Coinsult

Request your audit at coinsult.net

Advanced Manual
Smart Contract Audit

October 15, 2022

CoinsultAudits

info@coinsult.net

coinsult.net

Audit requested by

Undercity

0x7774d315c0809dbb44de281de8670a5a82e6db83

Table of Contents

Audit Summary

Project Name	Undercity
Website	https://www.undercity.fr/
Blockchain	Ethereum
Smart Contract Language	Solidity
Contract Address	0x7774d315c0809dbb44de281de8670a5a82e6db83
Audit Method	Static Analysis, Manual Review
Date of Audit	15 October 2022

This audit report has been prepared by Coinsult's experts at the request of the client. In this audit, the results of the static analysis and the manual code review will be presented. The purpose of the audit is to see if the functions work as intended, and to identify potential security issues within the smart contract.

The information in this report should be used to understand the risks associated with the smart contract. This report can be used as a guide for the development team on how the contract could possibly be improved by remediating the issues that were identified.

Audit Scope

Source Code

Coinsult was comissioned by Undercity to perform an audit based on the following code:

https://etherscan.io/address/0x7774d315c0809dbb44de281de8670a5a82e6db83#code

Note that we only audited the code available to us on this URL at the time of the audit. If the URL is not from any block explorer (main net), it may be subject to change. Always check the contract address on this audit report and compare it to the token you are doing research for.

Tokenomics

Rank	Address	Quantity (Token)	Percentage
1	0x7b0b6df7514e095a4dc8043c64d4febc7c187388	3,705,000	65.0000%
2	0xb8f24ee6d8e6f937c11465598b3fe5771a9951da	969,000	17.0000%
3	0x81ee8626949eea3abbc2fe97b573f44dfc6283a8	570,000	10.0000%
4	0x851fa25deb1b2d24750a3424e6033077136406a0	285,000	5.0000%
5	0x0dd2df837a369399ebeb9264c0af79f1febd3f34	171,000	3.0000%

Audit Method

Coinsult's manual smart contract audit is an extensive methodical examination and analysis of the smart contract's code that is used to interact with the blockchain. This process is conducted to discover errors, issues and security vulnerabilities in the code in order to suggest improvements and ways to fix them.

Automated Vulnerability Check

Coinsult uses software that checks for common vulnerability issues within smart contracts. We use automated tools that scan the contract for security vulnerabilities such as integer-overflow, integer-underflow, out-of-gas-situations, unchecked transfers, etc.

Manual Code Review

Coinsult's manual code review involves a human looking at source code, line by line, to find vulnerabilities. Manual code review helps to clarify the context of coding decisions. Automated tools are faster but they cannot take the developer's intentions and general business logic into consideration.

Used Tools

- Slither: Solidity static analysis framework
- Remix: IDE Developer Tool
- CWE: Common Weakness Enumeration
- SWC: Smart Contract Weakness Classification and Test Cases
- DEX: Testnet Blockchains

Risk Classification

Coinsult uses certain vulnerability levels, these indicate how bad a certain issue is. The higher the risk, the more strictly it is recommended to correct the error before using the contract.

Vulnerability Level	Description
Informational	Does not compromise the functionality of the contract in any way
Low-Risk	Won't cause any problems, but can be adjusted for improvement
Medium-Risk	Will likely cause problems and it is recommended to adjust
High-Risk	Will definitely cause problems, this needs to be adjusted

Coinsult has four statuses that are used for each risk level. Below we explain them briefly.

Risk Status	Description
Total	Total amount of issues within this category
Pending	Risks that have yet to be addressed by the team
Acknowledged	The team is aware of the risks but does not resolve them
Resolved	The team has resolved and remedied the risk

Disclaimer

This audit report has been prepared by Coinsult's experts at the request of the client. In this audit, the results of the static analysis and the manual code review will be presented. The purpose of the audit is to see if the functions work as intended, and to identify potential security issues within the smart contract.

The information in this report should be used to understand the risks associated with the smart contract. This report can be used as a guide for the development team on how the contract could possibly be improved by remediating the issues that were identified.

Coinsult is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

Coinsult is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your own research.

The information provided in this audit is for informational purposes only and should not be considered investment advice. Coinsult does not endorse, recommend, support or suggest to invest in any project.

Coinsult can not be held responsible for when a project turns out to be a rug-pull, honeypot or scam.

