Wolves of Wall Street

Smart Contract Audit Report

IMMUNE BYTES

Audits

WOLVES OF WALL STREET

April 1, 2021

Introduction

1. About Wolves of Wall Street

Wolves of Wall Street is focused on innovating in the DeFi/NFT sphere, keeping in mind two of the three pillars of DeFi fundamentals: lending, interest and market-making fees. With a mission to bring gamification, sit on top of these fundamentals through the use of SFTs as mini DAOs to trade in proven profit-seeking strategies and earn higher returns.

They are a team of five professionals, who are non-anon specialists in their field, intent on delivering a secure DeFi project in a way that merges visualized cryptocurrency and gamification with proven DeFi investment protocols, built entirely on open source code.

Visit to know more: https://wows.finance/

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 25+ blockchain projects by providing security services on different frameworks. ImmuneBytes team helps start-up 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

WOWS team has provided documentation for the purpose of conducting the audit. The documents are:

1. https://docs.google.com/document/d/1UTQWP1ensOlIVgSdAIRlCCRhknwfG3KlYkMRkUFYlug/edit?ts=605a1e74

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: Wolves of Wall Street
- Languages: Solidity(Smart contract), Javascript(Unit Testing)
- Github commit hash for audit: 2ba10596b53e309dc671fff94f3f9ff799a2a056
- Contract deployment on Kovan
    - WOWSCryptofolio: 0xbd3377752b9dcb152ea2ca8d6c61080c49fcc66d
    - WOWSERC1155: 0xabea0edbd2ace0264b2dcd05c17a60970f415230

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 References

Every issue in this report was 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	-	-	4
Closed	-	-	-

Low severity issues

1. **More events should be emitted when the contract's state changes.**
   In the **WOWSERC1155** smart contract no events are emitted. The contract has some functions which update the contract's state, those state changes should be notified and logged via events. The functions like **setURI**(), **setCustomDefaultURI**(), **setCustomCardLevel**(), **setWowsLevelCaps**() and **_relinkOwner**() update the internal state of the contract but emit no event. Same is the case with **setOwner**() and **burn**() functions of **WOWSCryptofolio** smart contract. It is always recommended that all state changing functions must emit necessary events.

   Recommendation:
   Consider implementing and emitting more events in above mentioned functions.

2. **No mechanism to update _*cryptofolio* address.**
   In the **_beforeTokenTransfer**() function of **WOWSERC1155** smart contract, a new clone contract of **_cryptofolio** address is created whenever **tokenAddress** is a zero address.

```
      // Create a new WOWSCryptofolio by cloning masterTokenReciver
      // The clone itself is a minimal delegate proxy.
      if (tokenAddress == address(0)) {
        tokenAddress = Clones.clone(_cryptofolio);
        _tokenIdToAddress[tokenId] = tokenAddress;
        IWOWSCryptofolio(tokenAddress).initialize();
      }
```

   The address variable **_cryptofolio** is set at the time of **WOWSERC1155** contract deployment. There is no mechanism present in smart contract to update the master **_cryptofolio** address. This update mechanism might be needed in case any unforeseen bug gets identified or any minor fix is needed in the master **_cryptofolio** contract.

   Recommendation:
   Consider adding some mechanism to update the master **_cryptofolio** address.

3. **WOWSERC1155 contract size exceeds 24 KB limit.**
   Ethereum has a smart contract size limit of 24.576 KB beyond which the smart contract cannot be deployed on the Ethereum blockchain. The **WOWSERC1155** contract's size exceeds the 24 KB limit and throws a compilation error when the solc optimization is disabled. Enabling the optimization decreases the smart contract code to some extent. However, the contract still remains bulky in terms of bytecode size. It is always recommended to break up the large contract into smaller sub contracts so that the size can be decreased.

