IMMUNEBYTES

Audits

GoingApe

SMART CONTRACT AUDIT
FINAL REPORT

March 11, 2022

TOC

TABLE OF CONTENTS

Introduction

1. About Going Ape

Going Ape is a composable metaverse that creates a shared space for both users and creators on the border of reality and virtuality, using blockchain technology, Non Fungible Tokens and cryptocurrencies.
The access to this metaverse will be granted via Going Ape NFTs that will also grant their holders commercial rights over their NFT, and a lot of other benefits and bonuses.
Going Ape will change the way you look at money, art, the Internet and the world around you.
Join our community and keep up with the forefront of the Evolution!

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

2. About ImmuneBytes

ImmuneBytes is a security start-up to provide 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, dydx.

The team has been able to secure 125+ 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 Going Ape team has provided the following doc for the purpose of audit:

1. Business Requirements on Smart Contract for Going Ape.pdf

Audit Process & Methodology

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

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

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

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

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

Audit Details

- Project Name: Going Ape
- Contracts Name: GoingApeSigVerifyNft
- Languages: Solidity(Smart contract), Typescript (Unit Testing)
- Audit Scope: https://github.com/GoingApe-Official/nft-smart-contract
- Github commits for the initial audit: 16d80a9bba1ee3621d8e5d19ea109ce068e1c0ab
- Github commits for the final audit: a3858c26b96bba32cc5d370df4a3d9867b9367ca
- Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Contract Library, Slither, SmartCheck, Echinda

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:

Admin/Owner Privileges can be misused either 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.

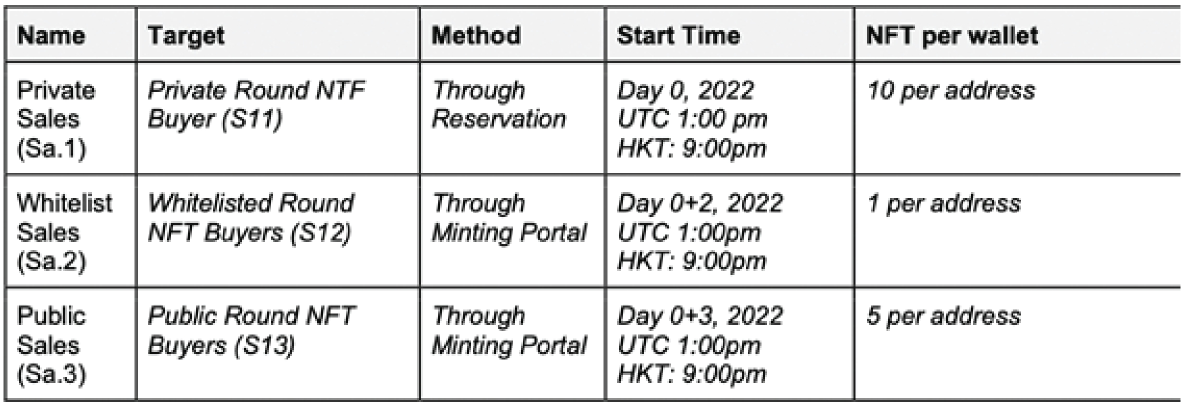
Issues	High	Medium	Low
Open	-	-	1
Closed	1	-	3

Contract Name: GoingApeSigVerifyNft

High Severity Issues

1. **Maximum mint amount for Private and Public Sales do not match the intended design.**
   Line no - 44-46

   Explanation:
   As per the intended architecture of the contract, the Maximum mint limit for a specific wallet in **private** and **public** sales should be 10 and 5 respectively.



Name	Target	Method	Start Time	NFT per wallet
Private Sales (Sa.1)	*Private Round NTF Buyer (S11)*	*Through Reservation*	*Day 0, 2022 UTC 1:00 pm HKT: 9:00pm*	*10 per address*
Whitelist Sales (Sa.2)	*Whitelisted Round NFT Buyers (S12)*	*Through Minting Portal*	*Day 0+2, 2022 UTC 1:00pm HKT: 9:00pm*	*1 per address*
Public Sales (Sa.3)	*Public Round NFT Buyers (S13)*	*Through Minting Portal*	*Day 0+3, 2022 UTC 1:00pm HKT: 9:00pm*	*5 per address*

   However, during the manual code review of the contract, it was found that the maximum mint limit for the **private** and **public** sales should be 5 and 3 respectively. This leads to an unwanted scenario where the users can mint less than they could, as per the intended design.

   Recommendation:
   If the above-mentioned issue is not intended, it is recommended to update the state variables accordingly. Otherwise, the document should be updated to match the updated contract design.

   Amended (March 11th, 2022): The issue was fixed by the Going Ape team and is no longer present in commit a3858c26b96bba32cc5d370df4a3d9867b9367ca.

