Cyberscope

Audit Report
ZeusPet

February 2023

Type	BEP20
Network	BSC
Address	0xd1922Ce67C82dF05E73F72e9dae2d386ACc7Dd5E
Audited by	© cyberscope

Table of Contents

Review

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

Audit Updates

Initial Audit	23 Feb 2023

Source Files

Filename	SHA256
ZeusPet.sol	d7e5f920b5dc4e66030754243967e0999dd591e5a4cc408f2f43cabe9e278eaf

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

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

OCTD - Transfers Contract's Tokens

Criticality	Minor / Informative
Location	ZeusPet.sol#L251
Status	Multisign

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.

Team Update

The team has implemented our recommendation by adopting a multi-signature wallet at the following transaction
https://bscscan.com/tx/0xdee6c8c9d2325b6127a516f1c2aeb281a4434424888b347a3616a5860f513dd4

ELFM - Exceeds Fees Limit

Criticality	Critical
Location	ZeusPet.sol#L231
Status	Multisign

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.

Team Update

The team has implemented our recommendation by adopting a multi-signature wallet at the following transaction
https://bscscan.com/tx/0xdee6c8c9d2325b6127a516f1c2aeb281a4434424888b347a3616a5860f513dd4

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

Criticality	Minor / Informative
Location	ZeusPet.sol#L222,223
Status	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

Criticality	Minor / Informative
Location	ZeusPet.sol#L239
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.

```
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

Criticality	Minor / Informative
Location	ZeusPet.sol#L235
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.

```
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

Criticality	Minor / Informative
Location	ZeusPet.sol#L226
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.

```
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

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

Criticality	Minor / Informative
Location	ZeusPet.sol#L2,87
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 ^0.8.9;
pragma solidity ^0.8.0;
```

Recommendation

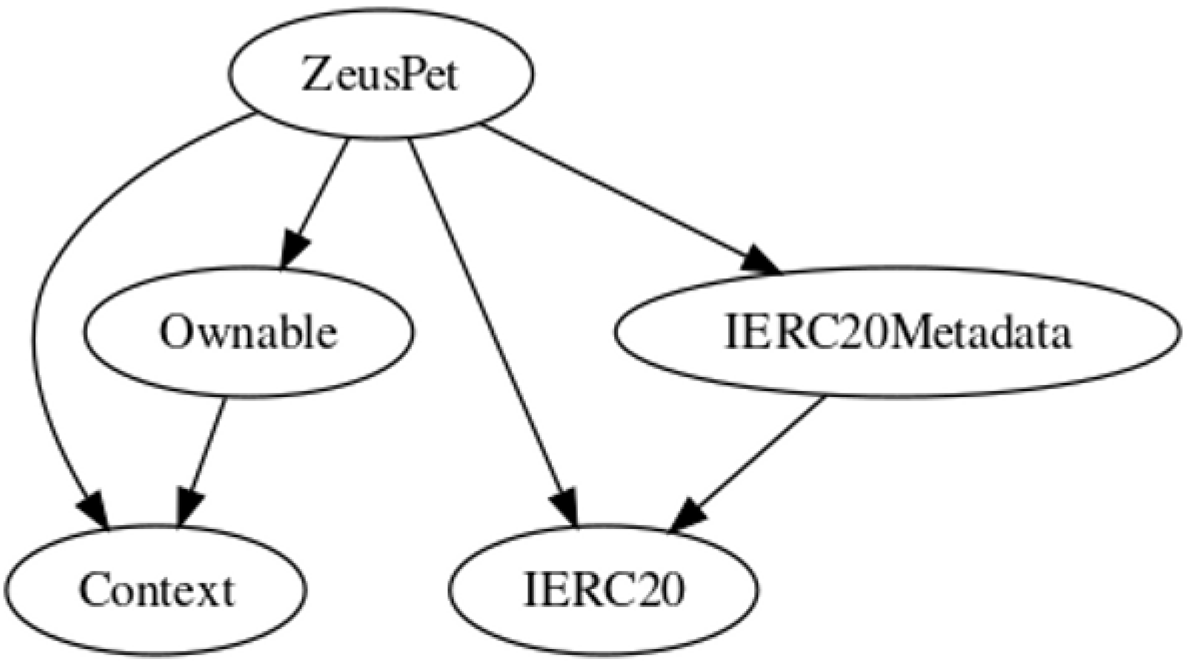
The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest 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**
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-

