Cyberscope

Audit Report

Eternex Network

March 2025

Files Token.sol, ITRC20.sol, SafeMath.sol, TRC20.sol, TRC20Detailed.sol

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
●	MEM	Missing Error Messages	Unresolved
●	OPSV	Outdated Pragma Solidity Version	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved
●	L19	Stable Compiler Version	Unresolved

Table of Contents

Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation**: This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation**: This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical**: Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium**: Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor**: Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative**: Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
Critical	Highly Likely / High Impact
Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
Minor / Informative	Unlikely / Low to no Impact

Review

Audit Updates

Initial Audit	27 Mar 2025

Source Files

Filename	SHA256
Token.sol	14c6f49829f9bc319e83ead25ecf8dca4e4eefacff7d6dcd37c10d4ba747b76e
TRC20Detailed.sol	991e462be167b8748627575a741f05d80f446f7f371009820e4f4286fe5358f3
TRC20.sol	be9028747914e124d12e61ff6bab2aaecd5a811defeca3cf8eec990d11538feb
SafeMath.sol	f96a3c3250569024dc0625232dc37496162ac97ca8324c024f64a6300d6201a7
ITRC20.sol	ff25c1817e6c665807735d7ac705417fa4eb6eaf2095606377b635e8f52a9b6d

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

MEM - Missing Error Messages

Criticality	Minor / Informative
Location	TRC20.sol#L99,135,157,193,194,225
Status	Unresolved

Description

The contract does not implement error messages to accurately reflect the problem, making it difficult to identify and fix the issue. As a result, the users will not be able to find the root cause of the error.

Specifically, during the approval of new allowances and when updating balances, error messages are missing to let users know that their current allowances or balances are sufficient.

```
_approve(sender, msg.sender,
_allowances[sender][msg.sender].sub(amount));
_approve(msg.sender, spender,
_allowances[msg.sender][spender].sub(subtractedValue));
_balances[sender] = _balances[sender].sub(amount);
_totalSupply = _totalSupply.sub(value);
_balances[account] = _balances[account].sub(value);
_approve(account, msg.sender,
_allowances[account][msg.sender].sub(amount));
```

Recommendation

The team is suggested to provide a descriptive message to the errors. This message can be used to provide additional context about the error that occurred or to explain why the contract execution was halted. This can be useful for debugging and for providing more information to users that interact with the contract.

OPSV - Outdated Pragma Solidity Version

Criticality	Minor / Informative
Location	Token.sol#L1 TRC20Detailed.sol#L1 TRC20.sol#L1 SafeMath.sol#L1 ITRC20.sol#L1
Status	Unresolved

Description

The contract is using an outdated version of Solidity. Outdated versions of Solidity lack optimizations, and improvements found in more recent versions, which could affect the performance and efficiency of the contract.

```
pragma solidity ^0.5.0;
```

Recommendation

The team should consider using a more recent version of Solidity for the reasons mentioned above.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	TRC20.sol#L190,225
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 _burn(address account, uint256 value) internal {
        require(account != address(0), "TRC20: burn from the
zero address");
        _totalSupply = _totalSupply.sub(value);
        _balances[account] = _balances[account].sub(value);
        emit Transfer(account, address(0), value);
}
...
function _burnFrom(address account, uint256 amount) internal {
        _burn(account, amount);
        _approve(account, msg.sender,
_allowances[account][msg.sender].sub(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.

L15 - Local Scope Variable Shadowing

Criticality	Minor / Informative
Location	TRC20Detailed.sol#L18
Status	Unresolved

Description

Local scope variable shadowing occurs when a local variable with the same name as a variable in an outer scope is declared within a function or code block. When this happens, the local variable "shadows" the outer variable, meaning that it takes precedence over the outer variable within the scope in which it is declared.

```
uint8 decimals
string memory symbol
string memory name
```

