



Cyberscope

# Audit Report

## **ZeusPet**

February 2023

Type       BEP20

Network     BSC

Address     0xd1922Ce67C82dF05E73F72e9dae2d386ACc7Dd5E

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>3</b>
<b>Analysis</b>	<b>4</b>
<b>OTUT - Transfers User's Tokens</b>	<b>5</b>
Description	5
Recommendation	5
<b>ELFM - Exceeds Fees Limit</b>	<b>6</b>
Description	6
Recommendation	6
<b>Diagnostics</b>	<b>7</b>
<b>IDI - Immutable Declaration Improvement</b>	<b>8</b>
Description	8
Recommendation	8
<b>L04 - Conformance to Solidity Naming Conventions</b>	<b>9</b>
Description	9
Recommendation	9
<b>L07 - Missing Events Arithmetic</b>	<b>10</b>
Description	10
Recommendation	10
<b>L16 - Validate Variable Setters</b>	<b>11</b>
Description	11
Recommendation	11
<b>L18 - Multiple Pragma Directives</b>	<b>12</b>
Description	12
Recommendation	12
<b>L19 - Stable Compiler Version</b>	<b>13</b>
Description	13
Recommendation	13
<b>Functions Analysis</b>	<b>14</b>

<b>Inheritance Graph</b>	<b>16</b>
<b>Flow Graph</b>	<b>17</b>
<b>Summary</b>	<b>18</b>
<b>Disclaimer</b>	<b>19</b>
<b>About Cyberscope</b>	<b>20</b>

## Review

<b>Contract Name</b>	ZeusPet
<b>Compiler Version</b>	v0.8.17+commit.8df45f5f
<b>Optimization</b>	200 runs
<b>Explorer</b>	<a href="https://bscscan.com/address/0xd1922ce67c82df05e73f72e9dae2d386acc7dd5e">https://bscscan.com/address/0xd1922ce67c82df05e73f72e9dae2d386acc7dd5e</a>
<b>Address</b>	0xd1922ce67c82df05e73f72e9dae2d386acc7dd5e
<b>Network</b>	BSC
<b>Symbol</b>	ZPET
<b>Decimals</b>	18
<b>Total Supply</b>	21,000,000,000

## Audit Updates

<b>Initial Audit</b>	23 Feb 2023
----------------------	-------------

## Source Files

<b>Filename</b>	SHA256
<b>ZeusPet.sol</b>	d7e5f920b5dc4e66030754243967e0999dd591e5a4cc408f2f43cabe9e278eaf

# Analysis

● Critical ● Medium ● Minor / Informative ● Pass

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

## OCTD - Transfers Contract's Tokens

<b>Criticality</b>	Minor / Informative
<b>Location</b>	ZeusPet.sol#L251
<b>Status</b>	Unresolved

### Description

The contract owner has the authority to claim all the balance of the contract. The owner may take advantage of it by calling the `rescueToken` function.

```
function rescueToken(IERC20 token, address to, uint256 amount)
    external
    onlyOwner
{
    require(token.transfer(to, amount), "Unable to transfer token");
}
```

### 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.

## ELFM - Exceeds Fees Limit

<b>Criticality</b>	Critical
<b>Location</b>	ZeusPet.sol#L231
<b>Status</b>	Unresolved

### Description

The contract owner has the authority to increase over the allowed limit of 25%. The owner may take advantage of it by calling the `setFeePercent` function with a high percentage value.

```
function setFeePercent(uint256 mktPercent, uint256 burnPercent)
    external
    onlyOwner
{
    mktPct = mktPercent;
    burnPct = burnPercent;
}
```

### Recommendation

The contract could embody a check for the maximum acceptable value. 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 threads but it is non-reversible.