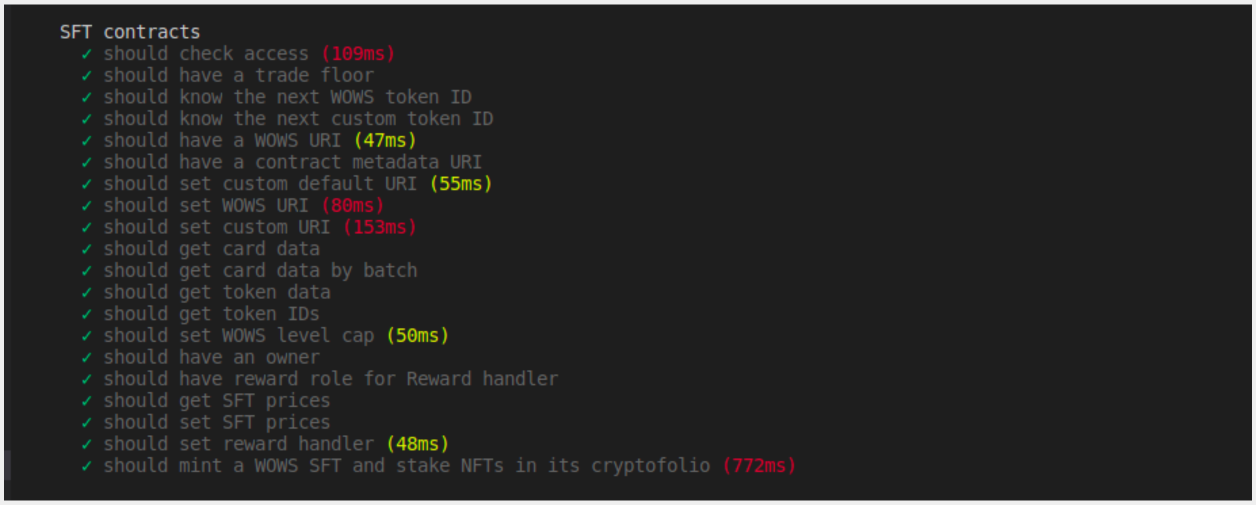
   Recommendation:
   Consider reducing the contract size if possible by removing unnecessary state variables and functions.

4. **Misspelled developer comment.**
   In the **_beforeTokenTransfer**() function (**Line 314**) of **WOWSERC1155** smart contract a developer comment contains a misspelled word. The word **masterTokenReceiver** is spelled as **masterTokenReciver.**

   Recommendation:
   Consider correcting the misspelled word.