Global Overview

Manual Code Review

In this audit report we will highlight the following issues:

Vulnerability Level	Total	Pending	Acknowledged	Resolved
Informational	0	0	0	0
Low-Risk	4	0	2	2
Medium-Risk	0	0	0	0
High-Risk	0	0	0	0

Centralization Risks

Coinsult checked the following privileges:

Contract Privilege	Description
Owner can mint?	Owner cannot mint new tokens
Owner can blacklist?	Owner cannot blacklist addresses
Owner can set fees > 25%?	Owner cannot set the sell fee to 25% or higher
Owner can exclude from fees?	Owner cannot exclude from fees
Owner can pause trading?	Owner cannot pause the contract
Owner can set Max TX amount?	Owner cannot set max transaction amount

More owner priviliges are listed later in the report.

Error Code	Description
SWC-116	CWE-829: Inclusion of Functionality from Untrusted Control Sphere

🟢 **Low-Risk:** Could be fixed, will not bring problems.

Avoid relying on block.timestamp

block.timestamp can be manipulated by miners.

```
function _transfer(address from, address to, uint256 amount) internal override {
    require(from != address(0), "UNDER: Transfer from the zero address");
    require(to != address(0), "UNDER: Transfer to the zero address");
    require(amount &gt;= 0, "UNDER: Transfer amount must be greater or equals to zero");

    bool isBuyTransfer = _automatedMarketMakerPairs[from];

    if(coolDownEnabled &amp;&amp; !isBuyTransfer &amp;&amp; !_isExcludedFromCooldown[from]){
        uint256 timePassed = block.timestamp - _lastTimeTx[from];
        require(timePassed &gt;= coolDownTime, "UNDER: The cooldown is not finished, please ret
    }

    // Buy
    if(isBuyTransfer &amp;&amp; coolDownEnabled){
       _lastTimeTx[to] = block.timestamp;
    }
```

Recommendation

Do not use `block.timestamp`, `now` or `blockhash` as a source of randomness

Exploit scenario

```
contract Game {

    uint reward_determining_number;

    function guessing() external{
      reward_determining_number = uint256(block.blockhash(10000)) % 10;
    }
}
```

Eve is a miner. Eve calls `guessing` and re-orders the block containing the transaction. As a result, Eve wins the game.

Error Code	Description
SLT: 076	Costly operations in a loop

🟢 **Low-Risk:** Could be fixed, will not bring problems.

Costly operations inside a loop

Costly operations inside a loop might waste gas, so optimizations are justified.

```
// To distribute airdrops easily
function batchTokensTransfer(address[] calldata _holders, uint256[] calldata _amounts) external
    require(_holders.length &lt;= 200);
    require(_holders.length == _amounts.length);
        for (uint i = 0; i &lt; _holders.length; i++) {
          if (_holders[i] != address(0)) {
            super._transfer(_msgSender(), _holders[i], _amounts[i]);
        }
    }
}
```

Recommendation

Use a local variable to hold the loop computation result.

Exploit scenario

```
contract CostlyOperationsInLoop{

    function bad() external{
        for (uint i=0; i < loop_count; i++){
            state_variable++;
        }
    }

    function good() external{
      uint local_variable = state_variable;
      for (uint i=0; i < loop_count; i++){
        local_variable++;
      }
      state_variable = local_variable;
    }
}
```

Incrementing `state_variable` in a loop incurs a lot of gas because of expensive `SSTORE`s, which might lead to an `out-of-gas`.

Maximum Fee Limit Check

Error Code	Description
CEN-01	Centralization: Operator Fee Manipulation

Coinsult tests if the owner of the smart contract can set the transfer, buy or sell fee to 25% or more. It is bad practice to set the fees to 25% or more, because owners can prevent healthy trading or even stop trading when the fees are set too high.

Type of fee	Description
Transfer fee	🟢 Owner cannot set the transfer fee to 25% or higher
Buy fee	🟢 Owner cannot set the buy fee to 25% or higher
Sell fee	🟢 Owner cannot set the sell fee to 25% or higher

Type of fee	Description
Max transfer fee	0%
Max buy fee	0%
Max sell fee	0%

Contract Pausability Check

Error Code	Description
CEN-02	Centralization: Operator Pausability

Coinsult tests if the owner of the smart contract has the ability to pause the contract. If this is the case, users can no longer interact with the smart contract; users can no longer trade the token.

