Cyberscope

A TAC Security Company

Audit Report

VIETDOGE

June 2025

Network BSC

Address 0xdf365809AFf206e41dcb12a3d68930f5aF0E1452

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
●	ROF	Redundant Owner Functions	Unresolved
●	L09	Dead Code Elimination	Unresolved
●	L18	Multiple Pragma Directives	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	VietDoge
Compiler Version	v0.8.24+commit.e11b9ed9
Optimization	200 runs
Explorer	https://bscscan.com/address/0xdf365809aff206e41dcb12a3d68930f5af0e1452
Address	0xdf365809aff206e41dcb12a3d68930f5af0e1452
Network	BSC
Symbol	VIETDOGE
Decimals	18
Total Supply	69,000,000,000
Badge Eligibility	Yes

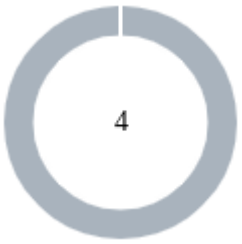
Audit Updates

Initial Audit	17 Jun 2025

Source Files

Filename	SHA256
contracts/VietDoge.sol	a632cbc1b3e00b76f6b02945b69d0022c613967f0874997c445dca213ffe0cfe

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	4

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	4	0	0	0

ROF - Redundant Owner Functions

Criticality	Minor / Informative
Location	contracts/VietDoge.sol#L14,19
Status	Unresolved

Description

The contract is implementing redundant functions that replicate existing standard functionality. Specifically, the `distributeTokens` function is restricted to the owner and performs a basic token transfer from the owner's balance to a recipient. This mirrors the behaviour already provided by the standard `transfer` function. Additionally, the `renounceOwnership` function is overridden without introducing any new logic or restrictions, effectively duplicating the default OpenZeppelin implementation unnecessarily. These duplications add code bloat and may cause confusion for maintainers or auditors reviewing the contract logic.

```
    function distributeTokens(address to, uint256 amount) external
onlyOwner {
        require(amount <= balanceOf(msg.sender), "Not enough tokens");
        _transfer(msg.sender, to, amount);
    }

    function renounceOwnership() public override onlyOwner {
        super.renounceOwnership();
    }
```

Recommendation

It is recommended to consider removing both functions, as they provide no added value beyond existing standard library capabilities. The `distributeTokens` function can be replaced by using the standard `transfer` mechanism, while the `renounceOwnership` function should be inherited directly without an override unless custom behaviour is needed. This will reduce contract size, simplify the codebase, and align with clean coding practices.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	@openzeppelin/contracts/utils/Context.sol#L25 @openzeppelin/contracts/token/ERC20/ERC20.sol#L241
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 _contextSuffixLength() internal view virtual returns
(uint256) {
        return 0;
    }

function _burn(address account, uint256 value) internal {
        if (account == address(0)) {
            revert ERC20InvalidSender(address(0));
        }
        _update(account, address(0), value);
    }
```

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.

L18 - Multiple Pragma Directives

Criticality	Minor / Informative
Location	contracts/VietDoge.sol#L2 @openzeppelin/contracts/utils/Context.sol#L4 @openzeppelin/contracts/token/ERC20/IERC20.sol#L4 @openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#L4 @openzeppelin/contracts/token/ERC20/ERC20.sol#L4 @openzeppelin/contracts/interfaces/draft-IERC6093.sol#L3 @openzeppelin/contracts/access/Ownable.sol#L4
Status	Unresolved

Description

If the contract includes multiple conflicting pragma directives, it may produce unexpected errors. To avoid this, it's important to include the correct pragma directive at the top of the contract and to ensure that it is the only pragma directive included in the contract.

