



Cyberscope

## Audit Report

# Panda Finance

February 2023

Type	ERC20
Network	ARBITRUM
Address	0x1f9074030214f8b681f20c169f036c03a25ae9da
Audited by	© cyberscope

# Table of Contents

<b>Table of Contents</b>	<b>1</b>
<b>Review</b>	<b>3</b>
<b>Audit Updates</b>	<b>3</b>
<b>Source Files</b>	<b>4</b>
<b>Analysis</b>	<b>5</b>
<b>Diagnostics</b>	<b>6</b>
<b>RSML - Redundant SafeMath Library</b>	<b>7</b>
Description	7
Recommendation	7
<b>IDI - Immutable Declaration Improvement</b>	<b>8</b>
Description	8
Recommendation	8
<b>L02 - State Variables could be Declared Constant</b>	<b>9</b>
Description	9
Recommendation	9
<b>L04 - Conformance to Solidity Naming Conventions</b>	<b>10</b>
Description	10
Recommendation	10
<b>L05 - Unused State Variable</b>	<b>12</b>
Description	12
Recommendation	12
<b>L07 - Missing Events Arithmetic</b>	<b>13</b>
Description	13
Recommendation	13
<b>L09 - Dead Code Elimination</b>	<b>14</b>
Description	14
Recommendation	14
<b>L16 - Validate Variable Setters</b>	<b>16</b>
Description	16
Recommendation	16
<b>L19 - Stable Compiler Version</b>	<b>17</b>
Description	17

<b>Recommendation</b>	<b>17</b>
<b>Functions Analysis</b>	<b>18</b>
<b>Inheritance Graph</b>	<b>20</b>
<b>Flow Graph</b>	<b>21</b>
<b>Summary</b>	<b>22</b>
<b>Disclaimer</b>	<b>23</b>
<b>About Cyberscope</b>	<b>24</b>

## Review

Contract Name	PANDAFI
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	200 runs
Explorer	<a href="https://arbiscan.io/address/0x1f9074030214f8b681f20c169f036c03a25ae9da">https://arbiscan.io/address/0x1f9074030214f8b681f20c169f036c03a25ae9da</a>
Address	0x1f9074030214f8b681f20c169f036c03a25ae9da
Network	ARBITRUM
Symbol	PANDA
Decimals	18
Total Supply	1,000,000,000

## Audit Updates

Initial Audit	25 Feb 2023
---------------	-------------

## Source Files

Filename	SHA256
Address.sol	b34455be5d038960030a948b1cadda19 b375bbf84bf08b7be89d09c197f29e35
Context.sol	1458c260d010a08e4c20a4a517882259a 23a4baa0b5bd9add9fb6d6a1549814a
IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db80 03d075c9c541019eb8dcf4122864d5
IUniswapV2Factory.sol	51d056199e3f5e41cb1a9f11ce581aa3e1 90cc982db5771ffeef8d8d1f962a0d
IUniswapV2Pair.sol	29c75e69ce173ff8b498584700fef76bc81 498c1d98120e2877a1439f0c31b5a
IUniswapV2Router01.sol	0439ffe0fd4a5e1f4e22d71ddbda76d63d 61679947d158cba4ee0a1da60cf663
IUniswapV2Router02.sol	a2900701961cb0b6152fc073856b97256 4f7c798797a4a044e83d2ab8f0e8d38
Ownable.sol	9d78bd0e818b158028955f235748b272d ad0124945dd627bba19dbff3ec4743a
PANDAFI.sol	1465899b2f1f15024a4a6bae7c83288084 cbabccfea7e91a748135d2a43cc0df
SafeMath.sol	0dc33698a1661b22981abad8e5c6f5ebc a0dfe5ec14916369a2935d888ff257a

# Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OCTD	Transfers Contract's Tokens	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

# Diagnostics

● Critical   ● Medium   ● Minor / Informative

Severity	Code	Description	Status
●	RSML	Redundant SafeMath Library	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L05	Unused State Variable	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L19	Stable Compiler Version	Unresolved

## RSML - Redundant SafeMath Library

<b>Criticality</b>	Minor / Informative
<b>Location</b>	PANDA.FI.sol
<b>Status</b>	Unresolved

### Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert on underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases unnecessarily the gas consumption.

```
library SafeMath {...}
```

### Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than 0.8.0 then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the unchecked { ... } statement.

Read more about the breaking change on

<https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes>.



