Cyberscope

Audit Report

APSA

January 2023

Type	BEP20
Network	BSC
Address	0x8e1564a6c63E5422f42ab1229a46407928e19404
Audited by	© cyberscope

Table of Contents

Review

Contract Name	FullFeatureToken
Compiler Version	v0.8.7+commit.e28d00a7
Optimization	1337 runs
Explorer	https://bscscan.com/address/0x8e1564a6c63e5422f42ab1229a46407928e19404
Address	0x8e1564a6c63e5422f42ab1229a46407928e19404
Network	BSC
Symbol	A$T
Decimals	18
Total Supply	500,000,000,000

Audit Updates

Initial Audit	09 Jan 2023

Source Files

Filename	SHA256
@openzeppelin/contracts/access/Ownable.sol	9353af89436556f7ba8abb3f37a6677249aa4df6024fbfaa94f79ab2f44f3231
@openzeppelin/contracts/security/Pausable.sol	2072248d2f79e661c149fd6a6593a8a3f038466557c9b75e50e0b001bcb5cf97
@openzeppelin/contracts/token/ERC20/ERC20.sol	5031430cc2613c32736d598037d3075985a2a09e61592a013dbd09a5bc2041b8
@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol	0344809a1044e11ece2401b4f7288f414ea41fa9d1dad24143c84b737c9fc02e
@openzeppelin/contracts/token/ERC20/extensions/ERC20Pausable.sol	ee84c2ac4bc96c21df80a3c39e38d508d658cbe386ec1ec7c04011d9925470f8
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	af5c8a77965cc82c33b7ff844deb9826166689e55dc037a7f2f790d057811990
@openzeppelin/contracts/token/ERC20/IERC20.sol	94f23e4af51a18c2269b355b8c7cf4db8003d075c9c541019eb8dcf4122864d5
@openzeppelin/contracts/utils/Context.sol	1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a
contracts/FullFeatureToken.sol	0f671f7ca5d21b4e63553674e59394f255b27db6c9e39e8341aa8d8a8b161e7e
contracts/lib/Helpers.sol	ebc24639f17289ce106f15e8d583562113fefe9036d00ae29f0259b6e459c58c

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	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	Unresolved
●	BT	Burns Tokens	Unresolved
●	BC	Blacklists Addresses	Unresolved

ST - Stops Transactions

Criticality	Medium
Location	contracts/FullFeatureToken.sol#L311,338,436
Status	Unresolved

Description

The contract owner has the authority to stop the transactions for all users excluding the owner. The owner may take advantage of it by setting the `maxTokenAmountPerAddress` to a minimum of 1. Additionally, the owner has the authority to pause all of the token's functionality, including transactions, by calling the `pause` function.

```
if (isMaxAmountOfTokensSet()) {
  if (balanceOf(to) + amount > maxTokenAmountPerAddress) {
    revert DestBalanceExceedsMaxAllowed(to);
  }
}
...
function pause() external onlyOwner {
  if (!isPausable()) {
    revert PausingNotEnabled();
  }
  _pause();
}
```

Recommendation

The contract could embody a check for not allowing setting the `maxTokenAmountPerAddress` less than a reasonable amount. A suggested implementation could check that the maximum amount should be more than a fixed percentage of the total supply. 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.

MT - Mints Tokens

Criticality	Critical
Location	contracts/FullFeatureToken.sol#L375
Status	Unresolved

Description

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

```
function mint(address to, uint256 amount) external onlyOwner whenNotPaused {
  if (!isMintable()) {
    revert MintingNotEnabled();
  }
  if (isMaxAmountOfTokensSet()) {
    if (balanceOf(to) + amount > maxTokenAmountPerAddress) {
      revert DestBalanceExceedsMaxAllowed(to);
    }
  }
  if (isBlacklistEnabled()) {
    if (_isBlacklisted[to]) {
      revert RecipientBlacklisted(to);
    }
  }
  if (isWhitelistEnabled()) {
    if (!whitelist[to]) {
      revert RecipientNotWhitelisted(to);
    }
  }

  super._mint(to, amount);
}
```

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.

