HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Mogul
Date: April 23rd, 2021

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

This document may contain confidential information about IT systems and the intellectual property of the Customer as well as information about potential vulnerabilities and methods of their exploitation.

The report containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed - upon a decision of the Customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for Mogul - Second Review
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	NFT with Marketplace
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Zip-archive	https://github.com/mogulproductions/nft-marketplace-contracts/commit/f04607c827b9c1eac2066f00bc369ad4bea3860f
Timeline	20 APRIL 2021 - 23 APRIL 2021
Changelog	21 APRIL 2021 - INITIAL AUDIT 23 APRIL 2021 - SECOND REVIEW

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by Mogul (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer's smart contract and its code review conducted on April 23rd, 2021.

Scope

The scope of the project is the following solidity contracts:
https://github.com/mogulproductions/nft-marketplace-contracts/commit/f04607c827b9c1eac2066f00bc369ad4bea3860f
contracts/MogulMarketplace.sol (**md5:**e63885915a4c7de50a9eebd07b180c7a)
contracts/MogulNFT.sol (**md5:**c0da89f5e3fd17d52c377b651295508a)

We have scanned these smart contracts for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Asset’s integrity ▪ User Balances manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

According to the assessment, the Customer's smart contract is well-secured.

Insecure	Poor secured	Secured	Well-secured

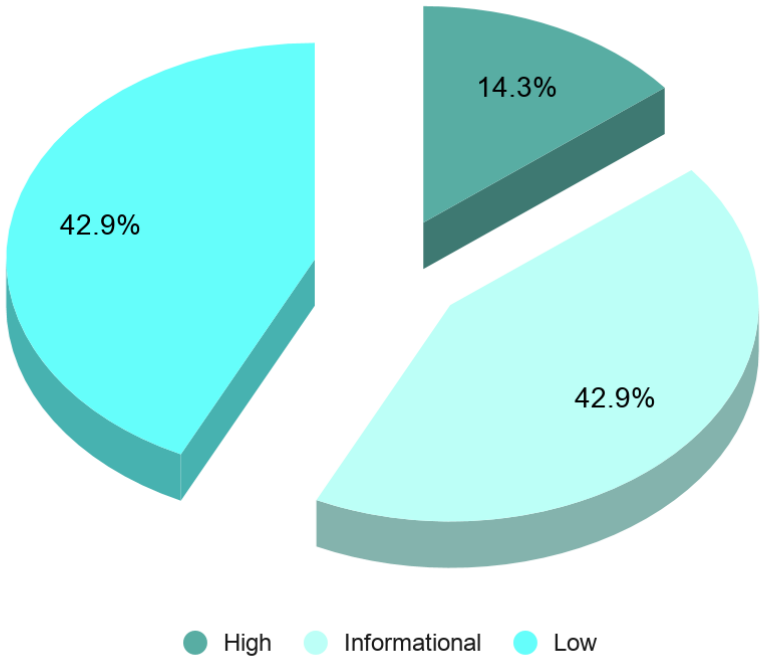
You are here

Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. A general overview is presented in AS-IS section, and all found issues can be found in the Audit overview section.

Security engineers found **1** high, **3** Low and **3** informational issues during the first review.

Security engineers found **no issues** during the second review.

Graph 1. The distribution of vulnerabilities after the first review.



Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

AS-IS overview

MogulMarketplace.sol

Description
MogulMarketplace is a contract for NFT Marketplace.

Imports
MogulMarketplace has following imports:
- import "@openzeppelin/contracts/token/ERC1155/IERC1155.sol"
- import "@openzeppelin/contracts/token/ERC20/IERC20.sol"
- import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol"
- import "@openzeppelin/contracts/token/ERC1155/utils/ERC1155Holder.sol"
- import "@openzeppelin/contracts/access/AccessControl.sol"
- import "@openzeppelin/contracts/utils/math/SafeMath.sol"
- import "@openzeppelin/contracts/utils/structs/EnumerableSet.sol"
- import "@openzeppelin/contracts/security/ReentrancyGuard.sol"
- import "@chainlink/contracts/src/v0.6/interfaces/AggregatorV3Interface.sol"

Inheritance
MogulMarketplace is ERC1155Holder, AccessControl and ReentrancyGuard

Usages
MogulMarketplace has following usages:
- EnumerableSet for EnumerableSet.UintSet
- SafeERC20 for IERC20
- SafeMath for uint256

Structs
MogulMarketplace has following structures:
- Listing – stores marketplace listing info
- Auction – stores auction info
- Bid – stores auction bid info

Enums
MogulMarketplace has no enums

Events
MogulMarketplace has following events:
- event ListingCreated(
  string label,
  address tokenAddress,
  uint256 tokenId,
  uint256 numTokens,

```
        uint256 price,
        bool isStarsListing);
```

