



Coinsult

Advanced Manual Smart Contract Audit



Project: GoldShield

Website: <http://goldshield.io>

Low-risk

5 low-risk code
issues found

Medium-risk

0 medium-risk code
issues found

High-risk

0 high-risk code
issues found

Contract address

TBA

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

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

TBA

Source code

Coinsult was commissioned by GoldShield to perform an audit based on the following smart contract:

TBA

Manual Code Review

● Low-risk

5 low-risk code issues found.

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) private
{
    require(from != address(0), "ERC20: transfer from the zero
address");
    require(to != address(0), "ERC20: transfer to the zero
address");
    require(amount > 0, "Transfer amount must be greater than
zero");

    if(!_isExcludedFromFee[from] && !_isExcludedFromFee[to] &&
antiBotEnabled){
        applyAntiBot(from, to);
    }

    uint256 contractTokenBalance = balanceOf(address(this));

    bool canSwap = contractTokenBalance >= swapThreshold;
    if (canSwap && !inSwapAndLiquify && from != pair &&
swapAndLiquifyEnabled) {
        swapAndLiquify(swapThreshold);
    }

    bool takeFee = true;

    if(_isExcludedFromFee[from] || _isExcludedFromFee[to]){
        takeFee = false;
    }

    bool isSale = false;
```

```

        if(to == pair) isSale = true;

        _tokenTransfer(from,to,amount,takeFee, isSale);
    }

```

- Missing events for critical arithmetic parameters
Emit an event for critical parameter changes.

```

function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
    _taxFee = taxFee;
    _previousTaxFee = taxFee;
}

function setMarketingFeePercent(uint256 marketingFee) external
onlyOwner() {
    _marketingFee = marketingFee;
    _previousMarketingFee = marketingFee;
}

function setLiquidityFeePercent(uint256 liquidityFee) external
onlyOwner() {
    _liquidityFee = liquidityFee;
    _previousLiquidityFee = liquidityFee;
}

function setSellFees(uint256 sellTax, uint256 sellMarketing,
uint256 sellLiquidity) external onlyOwner{
    sellTaxFee = sellTax;
    sellMarketingFee = sellMarketing;
    sellLiquidityFee = sellLiquidity;
}

```

- Block.timestamp can be manipulated by miners.
Avoid relying on block.timestamp.

More information:

<https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp>

```
function applyAntiBot(address from, address to) internal {
    bool fromCanTrade = (block.timestamp - _lastTrade[from]) > 60
seconds ;
    bool toCanTrade = (block.timestamp - _lastTrade[to]) > 60
seconds;
    if(from != pair && to != pair){
        require(fromCanTrade && toCanTrade, "You must wait 60
seconds between transactions");
        _lastTrade[from] = _lastTrade[to] = block.timestamp;
    }

    else if(from == pair) {
        require(toCanTrade, "You must wait 60 seconds between
transactions");
        _lastTrade[to] = block.timestamp;
    }
    else if(to == pair){
        require(fromCanTrade, "You must wait 60 seconds between
transactions");
        _lastTrade[from] = block.timestamp;
    }
}
```

- Missing zero address validation
Check that the new address is not zero.

```
function setMarketingWallet(address _marketingWallet) external  
onlyOwner() {  
    marketingWallet = _marketingWallet;  
    _isExcludedFromFee[marketingWallet] = true;  
}
```

- Hardcoded decimals
Keep in mind that entered value will always be 10^{**9} when entering swapthreshold

```
function setSwapThreshold(uint256 amount) external onlyOwner{  
    swapThreshold = amount * 10**9;  
}
```

● **Medium-risk**

0 medium-risk code issues found.

Should be fixed, could bring problems.

● **High-risk**

0 high-risk code issues found

Must be fixed, and will bring problems.

Extra notes by the team

● Fees can be set higher than 25%

```
function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
    _taxFee = taxFee;
    _previousTaxFee = taxFee;
}

function setMarketingFeePercent(uint256 marketingFee) external
onlyOwner() {
    _marketingFee = marketingFee;
    _previousMarketingFee = marketingFee;
}

function setLiquidityFeePercent(uint256 liquidityFee) external
onlyOwner() {
    _liquidityFee = liquidityFee;
    _previousLiquidityFee = liquidityFee;
}

function setSellFees(uint256 sellTax, uint256 sellMarketing,
uint256 sellLiquidity) external onlyOwner{
    sellTaxFee = sellTax;
    sellMarketingFee = sellMarketing;
    sellLiquidityFee = sellLiquidity;
}
```

● Owner can exclude from fees

● The ownership of the contract isn't renounced

● Owner can set swap threshold without a limit

```
function setSwapThreshold(uint256 amount) external onlyOwner{
    swapThreshold = amount * 10**9;
}
```


Contract Snapshot

```
contract goldshield is Context, IERC20, Ownable {

    address public immutable deadAddress =
0x0000000000000000000000000000000000000000dEaD;

    mapping (address => uint256) private _rOwned;
    mapping (address => uint256) private _tOwned;
    mapping (address => mapping (address => uint256)) private
_allowances;

    mapping (address => bool) private _isExcludedFromFee;

    mapping (address => bool) private _isExcluded;
    address[] private _excluded;

    mapping(address => uint256) _lastTrade;

    uint256 private constant MAX = ~uint256(0);
    uint256 private _tTotal = 1e11 * 10**9;
    uint256 private _rTotal = (MAX - (MAX % _tTotal));
    uint256 private _tFeeTotal;

    string private _name = "GoldShield";
    string private _symbol = "GSHIELD";
    uint8 private _decimals = 9;

    uint256 public _taxFee = 8;
    uint256 public _liquidityFee = 1;
    uint256 public _marketingFee = 1;

    uint256 public sellTaxFee = 10;
    uint256 public sellLiquidityFee = 1;
    uint256 public sellMarketingFee = 1;
```

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
- Contains no jQuery errors
- SSL Secured
- No major spelling errors

Loading speed: 87%

Rug-pull Review

Based on the available information analyzed by us, we come to the following conclusions:

- Locked Liquidity (no liquidity yet)
- Large unlocked wallets (tokens not distributed yet)
- No doxxed Team (yet)

Honeypot Review

Based on the available information analyzed by us, we come to the following conclusions:

- Ability to sell
 - Owner can set swap threshold without a limit
- Owner is not able to pause the contract
 - Owner can set swap threshold without a limit
- Router can be changed

Note: Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by the project owner.