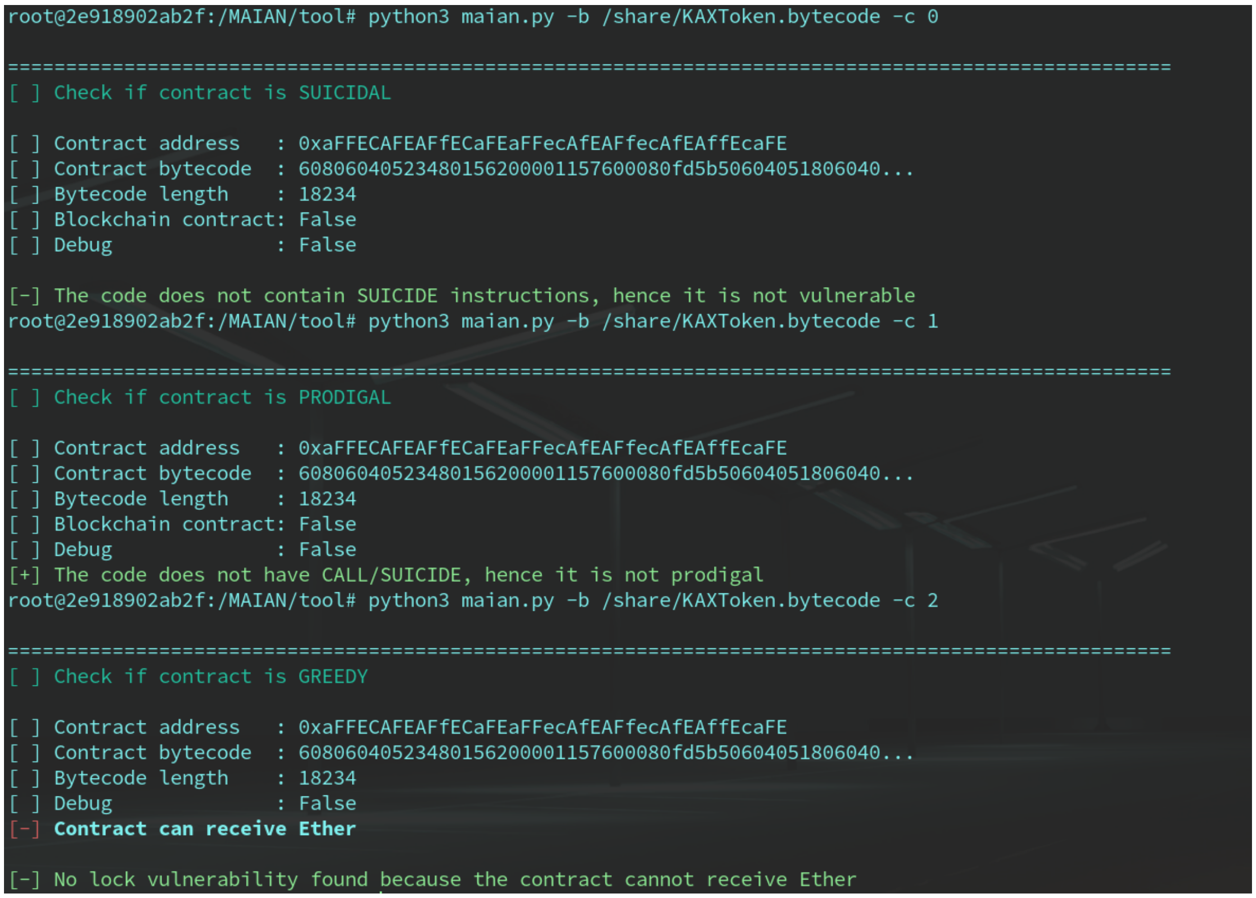


```
root@2e918902ab2f:/MAIAN/tool# python3 maian.py -s /share/KAXToken.sol KAXToken -c 0

====================================================================================================
[ ] Compiling Solidity contract from the file /share/KAXToken.sol ...   Done
[ ] Connecting to PRIVATE blockchain emptychain   . ESTABLISHED
[ ] Deploying contract  confirmed at address: 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract code length on the blockchain :    6716  : 0x6080604052348015610010576000...
[ ] Contract address saved in file: ./out/KAXToken.address
[ ] Check if contract is SUICIDAL

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 608060405234801561001057600080fd5b50600436106100f5...
[ ] Bytecode length    : 13432
[ ] Blockchain contract: True
[ ] Debug              : False

[-] The code does not contain SUICIDE instructions, hence it is not vulnerable
root@2e918902ab2f:/MAIAN/tool# python3 maian.py -s /share/KAXToken.sol KAXToken -c 1

====================================================================================================
[ ] Compiling Solidity contract from the file /share/KAXToken.sol ...   Done
[ ] Connecting to PRIVATE blockchain emptychain   . ESTABLISHED
[ ] Sending Ether to contract 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306  .... tx[0] mined  Sent!
[ ] Deploying contract  confirmed at address: 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract code length on the blockchain :    6716  : 0x6080604052348015610010576000...
[ ] Contract address saved in file: ./out/KAXToken.address
[ ] The contract balance: 44   Positive balance
[ ] Check if contract is PRODIGAL

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 608060405234801561001057600080fd5b50600436106100f5...
[ ] Bytecode length    : 13432
[ ] Blockchain contract: True
[ ] Debug              : False
[+] The code does not have CALL/SUICIDE, hence it is not prodigal
```