- `event Sale(address buyer, uint256 listingId, uint256 amount);`
- `event AuctionEnded(address winner, uint256 auctionId);`
- `event AuctionCancelled(uint256 auctionId);`

Modifiers

MogulMarketplace has the following modifier:

- ```
  modifier onlyAdmin {
      require(hasRole(ROLE_ADMIN, msg.sender), "Sender is not admin");
      _;
  }
  ```

Fields

MogulMarketplace has following fields and constants:

- `AggregatorV3Interface priceOracle`
- `bytes32 public constant ROLE_ADMIN = keccak256("ROLE_ADMIN")`
- `address payable public treasuryWallet`
- `IERC20 stars`
- `uint256 nextListingId = 0`
- `mapping(uint256 => Listing) public listings`
- `EnumerableSet.UintSet private listingIds`
- `mapping(uint256 => Auction) public auctions`
- `EnumerableSet.UintSet private auctionIds`

Functions

MogulMarketplace has following public and external functions

- *`constructor`*

  Description

  Initializes the contract. Assigns the ROLE_ADMIN role to the provided _admin address. Stores provided _treasuryWallet address into the treasuryWallet state variable. Stores provided starsAddress as IERC20 into the stars state variable

  Input parameters
  - address starsAddress
  - address _admin
  - address payable _treasuryWallet

  Constraints
  - Protected by ReentrancyGuard
  - _treasuryWallet should not be zero-address

  Events emit

  Emits RoleGranted and RoleAdminChanged events.

