



Cyberscope

Audit Report

Exchange Traded Fund Token

June 2023

Network ETH

Address 0x5e7ca3A35144896F8782f7ceE69611ACEE94737A

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	Passed
●	MT	Mints Tokens	Passed
●	BT	Burns Tokens	Passed
●	BC	Blacklists Addresses	Passed

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	MDP	Misleading Decimal Property	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L02	State Variables could be Declared Constant	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L19	Stable Compiler Version	Unresolved

Table of Contents

Analysis	1
Diagnostics	2
Table of Contents	3
Review	4
Audit Updates	4
Source Files	5
Findings Breakdown	6
MDP - Misleading Decimal Property	7
Description	7
Recommendation	7
IDI - Immutable Declaration Improvement	8
Description	8
Recommendation	8
L02 - State Variables could be Declared Constant	9
Description	9
Recommendation	9
L04 - Conformance to Solidity Naming Conventions	10
Description	10
Recommendation	10
L19 - Stable Compiler Version	10
Description	11
Recommendation	11
Functions Analysis	12
Inheritance Graph	14
Flow Graph	15
Summary	16
Disclaimer	17
About Cyberscope	18

Review

Contract Name	ENMT
Compiler Version	v0.8.1+commit.df193b15
Optimization	200 runs
Explorer	https://etherscan.io/address/0x5e7ca3a35144896f8782f7cee69611acee94737a
Address	0x5e7ca3a35144896f8782f7cee69611acee94737a
Network	ETH
Symbol	\$ETH
Decimals	18
Total Supply	111.222.333.444.555

Audit Updates

Initial Audit	09 Jun 2023
---------------	-------------

Source Files

Filename	SHA256
Context.sol	8eb8a77f3ef90eb4391cfd3a1937a2198d60cba90155fd017c5a66a7b7fca938
ENMT.sol	26184d8387ff9ea1c5fda134079006e4dd7040aab76f8ef1f1a378b3e46ee945
ERC20.sol	3bf9d6a7f30b9e63099f5d52180e4ae2239c191d71ea7ae8ddc23c26feb9cfb3
IERC20.sol	7d2a6a7c516b1ee659cbba418661ea002e6436c81e1aa1f7f31cc9dbac505605

Findings Breakdown



● Critical	0
● Medium	0
● Minor / Informative	5

Severity	Unresolved	Acknowledged	Resolved	Other
● Critical	0	0	0	0
● Medium	0	0	0	0
● Minor / Informative	5	0	0	0

MDP - Misleading Decimal Property

Criticality	Minor / Informative
Location	ENMT.sol#L29
Status	Unresolved

Description

The contract is using a `_decimals` parameter in the constructor that does not influence the `decimals` attribute of the actual ERC20 token. In the standard ERC20 implementation provided by OpenZeppelin, the decimals attribute is preset to 18 and cannot be modified through the constructor.

```
constructor(string memory _name, string memory _symbol, uint8 _decimals,
address _creator, uint256 _totalSupply) ERC20(_name, _symbol) {
    _mint(msg.sender, _totalSupply);
    INFO = TokenInfo(_decimals, _creator);
}
```

Recommendation

The team is advised to remove the `_decimals` parameter from the constructor, if it's not serving a specific purpose in the contract.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	ENMT.sol#L31
Status	Unresolved

Description

The contract declares state variables that their value is initialized once in the constructor and are not modified afterwards. The `immutable` is a special declaration for this kind of state variables that saves gas when it is defined.

INFO

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.

L02 - State Variables could be Declared Constant

Criticality	Minor / Informative
Location	ENMT.sol#L20
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 TOKEN_TYPE = 1
```

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	ENMT.sol#L20,27
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.

