Prime Ape Planet

Shop

Smart Contract Audit
Final Report

July 08, 2022

Introduction

1. About Prime Ape Planet

Prime Ape Planet is a collection of 7,979 Primus Ethereum mammals created by Disney/Marvel artist Kurtis Dawe. Prime Ape Planet is the genesis collection of an ever-evolving ecosystem consisting of Prime Kong Planet, Infected Prime Apes, and Poisoned Bananas. Deep dive into the Prime Planet Ecosystem of limitless possibilities and earn rewards such as NFT Whitelists, Prime Tokens, Prime Merchandise, Blue Chip NFTs, and much more!

Visit https://primeapeplanet.com/ 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 Prime Ape Planet team has provided the following doc for the purpose of audit:

1. https://github.com/pap-github/pap-shop/blob/master/README.md

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: Prime Ape Project
- Contracts Name: Shop.sol, ERC20.sol, IShop.sol
- Languages: Solidity(Smart contract), Typescript (Unit Testing)
- Link for codebase for audit: https://github.com/pap-github/pap-shop
- Final Commit: c15a55d3c403c5e8b445885dbe0839771b61cc0a
- 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	-	-	3

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

1. **Violation of Check_Effects_Interaction Pattern in the Withdraw function**
   Status: CLOSED
   Line no - 132-134

   Explanation:
   The Shop contract includes a few functions that update some of the crucial state variables of the contract after the external calls are being made.

   Although the call is made to a token address that is set by the contract deployer, it is still recommended, as per the Solidity Guidelines, that any modification of the state variables in the base contract must be performed before executing the external call.

   The following function in the contract updates the state variables after making an external call at the lines mentioned below:

   - buy() at Line 132-134

```
131
132     _token.safeTransferFrom(msg.sender, _recipient, item.price);
133     item.sold++;
134     _userItemCount[msg.sender][scSaleId]++;
135
```

   Recommendation:
   Check Effects Interaction Pattern must be followed while implementing external calls in a function.

   Amended (July 08th, 2022): The issue has been amended by the PAP team in commit c15a55d3c403c5e8b445885dbe0839771b61cc0a.

2. **Redundant function trigger could be avoided in changeStatus() function.**
   Status: CLOSED
   Line no: 103-118
   Explanation:
   The changeStatus() function allows the shop owner to change the status of a specific item.

   However, the function doesn't validate whether or not the boolean value being passed as the status of the item is already assigned to the item.

   Recommendation:
   It is always considered a better practice to validate the arguments being passed to avoid any redundant transaction. Therefore, adding the following require statement should validate the inputs adequately

```
require(item.active != active, "Item is already in the same Status");
```

   Acknowledged (July 08th, 2022): The issue has been acknowledged by the PAP team in commit c15a55d3c403c5e8b445885dbe0839771b61cc0a.

   Comment from the team:
   "If there is a requirement to increase the number of items on sale, we might not want to stop the sale in the process, so status should not change and this suggested require would prevent such functionality."

3. **Absence of Zero Address Validation**
   Line no- 35
   Status: Not Considered

   Explanation:
   The Shop contract includes an initialize() function that updates an imperative address, i.e., _recipient, etc.

   However, during the code review, it was found that no Zero Address Validation is implemented in the function despite the fact this state variable shall be the receiver of all the funds whenever a user buys any specific item.

   Recommendation:
   A require statement should be included in such functions to ensure no zero address is passed in the arguments.

   Acknowledged (July 08th, 2022): The issue has been acknowledged by the PAP team in commit c15a55d3c403c5e8b445885dbe0839771b61cc0a.

Recommendation / Informational

1. **Different Solidity versions used in contract files.**

   Explanation:
   It was found that the contract files include different solidity versions which should be fixed.

   Recommendation:
   It is recommended to use one specific solidity version for all contracts.

   Amended (July 08th, 2022): The issue has been amended by the PAP team in commit c15a55d3c403c5e8b445885dbe0839771b61cc0a.

Automated Audit Result

Slither

```
Compiled with solc
Number of lines: 1204 (+ 0 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 12 (+ 0 in dependencies, + 0 tests)

Number of optimization issues: 3
Number of informational issues: 48
Number of low issues: 2
Number of medium issues: 2
Number of high issues: 1
ERCs: ERC165, ERC20

+----------------------+-------------+--------+--------------------+--------------+--------------------+
|         Name         | # functions | ERCS   |     ERC20 info     | Complex code |      Features      |
+----------------------+-------------+--------+--------------------+--------------+--------------------+
|  AddressUpgradeable  |      9      |        |                    |      No      |      Send ETH      |
|                      |             |        |                    |              |      Assembly      |
|  StringsUpgradeable  |      4      |        |                    |     Yes      |                    |
|  IERC20Upgradeable   |      6      | ERC20  |     No Minting     |      No      |                    |
|                      |             |        | Approve Race Cond. |              |                    |
|                      |             |        |                    |              |                    |
| SafeERC20Upgradeable |      6      |        |                    |      No      |      Send ETH      |
|                      |             |        |                    |              | Tokens interaction |
|         Shop         |      36     | ERC165 |                    |     Yes      |      Send ETH      |
|                      |             |        |                    |              |    Upgradeable     |
+----------------------+-------------+--------+--------------------+--------------+--------------------+
Flat.sol analyzed (12 contracts)
```

Mythril

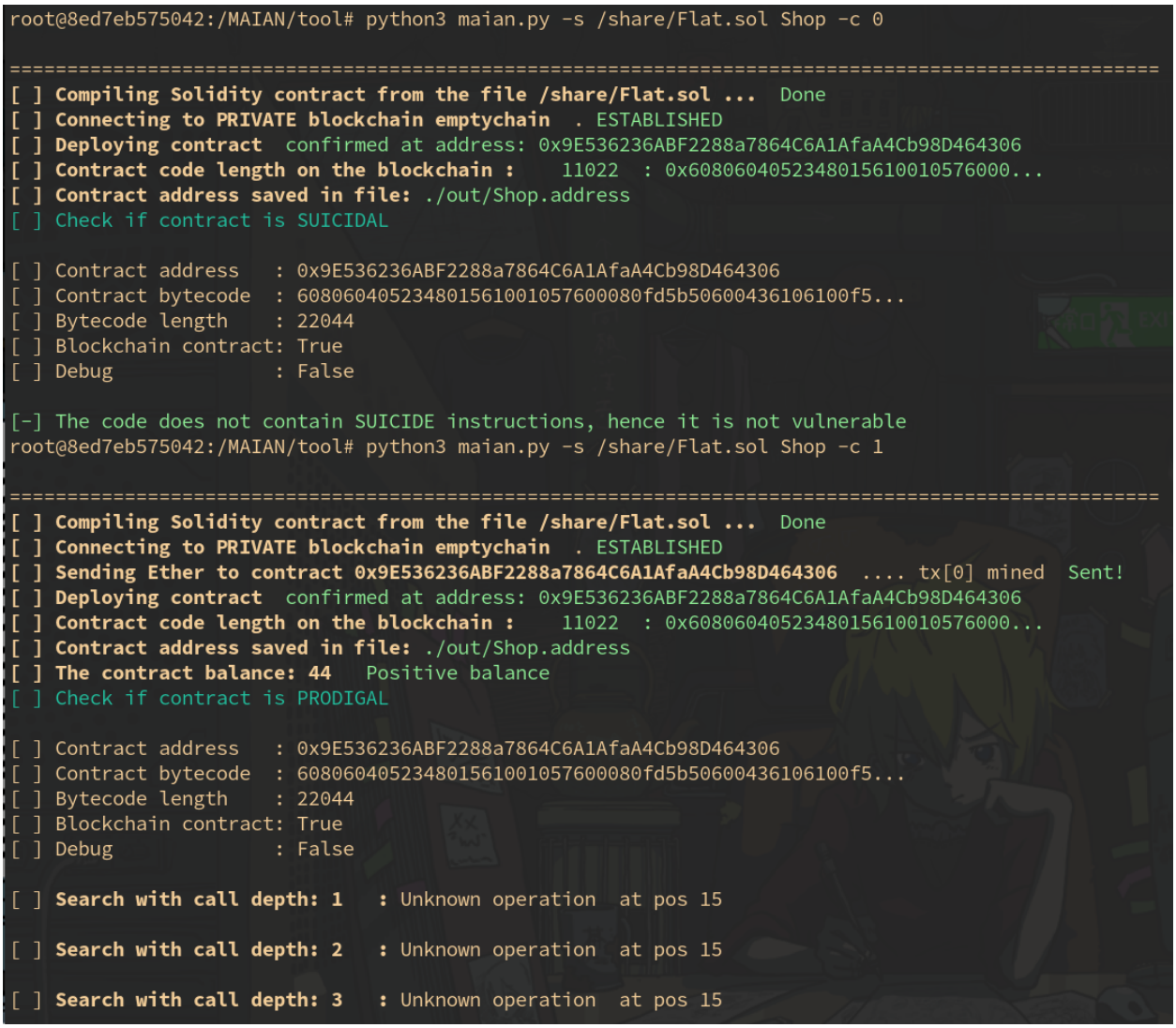
```
The analysis was completed successfully. No issues were detected.
```

