Token Raven

# Token Raven Security Audit

# Corps D'elite

Blockchain and Smart Contract

**Audit | KYC | Consultancy | Development**

Services

Token Raven

# Summary

Token Raven has performed a combination of manual and software analysis of the smart contracts provided by the project team. Each of them has been analyzed for the most common and exploits, vulnerabilities, and manipulation attacks.

| | |
|---|---|
| **Audit verdict** | Passed |
| **Manual Analysis Risks Found** | 6 |
| **Centralization Risk** | Low (contract is renounced) |
| **SWC Issues** | 3 |
| **Number of high-risk issues** | 0 |
| **Issues Resolved** | 0 |



The authenticity of this document can be verified on the Token Raven GitHub https://github.com/tokenraven

This audit was completed on: **05-06-2022**

Token Raven

# Table of Contents

# Audit Summary

## Provided Description

We are an Elite Blockchain Society dedicated to improve: Wisdom, Wealth, Knowledge, and Power. In due time you will be ranked according to your holdings.

| | |
|---|---|
| **Project Name** | Corps D'elite |
| **Platform** | Ethereum |
| **Language** | Solidity |
| **Contract Address** | 0x55B52e973fE7AaB49aeb090Dc4D66C99Ad3d0Bcb |
| **Compiler Version** | 0.8.9 |
| **Token Ticker** | Corps Delite |
| **Total Supply** | 11,000,000,000 |
| **Decimals** | 18 |
| **Tax Fees** | 3% |
| **Deployer Address** | 0xe702404a87CC748d3e71cD2fb78e6E9f0cc9d62f |
| **Current Owner** | 0x000000000000000000000000000000000000dead |

Link to contract:

https://etherscan.io/address/0x55B52e973fE7AaB49aeb090Dc4D66C99Ad3d0Bcb#code=

# Audit Process

Our rigorous audit process is as follows.

**Step 1**

The smart contract is tested on the Testnet of the corresponding network as well as using software analysis tools to make sure that everything is functioning. If we notice any issues, we will provide recommendations to get these issues resolved.

**Step 2**

Here we check whether there are functions in the contract that are only for the owner of the contract, which therefore entail privileges. If we find this type of feature, it will be communicated to our partners as we recommend this is to be changed.

**Step 3**

If our partner does not want to make changes to these owner-only functions, we recommend that you verify through a KYC verification with us. We then indicate during the audit that these functions are included in the contract, but that the owner has been verified at Token Raven.

**Step 4**

Once we check the contract line by line in the auditing process and our recommendations for problem fixes are implemented, this will be communicated to the project team.

**Step 5**

If everything is in order after the previous 4 steps, the audit report including the certificate will be displayed on our website and GitHub. In the report, all our comments are divided into categories, ranging from low risk to high risk labels.

# Common Risks

The cold hard logic of smart contracts allows them to facilitate the most secure and trustworthy kinds of transactions to date. This infallible ability to always execute the provided code does mean however, that the code should be free of errors in order to prevent security hazards.

The top ten Smart Contract Risks (SCR) fall into three categories

## Operational Risks

SCR-1: Super User Account or Privilege Management

SCR-2: Blacklisting and Burning Functions

SCR-3: Contract Logic or Asset Configuration can be arbitrarily changed

SCR-4: Self-Destruct Functions

SCR-5: Minting Functions

## Implementation Risks

SCR-6: Rolling Your Own Crypto and Unique Contract Logic

SCR-7: Unauthorized Transfers

SCR-8: Incorrect Signature Implementation or Arithmetic

## Design Risks

SCR-9: Untrusted Control Flow

SCR-10: Transaction Order Dependence

All of these risks are checked for either manually or through our software analyses.

# Manual Analysis

Our manual analysis is focused on attributes that can not be verified or assessed through software analysis.

