

Advanced Manual **Smart Contract Audit**

September 23, 2022

Audit requested by



UniWswap

Table of Contents

1. Audit Summary

- 1.1 Audit scope
- 1.2 Tokenomics
- 1.3 Source Code

2. Disclaimer

3. Global Overview

- 3.1 Informational issues
- 3.2 Low-risk issues
- 3.3 Medium-risk issues
- 3.4 High-risk issues

4. Vulnerabilities Findings

5. Contract Privileges

- 5.1 Maximum Fee Limit Check
- 5.2 Contract Pausability Check
- 5.3 Max Transaction Amount Check
- 5.4 Exclude From Fees Check
- 5.5 Ability to Mint Check
- 5.6 Ability to Blacklist Check
- 5.7 Owner Privileges Check

6. Notes

- 6.1 Notes by Coinsult
- 6.2 Notes by UniWswap

7. Contract Snapshot

8. Website Review

9. Certificate of Proof

Audit Summary

Audit Scope

| | |
|-------------------------|---|
| Project Name | UniWswap |
| Website | https://uniwswap.com/ |
| Blockchain | Ethereum PoW |
| Smart Contract Language | Solidity |
| Contract Address | - |
| Audit Method | Static Analysis, Manual Review |
| Date of Audit | 23 September 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.

Tokenomics

Not available

Source Code

Coinsult was commissioned by UniWswap to perform an audit based on the following code:

<https://github.com/uniwswap/uniw-contracts/blob/main/token/uniw.sol>

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 | 4 | 0 |
| ● Medium-Risk | 0 | 0 | 0 | 0 |
| ● High-Risk | 0 | 0 | 0 | 0 |

Privilege Overview

Coinsult checked the following privileges:

| Contract Privilege | Description |
|------------------------------|--|
| Owner can mint? | ● Owner can 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 can 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 privileges are listed later in the report.

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

Change boolean name


```
function updateFee(
    bool feeType,
    uint256 fee
) external onlyOwner {
    require(fee <= 1000,
        "Bork: Fee cannot be more than 10%");

    if (feeType) { // TRUE == BUY FEE
        BUY_FEE = fee;
    } else { // FALSE == SELL FEE
        SELL_FEE = fee;
    }

    emit FeeUpdated(_msgSender(), feeType, fee);
}
```

Recommendation

For readability it would be better to change 'feeType' to 'isBuy'. True for buys, false for sells.

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

Too many digits

Literals with many digits are difficult to read and review.

```
uint256 feeAmount = amount.mul(fee).div(10000);
```

Recommendation

Use: Ether suffix, Time suffix, or The scientific notation

Exploit scenario

```
contract MyContract{
    uint 1_ether = 1000000000000000000;
}
```

While `1_ether` looks like 1 ether, it is 10 ether. As a result, it's likely to be used incorrectly.

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

No zero address validation for some functions

Detect missing zero address validation.

```
function updateDevAddress(
    address payable _dev
) external onlyOwner {
    isExcludedFromFee[developmentAddress] = false;
    emit AddressExcluded(_msgSender(), developmentAddress, false);

    developmentAddress = _dev;
    isExcludedFromFee[developmentAddress] = true;

    emit AddressExcluded(_msgSender(), developmentAddress, true);
}
```

Recommendation

Check that the new address is not zero.

Exploit scenario

```
contract C {

    modifier onlyAdmin {
        if (msg.sender != owner) throw;
        _;
    }

    function updateOwner(address newOwner) onlyAdmin external {
        owner = newOwner;
    }
}
```

Bob calls updateOwner without specifying the newOwner, so Bob loses ownership of the contract.

● **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.

```
function excludeMultipleAccountsFromFees(
    address[] calldata _accounts,
    bool _excluded
) external onlyOwner {
    for (uint256 i = 0; i < _accounts.length; i++) {
        isExcludedFromFee[_accounts[i]] = _excluded;

        emit AddressExcluded(_msgSender(), _accounts[i], _excluded);
    }
}
```

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 `SSTOREs`, which might lead to an out-of-gas.

Contract Privileges

Maximum Fee Limit Check


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 | 10% |
| Max buy fee | 10% |
| Max sell fee | 10% |


Contract Pausability Check

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

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

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 can exclude from fees |

Ability To Mint Check

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 can mint new tokens |

Ability To Blacklist Check

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

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

✔ No other important owner privileges to mention.

Notes

Notes by UniWswap

No notes provided by the team.

