mogul

SMART CONTRACT AUDIT

ZOKYO.

Mar 23, 2021 | v. 1.0

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

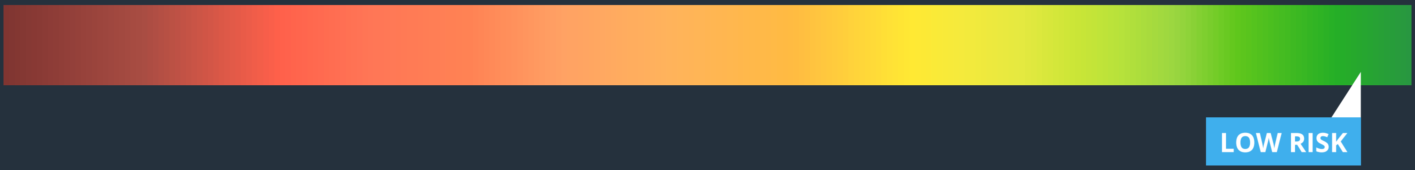
GOOD

TECHNICAL SUMMARY

This document outlines the overall security of the Mogul smart contracts, evaluated by Zokyo's Blockchain Security team.

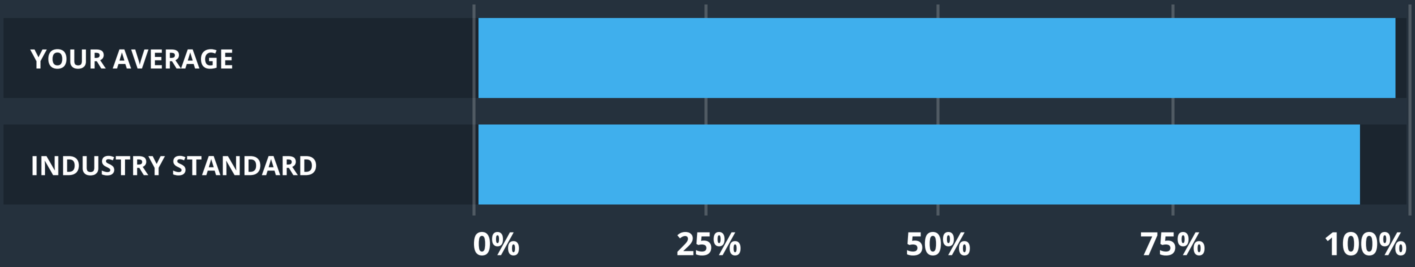
The scope of this audit was to analyze and document the Mogul smart contract codebase for quality, security, and correctness.

Contract Status



There was one critical issue found during the audit. After the report issued by Zokyo, Mogul team has resolved the issue.

Testable Code



Testable code is 98.81%, which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the Mogul team put in place a bug bounty program to encourage further and active analysis of the smart contract.

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the following github repository – https://github.com/mogulproductions/smart-contracts-matic/tree/main/contracts.
Our review focused on the commit hash – 3d2be783278155cd79d43edd425e9ee2da425d51.

Requirements:

Litepaper: https://docsend.com/view/6nuft4bkqz3b8ub7
STARS Info: https://docsend.com/view/abgmqp7fhysda98d

Throughout the review process, care was taken to ensure that the token contract:

- Implements and adheres to existing Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of Mogul smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1 Due diligence in assessing the overall code quality of the codebase.

3 Testing contract logic against common and uncommon attack vectors.

2 Cross-comparison with other, similar smart contracts by industry leaders.

4 Thorough, manual review of the codebase, line-by-line.

SUMMARY

There was one critical issue found during the audit, that was successfully resolved by the Mogul development team. All the mentioned findings may have an effect only in case of specific conditions performed by the contract owner.

Contracts are well written and structured. The findings during the audit have no impact on contract performance or security, so it is fully production-ready.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

● Critical

The issue affects the ability of the contract to compile or operate in a significant way.

● High

The issue affects the ability of the contract to compile or operate in a significant way.

● Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

● Low

The issue has minimal impact on the contract’s ability to operate.

● Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

Stars.sol wrong constructor arguments

CRITICAL | RESOLVED

"_initialHolders" variable was listed twice.