Privilege Check	Description
Can owner pause the contract?	🟢 Owner cannot pause the contract

Max Transaction Amount Check

Error Code	Description
CEN-03	Centralization: Operator Transaction Manipulation

Coinsult tests if the owner of the smart contract can set the maximum amount of a transaction. If the transaction exceeds this limit, the transaction will revert. Owners could prevent normal transactions to take place if they abuse this function.

Privilege Check	Description
Can owner set max tx amount?	🟢 Owner cannot set max transaction amount

Exclude From Fees Check

Error Code	Description
CEN-04	Centralization: Operator Exclusion

Coinsult tests if the owner of the smart contract can exclude addresses from paying tax fees. If the owner of the smart contract can exclude from fees, they could set high tax fees and exclude themselves from fees and benefit from 0% trading fees. However, some smart contracts require this function to exclude routers, dex, cex or other contracts / wallets from fees.

Privilege Check	Description
Can owner exclude from fees?	🟢 Owner cannot exclude from fees

Ability To Mint Check

Error Code	Description
CEN-05	Centralization: Operator Increase Supply

Coinsult tests if the owner of the smart contract can mint new tokens. If the contract contains a mint function, we refer to the token's total supply as non-fixed, allowing the token owner to "mint" more tokens whenever they want.

A mint function in the smart contract allows minting tokens at a later stage. A method to disable minting can also be added to stop the minting process irreversibly.

Minting tokens is done by sending a transaction that creates new tokens inside of the token smart contract. With the help of the smart contract function, an unlimited number of tokens can be created without spending additional energy or money.

Privilege Check	Description
Can owner mint?	🟢 Owner cannot mint new tokens

Ability To Blacklist Check

Error Code	Description
CEN-06	Centralization: Operator Dissalows Wallets

Coinsult tests if the owner of the smart contract can blacklist accounts from interacting with the smart contract. Blacklisting methods allow the contract owner to enter wallet addresses which are not allowed to interact with the smart contract.

This method can be abused by token owners to prevent certain / all holders from trading the token. However, blacklists might be good for tokens that want to rule out certain addresses from interacting with a smart contract.

Privilege Check	Description
Can owner blacklist?	🟢 Owner cannot blacklist addresses

Other Owner Privileges Check

Error Code	Description
CEN-100	Centralization: Operator Priviliges

Coinsult lists all important contract methods which the owner can interact with.

⚠️ Owner can set sell and transfer cooldown to maximum 600 seconds

⚠️ Owner can exclude from cooldowns

Notes

Notes by Undercity

No notes provided by the team.

