Ekta

NFT Marketplace

Smart Contract Audit
Final Report

August 04, 2022

Introduction 3
About Ekta 3
About ImmuneBytes 3
Documentation Details 3
Audit Process & Methodology 4
Audit Details 4
Audit Goals 5
Security Level Reference 5
Admin/Owner Privileges 6
High Severity Issues 6
Medium Severity Issues 6
Low Severity Issues 7
Recommendation / Informational 8
Automated Audit Result 10
Slither 10
Mythril 11
Maian Solidity Analysis 12
Maian Bytecode Analysis 17
Concluding Remarks 21
Disclaimer 21

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.

Introduction

1. About Ekta

Ekta's vision is to create a world where blockchain technology is used to give everyone a chance to live a better life. A new ecosystem is needed, one where people from different backgrounds and socio-economic circumstances can participate freely, without the barriers and inefficiencies introduced by centralized governing bodies.

Ekta's mission is to bridge the blockchain world with the world we live in and to create value in both. This is accomplished through various branches of the Ekta ecosystem, which include:

- The tokenization of real-world assets through Ekta Chain and Ekta's self-developed NFT Marketplace.
- Ekta's decentralized credit platform allows all users to participate.
- Physical spaces such as the island chain are currently being developed in Indonesia, where physical land and real estate assets will be brought on-chain.
- Ekta's startup incubator and innovation center are open to retail investors.

Visit https://ekta.io/ 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 Ekta team has provided the following doc for the purpose of audit:

1. https://whitepaper.ektaworld.io/

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.

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 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: Ekta
- Contracts Name: KaxToken.sol
- Languages: Solidity(Smart contract), Typescript (Unit Testing)
- Link for codebase for audit: https://github.com/Blockchainxtech/EKTA-NFT-Marketplace/tree/v2
- Final Commit: b33aa13c5cbffea138eac59dc9a455ae4759c749
- Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Contract Library, Slither, SmartCheck

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.

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	1	2	3

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.

Admin/Owner Privileges

--

High Severity Issues

1. **Re-entrency attack**
   Contract: PrivateSale.sol

   Description:
   The function withdrawFundRaised doesn't follow the CEI pattern, and the re-entrency can be achieved by making the transfer function of ektaRevenueWallet malicious.

   Recommendation:
   Use the CEI pattern always for implementation.

   Amended (August 04th, 2022): The issue has been fixed by the team and is no longer present in commit b33aa13c5cbffea138eac59dc9a455ae4759c749.

Medium Severity Issues

1. **Call vs. transfer**

   Description:
   Instead of transfer, use call because the function call() is designed for more individual interactions between smart contracts. It can call a function by name and send Ether to it. It is now the recommended way of sending Ether from a smart contract. To remove the re-entrency attack, use the gas limit and check the return value of the success of the call.

   Recommendation:
   We recommend using call instead of transfer.

   Amended (August 04th, 2022): The issue has been fixed by the team and is no longer present in commit b33aa13c5cbffea138eac59dc9a455ae4759c749.

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.

2. **tokens can be bought any time**
   Contract: CrowdSale.sol

   Description:
   user can buy tokens by calling the receive of crowdsale at any time. which is very bad behavior and can raise various vulnerabilities.

   Recommendation:
   We Recommend removing the call to buy tokens from the receive function.

   Amended (August 04th, 2022): The issue has been fixed by the team and is no longer present in commit b33aa13c5cbffea138eac59dc9a455ae4759c749.

Low Severity Issues

1. **Pragma is not locked**

   Description:
   The pragma versions used in the contract are not locked. Consider using the latest versions among 0.8.13 for deploying the contracts and libraries as it does not compile for any other version and can be confusing for a developer. Solidity source files indicate the versions of the compiler they can be compiled with.

   pragma solidity ^0.8.0; // bad: compiles between 0.8.0 and 0.8.13
   pragma solidity 0.8.0; // good : compiles w 0.8.0 only but not the latest version
   pragma solidity 0.8.13; // best: compiles w 0.8.13

   Recommendation:
   Fix the solidity version by removing the caret symbol and use the best compiles method(specified version numbers).

   Amended (August 04th, 2022): The issue has been fixed by the team and is no longer present in commit b33aa13c5cbffea138eac59dc9a455ae4759c749.

