Applicature

Smart Contract Audit

Project: WOWS
Date: October 13th, 2021

WOLVES OF WALL STREET

Table of contents

Summary

Applicature team has conducted a smart contract audit for the given codebase. During the auditing process, Applicature auditing team has found just 3 informational issues. No critical issues were spotted. It is worth mentioning that all of them are fixed by the WOWS team.

The contracts are in good condition. Based on the fixes provided by the WOWS team and on the quality and security of the codebase provided, Applicature team can give a score of 99 to the audited smart contracts.

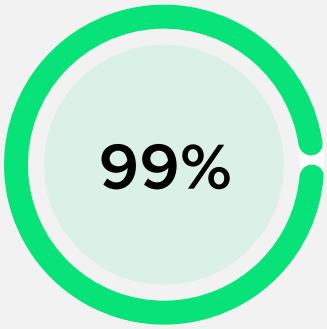
We can state that the contracts bear no security or operational risk to the end-user or contract owner, so they are fully production-ready

During the auditing process, Applicature team has found informational issues only.

Severity of the issue	Total found	Resolved	Unresolved
Critical	0 issue	0 issue	0 issues
High	0 issues	0 issues	0 issues
Medium	0 issue	0 issue	0 issues
Low	0 issue	0 issue	0 issues
Informational	3 issue	3 issue	0 issues
Total	**3 issues**	**3 issues**	**0 issues**

Evaluating the findings, we can assure that the contracts are safe to use. Under the given circumstances we can set the following risk level:

Based on the given findings, risk level, performance, and code style, Applicature team can grant the following overall score:



Applicature auditing team has conducted a bunch of integrated autotests to ensure that the given codebase has decent performance and security levels.

The testable code is 97.14% which is above the industry standard of 95%.

The test results and the coverage can be found in the accompanying section of this audit report.

Please mind that this audit does not certify the definite reliability and security level of the contract. This document describes all vulnerabilities, typos, performance issues, and security issues found by Applicature auditing team. If the code is under development, we recommend running one more audit once the code is finalized.

Scope of Work

WOLVES OF WALL STREET

WOWS is a user-friendly ecosystem of blockchain and AI applications built to power the streaming economy. Our vision is a universal system for turning attention into real-world goods and services.

Within the scope of this audit, two independent auditors deeply investigated the given codebase and analyze the overall security and performance of smart contracts.

The debrief took place on Oct 13th, 2021 and the final results are present in this document.

Applicature auditing team has made a review of the following contracts:

- ICFolioItemHantler
- IWOWSCryptofolio
- IWOWSERC1155
- TokenIds
- WOWSErc1155
- WOWSMinterPauser

The source code was taken from the following **source**:
https://github.com/wolvesofwallstreet/wolves.finance/commit/81f841667cbae891138359d9a91684f17cf17867

Last commit:
https://github.com/wolvesofwallstreet/wolves.finance/tree/48d6c3ce13acbcb099d9c69edeee10a3b57c1976

Initial commit submitted for the audit**:**
https://github.com/wolvesofwallstreet/wolves.finance/commit/81f841667cbae891138359d9a91684f17cf17867

Workflow of the auditing process

During the manual phase of the audit, Applicature team manually looks through the code in order to find any security issues, typos, or discrepancies with the logic of the contract.

Within the testing part, Applicature auditors run integration tests using the Truffle testing framework. The test coverage and the tests themselves are inserted into this audit report.

Applicature team uses the most sophisticated and contemporary methods and techniques to ensure the contract does not have any vulnerabilities or security risks:

- Re-entrancy;
- Access Management Hierarchy;
- Arithmetic Over/Under Flows;
- Unexpected Ether;
- Delegatecall;
- Default Public Visibility;
- Hidden Malicious Code
- Entropy Illusion (Lack of Randomness);
- External Contract Referencing;
- Short Address/Parameter Attack;
- Unchecked CALL Return Values;
- Race Conditions / Front Running;
- General Denial Of Service (DOS);
- Uninitialized Storage Pointers;
- Floating Points and Precision;
- Tx.Origin Authentication;
- Signatures Replay;
- Pool Asset Security (backdoors in the underlying ERC-20).

Structure and organization of the findings

For the convenience of reviewing the findings in this report, Applicature auditors classified them in accordance with the severity of the issues. (from most critical to least critical). The acceptance criteria are described below.

All issues are marked as "Resolved" or "Unresolved", depending on whether they have been fixed by WOWS or not. The latest commit, indicated in this audit report should include all the fixes made.

To ease the explanation, the Applicature team has provided a detailed description of the issues and the recommendation on how to fix them.

Hence, according to the statements above, we classified all the findings in the following way:

Finding	Description
Critical	The issue bear a definite risk to the contract, so it may affect the ability to compile or operate.
High	Major security or operational risk found, that may harm the end-user or the overall performance of the contract.
Medium	The issue affects the contract to operate in a way that doesn't significantly hinder its performance.
Low	The found issue has a slight impact on the performance of the contract or its security.
Informational	The issue does not affect the performance or security of the contract/recommendations on the improvements.

Manual Report

Unnecessary contract inheritance

Informational | Resolved

The WOWSMinterPauser inherits Context and *AccessControl* contracts but *AccessControl* already inherits Context.

Also regarding this conversation, the use of Context "doesn't by itself provide any more security", it should be used in solutions that support meta-transactions or upgradeable contracts. Since there is no such mechanic, Context will only use gas instead of providing more security.

Recommendation:

Remove Context inheritance from WOWSMinterPauser.

Re-Audit:

Removed unnecessary inheritance.

