SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

Mars Launchpad

AUDIT

SECURITY ASSESSMENT

23. September, 2024

FOR

Introduction

SolidProof.io is a brand of the officially registered company MAKE Network GmbH, based in Germany. We're mainly focused on Blockchain Security such as Smart Contract Audits and KYC verification for project teams.
Solidproof.io assess potential security issues in the smart contracts implementations, review for potential inconsistencies between the code base and the whitepaper/documentation, and provide suggestions for improvement.

Disclaimer

SolidProof.io reports are not, nor should be considered, an "endorsement"or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc'...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof's position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of the security or functionality of the technology we agree to analyze.

Project Overview

Summary

Project Name	Mars launchpad
Website	https://www.marslaunchpad.org/
About the project	Mars Launchpad is a unique Web3 ecosystem where you can easily find the best Web3 projects. These projects are carefully chosen and reviewed by our active community, ensuring a high-quality and enjoyable Web3 experience for every user, regardless of their background.
Chain	Binance smart chain
Language	Solidity
Codebase	Token:https://bscscan.com/address/0x145335c5fd3d53e2a361d995015a055484167666#code Presale:https://bscscan.com/address/0xF62dd37139e996977492Bc450c6cb4E139c054e9#code
Commit	N/A
Unit Tests	Not Provided

Social Medias

Telegram	https://t.me/MarsLaunchpad
Twitter	https://x.com/MarsLaunchpad
Facebook	N/A
Instagram	N/A
GitHub	N/A
Reddit	N/A
Medium	N/A
Discord	N/A
YouTube	N/A
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Change Log
v1.0	20. September 2024	· Layout Project · Automated/Manual-Security Testing · Summary
v1.1	23. September 2024	· Reaudit

Note – The following audit report presents a comprehensive security analysis of the smart contract utilized in the project that includes outside manipulation of the contract's functions in a malicious way. This analysis did not include functional testing (or unit testing) of the contract/s logic. We cannot guarantee 100% logical correctness of the contract as we did not functionally test it. This includes internal calculations in the formulae used in the contract.

File Overview

The Team provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an SHA-1 Hash.

File Name	SHA-1 Hash
contracts/Mars.sol	236e6815a1d49f8b9c4fdacc83fbded6ad84ad73
contracts/Presale.sol	91bc568577de4412153476bee0b4aa5e258791bc

Please note: Files with a different hash value than in this table have been modified after the security check, either intentionally or unintentionally. A different hash value may (but need not) be an indication of a changed state or potential vulnerability that was not the subject of this scan.

Imported packages.

Used code from other Frameworks/Smart Contracts.

N/A

Note for Investors: We only audited contracts mentioned in the scope above. All contracts related to the project apart from that are not a part of the audit, and we cannot comment on its security and are not responsible for it in any way. The presale of this project is going to be launched on their website, so it is recommended that research is done before investing.

External/Public functions

External/public functions are functions that can be called from outside of a contract, i.e., they can be accessed by other contracts or external accounts on the blockchain. These functions are specified using the function declaration's external or public visibility modifier.

State variables

State variables are variables that are stored on the blockchain as part of the contract'sstate. They are declared at the contract level and can be accessed and modified by any function within the contract. State variables can be needed within visibility modifier, such as public, private or internal, which determines the access level of the variable.

Components

📝 Contracts	📚 Libraries	🔍 Interfaces	🎨 Abstract
3	0	5	4

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐 Public	💰 Payable
84	2

External	Internal	Private	Pure	View
67	76	12	1	28

StateVariables

Total	🌐 Public
63	49

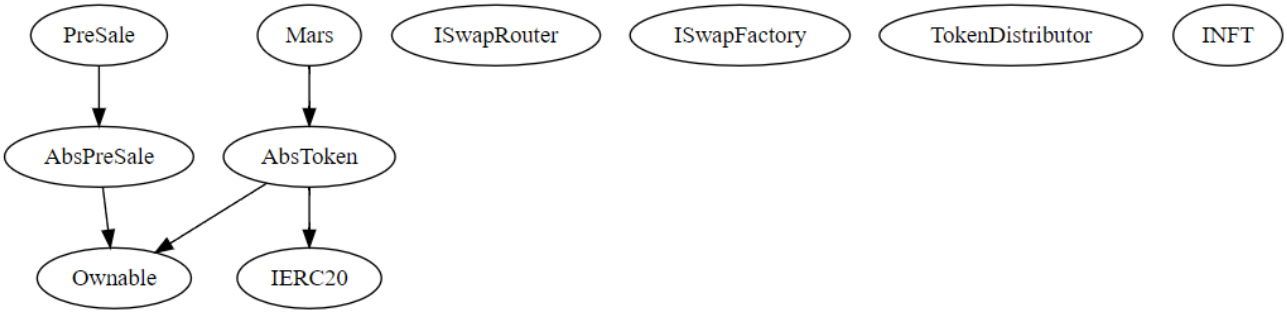
Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
^0.8.19	----------	Yes		----------

