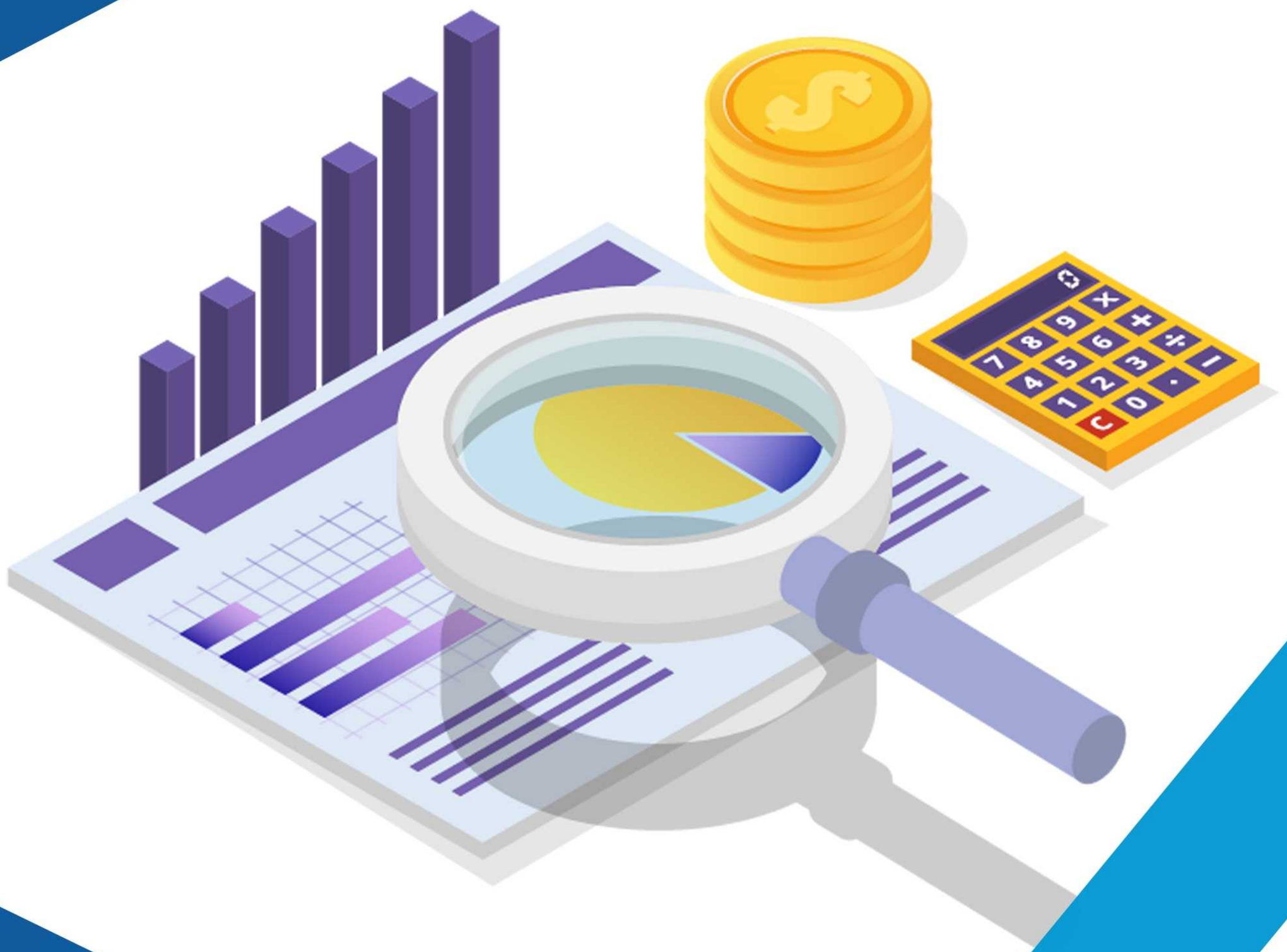
QuillAudits



Audit Report
May, 2021

UniCake

Contents

Audit Details and Target

1. Contract Link :

https://github.com/UniCake172/Smart-Contract/commit/98b8796a8599328647a5e8a1c626cbad1a63e029

2. Audit Target :

- To find the security bugs and issues regarding security, potential risks and critical bugs.
- Check gas optimization and check gas consumption.
- Check function reusability and code optimisation.
- Test the limit for token transfer and check the accuracy in decimals.
- Check the functions and naming conventions.
- Check the code for proper checks before every function call.
- Event trigger checks for security and logs.
- Checks for constant and function visibility and dependencies.
- Validate the standard functions and checks.
- Check the business logic and correct implementation.
- Automated script testing for multiple use cases including transfers, including values and multi transfer check.
- Automated script testing to check the validations and limit tests before every function call.
- Check the use of data type and storage optimisation.
- Calculation checks for different use cases based on the transaction, calculations and reflected values.

Functions list and audit details

1. Read Functions in Contract:

- name()
- symbol()

- decimals()
- totalSupply()
- balanceOf()
- allowance()
- owner()

2. Write Function in Contract:

- transfer()
- approve()
- transferFrom()
- increaseAllowance()
- decreaseAllowance()
- _transfer()
- _mint()
- _approve()
- _setupDecimals()
- renounceOwnership()
- transferOwnership()

Overview of UniCake

UniCake is an erc20 compatible token with mint, burn, transfer and balance compatibility with all erc20 standard compatible wallets. UniCake implements ownable security checks for all ownable functions, which make this contract safe and trusted token development. Implementing all standard functions for ERC20 tokens for compatibility with all standard swapping, exchange, dex and exchange platforms. Users can hold the token, transfer and check balances with read function calls.
The smart contract is developed with the security checks for balance, user address, transfer value, managed token allowance and approval for delegate token transfer functionality. UniCake tokens can be used to integrate any business logic compatible with the ERC20 token standard. The contract owner can burn and mint tokens to manage the supply and maximize the business for the tokenomics.