Using string instead of *keccak256* for declaring roles

Informational | Resolved

The WOWSMinterPauser uses *AccessControl* to give access to minting by specific role. To declare a role, use the *bytes32* variable which is initialized by string. Since the string is a dynamic-size variable and *bytes32* is fixed-size it can lead to unpredictable situations.

Recommendation:

Use *keccak256* instead of string to declare roles for AccessControl.

Re-Audit:

Fixed.

Unused function in *WOWSERC1155*

Informational | Resolved

The *WOWSERC1155* contract has an initialization function to set the owner and other variables. It has a check for the count of *DEFAULT_ADMIN_ROLE* to prevent it from being used more than one time. But *DEFAULT_ADMIN_ROLE* already has at least one address from the constructor. It means that it can't be called. Also, this function can be called from any address, which means that anyone can set the owner of the first point will be fixed.

For proper use of initialization functionality, I recommend using this approach.

Recommendation:

One of three:

- Use OpenZeppelin initializable contract.
- Use initialize inside constructor call instead of *_setupRole*
- Remove check for *DEFAULT_ADMIN_ROLE* count and add onlyOwner modifier.

Re-Audit:

Fixed, used initializable.

Test results

To verify the contract security and performance a bunch of integration tests were made using the Truffle testing framework.

Tests were based on the functionality of the code, business logic and requirements and for the purpose of finding the vulnerabilities in the contacts.

It's important to note that Applicature auditors do not modify, edit or add tests to the existing tests provided in the WOWS repo. We write totally separate tests with code coverage of a minimum of 95%, to meet the industry standards.

Tests are written by Applicature

Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered lines
cfolio\interfaces	100.00	100.00	100.00	100.00	
ICFolioItemHandler.sol	100.00	100.00	100.00	100.00	
token/	97.01	92.50	100.00	97.78	
WOWSErc1155.sol	96.43	89.66	100.00	97.35	473, 474, 475
WOWSMinterPauser.sol	100.00	100.00	100.00	100.00	
token/interfaces	100.00	100.00	100.00	100.00	
IWOWSCryptofolio	100.00	100.00	100.00	100.00	
IWOWSERC1155	100.00	100.00	100.00	100.00	

utils/	100.00	100.00	100.00	100.00
TokenIds.sol	100.00	100.00	100.00	100.00
All Files	**97.14**	**92.50**	**100.00**	**97.87**

Test Results

```
Contract: WOWSMinterPauser
  WOWSMinterPauser Phase Test Cases
    ✓ should pause of contract correctly (1321ms)
    ✓ shouldn't pause of contract if msg.sender isn't admin (898ms)
    ✓ should burn correctly (774ms)
    ✓ shouldn't burn if caller isn't owner or isn't approved (681ms)
    ✓ should burn batch correctly (967ms)
    ✓ shouldn't burn batch if caller isn't owner or isn't approved (1108ms)
    ✓ should mint correctly (457ms)
    ✓ shouldn't mint if msg.sender isn't minter (506ms)
    ✓ shouldn't mint if address is zero (383ms)
    ✓ shouldn't mint if transfer operation is paused (819ms)
    ✓ should mint batch correctly (500ms)
    ✓ shouldn't mint batch if msg.sender isn't minter (367ms)
    ✓ shouldn't mint batch if address is zero (407ms)
    ✓ shouldn't mint batch if array's lengths don't match (408ms)
    ✓ shouldn't mint batch if transfer operation is paused (780ms)
    ✓ should get result correctly if interface's support exist (377ms)

Contract: WOWSErc1155
  WOWSErc1155 Phase Test Cases
    ✓ shouldn't initialize contract more than once (1733ms)
    ✓ should get uint256(-1) if id isn’t existed (417ms)
    ✓ shouldn't set base metadata uri if msg.sender isn't admin (1055ms)
    ✓ shouldn't set custom metadata uri if msg.sender isn't admin (1099ms)
    ✓ shouldn't set cfolio metadata uri if msg.sender isn't admin (1447ms)
    ✓ shouldn't set contract's metadata uri if msg.sender isn't admin (1411ms)
    ✓ shouldn't set custom card level if msg.sender isn't minter (1041ms)
    ✓ shouldn't set custom card level if curds isn't custom (484ms)
    ✓ should set cfolio item type correctly (509ms)
    ✓ shouldn't set cfolio item type if msg.sender isn't minter (366ms)
    ✓ shouldn't set cfolio item type if tokenId is invalid (447ms)
    ✓ should set approval for all correctly (444ms)
    ✓ shouldn't set approval for all if caller isn't operator (468ms)
    ✓ should get uri correctly (624ms)
    ✓ should get token data correctly (1024ms)
    ✓ should get cfolio item type correctly (588ms)
    ✓ shouldn't get cfolio item type if id is incorrectly (175ms)
```

```
    ✓ should do before token transfer correctly (1999ms)
    ✓ should do before batch token transfer correctly (4710ms)
    ✓ shouldn't do before batch token transfer if array's
      length is mismatch (2296ms)
    ✓ shouldn't mint batch if amounts isn't 1 (440ms)
    ✓ shouldn't mint batch if already minted (2272ms)
    ✓ should mint with cfolio token id correctly (1213ms)
    ✓ should do before token transfer correctly if to is
      address of zero (2213ms)
    ✓ shouldn't do before token transfer if data is invalid (484ms)
    ✓ shouldn't mint if transfer operation is paused (637ms)
    ✓ shouldn't do before token transfer if address of data is invalid (3936ms)
    ✓ shouldn't do before batch token transfer if count mismatch (6ms)
    ✓ should get mask hash correctly (179ms)

  44 passing 5ms)
```

We are delighted to have a chance to work together with WOWS team and contribute to their success by reviewing and certifying the security of the smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

Website: applicature.com
Email: info@applicature.com

Applicature