Coinsult

Advanced Manual Smart Contract Audit

Project: Umbrella Protocol

Website: https://umbprotocol.com

Low-Risk	Medium-Risk	High-Risk
8 low-risk code issues found	2 medium-risk code issues found	0 high-risk code issues found

Contract Address

0x443fA16a2Ee89983A9558Ab93C85B67e2afe92Af

Disclaimer

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.

Tokenomics

Rank	Address	Quantity (Token)	Percentage
1	Null Address: 0x000...dEaD	650,000,000,000,000	65.0000%
2	0xb2bf5d5f19ab43a7902d36ef5069e8b6ca1b7354	220,000,000,000,000	22.0000%
3	0x7947d1fad96206c51c33c163fcb38983e1e1e4ea	50,000,000,000,000	5.0000%
4	0xd24959b67a4d708cc13d2581bb204a55bb2fb10d	50,000,000,000,000	5.0000%
5	0x3c0ed09a087f453e1c51aa400d0d5c41582f02d4	15,000,000,000,000	1.5000%

Source Code

Coinsult was comissioned by Umbrella Protocol to perform an audit based on the following smart contract:

https://bscscan.com/address/0x443fA16a2Ee89983A9558Ab93C85B67e2afe92Af#code

Manual Code Review

In this audit report we will highlight all these issues:

🟢 Low-Risk

8 low-risk code
issues found

🟡 Medium-Risk

2 medium-risk code
issues found

🔴 High-Risk

0 high-risk code
issues found

The detailed report continues on the next page...

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

Contract contains Reentrancy vulnerabilities

Additional information: This combination increases risk of malicious intent. While it may be justified by some complex mechanics (e.g. rebase, reflections, buyback).

More information: Slither

```
function _transfer(
    address from,
    address to,
    uint256 amount
) internal override {

    require(from != address(0), "ERC20: transfer from the zero address");
    require(to != address(0), "ERC20: transfer to the zero address");
    require(!_isBlacklisted[from], 'Blacklisted address');

    if(amount == 0) {
        super._transfer(from, to, 0);
        return;
    }

    // No adding liquidity before launched
    if (!liquidityLaunched) {
        if (to == uniswapV2Pair) {
            liquidityLaunched = true;
            // high tax ends in x blocks
            lastSnipeTaxBlock = block.number + snipeBlocks;
        }
```

Recommendation

Apply the check-effects-interactions pattern.

Exploit scenario

```
function withdrawBalance(){
    // send userBalance[msg.sender] Ether to msg.sender
    // if mgs.sender is a contract, it will call its fallback function
    if( ! (msg.sender.call.value(userBalance[msg.sender])() ) ){
        throw;
    }
    userBalance[msg.sender] = 0;
}
```

Bob uses the re-entrancy bug to call withdrawBalance two times, and withdraw more than its initial deposit to the contract.

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

Avoid relying on block.timestamp

block.timestamp can be manipulated by miners.

```
// make the swap
uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens(
    tokenAmount,
    0, // accept any amount of ETH
    path,
    address(this),
    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.

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

Too many digits

Literals with many digits are difficult to read and review.

```
// use by default 300,000 gas to process auto-claiming dividends
uint256 public gasForProcessing = 300000;
```

Recommendation

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

Exploit scenario

```
contract MyContract{
    uint 1_ether = 10000000000000000000;
}
```

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 setDevWallet(address payable wallet) external onlyOwner{
    _DevWalletAddress = wallet;
}

function setPresaleContract(address payable wallet) external onlyOwner{
    _PresaleAddress = wallet;
    excludeFromFees(_PresaleAddress, true);
}

function setFoudWalletAddress(address payable wallet) external onlyOwner{
    _FoudWalletAddress = wallet;
}
```

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`, soBob loses ownership of the contract.

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

Functions that send Ether to arbitrary destinations

Unprotected call to a function sending Ether to an arbitrary address.

```
// add the liquidity
uniswapV2Router.addLiquidityETH{value: ethAmount}(
    address(this),
    tokenAmount,
    0, // slippage is unavoidable
    0, // slippage is unavoidable
    address(0),
    block.timestamp
);
```

Recommendation

Ensure that an arbitrary user cannot withdraw unauthorized funds.

Exploit scenario

```
contract ArbitrarySend{
    address destination;
    function setDestination(){
        destination = msg.sender;
    }

    function withdraw() public{
        destination.transfer(this.balance);
    }
}
```

Bob calls `setDestination` and `withdraw`. As a result he withdraws the contract's balance.

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

Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
function setFee(
    uint256 _USDTRewardsFee,
    uint256 _USDTRewardsShare,
    uint256 _LPShare
) public onlyOwner {
    USDTRewardsFee = _USDTRewardsFee;
    USDTRewardsShare = _USDTRewardsShare;
    LPShare = _LPShare;
}
```

Recommendation

Emit an event for critical parameter changes.

Exploit scenario

```
contract C {

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

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

`updateOwner()` has no event, so it is difficult to track off-chain changes in the buy price.

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

Conformance to Solidity naming conventions

Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
address public _PresaleAddress = 0x000000000000000000000000000000000000dEaD;
```

Recommendation

Follow the Solidity naming convention.

Rule exceptions

- Allow constant variable name/symbol/decimals to be lowercase (`ERC20`).
- Allow _ at the beginning of the `mixed_case` match for private variables and unused parameters.

🟢 **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.