Recommendation:
change variables to match constructor's expectation:

- address _admin;
- address[] memory _initialHolders;
- uint256[] memory **_premintedAmounts**;
- uint256 _inflationBasisPoint;
- uint256 _mintLockPeriodSecs.

Re-audit:
Mogul development team has fixed the issue. Zokyo team has reviewed the issue mentioned above and confirms that it was successfully resolved.

Stars.sol no SafeMath applied

HIGH | RESOLVED

There are several calculations in the code that potentially could lead to under or overflows.

Examples:

- remainingYearlyInflationAmt -= amount; [line 77]
- nextMintStartTime += mintLockPeriodSecs; [line 97]
- remainingYearlyInflationAmt =
  (totalSupplyThisYear * _inflationBasisPoint) / 10000; [line 50-52]
- remainingYearlyInflationAmt =
  (totalSupplyThisYear * inflationBasisPoint) / 10000; [line 99-101]

Recommendation:
Example can be seen in AccessPassSale.sol contract:

- import "@openzeppelin/contracts/math/SafeMath.sol"; [line 4]
- using SafeMath for uint256; [line 9]
- newAmount.sub(currAmount) [line 134]

Re-audit:
Mogul development team has fixed the issue. Zokyo team has reviewed the issue mentioned above and confirms that it was successfully resolved.

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Mogul team

```
Contract: AccessPassSale
    ✓ sets withdraw wallet (192ms)
    ✓ initializes tiers correctly (297ms)
    ✓ doesn't allow initTiers if tiers 1-5 are non-empty (145ms)
    ✓ sets tiers correctly (361ms)
    ✓ Purchases work correctly (209ms)
    ✓ pauses & unpauses tiers (227ms)
    ✓ withdraws ether (162ms)
    ✓ withdraws from select tiers (202ms)
    ✓ withdraws all stars (163ms)
    ✓ grants & revokes admin permissions (144ms)

Contract: AccessPassSale
    ✓ Should init properly (154ms)
    ✓ Add Guardians (161ms)
    ✓ Remove Guardians (135ms)
    ✓ Change owner by owner (92ms)
    ✓ Create Change Owner Proposal (109ms)
    ✓ Add Vote Change Owner Proposal (132ms)
    ✓ Remove Vote Change Owner Proposal (171ms)
    ✓ Change Owner By Guardian (178ms)
    ✓ Set Min Guardian Votes Required (45ms)
    ✓ Approve ERC20 (89ms)
    ✓ Transfer ERC20 (73ms)
    ✓ Transfer from ERC20 (116ms)
    ✓ Approve ERC721 (125ms)
    ✓ Approve for all ERC721 (107ms)
    ✓ Transfer from ERC721 (267ms)
    ✓ Safe transfer from ERC721 (262ms)
    ✓ Safe transfer from with data ERC721 (250ms)
    ✓ Approve for all ERC1155 (113ms)
    ✓ Safe transfer from ERC1155 (238ms)
    ✓ Safe batch transfer from ERC115 (307ms)
    ✓ Transfer Native Currency (4091ms)
    ✓ Change Owner By Owner then create change owner proposal (128ms)
    ✓ Lock and Unlock wallet (10149ms)
    ✓ Wallet compromised (187ms)

  Contract: Mogul Smart Wallet Factory
    ✓ Should init properly
    ✓ Predict Mogul Smart Wallet Deterministic
    ✓ Deploy Mogul Smart Wallet Deterministic (7233ms)
    ✓ Change owner (45ms)

  Contract: Stars
    ✓ Should init properly (126ms)
    ✓ Mint (10097ms)
    ✓ Update Year Period (30486ms)
    ✓ Pause (126ms)
    ✓ Burn (65ms)
    ✓ Change Admin (44ms)
    ✓ Whitelist (238ms)

  45 passing (2m)
```

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts/	100.00	95.00	100.00	100.00	
AccessPassSale.sol	100.00	87.50	100.00	100.00	
MogulSmartWallet.sol	100.00	95.59	100.00	100.00	
MogulSmartWalletFactory.sol	100.00	100.00	100.00	100.00	
Stars.sol	100.00	100.00	100.00	100.00	
contracts/interfaces/	100.00	100.00	100.00	100.00	
IMogulSmartWallet.sol	100.00	100.00	100.00	100.00	
contracts/matic/	11.11	12.50	16.67	10.00	

