Coinsult

Advanced Manual Smart Contract Audit

Project: FitApe

Website: https://fitape.tech

Low-Risk	Medium-Risk	High-Risk
7 low-risk code issues found	1 medium-risk code issues found	0 high-risk code issues found

Contract Address

0x63a472ec95527dcb79A3aA23170b9ece00A8Ad64

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	9,000,000,000,000,000	90.0000%
2	0x1abeec887522ad33ee20c4f479886ef42b84ca7b	900,000,000,000,000	9.0000%
3	0x33873d45e47ff9c5994ea2635ef6744487b58011	100,000,000,000,000	1.0000%

Source Code

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

https://bscscan.com/address/0x63a472ec95527dcb79A3aA23170b9ece00A8Ad64#code

Manual Code Review

In this audit report we will highlight all these issues:

🟢 Low-Risk

7 low-risk code
issues found

🟡 Medium-Risk

1 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) private {
require(from != address(0), "ERC20: transfer from the zero address");
require(to != address(0), "ERC20: transfer to the zero address");
require(amount &gt; 0, "Transfer amount must be greater than zero");
require(!_isBlacklisted[from], 'Blacklisted address');

_redisFee = 0;
_taxFee = 0;

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

//antibot block
if (from != address(_PresaleAddress)) {
    if(liquidityLaunched &amp;amp;&amp;amp; block.number  0) {
```

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.

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

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.

No zero address validation for some functions

Detect missing zero address validation.

```
event appAddressUpdated(address indexed previous, address indexed adr);
function setNewAppAddress(address payable appaddr) public onlyOwner {
    emit appAddressUpdated(_appAddress, appaddr);
    _appAddress = appaddr;
    _isExcludedFromFee[_appAddress] = true;
}

event marketingAddressUpdated(address indexed previous, address indexed adr);
function setNewMarketingAddress(address payable markt) public onlyOwner {
    emit marketingAddressUpdated(_marketingAddress, markt);
    _marketingAddress = markt;
    _isExcludedFromFee[_marketingAddress] = true;
}

event burnAddressUpdated(address indexed previous, address indexed adr);
function setNewBurnAddress(address payable burnaddr) public onlyOwner {
    emit burnAddressUpdated(_burnAddress, burnaddr);
    _burnAddress = burnaddr;
    _isExcludedFromFee[_burnAddress] = 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`, 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.

```
function sendETHToFee(uint256 amount) private {
    uint256 devAmount = amount.div(2);
    uint256 mktAmount = amount.mul(3).div(8);
    uint256 burnAmount = amount.sub(devAmount).sub(mktAmount);
    _appAddress.transfer(devAmount);
    _marketingAddress.transfer(mktAmount);
    _burnAddress.transfer(burnAmount);
}
```

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.

Unchecked transfer

The return value of an external transfer/transferFrom call is not checked.

```
function rescueForeignTokens(address _tokenAddr, address _to, uint _amount) public onlyOwner {
    emit tokensRescued(_tokenAddr, _to, _amount);
    IERC20(_tokenAddr).transfer(_to, _amount);
}
```

Recommendation

Use `SafeERC20`, or ensure that the transfer/transferFrom return value is checked.

Exploit scenario

```
contract Token {
    function transferFrom(address _from, address _to, uint256 _value) public returns (bool success);
}
contract MyBank{
    mapping(address => uint) balances;
    Token token;
    function deposit(uint amount) public{
        token.transferFrom(msg.sender, address(this), amount);
        balances[msg.sender] += amount;
    }
}
```

Several tokens do not revert in case of failure and return false. If one of these tokens is used in `MyBank`, `deposit` will not revert if the transfer fails, and an attacker can call `deposit` for free..

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

Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
function setFee(uint256 redisFeeOnBuy, uint256 redisFeeOnSell, uint256 taxFeeOnBuy, uint256 taxFeeOn
        require(redisFeeOnBuy &lt; 11, &quot;Redis cannot be more than 10.&quot;);
        require(redisFeeOnSell &lt; 11, &quot;Redis cannot be more than 10.&quot;);
        require(taxFeeOnBuy &lt; 7, &quot;Tax cannot be more than 6.&quot;);
        require(taxFeeOnSell &lt; 7, &quot;Tax cannot be more than 6.&quot;);
    _redisFeeOnBuy = redisFeeOnBuy;
    _redisFeeOnSell = redisFeeOnSell;
    _taxFeeOnBuy = taxFeeOnBuy;
    _taxFeeOnSell = taxFeeOnSell;
}
```

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.

```
Function IUniswapV2Router01.WETH() (#308) is not in mixedCase
Event FitApetokensRescued(address,address,uint256) (#726) is not in CapWords
Event FitApeappAddressUpdated(address,address) (#732) is not in CapWords
Event FitApemarketingAddressUpdated(address,address) (#739) is not in CapWords
Event FitApeburnAddressUpdated(address,address) (#746) is not in CapWords
Parameter FitApe.setSnipeBlocks(uint8)._blocks (#566) is not in mixedCase
Parameter FitApe.rescueForeignTokens(address,address,uint256)._tokenAddr (#727) is not in mixedCase
Parameter FitApe.rescueForeignTokens(address,address,uint256)._to (#727) is not in mixedCase
Parameter FitApe.rescueForeignTokens(address,address,uint256)._amount (#727) is not in mixedCase
Parameter FitApe.toggleSwap(bool)._swapEnabled (#828) is not in mixedCase
Variable FitApe._isBlacklisted (#506) is not in mixedCase
Variable FitApe._PresaleAddress (#508) is not in mixedCase
Constant FitApe._tTotal (#515) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FitApe._name (#527) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FitApe._symbol (#528) is not in UPPER_CASE_WITH_UNDERSCORES
Constant FitApe._decimals (#529) is not in UPPER_CASE_WITH_UNDERSCORES
```

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.

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

Owner can change burn address

