Alpha Ape LLC

Security Audit report for
TheThinker

BSC Contract Address
0xd8B69FE5C65685b08c91E1119A79D775866E781B

October.15.2022

Table of Contents

SUMMARY OF AUDIT

DETAILS OF AUDITED PROJECT

Audited Project:	TheThinker – (TKR)
Source Code:	https://bscscan.com/address/0xd8B69FE5C65685b08c91E1119A79D775866E781B#code (verified)
Solidity File:	**TheThinker**.sol
Security Audit Date:	Oct. 15 - 2022
Revisions:	Initial Audit Oct.15.2022
Auditing Methods:	Automatic review + Manual review

AUDITING METHODS AND COVERING SECTORS

Evaluation objective for the security audit:

- Quality of smart contracts code
- Issues and vulnerabilities with security
- Documentation, project specifics and commenting on smart contract
- Correctness of specifications regarding the use-case
- Following the best practices on smart contract

Audit covers these sectors of smart contract for possible vulnerabilities, issues and recommendations for better practices in case of severe or medium issues:

- Dependence Transaction Order
- Single and Cross-Function Reentrancy
- Time Dependency
- Integer Overflow
- Integer Underflow
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Number rounding errors
- Insufficient gas issues
- Logical oversights
- Access control
- Centralization of power
- Logic-Specification
- Contradiction
- Functionality duplication
- Malicious contract behaviour and abusable functions
- Possible DoS vulnerabilities

The code review conducted for this audit follows the following structure:

1. Review of the specifications, documentation and commenting provided by the project owners regarding the functionality of the smart contract
2. Automated analysis of the smart contract followed by manual, line-by-line analysis of the smart contract
3. Assessment of smart contracts correctness regarding the documentation and commenting compared to functionality
4. Assessment of following the best practices
5. Recommendations for better practises in case severe or medium vulnerabilities

SMART CONTRACT DETAILS

Contract ID	0xd8B69FE5C65685b08c91E1119A79D775866E781B (verified)
Blockchain:	Binance Smart Chain
Language Used:	Solidity
Compiler Version:	v0.8.14+commit.80d49f37
Bscscan Verification:	2022-06-03
Type of Smart Contract:	BEP20 Token – Standard Token
Libraries Used:	
Optimization Enabled:	No with 200 runs

Number of Interfaces: 5
Number of Contracts: 5
Solidity Versions: ^0.4.24, 0.8.14
Total Lines: 935
Sell Tax: (11% Contract) 11%
Buy Tax: (5% Contract) 5%

SUMMARY OF AUDIT RESULTS

Alpha Ape LLC smart contract audit for TheThinker (TKR) is marked as **PASSED** result without severe issues on the logic and functions of the contract. Review of the documentation and description of the projects use-case and the contract follows the line of good practices for majority. Clean and default commented contract. There are no severe vulnerability findings in the contract and no concerns about malicious use of the contract functions.

*Contract owner can change buy and sell tax and maximum wallet holding.

*Contract owner is able to whitelist wallets from the limits.

*Contract is not renounced.

SEVERITY OF RISKS AND VULNERABILITIES

LOW-SEVERITY	MEDIUM-SEVERITY	HIGH-SEVERITY
0	0	0

REPORTED VULNERABILITIES AND ISSUES

LEVEL OF SEVERITY	DESCRIPTION	FILE	CODELINES AFFECTED

***Sidenote**

State variable visibility is not set. It is best practice to set the visibility of state variables explicitly. The default visibility for "inSwapAndLiquify" is internal. Other possible visibility settings are public and private.

Locations

```
457 address public uniswapV2Pair;
458
459 bool inSwapAndLiquify;
460 bool public swapAndLiquifyEnabled = false;
461
```

FINDINGS IN-DEPTH

NO FINDINGS ON THE CONTRACT OR NOTICABLE LOGICAL ISSUES

DOCUMENTATION AND COMMENTING

The code has a decent amount of comments and documentation. Improving stage of commenting on smart contracts, and cleaning the commenting, helps userbase to understand all the functionalities and use-case of the contract. Functions are marked in understandable way and commenting on the contract has been made human readable to understand.

CORRECTNESS OF SPECIFICATIONS

Smart contract follows the functionality that is stated in the documentation and description of the contract. The use-case is also in line what is described about the project.

FOLLOWING THE BEST PRACTICES

The contract follows the best practices in majority and there is no concerns from the auditor of any malicious use of the contracts functions.

HISTORY OF REVISIONS, FUNCTIONS AND VARIABLES

FUNCTIONS AND SIGNATURES

Signature hash | Function Signature
==========
11902160 => _getTValues(uint256)
16279055 => isContract(address)
39509351 => increaseAllowance(address,uint256)
119df25f => _msgSender()
8b49d47e => _msgData()
18160ddd => totalSupply()
70a08231 => balanceOf(address)
a9059cbb => transfer(address,uint256)
dd62ed3e => allowance(address,address)
095ea7b3 => approve(address,uint256)
23b872dd => transferFrom(address,address,uint256)
771602f7 => add(uint256,uint256)
b67d77c5 => sub(uint256,uint256)
e31bdc0a => sub(uint256,uint256,string)
c8a4ac9c => mul(uint256,uint256)
a391c15b => div(uint256,uint256)
b745d336 => div(uint256,uint256,string)
f43f523a => mod(uint256,uint256)
71af23e8 => mod(uint256,uint256,string)
24a084df => sendValue(address,uint256)
a0b5ffb0 => functionCall(address,bytes)