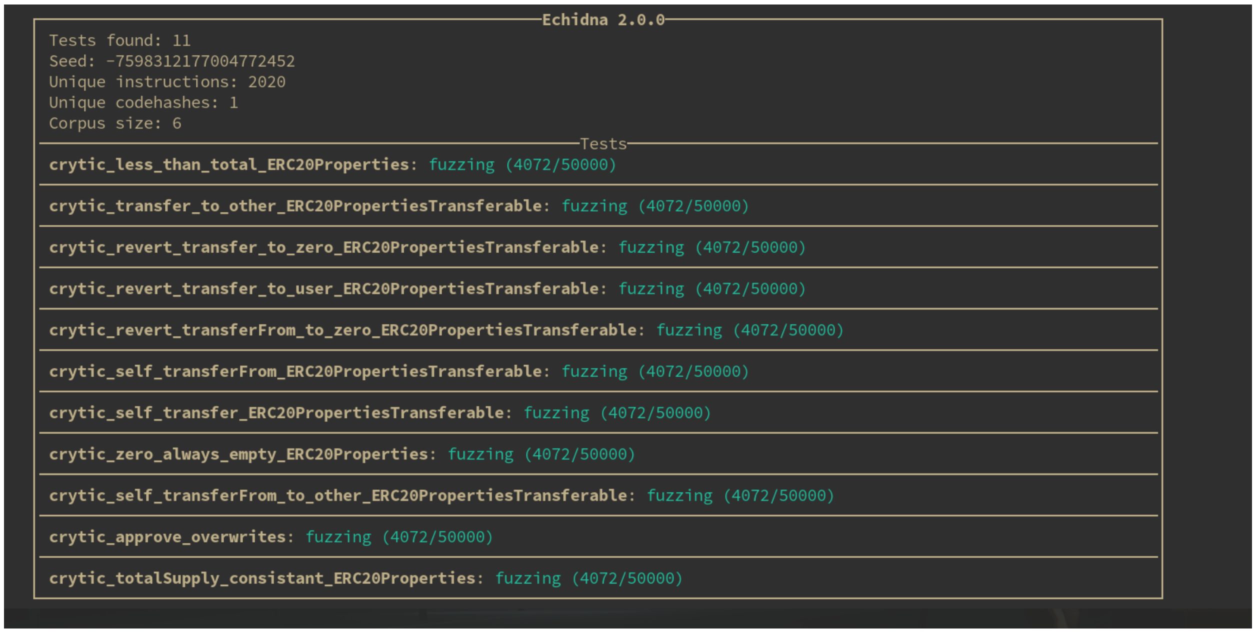
```
root@2e918902ab2f:/MAIAN/tool# python3 maian.py -s /share/KAXToken.sol KAXToken -c 2

====================================================================================================
[ ] Compiling Solidity contract from the file /share/KAXToken.sol ...   Done
[ ] Connecting to PRIVATE blockchain emptychain   . ESTABLISHED
[ ] Deploying contract  confirmed at address: 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract code length on the blockchain :    6716  : 0x6080604052348015610010576000...