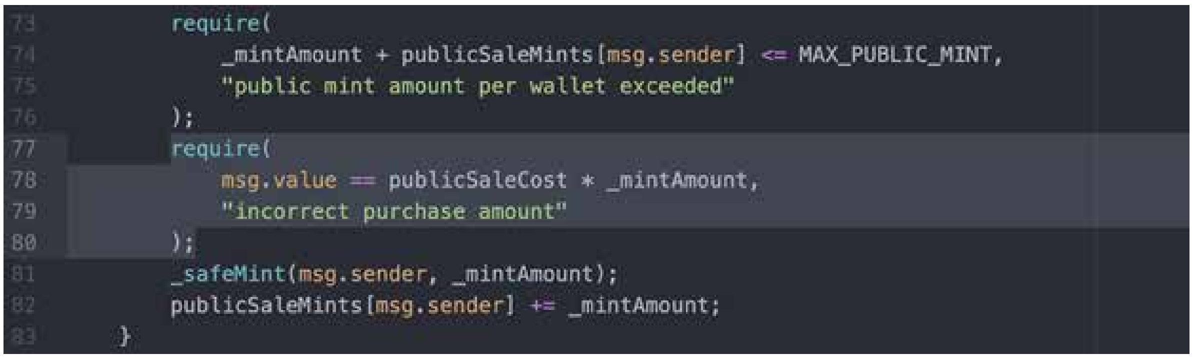
Medium severity issues

No issues were found.

Low severity issues

1. **Strict Equality should be avoided in Require Statements**
   Line no - 78, 103, 136

   Explanation:
   The **GoingApeSigVerifyNft** contract include a few functions where the require statements contains a strict equality check that ensures that the msg.value being passed during the transaction is exactly equal to purchase amount.



```
73            require(
74                _mintAmount + publicSaleMints[msg.sender] <= MAX_PUBLIC_MINT,
75                "public mint amount per wallet exceeded"
76            );
77            require(
78                msg.value == publicSaleCost * _mintAmount,
79                "incorrect purchase amount"
80            );
81            _safeMint(msg.sender, _mintAmount);
82            publicSaleMints[msg.sender] += _mintAmount;
83        }
```

   It is considered a better practice in Solidity to avoid strict equality checks on ether or token balances as any slight difference between the balances might revert the entire function.

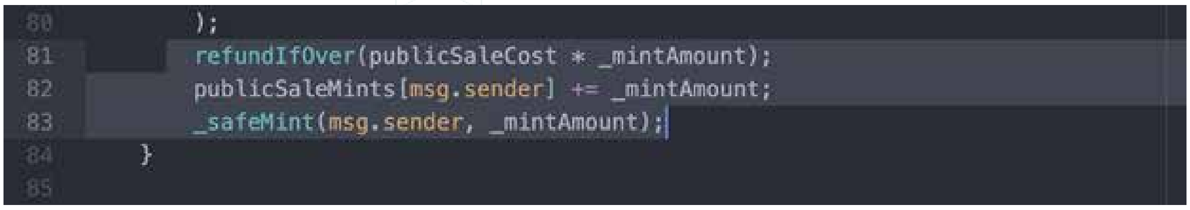
   Recommendation:
   It is recommended to avoid the use of strict equality in **require** statements unless the above-mentioned function design is intentional.

   Amended (March 11th, 2022): The issue was fixed by the **Going Ape** team and is no longer present in commit a3858c26b96bba32cc5d370df4a3d9867b9367ca.

2. **Violation of Check_Effects_Interaction Pattern in the Mint functions**
   Line no - 82, 115, 144, 145

   Note: Issue found during the final code review

   Explanation:
   During the code review, it was found that the **GoingApeSigVerifyNft** contract includes a few functions that update some of the very imperative state variables of the contract after the external calls are being made.



```
        );
        refundIfOver(publicSaleCost * _mintAmount);
        publicSaleMints[msg.sender] += _mintAmount;
        _safeMint(msg.sender, _mintAmount);
    }
```

In the updated code, the **publicmint, whitelistSaleMint, as well as privateSaleMint function,** include a private function, i.e., **refundIfOver()**, that transfers any remaining ETH to the caller.

However, the state variables in all of these functions were updated after this external call. Although the function has been assigned a **nonReentrant** modifier, it is not considered a better practice to violate the Check-Effects-Interaction pattern.

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

3. **No Events emitted after imperative State Variable modification**
Line no -173, 177, 181

Description:
Functions that update an imperative arithmetic state variable contract should emit an event after the state modification.

The following functions modify some crucial arithmetic parameters in the contract but don't emit any event after that:

- setPublicCost()
- setWhitelistSaleCost()
- setPrivateSaleCost()

Since there is no event emitted on updating these variables, it might be difficult to track it off-chain.

Recommendation:
An event should be fired after changing crucial arithmetic state variables.

Amended (March 11th, 2022): The issue was fixed by the **Going Ape** team and is no longer present in commit a3858c26b96bba32cc5d370df4a3d9867b9367ca.

4. **External Visibility should be preferred**

Explanation:
Functions that are never called throughout the contract should be marked as **external** visibility instead of **public** visibility.
This will effectively result in Gas Optimization as well.

