Prime Ape Planet

NFTStaking

Smart Contract Audit
Final Report

June 13, 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 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-staking#readme

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: NFTStaking.sol
- Languages: Solidity(Smart contract), Typescript (Unit Testing)
- Link for codebase for audit: https://github.com/pap-github/pap-staking
- Final Commit: 4e29e54c8ff5b371f601b6c85653bbb753ba127b
- 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. **Contract owner is capable of assigning NO daily reward for any given NFT contract**
   Line no: 110-115

   Explanation:
   The whitelist function of the contract allows the contract owner to add any given nft contract address to the whitelisted category as well as set the amount of the daily reward for that given nft address.

```
110  function whitelist(address nftContract, uint256 dailyReward)  //@audit-issue
111    external
112    onlyRole(DEFAULT_ADMIN_ROLE)
113  {
114    whitelistedNFTs.add(nftContract);
115    whitelistedNFTRewards[nftContract] = dailyReward;
116  }
```

   However, due to the lack of input validations in the function body, the owner is capable of setting a zero value (no daily rewards) for any given address. This leads to an unwanted scenario where users staking the NFT shall receive no rewards at all for their tokens.

   Recommendation:
   If the above-mentioned function design is not intentional, it is recommended to attach adequate input validations in the function and ensure that no invalid nft addresses of daily reward values are passed as arguments to the function.

   Acknowledged (June 13th, 2022): The issue has been acknowledged by the PAP team in commit 4e29e54c8ff5b371f601b6c85653bbb753ba127b.

2. **announceWithdraw() function fails to represent a definitive withdrawal timestamp.**
   Line no: 121-124

   Explanation:
   While the announceWithdraw() function allows the owner to set and announce a withdrawal timestamp for token withdrawals, it doesn't represent a definitive announcement.

   Due to the lack of validations/checks in the function, the function can be repeatedly called any number of times after the first announcement, which could technically delay/shift the initially announced withdrawal timestamp.

   Recommendation:
   In order to represent a better function design, the function can be modified to include necessary require statement to ensure that it can only be triggered if the withdrawTimestamp state variable has not been initialized already.

   Ameneded (June 13th, 2022): The issue has been fixed by the PAP team and is no longer present in commit 4e29e54c8ff5b371f601b6c85653bbb753ba127b.

3. **Absence of Input Validations found**
   Line no - 31, 137, 155

   Explanation:
   During the review of the contract, it was found that some functions execute imperative state changes without validating the arguments passed to the functions.

   Some of the imperative functions that might need adequate input validations are as follows:

   a. rewardToken variable in the constructor().
   b. The range of PlanID enum is not validated in the _stake function.
   c. Absence of zero address validation of the _recipient passed as argument in the withdraw() function

   Recommendation:
   Adequate input validations must be included in the functions.

   Acknowledged (June 13th, 2022): The issue has been acknowledged by the PAP team in commit 4e29e54c8ff5b371f601b6c85653bbb753ba127b.

Recommendation / Informational

Note: It was found that Solidity versions differ in NFTStaking.sol and INftStaking.sol. It is recommended to use similar versions in all contract files.

1. **WithdrawAnnounced event emission can be improved**
   Line no: 123

   Explanation:
   The WithdrawAnnounced event doesn't emit the withdrawTimestamp value after the function execution.

   Functions that update an imperative arithmetic state variable contract should emit an event after the update. Additionally, the absence of event emission for important state variables update also makes it difficult to track them off-chain as well.

   Recommendation:
   As per the best practices in smart contract development, an event should be fired after changing crucial arithmetic state variables.

   Ameneded (June 13th, 2022): The issue has been fixed by the PAP team and is no longer present in commit 4e29e54c8ff5b371f601b6c85653bbb753ba127b.