241b5886 => functionCall(address,bytes,string)
2a011594 => functionCallWithValue(address,bytes,uint256)
d525ab8a => functionCallWithValue(address,bytes,uint256,string)
36455e42 => _functionCallWithValue(address,bytes,uint256,string)
8da5cb5b => owner()
715018a6 => renounceOwnership()
f2fde38b => transferOwnership(address)
602bc62b => getUnlockTime()
557ed1ba => getTime()
dd467064 => lock(uint256)
a69df4b5 => unlock()
017e7e58 => feeTo()
094b7415 => feeToSetter()
e6a43905 => getPair(address,address)
1e3dd18b => allPairs(uint256)
574f2ba3 => allPairsLength()
c9c65396 => createPair(address,address)
f46901ed => setFeeTo(address)
a2e74af6 => setFeeToSetter(address)
06fdde03 => name()
95d89b41 => symbol()
313ce567 => decimals()
3644e515 => DOMAIN_SEPARATOR()
30adf81f => PERMIT_TYPEHASH()
7ecebe00 => nonces(address)
d505accf => permit(address,address,uint256,uint256,uint8,bytes32,bytes32)
ba9a7a56 => MINIMUM_LIQUIDITY()
c45a0155 => factory()
0dfe1681 => token0()
d21220a7 => token1()
0902f1ac => getReserves()
5909c0d5 => price0CumulativeLast()
5a3d5493 => price1CumulativeLast()
7464fc3d => kLast()
89afcb44 => burn(address)
022c0d9f => swap(uint256,uint256,address,bytes)
bc25cf77 => skim(address)
fff6cae9 => sync()
485cc955 => initialize(address,address)
ad5c4648 => WETH()
e8e33700 => addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f305d719 => addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
baa2abde => removeLiquidity(address,address,uint256,uint256,uint256,address,uint256)
02751cec => removeLiquidityETH(address,uint256,uint256,uint256,address,uint256)
2195995c =>
removeLiquidityWithPermit(address,address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
ded9382a => removeLiquidityETHWithPermit(address,uint256,uint256,uint256,address,uint256,bool,uint8,bytes32,bytes32)
38ed1739 => swapExactTokensForTokens(uint256,uint256,address[],address,uint256)
8803dbee => swapTokensForExactTokens(uint256,uint256,address[],address,uint256)
7ff36ab5 => swapExactETHForTokens(uint256,address[],address,uint256)
4a25d94a => swapTokensForExactETH(uint256,uint256,address[],address,uint256)
18cbafe5 => swapExactTokensForETH(uint256,uint256,address[],address,uint256)
fb3bdb41 => swapETHForExactTokens(uint256,address[],address,uint256)
ad615dec => quote(uint256,uint256,uint256)
054d50d4 => getAmountOut(uint256,uint256,uint256)
85f8c259 => getAmountIn(uint256,uint256,uint256)
d06ca61f => getAmountsOut(uint256,address[])
1f00ca74 => getAmountsIn(uint256,address[])
af2979eb => removeLiquidityETHSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256)

5b0d5984 =>
removeLiquidityETHWithPermitSupportingFeeOnTransferTokens(address,uint256,uint256,uint256,address,uint256,bool,uint8,
bytes32,bytes32)
5c11d795 => swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
b6f9de95 => swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
791ac947 => swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
a457c2d7 => decreaseAllowance(address,uint256)
88f82020 => isExcludedFromReward(address)
13114a9d => totalFees()
a073d37f => minimumTokensBeforeSwapAmount()
2d838119 => tokenFromReflection(uint256)
52390c02 => excludeFromReward(address)
3685d419 => includeInReward(address)
5342acb4 => isExcludedFromFee(address)
437823ec => excludeFromFee(address)
ea2f0b37 => includeInFee(address)
104e81ff => _approve(address,address,uint256)
30e0789e => _transfer(address,address,uint256)
173865ad => swapAndLiquify(uint256)
b28805f4 => swapTokensForEth(uint256)
b09bbc79 => _tokenTransfer(address,address,uint256,bool)
2852df65 => _transferStandard(address,address,uint256)
16f1cc83 => _transferToExcluded(address,address,uint256)
c7d9be66 => _transferFromExcluded(address,address,uint256)
6ff6cdf4 => _transferBothExcluded(address,address,uint256)
30280a71 => excludeFromTxLimit(address,bool)
d4780e36 => _getValues(uint256)
c2264c6a => _getRValues(uint256,uint256,uint256)
94e10784 => _getRate()
97a9d560 => _getCurrentSupply()
8c193ee4 => _takeWalletsFees(uint256)
9632c62f => calculateWalletsFees(uint256)
301370af => removeAllFee()
e7e3e3a7 => restoreAllFee()
de2557c1 => setSaleFee()
f82a2aec => setAllBuyFeePercent(uint256,uint256,uint256)
dcb81d33 => setAllSaleFeePercent(uint256,uint256,uint256)
dd5f7398 => setMaxBuyTxAmount(uint256)
b38406f2 => setMaxSaleTxAmount(uint256)
f0f165af => setNumTokensSellToAddToLiquidity(uint256)
906e9dd0 => setMarketingAddress(address)
0c9be46d => setCharityAddress(address)
525fa81f => setLiquidityAddress(address)
c49b9a80 => setSwapAndLiquifyEnabled(bool)
2beeb115 => excludeWalletsFromWhales()
ff94d9a2 => checkForWhale(address,address,uint256)
4e3c06b1 => setExcludedFromWhale(address,bool)
44d4225f => setWalletMaxHoldingLimit(uint256)

HISTORY OF REVISIONS

Initial audit was performed October. 15-2022 and no need for further revisions of the smart contract audit. Team has been informed of a clean audit result.