



**CONTRACT
WOLF**

Blockchain Security - Smart Contract Audits



Security Assessment

January 28, 2023

| | |
|--|-----------|
| Disclaimer | 3 |
| Scope of Work & Engagement | 3 |
| Project Description | 4 |
| Risk Level Classification | 5 |
| Methodology | 6 |
| Used Code from other Frameworks / Smart Contracts (Imports) | 7 |
| Token Description | 8 |
| Inheritance Graph | 9 |
| Overall Checkup | 10 |
| Verify Claim | 11 |
| Write Functions of Contract | 12 |
| Call Graph | 13 |
| SWC Attacks | 14 |
| Audit Result | 16 |
| Findings | 17 |
| Audit Comments | 19 |

Disclaimer

ContractWolf.io audits and reports should not be considered as a form of project's "advertisement" and does not cover any interaction and assessment from "project's contract" to "external contracts" such as Pancakeswap or similar.

ContractWolf does not provide any warranty on its released reports.

ContractWolf should not be used as a decision to invest into an audited project and is not affiliated nor partners to its audited contract projects.

ContractWolf provides transparent report to all its "clients" and to its "clients participants" and will not claim any guarantee of bug-free code within its **SMART CONTRACT**.

ContractWolf presence is to analyze, audit and assess the client's smart contract's code.

Each company or projects should be liable to its security flaws and functionalities.

Scope of Work

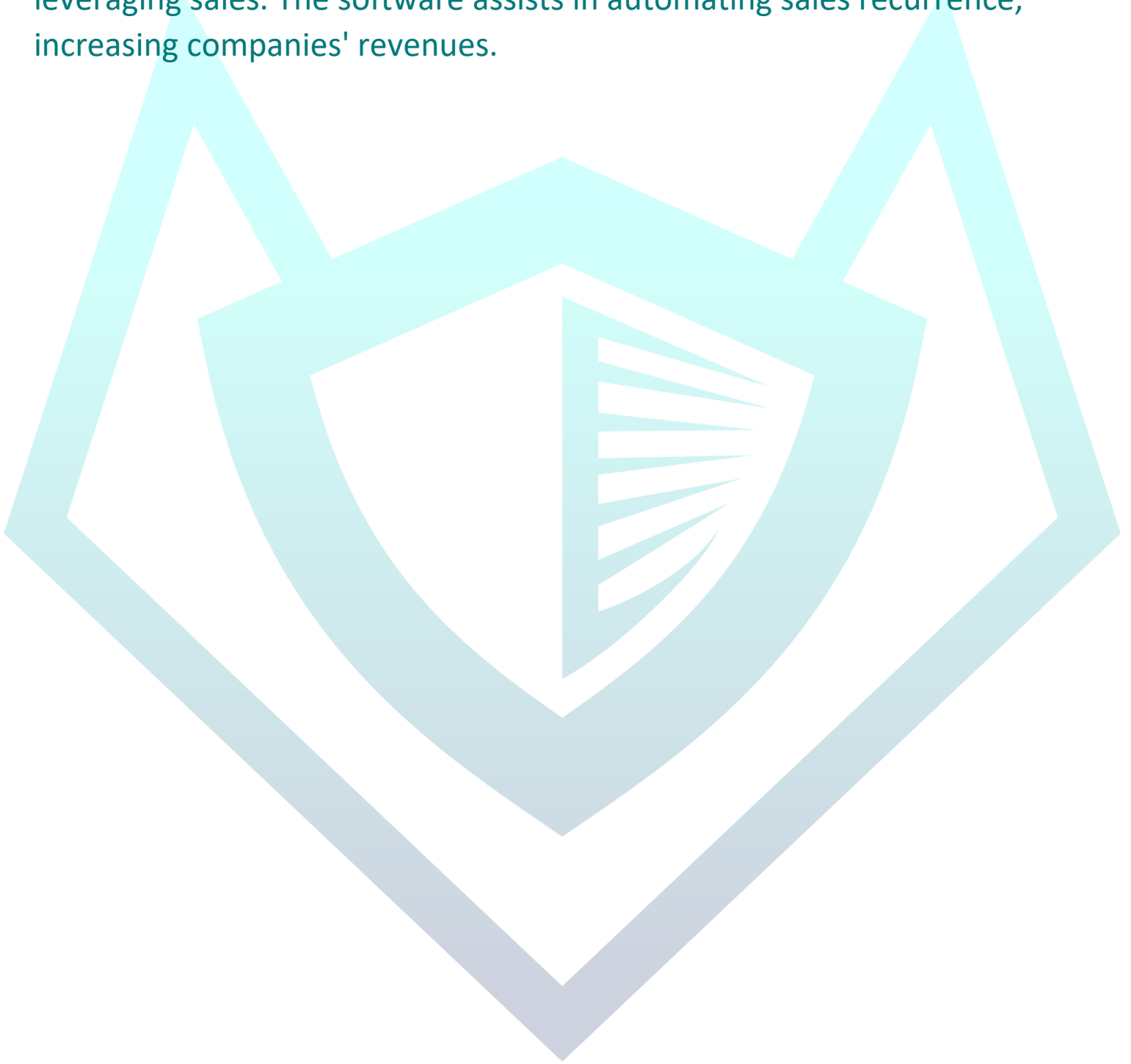
Prats Token team agreed and provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract.