[ ] Contract address saved in file: ./out/KAXToken.address
[ ] Check if contract is GREEDY

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 608060405234801561001057600080fd5b50600436106100f5...
[ ] Bytecode length    : 13432
[ ] Debug              : False
[-] Contract can receive Ether

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

Maian Bytecode Analysis

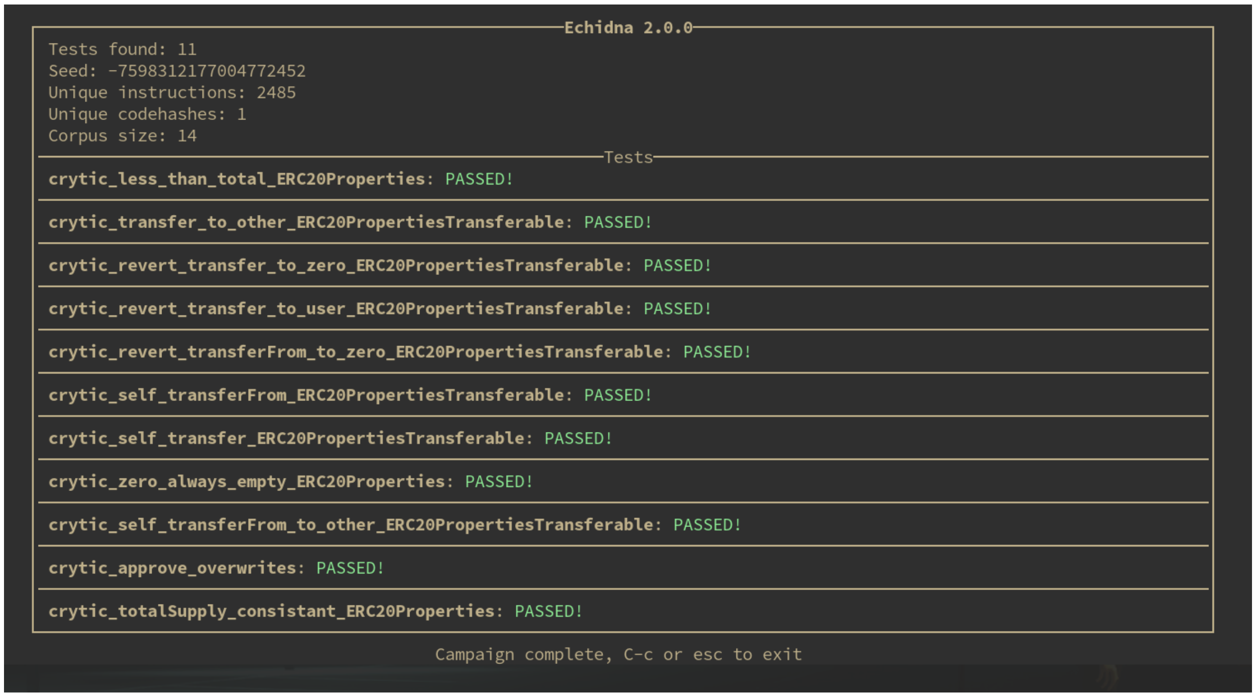


```
root@2e918902ab2f:/MAIAN/tool# python3 maian.py -b /share/KAXToken.bytecode -c 0

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

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 6080604052348015620001157600080fd5b50604051806040...
[ ] Bytecode length    : 18234
[ ] Blockchain contract: False
[ ] Debug              : False

[-] The code does not contain SUICIDE instructions, hence it is not vulnerable
root@2e918902ab2f:/MAIAN/tool# python3 maian.py -b /share/KAXToken.bytecode -c 1

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

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 6080604052348015620001157600080fd5b50604051806040...
[ ] Bytecode length    : 18234
[ ] Blockchain contract: False
[ ] Debug              : False
[+] The code does not have CALL/SUICIDE, hence it is not prodigal
root@2e918902ab2f:/MAIAN/tool# python3 maian.py -b /share/KAXToken.bytecode -c 2

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

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 6080604052348015620001157600080fd5b50604051806040...
[ ] Bytecode length    : 18234
[ ] Debug              : False
[-] Contract can receive Ether

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

Fuzz Test

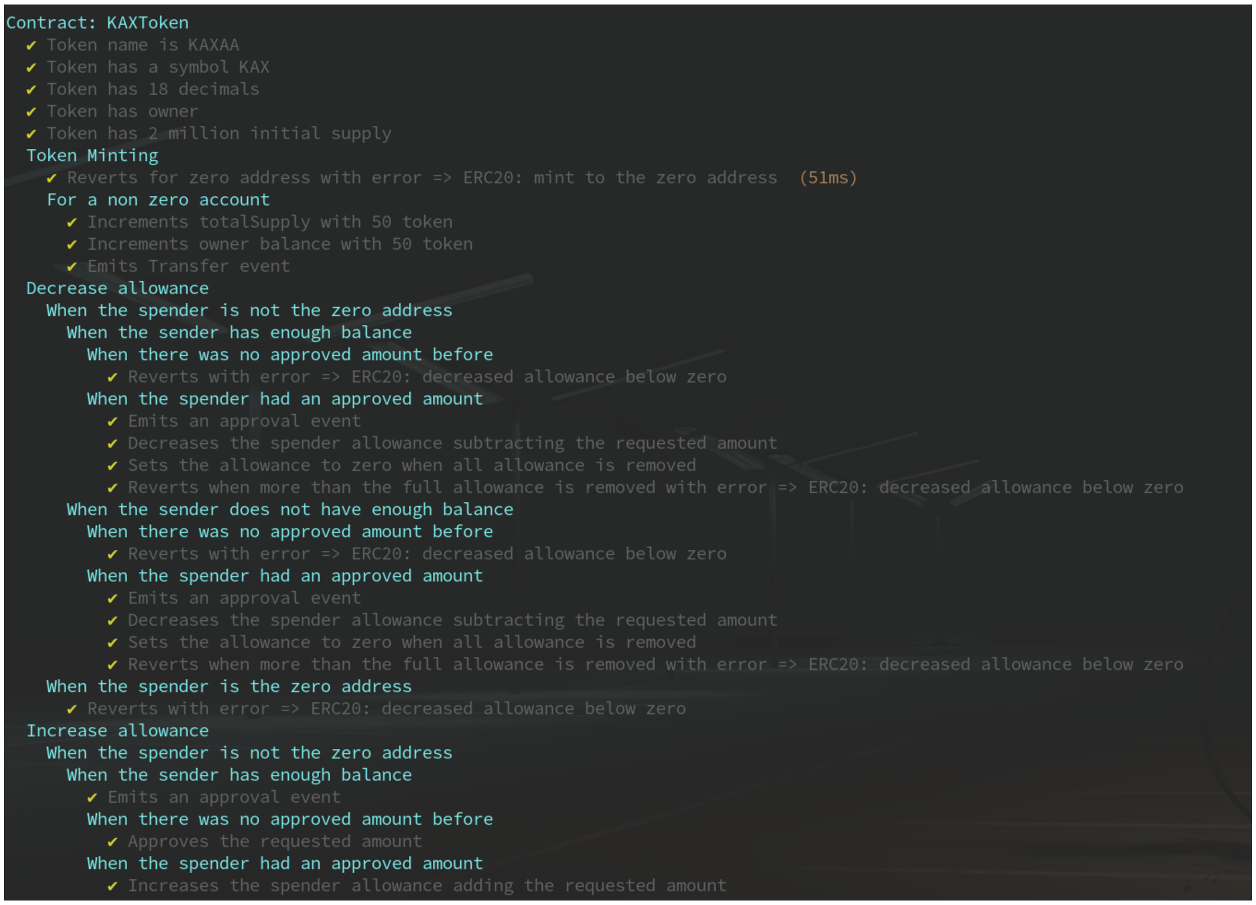


```
Echidna 2.0.0
Tests found: 11
Seed: -7598312177004772452
Unique instructions: 2020
Unique codehashes: 1
Corpus size: 6
Tests
crytic_less_than_total_ERC20Properties: fuzzing (4072/50000)
crytic_transfer_to_other_ERC20PropertiesTransferable: fuzzing (4072/50000)
crytic_revert_transfer_to_zero_ERC20PropertiesTransferable: fuzzing (4072/50000)
crytic_revert_transfer_to_user_ERC20PropertiesTransferable: fuzzing (4072/50000)
crytic_revert_transferFrom_to_zero_ERC20PropertiesTransferable: fuzzing (4072/50000)
crytic_self_transferFrom_ERC20PropertiesTransferable: fuzzing (4072/50000)
crytic_self_transfer_ERC20PropertiesTransferable: fuzzing (4072/50000)
crytic_zero_always_empty_ERC20Properties: fuzzing (4072/50000)
crytic_self_transferFrom_to_other_ERC20PropertiesTransferable: fuzzing (4072/50000)
crytic_approve_overwrites: fuzzing (4072/50000)
crytic_totalSupply_consistant_ERC20Properties: fuzzing (4072/50000)
```