Therefore, the following function must be marked as **external** within the contract:

- publicMint()
- whitelistSaleMint()
- privateSaleMint()
- numberMinted()
- getOwnershipData()
- setPublicCost()
- setWhitelistSaleCost()
- setPrivateSaleCost()
- setProvenanceHash()
- setBaseURI()
- setPublicSaleState()
- setWhitelistSaleState()
- setPrivateSaleState()
- setSigner()
- reserve()
- withdrawBalance()

Recommendation:
If the **PUBLIC** visibility of the above-mentioned functions is not intended, then the **EXTERNAL** Visibility keyword should be preferred.

Amended (March 11th, 2022): The issue was fixed by the **Going Ape** team and is no longer present in commit a3858c26b96bba32cc5d370df4a3d9867b9367ca.

Recommendations/Informational

1. **Absence of Error messages in Require Statements**
   Line no - 223

   Description:
   The **withdrawBalance**() function in the contract contains a **require** statement that doesn't involve any error messages.

   While this makes it troublesome to detect the reason behind a particular function revert, it also reduces the readability of the code.

   Recommendation:
   Error Messages must be included in every require statement in the contract.

   Amended (March 11th, 2022): The issue was fixed by the **Going Ape** team and is no longer present in commit a3858c26b96bba32cc5d370df4a3d9867b9367ca.

2. **Unlocked Pragma statements found in the contracts**
   Line no: 3

   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.

   Amended (March 11th, 2022): The issue was fixed by the **Going Ape** team and is no longer present in commit a3858c26b96bba32cc5d370df4a3d9867b9367ca.

3. **Test Cases need should be modified and fixed**

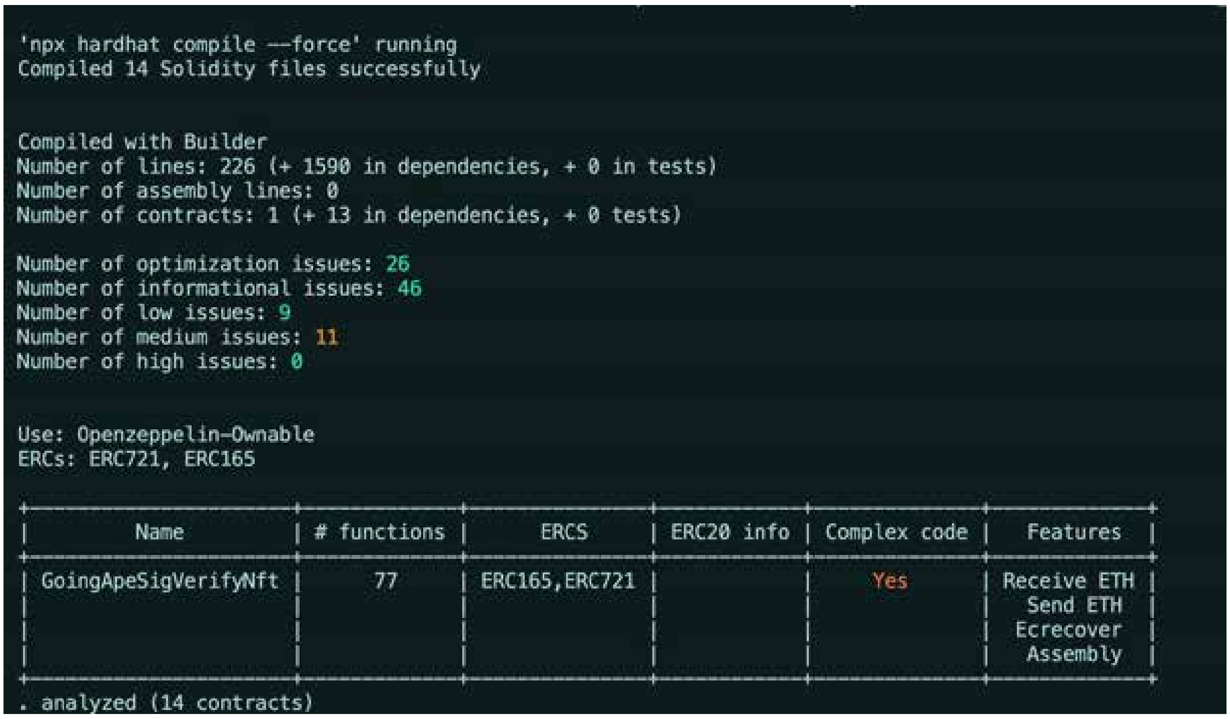
   Note: Issue found during the final code review

   Explanation:
   The test cases designed for the current contract structure fail at particular instances.

   Recommendation:
   In order to ensure that the contract is tested on all possible scenarios, the test cases must be modified and enhanced.

