



Cyberscope

Audit Report

YachtingVerse

April 2023

SHA256 5cbfa6bf559fa95d28bfe32cdd23905cd519653d250d0e1a48b4fa2542519d62

Audited by © cyberscope

Table of Contents

Table of Contents	1
Review	3
Audit Updates	3
Source Files	3
Findings Breakdown	4
Analysis	5
BC - Blacklists Addresses	6
Description	6
Recommendation	6
Team Update	6
Diagnostics	7
RAV - Router-Pair Argument Validation	8
Description	8
Recommendation	8
L04 - Conformance to Solidity Naming Conventions	9
Description	9
Recommendation	10
L07 - Missing Events Arithmetic	11
Description	11
Recommendation	11
L16 - Validate Variable Setters	12
Description	12
Recommendation	12
L18 - Multiple Pragma Directives	13
Description	13
Recommendation	13
L19 - Stable Compiler Version	14
Description	14
Recommendation	14
L20 - Succeeded Transfer Check	15
Description	15
Recommendation	15
Functions Analysis	16
Inheritance Graph	19
Flow Graph	20
Summary	21
Disclaimer	22
About Cyberscope	23

Review

Contract Name	YachtingVerse
Testing Deploy	https://testnet.bscscan.com/address/0xa9548633f5d52fd3a7be0293739e652a425e60e0
Symbol	YACHT
Decimals	18
Total Supply	250,000,000

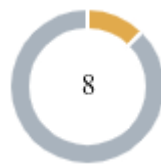
Audit Updates

Initial Audit	12 Feb 2022 https://github.com/cyberscope-io/audits/blob/main/yacht/v1/audit.pdf
Corrected Phase 2	26 Feb 2022 https://github.com/cyberscope-io/audits/blob/main/yacht/v2/audit.pdf
Corrected Phase 3	19 Apr 2023 https://github.com/cyberscope-io/audits/blob/main/yacht/v3/audit.pdf
Corrected Phase 4	25 Apr 2023

Source Files

Filename	SHA256
contracts/YachtingVerse.sol	5cbfa6bf559fa95d28bfe32cdd23905cd519653d250d0e1a48b4fa2542519d62

Findings Breakdown



Critical	0
Medium	1
Minor / Informative	7

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	1	0	0
Minor / Informative	7	0	0	0

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	Acknowledged

BC - Blacklists Addresses

Criticality	Medium
Location	YachtingVerse.sol#L327
Status	Acknowledged

Description

The contract owner has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `addRemoveBlacklist` function.

```
function addRemoveBlacklist(address _sinner) external onlyOwner {  
    blacklist[_sinner] = !blacklist[_sinner];  
}
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.
- Renouncing the ownership will eliminate the threats but it is non-reversible.

Team Update

The team has replied with the following statement:

"We haven't changed the blacklist function but will be securing the owner wallet using Gnosis Safe."

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	RAV	Router-Pair Argument Validation	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L18	Multiple Pragma Directives	Unresolved
●	L19	Stable Compiler Version	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

RAV - Router-Pair Argument Validation

Criticality	Minor / Informative
Location	YachtingVerse.sol#L321
Status	Unresolved

Description

The contract does not validate if the pair address has a valid pair with the router's native token. This lack of validation can lead to unintended behavior and potential security vulnerabilities.

```
function setPancake(address _router, address _pair) external onlyOwner {
    require(_router != address(0) && _pair != address(0), "Router and pair
cannot be address zero");
    pancakeV2Router = _router;
    pancakePair = _pair;
}
```

Recommendation

The team is advised to add proper validation checks to avoid any unintended behavior or potential security vulnerabilities.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/YachtingVerse.sol#L287,301,307,311,316,321,327,331,364
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 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.

```
mapping(address => bool) _isExcludedFromFees
uint _sellFee
uint _buyFee
address _excluded
address _devWallet
uint _timeLimit
address _router
address _pair
address _sinner
uint _amount
address _token
```

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>.

L07 - Missing Events Arithmetic

Criticality	Minor / Informative
Location	contracts/YachtingVerse.sol#L303,318,333
Status	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.

```
sellFee = _sellFee  
txTimeLimit = _timeLimit  
maxTxAmount = _amount
```

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.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	contracts/YachtingVerse.sol#L293
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.