Transfers ETH	Low-Level Calls	Delegate Call	Uses Hash Functions	ECRecover	New/Create/Create2
Yes			Yes		yes → NewContract:TokenDistributor

Inheritance Graph

An inheritance graph is a graphical representation of the inheritance hierarchy among contracts. In object-oriented programming, inheritance is a mechanism that allows one class (or contract, in the case of Solidity) to inherit properties and methodsfrom another class. It shows the relationships between different contracts and how they are related to each other through inheritance.



Audit Information

Vulnerability & Risk Level

Risk represents the probability that a certain source threat will exploit the vulnerability and the impact of that event on the organization or system.The risk level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executingthe contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to check the repository for security-related issues, code quality, and compliance with specifications and best practices. To this end, our team of experienced pen-testers andsmart contract developers reviewed the code line by line and documented any issues discovered.

We check every file manually. We use automated tools only so that they help us achieve faster and better results.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   a. Reviewing the specifications, sources, and instructions provided to
   SolidProof to ensure we understand the size, scope, and functionality of the
   smart contract.
   b. Manual review of the code, i.e., reading the source code line by line to identify potential vulnerabilities.
   c. Comparison to the specification, i.e., verifying that the code does what is described in the specifications, sources, and instructions provided to SolidProof.

2. Testing and automated analysis that includes the following:
   a. Test coverage analysis determines whether test cases cover code and how much code is executed when those test cases are executed.
   b. Symbolic execution, which is analysing a program to determine what inputs cause each part of a program to execute.

3. Review best practices, i.e., review smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on best practices, recommendations, and research from industry and academia.

4. Concrete, itemized and actionable recommendations to help you secure your smart contracts.

Overall Security

Upgradeability

Contract is not upgradable	✅ **Deployer cannot update the contract with new functionalities.**
Description	The contract is not an upgradeable contract. The Deployer is not able to change or add any functionalities to the contract after deploying.
Comment	N/A

Ownership

Contract ownership is not renounced.	❌ **The ownership is not renounced.**
Description	The owner has not renounced the ownership that means that the owner retains control over the contract's operations, including the ability to execute functions that may impact the contract's users or stakeholders. This can lead to several potential issues, including: • Centralizations • The owner has significant control over contract's operations.
Comment	N/A

Note – *The contract cannot be considered as renounced till it is not deployed or having some functionality that can change the state of the contract.*

Ownership Privileges

These functions can be dangerous. Please note that abuse can lead to financial loss. We have a guide where you can learn more about these Functions.

Minting tokens

Minting tokens refer to the process of creating new tokens in a cryptocurrency or blockchain network. This process is typically performed by the project's owner or designated authority, who has the ability to add new tokens to the network's total supply.

Contract owner cannot mint new tokens.	✅ **The owner cannot mint new tokens.**
Description	The owner is not able to mint new tokens once the contract is deployed.
Comment	N/A

Burning tokens

Burning tokens is the process of permanently destroying a certain number of tokens, reducing the total supply of a cryptocurrency or token. This is usually done to increase the value of the remaining tokens, as the reduced supply can create scarcity and potentially drive up demand.

Contract owner cannot burn tokens	✅ **The owner cannot burn tokens.**
Description	The owner is not able burn tokens without any allowances.
Comment	N/A

Blacklist addresses

Blacklisting addresses in smart contracts is the process of adding a certain address to a blacklist, effectively preventing them from accessing or participating in certain functionalities or transactions within the contract. This can be useful in preventing fraudulent or malicious activities, such as hacking attempts or money laundering.

Contract owner can blacklist addresses.	❌ **The owner can blacklist wallets.**
Description	The owner can blacklist wallets from transferring tokens.
Comment	The owner of the contract can blacklist any wallets from transferring tokens from the contract for an indefinite period of time, which is not recommended as this can cause the locking of funds for the user indefinitely. It is recommended that there must be a locking period so that the user will not be locked for an indefinite period of time.

