Cyberscope

Audit Report

Blabberix

October 2024

Network	AVAX
Address	0xabc72Ea2cF6d52D438619C653Ff9AE6c8ea07AFa
Audited by	© cyberscope

Analysis

🔴 Critical 🟠 Medium ⚪ Minor / Informative 🔵 Pass

Severity	Code	Description	Status
⚪	ST	Stops Transactions	Acknowledged
🔵	OTUT	Transfers User's Tokens	Passed
🔵	ELFM	Exceeds Fees Limit	Passed
⚪	MT	Mints Tokens	Acknowledged
🔵	BT	Burns Tokens	Passed
⚪	BC	Blacklists Addresses	Acknowledged

Diagnostics

🔴 Critical 🟠 Medium ● Minor / Informative

Severity	Code	Description	Status
●	IRA	Inconsistent Role Assignment	Unresolved
●	IDI	Immutable Declaration Improvement	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L17	Usage of Solidity Assembly	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

Contract Name	Token
Compiler Version	v0.8.24+commit.e11b9ed9
Optimization	100000 runs
Explorer	https://snowtrace.io/address/0xabc72ea2cf6d52d438619c653ff9ae6c8ea07afa
Address	0xabc72ea2cf6d52d438619c653ff9ae6c8ea07afa
Network	AVAX
Symbol	BBIX
Decimals	18
Total Supply	500,000,000

Audit Updates

Initial Audit	11 Oct 2024

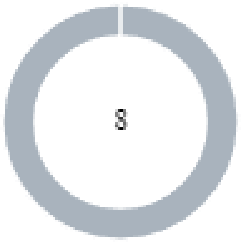
Source Files

Filename	SHA256
contracts/Token.sol	eaf1f4b5f9dfc9cca3688b11bb9f80e2dd046a41455b1a095e286403101b0c24
@openzeppelin/contracts/utils/Pausable.sol	6543160582b3c0319a180f31660faf6ba0a8444acbdb03357c09790a96256835
@openzeppelin/contracts/utils/Context.sol	847fda5460fee70f56f4200f59b82ae622bb03c79c77e67af010e31b7e2cc5b6

@openzeppelin/contracts/utils/math/SafeCast.sol	089488198de38548e4e8ee7940d307f18396e5b295de5bca7ff7567fd4142cc3
@openzeppelin/contracts/utils/math/Math.sol	a6ee779fc42e6bf01b5e6a963065706e882b016affbedfd8be19a71ea48e6e15
@openzeppelin/contracts/utils/introspection/IERC165.sol	07ae1ac964ab74dedada999e2dfc642031a6495469cffc0bf715daa4f1e4f904
@openzeppelin/contracts/utils/introspection/ERC165.sol	99348354365cbdeb90157e2903334b861a00d69faab7720ae542d911d5c70d87
@openzeppelin/contracts/token/ERC20/IERC20.sol	6f2faae462e286e24e091d7718575179644dc60e79936ef0c92e2d1ab3ca3cee
@openzeppelin/contracts/token/ERC20/ERC20.sol	2d874da1c1478ed22a2d30dcf1a6ec0d09a13f897ca680d55fb49fbcc0e0c5b1
@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol	1d079c20a192a135308e99fa5515c27acfbb071e6cdb0913b13634e630865939
@openzeppelin/contracts/token/ERC20/extensions/ERC20Pausable.sol	674e0f108dee059d5def448d592038d4dd8e1bbb6bb43730c651774c241cb19e
@openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol	2e6108a11184dd0caab3f3ef31bd15fed1bc7e4c781a55bc867ccedd8474565c
@openzeppelin/contracts/interfaces/draft-IERC6093.sol	4aea87243e6de38804bf8737bf86f750443d3b5e63dd0fd0b7ad92f77cdbd3e3
@openzeppelin/contracts/interfaces/IERC5313.sol	8bc20fb9e53bdb556386519448975a15afa3794230a90b6e3050fe00cbb48d50
@openzeppelin/contracts/access/IAccessControl.sol	1d6ef09193265172824fa1139e85cba422117ca918961183f080e692489d8c3b
@openzeppelin/contracts/access/AccessControl.sol	1086a1ad3788972b885ff3f209da510615dde6214d46b29e1cd2a4924f66c06d
@openzeppelin/contracts/access/extensions/IAccessControlDefaultAdminRules.sol	40a5073d17cd7439dd882db0d5ea64527e9c8be982109f4fbbc3fafc1a4973c7