BC - Blacklists Addresses

Criticality	Medium
Location	contracts/FullFeatureToken.sol#L264
Status	Unresolved

Description

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

```
function blackList(address addr) external onlyOwner whenNotPaused {
  Helpers.validateAddress(addr);
  if (!isBlacklistEnabled()) {
    revert BlacklistNotEnabled();
  }
  if (_isBlacklisted[addr]) {
    revert AddrAlreadyBlacklisted(addr);
  }
  if (isWhitelistEnabled() && whitelist[addr]) {
    revert CannotBlacklistWhitelistedAddr(addr);
  }

  _isBlacklisted[addr] = true;
  emit UserBlacklisted(addr);
}
```

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
●	CO	Code Optimization	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L14	Uninitialized Variables in Local Scope	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L17	Usage of Solidity Assembly	Unresolved
●	L18	Multiple Pragma Directives	Unresolved
●	L19	Stable Compiler Version	Unresolved

CO - Code Optimization

Criticality	Minor / Informative
Location	contracts/FullFeatureToken.sol#L129,319,350,379,389
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 contract uses nested `if-statements` on multiple occasions, where two conditions in total are being evaluated. It makes the code less readable and robust. These code segments can be altered to just one `if-statement` by using the `&&` operator. A sample of these code segments is listed below.

```
if (customConfigProps._isMaxAmountOfTokensSet) {
  if (maxTokenAmount == 0) {
    revert InvalidMaxTokenAmount(maxTokenAmount);
  }
}
...
if (isMaxAmountOfTokensSet()) {
  if (balanceOf(to) + amount > maxTokenAmountPerAddress) {
    revert DestBalanceExceedsMaxAllowed(to);
  }
}
...
```

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. A recommendation would be to combine the nested `if-statements` to one. A code sample can be the following:

```
if (customConfigProps._isMaxAmountOfTokensSet && maxTokenAmount == 0) {
  revert InvalidMaxTokenAmount(maxTokenAmount);
}
```

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	contracts/lib/Helpers.sol#L27
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.

```
function safeTransferETH(address to, uint256 amount) internal {
    bool success;
    // solhint-disable-next-line no-inline-assembly
    assembly {
      // Transfer the ETH and store if it succeeded or not.
      success := call(gas(), to, amount, 0, 0, 0, 0)
    }
    if (!success) {
      revert PaymentFailed(to, amount);
    }
  }
```

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.

L14 - Uninitialized Variables in Local Scope

Criticality	Minor / Informative
Location	contracts/FullFeatureToken.sol#L505,523
Status	Unresolved

Description

Using an uninitialized local variable can lead to unpredictable behavior and potentially cause errors in the contract. It's important to always initialize local variables with appropriate values before using them.

```
uint256 i
```

Recommendation

By initializing local variables before using them, the contract ensures that the functions behave as expected and avoid potential issues.

L17 - Usage of Solidity Assembly

Criticality	Minor / Informative
Location	contracts/lib/Helpers.sol#L30
Status	Unresolved

Description

Using assembly can be useful for optimizing code, but it can also be error-prone. It's important to carefully test and debug assembly code to ensure that it is correct and does not contain any errors.

Some common types of errors that can occur when using assembly in Solidity include Syntax, Type, Out-of-bounds, Stack, and Revert.

```
assembly {
      // Transfer the ETH and store if it succeeded or not.
      success := call(gas(), to, amount, 0, 0, 0, 0)
    }
```

Recommendation

It is recommended to use assembly sparingly and only when necessary, as it can be difficult to read and understand compared to Solidity code.

