Cyberscope

Audit Report

Greenvault

February 2025

Network	BSC
Address	0xD8C3bB1B6d8C6864a3C083Ba38a857df721F2B1d
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
	UAR	Unexcluded Address Restrictions	Unresolved
	IDI	Immutable Declaration Improvement	Unresolved
	MEE	Missing Events Emission	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L09	Dead Code Elimination	Unresolved
	L20	Succeeded Transfer Check	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	Greenvault
Compiler Version	v0.8.19+commit.7dd6d404
Optimization	200 runs
Explorer	https://bscscan.com/address/0xd8c3bb1b6d8c6864a3c083ba38a857df721f2b1d
Address	0xd8c3bb1b6d8c6864a3c083ba38a857df721f2b1d
Network	BSC
Symbol	GRVT
Decimals	18
Total Supply	100.000.000.000
Badge Eligibility	Yes

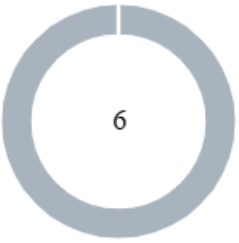
Audit Updates

Initial Audit	16 Feb 2025

Source Files

Filename	SHA256
Greenvault.sol	a05be5ea94169d9f5dc7f25c3284e77c148156230276db5df165b491f8ebe173

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	6

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	6	0	0	0

UAR - Unexcluded Address Restrictions

Criticality	Minor / Informative
Location	Greenvault.sol#L369,408
Status	Unresolved

Description

The contract incorporates operational restrictions on transactions, which can hinder seamless interaction with decentralized applications (dApps) such as launchpads, presales, lockers, or staking platforms. In scenarios where an external contract, such as a launchpad factory, needs to integrate with the contract, it should be exempt from the limitations to ensure uninterrupted service and functionality. Failure to provide such exemptions can block the successful process and operation of services reliant on this contract.

```
function _transfer(address from,address to,uint256 amount)
internal  override {
    //...
    uint256 _totalFees;
    if (_isExcludedFromFees[from] || _isExcludedFromFees[to] ||
swapping) {
        _totalFees = 0;
    } else if (from == uniswapV2Pair) {
        _totalFees = feeOnBuy;
    } else if (to == uniswapV2Pair) {
        _totalFees =  feeOnSell;
    } else {
        _totalFees = feeOnTransfer;
    }

    if (_totalFees > 0) {
        uint256 fees = (amount * _totalFees) / 100;
        amount = amount - fees;
        super._transfer(from, address(this), fees);
    }
    //...
}
```

Recommendation

It is advisable that the contract is able to handle the need for exclusion of designated addresses from transactional restrictions. This enhancement will allow specific addresses, such as those associated with decentralized applications (dApps) and service platforms, to operate without being hindered by the standard constraints imposed on other users. Implementing this feature will ensure smoother integration and functionality with external systems, thereby expanding the contract's versatility and effectiveness in diverse operational environments.

IDI - Immutable Declaration Improvement

Criticality	Minor / Informative
Location	Greenvault.sol#L286
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.

```
uniswapV2Pair
```

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.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	Greenvault.sol#L314
Status	Unresolved

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
function claimStuckTokens(address token) external onlyOwner {
    require(token != address(this), "CBUL: Owner cannot claim
contract's balance of its own tokens");
    if (token == address(0x0)) {
        payable(msg.sender).sendValue(address(this).balance);
        return;
    }

    IERC20(token).transfer(msg.sender,
IERC20(token).balanceOf(address(this)));
}
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	Greenvault.sol#L11,336,350,417
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 _feeOnSell
uint256 _feeOnBuy
uint256 _feeOnTransfer
address _feeReceiver
bool _swapEnabled
```

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.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	Greenvault.sol#L213
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 amount) internal
virtual {
        require(account != address(0), "ERC20: burn from the
zero address");

        uint256 accountBalance = _balances[account];
        require(accountBalance >= amount, "ERC20: burn amount
exceeds balance");
        unchecked {
            _balances[account] = accountBalance - amount;
        }
        _totalSupply -= amount;

        emit Transfer(account, address(0), 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.

L20 - Succeeded Transfer Check

Criticality	Minor / Informative
Location	Greenvault.sol#L321
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
IERC20(token).transfer(msg.sender,
IERC20(token).balanceOf(address(this)))
```

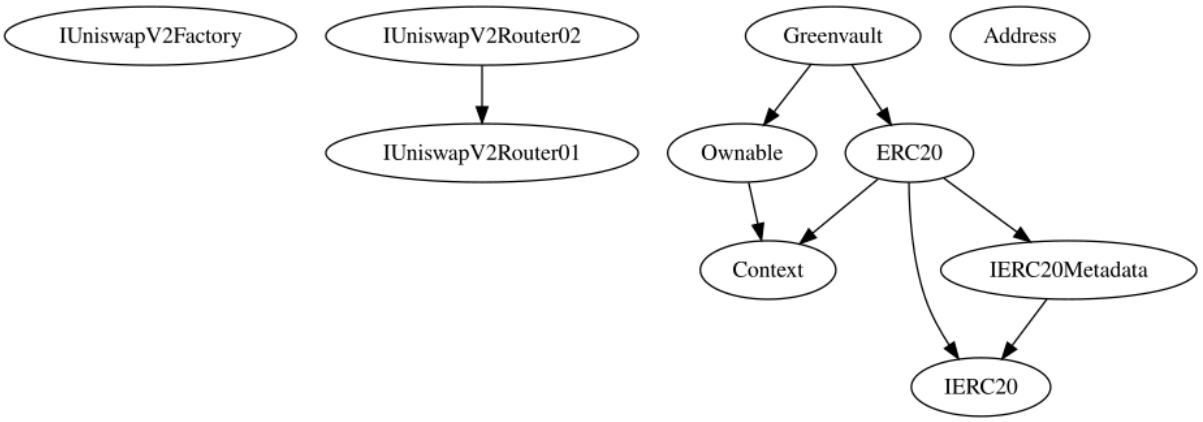
Recommendation

The contract should check if the result of the transfer methods is successful. The team is advised to check the SafeERC20 library from the Openzeppelin library.