```
event burnAddressUpdated(address indexed previous, address indexed adr);
function setNewBurnAddress(address payable burnaddr) public onlyOwner {
    emit burnAddressUpdated(_burnAddress, burnaddr);
    _burnAddress = burnaddr;
    _isExcludedFromFee[_burnAddress] = true;
}
```

Recommendation

There is no need to be able to change the burn address. Hard-code the burn address for more transparency.

Owner privileges

- Owner cannot set fees higher than 25%
- Owner cannot change max transaction amount
- Owner can exclude from fees
- Owner can pause the contract
- Owner can blacklist addresses

Extra notes by the team

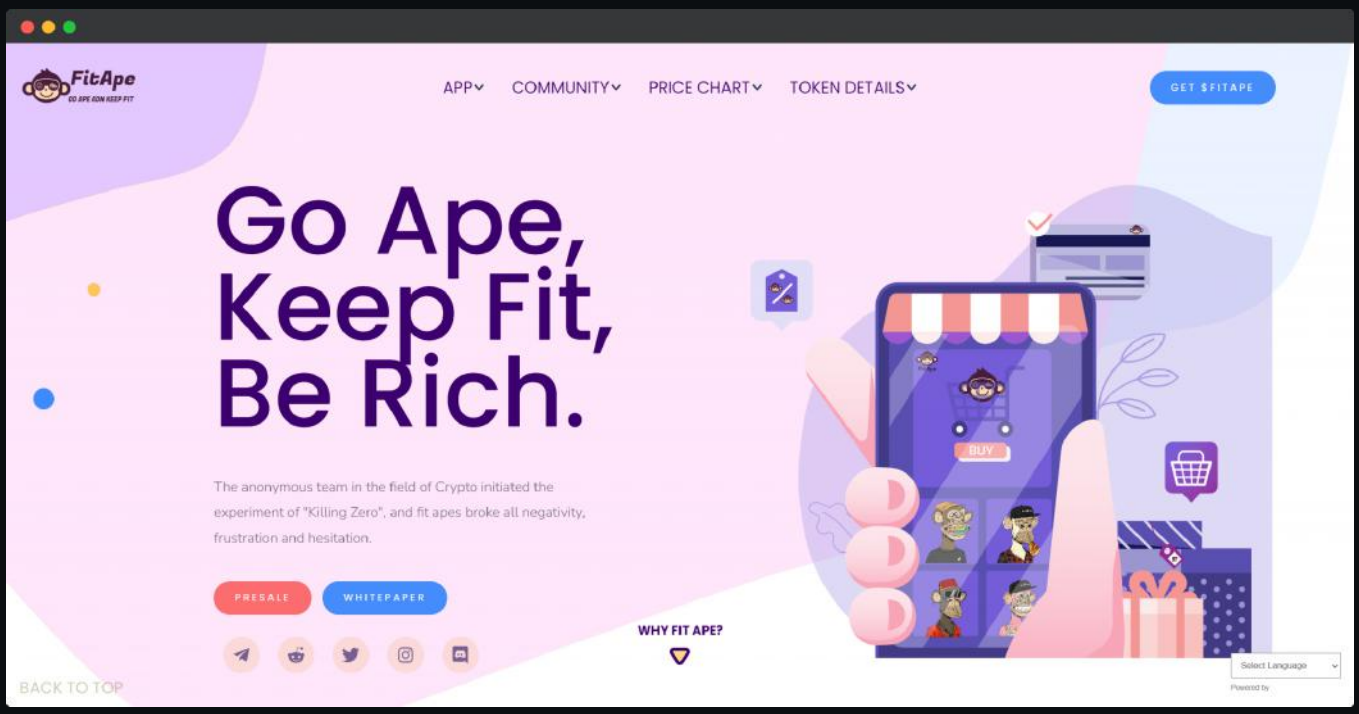
No notes

Contract Snapshot

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

using SafeMath for uint256;
mapping (address =&gt; uint256) private _rOwned;

mapping (address =&gt; mapping (address =&gt; uint256)) private _allowances;
mapping (address =&gt; bool) private _isExcludedFromFee;
mapping(address =&gt; bool) public _isBlacklisted;

address public _PresaleAddress = 0x000000000000000000000000000000000000dEaD;
bool public liquidityLaunched = false;
bool public isFirstLaunch = true;
uint256 public lastSnipeTaxBlock;
uint8 public snipeBlocks = 0;

uint256 private constant MAX = ~uint256(0);
uint256 private constant _tTotal = 10000 * (10**12) * (10**9);
uint256 private _rTotal = (MAX - (MAX % _tTotal));

uint256 private _redisFeeOnBuy = 2;
uint256 private _taxFeeOnBuy = 8;

uint256 private _redisFeeOnSell = 2;
uint256 private _taxFeeOnSell = 8;

uint256 private _redisFee;
uint256 private _taxFee;

string private constant _name = "FitApe";
string private constant _symbol = "FitApe";
uint8 private constant _decimals = 9;
```

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

FitApe

Audited by Coinsult.net

Date: 9 June 2022

Advanced Manual Smart Contract Audit