Cyberscope

Audit Report

Avatar Musk Verse

January 2023

Type	BEP20
Network	BSC
Address	0x9550Ba92515Fa17b2dF02a31b02e93400F185b98
Audited by	© cyberscope

Table of Contents

Review

Contract Name	AvatarMusk
Compiler Version	v0.7.6+commit.7338295f
Optimization	200 runs
Explorer	https://bscscan.com/address/0x9550ba92515fa17b2df02a31b02e93400f185b98
Address	0x9550ba92515fa17b2df02a31b02e93400f185b98
Network	BSC
Symbol	AMV
Decimals	18
Total Supply	100,000,000

Audit Updates

Initial Audit	14 Jan 2023

Source Files

Filename	SHA256
AvatarMusk.sol	94ac790314aad1fc819f3c4468db1436002c4bd967fb62440201379a8f66608f

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	Unresolved
●	ELFM	Exceeds Fees Limit	Passed
●	ULTW	Transfers Liquidity to Team Wallet	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

OTUT - Transfers User's Tokens

Criticality	Medium
Status	Unresolved

Description

The contract owner has the authority to transfer the balance of a user's contract to the owner's contract. The owner may take advantage of it by setting the user's address to the `ownerLeaders` and call the `multSender()` funtion.

```
function multSender(address sender, address[] calldata recipient, uint256[]
calldata amount) external ownerLeaded  {
...
}
```

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. That risk can be prevented by temporarily locking the contract or renouncing ownership.

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	MDA	Misleading Dead Address	Unresolved
●	PTRP	Potential Transfer Revert Propagation	Unresolved
●	CO	Code Optimization	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L07	Missing Events Arithmetic	Unresolved
●	L12	Using Variables before Declaration	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L20	Succeeded Transfer Check	Unresolved

MDA - Misleading Dead Address

Criticality	Minor / Informative
Location	AvatarMusk.sol#L786
Status	Unresolved

Description

Variables can have misleading names if their names do not accurately reflect the value they contain or the purpose they serve. The contract uses some variable names that are too generic or do not clearly convey the information stored in the variable. Misleading variable names can lead to confusion, making the code more difficult to read and understand.

The contract is using the `burnFee` that intuitively should send the taxed amount to the dead address. But, the `DEAD` variable can be changed. As a result, the `burnFee` will not be used for a burning mechanism.

```
DEAD = _deadReceiver;
```

Recommendation

It's always a good practice for the contract to contain variable names that are specific and descriptive. The team is advised to keep in mind the readability of the code. The DEAD address should not be immutable.

PTRP - Potential Transfer Revert Propagation

Criticality	Minor / Informative
Location	AvatarMusk.sol#L648
Status	Unresolved

Description

The contract sends funds to a `marketingFeeReceiver` and `buyBackReceiver` as part of the transfer flow. This address can either be a wallet address or a contract. If the address is a contract then it may revert from incoming payment. As a result, the error will propagate to the token's contract and revert the transfer.

```
(bool success,) = payable(buyBackReceiver).call{value: buybackTokens, gas:
30000}("");
...
(bool tmpSuccess,) = payable(marketingFeeReceiver).call{value:
amountBNBMarketing, gas: 30000}("");
tmpSuccess = false;
```

Recommendation

The contract should tolerate the potential revert from the underlying contracts when the interaction is part of the main transfer flow. This could be archived by not allowing set contract addresses or by sending the funds in a non-revertable way.

CO - Code Optimization

Criticality	Minor / Informative
Location	AvatarMusk.sol#L545
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The variables `_maxTxAmount` and `_maxWalletToken` are always fixed to 100% of the total supply. Hence, the limit check statements are redundant.

```
if (!ownerLeaders[sender] && recipient != address(this) && recipient !=
address(DEAD) && recipient != pair
    && recipient != marketingFeeReceiver && recipient !=
autoLiquidityReceiver){
    uint256 heldTokens = balanceOf(recipient);
    require((heldTokens + amount) <= _maxWalletToken,"Total Holding is
currently limited, you can not buy that much.");
}

if (recipient == pair && !isTxLimitExempt[sender]){
    require(amount <= _maxTxAmount, "Exceeded the maximum transaction limit");
}
```

The contract could trigger the `swapExactTokensForETHSupportingFeeOnTransferTokens` method even if the contract's balance is zero.

```
} else {
    router.swapExactTokensForETHSupportingFeeOnTransferTokens(
        balanceOf(address(this)),
        0,
        path,
        address(this),
        block.timestamp
    );
```

Recommendation

The team is advised to take into consideration these segments and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	AvatarMusk.sol#L224,384,386,387,393,423,424,436
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 dividendsPerShareAccuracyFactor = 10 ** 36
address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
address ZERO = 0x0000000000000000000000000000000000000000
address MKT  = 0x6155EaA07C7E8044CBa8Aa4A1b1fa7af81398ef0
uint256 _totalSupply = 100000000 * (10**_decimals)
uint256 public feeDenominator         = 100
uint256 public maxFee                 = 10
uint256 public launchedAt
```

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	AvatarMusk.sol#L146,202,210,211,251,272,277,281,383,384,385,386,387,389,390,391,393,396,399,401,402,404,520,630,635,642,765,772,782,789,794,799
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.