@openzeppelin/contracts/access/extensions/AccessControlDefaultAdminRules.sol	fefdd334836d2fd96871284d4cc3877d95f081bbc54fc179a8572a5f2d50f7ad

Findings Breakdown



●	Critical	0
●	Medium	0
●	Minor / Informative	8

Severity		Unresolved	Acknowledged	Resolved	Other
●	Critical	0	0	0	0
●	Medium	0	0	0	0
●	Minor / Informative	5	3	0	0

ST - Stops Transactions

Criticality	Minor / Informative
Location	contracts/Token.sol#L143,334
Status	Acknowledged

Description

The `PAUSER_ROLE` has the authority to stop the transfers for for all. They may take advantage of it by calling the `pause` function.

```
function pause() public {
        // Check that the caller has the required role
        require(
            hasRole(PAUSER_ROLE, msg.sender),
            "Token: caller does not have Pauser role"
        );

        // Pause the contract
        _pause();
    }
```

Additionally, the transfers to wallets can be stopped if the `_W2WEnabled` is set to false and to contracts if `_W2CEnabled` is set to false, excluding the whitelisted addresses.

```
if (!_isWhitelisted[from] && !_isWhitelisted[to]) {
            // If the recipient is not a contract
            if (!isContract(to)) {
                // Then wallet-to-wallet transfers must be
enabled
                require(_W2WEnabled, "Token: W2W Transfer
prohibited");
            } else {
                // Else, if the recipient is a contract,
wallet-to-contract transfers must be enabled
                require(_W2CEnabled, "Token: W2C Transfer
prohibited");
            }
        }
```

Recommendation

The team should carefully manage the private keys of the owner's account and the administrative roles. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

MT - Mints Tokens

Criticality	Minor / Informative
Location	contracts/Token.sol#L129
Status	Acknowledged

Description

The `MINTER_ROLE` has the authority to mint tokens. They 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) public {
        // Check that the caller has the required role
        require(
            hasRole(MINTER_ROLE, msg.sender),
            "Token: caller does not have Minter role"
        );

        // Mint new tokens
        _mint(to, amount);
    }
```

Recommendation

The team should carefully manage the private keys of the owner’s account and the administrative roles. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

BC - Blacklists Addresses

Criticality	Minor / Informative
Location	contracts/Token.sol#L226,327
Status	Acknowledged

Description

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

```
function addToBlackList(address _user) public {
        // Check that the caller has the required role
        require(
            hasRole(BLACKLIST_ADDER_ROLE, msg.sender),
            "Token: caller does not have Blacklist adder role"
        );

        _isBlacklisted[_user] = true;
        emit AddedToBlackList(_user);
    }

require(
            !_isBlacklisted[from] && !_isBlacklisted[to],
            "Token: Address is blacklisted. Transfer
prohibited"
        );
```

Recommendation

The team should carefully manage the private keys of the owner’s account and the administrative roles. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

IRA - Inconsistent Role Assignment

Criticality	Minor / Informative
Location	contracts/Token.sol#L69
Status	Unresolved

Description

The constructor of the contract takes an array of manager addresses to assign specific roles such as minter, pauser, unpauser, and managers for whitelist and blacklist. The comments in the constructor specify that the roles at indexes 5 and 6 are for the whitelist managers, and roles at indexes 7 and 8 are for the blacklist managers. However, in the subsequent role assignment logic, the roles are assigned in reverse order, with the blacklist roles being granted to indexes 5 and 6 and the whitelist roles being granted to indexes 7 and 8. This inconsistency between the comments and the actual role assignment can lead to confusion for developers and auditors reviewing the contract and might result in improper configuration of role-based access control.

