Cyberscope

Audit Report

The Worldwide Token

July 2023

SHA256	a7fdf03b661cfba9a3ec137d737b43a37a66ecb996b98da80f9df2bb070b0e90
Audited by	© cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Passed
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Unresolved
●	MT	Mints Tokens	Unresolved
●	BT	Burns Tokens	Unresolved
●	BC	Blacklists Addresses	Unresolved

Diagnostics

🔴 Critical 🟠 Medium ⚪ Minor / Informative

Severity	Code	Description	Status
🟠	TFD	Transfer Functions Distinction	Unresolved
⚪	MAF	Misleading APY Functionality	Unresolved
⚪	MEE	Missing Events Emission	Unresolved
⚪	L02	State Variables could be Declared Constant	Unresolved
⚪	L04	Conformance to Solidity Naming Conventions	Unresolved
⚪	L09	Dead Code Elimination	Unresolved
⚪	L11	Unnecessary Boolean equality	Unresolved
⚪	L13	Divide before Multiply Operation	Unresolved

Table of Contents

Review

Contract Name	BEP20Token
Testing Deploy	https://testnet.bscscan.com/address/0xc3f8b61d7b11ca1d2dd9eceb1fd066bce071ad66
Symbol	WORLD
Decimals	4

Audit Updates

Initial Audit	07 Jul 2023

Source Files

Filename	SHA256
contracts/WorldToken.sol	a7fdf03b661cfba9a3ec137d737b43a37a66ecb996b98da80f9df2bb070b0e90

Findings Breakdown

●	Critical	4
●	Medium	1
●	Minor / Informative	7

Severity		Unresolved	Acknowledged	Resolved	Other
●	Critical	4	0	0	0
●	Medium	1	0	0	0
●	Minor / Informative	7	0	0	0

ELFM - Exceeds Fees Limit

Criticality	Critical
Location	contracts/WorldToken.sol#L392
Status	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 `changeTax` function with a high percentage value.

```
function changeTax(uint256 newTax) public onlyOwner returns (bool) {
    tax = newTax;
}
```

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 threats but it is non-reversible.

MT - Mints Tokens

Criticality	Critical
Location	contracts/WorldToken.sol#L709,717
Status	Unresolved

Description

The contract owner has the authority to mint tokens. The owner may take advantage of it by calling the `mint` or `mintToAccount` function. As a result, the contract tokens will be highly inflated.

```
function mint(uint256 amount) public onlyOwner returns (bool) {
  _mint(_msgSender(), amount);
  if (!isHolder(_msgSender())) {
      holders.push(_msgSender());
  }
  return true;
}

function mintToAccount(address addr, uint256 amount) public onlyOwner
returns (bool) {
  _mint(address(addr), amount);
  if (!isHolder(addr)) {
      holders.push(addr);
  }
  return true;
}
```

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.

BT - Burns Tokens

Criticality	Critical
Location	contracts/WorldToken.sol#L660
Status	Unresolved

Description

The contract owner has the authority to burn tokens from a specific address. The owner may take advantage of it by calling the `burnFrom` function. As a result, the targeted address will lose the corresponding tokens.

```
function burnFrom(address addr, uint256 amount) public onlyOwner returns
(bool) {
  _burn(addr, amount);
  return true;
}
```

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.

BC - Blacklists Addresses

Criticality	Critical
Status	Unresolved

Description

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

```
function addToBlockList(address wallet) public onlyOwner returns (bool) {
  if(isInBlockList(wallet) == false){
     BlockList.push(wallet);
  }
}
```

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.

TFD - Transfer Functions Distinction

Criticality	Medium
Location	contracts/WorldToken.sol#L601,655
Status	Unresolved

Description

The `transfer` and `transferFrom` functions of an ERC20 token are used to transfer tokens from one user to another.The contract implements both functions. However, there is a distinction between the implementation of each function. For instance, the `transferFrom` function only transfers the given amount from the sender to the recipient, while the `transfer` function has additional functionality, like a fee mechanism and allows transaction only if the `msg.sender` is not blacklisted. As a result, the functions implementation is not consistent.

```
function transfer(address recipient, uint256 amount) external returns
(bool) {
  if(isInBlockList(msg.sender) == false){
    uint256 percentage = tax/4;
    uint256 amountTax = (amount * percentage / 100);
    uint256 transferAmount = amount - (amountTax*4);
    if(isInTransferAllowList(msg.sender) == false){
      _burn(msg.sender, amountTax);
      _transfer(_msgSender(), liquidCenterWallet, amountTax);
      _transfer(_msgSender(), worldwideTreasuryWallet, amountTax);
      _transfer(_msgSender(), teamWallet, amountTax);
    } else {
      transferAmount = amount;
    }

    _transfer(_msgSender(), recipient, transferAmount);

    if (!isHolder(recipient)) {
        holders.push(recipient);
    }
  }
  return true;
}

function transferFrom(address sender, address recipient, uint256 amount)
public onlyOwner returns (bool) {
  _transfer(sender, recipient, amount);
  return true;
}
```

Recommendation

The team is advised to ensure that the implementation of the `transfer` and `transferFrom` functions is consistent.

