Coinsult

Advanced Manual Smart Contract Audit

Project: Sapphires
Website: https://sapphires.finance

Low-risk	Medium-risk	High-risk
5 low-risk code issues found	0 medium-risk code issues found	0 high-risk code issues found

Contract address
0xeC8cd873280f7872e86441cf9d694F3E303E9ea1

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	0xca2e30bf3a59363f352c210d77b38e6698bf5062	9,999,999	100.0000%
2	0x298311d44051d0c7cdbf6d85a727ede2254f1bff	1	0.0000%

Source code

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

https://bscscan.com/address/0xeC8cd873280f7872e86441cf9d694F3E303E9ea1#code

Manual Code Review

🟢 Low-risk

5 low-risk code issues found.
Could be fixed, will not bring problems.

- Usage of block.timestamp. block.timestamp can be manipulated by miners

  Recommendation: Avoid relying on block.timestamp.

```
        if ( inSwap ) return;
        uint256 rebaseRate;
        uint256 deltaTimeFromInit = block.timestamp -
_initRebaseStartTime;
        uint256 deltaTime = block.timestamp - _lastRebasedTime;
        uint256 times = deltaTime.div(15 minutes);
        uint256 epoch = times.mul(15);
```

- Contract contains Reentrancy vulnerabilities: _transfer(address,address,uint256)

  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 _transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) internal returns (bool) {

        require(!blacklist[sender] && !blacklist[recipient],
"in_blacklist");

        if (inSwap) {
            return _basicTransfer(sender, recipient, amount);
        }
        if (shouldRebase()) {
```

```
            rebase();
        }

        if (shouldAddLiquidity()) {
            addLiquidity();
        }

        if (shouldSwapBack()) {
            swapBack();
        }

        uint256 gonAmount = amount.mul(_gonsPerFragment);
        _gonBalances[sender] = _gonBalances[sender].sub(gonAmount);
        uint256 gonAmountReceived = shouldTakeFee(sender, recipient)
            ? takeFee(sender, recipient, gonAmount)
            : gonAmount;
        _gonBalances[recipient] = _gonBalances[recipient].add(
            gonAmountReceived
        );


        emit Transfer(
            sender,
            recipient,
            gonAmountReceived.div(_gonsPerFragment)
        );
        return true;
    }
```

- Variables that are written but never read and written again.

  Recommendation: Fix or remove the writes.

```
        (bool success, ) = payable(treasuryReceiver).call{
            value: amountETHToTreasuryAndSIFAndDF.mul(treasuryFee).div(

treasuryFee.add(sapphiresInsuranceFundFee).add(developmentFee)
            ),
            gas: 30000
        }("");
        (success, ) = payable(sapphiresInsuranceFundReceiver).call{
```

```
            value:
amountETHToTreasuryAndSIFAndDF.mul(sapphiresInsuranceFundFee).div(

treasuryFee.add(sapphiresInsuranceFundFee).add(developmentFee)
            ),
            gas: 30000
        }("");
        (success, ) = payable(developmentFund).call{
            value:
amountETHToTreasuryAndSIFAndDF.mul(developmentFee).div(

treasuryFee.add(sapphiresInsuranceFundFee).add(developmentFee)
            ),
            gas: 30000
        }("");
```

- Calls to a function sending Ether to an arbitrary address.

```
        (bool success, ) = payable(treasuryReceiver).call{
            value: amountETHToTreasuryAndSIFAndDF.mul(treasuryFee).div(

treasuryFee.add(sapphiresInsuranceFundFee).add(developmentFee)
            ),
            gas: 30000
        }("");
```

- No zero address validation

Check that the new address is not the zero address

```
    function setFeeReceivers(
        address _autoLiquidityReceiver,
        address _treasuryReceiver,
        address _sapphiresInsuranceFundReceiver,
        address _firePit
    ) external onlyOwner {
        autoLiquidityReceiver = _autoLiquidityReceiver;
        treasuryReceiver = _treasuryReceiver;
        sapphiresInsuranceFundReceiver =
_sapphiresInsuranceFundReceiver;
        firePit = _firePit;
    }
```

🟡 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

🟢 Owner can only blacklist bot contracts, not normal addresses

```
    function setBotBlacklist(address _botAddress, bool _flag) external
onlyOwner {
        require(isContract(_botAddress), "only contract address, not
allowed exteranlly owned account");
        blacklist[_botAddress] = _flag;
    }
```

🟢 Owner can not change the fees

