Cyberscope

Audit Report

MyVolt Token

March 2024

Repository https://github.com/MyVoltEnergy/MyVolt-Solidity-

Commit 6877dc90f86b66d98bd764a64a9edccdc7d93823

Audited by © cyberscope

Table of Contents

Review

Contract Name	MyVoltToken
Repository	https://github.com/MyVoltEnergy/MyVolt-Solidity-
Commit	6877dc90f86b66d98bd764a64a9edccdc7d93823
Testing Deploy	https://testnet.bscscan.com/address/0xb830226bA584Ab395F41B38918e266FCb6E067eB
Symbol	MVOLT
Decimals	18
Total Supply	1,000,000,000

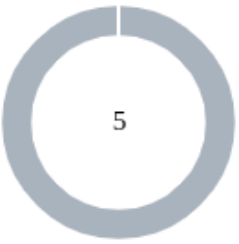
Audit Updates

Initial Audit	15 Mar 2024 https://github.com/cyberscope-io/audits/blob/main/mvolt/v1/audit.pdf
Corrected Phase 2	27 Mar 2024

Source Files

Filename	SHA256
contracts/MyVoltToken.sol	15a0107ed29f1bf019c9b4a179c6fa2b82c00729c16ed58973ed48b496902f9a

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

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
●	IMP	Inefficient Minting Process	Unresolved
●	ST	Stops Transactions	Unresolved
●	TUU	Time Units Usage	Unresolved
●	UFR	Unutilized Function Redundancy	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved

IMP - Inefficient Minting Process

Criticality	Minor / Informative
Location	contracts/MyVoltToken.sol#L418
Status	Unresolved

Description

The contract is designed to mint tokens to the same `vestingContractAddress` multiple times across different categories such as Seed Sale, Public Sale, Team, Treasury, Marketing, Advisors, and Liquidity Pool. This approach involves calling the `_mint` function separately for each category, despite all tokens being allocated to the same address. This method of individually minting tokens for different purposes, while using the same destination address, introduces unnecessary complexity and increases the transaction cost due to the repeated execution of the minting function. Moreover, this minting strategy does not leverage the potential for simplification and efficiency that could be achieved by consolidating these operations into a single minting action, especially since the cumulative total supply to be minted to the vestingContractAddress is predetermined.

```
    function _distributeTokens(address vestingContractAddress) private {
        //uint256 theToken = 1e18;

        // Seed Sale
        _mint(vestingContractAddress, 30000000 * 10 ** 18);

        // Public Sale
        _mint(vestingContractAddress, 90000000 * 10 ** 18);

        // Team
        _mint(vestingContractAddress, 100000000 * 10 ** 18);

        // Treasury
        _mint(vestingContractAddress, 280000000 * 10 ** 18);

        // Marketing
        _mint(vestingContractAddress, 85000000 * 10 ** 18);

        // Advisors
        _mint(vestingContractAddress, 55000000 * 10 ** 18);

        // Liquidity Pool
        _mint(vestingContractAddress, 105000000 * 10 ** 18);
    }
```

Recommendation

It is recommended to consolidate the multiple minting operations into a single minting action by calculating the total amount of tokens to be allocated to the vestingContractAddress across all categories beforehand. This can be achieved by summing up the individual amounts for Seed Sale, Public Sale, Team, Treasury, Marketing, Advisors, and Liquidity Pool, and then performing a single _mint operation with this total amount. This streamlined approach reduces the number of transactions required, thereby minimizing gas costs and simplifying the contract's logic. Additionally, consolidating the minting process enhances the contract's clarity and maintainability, making it easier for developers and auditors to understand the token distribution mechanism. This recommendation aims to optimize the contract's efficiency and cost-effectiveness while maintaining its intended functionality.

ST - Stops Transactions

Criticality	Minor / Informative
Location	contracts/MyVoltToken.sol#L373
Status	Unresolved

Description

The contract owner has the authority to stop the transaction for all users. The owner may take advantage of it by calling the `executePause` function.

```
    function executePause() external onlyOwner
executable(keccak256("pause")) {
        _pause();
    }
```

Recommendation

The contract could embody a check for not allowing setting the `_maxTxAmount` less than a reasonable amount. A suggested implementation could check that the minimum 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.

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.

TUU - Time Units Usage

Criticality	Minor / Informative
Location	contracts/MyVoltToken.sol#L325
Status	Unresolved

Description

The contract is using arbitrary numbers to form time-related values. As a result, it decreases the readability of the codebase and prevents the compiler to optimize the source code.

```
uint256 public constant MIN_DELAY = 60;
```

Recommendation

It is a good practice to use the time units reserved keywords like `seconds` , `minutes` , `hours` , `days` and `weeks` to process time-related calculations.

It's important to note that these time units are simply a shorthand notation for representing time in seconds, and do not have any effect on the actual passage of time or the execution of the contract. The time units are simply a convenience for expressing time in a more human-readable form.