The goal of this engagement was to identify if there is a possibility of security flaws in the implementation of the contract or system.

ContractWolf will be focusing on contract issues and functionalities along with the projects claims from smart contract to their website, whitepaper and repository which has been provided by **Prats Token**.

Description

Prats Token is the token of Prátis CRM which is a B2B software aimed at leveraging sales. The software assists in automating sales recurrence, increasing companies' revenues.



Risk Level Classification

Risk Level represents the classification or the probability that a certain function or threat that can exploit vulnerability and have an impact within the system or contract.

Risk Level is computed based on CVSS Version 3.0

| Level | Value | Vulnerability |
|---------------|-----------|---|
| Critical | 9 - 10 | An Exposure that can affect the contract functions in several events that can risk and disrupt the contract |
| High | 7 - 8.9 | An Exposure that can affect the outcome when using the contract that can serve as an opening in manipulating the contract in an unwanted manner |
| Medium | 4 - 6.9 | An opening that could affect the outcome in executing the contract in a specific situation |
| Low | 0.1 - 3.9 | An opening but doesn't have an impact on the functionality of the contract |
| Informational | 0 | An opening that consists of information's but will not risk or affect the contract |

Auditing Approach

Every line of code along with its functionalities will undergo manual review to check its security issues, quality, and contract scope of inheritance. The manual review will be done by our team that will document any issues that there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:

- Review of the specifications, sources, and instructions provided to ContractWolf to make sure we understand the size, scope, and functionality of the smart contract.
- Manual review of code, our team will have a process of reading the code line-by-line with the intention of identifying potential vulnerabilities and security flaws.

2. Testing and automated analysis that includes:

- Testing the smart contract functions with common test cases and scenarios, to ensure that it returns the expected results.

3. Best practices review, the team will review the contract with the aim to improve efficiency, effectiveness, clarifications, maintainability, security, and control within the smart contract.

4. Recommendations to help the project take steps to secure the smart contract.

Used Code from other Frameworks/Smart Contracts (Direct Imports)

Imported Packages

- Context
- IERC20
- IERC20Metadata
- ERC20
- Address
- Ownable
- IFactory
- IRouter
- PratsOfficialToken

Description

Optimization enabled: No

Decimal: 3

Symbol: PRATS

Max / Total Supply: 1,000,000,000

Capabilities

Components

| Version | Contracts | Libraries | Interfaces | Abstract |
|---------|-----------|-----------|------------|----------|
| 1.0 | 2 | 1 | 4 | 2 |

Exposed Functions

| Version | Public | Private | External | Internal |
|---------|--------|---------|----------|----------|
| 1.0 | 20 | 4 | 32 | 9 |