L18 - Multiple Pragma Directives

Criticality	Minor / Informative
Location	contracts/lib/Helpers.sol#L3 contracts/FullFeatureToken.sol#L3 @openzeppelin/contracts/utils/Context.sol#L4 @openzeppelin/contracts/token/ERC20/IERC20.sol#L4 @openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#L4 @openzeppelin/contracts/token/ERC20/extensions/ERC20Pausable.sol#L4 @openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol#L4 @openzeppelin/contracts/token/ERC20/ERC20.sol#L4 @openzeppelin/contracts/security/Pausable.sol#L4 @openzeppelin/contracts/access/Ownable.sol#L4
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 ^0.8.0;
pragma solidity 0.8.7;
```

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.

Contract Functions

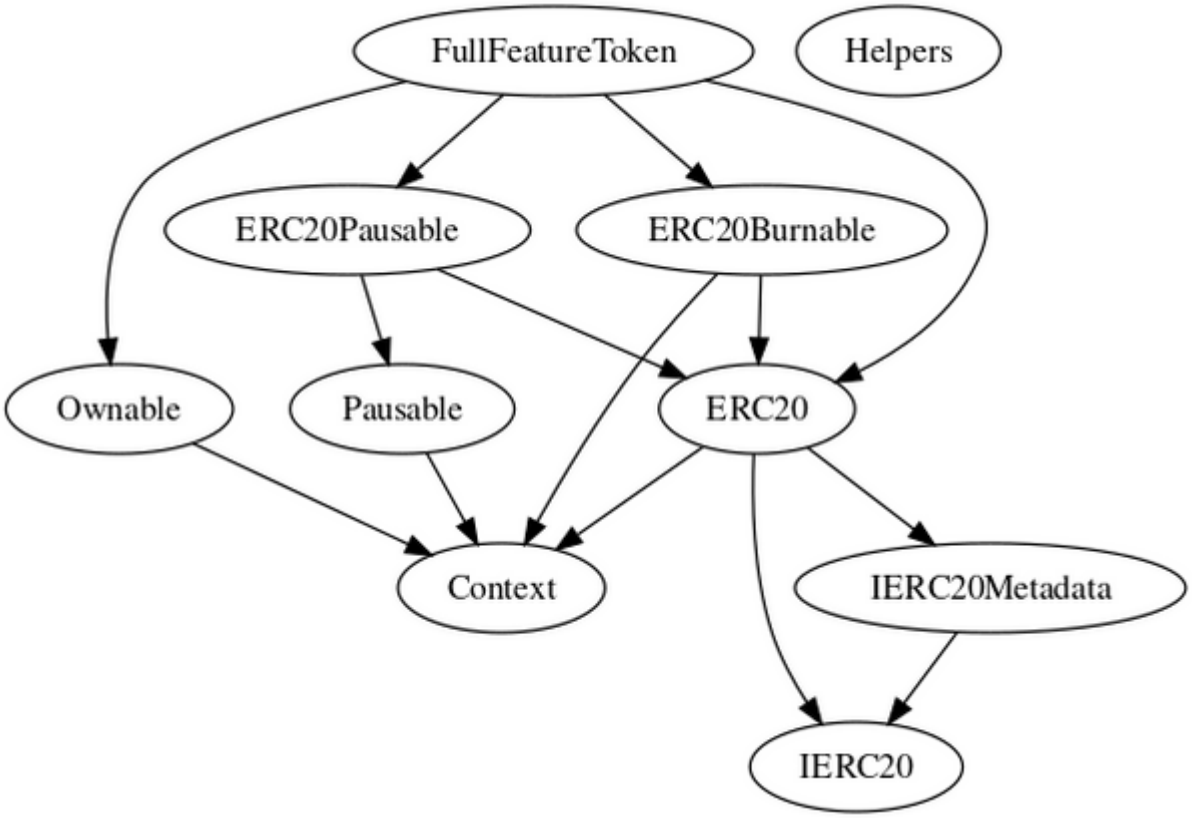
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
Pausable	Implementation	Context		
		Public	✓	-
	paused	Public		-
	_requireNotPaused	Internal		
	_requirePaused	Internal		
	_pause	Internal	✓	whenNotPaused
	_unpause	Internal	✓	whenPaused
ERC20	Implementation	Context, IERC20, IERC20Metadata		
		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	✓	-
ERC20Pausable	Implementation	ERC20, Pausable		
	_beforeTokenTransfer	Internal	✓	
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
IERC20	Interface			

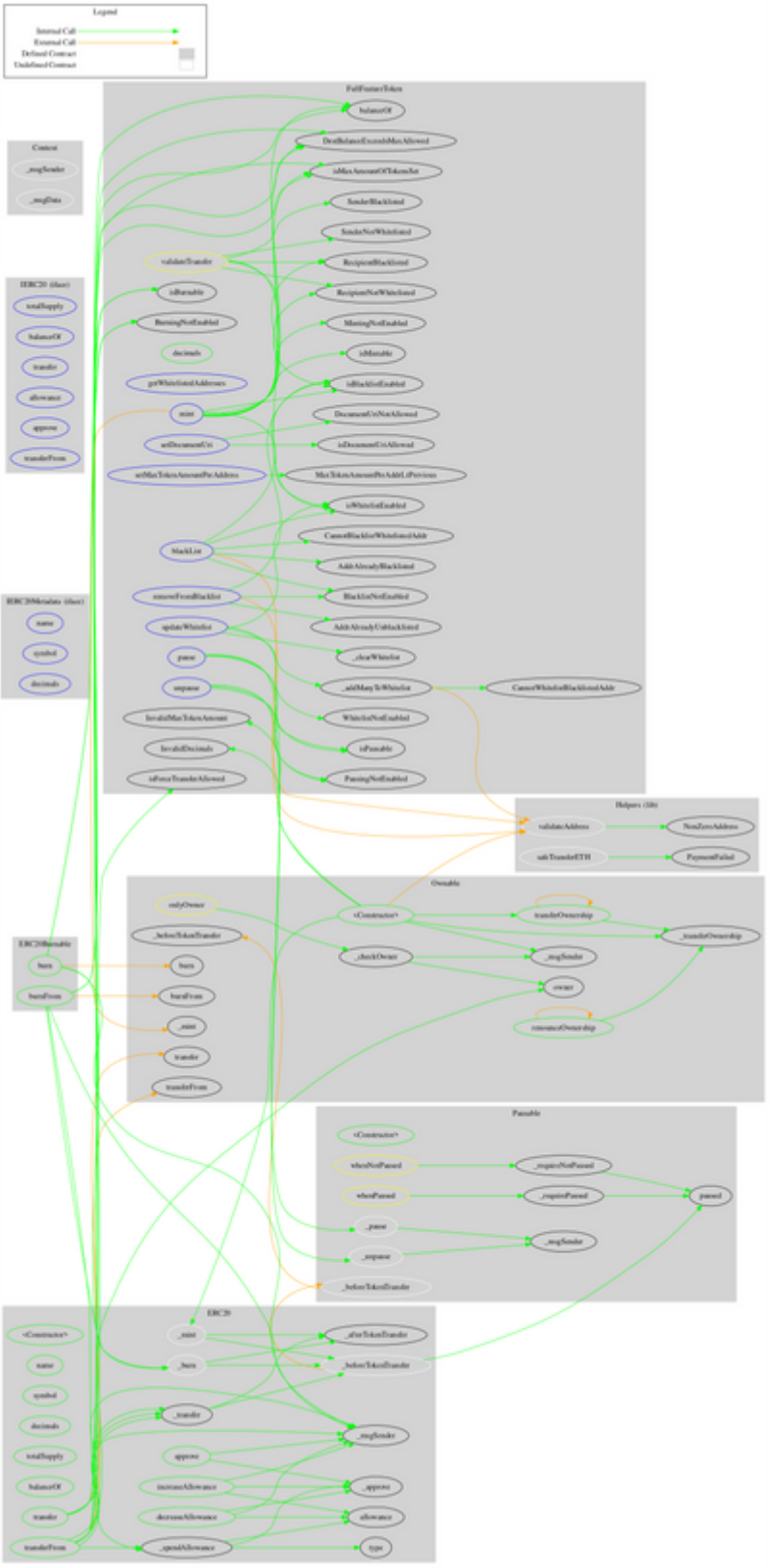
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
FullFeatureToken	Implementation	ERC20, ERC20Burnable, ERC20Pausable, Ownable		
		Public	✓	ERC20
	_beforeTokenTransfer	Internal	✓	