2. **Input checks are missing**

   Description:
   initialize, setSaleRevenue, setTradeRevenue for NFT registry, and initialize for LaunchPadFactory has input check missing.

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.

Recommendation:
We recommend using the require checks for valid input parameters.

Amended (August 04th, 2022): The issue has been fixed by the team and is no longer present in commit b33aa13c5cbffea138eac59dc9a455ae4759c749.

3. **Address should be hardcoded correctly**

Description:
WEKTA address should be hardcoded correctly in exchange.sol

Recommendation:
We recommend hardcoding the WEKTA address correctly as per the mainnet in the code.

Amended (August 04th, 2022): The issue has been fixed by the team and is no longer present in commit b33aa13c5cbffea138eac59dc9a455ae4759c749.

Recommendation / Informational

1. **Bad naming**

Description:
The variable name of the function is very bad we recommend following OpenZeppelin guidelines.

Recommendation:
All the internal functions should start with __(underscore).

Amended (August 04th, 2022): The issue has been fixed by the team and is no longer present in commit b33aa13c5cbffea138eac59dc9a455ae4759c749.

2. **Gas optimization and best practice**
   - The constructor initialize modifier is just a waste of gas on the constructor. It should get removed from all the upgradable contracts.
   - supportsInterface in erc1155.sol and erc721.sol, has closed in timedCrowdsale.sol, initialize in launchpadFactory, should be made external, and gas cost should be saved.
   - calculateRevenue in exchange.sol should not include the inbuilt safeMath in division operation as it's a positive quantity, so we should include the unchecked flag and save the gas for the inbuilt wrapper.

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.

- createERC721Token and createERC1155Token contain the redundant code in MarketplaceFactory. We recommend moving the redundant code into a helper function to improve readability and decrease deployment costs.
- The pre-increment operation is cheaper (about 5 GAS per iteration) so use ++i instead of i++ or i+= 1 in for loop. We recommend using pre-increment in all the for loops.
- Instead of using the && operator in a single require statement to check multiple conditions, using multiple require statements with 1 condition per require statement will save 3 GAS per &&. We recommend implementing it in all the contracts.
- In for loop, the default value initialization to 0 should not be there removed from all the for loop
- When the state gets updated, the event should always get fired. We recommend firing the events for all the state changes.
- All internal function names should get started with _ as a good practice to include.
- receive function is used in crowdsale, but natspec @dev states fallback function. Note that there is a significant difference between the fallback function and receive functions. We recommend using the correct name in the natspec comment.
- != 0 costs 6 less GAS compared to > 0 for unsigned integers in require statements with the optimizer enabled. We recommend to use !=0 instead of > 0 in all the contracts.
- In the EVM, there is no opcode for non-strict inequalities (>=, <=), and two operations are performed (> + =.) Consider replacing >= with the strict counterpart >. Recommend to follow the inequality with strict one.
- If the frontend does not read the constants, then we recommend making all the public constant variables private constant variables, which helps in the decrement of deployment gas.

Amended (August 04th, 2022): The issue has been fixed by the team and is no longer present in commit b33aa13c5cbffea138eac59dc9a455ae4759c749.

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.

