Coinsult

Advanced Manual Smart Contract Audit

SLEEP TO EARN

Project: Sleep To Earn

Website: https://sleeptoearn.finance/

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

Contract Address

0x312Ed6F4eaE6a3aA467D60d6ad1fd31CC3A8B5d0

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	0x859b6148835940e41aae58d9fb2df3dc42e5c69c	100,000,000	100.0000%

Source Code

Coinsult was comissioned by Sleep To Earn to perform an audit based on the following smart contract:

https://bscscan.com/address/0x312Ed6F4eaE6a3aA467D60d6ad1fd31CC3A8B5d0#code

Manual Code Review

In this audit report we will highlight all these issues:

🟢 Low-Risk

3 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.

Divide before multiply

Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision.

```
_balances[recipient] = _balances[recipient].add(amount/100*(100-tax));
```

Recommendation

Consider ordering multiplication before division.

Exploit scenario

```
contract A {
    function f(uint n) public {
        coins = (oldSupply / n) * interest;
    }
}
```

If n is greater than `oldSupply`, `coins` will be zero. For example, with `oldSupply = 5; n = 10, interest = 2`, coins will be zero. If `(oldSupply * interest / n)` was used, `coins` would have been 1. In general, it's usually a good idea to re-arrange arithmetic to perform multiplication before division, unless the limit of a smaller type makes this dangerous.

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

Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
function update_tax_Address(address _newAddress) public onlyOwner{
    require(_newAddress != address(0), "BEP20: new address must not be zero address");
    taxAddress = _newAddress;
}
```

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.

Redundant Statements

Detect the usage of redundant statements that have no effect.

```
function _msgData() internal view returns (bytes memory) {
  this; // silence state mutability warning without generating bytecode - see https://github.com/eth
  return msg.data;
}
```

Recommendation

Remove redundant statements if they congest code but offer no value.

Exploit scenario

```
contract RedundantStatementsContract {

    constructor() public {
        uint; // Elementary Type Name
        bool; // Elementary Type Name
        RedundantStatementsContract; // Identifier
    }

    function test() public returns (uint) {
        uint; // Elementary Type Name
        assert; // Identifier
        test; // Identifier
        return 777;
    }
}
```

Each commented line references types/identifiers, but performs no action with them, so no code will be generated for such statements and they can be removed.

Owner privileges

- Owner cannot set fees higher than 25%
- Owner cannot pause trading
- Owner cannot change max transaction amount

Extra notes by the team

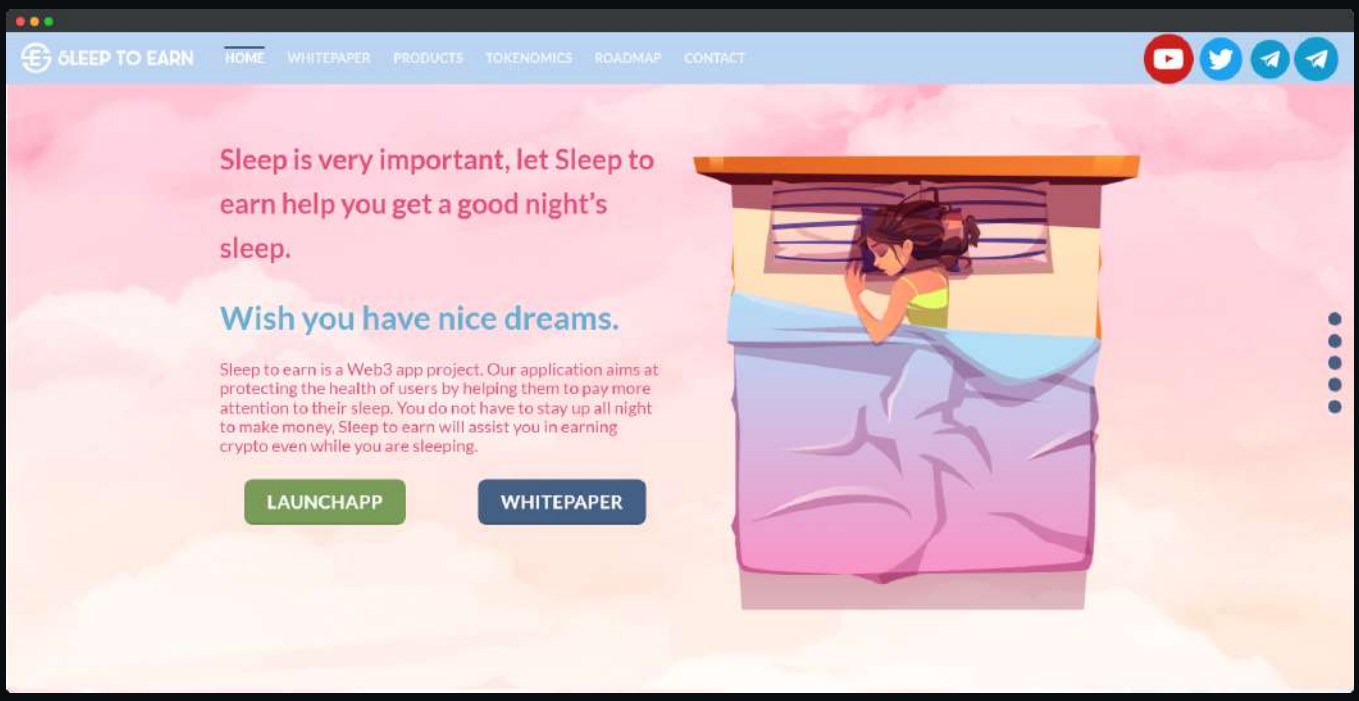
No notes

Contract Snapshot

```
contract BEP20Token_SleepToEarn is Context, IBEP20, Ownable {
using SafeMath for uint256;
mapping (address =&gt; uint256) private _balances;
mapping (address =&gt; mapping (address =&gt; uint256)) private _allowances;
uint256 private _totalSupply;
uint8 private _decimals;
string private _symbol;
string private _name;
address public masterWallet;
uint public tax=2;
bool public taxStatus = true;
address public taxAddress;
```

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