UFR - Unutilized Function Redundancy

Criticality	Minor / Informative
Location	contracts/MyVoltToken.sol#L402
Status	Unresolved

Description

The contract is designed to set the `vestingContract` address during its constructor phase, establishing the initial vesting contract to which tokens are allocated or managed. However, it also includes the `setVestingContract` function, that allows for updating the `vestingContract` address post-deployment. This function is intended to provide flexibility in managing the vesting contract address. Despite this intention, the audit reveals that the contract does not utilize the vestingContract address beyond the constructor phase for any operational purposes, such as token distribution or vesting management. Consequently, the ability to update the `vestingContract` address through the `setVestingContract` function becomes redundant, as changing the address after the initial setup does not impact the contract's functionality or behavior. This redundancy not only adds unnecessary complexity to the contract but also poses a risk of confusion, suggesting a level of flexibility and functionality that is not actually supported by the contract's logic.

```
    function setVestingContract(address newContract) external onlyOwner {
        require(newContract != address(0), "Invalid Address!");
        vestingContract = newContract;
    }
```

Recommendation

It is recommended to remove the redundant `setVestingContract` function from the contract. Eliminating this function simplifies the contract, reducing its complexity and potential attack surface. By removing the ability to change the vestingContract address post-deployment, the contract's code more accurately reflects its operational capabilities and limitations, enhancing clarity and maintainability. Additionally, this change ensures that the contract's behavior remains consistent with its design intentions, avoiding any confusion or misleading implications about the role and utility of the vestingContract

address within the contract's ecosystem. This recommendation aligns with best practices for smart contract development, emphasizing simplicity, security, and clarity.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/MyVoltToken.sol#L407
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.

```
address _stakingContract
```

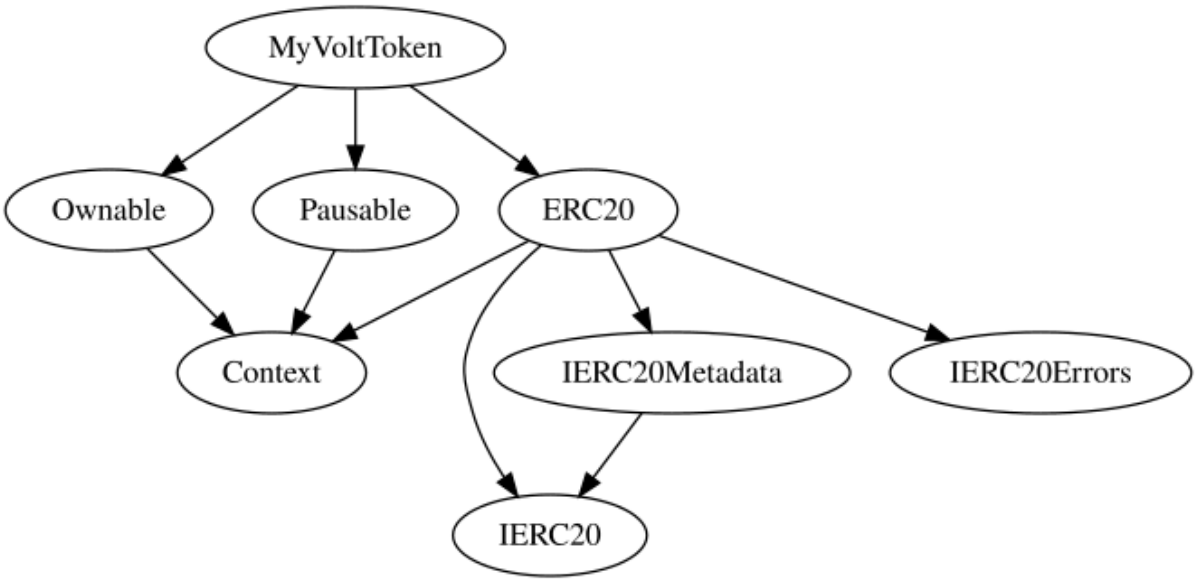
Recommendation

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.