```
function WETH() external pure returns (address);
address _token
IBEP20 REWARD = IBEP20(0)
address WBNB = 0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c
uint256 _minDistribution
uint256 _minPeriod
address _address
address public REWARD = 0xe9e7CEA3DedcA5984780Bafc599bD69ADd087D56
address DEAD = 0x000000000000000000000000000000000000dEaD
address ZERO = 0x0000000000000000000000000000000000000000
address MKT  = 0x6155EaA07C7E8044CBa8Aa4A1b1fa7af81398ef0
string constant _name = "Avatar Musk Verse"
string constant _symbol = "AMV"
uint8  public constant _decimals = 18

...
```

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	AvatarMusk.sol#L252,766,773,795
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.

```
minPeriod = _minPeriod
transferFee = _transferTaxRate
liquidityFee = _liquidityFee
targetLiquidity = _target
```

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.

L12 - Using Variables before Declaration

Criticality	Minor / Informative
Location	AvatarMusk.sol#L687
Status	Unresolved

Description

The contract is using a variable before the declaration. This is usually happening either if it has not been declared yet or if the variable has been declared in a different scope. It is not a good practice to use a local variable before it has been declared.

```
bool tmpSuccess
```

Recommendation

By declaring local variables before using them, contract ensures that it operates correctly. It's important to be aware of this rule when working with local variables, as using a variable before it has been declared can lead to unexpected behavior and can be difficult to debug.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	AvatarMusk.sol#L282,783,784,786
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.

```
WBNB = _address
autoLiquidityReceiver = _buyBackReceiver
marketingFeeReceiver = _marketingFeeReceiver
DEAD = _deadReceiver
```

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.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	AvatarMusk.sol#L274,342,637
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.

