Versatile Finance

Smart Contract Security Audit

Eterna Token

05 November 2022

Summary

Project Name: Eterna Token

Contract Address: 0xE156ef68e42f33B74d23B497e37CE00a74873DBe

Client contact: Eterna Team

Blockchain: Binance smart chain

Language: Solidity

Project website: https://eterna.exchange/

Buy Tax: 0%

Sell Tax: 0%

Token supply: 1,000,000,000

Token ticker: EHX

Decimals: 18

Contract deployer address: 0x823df60E1acC3637ecC9C2b716cb5E8971EAa783

Contract's current owner address: 0x823df60e1acc3637ecc9c2b716cb5e8971eaa783

Background

Versatile Finance was commissioned by Eterna Team to perform an audit of the smart contract.

https://bscscan.com/address/0x823df60e1acc3637ecc9c2b716cb5e8971eaa783

The purpose of this audit was to achieve the following:

- Identify potential security issues with smart contracts
- Formally check the logic behind given smart contracts.

Information in this report should be used for understanding the risk exposure of smart contracts, and as a guide to improving the security posture of smart contracts by remediating the issues that were identified.

What is an audit

A smart contract audit is a comprehensive review process designed to discover logical errors, security vulnerabilities, and optimization opportunities within code. Versatile Finance manages this a step further by verifying economic logic to ensure the stability of smart contracts and highlighting privileged functionality to create a report that is easy to understand for developers and community members.

Techniques and Methods

- The code quality
- Use of best practices
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.
- Code risk issue analysis and recommendations
- Ownership privileges
- Code documentation and comments match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.

The following techniques, methods, and tools were used to review all the smart contracts.

Structural Analysis

We analyze the design patterns and structure of smart contracts. A thorough check is done to ensure the smart contract is structured in a way that will not have any issues.

Static Analysis

A static Analysis of Smart Contracts is done to identify contract vulnerabilities. In this step, a series of automated tools and manual testings are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual Analysis or review of code is done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts is completely manually analyzed line by line, and the logic is checked and compared with what's mentioned in the whitepaper to make sure everything's functioned as intended.

Gas Consumption

We check the behavior of smart contracts in production. Manual tests are done in DEXs to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Issue Categories

Every issue in this report has been assigned a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

1 High severity issue found

The Owner can mint new tokens

```
ftrace | funcSig
function mintTokens(address destination, uint256 amount)
    public
    onlyAuthorized
{
    _mint(destination, amount);
}
```

Medium-level severity issues

No Medium severity issues found

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they can still be fixed. This can put users' funds at risk and has a medium to high probability of exploitation.

Low-level severity issues

No low severity issues found

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future. These issues have a low probability of occurring or may have a minimal impact.

Informational

No Informational issues found

These are severity four issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Centralization

1 Centralization issue found

The Owner can mint new tokens

```
ftrace | funcSig
function mintTokens(address destination, uint256 amount)
    public
    onlyAuthorized
{
    _mint(destination, amount);
}
```

Contracts Description Table

Contract	Type	Bases		
└	Function Name	Visibility	Mutability	Modifiers
Context	Implementation			
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗

ERC20	Implementation	Context, IERC20		
L		Public ❗	🛑	NO❗
L	name	Public ❗		NO❗
L	symbol	Public ❗		NO❗
L	decimals	Public ❗		NO❗
L	totalSupply	Public ❗		NO❗
L	balanceOf	Public ❗		NO❗
L	transfer	Public ❗	🛑	NO❗
L	allowance	Public ❗		NO❗
L	approve	Public ❗	🛑	NO❗
L	transferFrom	Public ❗	🛑	NO❗
L	increaseAllowance	Public ❗	🛑	NO❗
L	decreaseAllowance	Public ❗	🛑	NO❗
L	_transfer	Internal 🔒	🛑	
L	_mint	Internal 🔒	🛑	
L	_burn	Internal 🔒	🛑	
L	_approve	Internal 🔒	🛑	
Ownable	Implementation	Context		

