RUGDOG

SMART CONTRACT SECURITY AUDIT

Final report

Plan: Complex

Walletika

January 2024

rugdog.net

the@rugdog.net

✧ CONTENTS

✧ INTRODUCTION

The report has been prepared for Walletika.

This smart contract, named WalletikaToken, is built on the Solidity programming language (version 0.6.12) for the Ethereum blockchain. It is designed to implement a custom BEP20 token, which is a standard for creating and issuing tokens on the Binance Smart Chain (BSC). The contract inherits properties and methods from the BEP20 contract. Here are the key functionalities of this contract:

1. Token Details: The contract initializes a new BEP20 token named 'Walletika' with the symbol 'WLTK'.
2. Maximum Supply: The maximum supply of the token is set to 100,000,000 units, adhering to the BEP20 decimal standard (i.e., 18 decimals, as indicated by `1e18`).
3. Minting Tokens: In the constructor, which is executed once when the contract is deployed, all of these tokens are minted and assigned to the contract owner's address.
4. Transfer Multiple: This function allows the sender to transfer tokens to multiple addresses in a single transaction. It requires two arrays: one with the addresses and the other with the corresponding amounts. The function checks that: - The number of addresses does not exceed 100.- The length of both the addresses and amounts arrays is the same. - The sender has enough balance to cover the total amount to be transferred.- It then iterates through the arrays, transferring the specified amounts to each address.
5. Recover Token: An owner-only function that allows the contract owner to recover tokens from the contract. This is typically used to recover tokens that were mistakenly sent to the contract address.
6. Access Control: The contract includes access control mechanisms (`onlyOwner` modifier) to restrict certain functions (like `recoverToken`) to the contract owner.

Name	Walletika
Audit date	2024-01-12 - 2024-01-12
Language	Solidity
Network	Binance Smart Chain

✧ CONTRACTS CHECKED

Name	Address
WalletikaToken	0x9eE10d2E9571AecfE5a604aF7fE71B96eBa84b7b

✧ AUDIT PROCESS

The code was audited by the team according to the following order:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual confirmation of all the issues found by the tools

Manual audit

- Thorough manual analysis of smart contracts for security vulnerabilities
- Smart contracts' logic check

✧ ATTACKS CHECKED

Title	Check result
Unencrypted Private Data On-Chain	✓ passed
Code With No Effects	✓ passed
Message call with hardcoded gas amount	✓ passed

Typographical Error	✓ passed
DoS With Block Gas Limit	✓ passed
Presence of unused variables	✓ passed
Incorrect Inheritance Order	✓ passed
Requirement Violation	✓ passed
Weak Sources of Randomness from Chain Attributes	✓ passed
Shadowing State Variables	✓ passed
Incorrect Constructor Name	✓ passed
Block values as a proxy for time	✓ passed
Authorization through tx.origin	✓ passed
DoS with Failed Call	✓ passed
Delegatecall to Untrusted Callee	✓ passed
Use of Deprecated Solidity Functions	✓ passed
Assert Violation	✓ passed
State Variable Default Visibility	✓ passed
Reentrancy	✓ passed
Unprotected SELFDESTRUCT Instruction	✓ passed

Unprotected Ether Withdrawal	✓ passed
Unchecked Call Return Value	✓ passed
Floating Pragma	✓ passed
Outdated Compiler Version	✓ passed
Integer Overflow and Underflow	✓ passed
Function Default Visibility	✓ passed

✧ OVERVIEW OF RELEVANCE LEVELS

High relevance

Issues of high relevance may lead to losses of users' funds as well as changes of ownership of a contract or possible issues with the logic of the contract.
High-relevance issues require immediate attention and a response from the team.

Medium relevance

While issues of medium relevance don't pose as high a risk as the high-relevance ones do, they can be just as easily exploited by the team or a malicious user, causing a contract failure and damaging the project's reputation in the process. Usually, these issues can be fixed if the contract is redeployed.
Medium-relevance issues require a response from the team.

Low relevance

Issues of low relevance don't pose high risks since they can't cause damage to the functionality of the contract. However, it's still recommended to consider fixing them.

ISSUES

High relevance issues

No high relevance issues found

Medium relevance issues

No medium relevance issues found

Low relevance issues

No low relevance issues found

✧ CONCLUSION

Walletika WalletikaToken contract was audited. No relevance issues were found.

✧ DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without RugDog prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts RugDog to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.
The rights to publish the results of this audit are exclusively retained by RugDog.

