# Coinsult

# Advanced Manual Smart Contract Audit

**Project:** T10 Yield Fox
**Website:** https://yieldfox.co/

🟢 **Low-risk**

4 low-risk code issues found

🟡 **Medium-risk**

1 medium-risk code issues found

🔴 **High-risk**

0 high-risk code issues found

**Contract address**

0x238B49b0751dc14fE290620289fA318A1953d79F

# Disclaimer

# Tokenomics

| Rank | Address | Quantity (Token) | Percentage |
|------|---------|------------------|------------|
| 1 | 0x05a84f182b8b24067d4cd0d1d392a3043a1d34a7 | 345,510.43959 | 68.4953% |
| 2 | 0x881136965206899d25df1b196275c1f7e9a74dc0 | 158,918.66518 | 31.5047% |

# Source code

Coinsult was commissioned by T10 Yield Fox to perform an audit based on the following smart contract:

https://bscscan.com/address/0x238B49b0751dc14fE290620289fA318A19 53d79F#code

# Manual Code Review

## 🟢 Low-risk

4 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

```solidity
    function transferFrom(
        address from,
        address to,
        uint256 value
    ) external returns (bool);

    event Transfer(address indexed from, address indexed to, uint256 value);

    event Approval(
        address indexed owner,
        address indexed spender,
        uint256 value
    );
}
```

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

```solidity
_initRebaseStartTime = block.timestamp;
        _lastRebasedTime = block.timestamp;
```

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

```solidity
    function setFeeReceivers(
        address _autoLiquidityReceiver,
```

- Do not use write after write for the same variable usage

  More information: Slither

```
(bool success, ) = payable(treasuryReceiver).call{
        value: amountETHToTreasuryAndSIF.mul(treasuryFee).div(
            treasuryFee.add(clockInsuranceFundFee)
        ),
        gas: 30000
    }("");
    (success, ) = payable(clockInsuranceFundReceiver).call{
        value:
amountETHToTreasuryAndSIF.mul(clockInsuranceFundFee).div(
            treasuryFee.add(clockInsuranceFundFee)
        ),
        gas: 30000
    }("");
}
```

## ● Medium-risk
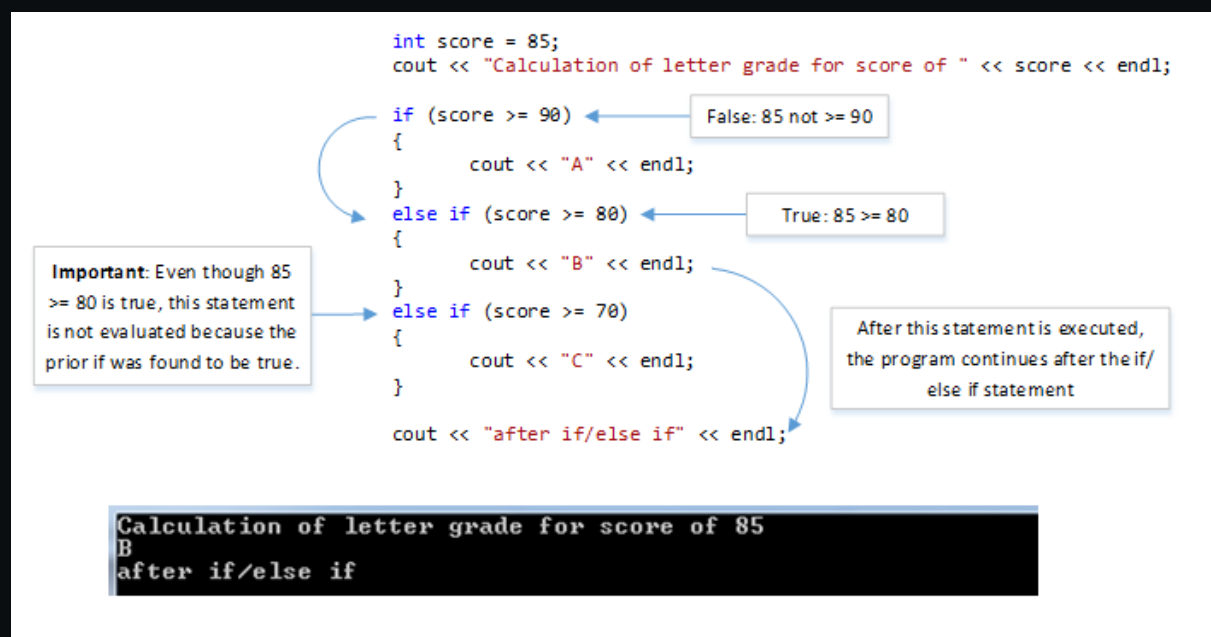
1 medium-risk code issues found.
Should be fixed, could bring problems.