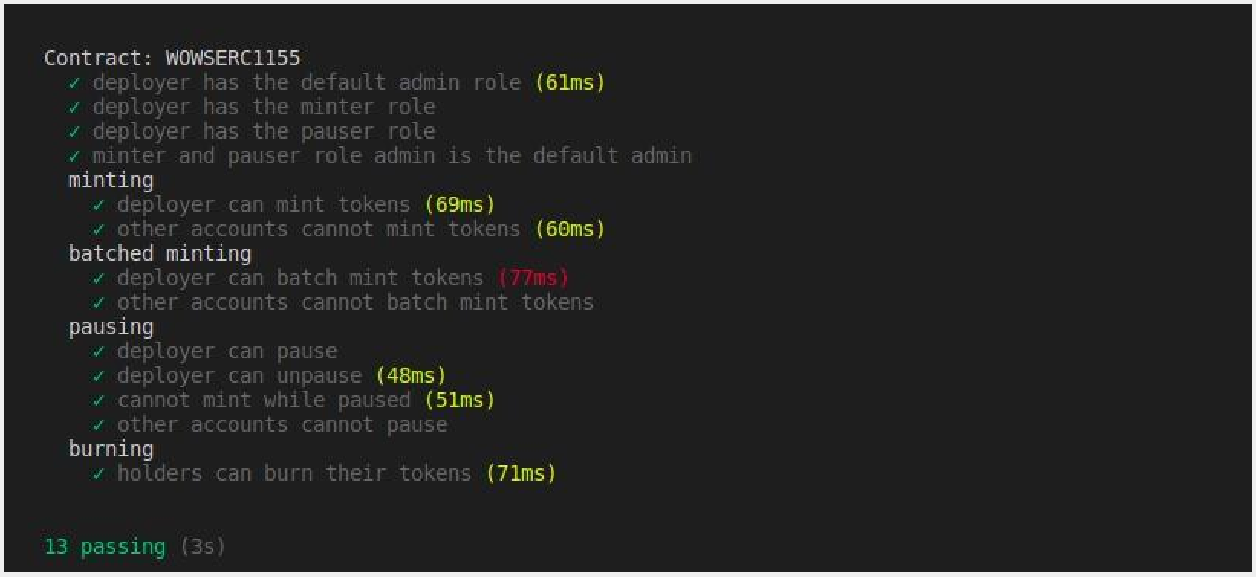
Unit Test

All unit tests provided by the WOWS team are passing successfully.



```
  SFT contracts
    ✓ should check access (109ms)
    ✓ should have a trade floor
    ✓ should know the next WOWS token ID
    ✓ should know the next custom token ID
    ✓ should have a WOWS URI (47ms)
    ✓ should have a contract metadata URI
    ✓ should set custom default URI (55ms)
    ✓ should set WOWS URI (80ms)
    ✓ should set custom URI (153ms)
    ✓ should get card data
    ✓ should get card data by batch
    ✓ should get token data
    ✓ should get token IDs
    ✓ should set WOWS level cap (50ms)
    ✓ should have an owner
    ✓ should have reward role for Reward handler
    ✓ should get SFT prices
    ✓ should set SFT prices
    ✓ should set reward handler (48ms)
    ✓ should mint a WOWS SFT and stake NFTs in its cryptofolio (772ms)
```

Test Cases by the auditors:



```
Contract: WOWSERC1155
  ✓ deployer has the default admin role (61ms)
  ✓ deployer has the minter role
  ✓ deployer has the pauser role
  ✓ minter and pauser role admin is the default admin
  minting
    ✓ deployer can mint tokens (69ms)
    ✓ other accounts cannot mint tokens (60ms)
  batched minting
    ✓ deployer can batch mint tokens (77ms)
    ✓ other accounts cannot batch mint tokens
  pausing
    ✓ deployer can pause
    ✓ deployer can unpause (48ms)
    ✓ cannot mint while paused (51ms)
    ✓ other accounts cannot pause
  burning
    ✓ holders can burn their tokens (71ms)


13 passing (3s)
```

Automated Auditing

Solhint Linting

Solhint is an open-source project for linting solidity code, providing both security and style guide validations. It integrates seamlessly into most mainstream IDEs. We used Solhint as a plugin within our VScode for this analysis. No Linting violations were detected by Solhint, Solhint's npm package is also used to lint the contracts.

Contract Library

Contract-library contains the most complete, high-level decompiled representation of all Ethereum smart contracts, with security analysis applied to them in real-time. We performed analysis using contract Library on the Kovan address of the SporeToken, SporeStake and LiquidityFarming contracts used during manual testing:

