Coinsult

Advanced Manual Smart Contract Audit

Project: Private: BNB ninja

Website: https://bnbninja.io/

● **Low-Risk**

7 low-risk code issues found

● **Medium-Risk**

0 medium-risk code issues found

● **High-Risk**

0 high-risk code issues found

Contract Address

0xa9bcc3f55fb920eb50878d4e15a820c5634ba2a7

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

Rank	Address	Quantity (Token)	Percentage
1	0xed43f10531d056a48365d8f598ce0f57faff7f55	100,000,000,000	100.0000%

Source Code

Coinsult was comissioned by Private: BNB ninja to perform an audit based on the following smart contract:

https://bscscan.com/address/0xa9bcc3f55fb920eb50878d4e15a820c5634ba2a7#code

Manual Code Review

In this audit report we will highlight all these issues:

🟢 Low-Risk

7 low-risk code issues found

🟡 Medium-Risk

0 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 sender,
    address recipient,
    uint256 amount
) internal virtual {
    require(sender != address(0), "ERC20: transfer from the zero address");
    require(recipient != address(0), "ERC20: transfer to the zero address");

    _beforeTokenTransfer(sender, recipient, amount);

    uint256 senderBalance = _balances[sender];
    require(senderBalance &gt;= amount, "ERC20: transfer amount exceeds balance");
unchecked {
    _balances[sender] = senderBalance - amount;
}
    _balances[recipient] += amount;

    emit Transfer(sender, recipient, amount);

    _afterTokenTransfer(sender, recipient, amount);
}
```

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.

```
if (!_isExcludedFromPeriodLimit[from]) {
            accountLastPeriodSellVolume[from] = accountLastPeriodSellVolume[from].add(amount);
            Sell memory sell;
            sell.amount = amount;
            sell.time = block.timestamp;
            accountSells[from].push(sell);
            emit AddLastPeriodSellInfo(sell.time, sell.amount);
        }
```

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.

```
require(newValue &gt;= 200000 &amp;&amp; newValue &lt;= 500000, &quot;BNB Ninja: gasForProcessing mu
```

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 updatePancakeRouter(address newAddress) public onlyOwner {
    require(newAddress != address(pancakeRouter), "BNB Ninja: The router already has that address");
    emit UpdatePancakeRouter(newAddress, address(pancakeRouter));
    pancakeRouter = IPancakeRouter02(newAddress);
    address _pancakePair = IPancakeFactory(pancakeRouter.factory())
    .createPair(address(this), pancakeRouter.WETH());
    pancakePair = _pancakePair;

    dividendTracker.excludeFromDividends(address(pancakeRouter));
    dividendTracker.excludeFromDividends(pancakePair);

    excludeFromAllLimits(newAddress, true);

    _isExcludedFromPeriodLimit[pancakePair] = true;
    _isExcludedFromMaxHoldLimit[pancakePair] = 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 swapAndDistributeBNB(uint256 tokens) private {
    swapTokensForEth(tokens);
    uint256 balance = address(this).balance;
    uint256 accTotal = accBuybackFee.add(accDividendFee).add(accMarketingFee);
    uint256 forMarketing = balance.mul(accMarketingFee).div(accTotal);
    uint256 forBuyback = balance.mul(accBuybackFee).div(accTotal);
    uint256 forDividends = balance.sub(forBuyback).sub(forMarketing);

    emit CalculatedBNBForEachRecipient(forMarketing, forBuyback, forDividends);

    (bool success,) = address(marketingWallet).call{value: forMarketing}("");

    if(success) {
        emit SwapAndSendTo(accMarketingFee, forMarketing, "MARKETING");
        accMarketingFee = 1;
    }

    (success,) = address(buybackWallet).call{value: forBuyback}("");

    if(success) {
        emit SwapAndSendTo(accBuybackFee, forBuyback, "ANTI DUMP");
        accBuybackFee = 1;
```

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 withdrawStuckTokens(address _token, uint256 _amount) public onlyOwner {
    IERC20(_token).transfer(msg.sender, _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.

Conformance to Solidity naming conventions

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

```
string public _name;
string public _symbol;
```

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.