```
if(canAutoClaim(lastClaimTimes[account])) {
    if(processAccount(payable(account), true)) {
        claims++;
    }
}

iterations++;

uint256 newGasLeft = gasleft();
```

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`.

🟡 **Medium-Risk:** Should be fixed, could bring problems.

Owner can mint new tokens

```
function setBalance(address payable account, uint256 newBalance) external onlyOwner {
    if(excludedFromDividends[account]) {
        return;
    }

    if(newBalance &gt;= minimumTokenBalanceForDividends) {
        _setBalance(account, newBalance);
        tokenHoldersMap.set(account, newBalance);
    }
    else {
```

Recommendation

No recommendation

🟡 **Medium-Risk:** Should be fixed, could bring problems.

Owner can blacklist all addresses

```
function blacklistAddress(address account, bool value) external onlyOwner{
    _isBlacklisted[account] = value;
}
```

Recommendation

Consider the option to only blacklist contract addresses or use anti-bot methods for better transparancy

Owner privileges

- Owner cannot pause trading
- Owner cannot change max transaction amount
- Owner can set fees higher than 25%
- Owner can exclude from fees
- Owner can mint new tokens
- Owner can blacklist addresses

Sell fees are 26%

Extra notes by the team

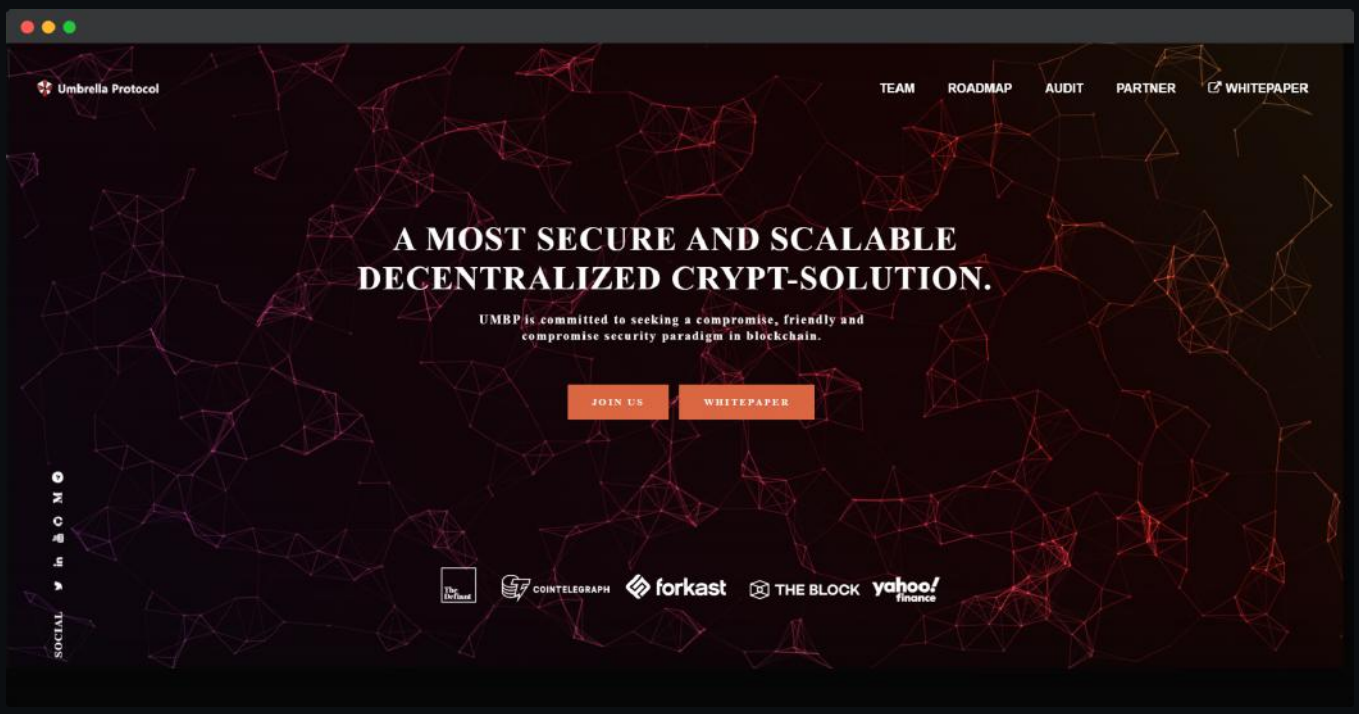
No notes

Contract Snapshot

```
contract UmbrellaProtocol is ERC20, Ownable {
using SafeMath for uint256;

IUniswapV2Router02 public uniswapV2Router;
address public  uniswapV2Pair;

bool private swapping;

USDTDividendTracker public dividendTracker;

address public deadWallet = 0x000000000000000000000000000000000000dEaD;
address public _DevWalletAddress = 0x3C0ed09a087f453e1c51aA400D0d5c41582F02d4;
address public _FoudWalletAddress = 0x64552ed8AAfe252Bee763fbe007FD21D7c93EDd8;

address public immutable USDT = address(0x55d398326f99059fF775485246999027B3197955);

uint256 public totalSupply_ = 1000 * (10**12) * (10**18);

uint256 public swapTokensAtAmount;

mapping(address =&gt; bool) public _isBlacklisted;

uint256 public USDTRewardsFee = 10;
uint256 public USDTRewardsShare = 4;
uint256 public LPShare = 1;

// anti bot
address public _PresaleAddress = 0x000000000000000000000000000000000000dEaD;
bool public liquidityLaunched = false; // to track if launchLiquidity function has been called
bool public isFirstLaunch = true; // to track if launchLiquidity function has been called
uint256 public lastSnipeTaxBlock; // set to blocks after liq added
uint8 public snipeBlocks = 0;

// use by default 300,000 gas to process auto-claiming dividends
uint256 public gasForProcessing = 300000;

 // exlcude from fees and max transaction amount
```

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.



- Mobile Friendly
- Does not contain jQuery errors
- SSL Secured
- No major spelling errors

Project Overview

Not KYC verified by Coinsult