## IDI - Immutable Declaration Improvement

<b>Criticality</b>	Minor / Informative
<b>Location</b>	PANDAFI.sol#L55,56,57,58,65,66,70,78
<b>Status</b>	Unresolved

### Description

The contract is using variables that initialize them only in the constructor. The other functions are not mutating the variables. These variables are not defined as `immutable`.

```
_nam  
_symbo  
_decimal  
_tTota  
_maxHeldAmoun  
_maxTxAmoun  
uniswapV2Pai  
uniswapV2Route
```

### Recommendation

By declaring a variable as immutable, the Solidity compiler is able to make certain optimizations. This can reduce the amount of storage and computation required by the contract, and make it more gas-efficient.

## L02 - State Variables could be Declared Constant

<b>Criticality</b>	Minor / Informative
<b>Location</b>	PANDA.FI.sol#L25,26,28
<b>Status</b>	Unresolved

### Description

State variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
address public router
address public basePair
uint256 public prevDevFee
```

### Recommendation

Constant state variables can be useful when the contract wants to ensure that the value of a state variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

## L04 - Conformance to Solidity Naming Conventions

<b>Criticality</b>	Minor / Informative
<b>Location</b>	PANDAFI.sol#L37,40,43,44,125,221,230,261,361
<b>Status</b>	Unresolved

### Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX\_VALUE, ERROR\_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
address public _devWalletAddress
uint256 public _devFee
uint256 public _maxTxAmount
uint256 public _maxHeldAmount
address _owner
address _addr
address _newAddr
uint256 _amount
uint256 ETHAmount
```

### Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.

Find more information on the Solidity documentation

<https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention>.

## L05 - Unused State Variable

<b>Criticality</b>	Minor / Informative
<b>Location</b>	PANDAFI.sol#L36
<b>Status</b>	Unresolved

### Description

An unused state variable is a state variable that is declared in the contract, but is never used in any of the contract's functions. This can happen if the state variable was originally intended to be used, but was later removed or never used.

Unused state variables can create clutter in the contract and make it more difficult to understand and maintain. They can also increase the size of the contract and the cost of deploying and interacting with it.

```
address[] private _excluded
```

### Recommendation

To avoid creating unused state variables, it's important to carefully consider the state variables that are needed for the contract's functionality, and to remove any that are no longer needed. This can help improve the clarity and efficiency of the contract.

## L07 - Missing Events Arithmetic

<b>Criticality</b>	Minor / Informative
<b>Location</b>	PANDAFI.sol#L217
<b>Status</b>	Unresolved

### Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task.

It's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

```
_devFee = devFee
```

### Recommendation

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues with its arithmetic.

## L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	PANDAFI.sol#L343,361
Status	Unresolved

### Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function swapTokensForEth(uint256 tokenAmount) private {
    // generate the uniswap pair path of token -> WHT
    address[] memory path = new address[](2);
    path[0] = address(this);
    path[1] = getBasePairAddr();

    ...

    tokenAmount,
    0, // accept any amount of ETH
    path,
    address(this),
    block.timestamp
);
}
```

### Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.



## L16 - Validate Variable Setters

<b>Criticality</b>	Minor / Informative
<b>Location</b>	PANDAFI.sol#L63
<b>Status</b>	Unresolved

### Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
_devWalletAddress = devWalletAddress_
```

### Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

## L19 - Stable Compiler Version

<b>Criticality</b>	Minor / Informative
<b>Location</b>	PANDAFI.sol#L4
<b>Status</b>	Unresolved

### Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.7;
```

### Recommendation

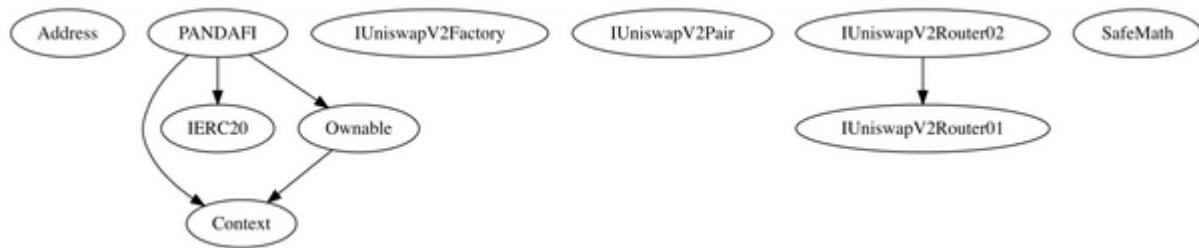
The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

# Functions Analysis

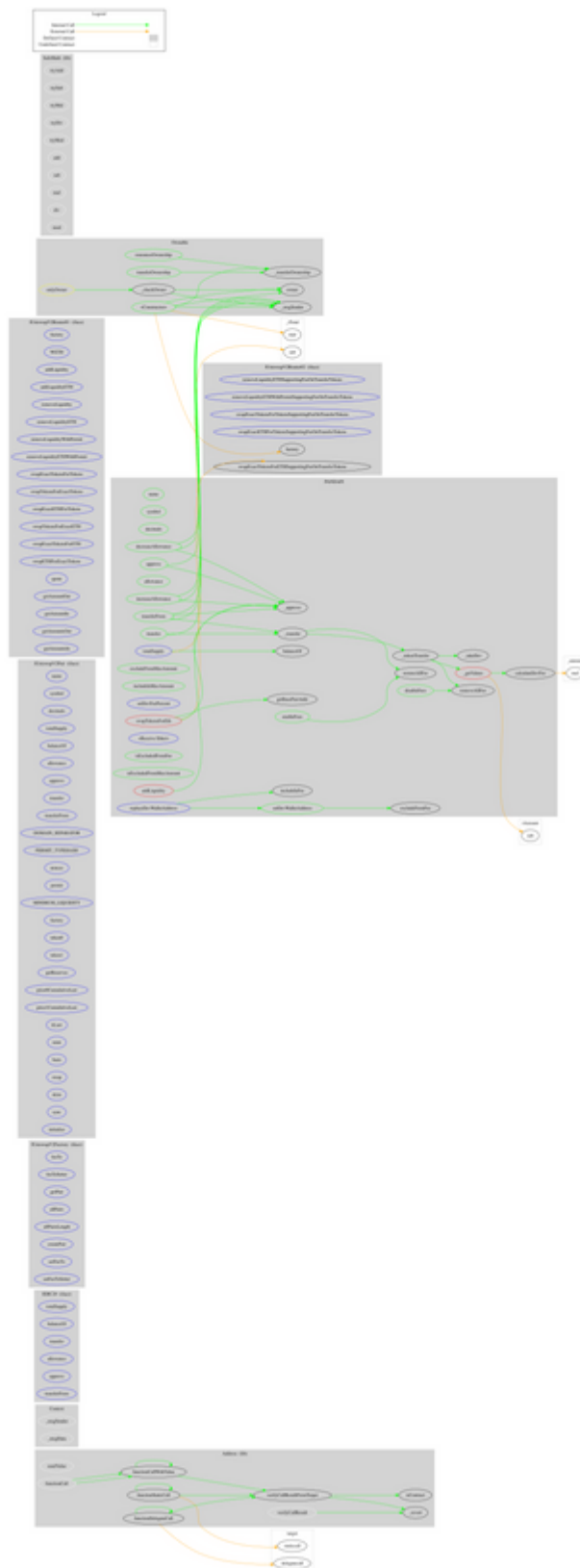
Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
PANDAFI	Implementation	Context, IERC20, Ownable		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	External		-
	balanceOf	Public		-
	getBasePairAddr	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	excludeFromFee	Public	✓	onlyOwner
	includeInFee	Public	✓	onlyOwner
	excludeFromMaxAmount	Public	✓	onlyOwner
	includeInMaxAmount	Public	✓	onlyOwner
	setDevFeePercent	External	✓	onlyOwner
	setDevWalletAddress	Public	✓	onlyOwner
	replaceDevWalletAddress	External	✓	onlyOwner
		External	Payable	-
	_getValues	Private		

	_takeDev	Private	✓	
	calculateDevFee	Private		
	removeAllFee	Private	✓	
	restoreAllFee	Private	✓	
	isExcludedFromFee	Public		-
	isExcludedFromMaxAmount	Public		-
	_approve	Private	✓	
	_transfer	Private	✓	
	swapTokensForEth	Private	✓	
	addLiquidity	Private	✓	
	_tokenTransfer	Private	✓	
	disableFees	Public	✓	onlyOwner
	enableFees	Public	✓	onlyOwner

# Inheritance Graph



# Flow Graph



## Summary

Panda Finance contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Panda Finance is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. There is also a limit of max 4% fees.

## Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. 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 Cyberscope's prior written consent. 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 regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document 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 report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment 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.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security. Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.



## About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.



The Cyberscope team

<https://www.cyberscope.io>