```
uint256 public TOKEN_TYPE = 1
TokenInfo public INFO
```

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>.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	ENMT.sol#L12
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 ^0.8.0;
```

Recommendation

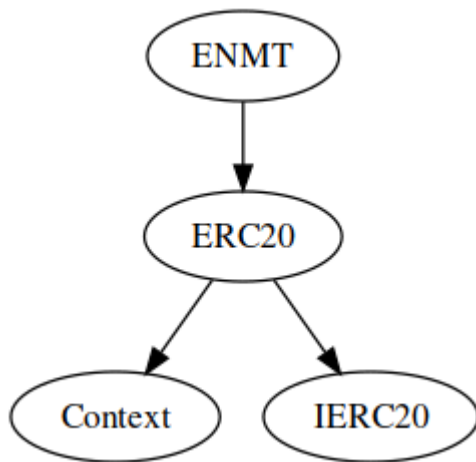
The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest 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

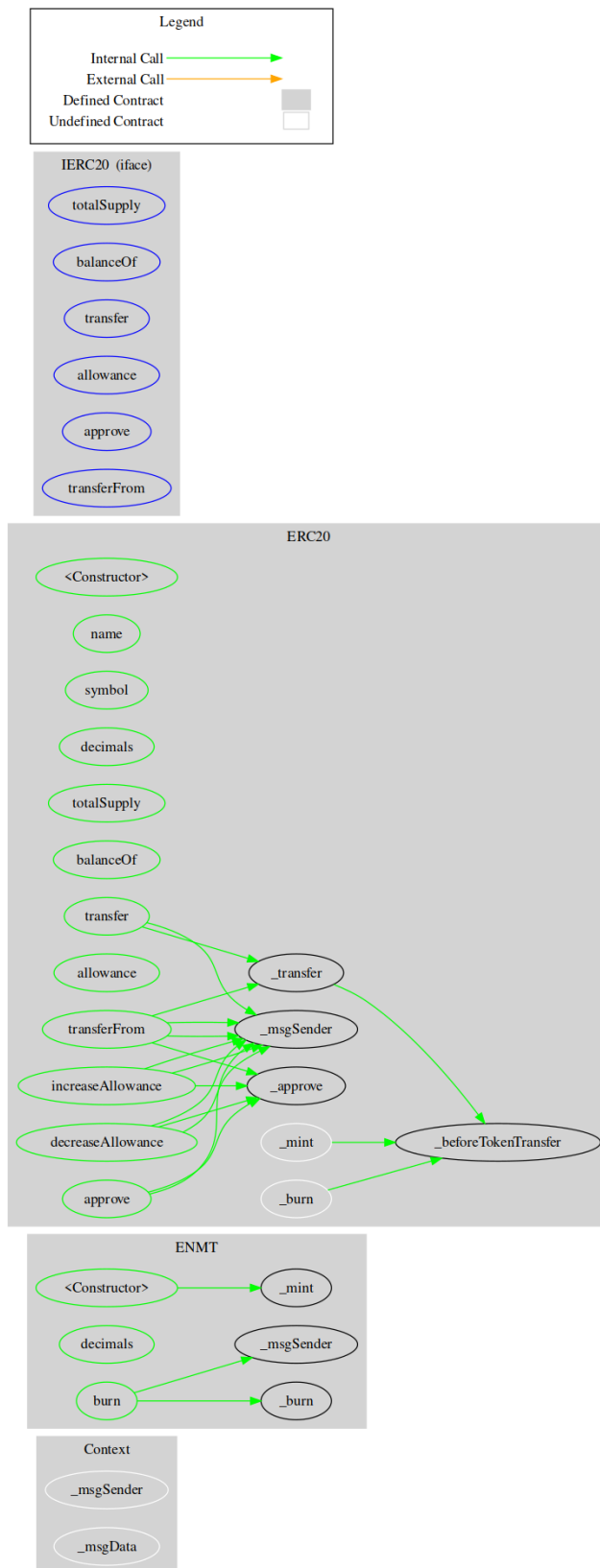
Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
ENMT	Implementation	ERC20		
		Public	✓	ERC20
	decimals	Public		-
	burn	Public	✓	-
ERC20	Implementation	Context, IERC20		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	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	✓	
	_beforeTokenTransfer	Internal	✓	
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-

Inheritance Graph



Flow Graph



Summary

\$ETH Exchange Traded Fund Token contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. \$ETH Exchange Traded Fund Token is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions.

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>