```
constructor(
        string memory name,         // Token name
        string memory symbol,       // Token symbol (ticker)
        uint256 totalSupply,        // Initial supply
        uint8 decimalPlaces,        // Number of decimal places
        address defaultAdmin,       // Default admin address
        address[9] memory manager   // Manager addresses
        // manager[0] - Token minter address
        // manager[1] - Pauser address
        // manager[2] - Unpauser address
        // manager[3] - Wallet-to-wallet transfer manager
address
        // manager[4] - Wallet-to-contract transfer manager
address
        // manager[5] - Whitelist adder address
        // manager[6] - Whitelist remover address
        // manager[7] - Blacklist adder address
        // manager[8] - Blacklist remover address
    ) ERC20(name, symbol) {
        // Assign admin for roles
        _setRoleAdmin(MINTER_ROLE, ADMIN_ROLE);
        _setRoleAdmin(PAUSER_ROLE, ADMIN_ROLE);
        _setRoleAdmin(UNPAUSER_ROLE, ADMIN_ROLE);
        _setRoleAdmin(W2W_MANAGER_ROLE, ADMIN_ROLE);
        _setRoleAdmin(W2C_MANAGER_ROLE, ADMIN_ROLE);
        _setRoleAdmin(BLACKLIST_ADDER_ROLE, ADMIN_ROLE);
        _setRoleAdmin(BLACKLIST_REMOVER_ROLE, ADMIN_ROLE);
        _setRoleAdmin(WHITELIST_ADDER_ROLE, ADMIN_ROLE);
        _setRoleAdmin(WHITELIST_REMOVER_ROLE, ADMIN_ROLE);

        // Grant roles
        _grantRole(ADMIN_ROLE, defaultAdmin);
        _grantRole(MINTER_ROLE, manager[0]);
        _grantRole(PAUSER_ROLE, manager[1]);
        _grantRole(UNPAUSER_ROLE, manager[2]);
        _grantRole(W2W_MANAGER_ROLE, manager[3]);
        _grantRole(W2C_MANAGER_ROLE, manager[4]);
        _grantRole(BLACKLIST_ADDER_ROLE, manager[5]);
        _grantRole(BLACKLIST_REMOVER_ROLE, manager[6]);
        _grantRole(WHITELIST_ADDER_ROLE, manager[7]);
        _grantRole(WHITELIST_REMOVER_ROLE, manager[8]);
```

Recommendation

The comments and role assignments should be reviewed for consistency. Either the comments should be updated to reflect the actual order in which roles are assigned, or the logic for granting roles should be adjusted to align with the intended role ordering as indicated in the comments. This will ensure clarity and prevent potential misconfigurations of roles.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	contracts/Token.sol#L110
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.

```
_decimals
```

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.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/Token.sol#L58,64,226,240,254,274,288,302,309
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.

```
bool private _W2WEnabled
bool private _W2CEnabled
address _user
address _addr
```

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/stable/style-guide.html#naming-conventions.

L17 - Usage of Solidity Assembly

Criticality	Minor / Informative
Location	contracts/Token.sol#L311
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 {
                size := extcodesize(_addr)
          }
```