✧ AUTOMATED ANALYSIS

```
INFO:Detectors:
BEP20.constructor(string,string).name (contracts/BEP20.sol#57) shadows:
        - BEP20.name() (contracts/BEP20.sol#66-68) (function)
        - IBEP20.name() (contracts/IBEP20.sol#25) (function)
BEP20.constructor(string,string).symbol (contracts/BEP20.sol#57) shadows:
        - BEP20.symbol() (contracts/BEP20.sol#80-82) (function)
        - IBEP20.symbol() (contracts/IBEP20.sol#20) (function)
BEP20.allowance(address,address).owner (contracts/BEP20.sol#114) shadows:
        - Ownable.owner() (contracts/Ownable.sol#37-39) (function)
BEP20._approve(address,address,uint256).owner (contracts/BEP20.sol#228) shadows:
        - Ownable.owner() (contracts/Ownable.sol#37-39) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-
variable-shadowing
INFO:Detectors:
Address.isContract(address) (contracts/Address.sol#27-38) uses assembly
        - INLINE ASM (contracts/Address.sol#34-36)
Address._functionCallWithValue(address,bytes,uint256,string) (contracts/
Address.sol#135-161) uses assembly
        - INLINE ASM (contracts/Address.sol#153-156)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-
usage
INFO:Detectors:
Address._functionCallWithValue(address,bytes,uint256,string) (contracts/
Address.sol#135-161) is never used and should be removed
Address.functionCall(address,bytes) (contracts/Address.sol#82-84) is never used and
should be removed
Address.functionCall(address,bytes,string) (contracts/Address.sol#92-98) is never
used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (contracts/Address.sol#111-117)
is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (contracts/
Address.sol#125-133) is never used and should be removed
Address.isContract(address) (contracts/Address.sol#27-38) is never used and should
```

```
be removed
Address.sendValue(address,uint256) (contracts/Address.sol#56-62) is never used and
should be removed
BEP20._burn(address,uint256) (contracts/BEP20.sol#264-270) is never used and should
be removed
BEP20._burnFrom(address,uint256) (contracts/BEP20.sol#278-285) is never used and
should be removed
Context._msgData() (contracts/Context.sol#25-28) is never used and should be removed
SafeMath.div(uint256,uint256) (contracts/SafeMath.sol#108-110) is never used and
should be removed
SafeMath.div(uint256,uint256,string) (contracts/SafeMath.sol#124-134) is never used
and should be removed
SafeMath.mod(uint256,uint256) (contracts/SafeMath.sol#148-150) is never used and
should be removed
SafeMath.mod(uint256,uint256,string) (contracts/SafeMath.sol#164-171) is never used
and should be removed
SafeMath.mul(uint256,uint256) (contracts/SafeMath.sol#82-94) is never used and
should be removed
SafeMath.sub(uint256,uint256) (contracts/SafeMath.sol#47-49) is never used and
should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version0.6.12 (contracts/Address.sol#4) allows old versions
Pragma version0.6.12 (contracts/BEP20.sol#4) allows old versions
Pragma version0.6.12 (contracts/Context.sol#4) allows old versions
Pragma version0.6.12 (contracts/IBEP20.sol#4) allows old versions
Pragma version0.6.12 (contracts/Ownable.sol#4) allows old versions
Pragma version0.6.12 (contracts/SafeMath.sol#4) allows old versions
Pragma version0.6.12 (contracts/WalletikaToken.sol#4) allows old versions
solc-0.6.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-
versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (contracts/Address.sol#56-62):
        - (success) = recipient.call{value: amount}() (contracts/Address.sol#60)
Low level call in Address._functionCallWithValue(address,bytes,uint256,string)
(contracts/Address.sol#135-161):
        - (success,returndata) = target.call{value: weiValue}(data) (contracts/
Address.sol#144)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-
calls
INFO:Detectors:
Redundant expression "this (contracts/Context.sol#26)" inContext (contracts/
Context.sol#16-30)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-
statements
INFO:Detectors:
WalletikaToken.slitherConstructorVariables() (contracts/WalletikaToken.sol#8-38)
uses literals with too many digits:
        - _maxSupply = 100000000e18 (contracts/WalletikaToken.sol#9)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-
digits
INFO:Detectors:
WalletikaToken._maxSupply (contracts/WalletikaToken.sol#9) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-
variables-that-could-be-declared-constant
INFO:Slither:. analyzed (7 contracts with 85 detectors), 35 result(s) found
```

RUGDOG

WOOF!

rugdog.net

✉ the@rugdog.net