## Owner Privileges

### Owner can pause and unpause contract

The owner of the contract can pause the contract and thus all token transfers.

```
    function pause() public onlyOwner {
        require(!paused(), "CoinToken: Contract is already paused");
        _pause();
    }

    function unpause() public onlyOwner {
        require(paused(), "CoinToken: Contract is not paused");
        _unpause();
    }
```

**Recommendation:**

Delete this function or renounce the contract so that holders will always be able to transfer tokens.

### Owner can set fees higher than 25%

The owner can set fees higher than 25%.

```
    function setBuyTax(uint256 dev, uint256 marketing, uint256 liquidity,
uint256 charity) public onlyOwner {
        buyTaxes["dev"] = dev;
        buyTaxes["marketing"] = marketing;
        buyTaxes["liquidity"] = liquidity;
        buyTaxes["charity"] = charity;
    }

    function setSellTax(uint256 dev, uint256 marketing, uint256 liquidity,
uint256 charity) public onlyOwner {

        sellTaxes["dev"] = dev;
        sellTaxes["marketing"] = marketing;
        sellTaxes["liquidity"] = liquidity;
        sellTaxes["charity"] = charity;
    }
```

**Recommendation:**

We advise the project owner to implement a maximum fee setting to at most 25%.

This to protect the holders and investors.

### Owner can exclude wallets from fee and transfers

Owner has the ability to exclude wallets from transaction fees or can blacklist wallets so that they are not able to send or receive tokens.

```
    function exclude(address account) public onlyOwner {
        require(!isExcluded(account), "CoinToken: Account is already
excluded");
        excludeList[account] = true;
    }

    function enableBlacklist(address account) public onlyOwner {
        require(!blacklist[account], "CoinToken: Account is already
blacklisted");
        blacklist[account] = true;
    }
```

**Recommendation:**

As these functions can also be used to protect holders and the project we do not advise to get rid of the functions. However, we recommend to communicate all these wallets to the community to stay transparent about these functions.

## Contract Name Fault

The name of the contract does not match the name of the project.

```
contract CoinToken is ERC20, Ownable, Pausable {}
```

**Recommendation:**

Change the name of the contract so that it matches the project name.

## Use of block.timestamp

Contract makes use of block.timestamp, timestamp can be manipulated by miners.

```
                    uniswapV2Router02.swapExactTokensForETH(
                        toSell,
                        0,
                        sellPath,
```

```
                    address(this),
                    block.timestamp
                );
```

**Recommendation:**

We recommend to avoid relying on the block timestamp.

## Call values not used

Return value of low-level calls not used.

```
                taxWallets["marketing"].call{value: marketingETH}("");
                taxWallets["dev"].call{value: devETH}("");
                taxWallets["charity"].call{value: charityETH}("");

                if(ethGained - (marketingETH + devETH + liquidityETH +
charityETH) > 0) {
                    taxWallets["marketing"].call{value: ethGained -
(marketingETH + devETH + liquidityETH + charityETH)}("");
                }
```

**Recommendation:**

We advise the project developers to take a look at these variables and values to find out whether they should be used or not.

| Issue | Category | Risk | Status |
|---|---|---|---|
| Owner can pause and unpause contract | Owner privileges | Medium | Unresolved |
| Owner can set fees higher than 25% | Owner privileges | Medium | Unresolved |
| Owner can exclude wallets from fee and transfers | Owner privileges | Low | Unresolved |
| Contract Name Fault | Contract name fault | Low | Unresolved |

| | | | |
|---|---|---|---|
| Use of block.timestamp | Use of block.timestamp | Low | Unresolved |
| Call values not used | Unused values | Low | Unresolved |

# Software Analysis

## SWC Attacks

Part of the software analysis is the check for smart contract vulnerabilities. These vulnerabilities are registered at the SWC Registry which is an implementation of the weakness classification scheme proposed in EIP-1470. The goals of the SWC Registry are:

- Providing a straightforward way to classify security issues in smart contract systems
- Define a common language for describing security issues in smart contract systems' architecture, design, or code
- Serve as a way to train and increase performance for smart contract security analysis tools