Owner privileges

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

🟡 Owner can exclude addresses from all limits (fees, max transaction, period and max hold)

🟡 Owner can exclude addresses from dividends

Extra notes by the team

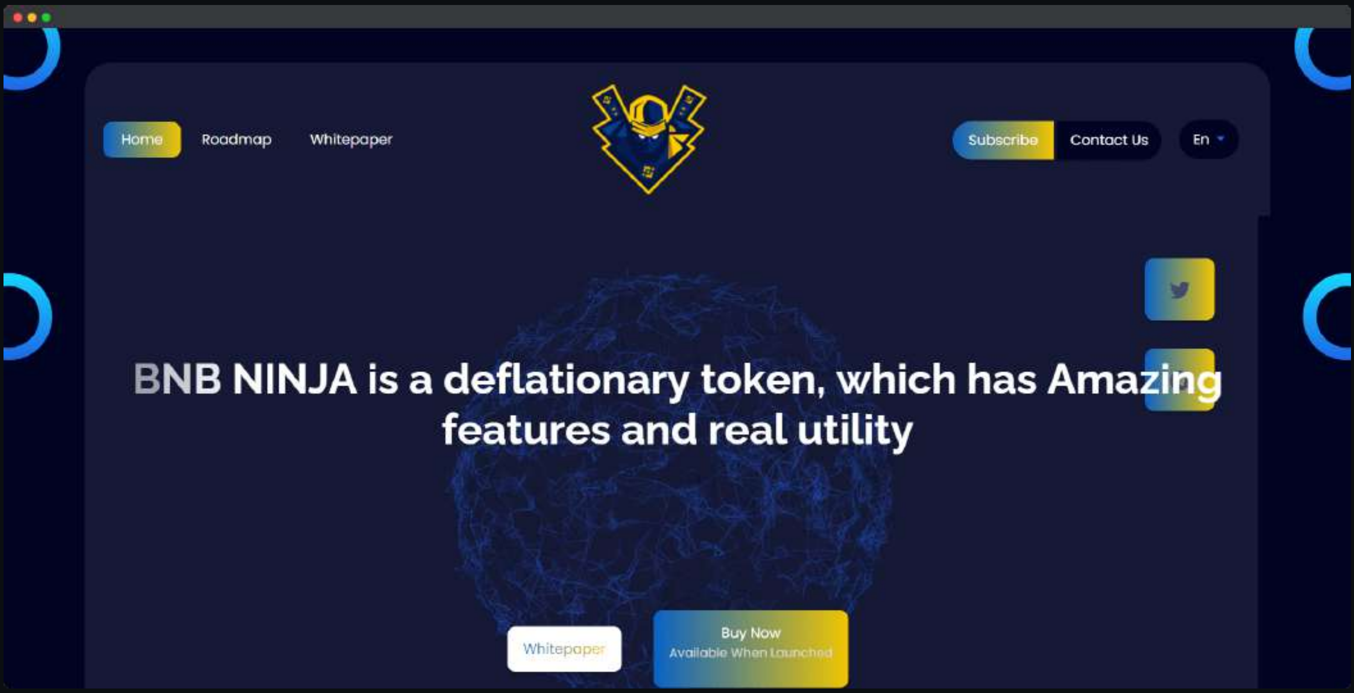
No notes

Contract Snapshot

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

IPancakeRouter02 public pancakeRouter;
address public pancakePair;

bool private swapping;

BnbnDividendTracker public dividendTracker;
address public marketingWallet = 0xEaC29c20b5bd562FFE483Db0A3EBBA1d402b9738; //0x455a2A20d3f7DfC4891
address public buybackWallet = 0xF0946cB3Ddf998C54F652C45D78023d918EA229b; //0x9A2CBA7d715b03Dd4b99b

uint256 public maxTransactionAmount;
uint256 public maxHoldingAmount;
uint256 public swapTokensAtAmount = 10**6 * (10**decimals());

uint256 public buyback = 5;
uint256 public marketingFee = 20;
uint256 public dividendFee = 5;
```

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.



BNB NINJA is a deflationary token, which has Amazing features and real utility

● Mobile Friendly

● Does not contain jQuery errors

● SSL Secured

● No major spelling errors

Project Overview

● Not KYC verified by Coinsult