Tokenomics

As per the information provided, the tokens generated will be initially transferred to the contract owner and then further division will be done based on the business logic of the application.

Tokens cannot be directly purchased from the smart contract, so there will be an additional or third-party platform that will help users to purchase the tokens.

Tokens can be held, transferred and delegated freely. Contract owners can increase and decrease supply based on the business use case and supply management.

Scope of Audit

The scope of this audit was to analyse UniCake smart contracts codebase for quality, security, and correctness.

Checked Vulnerabilities

The smart contract is scanned and checked for multiple types of possible bugs and issues. This mainly focuses on issues regarding security, attacks, mathematical errors, logical and business logic issues. Here are some of the commonly known vulnerabilities that are considered:

- TimeStamp dependencies.
- Variable and overflow
- Calculations and checks
- SHA values checks
- Vulnerabilities check for use case
- Standard function checks
- Checks for functions and required checks
- Gas optimisations and utilisation
- Check for token values after transfer
- Proper value updates for decimals
- Array checks
- Safemath checks
- Variable visibility and checks
- Error handling and crash issues
- Code length and function failure check
- Check for negative cases
- D-DOS attacks
- Comments and relevance
- Address hardcoded
- Modifiers check
- Library function use check
- Throw and inline assembly functions
- Locking and unlocking (if any)
- Ownable functions and transfer ownership checks

- Array and integer overflow possibility checks
- Revert or Rollback transactions check

Techniques and Methods

- Manual testing for each and every test cases for all functions.
- Running the functions, getting the outputs and verifying manually for multiple test cases.
- Automated script to check the values and functions based on automated test cases written in JS frameworks.
- Checking standard function and check compatibility with multiple wallets and platforms
- Checks with negative and positive test cases.
- Checks for multiple transactions at the same time and checks d-dos attacks.
- Validating multiple addresses for transactions and validating the results in managed excel.
- Get the details of failed and success cases and compare them with the expected output.
- Verifying gas usage and consumption and comparing with other standard token platforms and optimizing the results.
- Validate the transactions before sending and test the possibilities of

Structural Analysis
In this step we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.
SmartCheck.

Static Analysis
Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

Gas Consumption

In this step we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

Issue Categories

High severity issues

Issues that must be fixed before deployment else they can create major issues.

Medium level severity issues

These issues will not create major issues in working but affect the performance of the smart contract.

Low level severity issues

These issues are more suggestions that should be implemented to refine the code in terms of gas, fees, speed and code accuracy

Informational

- Code is written in a very concise way which can be more informative and secure by more checks and modifier use.
- The use of variables and naming conventions can be minimized
- Gas use is proper and optimized as per the testing on the Ropsten test network and checking of the old transaction from this contract.
- **Suggestion**: There must be some locking function in case of any loss or security issue so that the admin can lock the contract in case of any security issue to the token.

Number of issues per severity

	High	Medium	Low	Total Issues
Open	0	0	0	0
Closed	0	2	4	6

Issues Found – Code Review / Manual Testing

High severity issues

1. No major high severity issue found.

Medium severity issues

1. Check for zero amount, in transfer is recommended to avoid any unintentional loss of gas fees or zero value transactions.
   Line: 457

   Closed

2. Approval check missing. Risk to the security of the contract and user tokens. Failed in manual testing.

   Closed

```
ftrace | funcSig
401     function transferFrom(address sender, address recipient, uint256 amount) public
402         _transfer(sender, recipient, amount);
403         _approve(sender, _msgSender(), _allowances[sender][_msgSender()].sub(amount,
404         return true;
405     }
406
```

Low level severity issues

1. The naming convention can be improved for a better understanding of the contract. For _total.
   Line: 599

   Closed

2. This constructor will be called once at the time of deployment. Unable to understand the need for unnecessary function to increase code complexity.
   Line: 603

   Closed

```
601        constructor () public {
602            // mint tokens
603            _mint(msg.sender, _total.mul(10 ** 18));
604        }
605    }
```

3. Not recommended to use total supply as 0 initially. This value will be used by etherscan and other platforms to get the exact value of the supply in the contract.
Line: 289

Closed

```
288        mapping (address => mapping (address => uint256)) private _allowances;
289        uint256 private _totalSupply = 0;
290
```

4. Unused Internal function found without justifiable logic.
Line: 512

```
511        */
           ftrace | funcSig
512        function _setupDecimals(uint8 decimals_) internal {
513            _decimals = decimals_;
514        }
515
```

Suggestion: Either make this function external or remove this as changing decimals after deployment is not recommended as per the ERC20 standards.

Closed

Closing Summary

Code is written with implementing all guidelines of erc20 contract development protocols.
All functions are checked and verified for the multiple types of test cases, and all test cases passed.
Suggestions and issues have been fixed and code is tested after implementing all the changes. To achieve the business logic, some functions are changed by implementing mint, add balance and burn functionality of the contract development. Proper checks are implemented to make sure the security of the tokens and business logics.

Disclaimer

QuillHash audit is not a security warranty, investment advice, or an endorsement of the UniCake platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the UniCake Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

UniCake

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

hello@quillhash.com