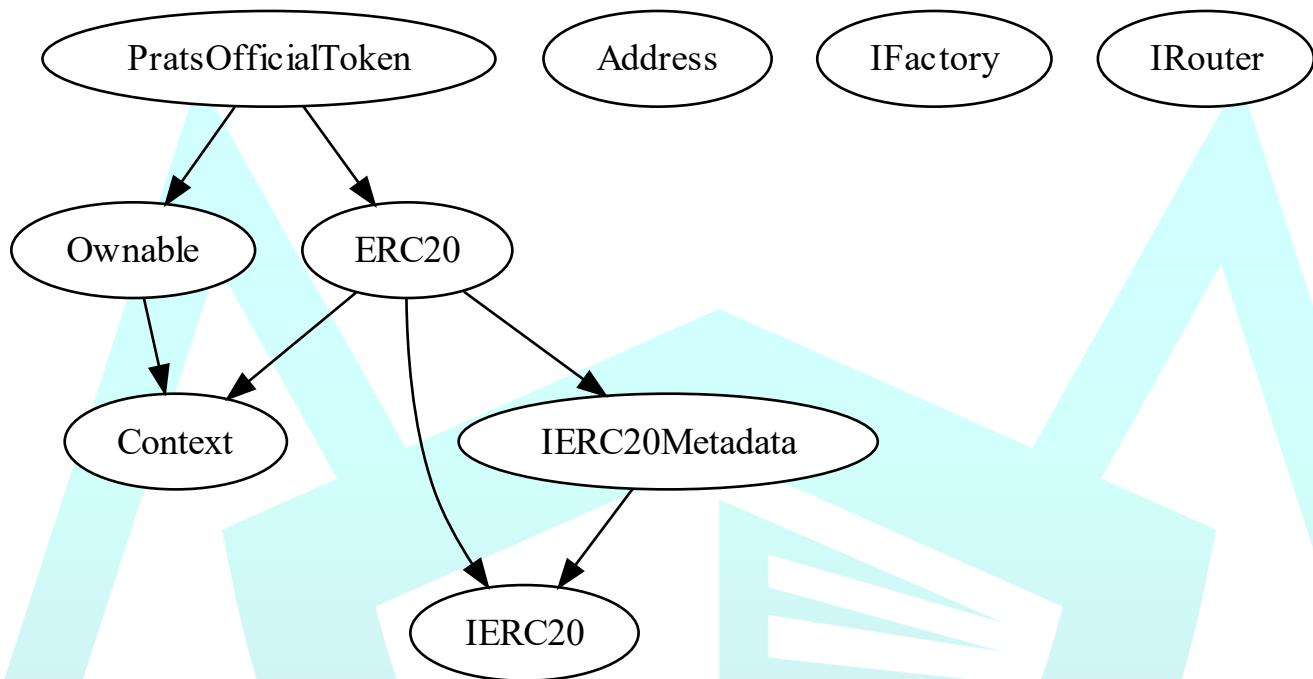
State Variables

| Version | Total | Public |
|---------|-------|--------|
| 1.0 | 26 | 12 |

Capabilities

| Version | Solidity Versions Observed | Experimental Features | Can Receive Funds | Uses Assembly | Has Destroyable Contracts |
|---------|----------------------------|-----------------------|-------------------|---------------|---------------------------|
| 1.0 | v0.8.7 | | Yes | No | No |

Inheritance Graph



Correct implementation of Token Standard

| Tested | Verified |
|--------|----------|
| ✓ | ✓ |

Overall Checkup (Smart Contract Security)

| Tested | Verified |
|--------|----------|
| ✓ | ✓ |

| Function | Description | Exist | Tested | Verified |
|--------------|--|-------|--------|----------|
| TotalSupply | Information about the total coin or token supply | ✓ | ✓ | ✓ |
| BalanceOf | Details on the account balance from a specified address | ✓ | ✓ | ✓ |
| Transfer | An action that transfers a specified amount of coin or token to a specified address | ✓ | ✓ | ✓ |
| TransferFrom | An action that transfers a specified amount of coin or token from a specified address | ✓ | ✓ | ✓ |
| Approve | Provides permission to withdraw specified number of coin or token from a specified address | ✓ | ✓ | ✓ |

Verify Claims

| Statement | Exist | Tested | Deployer |
|--------------------|-------|--------|----------|
| Renounce Ownership | ✓ | ✓ | ✓ |
| Mint | — | — | — |
| Burn | ✓ | ✓ | ✓ |
| Block | ✓ | ✓ | ✓ |
| Pause | — | — | — |

Legend

| Attribute | Symbol |
|--------------------------|--------|
| Verified / Can | ✓ |
| Verified / Cannot | ✗ |
| Unverified / Not checked | 🚩 |
| Not Available | — |

Write Functions of Contract

1. BatchBlockBot (0x45a82e37)

2. BlockBot (0xf68366a0)

3. SetSellTaxes (0x4e736f22)

4. UpdateBuyTaxes (0x9d282cd6)

5. approve (0x095ea7b3)

6. bulkAllowedTransfer (0x81428be1)

7. bulkExemptFee (0x0e375a5c)

8. burn (0x42966c68)

9. decreaseAllowance (0xa457c2d7)

10. enableTrading (0x8a8c523c)

11. increaseAllowance (0x39509351)

12. renounceOwnership (0x715018a6)

13. transfer (0xa9059cbb)

14. transferFrom (0x23b872dd)

15. transferOwnership (0xf2fde38b)

16. updateAllowedTransfer (0xb5c57145)

17. updateCooldown (0xe517f2b9)