```
pragma solidity ^0.8.20;
pragma solidity ^0.8.24;
```

Recommendation

It is important to include only one pragma directive at the top of the contract and to ensure that it accurately reflects the version of Solidity that the contract is written in.

By including all required compiler options and flags in a single pragma directive, the potential conflicts could be avoided and ensure that the contract can be compiled correctly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	contracts/VietDoge.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;
```

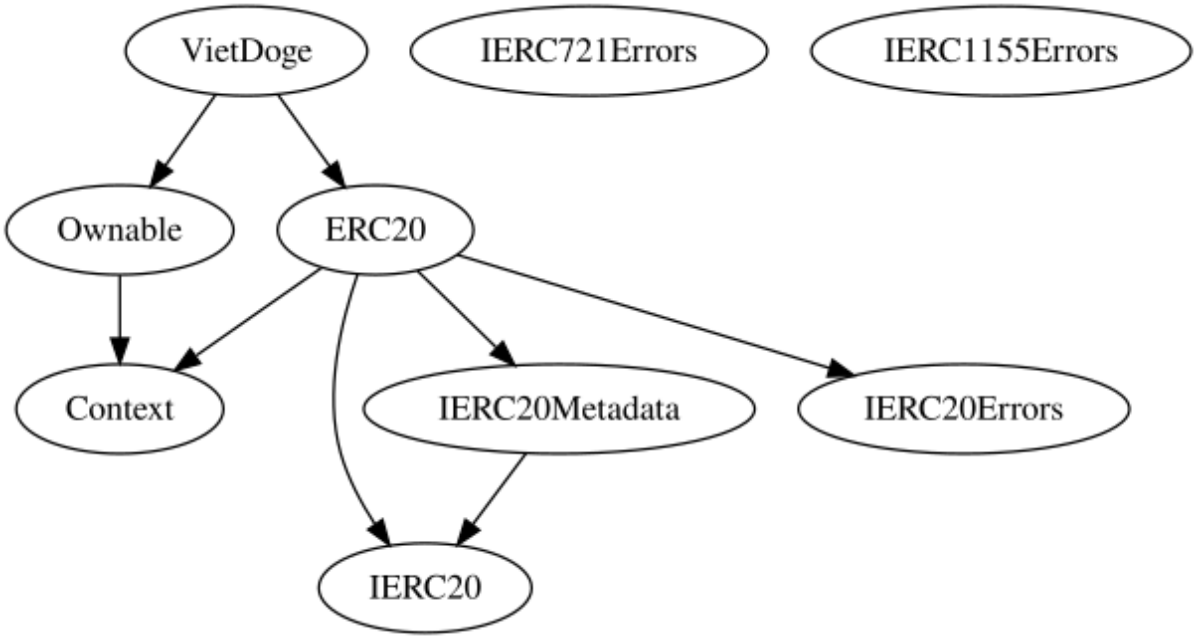
Recommendation

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.