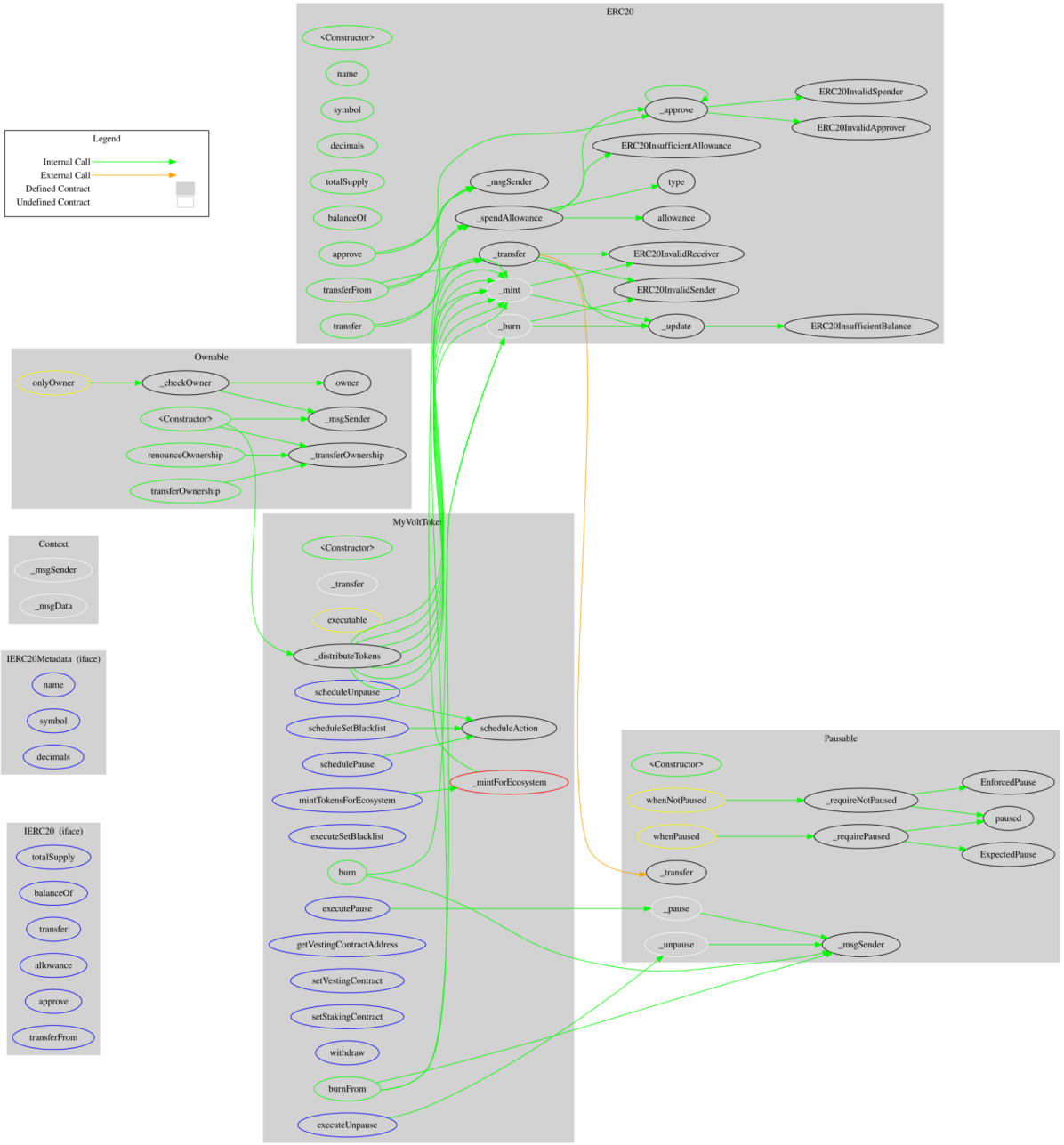
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
MyVoltToken	Implementation	ERC20, Ownable, Pausable		
		Public	✓	ERC20
	_transfer	Internal	✓	whenNotPaused
	scheduleAction	Internal	✓	onlyOwner
	schedulePause	External	✓	onlyOwner
	executePause	External	✓	onlyOwner executable
	scheduleUnpause	External	✓	onlyOwner
	executeUnpause	External	✓	onlyOwner executable
	burn	Public	✓	-
	burnFrom	Public	✓	-
	getVestingContractAddress	External		-
	setVestingContract	External	✓	onlyOwner
	setStakingContract	External	✓	onlyOwner
	withdraw	External	✓	onlyOwner
	_distributeTokens	Private	✓	

Inheritance Graph



Flow Graph


ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
approve
transferFrom
transfer
_msgSender
_spendAllowance
_transfer
_mint
_burn
_approve
ERC20InvalidSpender
ERC20InvalidApprover
ERC20InsufficientAllowance
type
allowance
ERC20InvalidReceiver
ERC20InvalidSender
_update
ERC20InsufficientBalance
Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Ownable
onlyOwner
_checkOwner
owner
<Constructor>
_msgSender
renounceOwnership
_transferOwnership
transferOwnership
Context
_msgSender
_msgData
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
MyVoltToken
<Constructor>
_transfer
executable
_distributeTokens
scheduleUnpause
scheduleSetBlacklist
schedulePause
mintTokensForEcosystem
executeSetBlacklist
burn
executePause
getVestingContractAddress
setVestingContract
setStakingContract
withdraw
burnFrom
executeUnpause
scheduleAction
_mintForEcosystem
Pausable
<Constructor>
whenNotPaused
whenPaused
_transfer
_pause
_unpause
_requireNotPaused
_requirePaused
_msgSender
EnforcedPause
paused
ExpectedPause

Summary

MyVolt contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements.

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