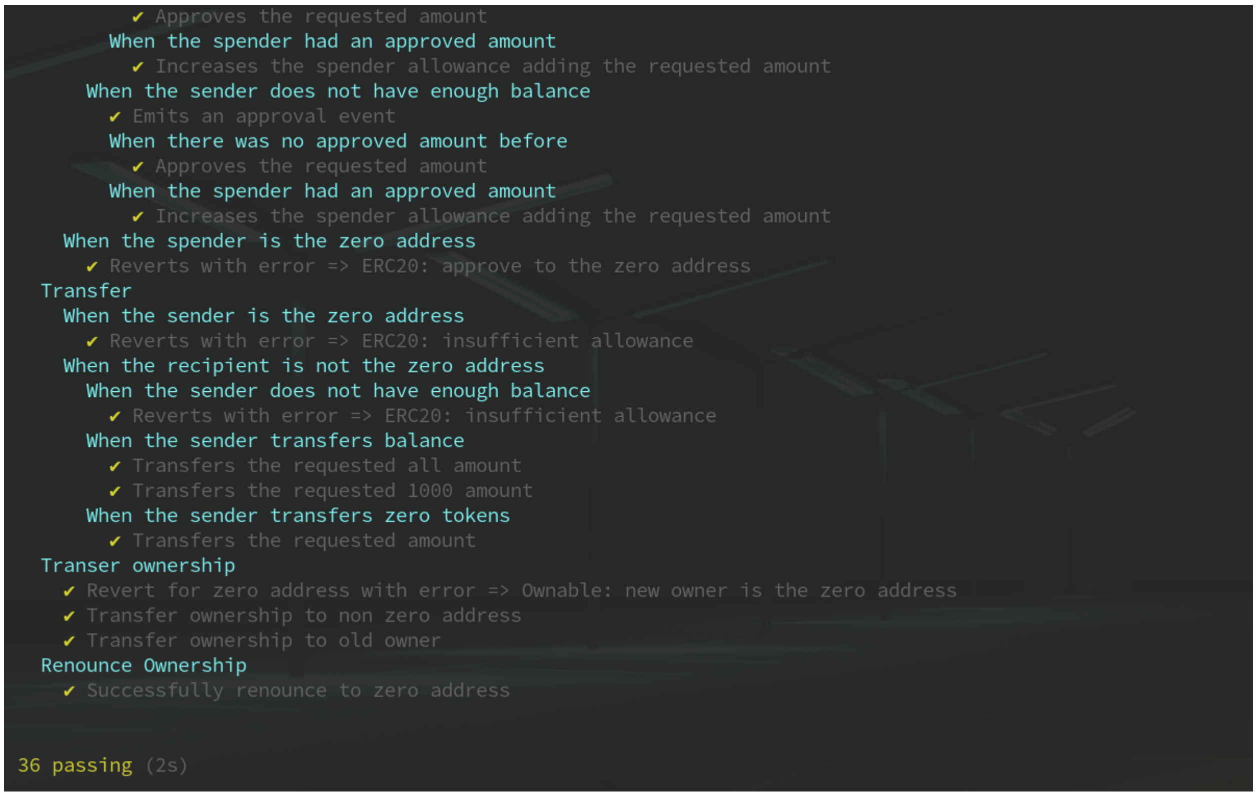


```
Echidna 2.0.0
Tests found: 11
Seed: -7598312177004772452
Unique instructions: 2485
Unique codehashes: 1
Corpus size: 14
Tests
crytic_less_than_total_ERC20Properties: PASSED!
crytic_transfer_to_other_ERC20PropertiesTransferable: PASSED!
crytic_revert_transfer_to_zero_ERC20PropertiesTransferable: PASSED!
crytic_revert_transfer_to_user_ERC20PropertiesTransferable: PASSED!
crytic_revert_transferFrom_to_zero_ERC20PropertiesTransferable: PASSED!
crytic_self_transferFrom_ERC20PropertiesTransferable: PASSED!
crytic_self_transfer_ERC20PropertiesTransferable: PASSED!
crytic_zero_always_empty_ERC20Properties: PASSED!
crytic_self_transferFrom_to_other_ERC20PropertiesTransferable: PASSED!
crytic_approve_overwrites: PASSED!
crytic_totalSupply_consistant_ERC20Properties: PASSED!
Campaign complete, C-c or esc to exit
```