- If statements might never be reached
  If the second statement is reached (>= 365 days), then the other two below will not be called upon.

```
if (deltaTimeFromInit < (365 days)) {
    rebaseRate = 1800;
} else if (deltaTimeFromInit >= (7 * 365 days)) {
    rebaseRate = 2;
} else if (deltaTimeFromInit >= ((15 * 365 days) / 10)) {
    rebaseRate = 18;
} else if (deltaTimeFromInit >= (365 days)) {
    rebaseRate = 180;
}
```



## ● High-risk

0 high-risk code issues found
Must be fixed, and will bring problems.

## Extra notes by the team

🟢 Use of anti bot smart contract

🟡 The ownership of the contract isn't renounced.

🟡 Owner can exclude addresses from fees.

🟡 Owner can change the router address

🟡 Fees can be set up to 100% for both buy and sell fees.

🔴 Owner is able to blacklist contract addresses

# Contract Snapshot

```solidity
contract T10 is ERC20Detailed, Ownable {

    using SafeMath for uint256;
    using SafeMathInt for int256;
    IPinkAntiBot public pinkAntiBot;
    bool public antiBotEnabled;

    event LogRebase(uint256 indexed epoch, uint256 totalSupply);

    string public _name = "T10";
    string public _symbol = "T10";
    uint8 public _decimals = 5;

    IPancakeSwapPair public pairContract;
    mapping(address => bool) _isFeeExempt;

    modifier validRecipient(address to) {
        require(to != address(0x0));
        _;
    }

    uint256 public constant DECIMALS = 5;
    uint256 public constant MAX_UINT256 = ~uint256(0);
    uint8 public constant RATE_DECIMALS = 7;

    uint256 private constant INITIAL_FRAGMENTS_SUPPLY =
        500 * 10**3 * 10**DECIMALS;

    uint256 public liquidityFee = 20;
    uint256 public treasuryFee = 20;
    uint256 public clockInsuranceFundFee = 10;
    uint256 public sellFee1 = 5;
    uint256 public sellFeet = 5;
    uint256 public sellFees = 5;
    uint256 public sellFeef = 5;
    uint256 public firePitFee = 10;
    uint256 public totalFee =
        liquidityFee.add(treasuryFee).add(clockInsuranceFundFee).add(
            firePitFee
        );
```
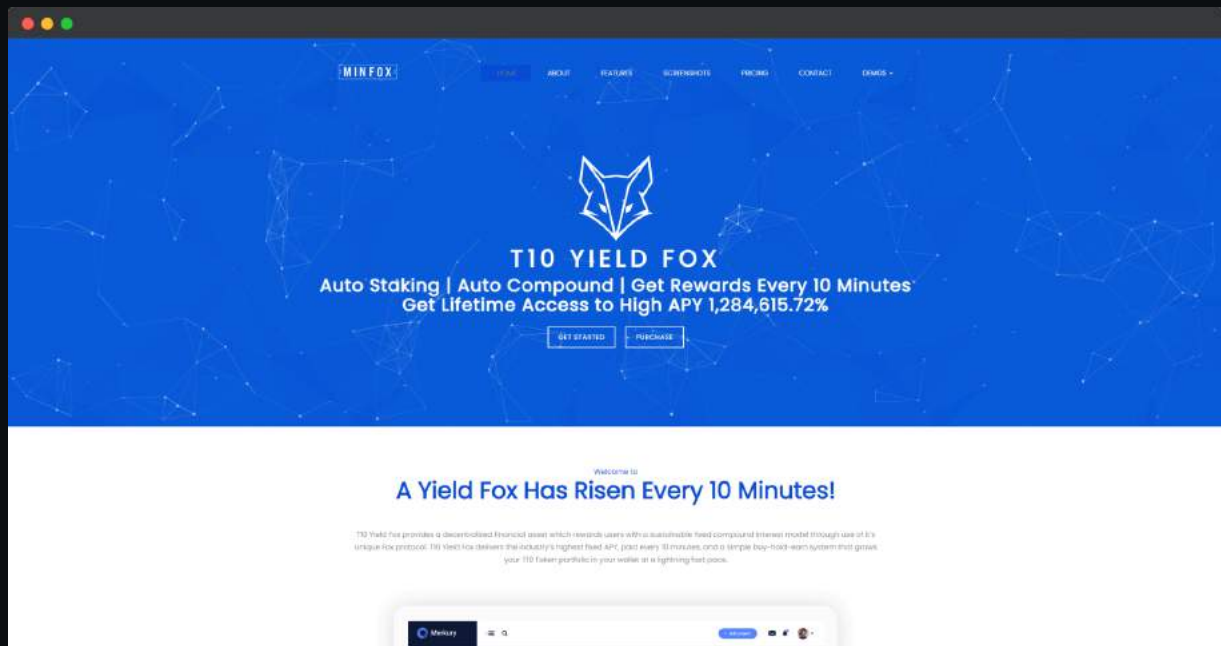
# Website Review



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

**Note: website has some small design problems. For example there is no favicon and on the bottom of the website there are multiple home links. (website is still under development)**

Loading speed: 74%

# 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
- Note: Tokens not distributed yet

🔴 No doxxed Team

# Honeypot Review

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

🟢 Ability to sell

🟢 Owner is not able to pause the contract

🟡 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.