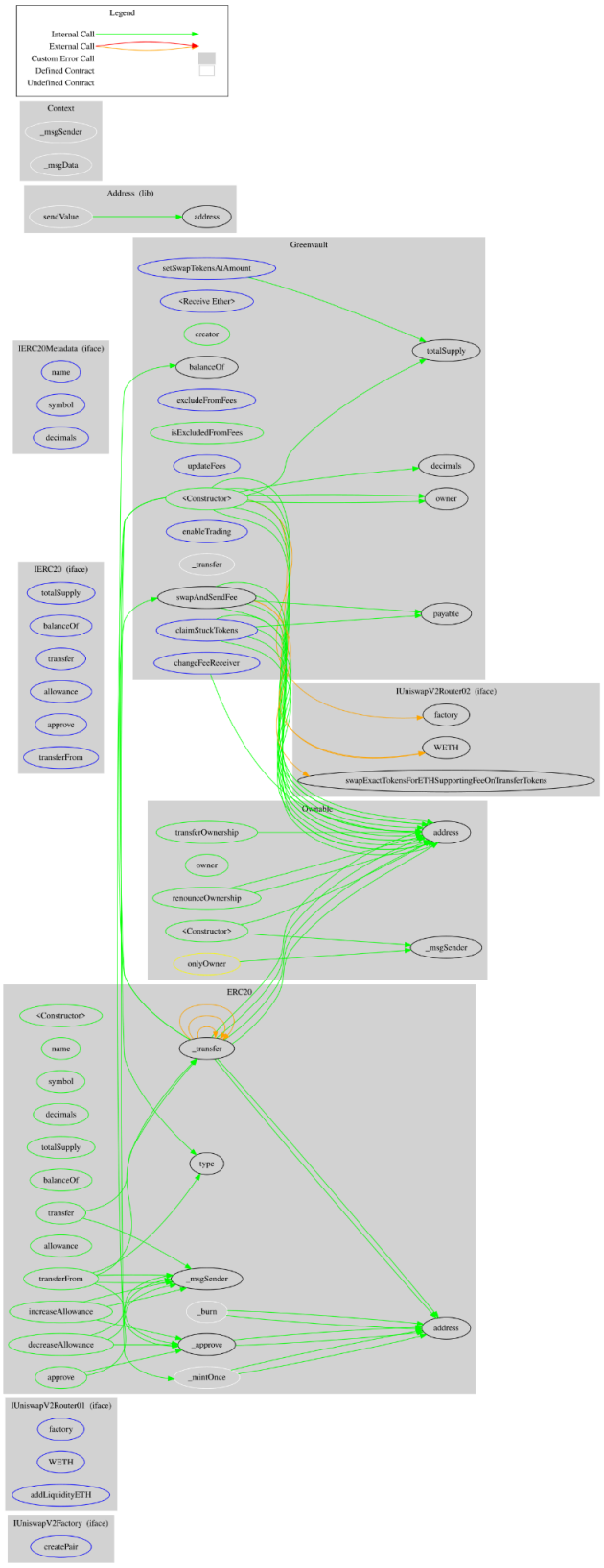
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Greenvault	Implementation	ERC20, Ownable		
		Public	✓	ERC20
		External	Payable	-
	creator	Public		-
	claimStuckTokens	External	✓	onlyOwner
	excludeFromFees	External	✓	onlyOwner
	isExcludedFromFees	Public		-
	updateFees	External	✓	onlyOwner
	changeFeeReceiver	External	✓	onlyOwner
	enableTrading	External	✓	onlyOwner
	_transfer	Internal	✓	
	setSwapTokensAtAmount	External	✓	onlyOwner
	swapAndSendFee	Private	✓	

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Custom Error Call
Defined Contract
Undefined Contract
Context
_msgSender
_msgData
Address (lib)
sendValue
address
Greenvault
setSwapTokensAtAmount
<Receive Ether>
creator
balanceOf
excludeFromFees
isExcludedFromFees
updateFees
<Constructor>
enableTrading
_transfer
swapAndSendFee
claimStuckTokens
changeFeeReceiver
totalSupply
decimals
owner
payable
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
IUniswapV2Router02 (iface)
factory
WETH
swapExactTokensForETHSupportingFeeOnTransferTokens
Ownable
transferOwnership
owner
renounceOwnership
<Constructor>
onlyOwner
address
_msgSender
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
transfer
allowance
transferFrom
increaseAllowance
decreaseAllowance
approve
_transfer
type
_msgSender
_burn
_approve
_mintOnce
address
IUniswapV2Router01 (iface)
factory
WETH
addLiquidityETH
IUniswapV2Factory (iface)
createPair

Summary

GreenVault contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. GreenVault is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues. There is a limit of max 10% fees.

The contract's ownership has been renounced. The information regarding the transaction can be accessed through the following link:

https://bscscan.com/tx/0x5d79ad66ca2757a177ac07ab76ac80af03caa911180dd3a27f27096fd77312ae

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