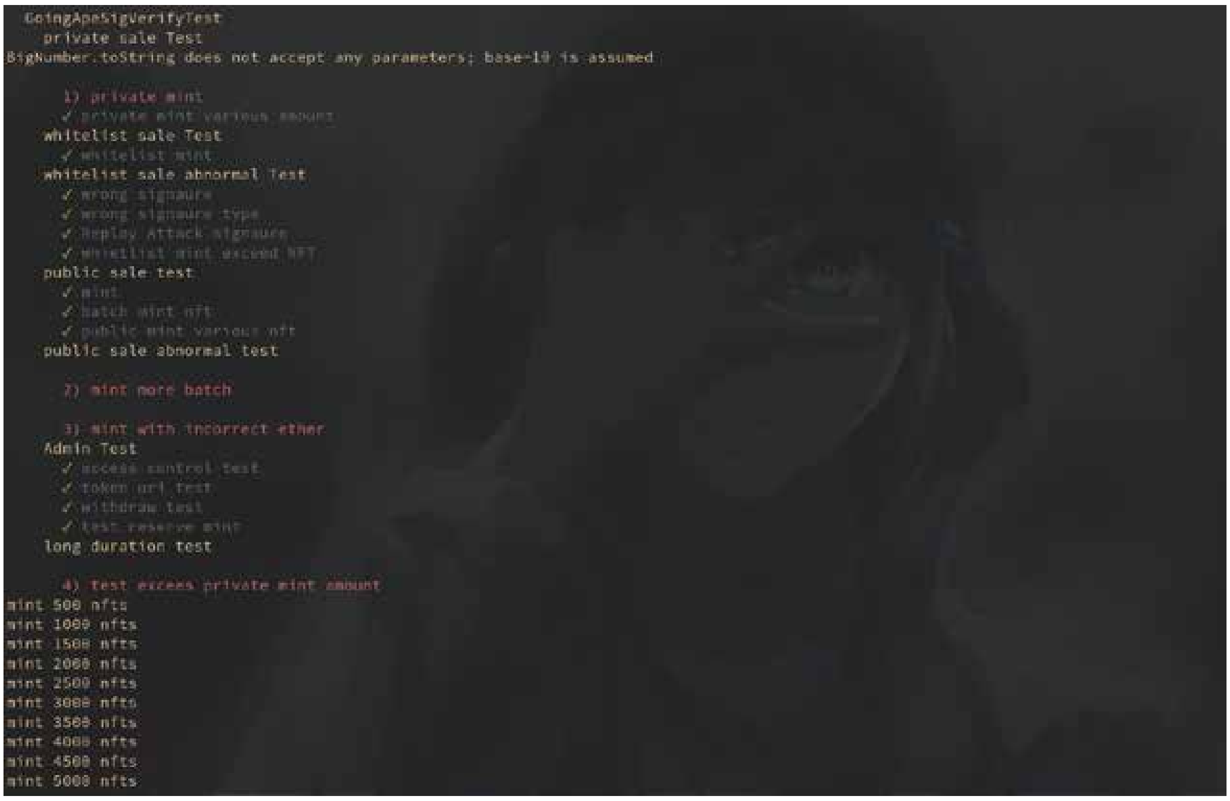
IMMUNEBYTES

Automated Audit Result

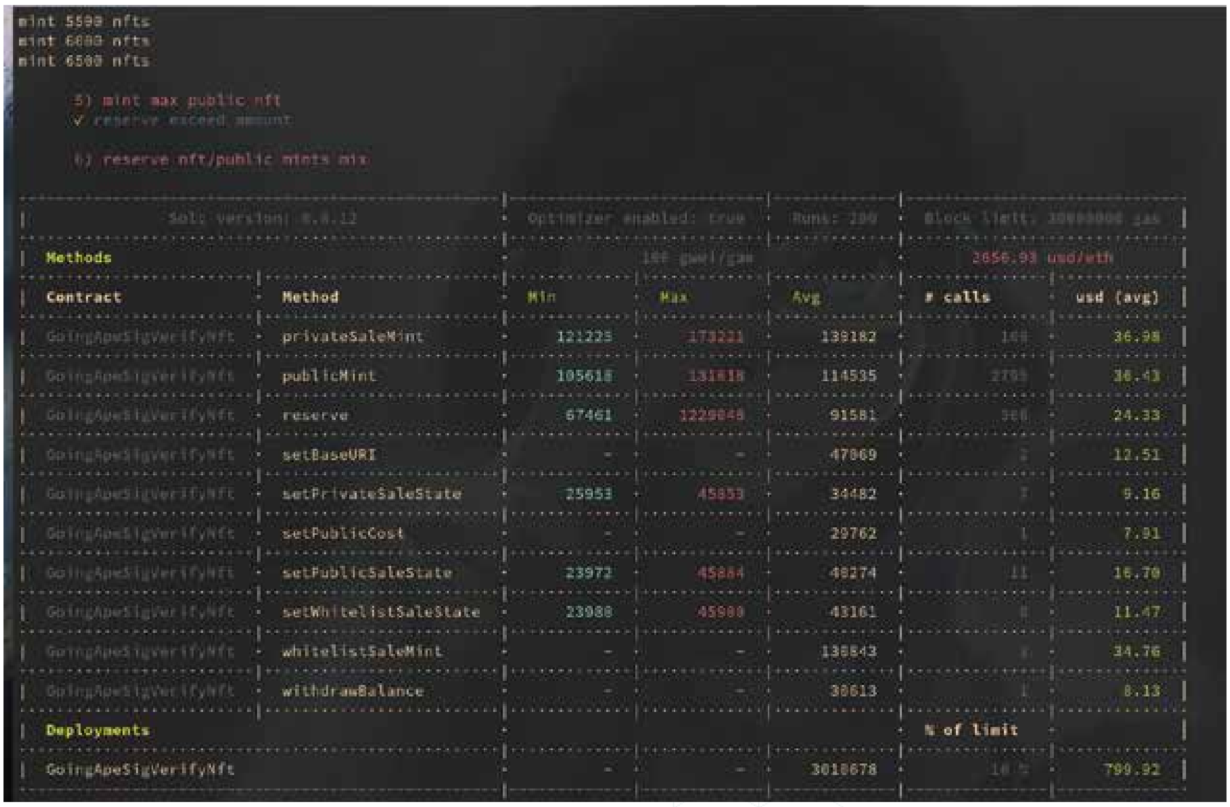


```
'npx hardhat compile --force' running
Compiled 14 Solidity files successfully


Compiled with Builder
Number of lines: 226 (+ 1590 in dependencies, + 0 in tests)
Number of assembly lines: 0
Number of contracts: 1 (+ 13 in dependencies, + 0 tests)

Number of optimization issues: 26
Number of informational issues: 46
Number of low issues: 9
Number of medium issues: 11
Number of high issues: 0


Use: Openzeppelin-Ownable
ERCs: ERC721, ERC165

+-----------------------+-------------+-----------------+------------+--------------+-------------+
|         Name          | # functions |      ERCS       | ERC20 info | Complex code |  Features   |
+-----------------------+-------------+-----------------+------------+--------------+-------------+
| GoingApeSigVerifyNft  |     77      | ERC165,ERC721   |            |     Yes      | Receive ETH |
|                       |             |                 |            |              |  Send ETH   |
|                       |             |                 |            |              | Ecrecover   |
|                       |             |                 |            |              |  Assembly   |
+-----------------------+-------------+-----------------+------------+--------------+-------------+
. analyzed (14 contracts)
```