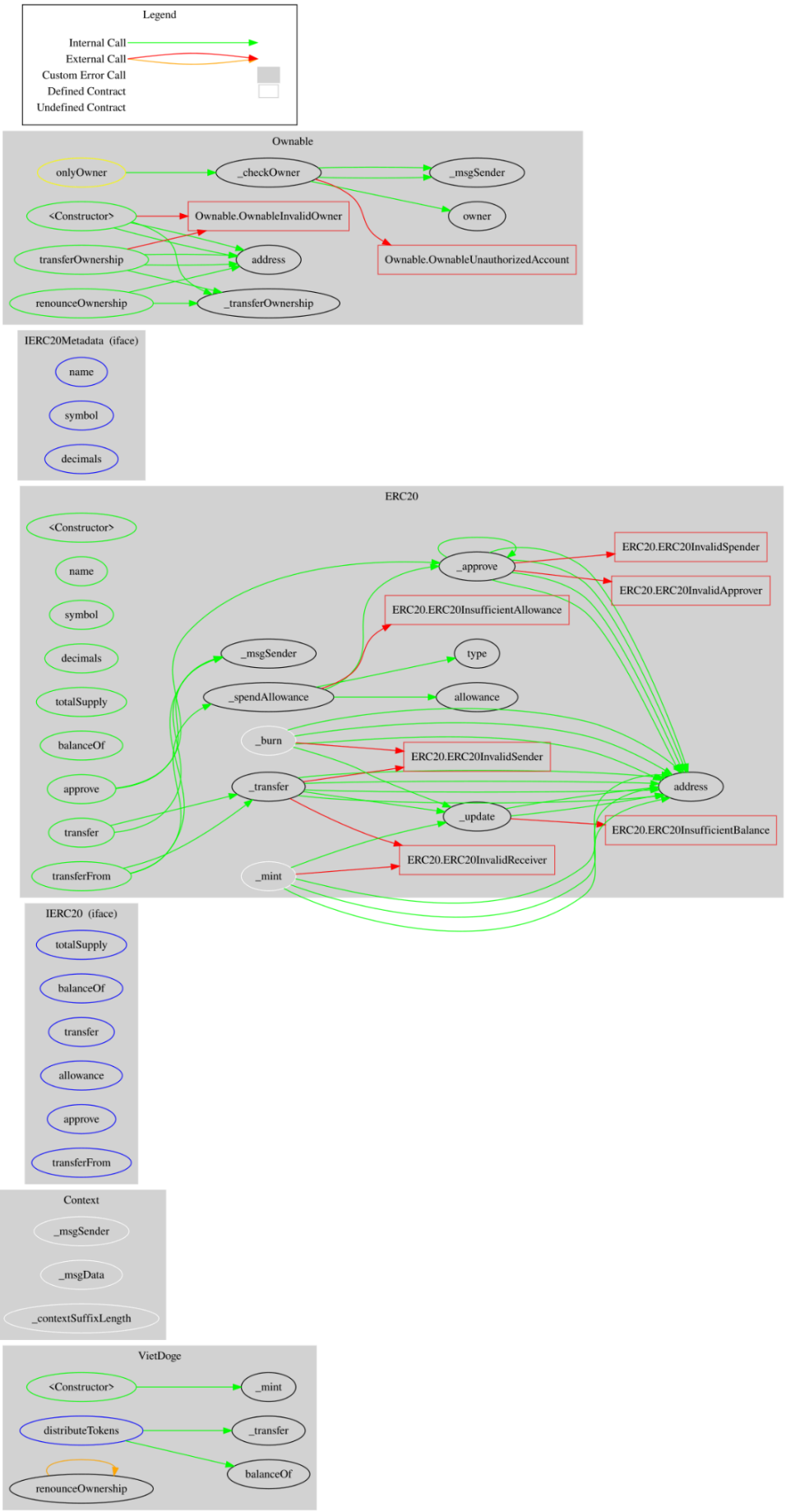
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
VietDoge	Implementation	ERC20, Ownable		
		Public	✓	ERC20 Ownable
	distributeTokens	External	✓	onlyOwner
	renounceOwnership	Public	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Custom Error Call
Defined Contract
Undefined Contract
Ownable
onlyOwner
_checkOwner
_msgSender
owner
<Constructor>
Ownable.OwnableInvalidOwner
transferOwnership
address
Ownable.OwnableUnauthorizedAccount
renounceOwnership
_transferOwnership
IERC20Metadata (iface)
name
symbol
decimals
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
approve
transfer
transferFrom
_approve
ERC20.ERC20InvalidSpender
ERC20.ERC20InvalidApprover
ERC20.ERC20InsufficientAllowance
_msgSender
type
_spendAllowance
allowance
_burn
ERC20.ERC20InvalidSender
_transfer
address
_update
ERC20.ERC20InsufficientBalance
ERC20.ERC20InvalidReceiver
_mint
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Context
_msgSender
_msgData
_contextSuffixLength
VietDoge
<Constructor>
_mint
distributeTokens
_transfer
balanceOf
renounceOwnership

Summary

VIETDOGE contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. VIETDOGE is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions.

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 TAC 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.

A **TAC Security** Company

The Cyberscope team

cyberscope.io