18. updateDevWallet (0x1816467f)

19. updateExemptFee (0x355496ca)

20. updateLiquidityProvide (0x1340538f)

21. updateLiquidityTreshhold (0x42b6fa11)

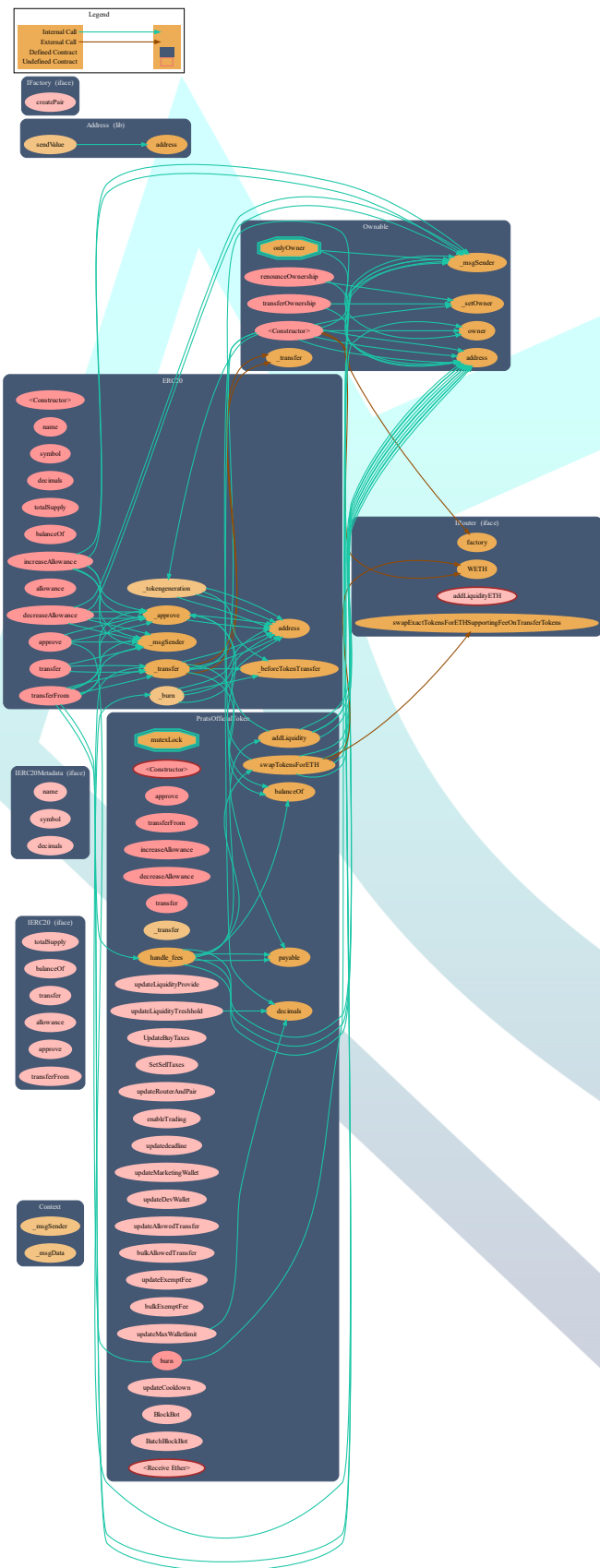
22. updateMarketingWallet (0xaacebbe3)

23. updateMaxWalletlimit (0xd8672e51)

24. updateRouterAndPair (0x40b28c2f)

25. updatedeadline (0xeda1168)

Call Graph



SWC Attacks

| ID | Title | Status |
|-------------------------|---|-----------|
| SWC-136 | Unencrypted Private Data On-Chain | PASSED |
| SWC-135 | Code With No Effects | PASSED |
| SWC-134 | Message call with hardcoded gas amount | PASSED |
| SWC-133 | Hash Collisions with Multiple Variable Length Arguments | PASSED |
| SWC-132 | Unexpected Ether balance | PASSED |
| SWC-131 | Presence of unused variables | PASSED |
| SWC-130 | Right-To Left Override control character (U+202E) | PASSED |
| SWC-129 | Typographical Error | PASSED |
| SWC-128 | DoS With Block Gas Limit | PASSED |
| SWC-127 | Arbitrary Jump with Function Type Variable | PASSED |
| SWC-126 | Insufficient Gas Griefing | PASSED |
| SWC-125 | Incorrect Inheritance Order | PASSED |
| SWC-124 | Write to Arbitrary Storage Location | PASSED |
| SWC-123 | Requirement Violation | PASSED |
| SWC-122 | Lack of Proper Signature Verification | PASSED |
| SWC-121 | Missing Protection against Signature Replay Attacks | PASSED |
| SWC-120 | Weak Sources of Randomness from Chain Attributes | LOW ISSUE |
| SWC-119 | Shadowing State Variables | PASSED |
| SWC-118 | Incorrect Constructor Name | PASSED |
| SWC-117 | Signature Malleability | PASSED |
| SWC-116 | Block values as a proxy for time | PASSED |
| SWC-115 | Authorization through tx.origin | PASSED |
| SWC-114 | Transaction Order Dependence | PASSED |
| SWC-113 | DoS with Failed Call | PASSED |
| SWC-112 | Delegate call to Untrusted Callee | PASSED |
| SWC-111 | Use of Deprecated Solidity Functions | PASSED |