	isPausable	Public		-
	isMintable	Public		-
	isBurnable	Public		-
	isBlacklistEnabled	Public		-
	isWhitelistEnabled	Public		-
	isMaxAmountOfTokensSet	Public		-
	isDocumentUriAllowed	Public		-
	isForceTransferAllowed	Public		-
	decimals	Public		-
	getWhitelistedAddresses	External		-
	setDocumentUri	External	✓	onlyOwner
	setMaxTokenAmountPerAddress	External	✓	onlyOwner
	blackList	External	✓	onlyOwner whenNotPaus

				ed
	removeFromBlacklist	External	✓	onlyOwner whenNotPaused
	transfer	Public	✓	whenNotPaused validateTransfer
	transferFrom	Public	✓	whenNotPaused validateTransfer
	mint	External	✓	onlyOwner whenNotPaused
	burn	Public	✓	onlyOwner whenNotPaused
	burnFrom	Public	✓	onlyOwner whenNotPaused
	pause	External	✓	onlyOwner
	unpause	External	✓	onlyOwner
	renounceOwnership	Public	✓	onlyOwner whenNotPaused
	transferOwnership	Public	✓	onlyOwner whenNotPaused
	updateWhitelist	External	✓	onlyOwner
	_addManyToWhitelist	Private	✓	
	_clearWhitelist	Private	✓	
Helpers	Library			
	validateAddress	Internal		
	safeTransferETH	Internal	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
FullFeatureToken
Context
IERC20 (iface)
IERC20Metadata (iface)
IERC20Burnable
Helpers (lib)
Ownable
Pausable
ERC20

Summary

There are some functions that can be abused by the owner like stop transactions, mint tokens and blacklist addresses. if the contract owner abuses the mint functionality, then the contract will be highly inflated. 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