Automated Audit Result

Slither

```
Compiled with solc
Number of lines: 1364 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 14 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 3
Number of informational issues: 78
Number of low issues: 11
Number of medium issues: 0
Number of high issues: 5

+-------------------------+-------------+------+------------+--------------+---------------+
|          Name           | # functions | ERCS | ERC20 info | Complex code |   Features    |
+-------------------------+-------------+------+------------+--------------+---------------+
|   AddressUpgradeable    |      9      |      |            |      No      |   Send ETH    |
|                         |             |      |            |              |   Assembly    |
|   IBeaconUpgradeable    |      1      |      |            |      No      |               |
| StorageSlotUpgradeable  |      4      |      |            |      No      |   Assembly    |
|    IEktaNftRegistry     |      3      |      |            |      No      |               |
|      PrivateSale        |      65     |      |            |      No      | Receive ETH   |
|                         |             |      |            |              |   Send ETH    |
|                         |             |      |            |              | Delegatecall  |
|                         |             |      |            |              | Upgradeable   |
+-------------------------+-------------+------+------------+--------------+---------------+
PrivateSale_flat.sol analyzed (14 contracts)
```

```
Compiled with solc
Number of lines: 1004 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 10 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 3
Number of informational issues: 73
Number of low issues: 2
Number of medium issues: 0
Number of high issues: 5

+-------------------------+-------------+------+------------+--------------+---------------+
|          Name           | # functions | ERCS | ERC20 info | Complex code |   Features    |
+-------------------------+-------------+------+------------+--------------+---------------+
|   AddressUpgradeable    |      9      |      |            |      No      |   Send ETH    |
|                         |             |      |            |              |   Assembly    |
|   IBeaconUpgradeable    |      1      |      |            |      No      |               |
| StorageSlotUpgradeable  |      4      |      |            |      No      |   Assembly    |
|    EktaNftRegistry      |      44     |      |            |      No      | Receive ETH   |
|                         |             |      |            |              | Delegatecall  |
|                         |             |      |            |              | Upgradeable   |
+-------------------------+-------------+------+------------+--------------+---------------+
NftRegistry_flat.sol analyzed (10 contracts)
```

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.

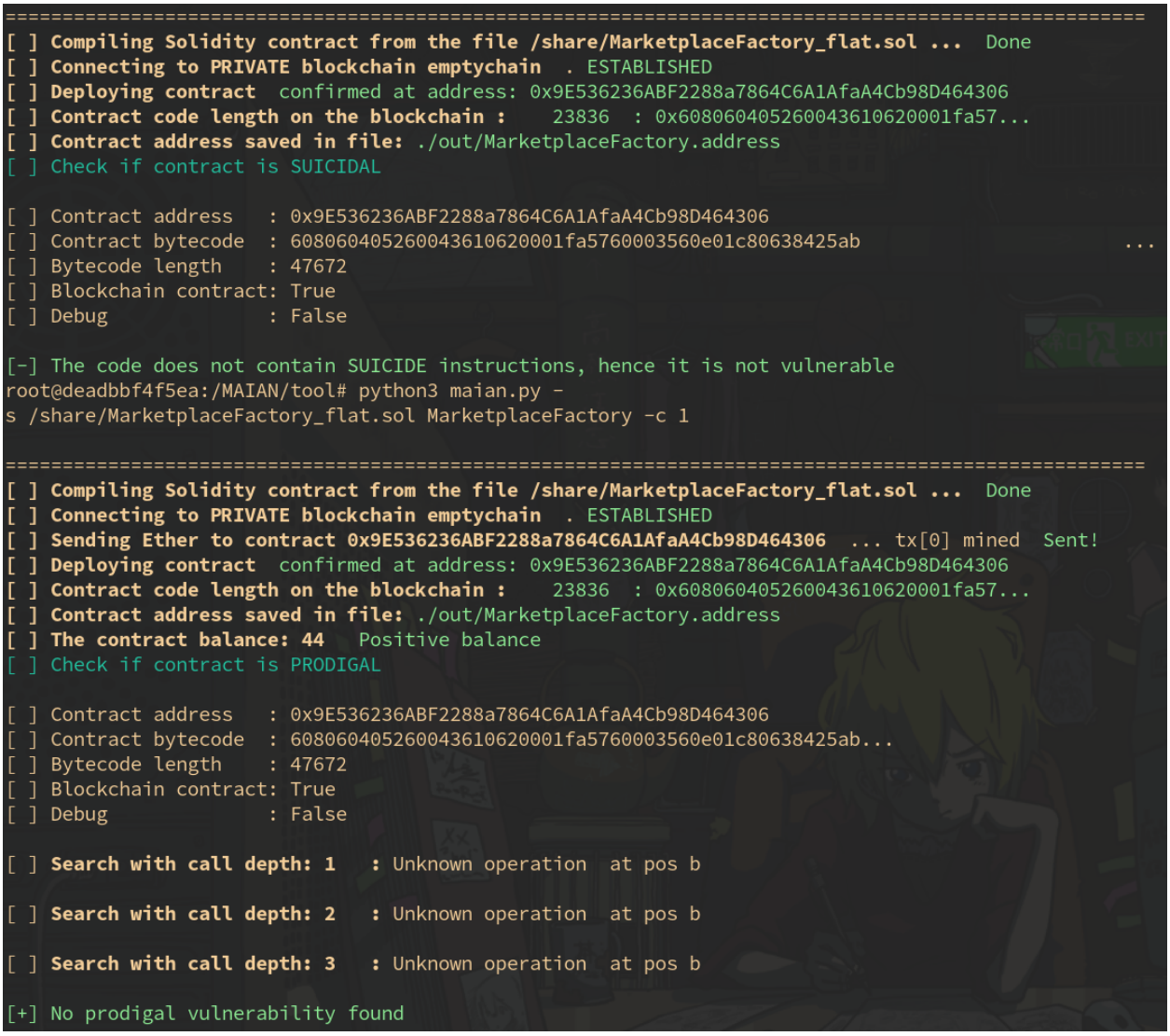
```
Compiled with solc
Number of lines: 2258 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 23 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 4
Number of informational issues: 131
Number of low issues: 15
Number of medium issues: 4
Number of high issues: 6

+-------------------------+-------------+------+------------+--------------+---------------+
|          Name           | # functions | ERCS | ERC20 info | Complex code |   Features    |
+-------------------------+-------------+------+------------+--------------+---------------+
|   AddressUpgradeable    |      9      |      |            |      No      |   Send ETH    |
|                         |             |      |            |              |   Assembly    |
|   IBeaconUpgradeable    |      1      |      |            |      No      |               |
| StorageSlotUpgradeable  |      4      |      |            |      No      |   Assembly    |
|        IBeacon          |      1      |      |            |      No      |               |
|        Address          |      11     |      |            |      No      |   Send ETH    |
|                         |             |      |            |              | Delegatecall  |
|                         |             |      |            |              |   Assembly    |
|      StorageSlot        |      4      |      |            |      No      |   Assembly    |
|    EKTABeaconProxy      |      23     |      |            |      No      | Receive ETH   |
|                         |             |      |            |              |   Assembly    |
|                         |             |      |            |              |    Proxy      |
|    IEktaNftRegistry     |      3      |      |            |      No      |               |
|      PrivateSale        |      65     |      |            |      No      | Receive ETH   |
|                         |             |      |            |              |   Send ETH    |
|                         |             |      |            |              | Delegatecall  |
|                         |             |      |            |              | Upgradeable   |
|    LaunchpadFactory     |      51     |      |            |      No      | Receive ETH   |
|                         |             |      |            |              | Delegatecall  |
|                         |             |      |            |              |   Assembly    |
|                         |             |      |            |              | Upgradeable   |
+-------------------------+-------------+------+------------+--------------+---------------+
```