  Output

  None
  ```

- *setPriceOracle*
  Description
  Stores provided priceOracleAddress as AggregatorV3Interface into the priceOracle state variable

  Input parameters
  - address priceOracleAddress

  Constraints
  - onlyAdmin modifier used

  Events emit
  None
  Output
  None

- *listTokens*
  Description
  Creates a new listing

  Input parameters
  - string memory label
  - address tokenAddress
  - uint256 tokenId
  - uint256 numTokens
  - uint256 price
  - bool isStarsListing

  Constraints
  - onlyAdmin modifier used
  - Protected by ReentrancyGuard

  Events emit
  Emits ListingCreated event
  Output
  None

- *removeListing*
  Description
  Removes a listing

  Input parameters
  - uint256 listingId

  Constraints
  - onlyAdmin modifier used

  Events emit
  Emits ListingCreated event
  Output
  None

- *buyTokens*
  Description
  Buy a token. Payable function.

  Input parameters
  - uint256 listingId
  - uint256 amount

  Constraints
  - Protected by ReentrancyGuard
  - listingIds should contain provided listingId
  - listing.numTokens should be greater or equal to the provided amount
  - sent msg.value should be equal listing.price multiplied by the provided amount in case of listing is not the stars listing

  Events emit
  Emits Sale event
  Output
  None

- *startAuction*
  Description
  Start an auction

  Input parameters
  - string memory label
  - address tokenAddress
  - uint256 tokenId
  - uint256 numTokens
  - uint256 startingStarsPrice
  - uint256 startingEthPrice
  - uint256 startTime
  - uint256 endTime
  - bool allowStarsBids
  - bool allowEthBids

  Constraints
  - onlyAdmin modifier used
  - Protected by ReentrancyGuard
  - allowStarsBids or allowEthBids should be true

  Events emit
  None
  Output
  None

- *bid*
  Description
  Send in a bid and refund the previous highest bidder. Payable function

  Input parameters

- uint256 auctionId
- bool isStarsBid
- uint256 amount

Constraints

- Protected by ReentrancyGuard
- auction should be already started
- in case of starsBid: auction should have allowStarsBids set true
- in case of starsBid: provided amount should be higher than highestStarsBid and also be higher than auction's startingStarsPrice
- in case of not starsBid: auctions should have allowEthBids set true
- in case of not starsBid: provided amount should be higher than highestEthBid and also be higher than auction's startingEthPrice
- in case of not starsBid: provided amount should be equal to the sent msg.value

Events emit

None

Output

None

- *endAuctionWithoutOracle*

Description

End auctions and reward the winner without needing a price Oracle. The caller chooses whether the Stars bid or Ether bid was higher.

Input parameters

- uint256 auctionId
- bool didStarsBidWin

Constraints

- Protected by ReentrancyGuard
- onlyAdmin modifier used
- auctionId should exist in the auctionIds
- timestamp should be greater or equal to auction's endTime
- auction should have allowStarsBids set true if didStarsBidWin is true or allowEthBids otherwise

Events emit

Emits AuctionEnded event

Output

None

- *endAuction*

Description

End auctions and reward the winner. If the auction supported both Stars and eth bids, uses the oracle to determine who won

Input parameters

- uint256 auctionId

Constraints

- onlyAdmin modifier used
- Protected by ReentrancyGuard
- auctionId should exist in the auctionIds
- timestamp should be greater or equal to auction's endTime

Events emit
Emits AuctionEnded event
Output
None

- *cancelAuction*
  Description
  Cancel auction and refund bidders

  Input parameters
  - uint256 auctionId

  Constraints
  - onlyAdmin modifier used
  - Protected by ReentrancyGuard
  - auctionId should exist in the auctionIds

  Events emit
  Emits AuctionCancelled event
  Output
  None

- *withdrawETH*
  Description
  Withdraw ETH to treasury wallet

  Input parameters
  None
  Constraints
  - onlyAdmin modifier used
  - should be no running auctions

  Events emit
  None
  Output
  None
- *withdrawStars*
  Description
  Withdraw Stars to treasury wallet

  Input parameters
  None
  Constraints
  - onlyAdmin modifier used
  - should be no running auctions

  Events emit

None
Output
None

- *getStarsPrice, supportsInterface, getNumListings, getListingIds, getListingAtIndex, getNumAuctions, getAuctionIds, getAuctionAtIndex,*
  Description
  Simple View functions.

MogulNFT.sol

Description
MogulNFT is an ERC1155 contract

Imports
MogulNFT has following imports:
- import "@openzeppelin/contracts/token/ERC1155/ERC1155.sol"
- import "@openzeppelin/contracts/access/AccessControl.sol"

Inheritance
MogulNFT is ERC1155 and AccessControl

Usages
MogulNFT has no usages

Structs
MogulNFT has no structures

Enums
MogulNFT has no enums

Events
MogulNFT has no custom events

Modifiers
MogulNFT has the following modifier:
- modifier onlyAdmin {
  require(hasRole(ROLE_ADMIN, msg.sender), "Sender is not admin");
  _;
  }

Fields
MogulNFT has following fields and constants:
- bytes32 public constant ROLE_ADMIN = keccak256("ROLE_ADMIN");

- uint256 nextTokenId = 0;

Functions

MogulNFT has following public and external functions

- *constructor*
  Description
  Initializes the contract. Assigns the ROLE_ADMIN role to the provided _admin address.

  Input parameters
  - address _admin

  Constraints
  None
  Events emit
  Emits RoleGranted and RoleAdminChanged events.
  Output
  None

- *setUri*
  Description
  Sets a new URI for all token types, by relying on the token type ID substitution mechanism

  Input parameters
  - string memory newUri

  Constraints
  - onlyAdmin modifier used

  Events emit
  None
  Output
  None

- *mintToken*
  Description
  Mint a new ERC1155 Token

  Input parameters
  - address recipient,
  - uint256 amount,
  - bytes memory data

  Constraints
  - onlyAdmin modifier used

  Events emit
  None
  Output
  None

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Audit overview

Critical

No Critical severity issues were found.

High

1. **Vulnerability:** Re-entrancy bug
   Contracts: MogulMarketplace
   Method: buyTokens(uint256, uint256)

   State variable updated after calling an external function.

   Fixed before second review

Medium

No Medium severity issues were found.

Low

1. **Vulnerability:** Missing zero address validation.
   Contract: MogulMarketplace

   No checking for zero address for _treasuryWallet in the constructor. In case of zero address provided there is no ability to change it later, therefore withdrawETH and withdrawStars functions will fail.

   Fixed before second review

2. **Vulnerability:** Benign reentrancy.
   Contract: MogulMarketplace

   listTokens and startAuction contain a reentrancy. The reentrancy is benign because it's exploitation would have the same effect as two consecutive calls.

   Fixed before second review

3. **Vulnerability:** Events reentrancy.
   Contract: MogulMarketplace

buyTokens, endAuction, endAuctionWithoutOracle and listTokens contain a reentrancy. If such reentrancies happen, the corresponding events fired by functions will be shown in an incorrect order, which might lead to issues for third parties.

Fixed before second review

Lowest / Code style / Best Practice

1. **Vulnerability:** send / transfer reentrancy.
   Contract: MogulMarketplace

   send and *transfer* do not protect from reentrancies in case of gas price changes

   Fixed before second review

2. **Vulnerability:** Unused state variable
   Contract: MogulMarketplace

   nextAuctionId is defined but never used in the contract

   Fixed before second review

3. **Vulnerability:** Public function that could be declared external
   Contracts: MogulMarketplace, MogulNFT

   public functions that are never called by the contract should be declared **external** to save gas.

   Fixed before second review

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, high-level description of functionality was presented in As-Is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security engineers found **1** high, **3** Low and **3** informational issues during the first review.

Security engineers found **no issues** during the second review.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.