# Diagnostics

● Critical   ● Medium   ● Minor / Informative

Severity	Code	Description	Status
●	IDI	Immutable Declaration Improvement	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



## IDI - Immutable Declaration Improvement

<b>Criticality</b>	Minor / Informative
<b>Location</b>	ZeusPet.sol#L222,223
<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
```

### 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.

## L04 - Conformance to Solidity Naming Conventions

<b>Criticality</b>	Minor / Informative
<b>Location</b>	ZeusPet.sol#L239
<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 _marketingWallet
```

### 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

<b>Criticality</b>	Minor / Informative
<b>Location</b>	ZeusPet.sol#L235
<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.

```
mktPct = mktPercent
```

### 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

<b>Criticality</b>	Minor / Informative
<b>Location</b>	ZeusPet.sol#L226
<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.

```
marketingWallet = _marketingWallet
```

### 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

<b>Criticality</b>	Minor / Informative
<b>Location</b>	ZeusPet.sol#L2,87
<b>Status</b>	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 ^0.8.9;  
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

<b>Criticality</b>	Minor / Informative
<b>Location</b>	ZeusPet.sol#L2,87
<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 ^0.8.9;  
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.

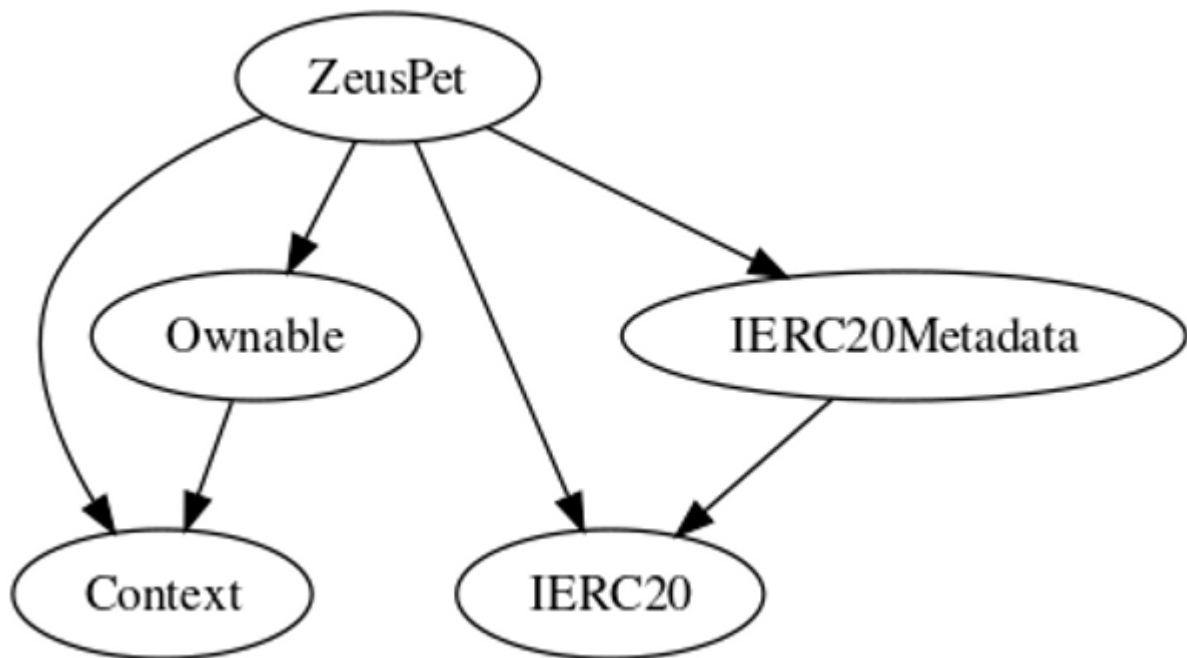
# Functions Analysis

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
<b>Context</b>	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
<b>Ownable</b>	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
<b>IERC20</b>	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
<b>IERC20Metadata</b>	Interface	IERC20		
	name	External		-
	symbol	External		-

	decimals	External		-
<b>ZeusPet</b>	Implementation	Context, IERC20, IERC20Met adata, Ownable		
		Public	✓	-
	setFeePercent	External	✓	onlyOwner
	setMarketingWallet	External	✓	onlyOwner
	setExcludeFromFee	Public	✓	onlyOwner
	rescueToken	External	✓	onlyOwner
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_mint	Internal	✓	
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_transfer	Internal	✓	
	_takeFee	Internal	✓	



## Inheritance Graph



# Flow Graph



## Summary

There are some functions that can be abused by the owner like draining the contract's tokens and manipulating the fees. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

## 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>