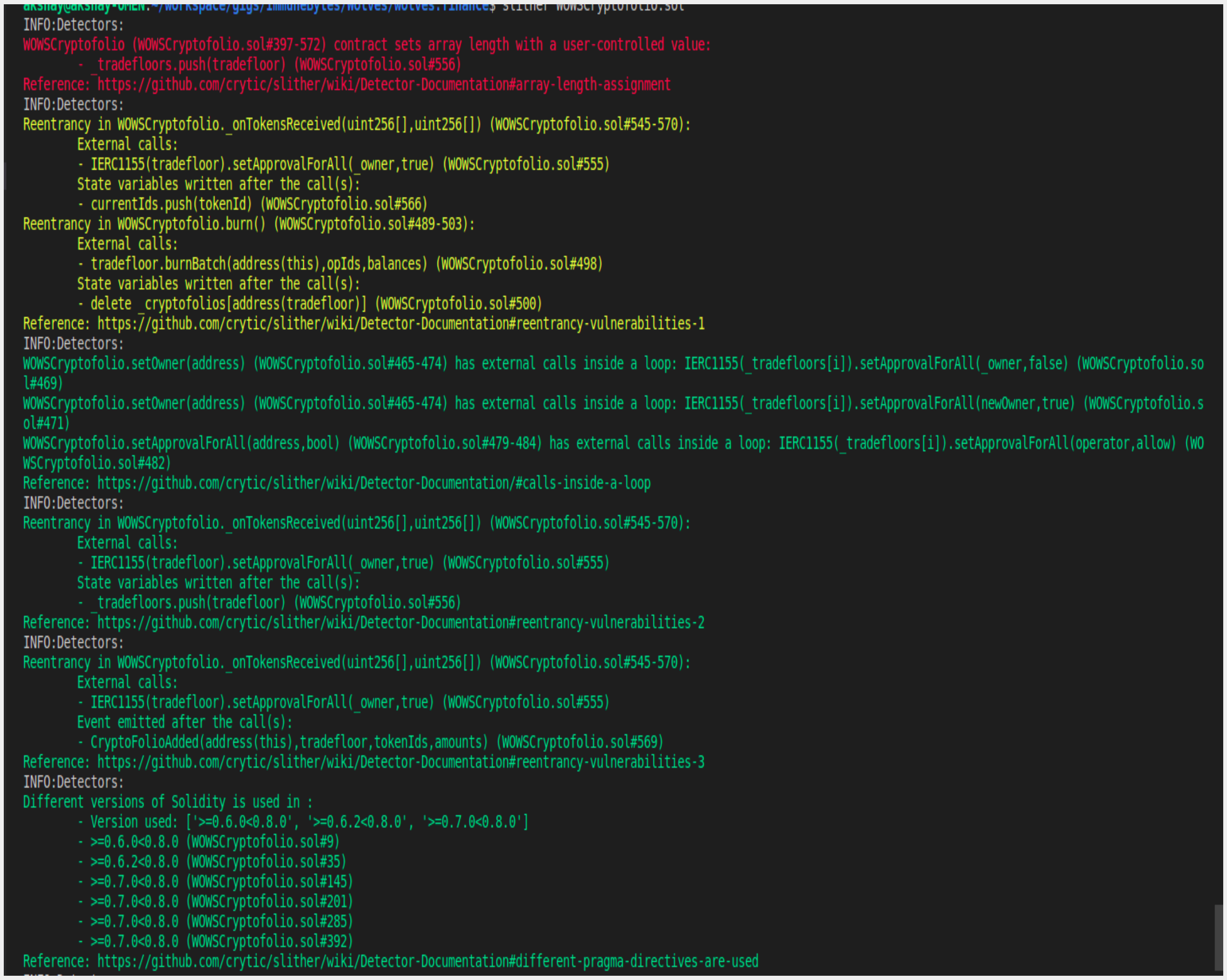
- WOWSCryptofolio: 0xbd3377752b9dcb152ea2ca8d6c61080c49fcc66d
- WOWSERC1155: 0xabea0edbd2ace0264b2dcd05c17a60970f415230

It raises no major concern for the contracts.

Slither

Slither, an open-source static analysis framework. This tool provides rich information about Ethereum smart contracts and has critical properties. While Slither is built as a security-oriented static analysis framework, it is also used to enhance the user's understanding of smart contracts, assist in code reviews, and detect missing optimizations.

