HAECHI AUDIT

MVL

Smart Contract Security Analysis
Published on : Jan 7, 2022
Version v1.0

HAECHI AUDIT

Smart Contract Audit Certificate

BEP20Mintable

Security Report Published by HAECHI AUDIT
v1.0 Jan 7, 2022

Auditor : Hoon Won

[illegible]

Executive Summary

Severity of Issues	Findings	Resolved	Unresolved	Acknowledged	Comment
Critical	-	-	-	-	-
Major	-	-	-	-	-
Minor	-	-	-	-	-
Tips	1	-	-	-	-

TABLE OF CONTENTS

0 Issues (0 Critical, 0 Major, 0 Minor) Found

ABOUT US

HAECHI AUDIT believes in the power of cryptocurrency and the next paradigm it will bring.

We have the vision to *empower the next generation of finance*. By providing security and trust in the blockchain industry, we dream of a world where everyone has easy access to blockchain technology.

HAECHI AUDIT is a flagship service of HAECHI LABS, the leader of the global blockchain industry. HAECHI AUDIT provides specialized and professional smart contract security auditing and development services.

We are a team of experts with years of experience in the blockchain field and have been trusted by 300+ project groups. Our notable partners include Universe,1inch, Klaytn, Badger, etc.

HAECHI AUDIT is the only blockchain technology company selected for the Samsung Electronics Startup Incubation Program in recognition of our expertise. We have also received technology grants from the Ethereum Foundation and Ethereum Community Fund.

Inquiries: audit@haechi.io

Website: audit.haechi.io

INTRODUCTION

This report was prepared to audit the security of the BEP20MintableToken smart contract created by MVL team. HAECHI AUDIT conducted the audit focusing on whether the smart contract created by MVL team is soundly implemented and designed as specified in the published materials, in addition to the safety and security of the smart contract.

CRITICAL	Critical issues must be resolved as critical flaws that can harm a wide range of users.
MAJOR	Major issues require correction because they either have security problems or are implemented not as intended.
MINOR	Minor issues can potentially cause problems and therefore require correction.
TIPS	Tips issues can improve the code usability or efficiency when corrected.

HAECHI AUDIT recommends MVL team improve all issues discovered.

The following issue explanation uses the format of {file name}#{line number}, {contract name}#{function/variable name} to specify the code. For instance, *Sample.sol:20* points to the 20th line of Sample.sol file, and *Sample#fallback()* means the fallback() function of the Sample contract.

Please refer to the Appendix to check all results of the tests conducted for this report.

SUMMARY

The codes used in this Audit can be found at GitHub (https://github.com/mvlchain/bmvl-contracts/blob/master/contracts/FlatBEP20Mintable.sol). The last commit of the code used for this Audit is"89b50a394a71235466a805baff7d3384aca16bd0".

Issues HAECHI AUDIT found 0 critical issues, 0 major issues, and 0 minor issues. There is 1 Tips issue explained that would improve the code's usability or efficiency upon modification.

Severity	Issue	Status
Notice	Additional tokens can be issued without restrictions	(Found - v1.0)
TIPS	There are missing Events	(Found - v1.0)

OVERVIEW

Contracts subject to audit

- ❖ Context
- ❖ AddressUpgradeable
- ❖ Initializable
- ❖ Ownable
- ❖ SafeMath
- ❖ BEP20Mintable

FINDINGS

Notice

Additional tokens can be issued without restrictions. (Found - v.1.0)

```
function mint(uint256 amount) public onlyOwner returns (bool) {
    _mint(_msgSender(), amount);
    return true;
}
```

[https://github.com/mvlchain/bmvl-contracts/blob/master/contracts/FlatBEP20Mintable.sol#L806-L809]

Owner can issue additional tokens without restrictions by calling the *BEP20Mintable#mint()* function.

💡 **TIPS**

There are missing Events. (Found - v.1.0)

The following list shows functions with missing Events.

Function	Expected Event	Emitted Event	Omitted Event
burn	Transfer, Burn	Transfer	Burn
burnFrom	Transfer, Burn, Approval	Transfer, Approval	Burn
mint	Transfer, Mint	Transfer	Mint

Without Event, it is difficult to identify in real-time whether correct values are recorded on the blockchain. In this case, it becomes problematic to determine whether the corresponding value has been changed in the application and whether the corresponding function has been called.

Thus, we recommend adding Events corresponding to the change occurring in the function.

DISCLAIMER

This report does not guarantee investment advice, the suitability of the business models, and codes that are secure without bugs. This report shall only be used to discuss known technical issues. Other than the issues described in this report, undiscovered issues may exist such as defects on Binance Smart Chain. In order to write secure smart contracts, correction of discovered problems and sufficient testing thereof are required.