Mythril

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

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.

Maian Solidity Analysis



```
================================================================================================
[ ] Compiling Solidity contract from the file /share/MarketplaceFactory_flat.sol ...  Done
[ ] Connecting to PRIVATE blockchain emptychain  . ESTABLISHED
[ ] Deploying contract  confirmed at address: 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract code length on the blockchain :    23836  : 0x6080604052600436106200001fa57...
[ ] Contract address saved in file: ./out/MarketplaceFactory.address
[ ] Check if contract is SUICIDAL

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 6080604052600436106200001fa5760003560e01c80638425ab                    ...
[ ] Bytecode length    : 47672
[ ] Blockchain contract: True
[ ] Debug              : False

[-] The code does not contain SUICIDE instructions, hence it is not vulnerable
root@deadbbf4f5ea:/MAIAN/tool# python3 maian.py -
s /share/MarketplaceFactory_flat.sol MarketplaceFactory -c 1

================================================================================================
[ ] Compiling Solidity contract from the file /share/MarketplaceFactory_flat.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 :    23836  : 0x6080604052600436106200001fa57...
[ ] Contract address saved in file: ./out/MarketplaceFactory.address
[ ] The contract balance: 44   Positive balance
[ ] Check if contract is PRODIGAL

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 6080604052600436106200001fa5760003560e01c80638425ab...
[ ] Bytecode length    : 47672
[ ] Blockchain contract: True
[ ] Debug              : False

[ ] Search with call depth: 1   : Unknown operation  at pos b

[ ] Search with call depth: 2   : Unknown operation  at pos b

[ ] Search with call depth: 3   : Unknown operation  at pos b

[+] No prodigal vulnerability found
```