The concerns slither raises have already been covered in the manual audit section.



```
akshay@akshay-OMEN:~/workspace/gigs/immunebytes/wolves/wolves.finance$ slither WOWSCryptofolio.sol
INFO:Detectors:
WOWSCryptofolio (WOWSCryptofolio.sol#397-572) contract sets array length with a user-controlled value:
        - _tradefloors.push(tradefloor) (WOWSCryptofolio.sol#556)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#array-length-assignment
INFO:Detectors:
Reentrancy in WOWSCryptofolio._onTokensReceived(uint256[],uint256[]) (WOWSCryptofolio.sol#545-570):
        External calls:
        - IERC1155(tradefloor).setApprovalForAll(_owner,true) (WOWSCryptofolio.sol#555)
        State variables written after the call(s):
        - currentIds.push(tokenId) (WOWSCryptofolio.sol#566)
Reentrancy in WOWSCryptofolio.burn() (WOWSCryptofolio.sol#489-503):
        External calls:
        - tradefloor.burnBatch(address(this),opIds,balances) (WOWSCryptofolio.sol#498)
        State variables written after the call(s):
        - delete _cryptofolios[address(tradefloor)] (WOWSCryptofolio.sol#500)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
INFO:Detectors:
WOWSCryptofolio.setOwner(address) (WOWSCryptofolio.sol#465-474) has external calls inside a loop: IERC1155(_tradefloors[i]).setApprovalForAll(_owner,false) (WOWSCryptofolio.so
l#469)
WOWSCryptofolio.setOwner(address) (WOWSCryptofolio.sol#465-474) has external calls inside a loop: IERC1155(_tradefloors[i]).setApprovalForAll(newOwner,true) (WOWSCryptofolio.s
ol#471)
WOWSCryptofolio.setApprovalForAll(address,bool) (WOWSCryptofolio.sol#479-484) has external calls inside a loop: IERC1155(_tradefloors[i]).setApprovalForAll(operator,allow) (WO
WSCryptofolio.sol#482)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation/#calls-inside-a-loop
INFO:Detectors:
Reentrancy in WOWSCryptofolio._onTokensReceived(uint256[],uint256[]) (WOWSCryptofolio.sol#545-570):
        External calls:
        - IERC1155(tradefloor).setApprovalForAll(_owner,true) (WOWSCryptofolio.sol#555)
        State variables written after the call(s):
        - _tradefloors.push(tradefloor) (WOWSCryptofolio.sol#556)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
Reentrancy in WOWSCryptofolio._onTokensReceived(uint256[],uint256[]) (WOWSCryptofolio.sol#545-570):
        External calls:
        - IERC1155(tradefloor).setApprovalForAll(_owner,true) (WOWSCryptofolio.sol#555)
        Event emitted after the call(s):
        - CryptoFolioAdded(address(this),tradefloor,tokenIds,amounts) (WOWSCryptofolio.sol#569)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
INFO:Detectors:
Different versions of Solidity is used in :
        - Version used: ['>=0.6.0<0.8.0', '>=0.6.2<0.8.0', '>=0.7.0<0.8.0']
        - >=0.6.0<0.8.0 (WOWSCryptofolio.sol#9)
        - >=0.6.2<0.8.0 (WOWSCryptofolio.sol#35)
        - >=0.7.0<0.8.0 (WOWSCryptofolio.sol#145)
        - >=0.7.0<0.8.0 (WOWSCryptofolio.sol#201)
        - >=0.7.0<0.8.0 (WOWSCryptofolio.sol#285)
        - >=0.7.0<0.8.0 (WOWSCryptofolio.sol#392)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
```

Concluding Remarks

While conducting the audits of Wolves of Wall Street smart contract, it was observed that the contracts contain only Low severity issues.

Our auditors suggest that Low severity issues should be resolved by the developers. Resolving the areas of recommendations are up to the team's discretion. The recommendations given will improve the operations of the smart contract.

Disclaimer

ImmuneBytes’s audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

Our team does not endorse the Wolves of Wall Street platform or its product neither 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 Pvt Ltd.