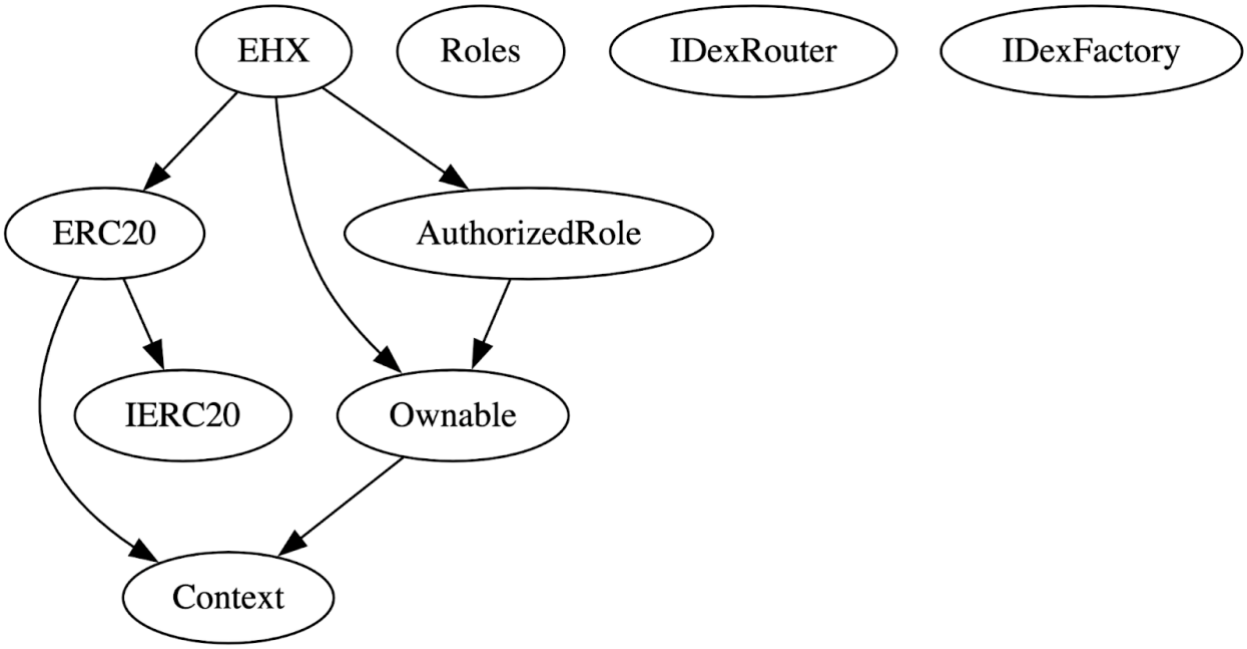
L		Public ❗	🛑	NO❗
L	owner	Public ❗		NO❗
L	renounceOwnership	External ❗	🛑	onlyOwner
L	transferOwnership	Public ❗	🛑	onlyOwner
Roles	Library			
L	add	Internal 🔒	🛑	
L	remove	Internal 🔒	🛑	
L	has	Internal 🔒		
AuthorizedRole	Implementation	Ownable		
L	isAuthorized	Public ❗		NO❗
L	addAuthorized	Public ❗	🛑	onlyOwner
L	removeAuthorized	Public ❗	🛑	onlyOwner
L	renounceAuthorized	Public ❗	🛑	NO❗
L	_addAuthorized	Internal 🔒	🛑	
L	_removeAuthorized	Internal 🔒	🛑	
IDexRouter	Interface			
L	factory	External ❗		NO❗

└	WETH	External ❗		NO❗
IDexFactory	Interface			
└	createPair	External ❗	🛑	NO❗
EHX	Implementation	ERC20, Ownable, AuthorizedRole		
└		Public ❗	🛑	ERC20
└	enableTrading	External ❗	🛑	onlyOwner
└	manageRestrictedWallets	External ❗	🛑	onlyOwner
└	removeLimits	External ❗	🛑	onlyOwner
└	setRobinHoodActive	External ❗	🛑	onlyOwner
└	setRobinHoodPercent	External ❗	🛑	onlyOwner
└	setRobinHoodAddress	External ❗	🛑	onlyOwner
└	updateMaxTransaction	External ❗	🛑	onlyOwner
└	transferForeignToken	External ❗	🛑	onlyOwner
└	setAutomatedMarketMakerPair	Public ❗	🛑	onlyOwner
└	setWhitelistedAddress	Public ❗	🛑	onlyOwner
└	mintTokens	Public ❗	🛑	onlyAuthorized

L	burnTokens	Public ❗	🛑	onlyAuthorized
L	_transfer	Internal 🔒	🛑	

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable



Owner privileges

The owner can enable trading, once enabled can not disable again

```
ftrace | funcSig
function enableTrading() external onlyOwner {
    require(!tradingActive, "Trading is already active, cannot relaunch.");
    tradingActive = true;
    tradingActiveBlock = block.number;
    emit EnabledTrading();
}
```