| | | |
|--------------------------------|--------------------------------------|------------------|
| <u>SWC-110</u> | Assert Violation | PASSED |
| <u>SWC-109</u> | Uninitialized Storage Pointer | PASSED |
| <u>SWC-108</u> | State Variable Default Visibility | PASSED |
| <u>SWC-107</u> | Reentrancy | PASSED |
| <u>SWC-106</u> | Unprotected SELFDESTRUCT Instruction | PASSED |
| <u>SWC-105</u> | Unprotected Ether Withdrawal | PASSED |
| <u>SWC-104</u> | Unchecked Call Return Value | PASSED |
| <u>SWC-103</u> | Floating Pragma | LOW ISSUE |
| <u>SWC-102</u> | Outdated Compiler Version | PASSED |
| <u>SWC-101</u> | Integer Overflow and Underflow | PASSED |
| <u>SWC-100</u> | Function Default Visibility | PASSED |

AUDIT PASSED

Low Issues

| | |
|---|----------------|
| A floating pragma is set (SWC-103) | L: 6 |
| Potential use of “block.number” as source of randomness (SWC-120) | L: 606, L: 761 |

Findings

Description:

A floating pragma is set (SWC-103)

Suggestion:

Specific version to ensure that the bytecode does not vary between builds.

Description:

Potential use of “block.number” as source of randomness (SWC-120)

Suggestion:

- Using commitment scheme, e.g. RANDAO.
 - Using external sources of randomness via oracles, e.g. Oraclize. Note that this approach requires trusting in oracle, thus it may be reasonable to use multiple oracles.
 - Using of Bitcoin block hashes, as they are more expensive to mine.
-

Owner can block user

```
function BlockBot(address account↑, bool state↑) external onlyOwner {
    isBot[account↑] = state↑;
}
```

```
function BatchBlockBot(address[] memory accounts↑, bool state↑) external onlyOwner {
    for (uint256 i = 0; i < accounts↑.length; i++) {
        isBot[accounts↑[i]] = state↑;
    }
}
```

Owner can update buy taxes up to 25%

```
function UpdateBuyTaxes(  
    uint256 _marketing↑,  
    uint256 _liquidity↑,  
    uint256 _dev↑  
) external onlyOwner {  
    taxes = Taxes(_marketing↑, _liquidity↑, _dev↑);  
    require((_marketing↑ + _liquidity↑ + _dev↑) <= 25, "Buy taxes up to 25% only");  
}
```

Owner can set sell taxes up to 25%

```
function SetSellTaxes(  
    uint256 _marketing↑,  
    uint256 _liquidity↑,  
    uint256 _dev↑  
) external onlyOwner {  
    sellTaxes = Taxes(_marketing↑, _liquidity↑, _dev↑);  
    require((_marketing↑ + _liquidity↑ + _dev↑) <= 25, "Sell taxes up to 25% only");  
}
```

Audit Comments

- Owner can burn tokens
- Owner can renounce ownership
- Owner can transfer ownership
- Owner can toggle liquidity providing status
- Owner can update max wallet limit not lower than 0.,1%
- Owner can update cooldown up to 5 minutes
- Owner can update liquidity threshold
- Owner can update buy taxes up to 25%
- Owner can update router and pair addresses
- Owner can update deadline
- Owner can enable trading
- Owner can set sell taxes up to 25%
- Owner can set marketing wallet
- Owner can set dev wallet
- Owner can include/exclude addresses from allowed transfers
- Owner can include/exclude addresses from fees
- Owner can block user
- Owner cannot pause contract
- Owner cannot mint after initial deployment



CONTRACTWOLF

Blockchain Security - Smart Contract Audits