Cyberscope

Audit Report

Bionic Inu

December 2022

SHA256 34bb32d4508575967b10ed9db842a393c1944809ce9a833da6699d56d13c1fd8

db7f655f8cf535ccf63be1483375ab31075270399eae892c30ec166da981e3ed

Audited by © cyberscope

Table of Contents

Table of Contents 1
Review 3
Router Contract 3
Factory Contract 3
Audit Updates 4
Source Files 4
Swap Mechanism 5
Diagnostics 6
SFSE - Swap Fee Side Effects 7
Description 7
Recommendation 7
L04 - Conformance to Solidity Naming Conventions 8
Description 8
Recommendation 8
L09 - Dead Code Elimination 10
Description 10
Recommendation 10
L14 - Uninitialized Variables in Local Scope 11
Description 11
Recommendation 11
L16 - Validate Variable Setters 12
Description 12
Recommendation 12
L17 - Usage of Solidity Assembly 13
Description 13
Recommendation 13
L20 - Succeeded Transfer Check 14
Description 14
Recommendation 14
Functions Analysis 15
Inheritance Graph 20

Flow Graph 21
Summary 22
Disclaimer 23
About Cyberscope 24

Review

Router Contract

Contract Name	BioRouter
Compiler Version	v0.8.16+commit.07a7930e
Optimization	200 runs
Testing Deploy	https://testnet.bscscan.com/address/0x5df1c249f8927ce34a74255b37a96141e86b0213
Address	0x5dF1C249F8927ce34A74255b37A96141e86b0213
Network	BSC_TESTNET

Factory Contract

Contract Name	BioFactory
Compiler Version	v0.8.16+commit.07a7930e
Optimization	200 runs
Testing Deploy	https://testnet.bscscan.com/address/0x0643f973065709c636afd7a328f2f6a7c2602f5a
Address	0x0643f973065709c636afd7a328f2f6a7c2602f5a
Network	BSC_TESTNET

Audit Updates

Initial Audit	21 Dec 2022 https://github.com/cyberscope-io/audits/blob/main/bionic/v1/audit.pdf
Corrected Phase 2	27 Dec 2022

Source Files

Filename	SHA256
Factory.sol	34bb32d4508575967b10ed9db842a393c1944809ce9a833da6699d56d13c1fd8
Pair.sol	418d9f75e0b4263d87390915b8a39755f169284a3b8d63656d3e3dff5d5070d1
Router.sol	db7f655f8cf535ccf63be1483375ab31075270399eae892c30ec166da981e3ed

Swap Mechanism

The Bionic Inu implements a classic decentralized exchange swap mechanism. The implementation of the Router and Factory contracts is forked by the Uniswap repository. The team has implemented an additional feature. If the caller address is a holder of the Bionic token, then the swap is taxed by 0.25%, otherwise, it is taxed by 0.5%. This may produce some side effects regarding the applicable addresses that might receive the discount. Read more on the Swap Fee Side Effects finding.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	SFSE	Swap Fee Side Effects	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L09	Dead Code Elimination	Unresolved
	L14	Uninitialized Variables in Local Scope	Unresolved
	L16	Validate Variable Setters	Unresolved
	L17	Usage of Solidity Assembly	Unresolved
	L20	Succeeded Transfer Check	Unresolved

SFSE - Swap Fee Side Effects

Criticality	Minor / Informative
Location	Router.sol#L91
Status	Unresolved

Description

The swap fee for the Bionic router depends on the caller's address. If the caller is a holder of the Bionic token, then 0.25% tax will be applied, otherwise 0.5%.

The swap methods are commonly used by:

- Swap aggregators
- Trading Bots
- Swap proxies
- During the transaction of a Token

All of the above-mentioned contracts are working as a proxy between the caller and the Bionic router. If the intermediate contract holds at least 1 bionic token, then all the issuers will swap with the discounted fee.