Test Cases

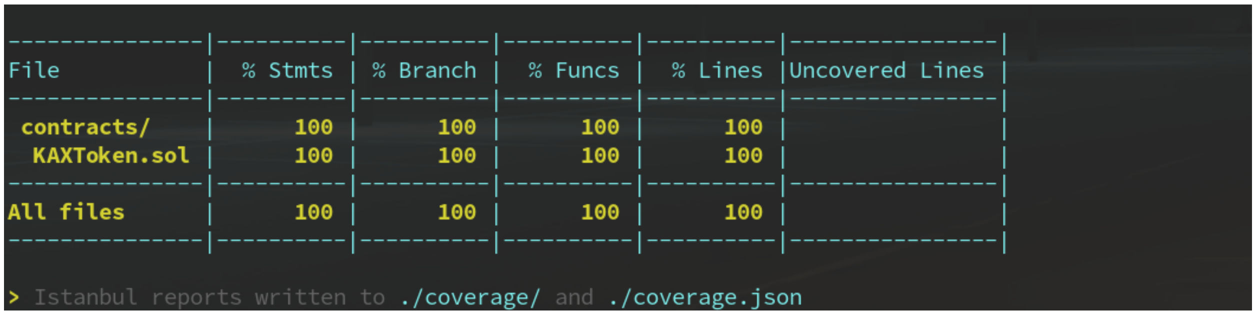


```
Contract: KAXToken
  ✔ Token name is KAXAA
  ✔ Token has a symbol KAX
  ✔ Token has 18 decimals
  ✔ Token has owner
  ✔ Token has 2 million initial supply
  Token Minting
    ✔ Reverts for zero address with error => ERC20: mint to the zero address  (51ms)
    For a non zero account
      ✔ Increments totalSupply with 50 token
      ✔ Increments owner balance with 50 token
      ✔ Emits Transfer event
  Decrease allowance
    When the spender is not the zero address
      When the sender has enough balance
        When there was no approved amount before
          ✔ Reverts with error => ERC20: decreased allowance below zero
        When the spender had an approved amount
          ✔ Emits an approval event
          ✔ Decreases the spender allowance subtracting the requested amount
          ✔ Sets the allowance to zero when all allowance is removed
          ✔ Reverts when more than the full allowance is removed with error => ERC20: decreased allowance below zero
      When the sender does not have enough balance
        When there was no approved amount before
          ✔ Reverts with error => ERC20: decreased allowance below zero
        When the spender had an approved amount
          ✔ Emits an approval event
          ✔ Decreases the spender allowance subtracting the requested amount
          ✔ Sets the allowance to zero when all allowance is removed
          ✔ Reverts when more than the full allowance is removed with error => ERC20: decreased allowance below zero
    When the spender is the zero address
      ✔ Reverts with error => ERC20: decreased allowance below zero
  Increase allowance
    When the spender is not the zero address
      When the sender has enough balance
        ✔ Emits an approval event
        When there was no approved amount before
          ✔ Approves the requested amount
        When the spender had an approved amount
          ✔ Increases the spender allowance adding the requested amount
```



```
          ✔ Approves the requested amount
        When the spender had an approved amount
          ✔ Increases the spender allowance adding the requested amount
      When the sender does not have enough balance
        ✔ Emits an approval event
        When there was no approved amount before
          ✔ Approves the requested amount
        When the spender had an approved amount
          ✔ Increases the spender allowance adding the requested amount
    When the spender is the zero address
      ✔ Reverts with error => ERC20: approve to the zero address
  Transfer
    When the sender is the zero address
      ✔ Reverts with error => ERC20: insufficient allowance
    When the recipient is not the zero address
      When the sender does not have enough balance
        ✔ Reverts with error => ERC20: insufficient allowance
      When the sender transfers balance
        ✔ Transfers the requested all amount
        ✔ Transfers the requested 1000 amount
      When the sender transfers zero tokens
        ✔ Transfers the requested amount
  Transer ownership
    ✔ Revert for zero address with error => Ownable: new owner is the zero address
    ✔ Transfer ownership to non zero address
    ✔ Transfer ownership to old owner
  Renounce Ownership
    ✔ Successfully renounce to zero address


36 passing (2s)
```

Code Coverage



```
----------------|----------|----------|----------|----------|----------------|
File            |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
----------------|----------|----------|----------|----------|----------------|
 contracts/     |      100 |      100 |      100 |      100 |                |
  KAXToken.sol  |      100 |      100 |      100 |      100 |                |
----------------|----------|----------|----------|----------|----------------|
All files       |      100 |      100 |      100 |      100 |                |
----------------|----------|----------|----------|----------|----------------|

> Istanbul reports written to ./coverage/ and ./coverage.json
```

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.

Concluding Remarks

While conducting the audits of the Kaxaa smart contract, it was observed that the contracts contain no severities.

The recommendations given will improve the operations of the smart contract.

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 Kaxaa 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