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.

```
root@deadbbf4f5ea:/MAIAN/tool# python3 maian.py -s /share/MarketplaceFactory_flat.sol MarketplaceFactory -c 2

================================================================================================
[ ] Compiling Solidity contract from the file /share/MarketplaceFactory_flat.sol ...  Done
[ ] Connecting to PRIVATE blockchain emptychain  .. ESTABLISHED
[ ] Deploying contract  confirmed at address: 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract code length on the blockchain :    23836  : 0x6080604052600436106200001fa57...
[ ] Contract address saved in file: ./out/MarketplaceFactory.address
[ ] Check if contract is GREEDY

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 6080604052600436106200001fa5760003560e01c80638425ab...
[ ] Bytecode length    : 47672
[ ] Debug              : False
Unknown operation  at pos b
[-] Contract can receive Ether

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

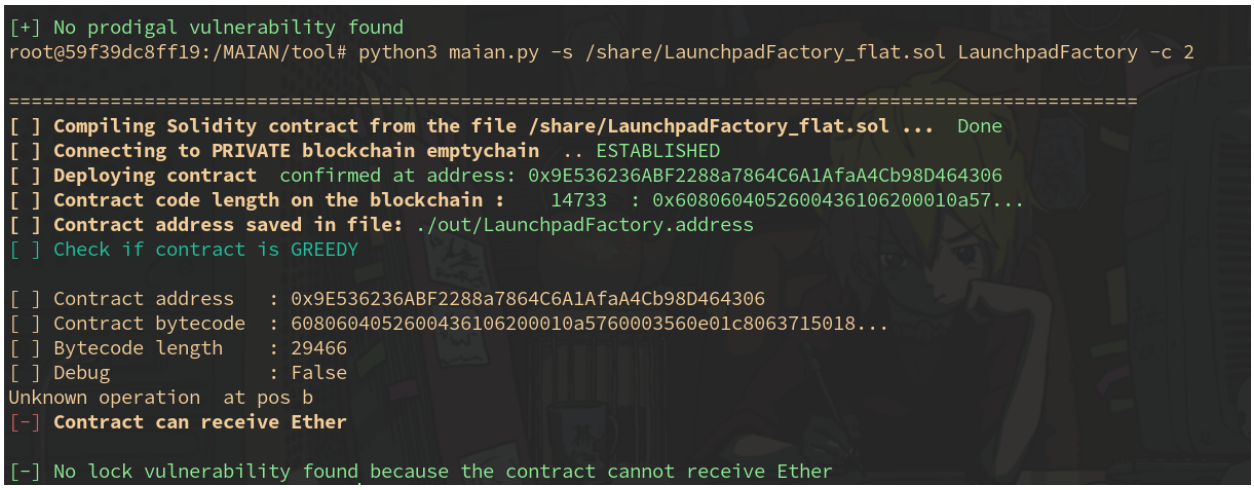
```
root@59f39dc8ff19:/MAIAN/tool# python3 maian.py -s /share/NftRegistry_flat.sol EktaNftRegistry -c 0

================================================================================================
[ ] Compiling Solidity contract from the file /share/NftRegistry_flat.sol ...  Done
[ ] Connecting to PRIVATE blockchain emptychain  . ESTABLISHED
[ ] Deploying contract  confirmed at address: 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract code length on the blockchain :    8760  : 0x6080604052600436106100e85760...
[ ] Contract address saved in file: ./out/EktaNftRegistry.address
[ ] Check if contract is SUICIDAL

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 6080604052600436106100e85760003560e01c8063715018a6...
[ ] Bytecode length    : 17520
[ ] Blockchain contract: True
[ ] Debug              : False

[-] The code does not contain SUICIDE instructions, hence it is not vulnerable
root@59f39dc8ff19:/MAIAN/tool# python3 maian.py -s /share/NftRegistry_flat.sol EktaNftRegistry -c 1

================================================================================================
[ ] Compiling Solidity contract from the file /share/NftRegistry_flat.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 :    8760  : 0x6080604052600436106100e85760...
[ ] Contract address saved in file: ./out/EktaNftRegistry.address
[ ] The contract balance: 44   Positive balance
[ ] Check if contract is PRODIGAL

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 6080604052600436106100e85760003560e01c8063715018a6...
[ ] Bytecode length    : 17520
[ ] Blockchain contract: True
[ ] Debug              : False
[+] The code does not have CALL/SUICIDE, hence it is not prodigal
```

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.