```
devWallet = _devWallet
```

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.

L18 - Multiple Pragma Directives

Criticality	Minor / Informative
Location	contracts/YachtingVerse.sol#L3,28,40,50,220,234
Status	Unresolved

Description

If the contract includes multiple conflicting pragma directives, it may produce unexpected errors. To avoid this, it's important to include the correct pragma directive at the top of the contract and to ensure that it is the only pragma directive included in the contract.

```
pragma solidity 0.8.7;  
pragma solidity ^0.8.0;
```

Recommendation

It is important to include only one pragma directive at the top of the contract and to ensure that it accurately reflects the version of Solidity that the contract is written in.

By including all required compiler options and flags in a single pragma directive, the potential conflicts could be avoided and ensure that the contract can be compiled correctly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/YachtingVerse.sol#L28,40,50,220,234
Status	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 ^0.8.0;
```

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.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	contracts/YachtingVerse.sol#L365
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.

```
IERC20(_token).transfer(_msgSender(), _amount)
```

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](#).

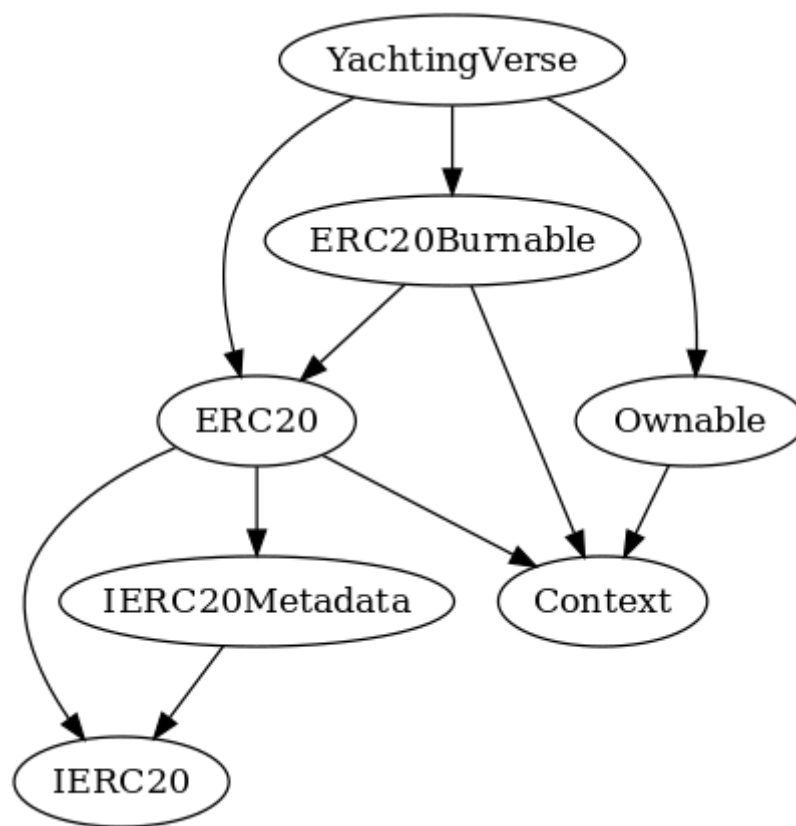
Functions Analysis

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-

ERC20	Implementation	Context, IERC20, IERC20Meta data		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_afterTokenTransfer	Internal	✓	
ERC20Burnable	Implementation	Context, ERC20		

	burn	Public	✓	-
	burnFrom	Public	✓	-
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
YachtingVerse	Implementation	ERC20, ERC20Burnable, Ownable		
		Public	✓	ERC20
	openMarket	External	✓	onlyOwner
	setSellBuyFee	External	✓	onlyOwner
	excludeFromFees	External	✓	onlyOwner
	setDevWallet	External	✓	onlyOwner
	setTxTimeLimit	External	✓	onlyOwner
	setPancake	External	✓	onlyOwner
	addRemoveBlacklist	External	✓	onlyOwner
	setMaxTxAmount	External	✓	onlyOwner
	_transfer	Internal	✓	
	recoverERC20	External	✓	onlyOwner

Inheritance Graph



Flow Graph



Summary

YachtingVerse contract implements a token mechanism. This audit investigates security issues, business logic concerns, and potential improvements. There are some functions that can be abused by the owner like blacklist addresses. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats. There is also a limit of max 24% fee.

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>