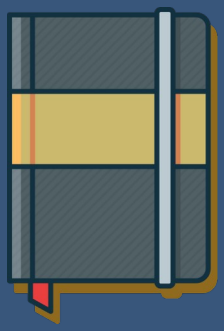
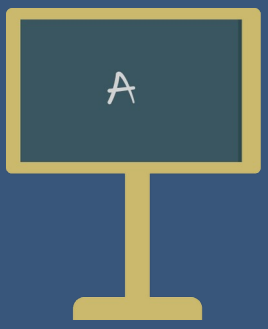
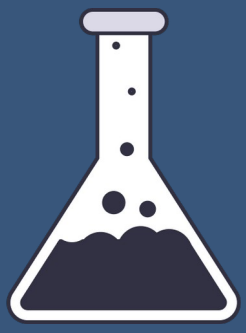
Perseus Audit

SpaceMonkey Contracts October 2021

Contents







Introduction

Audit:

In October 2021 Perseus's audit report division performed an audit for the SpaceMonkey Contract.

https://bscscan.com/address/0x9298f766581650e81357f5C66C91cc003f2b75Aa

Introduction

Overview:

Information:

Name: SpaceMonkey

Pool, Asset or Contract address:
https://bscscan.com/address/0x9298f766581650e81357f5C66C91cc003f2b75Aa

Supply:

Current: NA

Explorers:
https://bscscan.com/address/0x9298f766581650e81357f5C66C91cc003f2b75Aa

Websites: https://officialspacemonkey.com/

Links: https://twitter.com/SPMKGame

Introduction

Compiler related issues:

It is best practice to use the latest version of the solidity compiler supported by the toolset you use. This so it includes all the latest bug fixes of the solidity compiler. When you use for instance the openzeppelin contracts in your code the solidity version you should use should be 0.8.0 because this is the latest version supported.

Caution:

The solidity versions used for the audited contracts can be 0.6.0 --> 0.8.0 these versions have for instance the following known bugs so the compiled contract might be susceptible to:

EmptyByteArrayCopy – Medium risk

Copying an empty byte array (or string) from memory or calldata to storage can result in data corruption if the target array's length is increased subsequently without storing new data.
https://etherscan.io/solcbuginfo?a=EmptyByteArrayCopy

DynamicArrayCleanup – Medium risk

When assigning a dynamically-sized array with types of size at most 16 bytes in storage causing the assigned array to shrink, some parts of deleted slots were not zeroed out.
https://etherscan.io/solcbuginfo?a=DynamicArrayCleanup

Advice:

Update the contracts to the latest supported version of solidity by your contract. And set it as a fixed parameter not a floating pragma.

Audit Report Scope

Assertions and Property Checking:

1. Solidity assert violation.
2. Solidity AssertionFailed event.

ERC Standards:

3. Incorrect ERC20 implementation.

Solidity Coding Best Practices:

4. Outdated compiler version.
5. No or floating compiler version set.
6. Use of right-to-left-override control character.
7. Shadowing of built-in symbol.
8. Incorrect constructor name.
9. State variable shadows another state variable.
10. Local variable shadows a state variable.
11. Function parameter shadows a state variable.
12. Named return value shadows a state variable.
13. Unary operation without effect Solidity code analysis.
14. Unary operation directly after assignment.
15. Unused state variable.
16. Unused local variable.
17. Function visibility is not set.
18. State variable visibility is not set.
16. Use of deprecated functions: call code(), sha3(), ...
17. Use of deprecated global variables (msg.gas, ...).
18. Use of deprecated keywords (throw, var).
19. Incorrect function state mutability.
20. Does the code conform to the Solidity styleguide.

Convert code to conform Solidity styleguide:

21. Convert all code so that it is structured accordingly the Solidity styleguide.

Audit Report Scope

Categories:

High Severity:

High severity issues opens the contract up for exploitation from malicious actors. We do not recommend deploying contracts with high severity issues.

Medium Severity Issues:

Medium severity issues are errors found in contracts that hampers the effectiveness of the contract and may cause outcomes when interacting with the contract. It is still recommended to fix these issues.

Low Severity Issues:

Low severity issues are warning of minor impact on the overall integrity of the contract. These can be fixed with less urgency.