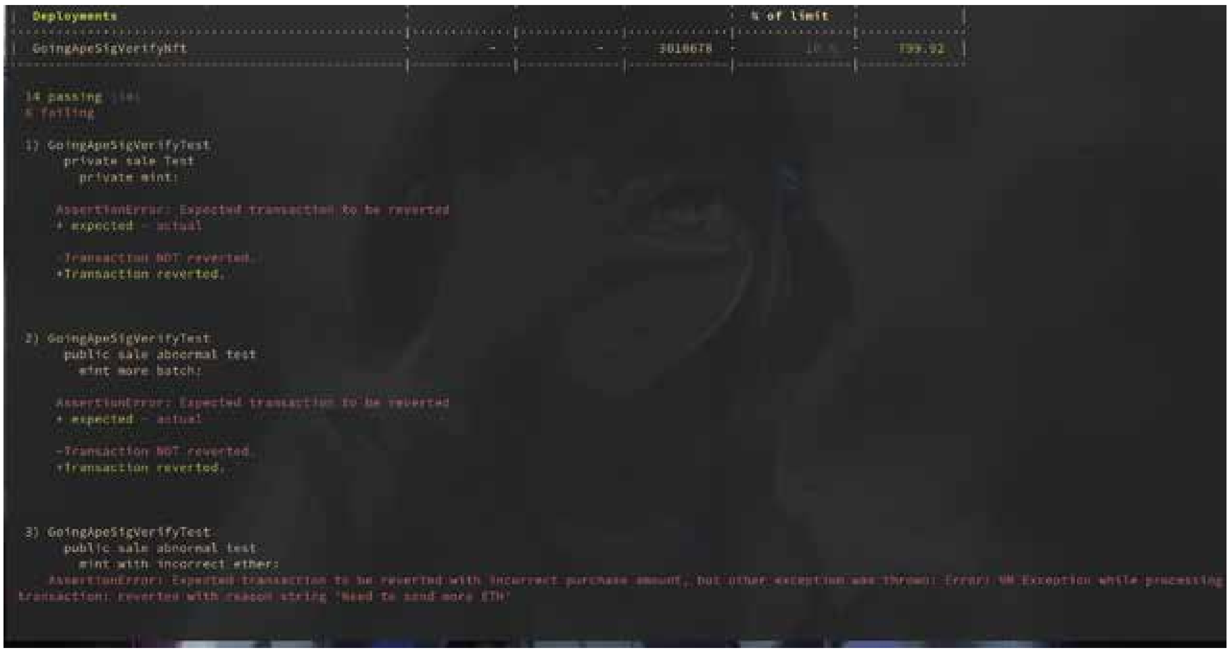
Unit Test



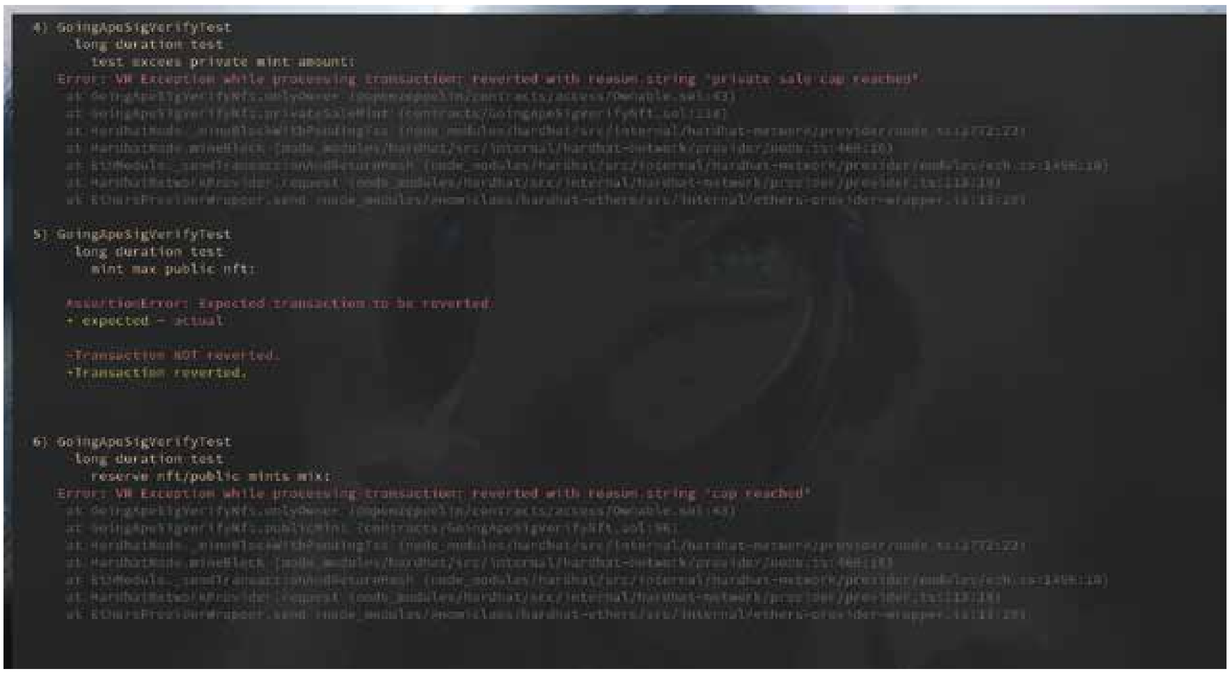
```
  GoingApeSigVerifyTest
    private sale Test
BigNumber.toString does not accept any parameters; base-10 is assumed

      1) private mint
      ✓ private mint various amount
    whitelist sale Test
      ✓ whitelist mint
    whitelist sale abnormal Test
      ✓ wrong signaure
      ✓ wrong signaure type
      ✓ Replay Attack signaure
      ✓ whitelist mint exceed NFT
    public sale test
      ✓ mint
      ✓ batch mint nft
      ✓ public mint various nft
    public sale abnormal test

      2) mint more batch

      3) mint with incorrect ether
    Admin Test
      ✓ access control test
      ✓ token uri test
      ✓ withdraw test
      ✓ test reserve mint
    long duration test

      4) test excees private mint amount
mint 500 nfts
mint 1000 nfts
mint 1500 nfts
mint 2000 nfts
mint 2500 nfts
mint 3000 nfts
mint 3500 nfts
mint 4000 nfts
mint 4500 nfts
mint 5000 nfts
```



```
mint 5500 nfts
mint 6000 nfts
mint 6500 nfts

      5) mint max public nft
      ✓ reserve exceed amount

      6) reserve nft/public mints mix

-------------------------------------------|-------------------------|-------------|-----------------------------
|          Solc version: 0.8.12            ·  Optimizer enabled: true  ·  Runs: 200  ·  Block limit: 30000000 gas  │
···········································|·························|·············|·····························
|  Methods                                 ·            100 gwei/gas           ·       2656.93 usd/eth        │
························|··················|············|············|·············|··············|··············
|  Contract             ·  Method          ·  Min       ·  Max       ·  Avg        ·  # calls     ·  usd (avg)   │
························|··················|············|············|·············|··············|··············
|  GoingApeSigVerifyNft ·  privateSaleMint ·    121225  ·    173221  ·     139182  ·         168  ·       36.98  │
························|··················|············|············|·············|··············|··············
|  GoingApeSigVerifyNft ·  publicMint      ·    105618  ·    131818  ·     114535  ·        2795  ·       30.43  │
························|··················|············|············|·············|··············|··············