Recommendation

It's important to be aware of shadowing when working with local variables, as it can lead to confusion and unintended consequences if not used correctly. It's generally a good idea to choose unique names for local variables to avoid shadowing outer variables and causing confusion.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	TRC20Detailed.sol#L1 SafeMath.sol#L1 ITRC20.sol#L1 Token.sol#L1 TRC20.sol#L1
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 ^0.5.0;
```

Recommendation

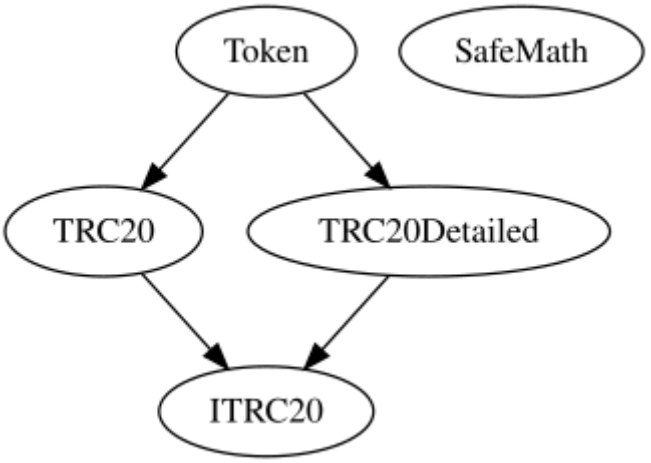
The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest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

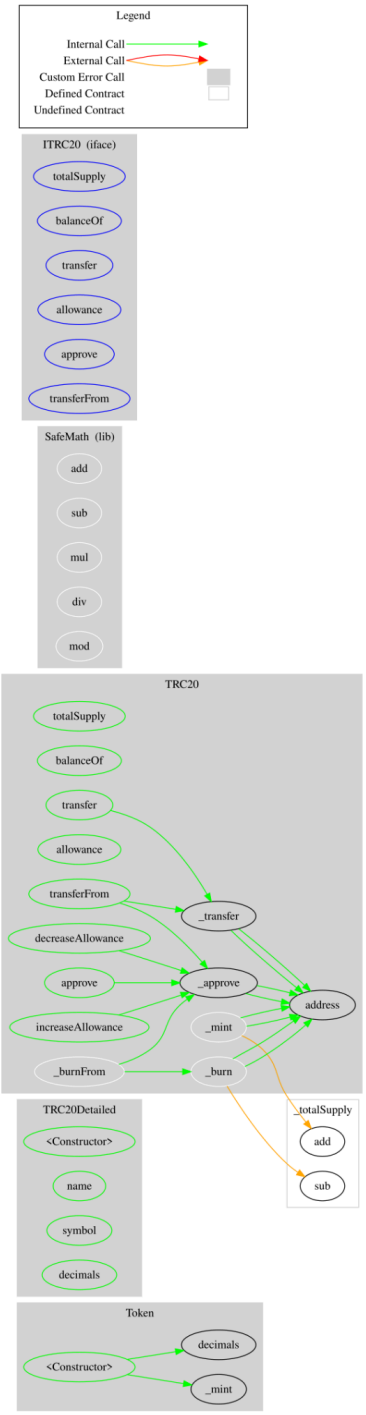
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Token	Implementation	TRC20, TRC20Detailed		
		Public	✓	TRC20Detailed
TRC20Detailed	Implementation	ITRC20		
		Public	✓	-
	name	Public		-
	symbol	Public		-
	decimals	Public		-
TRC20	Implementation	ITRC20		
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Internal	✓	

	_burn	Internal	✓	
	_approve	Internal	✓	
	_burnFrom	Internal	✓	
SafeMath	Library			
	add	Internal		
	sub	Internal		
	mul	Internal		
	div	Internal		
	mod	Internal		
ITRC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Custom Error Call
Defined Contract
Undefined Contract
ITRC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
SafeMath (lib)
add
sub
mul
div
mod
TRC20
totalSupply
balanceOf
transfer
allowance
transferFrom
decreaseAllowance
approve
increaseAllowance
_burnFrom
_transfer
_approve
_mint
_burn
address
_totalSupply
add
sub
TRC20Detailed
<Constructor>
name
symbol
decimals
Token
<Constructor>
decimals
_mint

Summary

Eternex Network contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. Eternex Network is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues.

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

cyberscope.io