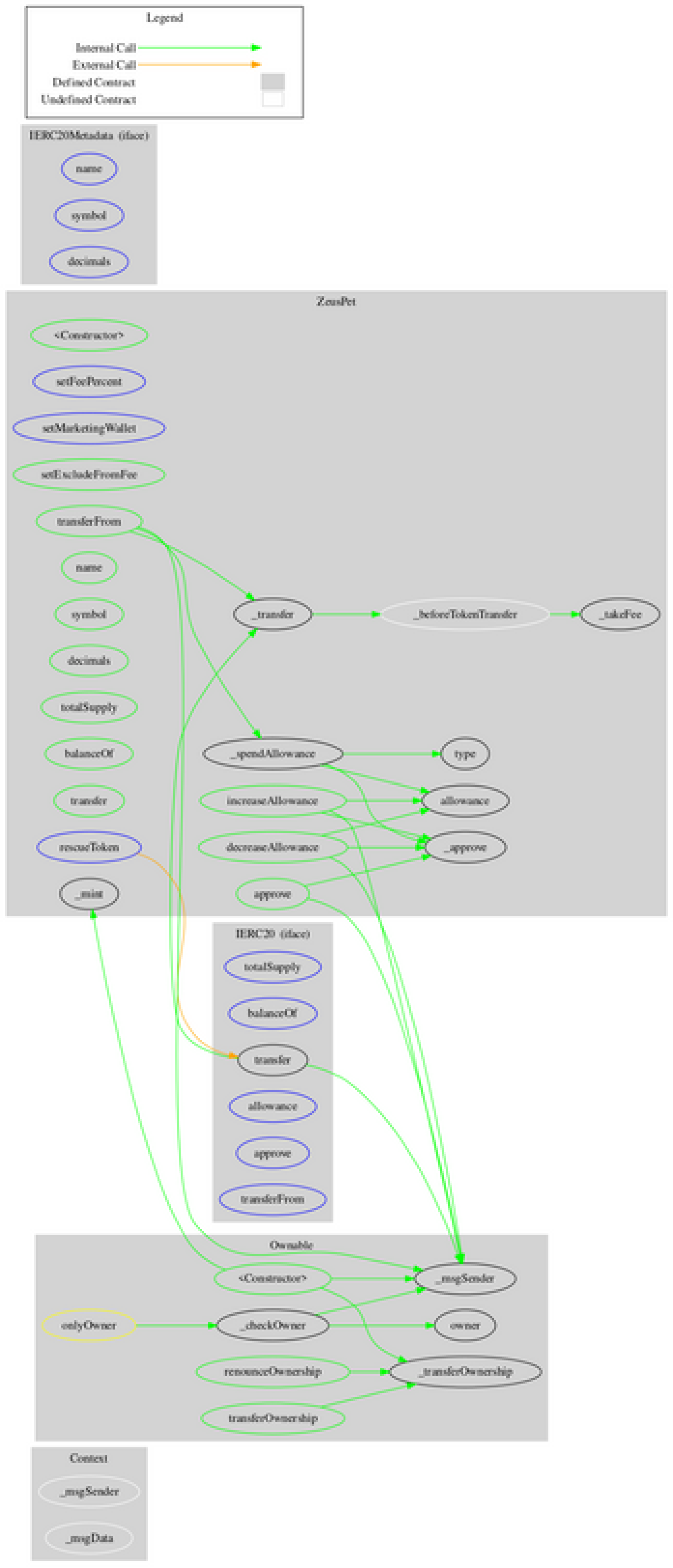
	decimals	External		-
ZeusPet	Implementation	Context, IERC20, IERC20Metadata, Ownable		
		Public	✓	-
	setFeePercent	External	✓	onlyOwner
	setMarketingWallet	External	✓	onlyOwner
	setExcludeFromFee	Public	✓	onlyOwner
	rescueToken	External	✓	onlyOwner
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	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



$$\begin{tikzcd}
 & \text{ZeusPet} \arrow[dl] \arrow[dr] \arrow[drr] \arrow[ddl, bend right] & & \\
\text{Ownable} \arrow[d] & & \text{IERC20} & \text{IERC20Metadata} \arrow[l] \\
\text{Context} & & &
\end{tikzcd}$$

Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IERC20Metadata (iface)
name
symbol
decimals
ZeusPet
<Constructor>
setFeePercent
setMarketingWallet
setExcludeFromFee
transferFrom
name
symbol
decimals
totalSupply
balanceOf
transfer
rescueToken
_mint
_transfer
_beforeTokenTransfer
_takeFee
_spendAllowance
type
increaseAllowance
allowance
decreaseAllowance
_approve
approve
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Ownable
<Constructor>
_msgSender
onlyOwner
_checkOwner
owner
renounceOwnership
_transferOwnership
transferOwnership
Context
_msgSender
_msgData

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.

The team has implemented our recommendation by adopting a multi-signature wallet at the following transaction https://bscscan.com/tx/0xdee6c8c9d2325b6127a516f1c2aeb281a4434424888b347a3616a5860f513dd4

Disclaimer

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