Mythx

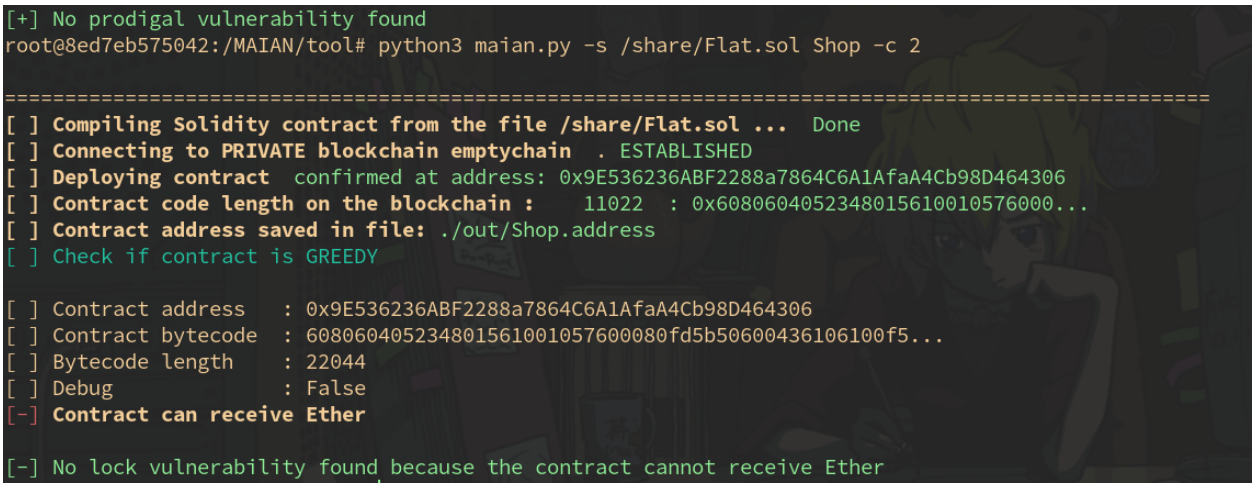
Report for Flat.sol
https://dashboard.mythx.io/#/console/analyses/055664b6-d079-4775-a13f-bcc94bfdb99d

Line	SWC Title	Severity	Short Description
6	(SWC-103) Floating Pragma	Low	A floating pragma is set.
47	(SWC-103) Floating Pragma	Low	A floating pragma is set.
246	(SWC-103) Floating Pragma	Low	A floating pragma is set.
328	(SWC-103) Floating Pragma	Low	A floating pragma is set.
420	(SWC-103) Floating Pragma	Low	A floating pragma is set.
460	(SWC-103) Floating Pragma	Low	A floating pragma is set.
531	(SWC-103) Floating Pragma	Low	A floating pragma is set.
560	(SWC-103) Floating Pragma	Low	A floating pragma is set.
604	(SWC-103) Floating Pragma	Low	A floating pragma is set.
842	(SWC-103) Floating Pragma	Low	A floating pragma is set.
928	(SWC-103) Floating Pragma	Low	A floating pragma is set.
1045	(SWC-108) State Variable Default Visibility	Low	State variable visibility is not set.