|  GoingApeSigVerifyNft ·  reserve         ·     67461  ·   1229848  ·      91581  ·         366  ·       24.33  │
························|··················|············|············|·············|··············|··············
|  GoingApeSigVerifyNft ·  setBaseURI      ·         -  ·         -  ·      47069  ·           2  ·       12.51  │
························|··················|············|············|·············|··············|··············
|  GoingApeSigVerifyNft ·  setPrivateSaleState  ·  25953  ·  45853  ·     34482  ·           7  ·        9.16  │
························|··················|············|············|·············|··············|··············
|  GoingApeSigVerifyNft ·  setPublicCost   ·         -  ·         -  ·      29762  ·           1  ·        7.91  │
························|··················|············|············|·············|··············|··············
|  GoingApeSigVerifyNft ·  setPublicSaleState  ·   23972  ·   45884  ·     40274  ·          11  ·       10.70  │
························|··················|············|············|·············|··············|··············
|  GoingApeSigVerifyNft ·  setWhitelistSaleState  ·  23988  ·  45980  ·  43161  ·           8  ·       11.47  │
························|··················|············|············|·············|··············|··············
|  GoingApeSigVerifyNft ·  whitelistSaleMint  ·       -  ·         -  ·     130843  ·           2  ·       34.76  │
························|··················|············|············|·············|··············|··············
|  GoingApeSigVerifyNft ·  withdrawBalance ·         -  ·         -  ·      30613  ·           1  ·        8.13  │
························|··················|············|············|·············|··············|··············
|  Deployments                              ·                                      ·  % of limit  ·              │
···········································|············|············|·············|··············|··············
|  GoingApeSigVerifyNft                     ·         -  ·         -  ·    3010678  ·      10 %    ·      799.92  │
-------------------------------------------|------------|------------|-------------|--------------|--------------
```