Recommendation

The team is advised to carefully check the different ways that the Bionic router could be called.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Router.sol#L128,129 Factory.sol#L16,49,54
Status	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 your 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 your code.
7. Keep lines short (around 120 characters) to improve readability.

```
address public immutable WETH
address public immutable BIONIC
address _feeTo
address _feeToSetter
```

Recommendation

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

You can find more information on the Solidity documentation https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	Router.sol#L8
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 safeApprove(address token, address to, uint value) internal {
        // bytes4(keccak256(bytes('approve(address,uint256)')));
        (bool success, bytes memory data) =
token.call(abi.encodeWithSelector(0x095ea7b3, to, value));
        require(success && (data.length == 0 || abi.decode(data, (bool))),
'TransferHelper: APPROVE_FAILED');
}
```

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.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	Router.sol#L106,333,444
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in your contract. It's important to always initialize local variables with appropriate values before using them.

```
uint i
uint i
uint i
```

Recommendation

By initializing local variables before using them, you can help ensure that your contract functions behave as expected and avoid potential issues.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	Router.sol#L137,138,139 Factory.sol#L24,25,51,56
Status	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.

```
factory = _factory
WETH = _WETH
BIONIC = _BIONIC
feeToSetter = _feeToSetter
feeTo = _feeTo
```

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.

L17 - Usage of Solidity Assembly

Criticality	Minor / Informative
Location	Factory.sol#L39
Status	Unresolved

Description

Using assembly can be useful for optimizing code, but it can also be error-prone. It's important to carefully test and debug assembly code to ensure that it is correct and does not contain any errors.

Some common types of errors that can occur when using assembly in Solidity include Syntax, Type, Out-of-bounds, Stack, and Revert.

```
assembly {
            pair := create2(0, add(bytecode, 32), mload(bytecode), salt)
}
```

Recommendation

It is recommended to use assembly sparingly and only when necessary, as it can be difficult to read and understand compared to Solidity code.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	Router.sol#L227
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
IBioPair(pair).transferFrom(msg.sender, pair, liquidity)
```

Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IBioPair	Interface			
	initialize	External	✓	-
BioFactory	Implementation			
		Public	✓	-
	allPairsLength	External		-
	createPair	External	✓	-
	setFeeTo	External	✓	-
	setFeeToSetter	External	✓	-
BioERC20	Implementation			
		Public	✓	-
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Private	✓	
	_transfer	Private	✓	
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	permit	External	✓	-
Math	Library			
	min	Internal		
	sqrt	Internal		

UQ112x112	Library			
	encode	Internal		
	uqdiv	Internal		
IERC20	Interface			
	name	External		-
	symbol	External		-
	decimals	External		-
	totalSupply	External		-
	balanceOf	External		-
	allowance	External		-
	approve	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
IBioFactory	Interface			
	feeTo	External		-
IBioCallee	Interface			
	bioCall	External	✓	-
BioPair	Implementation	BioERC20		
	getReserves	Public		-
	_safeTransfer	Private	✓	
		Public	✓	-
	initialize	External	✓	-
	_update	Private	✓	
	_mintFee	Private	✓	