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
BasicMetaTransaction.sol	11.11	12.50	16.67	10.00	... 104, 105, 106
contracts/mocks/	100.00	100.00	100.00	100.00	
MockERC1155.sol	100.00	100.00	100.00	100.00	
MockERC20.sol	100.00	100.00	100.00	100.00	
MockERC721.sol	100.00	100.00	100.00	100.00	
MockInitializable.sol	100.00	100.00	100.00	100.00	
contracts/openzeppelinModified/	67.11	50.00	60.00	67.11	
AccessControl.sol	35.29	20.00	36.36	35.29	... 173, 202, 203
ERC20.sol	78.05	50.00	61.11	78.05	... 190, 191, 288
ERC20Burnable.sol	25.00	100.00	50.00	25.00	37, 39, 40
ERC20Pausable.sol	100.00	100.00	100.00	100.00	
ERC20PresetMinterPauser.sol	81.82	75.00	85.71	81.82	51, 55
contracts/utils/	45.07	28.57	43.59	44.87	
AddressModified.sol	0.00	0.00	0.00	0.00	... 176, 180, 185
Context.sol	40.00	50.00	50.00	28.57	18, 19, 20, 30, 31
EnumerableSet.sol	64.71	50.00	50.00	65.71	... 274, 281, 295
Pausable.sol	100.00	100.00	100.00	100.00	
All files	**76.71**	**71.43**	**68.61**	**75.84**	

Tests written by Zokyo Secured team

As part of our work assisting Mogul in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Truffle testing framework.