Notes by Coinsult

No notes provided by Coinsult

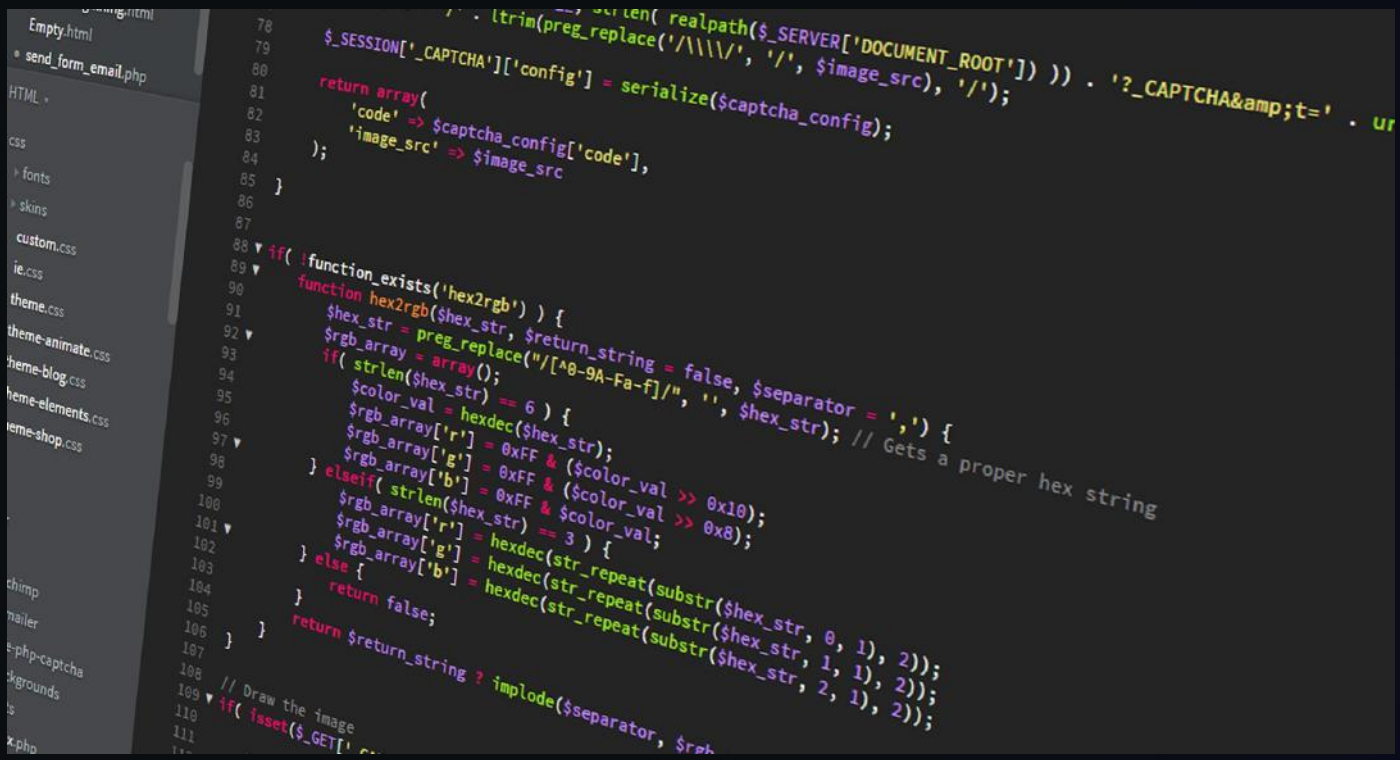
Contract Snapshot

This is how the constructor of the contract looked at the time of auditing the smart contract.

```
contract Undercity is Context, Ownable, ERC20  {

using Address for address payable;

mapping (address =&gt; bool) private _isExcludedFromCooldown;

address constant private _PRESALE_WALLET = 0x7B0b6dF7514E095a4dc8043C64D4fEBC7c187388;
address constant private _RESERVE_WALLET = 0xb8f24EE6d8e6f937C11465598b3fE5771A9951DA;
address constant private _TEAM_WALLET = 0x81ee8626949EEA3aBbc2fE97B573F44dFC6283a8;
address constant private _MARKETING_WALLET = 0x851FA25DEB1B2D24750A3424e6033077136406a0;
address constant private _AIRDROP_WALLET = 0x0dD2df837a369399eBEb9264C0af79F1feBd3f34;
address constant private _DEAD = 0x000000000000000000000000000000000000dEaD;

// CoolDown system
mapping(address =&gt; uint256) private _lastTimeTx;
bool public coolDownEnabled = true;
uint32 public coolDownTime = 60 seconds;

// Routers and Factories
IUniswapV2Router02 constant private UNISWAPV2_ROUTER = IUniswapV2Router02(0x7a250d5630B4cF53973

IUniswapV3Factory constant private UNISWAPV3_FACTORY = IUniswapV3Factory(0x1F98431c8aD98523631A
address constant private USDT = 0xdAC17F958D2ee523a2206206994597C13D831ec7;
address constant private WETH = 0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2;

// Any transfer to these addresses could be subject to some cooldown
mapping (address =&gt; bool) private _automatedMarketMakerPairs;

event ExcludeFromCooldown(address indexed account, bool isExcluded);

event AddAutomatedMarketMakerPair(address indexed pair, bool indexed value);

event Burn(uint256 amount);

event CoolDownUpdated(bool state,uint32 timeInSeconds);

constructor() ERC20("Undercity", "UNDER") {
```

Website Review

Coinsult checks the website completely manually and looks for visual, technical and textual errors. We also look at the security, speed and accessibility of the website. In short, a complete check to see if the website meets the current standard of the web development industry.



Not live yet

Type of check	Description
Mobile friendly?	● The website is mobile friendly
Contains jQuery errors?	● The website does not contain jQuery errors
Is SSL secured?	● The website is SSL secured
Contains spelling errors?	● The website does not contain spelling errors

Certificate of Proof

Not KYC verified by Coinsult

Undercity

Audited by Coinsult.net

Date: 15 October 2022

✔ Advanced Manual Smart Contract Audit

Coinsult

coinsult.net

End of report
Smart Contract Audit

CoinsultAudits

info@coinsult.net

coinsult.net