Fileers

Smart Contract Audit Report

SYSFIXED

16 Jan 2023

TABLE OF CONTENTS

AUDITED DETAILS

Audited Project

Project name	Token ticker	Blockchain
Fileers	FPFS	BSC

Addresses

Contract address	0xf175AA11B1439E02D3Ce4B14Aad24D046dB3Bc3B
Contract deployer address	0x8c3f02a05e0e36229D7688393A9499E695f04281

Project Website

https://fileers.com/

Codebase

https://bscscan.com/address/0xf175AA11B1439E02D3Ce4B14Aad24D046dB3Bc3B#code

SUMMARY

Scalability is a major challenge facing by Decentralized File-Sharing & Storage Networks. Imagine the potential of a project that can address this issue and create a revolution. Fileers is focused on providing enterprise-level scalability and aims to solve the issue of decentralization in the storage industry.

Contract Summary

Documentation Quality

This project has a standard of documentation.

- Technical description provided.

Code Quality

The quality of the code in this project is up to standard.

- The official Solidity style guide is followed.

Test Scope

Project test coverage is 100% (Via Codebase).

Audit Findings Summary

Issues Found

- SWC-101 | Arithmetic operation issues discovered on lines 23, 26, 29, 32, 35, 44, 54, 196, 197, 373, 375, 543, 568, and 576.
- SWC-101 | Compiler-rewritable issue discovered on line 375.
- SWC-103 | A floating pragma is set on line 1, The current pragma Solidity directive is ""^0.8.4"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.
- SWC-110 | Out of bounds array access discovered on lines 374, 375, 545, 546, 548, and 549.

CONCLUSION

We have audited the Fileers project which has released on January 2023, to discover issues and identify potential security vulnerabilities in Fileers Project. This process is used to find technical issues and security loopholes that find common issues in the code.

The security audit report produced satisfactory results with low-risk issues.

The most common issue found in writing code on contracts that do not pose a big risk is that writing on contracts is close to the standard of writing contracts in general. The low-level issue found is a floating pragma being set and out of bounds array access which the index access expression can cause an exception in case of use of an invalid array index value.

AUDIT RESULT

Article	Category	Description	Result
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	PASS
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	ISSUE FOUND
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	PASS
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	ISSUE FOUND
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	PASS
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	PASS
Check-Effect Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	PASS
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	ISSUE FOUND
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	PASS
Delegate call to Untrusted Caller	SWC-112	Delegatecalls should only be allowed to trusted addresses.	PASS
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	PASS
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	PASS

Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	PASS
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	PASS
Signature Unique Id	SWC-117 SWC-121 SWC-122	Signed messages should always have a unique id. A transaction hash should not be used as a unique id.	PASS
Shadowing State Variable	SWC-119	State variables should not be shadowed.	PASS
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	PASS
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order. The rule of thumb is to inherit contracts from more /general/ to more /specific/.	PASS

SMART CONTRACT ANALYSIS

Started	Sun Jan 15 2023 23:14:51 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jan 16 2023 00:02:24 GMT+0000 (Coordinated Universal Time)
Mode	Standard
Main Source File	Fileers.sol

Detected Issues

ID	Title	Severity	Status
SWC-101	ARITHMETIC OPERATION "+" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "%" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "/" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "*" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "-" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "++" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged
SWC-101	ARITHMETIC OPERATION "**" DISCOVERED	low	acknowledged

SWC-101	COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED	low	acknowledged
SWC-103	A FLOATING PRAGMA IS SET	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged
SWC-110	OUT OF BOUNDS ARRAY ACCESS	low	acknowledged

SWC-101 | ARITHMETIC OPERATION "+" DISCOVERED

LINE 23

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- Fileers.sol

Locations

```
22   function add(uint256 a, uint256 b) internal pure returns (uint256) {
23   return a + b;
24   }
25   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 26

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- Fileers.sol

Locations

```
25   function sub(uint256 a, uint256 b) internal pure returns (uint256) {
26   return a - b;
27   }
28   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 29

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- Fileers.sol

Locations

```
28   function mul(uint256 a, uint256 b) internal pure returns (uint256) {
29   return a * b;
30   }
31   function div(uint256 a, uint256 b) internal pure returns (uint256) {
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 32

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- Fileers.sol

Locations

```
31   function div(uint256 a, uint256 b) internal pure returns (uint256) {
32   return a / b;
33   }
34   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
```

SWC-101 | ARITHMETIC OPERATION "%" DISCOVERED

LINE 35

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- Fileers.sol

Locations

```
34   function mod(uint256 a, uint256 b) internal pure returns (uint256) {
35   return a % b;
36   }
37   function sub(
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 44

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- Fileers.sol

Locations

```
43   require(b <= a, errorMessage);
44   return a - b;
45   }
46   }
```

SWC-101 | ARITHMETIC OPERATION "/" DISCOVERED

LINE 54

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- Fileers.sol

Locations

```
53   require(b > 0, errorMessage);
54   return a / b;
55   }
56   }
```