Tests were based on the functionality of the code, as well as a review of the Mogul contract requirements for details about issuance amounts and how the system handles these.

```
Contract: AccessControl
  default admin
    ✓ deployer has default admin role (48ms)
    ✓ other roles's admin is the default admin role
    ✓ default admin role's admin is itself
  granting
    ✓ admin can grant role to other accounts (86ms)
    ✓ non-admin cannot grant role to other accounts (237ms)
    ✓ accounts can be granted a role
  revoking
    ✓ roles that are not had can be revoked (56ms)
    with granted role
      ✓ admin can revoke role (78ms)
      ✓ non-admin cannot revoke role (39ms)
      ✓ a role can be revoked multiple times (102ms)
  renouncing
    ✓ roles that are not had can be renounced
    with granted role
      ✓ bearer can renounce role (60ms)
      ✓ only the sender can renounce their roles
      ✓ a role can be renounced multiple times (97ms)
  setting role admin
    ✓ a role's admin role can be changed
    ✓ the new admin can grant roles (109ms)
    ✓ the new admin can revoke roles (63ms)
    ✓ a role's previous admins no longer grant roles
    ✓ a role's previous admins no longer revoke roles
  getting role
    ✓ should return correct count role member count (58ms)
    ✓ should return correct role member (57ms)
```

```
Contract: AccessPassSale
  ✓ Only admin can set withdraw wallet
  ✓ Should set withdraw wallet correctly (38ms)
  ✓ Only admin can init tiers
  ✓ Cannot init tiers if tiers 1-5 are not empty (83ms)
  ✓ Should init tiers correctly (143ms)
  ✓ Cannot make purchape if transaction value is incorrect (144ms)
  ✓ Cannot make purchape if it have been paused (138ms)
  ✓ Purchase should work correctly (195ms)
  ✓ Only admin can withdraw ethers
  ✓ Withdraws ether should work correctly (149ms)
  ✓ Only admin can remove tiers
  ✓ Remove tiers (195ms)
  ✓ Only admin can reduce passes from tiers
  ✓ withdraws from select tiers (178ms)

Contract: Address
  isContract
    ✓ returns false for account address
    ✓ returns true for contract address
  sendValue
    when sender contract has no funds
      ✓ sends 0 wei
      ✓ reverts when sending non-zero amounts (176ms)
    when sender contract has funds
      ✓ sends 0 wei
      ✓ sends non-zero amounts
      ✓ sends the whole balance
      ✓ reverts when sending more than the balance
  functionCall
    with valid contract receiver
      ✓ calls the requested function
      ✓ reverts when the called function reverts with no reason (52ms)
      ✓ reverts when the called function reverts, bubbling up the revert reason (38ms)
      ✓ reverts when the called function runs out of gas (63ms)
      ✓ reverts when the called function throws (64ms)
      ✓ 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
        ✓ everts if insufficient sender balance
        ✓ calls the requested function with existing value (45ms)
        ✓ calls the requested function with transaction funds
        ✓ reverts when calling non-payable functions (46ms)
  functionStaticCall
    ✓ calls the requested function
    ✓ reverts on a non-static function (39ms)
    ✓ bubbles up revert reason
    ✓ reverts when address is not a contract
  functionDelegateCall
    ✓ delegate calls the requested function
    ✓ bubbles up revert reason
    ✓ reverts when address is not a contract

Contract: Context
  ✓ returns the transaction sender when called from an EOA
  ✓ returns the transaction sender when called from an EOA
  ✓ returns the transaction from Context address
  ✓ returns the transaction data when called from an EOA
  ✓ returns the transaction sender when from another contract

Contract: ERC20
  ✓ Should deploy with correct details
  ✓ Cannot mint to the zero address
  ✓ Mint should work correctly (56ms)
  ✓ Cannot burn from the zero address
  ✓ Burn should work correctly (55ms)
  ✓ Cannot approve to the zero address
  ✓ Cannot approve from the zero address
  ✓ approve should work correctly (41ms)
  ✓ Cannot transfer to the zero address
  ✓ transfer should work correctly (74ms)
  ✓ Cannot transfer from the zero address
```

```
    ✓ transferFrom should work correctly (74ms)
    ✓ setupDecimals should work correctly
    ✓ increaseAllowance() (38ms)
    ✓ decreaseAllowance() (57ms)

Contract: ERC20PresetMinterPauser
    ✓ Cannot mint if sender is not admin
    ✓ Mint should work correctly (44ms)
    ✓ Burn should work correctly (56ms)
    ✓ BurnFrom should work correctly (87ms)
    ✓ Cannot pause if caller not admin
    ✓ Should pause correctly (39ms)
    ✓ Cannot unpause if caller not admin
    ✓ Should unpause correctly (56ms)
    ✓ Cannot unpause if caller not admin
    ✓ Cannot change admin if caller not admin
    ✓ changeAdmin() should work correctly
    ✓ Should check _beforeTokenTransfer correctly
    ✓ Should change whitelist wtatus correctly (53ms)

Contract: Stars
    ✓ Cannot deploy with wrong lengths of arrays (55ms)
    ✓ Should deploy with correct details (55ms)
    ✓ Cannot call mint() if caller nit admit
    ✓ Cannot mint tokens if minting too many tokens for this year
    ✓ Mint should work correctly (68ms)
    ✓ Burn should work correctly (64ms)
    ✓ Cannot update year period if caller nit admi
    ✓ Cannot update if year period is not over
    ✓ updateYearPeriod() should work correctly (98ms)
    ✓ Cannot change whitelist status if caller not admin
    ✓ changeWhitelistStatus() should work correctly (62ms)
    ✓ Cannot change admin if caller not admin
    ✓ changeAdmin() should work correctly
    ✓ Cannot pause if caller not admin
    ✓ Should pause correctly (43ms)
    ✓ Cannot unpause if caller not admin
    ✓ Should pause correctly (83ms)
    ✓ increaseAllowance()
    ✓ decreaseAllowance() (78ms)
    ✓ BurnFrom should work correctly (115ms)


  115 passing (36s)
```

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts/	96.83	85.71	100.00	98.41	
AccessPassSale.sol	95.45	75.00	100.00	97.73	137
Stars.sol	100.00	100.00	100.00	100.00	
contracts/openzeppelinModified/	100.00	100.00	100.00	100.00	
AccessControl.sol	100.00	100.00	100.00	100.00	
ERC20.sol	100.00	100.00	100.00	100.00	
ERC20Burnable.sol	100.00	100.00	100.00	100.00	
ERC20Pausable.sol	100.00	100.00	100.00	100.00	
ERC20PresetMinterPauser.sol	100.00	100.00	100.00	100.00	
contracts/utils/	100.00	94.44	100.00	100.00	
AddressModified.sol	100.00	93.75	100.00	100.00	
Context.sol	100.00	100.00	100.00	100.00	
All files	**98.81**	**93.59**	**100.00**	**99.42**	

We are grateful to have been given the opportunity to work with the Mogul team.

ZOKYO.