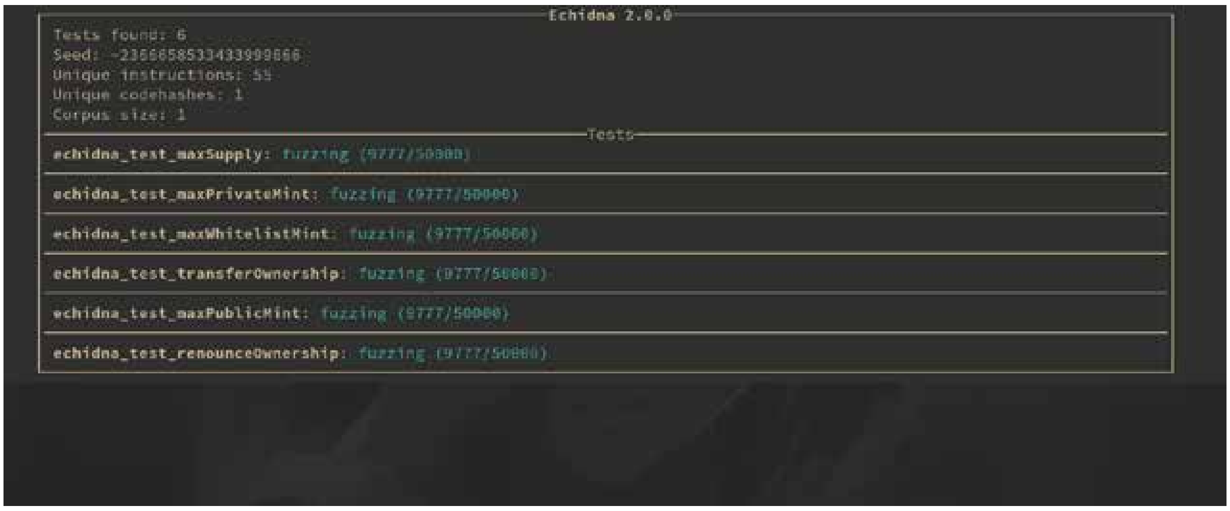


```
|  Deployments                                 ·              ·              ·              ·  % of limit    ·              |
|··············································|··············|··············|··············|················|··············|
|  GoingApeSigVerifyNft                        ·            - ·            - ·      3018678 ·         10 %   ·       799.92 |
|----------------------------------------------|--------------|--------------|--------------|----------------|--------------|

  14 passing (14s)
  6 failing

  1) GoingApeSigVerifyTest
       private sale Test
         private mint:

     AssertionError: Expected transaction to be reverted
     + expected - actual

     -Transaction NOT reverted.
     +Transaction reverted.


  2) GoingApeSigVerifyTest
       public sale abnormal test
         mint more batch:

     AssertionError: Expected transaction to be reverted
     + expected - actual

     -Transaction NOT reverted.
     +Transaction reverted.


  3) GoingApeSigVerifyTest
       public sale abnormal test
         mint with incorrect ether:
    AssertionError: Expected transaction to be reverted with incorrect purchase amount, but other exception was thrown: Error: VM Exception while processing
transaction: reverted with reason string 'Need to send more ETH'
```