2. **Unlocked Pragma statements found in the contracts**
   Line no: 2

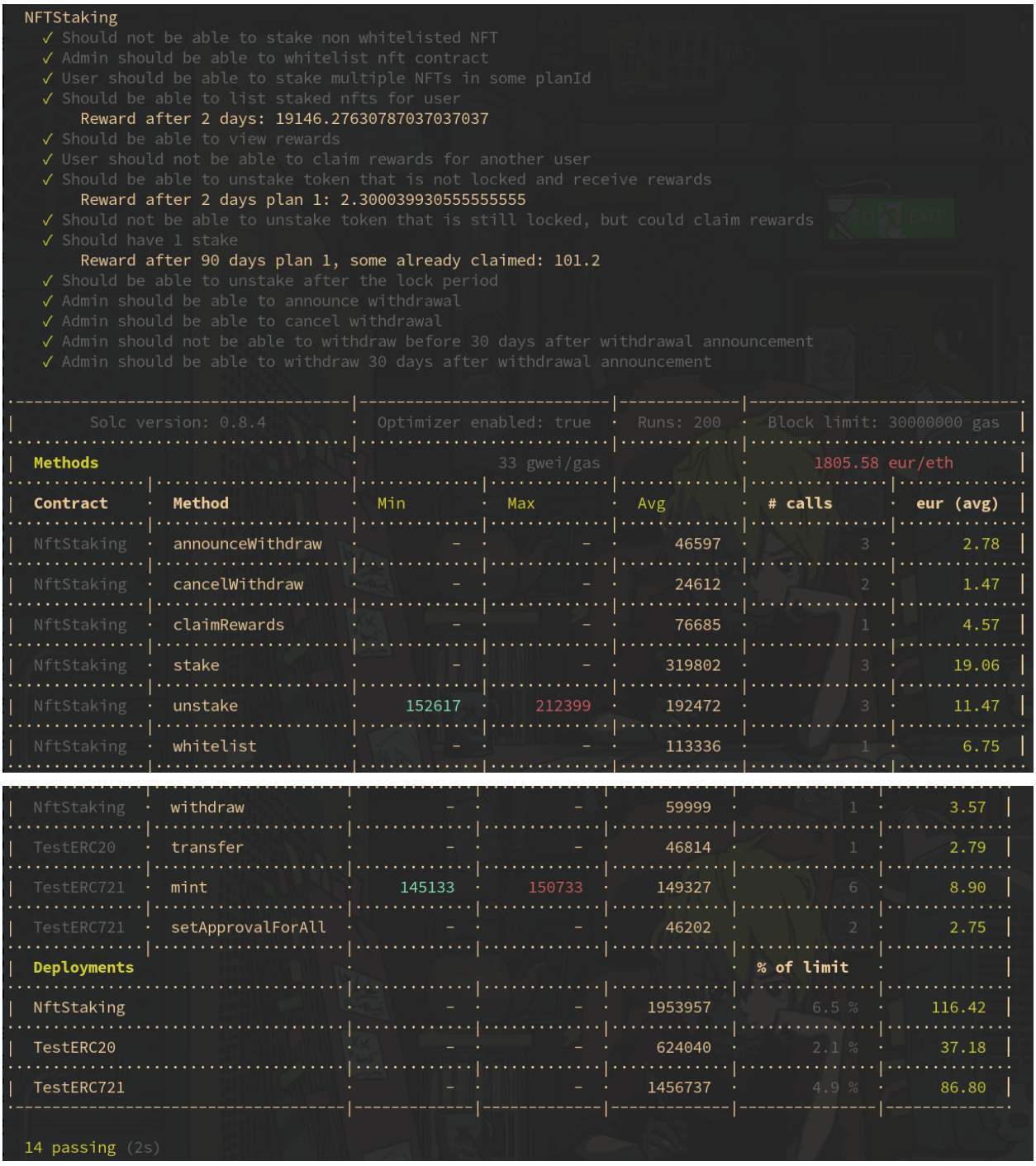
   Explanation:
   During the code review, it was found that the contracts included unlocked pragma solidity version statements.

   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.

   Ameneded (June 13th, 2022): The issue has been fixed by the PAP team and is no longer present in commit 4e29e54c8ff5b371f601b6c85653bbb753ba127b.