File/Line(s): L555-566

Codebase: Mars.sol

```
ftrace | funcSig
function setBlackList(address addr, bool enable) external onlyOwner {
    _blackList[addr] = enable;
}

ftrace | funcSig
function batchSetBlackList(
    address[] memory addr,
    bool enable
) external onlyOwner {
    for (uint i = 0; i < addr.length; i++) {
        _blackList[addr[i]] = enable;
    }
}
```

Fees and Tax

In some smart contracts, the owner or creator of the contract can set fees for certain actions or operations within the contract. These fees can be used to cover the cost of running the contract, such as paying for gas fees or compensating the contract's owner for their time and effort indeveloping and maintaining the contract.

Contract owner can levy high taxes	❌ **The owner can set fees more than 25%.**
Description	The owner can set fees of more than 25%.
Comment	The owner of the contract can set any arbitrary value as the buy, sell, and transfer fees in the contract, which is not recommended as the fees in the contract should not be more than 25% in the contract. Also, the owner of the contract can update the invite fees to any arbitrary value in the contract, as this can increase the price of buying the token. It is recommended that there must be a check so that the value cannot be set to any arbitrary value in the contract.

File/Line(s): L473-493

Codebase: Mars.sol

```
function setBuyFee(uint256 fundFee) external onlyOwner {
    _buyFundFee = fundFee;
}

ftrace | funcSig
function setSellFee(
    uint256 nft1Fee,
    uint256 nft2Fee,
    uint256 nft3Fee,
    uint256 lpFee
) external onlyOwner {
    _sellNFT1Fee = nft1Fee;
    _sellNFT2Fee = nft2Fee;
    _sellNFT3Fee = nft3Fee;
    _sellLPFee = lpFee;
}

uint256 public _transferFee = 0;

ftrace | funcSig
function setTransferFee(uint256 fee) external onlyOwner {
    _transferFee = fee;
}
```

File/Line(s): L320-322

Codebase: Presale.sol

```
ftrace | funcSig
function setInviteFee(uint256 fee) external onlyOwner {
    _inviteFee = fee;
}
```

Lock User Funds

In a smart contract, locking refers to the process of restricting access to certain tokens or assets for a specified period of time. When token or assets are locked in a smart contract, they cannot be transferred or used until the lock-up period has expired or certain conditions have been met.

Contract owner can lock function.	❌ **The owner can lock function.**
Description	The owner can lock the claim.
Comment	The contract contains the functionality in which the owner of the contract can lock the claimed functionality in the contract, which can lock the funds, and the user will not be able to claim their tokens, which is not recommended. There must be a locking period, or the claimed functionality should not be locked in an indefinite period. Also, the owner of the contract can set any arbitrary value at the end time of presale as this can increase the presale period if the valuer is set to an excessive number in the contract. There must be a check so that the value cannot be set to any arbitrary value in the contract.

Centralization Privileges

Centralization can arise when one or more parties have privileged access or control over the contract's functionality, data, or decision-making. This can occur, for example,if the contract is controlled by a single entity or if certain participants have special permissions or abilities that others do not.

In the project, there are authorities that have access to the following functions:

File	Privileges
Mars.sol	➢ The owner can update the fund address. ➢ The owner can update any arbitrary amount in the minimum swap amount value. ➢ The owner can update any arbitrary fee in the buy, sell, and transfer fees in the contract. ➢ The owner can enable trading only once. ➢ The owner can whitelist multiple addresses from fees. ➢ The owner can update the pair list in the contract. ➢ The owner can update the reward gas between 200000 to 2000000. ➢ The owner can blacklist wallets from transferring tokens. ➢ The owner can set the nft1reward minimum reward value in the contract. ➢ The owner can set any arbitrary address as the nft1 contract address. ➢ The owner can set any arbitrary value in the nft2 minimum reward value in the contract. ➢ The owner can set any arbitrary address as the nft2 contract address. ➢ The owner can update the nft3 reward value in the contract. ➢ The owner can set the nft3 contract address.
Presale.sol	➢ The owner can set any arbitrary address as the token address in the contract. ➢ The owner can update any arbitrary address as the cash address in the contract. ➢ The owner can claim tokens to the recipient wallet address. ➢ The owner can enable/disable the buying functionality in the contract.

- The owner can enable/disable the claim functionality in the contract.
- The owner can set any arbitrary value in the invite fee for the contract.
- The owner can set any arbitrary value in the minimum buy and maximum buying amount, including zero.
- The owner can update the end-time value in the contract.
- The owner can update any arbitrary value in the quantity of tokens.

Recommendations

To avoid potential hacking risks, it is advisable for the client to manage the private key of the privileged account with care. Additionally, we recommend enhancing the security practices of centralized privileges or roles in the protocol through a decentralized mechanism or smart-contract-based accounts, such as multi-signature wallets.