Appendix A. Test Results

The following results show the unit test results covering the key logic of the smart contract subject to the security audit. Parts marked in red are test cases that failed to pass the test due to existing issues.

```
MVL
  #initialize()
    ✓ should set name properly
    ✓ should set symbol properly
    ✓ should set decimals properly
    ✓ should set initial supply properly
  ERC20 Spec
    #transfer()
      ✓ should fail if recipient is ZERO_ADDRESS
      ✓ should fail if sender's amount is lower than balance
      when succeeded
        ✓ sender's balance should decrease
        ✓ recipient's balance should increase
        ✓ should emit Transfer event
    #transferFrom()
      ✓ should fail if sender is ZERO_ADDRESS
      ✓ should fail if recipient is ZERO_ADDRESS
      ✓ should fail if sender's amount is lower than transfer amount
      ✓ should fail if allowance is lower than transfer amount
      ✓ should fail even if try to transfer sender's token without approval process
      when succeeded
        ✓ sender's balance should decrease
        ✓ recipient's balance should increase
        ✓ should emit Transfer event
        ✓ allowance should decrease
        ✓ should emit Approval event
    #approve()
      ✓ should fail if spender is ZERO_ADDRESS
      valid case
        ✓ allowance should set appropriately
        ✓ should emit Approval event
    #increaseAllowance()
      ✓ should fail if spender is ZERO_ADDRESS
      ✓ should fail if overflows
      valid case
```

```
        ✓ allowance should set appropriately
        ✓ should emit Approval event
    #decreaseAllowance()
      ✓ should fail if spender is ZERO_ADDRESS
      ✓ should fail if overflows
      valid case
        ✓ allowance should set appropriately
        ✓ should emit Approval event
  ERC20 Burnable spec
    #burn()
      ✓ should fail if try to burn more than burner's balance
      valid case
        ✓ totalSupply should decrease
        ✓ account's balance should decrease
        ✓ should emit Transfer event
        1) should emit Burn event
    #burnFrom()
      ✓ should fail if account is ZERO_ADDRESS
      ✓ should fail if account's amount is lower than burn amount
      ✓ should fail if allowance is lower than burn amount
      ✓ should fail even if try to burn account's this.token without approval process
      valid case
        ✓ totalSupply should decrease
        ✓ account's balance should decrease
        ✓ allowance should decrease
        ✓ should emit Transfer event
        2) should emit Burn event
        ✓ should emit Approval event
  ERC20 Mintable spec
    #mint()
      ✓ should fail if msg.sender is not owner
      valid case
        ✓ receiver's amount should increase
        ✓ totalSupply should increase
        ✓ should emit Transfer event
        3) should emit Mint event

Address
  #isContract()
    ✓ returns false for account address
    ✓ returns true for contract address
  #sendValue()
    ✓ should fail if try to transfer more than sender contract amounts
    when sender contract has ethers
```

```
      ✓ sends 0 wei
      ✓ sends non-zero amounts
      ✓ sends the whole balance
      ✓ should fail if try to send more than the amounts
    with contract recipient
      ✓ sends ether
      ✓ shoudl fail if recipient reverts
  #functionCall()
    with valid contract receiver
      ✓ calls the requested function
      ✓ reverts when the called function reverts with no reason
      ✓ reverts when the called function reverts, bubbling up the revert reason
      ✓ reverts when the called function runs out of gas
      ✓ reverts when the called function throws
      ✓ reverts when function does not exist
    with non-contract receiver
      ✓ reverts when address is not a contract
  #functionCallWithValue()
    with zero value
      ✓ calls the requested function
    with non-zero value
      ✓ reverts if insufficient sender balance
      ✓ calls the requested function with existing value
      ✓ calls the requested function with transaction funds
      ✓ reverts when calling non-payable functions
  #functionStaticCall()
    ✓ calls the requested function
    ✓ reverts on a non-static function
    ✓ bubbles up revert reason
    ✓ reverts when address is not a contract

Ownable
  #initialize()
    ✓ should set msg.sender to owner
  #transferOwnership()
    ✓ should fail if msg.sender is not owner
    ✓ should fail if try to transfer ownership to AddressZero
    valid case
      ✓ should change owner to newOwner
      ✓ should emit OwnershipTransferred event
  #renounceOwnership()
    ✓ should fail if msg.sender is not owner
    valid case
      ✓ should change owner to AddressZero
    ✓ should emit OwnershipTransferred event

SafeMath
  add
    ✓ adds correctly
    ✓ reverts on addition overflow
  sub
    ✓ subtracts correctly
    ✓ reverts if subtraction result would be negative
  mul
    ✓ multiplies correctly
    ✓ multiplies by zero correctly
    ✓ reverts on multiplication overflow
  div
    ✓ divides correctly
    ✓ divides zero correctly
    ✓ returns complete number result on non-even division
    ✓ reverts on division by zero
  mod
    ✓ reverts with a 0 divisor
    modulos correctly
      ✓ when the dividend is smaller than the divisor
      ✓ when the dividend is equal to the divisor
      ✓ when the dividend is larger than the divisor
      ✓ when the dividend is a multiple of the divisor
```

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/					
FlatBEP20Mintable.sol	100	100	100	100	

[Table 1] Test Case Coverage

End of Document