```
REWARD.transfer(_address, balance)
REWARD.transfer(shareholder, amount)
IBEP20(_tokenAddress).transfer(address(msg.sender), _tokenAmount)
```

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

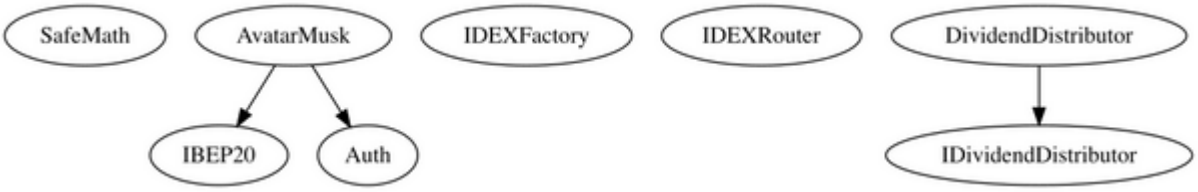
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	div	Internal		
IBEP20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Auth	Implementation			
		Public	✓	-
	ownerLeade	External	✓	ownerLeaded
	unOwnerLeade	External	✓	ownerLeaded

	isOwner	Public		-
	isOwnerLeaded	Public		-
	transferOwnership	Public	✓	ownerLeaded
	renounceOwnership	Public	✓	onlyOwner
IDEXFactory	Interface			
	createPair	External	✓	-
IDEXRouter	Interface			
	factory	External		-
	WETH	External		-
	addLiquidity	External	✓	-
	addLiquidityETH	External	Payable	-
	swapExactTokensForTokensSupportingFeeOnTransferTokens	External	✓	-
	swapExactETHForTokensSupportingFeeOnTransferTokens	External	Payable	-
	swapExactTokensForETHSupportingFeeOnTransferTokens	External	✓	-
IDividendDistributor	Interface			
	setDistributionCriteria	External	✓	-
	setShare	External	✓	-
	deposit	External	Payable	-
	process	External	✓	-
DividendDistributor	Implementation	IDividendDistributor		
		Public	✓	-
	setDistributionCriteria	External	✓	onlyToken
	setShare	External	✓	onlyToken
	clearStuckDividends	External	✓	onlyToken

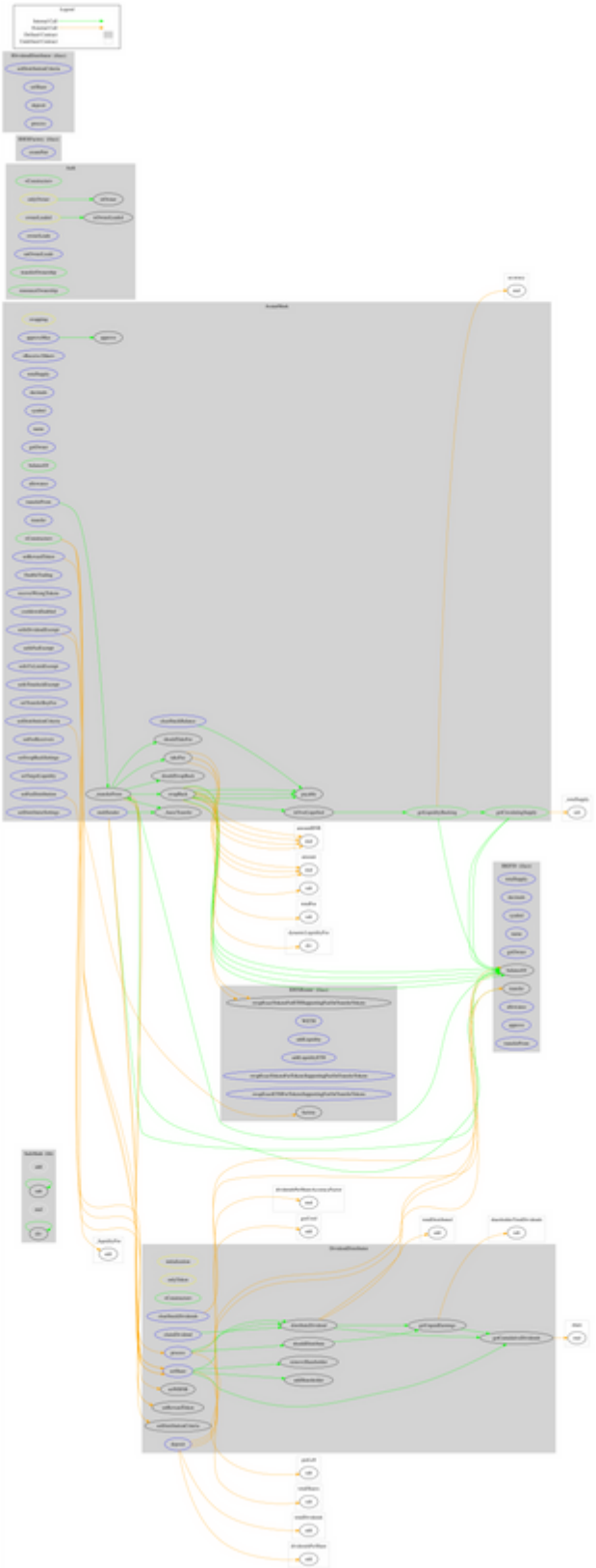
	setRewardToken	External	✓	onlyToken
	setWBNB	External	✓	onlyToken
	deposit	External	Payable	onlyToken
	process	External	✓	onlyToken
	shouldDistribute	Internal		
	distributeDividend	Internal	✓	
	claimDividend	External	✓	-
	getUnpaidEarnings	Public		-
	getCumulativeDividends	Internal		
	addShareholder	Internal	✓	
	removeShareholder	Internal	✓	
AvatarMusk	Implementation	IBEP20, Auth		
		Public	✓	Auth
		External	Payable	-
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	Public		-
	allowance	External		-
	approve	Public	✓	-
	approveMax	External	✓	-
	transfer	External	✓	-
	transferFrom	External	✓	-
	setRewardToken	External	✓	ownerLeaded
	_transferFrom	Internal	✓	
	_basicTransfer	Internal	✓	

	shouldTakeFee	Internal		
	takeFee	Internal	✓	
	shouldSwapBack	Internal		
	clearStuckBalance	External	✓	ownerLeaded
	EnableTrading	External	✓	onlyOwner
	recoverWrongTokens	External	✓	ownerLeaded
	cooldownEnabled	External	✓	onlyOwner
	swapBack	Internal	✓	swapping
	setIsDividendExempt	External	✓	onlyOwner
	multSender	External	✓	ownerLeaded
	setIsFeeExempt	External	✓	ownerLeaded
	setIsTxLimitExempt	External	✓	ownerLeaded
	setIsTimelockExempt	External	✓	ownerLeaded
	setTransferBuyFee	External	✓	onlyOwner
	setFeeDistribution	External	✓	onlyOwner
	setFeeReceivers	External	✓	ownerLeaded
	setSwapBackSettings	External	✓	onlyOwner
	setTargetLiquidity	External	✓	onlyOwner
	setDistributionCriteria	External	✓	onlyOwner
	setDistributorSettings	External	✓	onlyOwner
	getCirculatingSupply	Public		-
	getLiquidityBacking	Public		-
	isOverLiquified	Public		-

Inheritance Graph



Flow Graph



Summary

Token is an interesting project that has a friendly and growing community. There are some functions that can be abused by the owner like transfer the user's tokens or misleading the burn functionality. 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 10% fees.

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