```
root@59f39dc8ff19:/MAIAN/tool# python3 maian.py -s /share/NftRegistry_flat.sol EktaNftRegistry -c 2

================================================================================================
[ ] Compiling Solidity contract from the file /share/NftRegistry_flat.sol ...  Done
[ ] Connecting to PRIVATE blockchain emptychain  .. ESTABLISHED
[ ] Deploying contract  confirmed at address: 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract code length on the blockchain :    8760  : 0x6080604052600436106100e85760...
[ ] Contract address saved in file: ./out/EktaNftRegistry.address
[ ] Check if contract is GREEDY

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 6080604052600436106100e85760003560e01c8063715018a6...
[ ] Bytecode length    : 17520
[ ] Debug              : False
Unknown operation  at pos b
[-] Contract can receive Ether

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

```
root@59f39dc8ff19:/MAIAN/tool# python3 maian.py -s /share/LaunchpadFactory_flat.sol LaunchpadFactory -c 0

================================================================================================
[ ] Compiling Solidity contract from the file /share/LaunchpadFactory_flat.sol ...  Done
[ ] Connecting to PRIVATE blockchain emptychain  . ESTABLISHED
[ ] Deploying contract  confirmed at address: 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract code length on the blockchain :    14733  : 0x6080604052600436106200010a57...
[ ] Contract address saved in file: ./out/LaunchpadFactory.address
[ ] Check if contract is SUICIDAL

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 6080604052600436106200010a5760003560e01c8063715018...
[ ] Bytecode length    : 29466
[ ] Blockchain contract: True
[ ] Debug              : False

[-] The code does not contain SUICIDE instructions, hence it is not vulnerable
root@59f39dc8ff19:/MAIAN/tool# python3 maian.py -s /share/LaunchpadFactory_flat.sol LaunchpadFactory -c 1

================================================================================================
[ ] Compiling Solidity contract from the file /share/LaunchpadFactory_flat.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 :    14733  : 0x6080604052600436106200010a57...
[ ] Contract address saved in file: ./out/LaunchpadFactory.address
[ ] The contract balance: 44   Positive balance
[ ] Check if contract is PRODIGAL

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 6080604052600436106200010a5760003560e01c8063715018...
[ ] Bytecode length    : 29466
[ ] Blockchain contract: True
[ ] Debug              : False

[ ] Search with call depth: 1   : Unknown operation  at pos b

[ ] Search with call depth: 2   : Unknown operation  at pos b

[ ] Search with call depth: 3   : Unknown operation  at pos b
```

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.



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.

```
================================================================================================
[ ] Compiling Solidity contract from the file /share/PrivateSale_flat.sol ...  Done
[ ] Connecting to PRIVATE blockchain emptychain  . ESTABLISHED
[ ] Deploying contract  confirmed at address: 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract code length on the blockchain :    19318  : 0x6080604052600436106101fd5760...
[ ] Contract address saved in file: ./out/PrivateSale.address
[ ] Check if contract is SUICIDAL

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 6080604052600436106101fd5760003560e01c806374e7493b...
[ ] Bytecode length    : 38636
[ ] Blockchain contract: True
[ ] Debug              : False

[-] The code does not contain SUICIDE instructions, hence it is not vulnerable
root@fc4a3fc9b79b:/MAIAN/tool# python3 maian.py -s /share/PrivateSale_flat.sol PrivateSale -c 1

================================================================================================
[ ] Compiling Solidity contract from the file /share/PrivateSale_flat.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 :    19318  : 0x6080604052600436106101fd5760...
[ ] Contract address saved in file: ./out/PrivateSale.address
[ ] The contract balance: 44   Positive balance
[ ] Check if contract is PRODIGAL

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 6080604052600436106101fd5760003560e01c806374e7493b...
[ ] Bytecode length    : 38636
[ ] Blockchain contract: True
[ ] Debug              : False

[ ] Search with call depth: 1   : Unknown operation  at pos b

[ ] Search with call depth: 2   : Unknown operation  at pos b

[ ] Search with call depth: 3   : Unknown operation  at pos b

[+] No prodigal vulnerability found
```