Maian Solidity Analysis



```
root@8ed7eb575042:/MAIAN/tool# python3 maian.py -s /share/Flat.sol Shop -c 0

================================================================================================
[ ] Compiling Solidity contract from the file /share/Flat.sol ...  Done
[ ] Connecting to PRIVATE blockchain emptychain  . ESTABLISHED
[ ] Deploying contract  confirmed at address: 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract code length on the blockchain :    11022  : 0x6080604052348015610010576000...
[ ] Contract address saved in file: ./out/Shop.address
[ ] Check if contract is SUICIDAL

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 6080604052348015610010576000080fd5b50600436106100f5...
[ ] Bytecode length    : 22044
[ ] Blockchain contract: True
[ ] Debug              : False

[-] The code does not contain SUICIDE instructions, hence it is not vulnerable
root@8ed7eb575042:/MAIAN/tool# python3 maian.py -s /share/Flat.sol Shop -c 1

================================================================================================
[ ] Compiling Solidity contract from the file /share/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 :    11022  : 0x6080604052348015610010576000...
[ ] Contract address saved in file: ./out/Shop.address
[ ] The contract balance: 44   Positive balance
[ ] Check if contract is PRODIGAL

[ ] Contract address   : 0x9E536236ABF2288a7864C6A1AfaA4Cb98D464306
[ ] Contract bytecode  : 6080604052348015610010576000080fd5b50600436106100f5...
[ ] Bytecode length    : 22044
[ ] Blockchain contract: True
[ ] Debug              : False

[ ] Search with call depth: 1   : Unknown operation  at pos 15

[ ] Search with call depth: 2   : Unknown operation  at pos 15

[ ] Search with call depth: 3   : Unknown operation  at pos 15
```



Maian Bytecode Analysis

```
root@8ed7eb575042:/MAIAN/tool# python3 maian.py -b /share/Flat.bytecode -c 0

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

[ ] Contract address    : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode   : =======Flat.sol:AccessControlUpgradeable=======Bin...
[ ] Bytecode length     : 23309
[ ] Blockchain contract: False
[ ] Debug               : False

[ ] Search with call depth: 1    : Unknown operation  at pos 0

[ ] Search with call depth: 2    : Unknown operation  at pos 0

[ ] Search with call depth: 3    : Unknown operation  at pos 0

[-] No suicidal vulnerability found
root@8ed7eb575042:/MAIAN/tool# python3 maian.py -b /share/Flat.bytecode -c 1

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

[ ] Contract address    : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode   : =======Flat.sol:AccessControlUpgradeable=======Bin...
[ ] Bytecode length     : 23309
[ ] Blockchain contract: False
[ ] Debug               : False

[ ] Search with call depth: 1    : Unknown operation  at pos 0

[ ] Search with call depth: 2    : Unknown operation  at pos 0

[ ] Search with call depth: 3    : Unknown operation  at pos 0

[+] No prodigal vulnerability found
root@8ed7eb575042:/MAIAN/tool# python3 maian.py -b /share/Flat.bytecode -c 2

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

```
[ ] Check if contract is GREEDY

[ ] Contract address    : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode   : =======Flat.sol:AccessControlUpgradeable=======Bin...
[ ] Bytecode length     : 23309
[ ] Debug               : False
Unknown operation  at pos 0
[-] Contract can receive Ether

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

Concluding Remarks

While conducting the audits of the Prime Ape Planet smart contracts, it was observed that the contracts contain only Low severity issues.

Our auditors suggest that the Low severity issues should be resolved by the developers. The recommendations given will improve the operations of the smart contract.

Notes

- ***The PAP team has fixed the issues based on the auditor's recommendation.***

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 Prime Ape Planet 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