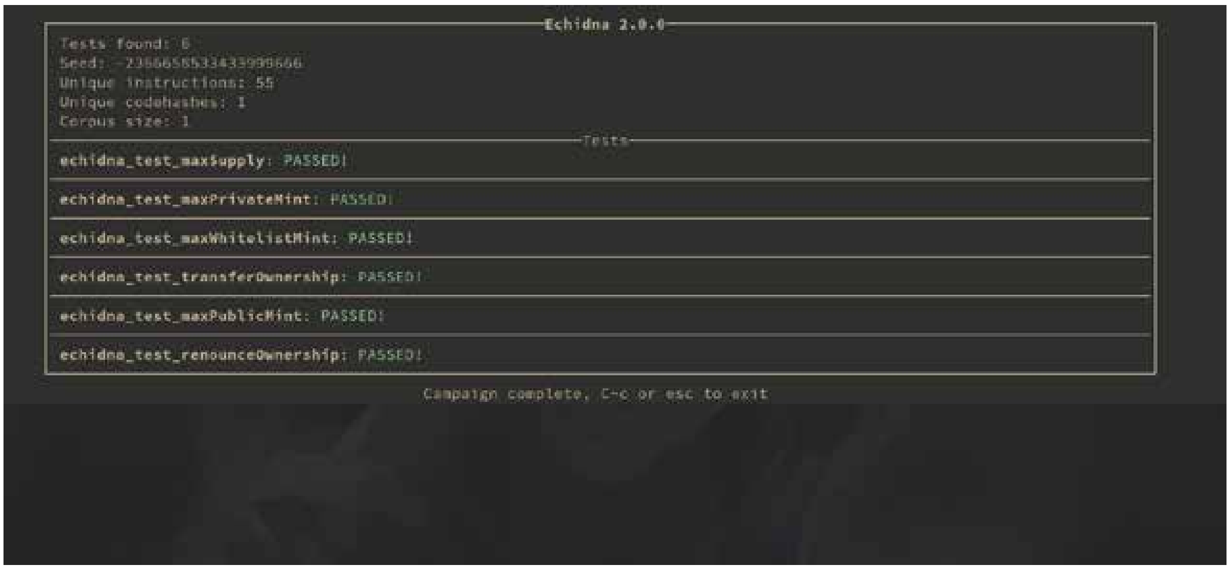


```
  4) GoingApeSigVerifyTest
       long duration test
         test excees private mint amount:
     Error: VM Exception while processing transaction: reverted with reason string 'private sale cap reached'
      at GoingApeSigVerifyNft.onlyOwner (@openzeppelin/contracts/access/Ownable.sol:43)
      at GoingApeSigVerifyNft.privateSaleMint (contracts/GoingApeSigVerifyNft.sol:114)
      at HardhatNode._mineBlockWithPendingTxs (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:1772:23)
      at HardhatNode.mineBlock (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:466:16)
      at EthModule._sendTransactionAndReturnHash (node_modules/hardhat/src/internal/hardhat-network/provider/modules/eth.ts:1496:18)
      at HardhatNetworkProvider.request (node_modules/hardhat/src/internal/hardhat-network/provider/provider.ts:118:18)
      at EthersProviderWrapper.send (node_modules/@nomiclabs/hardhat-ethers/src/internal/ethers-provider-wrapper.ts:13:20)

  5) GoingApeSigVerifyTest
       long duration test
         mint max public nft:

      AssertionError: Expected transaction to be reverted
      + expected - actual

      -Transaction NOT reverted.
      +Transaction reverted.


  6) GoingApeSigVerifyTest
       long duration test
         reserve nft/public mints mix:
     Error: VM Exception while processing transaction: reverted with reason string 'cap reached'
      at GoingApeSigVerifyNft.onlyOwner (@openzeppelin/contracts/access/Ownable.sol:43)
      at GoingApeSigVerifyNft.publicMint (contracts/GoingApeSigVerifyNft.sol:96)
      at HardhatNode._mineBlockWithPendingTxs (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:1772:23)
      at HardhatNode.mineBlock (node_modules/hardhat/src/internal/hardhat-network/provider/node.ts:466:16)
      at EthModule._sendTransactionAndReturnHash (node_modules/hardhat/src/internal/hardhat-network/provider/modules/eth.ts:1496:18)
      at HardhatNetworkProvider.request (node_modules/hardhat/src/internal/hardhat-network/provider/provider.ts:118:18)
      at EthersProviderWrapper.send (node_modules/@nomiclabs/hardhat-ethers/src/internal/ethers-provider-wrapper.ts:13:20)
```

Fuzz Testing



```
Echidna 2.0.0
Tests found: 6
Seed: -2366658533433999666
Unique instructions: 55
Unique codehashes: 1
Corpus size: 1
Tests
echidna_test_maxSupply: fuzzing (9777/50000)
echidna_test_maxPrivateMint: fuzzing (9777/50000)
echidna_test_maxWhitelistMint: fuzzing (9777/50000)
echidna_test_transferOwnership: fuzzing (9777/50000)
echidna_test_maxPublicMint: fuzzing (9777/50000)
echidna_test_renounceOwnership: fuzzing (9777/50000)
```



```
Echidna 2.0.0
Tests found: 6
Seed: -2366658533433999666
Unique instructions: 55
Unique codehashes: 1
Corpus size: 1
Tests
echidna_test_maxSupply: PASSED!
echidna_test_maxPrivateMint: PASSED!
echidna_test_maxWhitelistMint: PASSED!
echidna_test_transferOwnership: PASSED!
echidna_test_maxPublicMint: PASSED!
echidna_test_renounceOwnership: PASSED!
Campaign complete, C-c or esc to exit
```

Concluding Remarks

While conducting the audits of the Going Ape smart contract, it was observed that the contracts contain High and Low severity issues along and a few recommendations to optimize the code.

Our auditors suggest that the developers should resolve High and Low severity issues. The recommendations given will improve the operations of the smart contract.

Disclaimer

ImmuneBytes's audit does not provide the audited smart contract's security or correctness guarantee. 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 Going Ape platform or its product, nor this audit is an investment advice.

Note:

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

IMMUNEBYTES

Audits