Token Raven uses Consensys software tools to analyze smart contracts on 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 | Failed |
| 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 | Passed |
| 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 | Failed |
| SWC-115 | Authorization through tx.origin | Passed |
| SWC-114 | Transaction Order Dependence | Passed |
| SWC-113 | DoS with Failed Call | Passed |
| SWC-112 | Delegatecall to Untrusted Callee | Passed |
| SWC-111 | Use of Deprecated Solidity Functions | Passed |
| SWC-110 | Assert Violation | Passed |
| SWC-109 | Uninitialized Storage Pointer | Passed |
| SWC-108 | State Variable Default Visibility | Passed |
| SWC-107 | Reentrancy | Passed |
| SWC-106 | Unprotected SELFDESTRUCT Instruction | Passed |
| SWC-105 | Unprotected Ether Withdrawal | Passed |
| SWC-104 | Unchecked Call Return Value | Failed |
| SWC-103 | Floating Pragma | Passed |
| SWC-102 | Outdated Compiler Version | Passed |

| | | |
|---|---|---|
| SWC-101 | Integer Overflow and Underflow | Passed |
| SWC-100 | Function Default Visibility | Passed |

## Call Graph

A call graph is a control-flow graph which represents calling relationships between subroutines in a smart contract. Each node represents a procedure, and each edge (f, g) indicates that procedure f call procedure g.

Link:

https://github.com/TokenRaven/Audits/blob/main/Corps%20D'elite/Corps%20Delite%20Call%20Graph.png

## Risk Summary



| Vulnerability level | Number of Issues | Issues resolved |
|---|---|---|
| 🔴 High-risk | 0 | 0 |
| 🟡 Medium-risk | 2 | 0 |
| 🟢 Low-risk | 4 | 0 |

## Conclusion

Token Raven has performed a combination of manual and software analysis of the smart contracts provided by the project team. Each of them has been analyzed for the most common and exploits, vulnerabilities, and manipulation attacks.

| | |
|---|---|
| **Audit verdict** | Passed |
| **Manual Analysis Risks Found** | 6 |
| **Centralization Risk** | Low (contract is renounced) |
| **SWC Issues** | 3 |
| **Number of high-risk issues** | 0 |
| **Issues Resolved** | 0 |



## Disclaimer

The audits provided by Token Raven provide projects with a comprehensive analysis regarding all possible vulnerabilities in their smart contracts. This includes for example, warnings, typos, unused functions & variables, compiler version, centralization check and any other hazard. As such our report and recommendations can be used to remedy vulnerabilities.

The Token Raven audit service is sure to improve the trustworthiness and safety of any legitimate project.

However, all the content provided in the Token Raven audit reports are for general information and should not be used as financial advice nor a reason to invest in a project. The Token Raven team is not responsible in the event of a fraud or scam

committed by any of our partners. We implore everyone to always do your own due diligence.

### In the event of a scam

If a project that we have KYC verified commits illegal or scam activities, it will be reported to the proper authorities (i.e., law enforcement etc.) After reporting the scam, we will conduct a thorough investigation into its activities and may release the KYC information to the public and on our website if it turns out that the reported project is confirmed to be a scam. This to prevent individuals with malicious intent from committing further frauds in the future.

To report a scam project, you can contact us on telegram and provide us with the following information:

- Presale link (if available)
- Smart contract address
- Proof of scam activities
- Audit and KYC certificate



## About

We provide your blockchain project with KYC, Audit & Consultancy services on a daily timeframe. Thus far we have assisted over 50 projects to launch successfully.

### Why Token Raven?

The cold hard logic of smart contracts allows them to facilitate the most secure and trustworthy kinds of transactions to date. This infallible ability to always execute the provided code does mean however, that the code should be free of errors to prevent security hazards.

Token Raven takes pride in the fact that our audits stand for a secure smart contract. Not only do we provide a seal of approval, before publishing any audit we

also provide developer teams with detailed insights as to how they can resolve potential security issues. The track record at Token Raven means that our audits provide legitimacy and security to any legitimate project.