Here are some suggestions of what the client can do:

- Consider using multi-signature wallets: Multi-signature wallets require multiple parties to sign off on a transaction before it can be executed, providing an extra layer of security e.g. Gnosis Safe
- Use of a timelock at least with a latency of e.g. 48-72 hours for awareness of privileged operations
- Introduce a DAO/Governance/Voting module to increase transparency and user involvement
- Consider Renouncing the ownership so that the owner cannot modify any state variables of the contract anymore. Make sure to set up everything before renouncing.

Audit Result

Critical Issues

No critical issues

High Issues

#1 | Unsafe use of tx.origin.

File	Severity	Location	Status
Mars.sol	High	L115	ACK

Description – The contract contains the functionality in which the tx.origin is used to authenticate the user, which is not recommended as tx.origin is the address that initiated the transaction. An attacker could influence a user to execute a malicious transaction on the attacker's contract and then call a function to authenticate the target contract. It is best practice to use msg.sender instead of tx.origin in the contract for authentication.

#2 | The owner can pause claim.

File	Severity	Location	Status
Presale.sol	High	L316-318	ACK

Description – The contract contains the functionality in which the owner of the contract can pause the claim functionality of the token mid-sale which is not recommended as this can lock the user from claiming tokens for an indefinite period of time. It is recommended that the claim functionality should not be locked in the contract indefinitely.

Alleviation – The project owner confirms it as a part of business logic. Hence, marking it to acknowleged.

Medium Issue

#1 | Missing Threshold.

File	Severity	Location	Status
Presale.sol	Medium	L324-326, L328-330, L336-340	ACK

Description – The owner can update any arbitrary value in the minimum, maximum, and token quantity value, including zero, which is not recommended as this can lock the buying and claiming functionality in the contract if the value is set to zero. It is recommended that there must be a threshold value in the contract so that the number of tokens for buying and claiming cannot get zero.

#2 | The owner can update the token address mid-sale.

File	Severity	Location	Status
Presale.sol	Medium	L295-301	ACK

Description – The contract contains the functionality in which the owner of the contract can update any arbitrary address as the token and cash address, including zero or dead address, which is not recommended as this can cause the loss of funds for the user if the value is set to any arbitrary address. It is recommended that the contract address cannot be set mid-sale to avoid these circumstances in the contract.

#3 | The owner can update reward address mid-sale.

File	Severity	Location	Status
Presale.sol	Medium	L623-625, L633-635, L708-710, L718-720, L775-777, L785-787	ACK

Description – The owner of the contract can update the reward NFT contracts to any arbitrary address which is not recommended as this can cause the loss of funds for the user of the address is set to any arbitrary address in the contract. There must be a check in which the owner of the contract cannot update the value mid sale to avoid these circumstances in the contract.

#4 | Liquidity is added to externally owned address.

File	Severity	Location	Status
Mars.sol	Medium	L442-452	ACK

Description – The contract's liquidity is automatically added to the 'fundAddress' wallet address, which is not recommended because, in an extreme scenario, this can be used to drain liquidity from the contract. Liquidity should be added to the contract or dead address so that any wallet cannot drain it from the contract.

Low Issue

#1 | Local variable shadowing.

File	Severity	Location	Status
Mars.sol	Low	L263, 289, 290	ACK

Description – It is recommended that the component that shadows another component be renamed.

#2 | Floating pragma solidity version.

File	Severity	Location	Status
All	Low	L7	ACK

Description – Adding the constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

#3 | Missing events arithmetic.

File	Severity	Location	Status
Mars.sol	Low	L464-515, L540-543, L555-557, L623-625, L633-635, L708-710, L718-720, L775-777, L785-787	ACK
Presale.sol	Low	L295-301, L312-341	ACK

Description – It is recommended to emit all the critical parameter changes.

#4 | Missing zero or dead address check.

File	Severity	Location	Status
Mars.sol	Low	L464-467, L513-515, L633-635, L718-720, L785-787	ACK
Presale.sol	Low	L295-301	ACK

Description – It is recommended to check that the address cannot be set to zero or dead address.

Informational Issue

#1 | Natspec documentation missing.

File	Severity	Location	Status
All	Informational	--	ACK

Description – If you started to comment on your code, also comment on all other functions, variables, etc.

Legend for the Issue Status

Attribute or Symbol	Meaning
Open	The issue is not fixed by the project team.
Fixed	The issue is fixed by the project team.
Acknowledged(ACK)	The issue has been acknowledged or declared as part of business logic.

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY