BLOCKHAT
SECURITY

BNBDog Inu

Smart Contract Security Audit

Prepared by BlockHat

March 30th, 2023 - April 1st, 2023

BlockHat.io

contact@blockhat.io

Document Properties

Client	Young Su
Version	0.1
Classification	Public

Scope

The BNBDog Inu Contract in the BNBDog Inu Repository

Repo	Owner
https://bscscan.com/address/0x67bc330e9f1cf01C37585c3f274690c01d892215#code	0x67bc330e9f1cf01C37585c3f274690c01d892215

Files	MD5 Hash
BNBDOGToken.sol	3bc75e58cc9dc64b68573d6ff0d4f5ad

Contacts

COMPANY	CONTACT
BlockHat	contact@blockhat.io

Contents

1 Introduction

BNBDog Inu engaged BlockHat to conduct a security assessment on the BNBDog Inu beginning on March 30th, 2023 and ending April 1st, 2023. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About BNBDog Inu

-

Issuer	Young Su
Website	https://bnbdoginu.com/
Type	Solidity Smart Contract
Audit Method	Whitebox

1.2 Approach & Methodology

BlockHat used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart contracts and can quickly detect code that does not comply with security best practices.

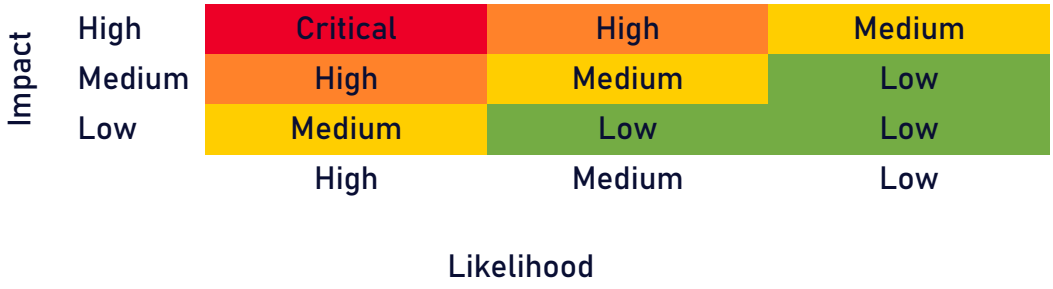
1.2.1 Risk Methodology

Vulnerabilities or bugs identified by BlockHat are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.

Its quantitative paradigm enables repeatable and precise measurement, while also revealing the underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to 1, with 5 indicating the greatest possibility or impact.

- Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed.
- Impact quantifies the technical and economic costs of a successful attack.
- Severity indicates the risk's overall criticality.

Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low, respectively. Severity is determined by probability and impact and is categorized into four levels, namely Critical, High, Medium, and Low.



2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the BNBDog Inu implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings

In general, these smart contracts are well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include , 1 high-severity, 2 low-severity vulnerabilities.

Vulnerabilities	Severity	Status
Owner can control fees	HIGH	Not Fixed
Missing address verification	LOW	Not Fixed
Floating Pragma	LOW	Not Fixed

3 Finding Details

A BNBDOGToken.sol

A.1 Owner can control fees [HIGH]

Description:

The owner can use the 'setTransferBurnRate' function to set a transfer burn rate value up to 100, which can result in the burning of all transferred tokens.

Code:

Listing 1: BNBDOGToken.sol

```
1080    function setTransferBurnRate(uint256 _tranferBurnRate) public
            ↪ onlyOwner {
1081        require(_tranferBurnRate <= 100, "Burning Rate on Transfer cannot
                ↪  be more than 100%");
1082        transferBurnRate = _tranferBurnRate;
1083    }
```

Risk Level:

Likelihood – 4
Impact - 5

Recommendation:

We recommend that the transferBurnRate value be limited to a lower percentage to mitigate the risk of excessive token burning.

Status - Not Fixed

A.2 Missing address verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument _transferBurnExceptAddress should include a zero-address test.

Code:

Listing 2: BNBDOGToken.sol

```
1089    function removeTransferBurnExceptAddress(address
            ↪ _transferBurnExceptAddress) public onlyOwner {
1090        delete _transferBurnExceptAddresses[_transferBurnExceptAddress];
1091    }
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

It is recommended to verify that the address provided in the arguments is different from the address(0) .

Status - Not Fixed

A.3 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.0 . Contracts should be deployed using the same compiler version and flags that were used during the testing process.Lock-

ing the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 3: BNBDOGToken.sol

```
7 pragma solidity ^0.8.0;
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Not Fixed

4 Best Practices

BP.1 Explicit Constructor Visibility and Placement

Description:

It is generally considered a best practice to declare the constructor at the beginning of the contract and to explicitly specify its visibility setting.

Code:

Listing 4: BNBDOGToken.sol

```
1093     constructor() public {
1094         _mint(msg.sender, _initial_supply);
1095      }
```

BP.2 Use super Keyword Only When Overriding Functions

Description:

When a contract inherits from a parent contract, it can call the parent contract's functions using the super keyword. However, using super unnecessarily can add unnecessary complexity and reduce code readability. Therefore, it's a best practice to use the super keyword only when overriding a function in the parent contract, and to avoid using it when there is no overridden function. By doing so, developers can write cleaner, more concise code that is easier to read and maintain.

Code:

Listing 5: BNBDOGToken.sol

```
1046     function _transfer(address sender, address recipient, uint256 amount
             ↪ ) internal virtual override {
```

```
1047        if (transferBurnRate > 0 && _transferBurnExceptAddresses[sender]
                ↪ != true && _transferBurnExceptAddresses[recipient] != true
                ↪  && recipient != address(0)) {
1048            uint256 _burntAmount = amount * transferBurnRate / 100;
1049            // Burn transferBurnRate% from amount
1050            super._burn(sender, _burntAmount);
1051            // Recalibrate the transfer amount
1052            amount = amount - _burntAmount;
1053        }
1054
1055        super._transfer(sender, recipient, amount);
1056    }
```

5 Static Analysis (Slither)

Description:

Block Hat expanded the coverage of the specific contract areas using automated testing methodologies. Slither, a Solidity static analysis framework, was one of the tools used. Slither was run on all-scoped contracts in both text and binary formats. This tool can be used to test mathematical relationships between Solidity instances statically and variables that allow for the detection of errors or inconsistent usage of the contracts' APIs throughout the entire codebase.

Results:

```
BNBDOGToken.setTransferBurnRate(uint256) (BNBDOGToken.sol#1080-1083)
   ↪ should emit an event for:
       - transferBurnRate = _tranferBurnRate (BNBDOGToken.sol#1082)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #missing-events-arithmetic

Address._revert(bytes,string) (BNBDOGToken.sol#446-458) uses assembly
       - INLINE ASM (BNBDOGToken.sol#451-454)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #assembly-usage

BNBDOGToken._transfer(address,address,uint256) (BNBDOGToken.sol
   ↪ #1042-1052) compares to a boolean constant:
       -transferBurnRate > 0 && _transferBurnExceptAddresses[sender] !=
          ↪ true && _transferBurnExceptAddresses[recipient] != true &&
          ↪  recipient != address(0) (BNBDOGToken.sol#1043)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #boolean-equality

Address._revert(bytes,string) (BNBDOGToken.sol#446-458) is never used
   ↪ and should be removed
```

```
Address.functionCall(address,bytes) (BNBDOGToken.sol#304-306) is never
   ↪ used and should be removed
Address.functionCall(address,bytes,string) (BNBDOGToken.sol#314-320) is
   ↪ never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (BNBDOGToken.sol
   ↪ #333-335) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (BNBDOGToken
   ↪ .sol#343-352) is never used and should be removed
Address.functionDelegateCall(address,bytes) (BNBDOGToken.sol#385-387) is
   ↪  never used and should be removed
Address.functionDelegateCall(address,bytes,string) (BNBDOGToken.sol
   ↪ #395-402) is never used and should be removed
Address.functionStaticCall(address,bytes) (BNBDOGToken.sol#360-362) is
   ↪ never used and should be removed
Address.functionStaticCall(address,bytes,string) (BNBDOGToken.sol
   ↪ #370-377) is never used and should be removed
Address.isContract(address) (BNBDOGToken.sol#255-261) is never used and
   ↪ should be removed
Address.sendValue(address,uint256) (BNBDOGToken.sol#279-284) is never
   ↪ used and should be removed
Address.verifyCallResult(bool,bytes,string) (BNBDOGToken.sol#434-444) is
   ↪  never used and should be removed
Address.verifyCallResultFromTarget(address,bool,bytes,string) (
   ↪ BNBDOGToken.sol#410-426) is never used and should be removed
Context._msgData() (BNBDOGToken.sol#575-577) is never used and should be
   ↪  removed
SafeMath.add(uint256,uint256) (BNBDOGToken.sol#96-98) is never used and
   ↪ should be removed
SafeMath.div(uint256,uint256) (BNBDOGToken.sol#138-140) is never used
   ↪ and should be removed
SafeMath.div(uint256,uint256,string) (BNBDOGToken.sol#190-195) is never
   ↪ used and should be removed
SafeMath.mod(uint256,uint256) (BNBDOGToken.sol#154-156) is never used
   ↪ and should be removed
```

```
SafeMath.mod(uint256,uint256,string) (BNBDOGToken.sol#212-217) is never
    ↪ used and should be removed
SafeMath.mul(uint256,uint256) (BNBDOGToken.sol#124-126) is never used
    ↪ and should be removed
SafeMath.sub(uint256,uint256) (BNBDOGToken.sol#110-112) is never used
    ↪ and should be removed
SafeMath.sub(uint256,uint256,string) (BNBDOGToken.sol#171-176) is never
    ↪ used and should be removed
SafeMath.tryAdd(uint256,uint256) (BNBDOGToken.sol#25-31) is never used
    ↪ and should be removed
SafeMath.tryDiv(uint256,uint256) (BNBDOGToken.sol#67-72) is never used
    ↪ and should be removed
SafeMath.tryMod(uint256,uint256) (BNBDOGToken.sol#79-84) is never used
    ↪ and should be removed
SafeMath.tryMul(uint256,uint256) (BNBDOGToken.sol#50-60) is never used
    ↪ and should be removed
SafeMath.trySub(uint256,uint256) (BNBDOGToken.sol#38-43) is never used
    ↪ and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Pragma version^0.8.0 (BNBDOGToken.sol#7) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (BNBDOGToken.sol
    ↪ #279-284):
        - (success) = recipient.call{value: amount}() (BNBDOGToken.sol
            ↪ #282)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
    ↪ string) (BNBDOGToken.sol#343-352):
        - (success,returndata) = target.call{value: value}(data) (
            ↪ BNBDOGToken.sol#350)
```

```
Low level call in Address.functionStaticCall(address,bytes,string) (
    ↪ BNBDOGToken.sol#370-377):
        - (success,returndata) = target.staticcall(data) (BNBDOGToken.sol
            ↪ #375)
Low level call in Address.functionDelegateCall(address,bytes,string) (
    ↪ BNBDOGToken.sol#395-402):
        - (success,returndata) = target.delegatecall(data) (BNBDOGToken.
            ↪ sol#400)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

Parameter BNBDOGToken.setTransferBurnRate(uint256)._tranferBurnRate (
    ↪ BNBDOGToken.sol#1080) is not in mixedCase
Parameter BNBDOGToken.addTransferBurnExceptAddress(address).
    ↪ _transferBurnExceptAddress (BNBDOGToken.sol#1085) is not in
    ↪ mixedCase
Parameter BNBDOGToken.removeTransferBurnExceptAddress(address).
    ↪ _transferBurnExceptAddress (BNBDOGToken.sol#1089) is not in
    ↪ mixedCase
Variable BNBDOGToken._initial_supply (BNBDOGToken.sol#1022) is not in
    ↪ mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

Variable BNBDOGToken.setTransferBurnRate(uint256)._tranferBurnRate (
    ↪ BNBDOGToken.sol#1080) is too similar to BNBDOGToken.
    ↪ transferBurnRate (BNBDOGToken.sol#1024)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #variable-names-are-too-similar

BNBDOGToken.slitherConstructorVariables() (BNBDOGToken.sol#1020-1096)
    ↪ uses literals with too many digits:
        - _initial_supply = 2000000000000000000e18 (BNBDOGToken.sol#1022)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #too-many-digits

BNBDOGToken._initial_supply (BNBDOGToken.sol#1022) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #state-variables-that-could-be-declared-constant

renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (BNBDOGToken.sol#634-636)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (BNBDOGToken.sol#642-645)
name() should be declared external:
        - ERC20.name() (BNBDOGToken.sol#714-716)
symbol() should be declared external:
        - ERC20.symbol() (BNBDOGToken.sol#722-724)
decimals() should be declared external:
        - ERC20.decimals() (BNBDOGToken.sol#739-741)
totalSupply() should be declared external:
        - ERC20.totalSupply() (BNBDOGToken.sol#746-748)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (BNBDOGToken.sol#753-755)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (BNBDOGToken.sol#765-769)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (BNBDOGToken.sol#788-792)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (BNBDOGToken.sol
            ↪ #810-815)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (BNBDOGToken.sol
            ↪ #829-833)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (BNBDOGToken.sol
            ↪ #849-858)
```

```
burn(uint256) should be declared external:
        - BNBDOGToken.burn(uint256) (BNBDOGToken.sol#1059-1061)
burnFrom(address,uint256) should be declared external:
        - BNBDOGToken.burnFrom(address,uint256) (BNBDOGToken.sol
            ↪ #1074-1077)
setTransferBurnRate(uint256) should be declared external:
        - BNBDOGToken.setTransferBurnRate(uint256) (BNBDOGToken.sol
            ↪ #1080-1083)
addTransferBurnExceptAddress(address) should be declared external:
        - BNBDOGToken.addTransferBurnExceptAddress(address) (BNBDOGToken.
            ↪ sol#1085-1087)
removeTransferBurnExceptAddress(address) should be declared external:
        - BNBDOGToken.removeTransferBurnExceptAddress(address) (
            ↪ BNBDOGToken.sol#1089-1091)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #public-function-that-could-be-declared-external
BNBDOGToken.sol analyzed (8 contracts with 78 detectors), 59 result(s)
    ↪ found
```

Conclusion:

Most of the vulnerabilities found by the analysis have already been addressed by the smart contract code review. w

6 Conclusion

We examined the design and implementation of BNBDog Inu in this audit and found several issues of various severities. We advise Young Su team to implement the recommendations contained in all 3 of our findings to further enhance the code's security. It is of utmost priority to start by addressing the most severe exploit discovered by the auditors then followed by the remaining exploits, and finally we will be conducting a re-audit following the implementation of the remediation plan contained in this report.

We would much appreciate any constructive feedback or suggestions regarding our methodology, audit findings, or potential scope gaps in this report.

BLOCKHAT
SECURITY

For a Contract Audit, contact us at contact@blockhat.io