Audit Report

7	7	0
Identified	Confirmed	Critical

0	0	7
High	Medium	Low

Analysis:
https://bscscan.com/address/0x9298f766581650e81357f5C66C91cc003f2b75Aa

Risk:
Low

Low severity:

shadowing-local
SpaceMonkey.allowance(address,address).owner
(contracts/SPMK/SpaceMonkey.sol#65) shadows:
- Ownable.owner() (contracts/library/Ownable.sol#16-18) (function)

shadowing-local
SpaceMonkey._approve(address,address,uint256).owner
(contracts/SPMK/SpaceMonkey.sol#192) shadows:
- Ownable.owner() (contracts/library/Ownable.sol#16-18) (function)

missing-zero-check
SpaceMonkey.setSecondOwner(address).newSecond
(contracts/SPMK/SpaceMonkey.sol#594) lacks a zero-check on :
- _secondOwner = newSecond (contracts/SPMK/SpaceMonkey.sol#595)

missing-zero-check
SpaceMonkey.setMarketingAddress(address)._marketingAddress
(contracts/SPMK/SpaceMonkey.sol#645) lacks a zero-check on :
- marketingAddress = address(_marketingAddress)
(contracts/SPMK/SpaceMonkey.sol#646)

missing-zero-check
SpaceMonkey.setTeamAddress(address)._teamAddress (contracts/SPMK/
SpaceMonkey.sol#649) lacks a zero-check on :
- teamAddress = address(_teamAddress)
(contracts/SPMK/SpaceMonkey.sol#650)

Low severity:

Reentrancy in SpaceMonkey.transfer(address,uint256,bytes)
(contracts/SPMK/SpaceMonkey.sol#52-63):
External calls:
- recipient.isContract() (contracts/SPMK/SpaceMonkey.sol#53)
State variables written after the call(s):
- _transfer(_msgSender(),recipient,amount)
(contracts/SPMK/SpaceMonkey.sol#57)
- _burnFee = _previousBurnFee (contracts/SPMK/SpaceMonkey.sol#578)
- _burnFee = 0 (contracts/SPMK/SpaceMonkey.sol#572)
- _transfer(_msgSender(),recipient,amount)
(contracts/SPMK/SpaceMonkey.sol#57)
- _liquidityFee = _liquidityFee * fee / 100
(contracts/SPMK/SpaceMonkey.sol#582)
- _liquidityFee = _previousLiquidityFee
(contracts/SPMK/SpaceMonkey.sol#577)
- _liquidityFee = 0 (contracts/SPMK/SpaceMonkey.sol#571)
- _transfer(_msgSender(),recipient,amount)
(contracts/SPMK/SpaceMonkey.sol#57)
- _previousBurnFee = _burnFee (contracts/SPMK/SpaceMonkey.sol#568)
- _transfer(_msgSender(),recipient,amount)
(contracts/SPMK/SpaceMonkey.sol#57)
- _previousLiquidityFee = _liquidityFee
(contracts/SPMK/SpaceMonkey.sol#567)
- _transfer(_msgSender(),recipient,amount)
(contracts/SPMK/SpaceMonkey.sol#57)
- _previousTaxFee = _taxFee (contracts/SPMK/SpaceMonkey.sol#566)

Low severity:

Reentrancy-benign

- _transfer(_msgSender(),recipient,amount)
(contracts/SPMK/SpaceMonkey.sol#57)
- _rOwned[recipient] = _rOwned[recipient] + (rRecipientAmount)
(contracts/SPMK/SpaceMonkey.sol#408)
- _rOwned[deadAddress] = _rOwned[deadAddress] + (rBurnAmount)
(contracts/SPMK/SpaceMonkey.sol#409)
- _rOwned[sender] = _rOwned[sender] - (rAmount)
(contracts/SPMK/SpaceMonkey.sol#264)
- _rOwned[marketingAddress] += rOurShare
(contracts/SPMK/SpaceMonkey.sol#530)
- _rOwned[sender] = _rOwned[sender] - (rAmount)
(contracts/SPMK/SpaceMonkey.sol#298)
- _rOwned[sender] = _rOwned[sender] - (rAmount)
(contracts/SPMK/SpaceMonkey.sol#334)
- _rOwned[sender] = _rOwned[sender] - (rAmount)
(contracts/SPMK/SpaceMonkey.sol#369)
- _rOwned[teamAddress] += rTeamShare
(contracts/SPMK/SpaceMonkey.sol#538)
- _transfer(_msgSender(),recipient,amount)
(contracts/SPMK/SpaceMonkey.sol#57)
- _rTotal = _rTotal - (rFee) (contracts/SPMK/SpaceMonkey.sol#440)
- _transfer(_msgSender(),recipient,amount)
(contracts/SPMK/SpaceMonkey.sol#57)
- _tFeeTotal = _tFeeTotal + (tFee) (contracts/SPMK/SpaceMonkey.sol#441)
- _transfer(_msgSender(),recipient,amount)
(contracts/SPMK/SpaceMonkey.sol#57)
- _tOwned[recipient] = _tOwned[recipient] + (tRecipientAmount)
(contracts/SPMK/SpaceMonkey.sol#398)
- _tOwned[deadAddress] = _tOwned[deadAddress] + (tBurnAmount)
(contracts/SPMK/SpaceMonkey.sol#399)
- _tOwned[sender] = _tOwned[sender] - (tAmount)
(contracts/SPMK/SpaceMonkey.sol#333)

Low severity:

Reentrancy-benign

- _tOwned[sender] = _tOwned[sender] - (tAmount)
(contracts/SPMK/SpaceMonkey.sol#368)
- _tOwned[marketingAddress] += tOurShare
(contracts/SPMK/SpaceMonkey.sol#532)
- _tOwned[teamAddress] += tTeamShare
(contracts/SPMK/SpaceMonkey.sol#541)
- _transfer(_msgSender(),recipient,amount)
(contracts/SPMK/SpaceMonkey.sol#57)
- _taxFee = _previousTaxFee (contracts/SPMK/SpaceMonkey.sol#576)
- _taxFee = _taxFee * fee / 100 (contracts/SPMK/SpaceMonkey.sol#583)
- _taxFee = 0 (contracts/SPMK/SpaceMonkey.sol#570)

Low severity:

Reentrancy in SpaceMonkey.transfer(address,uint256,bytes)
(contracts/SPMK/SpaceMonkey.sol#52-63):
External calls:
- recipient.isContract() (contracts/SPMK/SpaceMonkey.sol#53)
Event emitted after the call(s):
- Transfer(sender,recipient,balanceOf(recipient) - (recipientBalBefore))
(contracts/SPMK/SpaceMonkey.sol#425-429)
- _transfer(_msgSender(),recipient,amount)
(contracts/SPMK/SpaceMonkey.sol#57)
- Transfer(sender,deadAddress,balanceOf(deadAddress) -
(deadBalBefore)) (contracts/SPMK/SpaceMonkey.sol#431-435)
- _transfer(_msgSender(),recipient,amount)
(contracts/SPMK/SpaceMonkey.sol#57)
- Transfer(sender,marketingAddress,tOurShare)
(contracts/SPMK/SpaceMonkey.sol#533)
- _transfer(_msgSender(),recipient,amount)
(contracts/SPMK/SpaceMonkey.sol#57)
- Transfer(sender,marketingAddress,tokenFromReflection(rOurShare))
(contracts/SPMK/SpaceMonkey.sol#535)
- _transfer(_msgSender(),recipient,amount)
(contracts/SPMK/SpaceMonkey.sol#57)
- Transfer(sender,teamAddress,tTeamShare)
(contracts/SPMK/SpaceMonkey.sol#540)
- _transfer(_msgSender(),recipient,amount)
(contracts/SPMK/SpaceMonkey.sol#57)
- Transfer(sender,teamAddress,tokenFromReflection(rTeamShare))
(contracts/SPMK/SpaceMonkey.sol#543)
- _transfer(_msgSender(),recipient,amount)
(contracts/SPMK/SpaceMonkey.sol#57)

Conclusion

We performed the procedures as laid out in the scope of the audit and there were 7 findings, 7 low. We recommend fixing the high issues as soon as possible.

Disclaimer

Perseus audit is not a security warranty, investment advice, or an endorsement of the Space Monkey protocol. 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 the Space Monkey protocol team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.