MAF - Misleading APY Functionality

Criticality	Minor / Informative
Location	contracts/WorldToken.sol#L396,404
Status	Unresolved

Description

The functions `calculateAPY` and `distributeAPY` are misleadingly named as they do not actually calculate the Annual Percentage Yield (APY) as expected. Instead, they utilize a fixed apyPercentage value to mint tokens exponentially based on the holders' balances.

```
function calculateAPY(address holder) external view returns (uint256) { ... }
function distributeAPY() public onlyOwner returns (bool)  { ... }
```

Recommendation

The team is advised to take these segments into consideration and rewrite them to accurately reflect their purpose.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	contracts/WorldToken.sol#L425,431,437
Status	Unresolved

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
allowListTransferTax.push(wallet);
BlockList.push(wallet);
dexAddressList.push(wallet);
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	contracts/WorldToken.sol#L348,364,366
Status	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.

```
uint256 public totalAPY
uint256 public maxTotalSupply
address public magmaVaporizerWallet
```

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

Criticality	Minor / Informative
Location	contracts/WorldToken.sol#L358
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[] public BlockList
```

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.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	contracts/WorldToken.sol#L813
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.

```
internal {
    _burn(account, amount);
    _approve(account, _msgSender(),
_allowances[account][_msgSender()].sub(amount, "BEP20: burn amount exceeds
allowance"));
  }
}
```

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.

L11 - Unnecessary Boolean equality

Criticality	Minor / Informative
Location	contracts/WorldToken.sol#L424,430,436,602,606
Status	Unresolved

Description

Boolean equality is unnecessary when comparing two boolean values. This is because a boolean value is either true or false, and there is no need to compare two values that are already known to be either true or false.

it's important to be aware of the types of variables and expressions that are being used in the contract's code, as this can affect the contract's behavior and performance. The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
lowListTransferTax.push(wallet);


push(wallet);
    }
  }
...
  }

 percentage = tax/4;
      uint25

   _burn(msg.sender, amountTax);
```

Recommendation

Using the boolean value itself is clearer and more concise, and it is generally considered good practice to avoid unnecessary boolean equalities in Solidity code.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	contracts/WorldToken.sol#L603,604,605
Status	Unresolved