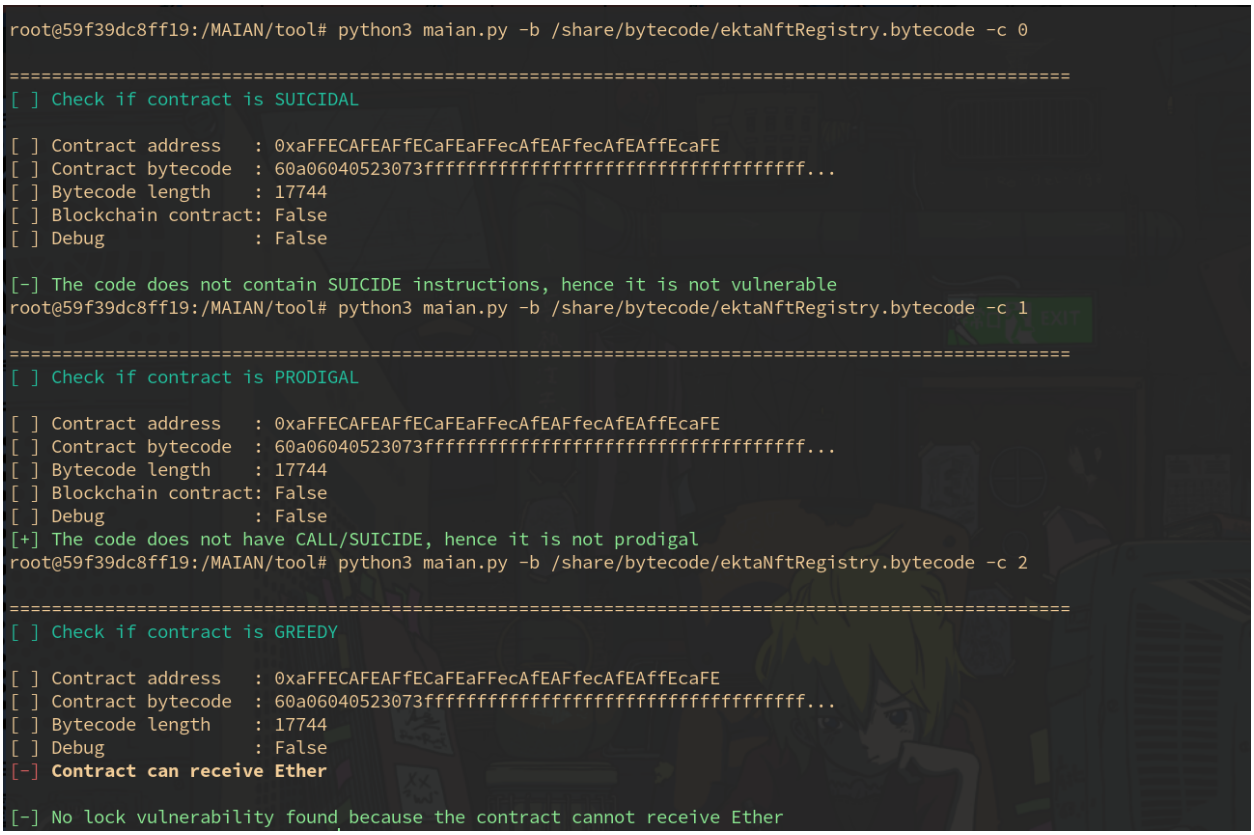
```
root@fc4a3fc9b79b:/MAIAN/tool# python3 maian.py -s /share/PrivateSale_flat.sol PrivateSale -c 2

================================================================================================
[ ] Compiling Solidity contract from the file /share/PrivateSale_flat.sol ...  Done
[ ] Connecting to PRIVATE blockchain emptychain  . ESTABLISHED
[ ] Deploying contract  confirmed at address: 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract code length on the blockchain :    19318  : 0x6080604052600436106101fd5760...
[ ] Contract address saved in file: ./out/PrivateSale.address
[ ] Check if contract is GREEDY

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 6080604052600436106101fd5760003560e01c806374e7493b...
[ ] Bytecode length    : 38636
[ ] Debug              : False
Unknown operation  at pos b
[-] 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.

Maian Bytecode Analysis