Notes by Coinconsult

 No notes provided by Coinconsult

Contract Snapshot

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

```
contract UniW is ERC20Burnable, Ownable {
    using SafeMath for uint256;

    mapping(address => bool) public isExcludedFromFee;
    mapping(address => bool) public isMinter;
    mapping(address => bool) public whiteListedPair;

    uint256 immutable public MAX_SUPPLY;
    uint256 public BUY_FEE = 0;
    uint256 public SELL_FEE = 500;

    address payable public developmentAddress;

    event TokenRecoverd(address indexed _user, uint256 _amount);
    event FeeUpdated(address indexed _user, bool _feeType, uint256 _fee);
    event ToggleV2Pair(address indexed _user, address indexed _pair, bool _flag);
    event AddressExcluded(address indexed _user, address indexed _account, bool _flag);
    event MinterRoleAssigned(address indexed _user, address indexed _account);
    event MinterRoleRevoked(address indexed _user, address indexed _account);

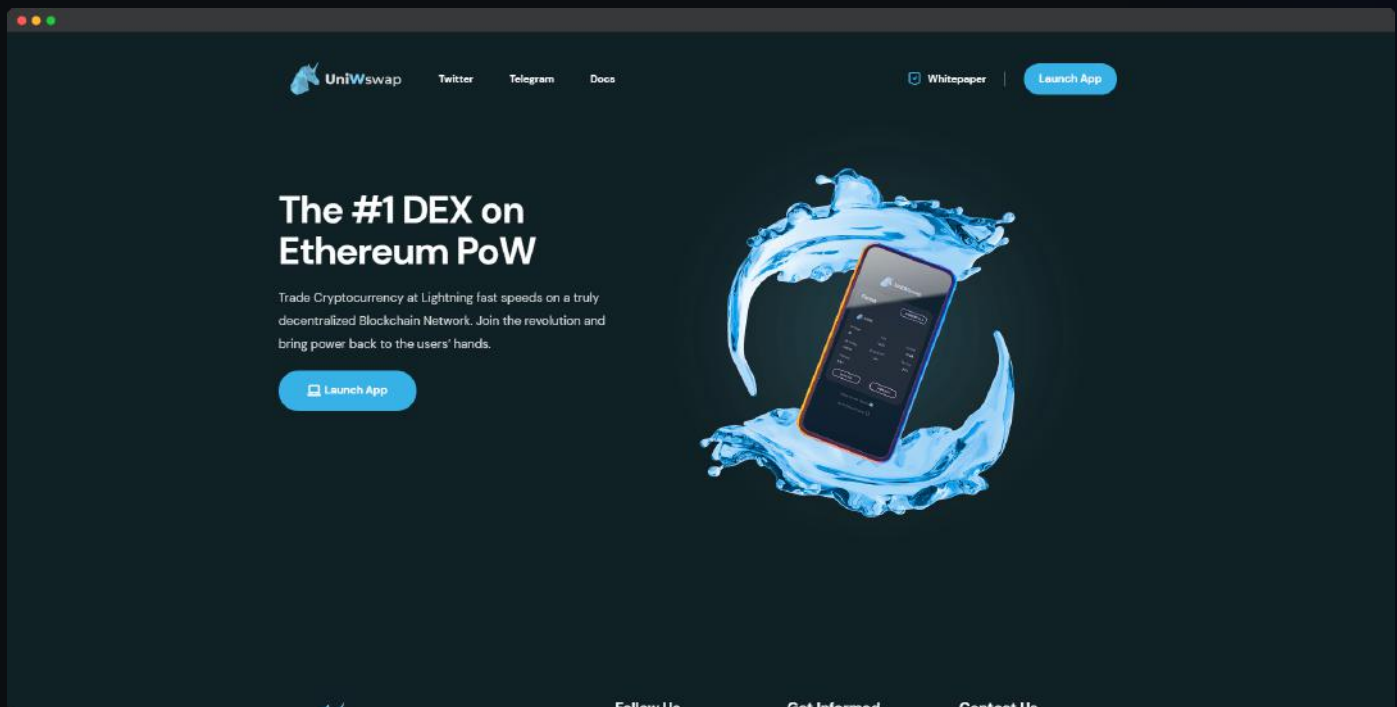
    constructor(string memory __name, string memory __symbol, uint256 _maxSupply, uint256 _initialSupply) E

        require(_initialSupply > 0) {
            _mint(_msgSender(), _initialSupply);
        }
    }

    modifier hasMinterRole () {
        require(isMinter[_msgSender()],
            "UNIW: Permission Denied!!!");
        _;
    }
}
```

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.



| 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

UniWswap

Audited by Coinsult.net



Date: 23 September 2022

✓ Advanced Manual Smart Contract Audit

End of report **Smart Contract Audit**

Request your smart contract audit / KYC

t.me/coinsult_tg