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of prediction.

```
uint256 transferAmount = amount - (amountT
     if(isInTransferAllowList(msg.sender) == fa
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

Functions Analysis

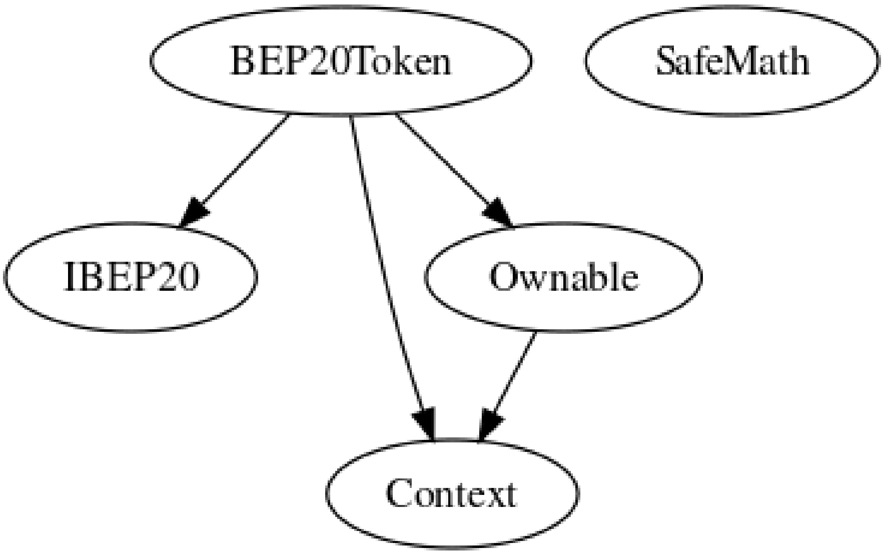
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IBEP20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
		Internal	✓	
	_msgSender	Internal		
	_msgData	Internal		
SafeMath	Library			

	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
	mod	Internal		
	mod	Internal		
Ownable	Implementation	Context		
		Internal	✓	
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
BEP20Token	Implementation	Context, IBEP20, Ownable		
		Public	✓	-
	changeTax	Public	✓	onlyOwner
	calculateAPY	External		-
	distributeAPY	Public	✓	onlyOwner
	addToTransferAllowList	Public	✓	onlyOwner
	addToBlockList	Public	✓	onlyOwner

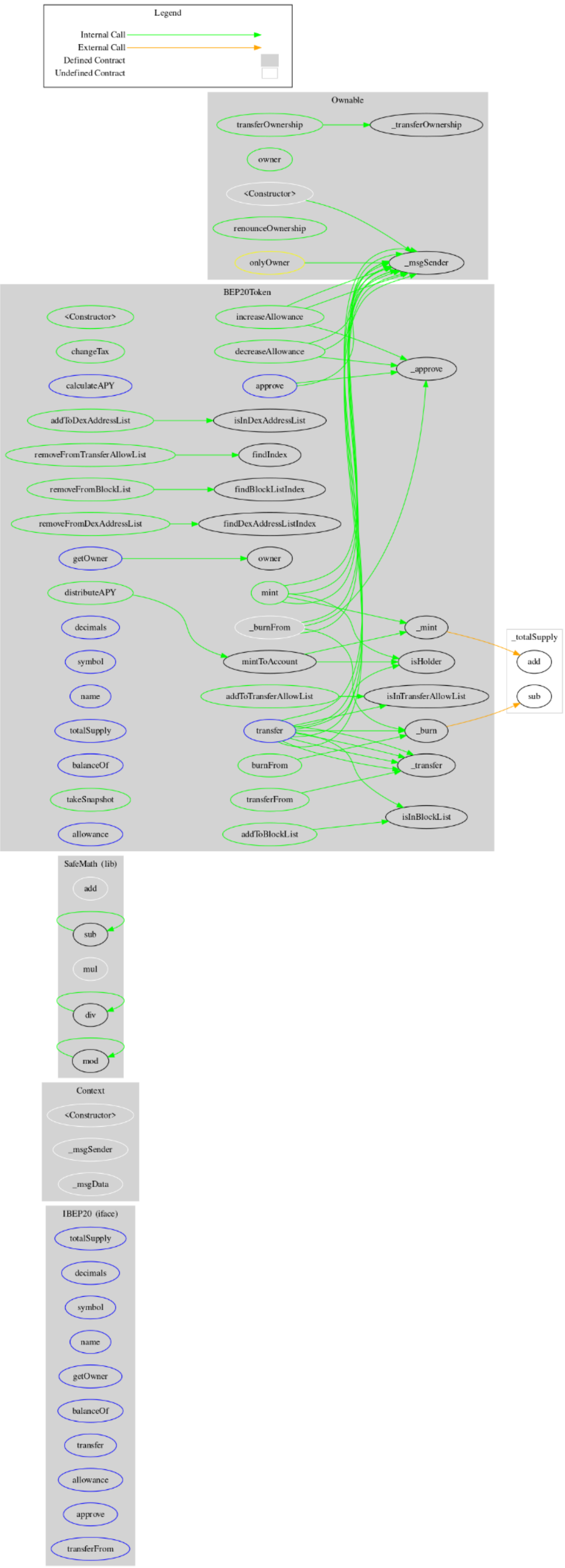
	addToDexAddressList	Public	✓	onlyOwner
	findIndex	Public		-
	findBlockListIndex	Public		-
	findDexAddressListIndex	Public		-
	removeFromTransferAllowList	Public	✓	onlyOwner
	removeFromBlockList	Public	✓	onlyOwner
	removeFromDexAddressList	Public	✓	onlyOwner
	isInTransferAllowList	Internal		
	isInDexAddressList	Internal		
	isHolder	Internal		
	isInBlockList	Internal		
	getOwner	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	totalSupply	External		-
	balanceOf	External		-
	takeSnapshot	Public	✓	onlyOwner
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	Public	✓	onlyOwner
	burnFrom	Public	✓	onlyOwner

	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	mint	Public	✓	onlyOwner
	mintToAccount	Public	✓	onlyOwner
	_transfer	Internal	✓	
	_mint	Internal	✓	
	_burn	Internal	✓	
	_approve	Internal	✓	
	_burnFrom	Internal	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Ownable
transferOwnership
_transferOwnership
owner
<Constructor>
renounceOwnership
onlyOwner
_msgSender
BEP20Token
<Constructor>
increaseAllowance
changeTax
decreaseAllowance
_approve
calculateAPY
approve
addToDexAddressList
isInDexAddressList
removeFromTransferAllowList
findIndex
removeFromBlockList
findBlockListIndex
removeFromDexAddressList
findDexAddressListIndex
getOwner
owner
distributeAPY
mint
decimals
_burnFrom
_mint
_totalSupply
add
symbol
mintToAccount
isHolder
name
addToTransferAllowList
isInTransferAllowList
sub
totalSupply
transfer
_burn
balanceOf
burnFrom
_transfer
takeSnapshot
transferFrom
isInBlockList
allowance
addToBlockList
SafeMath (lib)
add
sub
mul
div
mod
Context
<Constructor>
_msgSender
_msgData
IBEP20 (iface)
totalSupply
decimals
symbol
name
getOwner
balanceOf
transfer
allowance
approve
transferFrom

Summary

The Worldwide Token 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 manipulate the fees, mint tokens, burn tokens from any address and massively blacklist addresses. if the contract owner abuses the mint functionality, then the contract will be highly inflated. if the contract owner abuses the burn functionality, then the users could lost their tokens. 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