```
root@deadbbf4f5ea:/MAIAN/tool# python3 maian.py -b /share/bytecode/Marketplace.bytecode -c 0

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

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 60a06040523073ffffffffffffffffffffffffffffffffffff...
[ ] Bytecode length    : 47891
[ ] Blockchain contract: False
[ ] Debug              : False

[-] The code does not contain SUICIDE instructions, hence it is not vulnerable
root@deadbbf4f5ea:/MAIAN/tool# python3 maian.py -b /share/bytecode/Marketplace.bytecode -c 1

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

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 60a06040523073ffffffffffffffffffffffffffffffffffff...
[ ] Bytecode length    : 47891
[ ] Blockchain contract: False
[ ] Debug              : False

[ ] Search with call depth: 1   : 1
[ ] Search with call depth: 2   : 1
[ ] Search with call depth: 3   : 1
[+] No prodigal vulnerability found
root@deadbbf4f5ea:/MAIAN/tool# python3 maian.py -b /share/bytecode/Marketplace.bytecode -c 2

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

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 60a06040523073ffffffffffffffffffffffffffffffffffff...
[ ] Bytecode length    : 47891
[ ] 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.



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.

```
root@59f39dc8ff19:/MAIAN/tool# python3 maian.py -b /share/bytecode/LaunchpadFactory.bytecode -c 0

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

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 60a06040523073ffffffffffffffffffffffffffffffffffffffff...
[ ] Bytecode length    : 29684
[ ] Blockchain contract: False
[ ] Debug              : False

[-] The code does not contain SUICIDE instructions, hence it is not vulnerable
root@59f39dc8ff19:/MAIAN/tool# python3 maian.py -b /share/bytecode/LaunchpadFactory.bytecode -c 1

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

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 60a06040523073ffffffffffffffffffffffffffffffffffffffff...
[ ] Bytecode length    : 29684
[ ] Blockchain contract: False
[ ] Debug              : False

[ ] Search with call depth: 1   : 1
[ ] Search with call depth: 2   : 1
[ ] Search with call depth: 3   : 1
[+] No prodigal vulnerability found
root@59f39dc8ff19:/MAIAN/tool# python3 maian.py -b /share/bytecode/LaunchpadFactory.bytecode -c 2

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

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 60a06040523073ffffffffffffffffffffffffffffffffffffffff...
[ ] Bytecode length    : 29684
[ ] 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.

```
root@fc4a3fc9b79b:/MAIAN/tool# python3 maian.py -b /share/bytecode/PrivateSale.bytecode -c 0

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

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 60a06040523073ffffffffffffffffffffffffffffffffffff...
[ ] Bytecode length    : 38850
[ ] Blockchain contract: False
[ ] Debug              : False

[-] The code does not contain SUICIDE instructions, hence it is not vulnerable
root@fc4a3fc9b79b:/MAIAN/tool# python3 maian.py -b /share/bytecode/PrivateSale.bytecode -c 1

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

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 60a06040523073ffffffffffffffffffffffffffffffffffff...
[ ] Bytecode length    : 38850
[ ] Blockchain contract: False
[ ] Debug              : False

[ ] Search with call depth: 1   : 1
[ ] Search with call depth: 2   : 1
[ ] Search with call depth: 3   : 1
[+] No prodigal vulnerability found
root@fc4a3fc9b79b:/MAIAN/tool# python3 maian.py -b /share/bytecode/PrivateSale.bytecode -c 2

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

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 60a06040523073ffffffffffffffffffffffffffffffffffff...
[ ] Bytecode length    : 38850
[ ] 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.

Concluding Remarks

While conducting the audits of the Ekta smart contract, it was observed that the contracts contain High, Medium, and Low severity issues.

Our auditors suggest that High, Medium, and Low severity issues should be resolved by the developers. 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 to complement this audit is strongly recommended.

Our team does not endorse the Ekta 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