```
    uint256 public liquidityFee = 40;
    uint256 public treasuryFee = 25;
    uint256 public sapphiresInsuranceFundFee = 50;
    uint256 public sellFee = 20;
    uint256 public firePitFee = 25;
    uint256 public developmentFee = 10;

    uint256 public totalFee =

liquidityFee.add(treasuryFee).add(sapphiresInsuranceFundFee).add(
            firePitFee).add(developmentFee);
    uint256 public feeDenominator = 1000;
```

🟡 Owner can withdraw all tokens in the contract to Treusury wallet

```
   function withdrawAllToTreasury() external swapping onlyOwner {

       uint256 amountToSwap = _gonBalances[address(this)].div(_gonsPerFragment);
       require( amountToSwap > 0,"There is no Sapphires token deposited in token contract");
       address[] memory path = new address[](2);
       path[0] = address(this);
       path[1] = router.WETH();
       router.swapExactTokensForETHSupportingFeeOnTransferTokens(
           amountToSwap,
           0,
           path,
           treasuryReceiver,
           block.timestamp
       );
   }
```

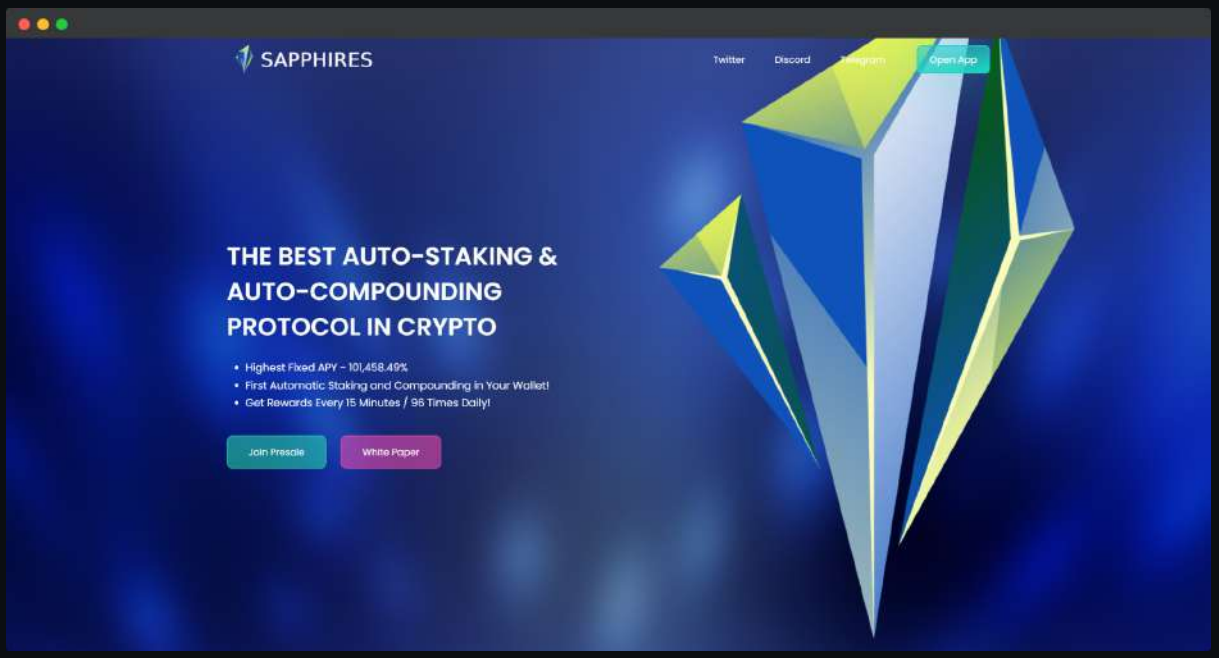
- Ownership is not renounced
- Owner can exclude from fees

```
    function setWhitelist(address _addr) external onlyOwner {
        _isFeeExempt[_addr] = true;
    }
```

Contract Snapshot

```
contract Sapphires is ERC20Detailed, Ownable {

    using SafeMath for uint256;
    using SafeMathInt for int256;

    event LogRebase(uint256 indexed epoch, uint256 totalSupply);

    string public _name = "Sapphires";
    string public _symbol = "SPHS";
    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 =
        10 * 10**6 * 10**DECIMALS;

    uint256 public liquidityFee = 40;
    uint256 public treasuryFee = 25;
    uint256 public sapphiresInsuranceFundFee = 50;
    uint256 public sellFee = 20;
    uint256 public firePitFee = 25;
    uint256 public developmentFee = 10;

    uint256 public totalFee =

liquidityFee.add(treasuryFee).add(sapphiresInsuranceFundFee).add(
            firePitFee).add(developmentFee);
    uint256 public feeDenominator = 1000;

    address DEAD = 0x000000000000000000000000000000000000dEaD;
```

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: 91%

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 (KYC is planned in the future with Coinsult)

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.