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.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/Token.sol#L2
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 ^0.8.24;
```

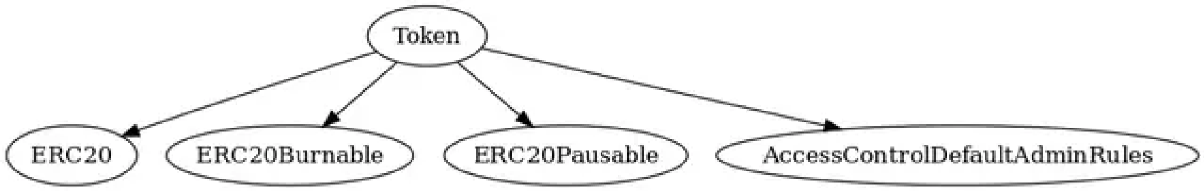
Recommendation

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.

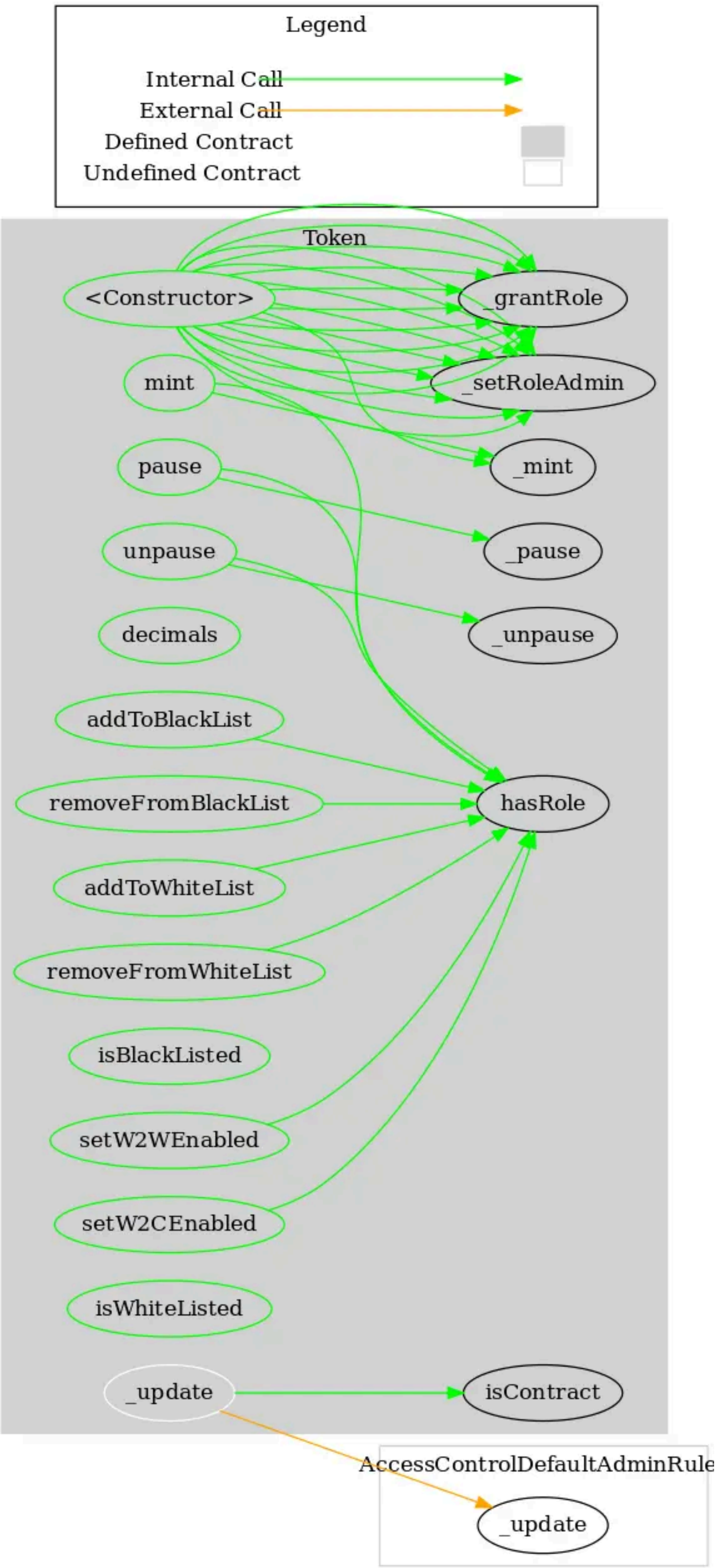
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Token	Implementation	ERC20, ERC20Burnable, ERC20Pausable, AccessControlDefaultAdminRules		
		Public	✓	ERC20
	mint	Public	✓	-
	pause	Public	✓	-
	unpause	Public	✓	-
	decimals	Public		-
	setW2WEnabled	Public	✓	-
	setW2CEnabled	Public	✓	-
	addToBlackList	Public	✓	-
	removeFromBlackList	Public	✓	-
	isBlackListed	Public		-
	addToWhiteList	Public	✓	-
	removeFromWhiteList	Public	✓	-
	isWhiteListed	Public		-
	isContract	Private		
	_update	Internal	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Token
<Constructor>
_grantRole
mint
_setRoleAdmin
pause
_mint
unpause
_pause
decimals
_unpause
addToBlackList
removeFromBlackList
hasRole
addToWhiteList
removeFromWhiteList
isBlackListed
setW2WEnabled
setW2CEnabled
isWhiteListed
_update
isContract
AccessControlDefaultAdminRule
_update

Summary

Blabberix contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the owner like stop transactions, mint tokens and massively 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

cyberscope.io