The owner can remove transactions limit, once removed can not enable again

```
ftrace | funcSig
function removeLimits() external onlyOwner {
    limitsInEffect = false;
    maxTransaction = totalSupply();
    emit RemovedLimits();
}
```

The owner can active/deactivate Robinhood method (robin hood method is an anti-bot mechanism it will take some extra percentage of tax if someone buys and sell in the same block)

```
ftrace | funcSig
function setRobinHoodActive(bool active) external onlyOwner {
    robinHoodActive = active;
}
```

The owner can change robin hood tax percentage

```
ftrace | funcSig
function setRobinHoodPercent(uint256 perc) external onlyOwner {
    require(perc <= 10000, "too high");
    robinHoodPercent = perc;
}
```

The owner can change robin hood tax receiver address

```
ftrace | funcSig
function setRobinHoodAddress(address wallet) external onlyOwner {
    require(wallet != address(0), "zero address");
    robinHoodWallet = wallet;
}
```

The owner can change max transaction amount minimum up to 0.1%

```
ftrace | funcSig
function updateMaxTransaction(uint256 newNum) external onlyOwner {
    require(
        newNum >= ((totalSupply() * 1) / 1000) / (10**decimals()),
        "Cannot set max buy amount lower than 0.1%"
    );
    maxTransaction = newNum * (10**decimals());
    emit UpdatedMaxTransaction(maxTransaction);
}
```

The owner can take any BEP20 tokens from the contract

```
ftrace | funcSig
function transferForeignToken(address _token, address _to)
    external
    onlyOwner
    returns (bool _sent)
{
    require(_token != address(0), "_token address cannot be 0");
    uint256 _contractBalance = IERC20(_token).balanceOf(address(this));
    _sent = IERC20(_token).transfer(_to, _contractBalance);
    emit TransferForeignToken(_token, _contractBalance);
}
```

The owner can add any extra LP address

```
ftrace | funcSig
function setAutomatedMarketMakerPair(address pair, bool value)
    public
    onlyOwner
{
    require(
        pair != lpPair || value,
        "The pair cannot be removed from automatedMarketMakerPairs"
    );
    automatedMarketMakerPairs[pair] = value;
    emit SetAutomatedMarketMakerPair(pair, value);
}
```

The owner can whitelist and remove addresses

```
ftrace | funcSig
function setWhitelistedAddress(address account, bool excluded)
    public
    onlyOwner
{
    _isWhitelisted[account] = excluded;
    emit Whitelisted(account, excluded);
}
```

The owner can mint any new tokens

```
ftrace | funcSig
function mintTokens(address destination, uint256 amount)
    public
    onlyAuthorized
{
    _mint(destination, amount);
}
```

The owner can burn tokens from his wallet

```
ftrace | funcSig
function burnTokens(uint256 amount) public onlyAuthorized {
    _burn(msg.sender, amount);
}
```

Audit Results

Vulnerability Category	Status
Arbitrary Jump/Storage Write	pass
BRC20 Token standards	pass
Compiler errors	pass
Latest compiler version	pass
Authorization of function call to untrusted contract	pass
Dependence on Predictable Variables	pass
Ether/Token Theft	pass
Gas consumption	pass
Safemath features	pass
Fallback usage	pass
Deprecated items	pass
Redundant code	pass
Overriding variables	pass
Flash Loans	pass
Front Running	pass
Improper Events	pass
Improper Authorization Scheme	pass
Integer Over/Underflow	pass
Business logic issues	pass

Oracle issues	pass
Race Conditions	pass
Reentrancy	pass
Signature Issues	pass
Unbounded Loops	pass
Unused Code	pass
Pseudo-random number generator (PRNG)	pass
Fake deposit	pass
Centralisation	**High severity issues**

Audit conclusion

Versatile Finance team has performed in-depth testings, line by line manual code review, and automated audit of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, manipulations, and hacks. According to the smart contract audit.

Smart contract functional Status: **PASS**

Number of risk issues: **1**

Solidity code functional issue level: **PASS**

Number of owner privileges: **11**

Centralization risk correlated to the active owner: **HIGH**

Smart contract active ownership: **ACTIVE**

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as of the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Versatile Finance and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers, and other representatives) (Versatile Finance) owe no duty of care towards you or any other person, nor does Versatile Finance make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties, or other terms of any kind except as set out in this disclaimer, and Versatile Finance hereby excludes all representations, warranties, conditions, and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Versatile Finance hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Versatile Finance, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of the use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.