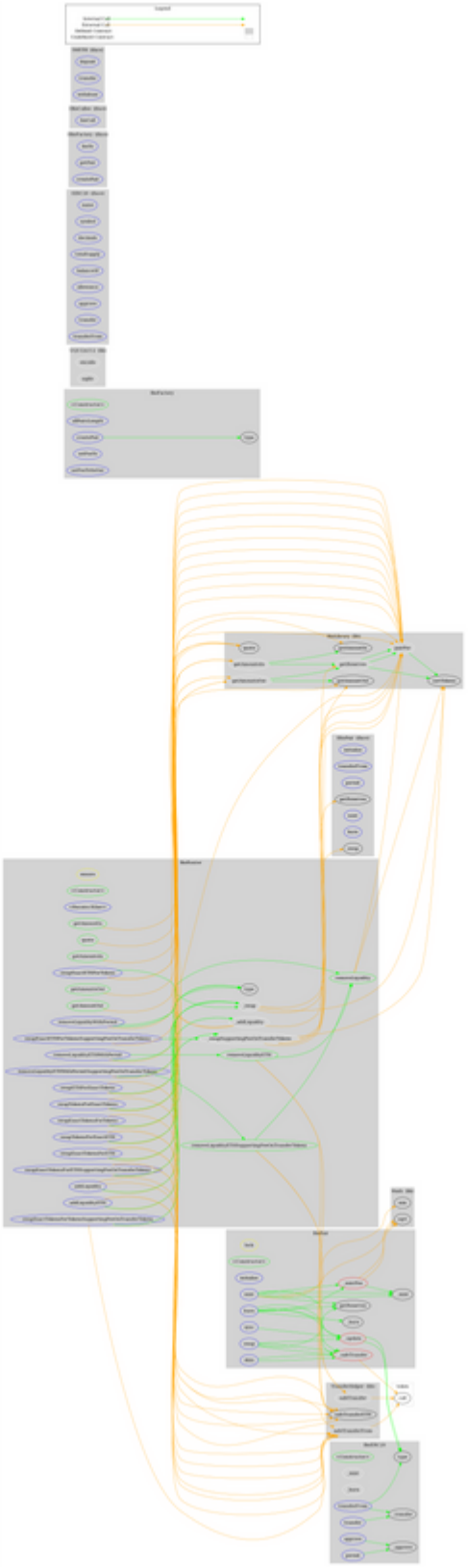
	mint	External	✓	lock
	burn	External	✓	lock
	swap	External	✓	lock
	skim	External	✓	lock
	sync	External	✓	lock
TransferHelper	Library			
	safeTransfer	Internal	✓	
	safeTransferFrom	Internal	✓	
	safeTransferETH	Internal	✓	
IBioFactory	Interface			
	getPair	External		-
	createPair	External	✓	-
IBioPair	Interface			
	transferFrom	External	✓	-
	permit	External	✓	-
	getReserves	External		-
	mint	External	✓	-
	burn	External	✓	-
	swap	External	✓	-
BioLibrary	Library			
	sortTokens	Internal		
	pairFor	Internal		
	getReserves	Internal		
	quote	Internal		
	getAmountOut	Internal		

	getAmountIn	Internal		
	getAmountsOut	Internal		
	getAmountsIn	Internal		
IWETH	Interface			
	deposit	External	Payable	-
	transfer	External	✓	-
	withdraw	External	✓	-
BioRouter	Implementation			
		Public	✓	-
		External	Payable	-
	_addLiquidity	Internal	✓	
	addLiquidity	External	✓	ensure
	addLiquidityETH	External	Payable	ensure
	removeLiquidity	Public	✓	ensure
	removeLiquidityETH	Public	✓	ensure
	removeLiquidityWithPermit	External	✓	-
	removeLiquidityETHWithPermit	External	✓	-
	removeLiquidityETHSupportingFeeOnTransferTokens	Public	✓	ensure
	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External	✓	-
	_swap	Internal	✓	
	swapExactTokensForTokens	External	✓	ensure
	swapTokensForExactTokens	External	✓	ensure
	swapExactETHForTokens	External	Payable	ensure
	swapTokensForExactETH	External	✓	ensure
	swapExactTokensForETH	External	✓	ensure
	swapETHForExactTokens	External	Payable	ensure
	_swapSupportingFeeOnTransferToke	Internal	✓	

	ns			
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	ensure
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	ensure
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	ensure
	quote	Public		-
	getAmountOut	Public		-
	getAmountIn	Public		-
	getAmountsOut	Public		-
	getAmountsIn	Public		-

Inheritance Graph

Flow Graph



Summary

Bionic Inu is a decentralized exchange that offers a wide variety of services. It provides a full suite of tools for users and partners to take advantage of decentralized finance opportunities. Cyberscope's audit focuses on the Router and Factory features. The audit investigates the main features, mentions security recommendation, performance improvements and potential optimizations.

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