SWC-101 | ARITHMETIC OPERATION "*" DISCOVERED

LINE 196

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- Fileers.sol

Locations

```
195   _decimals = 18;
196   _tTotal = 2000000000 * 10**_decimals;
197   _rTotal = (MAX - (MAX % _tTotal));
198   _marketingWalletAddress = 0xc440c5C1a866189199F616E3D65B4F33a1e6f10E;
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 197

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- Fileers.sol

Locations

```
196   _tTotal = 2000000000 * 10**_decimals;
197   _rTotal = (MAX - (MAX % _tTotal));
198   _marketingWalletAddress = 0xc440c5C1a866189199F616E3D65B4F33a1e6f10E;
199   |
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 373

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- Fileers.sol

Locations

```
372   require(_isExcluded[account], "Account is already included");
373   for (uint256 i = 0; i < _excluded.length; i++) {
374   if (_excluded[i] == account) {
375   _excluded[i] = _excluded[_excluded.length - 1];
```

SWC-101 | ARITHMETIC OPERATION "-" DISCOVERED

LINE 375

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- Fileers.sol

Locations

```
374   if (_excluded[i] == account) {
375   _excluded[i] = _excluded[_excluded.length - 1];
376   _tOwned[account] = 0;
377   _isExcluded[account] = false;
```

SWC-101 | ARITHMETIC OPERATION "++" DISCOVERED

LINE 543

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- Fileers.sol

Locations

```
542   uint256 tSupply = _tTotal;
543   for (uint256 i = 0; i < _excluded.length; i++) {
544   if (
545   _rOwned[_excluded[i]] > rSupply ||
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 568

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- Fileers.sol

Locations

```
567   function calculateTaxFee(uint256 _amount) private view returns (uint256) {
568   return _amount.mul(_taxFee).div(10**2);
569   }
570
571   |
```

SWC-101 | ARITHMETIC OPERATION "**" DISCOVERED

LINE 576

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- Fileers.sol

Locations

```
575   {
576   return _amount.mul(_marketingFee).div(10**2);
577   }
578   |
```

SWC-101 | COMPILER-REWRITABLE "<UINT> - 1" DISCOVERED

LINE 375

low SEVERITY

This plugin produces issues to support false positive discovery within Mythril.

Source File

- Fileers.sol

Locations

```
374   if (_excluded[i] == account) {
375   _excluded[i] = _excluded[_excluded.length - 1];
376   _tOwned[account] = 0;
377   _isExcluded[account] = false;
```

SWC-103 | A FLOATING PRAGMA IS SET

LINE 7

low SEVERITY

The current pragma Solidity directive is ""^0.8.4"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source File

- Fileers.sol

Locations

```
6   // SPDX-License-Identifier: Unlicensed
7   pragma solidity ^0.8.4;
8   interface IERC20 {
9   |
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 374

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Fileers.sol

Locations

```
373   for (uint256 i = 0; i < _excluded.length; i++) {
374   if (_excluded[i] == account) {
375   _excluded[i] = _excluded[_excluded.length - 1];
376   _tOwned[account] = 0;
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 375

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Fileers.sol

Locations

```
374   if (_excluded[i] == account) {
375   _excluded[i] = _excluded[_excluded.length - 1];
376   _tOwned[account] = 0;
377   _isExcluded[account] = false;
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 545

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Fileers.sol

Locations

```
544   if (
545   _rOwned[_excluded[i]] > rSupply ||
546   _tOwned[_excluded[i]] > tSupply
547   ) return (_rTotal, _tTotal);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 546

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Fileers.sol

Locations

```
545   _rOwned[_excluded[i]] > rSupply ||
546   _tOwned[_excluded[i]] > tSupply
547   ) return (_rTotal, _tTotal);
548   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 548

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Fileers.sol

Locations

```
547   ) return (_rTotal, _tTotal);
548   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
549   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
550   }
```

SWC-110 | OUT OF BOUNDS ARRAY ACCESS

LINE 549

low SEVERITY

The index access expression can cause an exception in case of use of invalid array index value.

Source File

- Fileers.sol

Locations

```
548   rSupply = rSupply.sub(_rOwned[_excluded[i]]);
549   tSupply = tSupply.sub(_tOwned[_excluded[i]]);
550   }
551   if (rSupply < _rTotal.div(_tTotal)) return (_rTotal, _tTotal);
```

DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to, or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Sysfixed's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Sysfixed to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model, or legal compliance.

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Sysfixed and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Sysfixed) owe no duty of care.

ABOUT US

Sysfixed is a blockchain security certification organization established in 2021 with the objective to provide smart contract security services and verify their correctness in blockchain-based protocols. Sysfixed automatically scans for security vulnerabilities in Ethereum and other EVM-based blockchain smart contracts. Sysfixed a comprehensive range of analysis techniques—including static analysis, dynamic analysis, and symbolic execution—can accurately detect security vulnerabilities to provide an in-depth analysis report. With a vibrant ecosystem of world-class integration partners that amplify developer productivity, Sysfixed can be utilized in all phases of your project's lifecycle. Our team of security experts is dedicated to the research and improvement of our tools and techniques used to fortify your code.