Unit Tests



```
NFTStaking
  ✓ Should not be able to stake non whitelisted NFT
  ✓ Admin should be able to whitelist nft contract
  ✓ User should be able to stake multiple NFTs in some planId
  ✓ Should be able to list staked nfts for user
      Reward after 2 days: 19146.27630787037037037
  ✓ Should be able to view rewards
  ✓ User should not be able to claim rewards for another user
  ✓ Should be able to unstake token that is not locked and receive rewards
      Reward after 2 days plan 1: 2.300039930555555555
  ✓ Should not be able to unstake token that is still locked, but could claim rewards
  ✓ Should have 1 stake
      Reward after 90 days plan 1, some already claimed: 101.2
  ✓ Should be able to unstake after the lock period
  ✓ Admin should be able to announce withdrawal
  ✓ Admin should be able to cancel withdrawal
  ✓ Admin should not be able to withdraw before 30 days after withdrawal announcement
  ✓ Admin should be able to withdraw 30 days after withdrawal announcement

·-------------------------------------|---------------------------|-------------|-----------------------------·
|        Solc version: 0.8.4          ·  Optimizer enabled: true  ·  Runs: 200  ·  Block limit: 30000000 gas  |
·-------------------------------------|---------------------------|-------------|-----------------------------·
|  Methods                            ·              33 gwei/gas  ·       1805.58 eur/eth       |
·-----------------|-------------------|-----------|-----------|-------------|--------------|-------------·
|  Contract       ·  Method           ·  Min      ·  Max      ·  Avg        ·  # calls     ·  eur (avg)  |
·-----------------|-------------------|-----------|-----------|-------------|--------------|-------------·
|  NftStaking     ·  announceWithdraw ·         -  ·         -  ·      46597  ·            3  ·       2.78  |
·-----------------|-------------------|-----------|-----------|-------------|--------------|-------------·
|  NftStaking     ·  cancelWithdraw   ·         -  ·         -  ·      24612  ·            2  ·       1.47  |
·-----------------|-------------------|-----------|-----------|-------------|--------------|-------------·
|  NftStaking     ·  claimRewards     ·         -  ·         -  ·      76685  ·            1  ·       4.57  |
·-----------------|-------------------|-----------|-----------|-------------|--------------|-------------·
|  NftStaking     ·  stake            ·         -  ·         -  ·     319802  ·            3  ·      19.06  |
·-----------------|-------------------|-----------|-----------|-------------|--------------|-------------·
|  NftStaking     ·  unstake          ·    152617  ·    212399  ·     192472  ·            3  ·      11.47  |
·-----------------|-------------------|-----------|-----------|-------------|--------------|-------------·
|  NftStaking     ·  whitelist        ·         -  ·         -  ·     113336  ·            1  ·       6.75  |
·-----------------|-------------------|-----------|-----------|-------------|--------------|-------------·
|  NftStaking     ·  withdraw         ·         -  ·         -  ·      59999  ·            1  ·       3.57  |
·-----------------|-------------------|-----------|-----------|-------------|--------------|-------------·
|  TestERC20      ·  transfer         ·         -  ·         -  ·      46814  ·            1  ·       2.79  |
·-----------------|-------------------|-----------|-----------|-------------|--------------|-------------·
|  TestERC721     ·  mint             ·    145133  ·    150733  ·     149327  ·            6  ·       8.90  |
·-----------------|-------------------|-----------|-----------|-------------|--------------|-------------·
|  TestERC721     ·  setApprovalForAll·         -  ·         -  ·      46202  ·            2  ·       2.75  |
·-----------------|-------------------|-----------|-----------|-------------|--------------|-------------·
|  Deployments                        ·                                     ·  % of limit   ·             |
·-------------------------------------|-----------|-----------|-------------|--------------|-------------·
|  NftStaking                         ·         -  ·         -  ·    1953957  ·       6.5 %  ·     116.42  |
·-------------------------------------|-----------|-----------|-------------|--------------|-------------·
|  TestERC20                          ·         -  ·         -  ·     624040  ·       2.1 %  ·      37.18  |
·-------------------------------------|-----------|-----------|-------------|--------------|-------------·
|  TestERC721                         ·         -  ·         -  ·    1456737  ·       4.9 %  ·      86.80  |
·-------------------------------------|-----------|-----------|-------------|--------------|-------------·

  14 passing (2s)
```

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.

Code Coverage

```
No need to generate any newer typings.


  NFTStaking
    ✔ Should not be able to stake non whitelisted NFT
    ✔ Admin should be able to whitelist nft contract
    ✔ User should be able to stake multiple NFTs in some planId (74ms)
    ✔ Should be able to list staked nfts for user
        Reward after 2 days: 19146.275787037037037037
    ✔ Should be able to view rewards
    ✔ User should not be able to claim rewards for another user
    ✔ Should be able to unstake token that is not locked and receive rewards (59ms)
        Reward after 2 days plan 1: 2.30005324074074074
    ✔ Should not be able to unstake token that is still locked, but could claim rewards (52ms)
    ✔ Should have 1 stake
        Reward after 90 days plan 1, some already claimed: 101.2
    ✔ Should be able to unstake after the lock period (48ms)
    ✔ Admin should be able to announce withdrawal
    ✔ Admin should be able to cancel withdrawal
    ✔ Admin should not be able to withdraw before 30 days after withdrawal announcement
    ✔ Admin should be able to withdraw 30 days after withdrawal announcement


  14 passing (804ms)

------------------|----------|----------|----------|----------|----------------|
File              |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
------------------|----------|----------|----------|----------|----------------|
 staking/         |      100 |    83.33 |      100 |      100 |                |
  INftStaking.sol |      100 |      100 |      100 |      100 |                |
  NftStaking.sol  |      100 |    83.33 |      100 |      100 |                |
 tokens/          |      100 |      100 |      100 |      100 |                |
  ERC20.sol       |      100 |      100 |      100 |      100 |                |
  ERC721.sol      |      100 |      100 |      100 |      100 |                |
------------------|----------|----------|----------|----------|----------------|
All files         |      100 |    83.33 |      100 |      100 |                |
------------------|----------|----------|----------|----------|----------------|
```

Automated Audit Result

Slither

Name	# functions	ERCS	ERC20 info	Complex code	Features
IERC20	6	ERC20	No Minting Approve Race Cond.	No	
Address	11			No	Send ETH Delegatecall Assembly
SafeERC20	6			No	Send ETH Tokens interaction
IERC721	10	ERC165,ERC721		No	
EnumerableSet	24			No	Assembly
Strings	4			Yes	
NftStaking	40	ERC165		Yes	Send ETH Tokens interaction

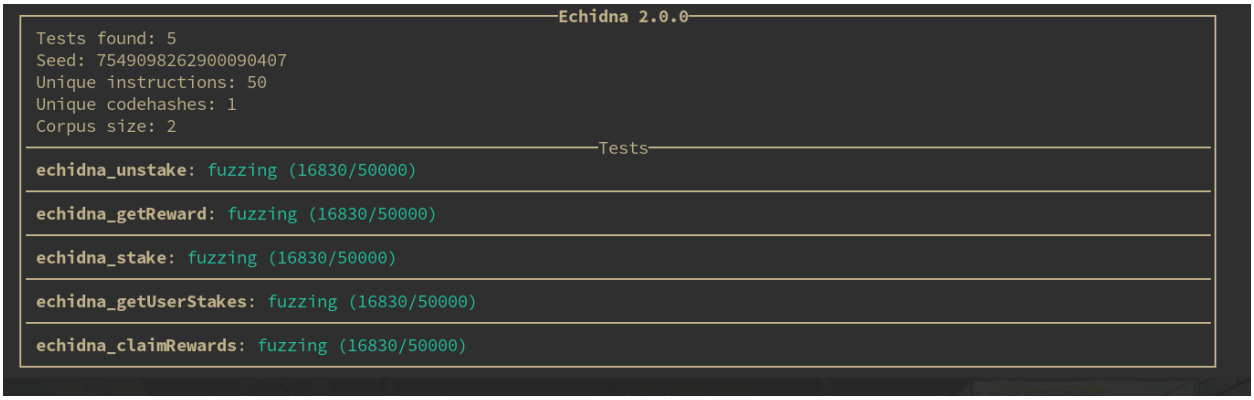
Mythril

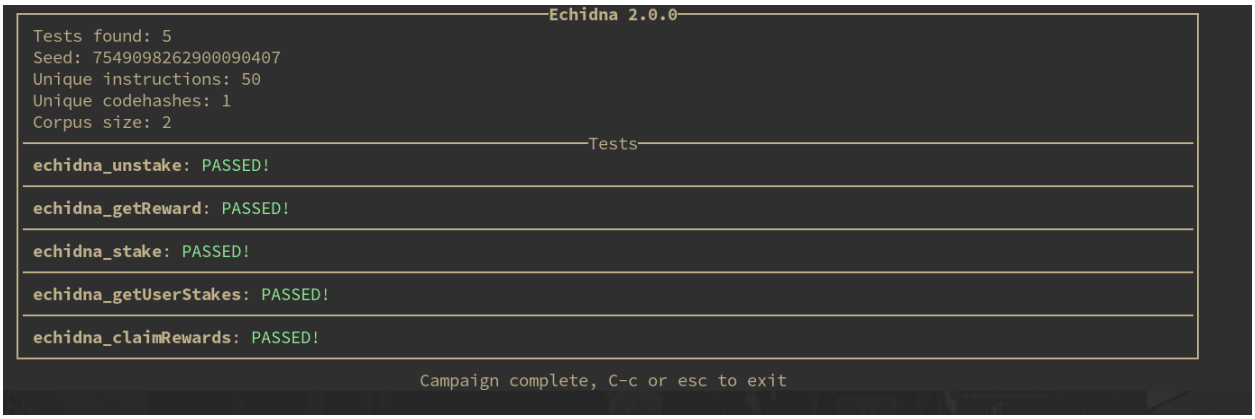
```
The analysis was completed successfully. No issues were detected.
```

Mythx

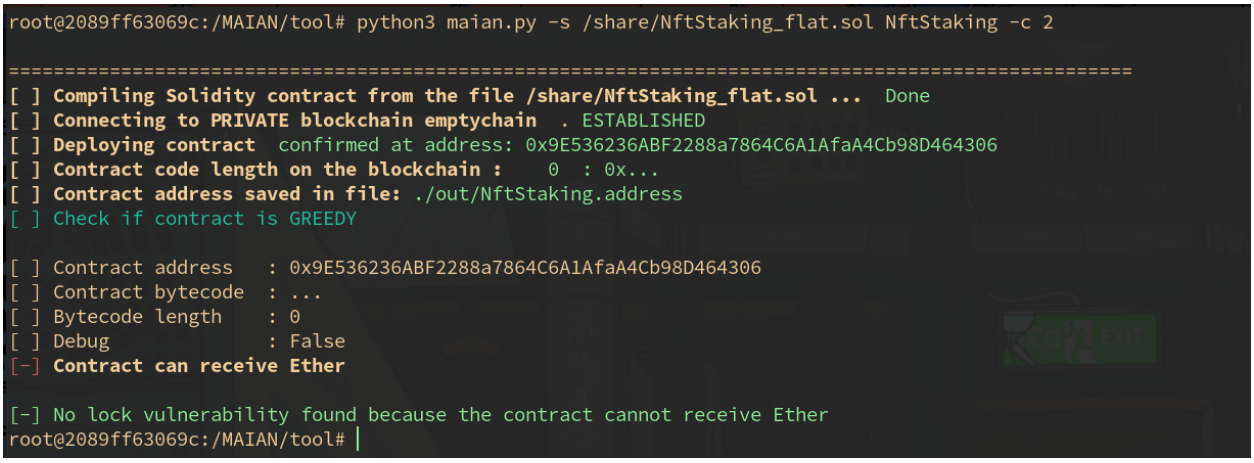
Line	SWC Title	Severity	Short Description
8	(SWC-103) Floating Pragma	Low	A floating pragma is set.
94	(SWC-103) Floating Pragma	Low	A floating pragma is set.
320	(SWC-103) Floating Pragma	Low	A floating pragma is set.
421	(SWC-103) Floating Pragma	Low	A floating pragma is set.
452	(SWC-103) Floating Pragma	Low	A floating pragma is set.
481	(SWC-103) Floating Pragma	Low	A floating pragma is set.
626	(SWC-103) Floating Pragma	Low	A floating pragma is set.
987	(SWC-103) Floating Pragma	Low	A floating pragma is set.
1079	(SWC-103) Floating Pragma	Low	A floating pragma is set.
1107	(SWC-103) Floating Pragma	Low	A floating pragma is set.
1178	(SWC-103) Floating Pragma	Low	A floating pragma is set.
1209	(SWC-103) Floating Pragma	Low	A floating pragma is set.
1445	(SWC-103) Floating Pragma	Low	A floating pragma is set.
1519	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Echidna





Maian Solidity Analysis



Maian Bytecode Analysis

```
root@2089ff63069c:/MAIAN/tool# python3 maian.py -b /share/flat.bytecode -c 0

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

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 6080604052348015620001157600080fd5b5060405162003e...
[ ] Bytecode length    : 31818
[ ] Blockchain contract: False
[ ] Debug              : False

[-] The code does not contain SUICIDE instructions, hence it is not vulnerable
root@2089ff63069c:/MAIAN/tool# python3 maian.py -b /share/flat.bytecode -c 1

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

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 6080604052348015620001157600080fd5b5060405162003e...
[ ] Bytecode length    : 31818
[ ] Blockchain contract: False
[ ] Debug              : False

[ ] Search with call depth: 1   :
[ ] Search with call depth: 2   :
[ ] Search with call depth: 3   :
[+] No prodigal vulnerability found
root@2089ff63069c:/MAIAN/tool# python3 maian.py -b /share/flat.bytecode -c 2

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

[ ] Contract address   : 0xaFFECAFEAFfECaFEaFFecAfEAFfecAfEAffEcaFE
[ ] Contract bytecode  : 6080604052348015620001157600080fd5b5060405162003e...
[ ] Bytecode length    : 31818
[ ] Debug              : False
[-] Contract can receive Ether

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

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

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