SOLIDProof

*Bring trust into your projects*

**Blockchain Security | Smart Contract Audits | KYC**

MADE IN GERMANY

# Mountain Miner

# Audit

## Security Assessment

## 21. May, 2022

**For**

MOUNTAIN MINER IN SPACE

Disclaimer 3
Description 5
Project Engagement 5
Logo 5
Contract Link 5
Methodology 8
Used Code from other Frameworks/Smart Contracts (direct imports) 9
Tested Contract Files 10
Source Lines 11
Risk Level 11
Capabilities 12
Inheritance Graph 13
CallGraph 14
Scope of Work/Verify Claims 15
Modifiers and public functions 18
Source Units in Scope 21
Critical issues 22
High issues 22
Medium issues 22
Low issues 22
Informational issues 22
Audit Comments 23
SWC Attacks 24

# 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 security or functionality of the technology we agree to analyze.

| Version | Date | Description |
|---|---|---|
| 1.0 | 18. May 2022 | • Layout project<br>• Automated- /Manual-Security Testing<br>• Summary |
| 1.1 | 21. May 2022 | • Reaudit |

## Network
Binance Smart Chain (BEP20)

## Website
https://mountainminer.app/

## Telegram
https://t.me/paramountstaker

## Twitter
https://twitter.com/prmntfi

## Medium
https://medium.com/@paramountstaker

## Description
TBA

## Project Engagement
During the 14th of May 2022, **Mountain Miner Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

## Logo



## Contract Link
### v1.0
- Testnet
  - https://testnet.bscscan.com/address/0xcC6918E89E91A1F18558503449C28b8Cb0689E5B#code

### v1.1
- https://ftmscan.com/address/0xad1BA7A0335565dF7aAf57396eaAe7c5D809BDbe#code
- https://polygonscan.com/address/0xE26adfF25b4AFDA14119388f027aD54E2426A7ae#code
- https://cronoscan.com/address/0xad1BA7A0335565dF7aAf57396eaAe7c5D809BDbe#code
- https://snowtrace.io/address/0xa60A47C7c6F0b67930b6E4DfF0bfC9C78087B04F#code
- https://snowtrace.io/address/0xe26adff25b4afda14119388f027ad54e2426a7ae#code
- https://bscscan.com/address/0x6FA261A5e015a93a3B50Ab3CD9b80F95c5a0C30e#code

- https://bscscan.com/address/0xa59537bcb905c228f7EA4F64e7E70f429eBfa838#code

# Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. 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 executing the 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 evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

## Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
    i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
    ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.

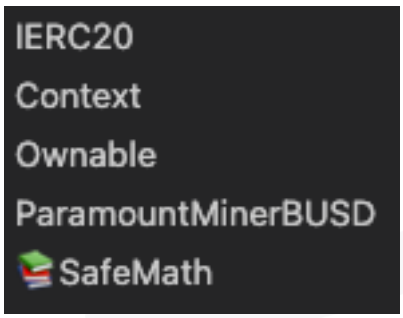
2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

## Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:



## Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

*A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.*

### v1.0

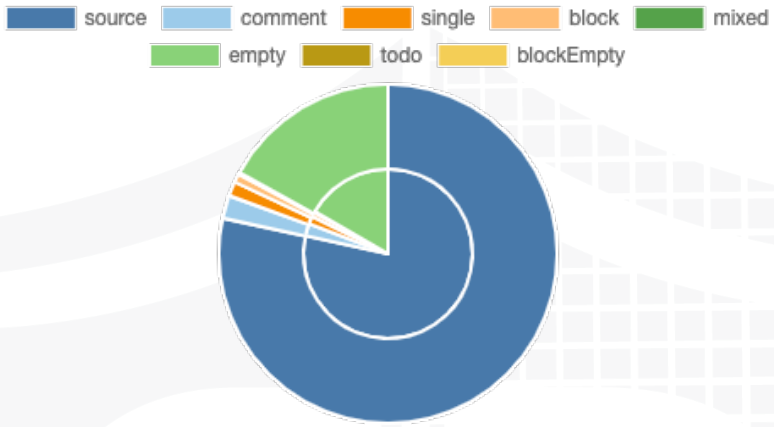
| File Name | SHA-1 Hash |
| --- | --- |
| contracts/paramountminer.sol | e3c29cf4013ebaf8830ae8c76284cb63fed77d9f |

### v1.1

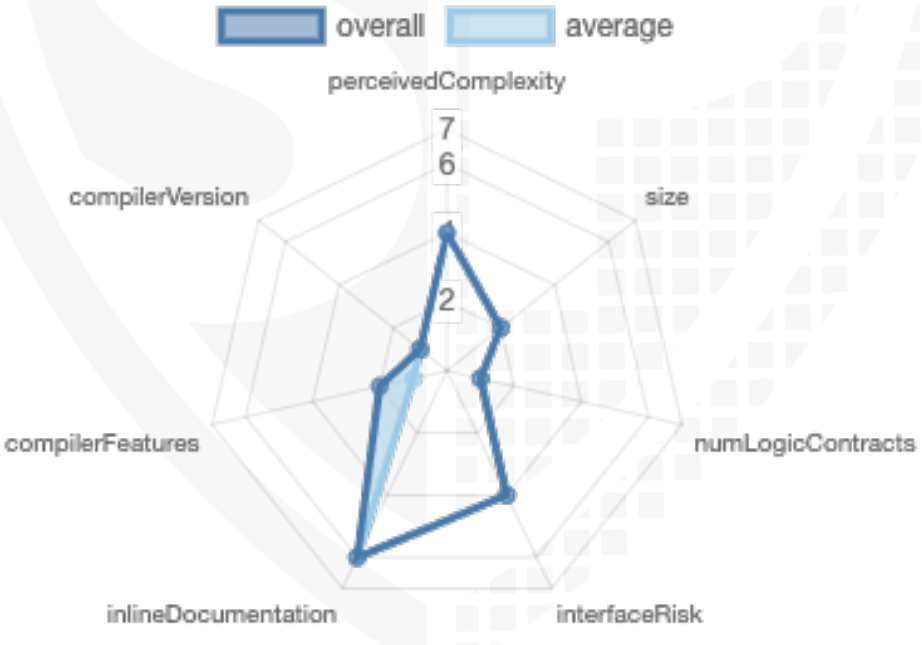
| File Name | SHA-1 Hash |
| --- | --- |
| contracts/BUSD miner.sol | 7c3a6eacd491441ac72e9c207e873d813d06229d |

# Metrics

## Source Lines
### v1.0




## Risk Level
### v1.0



# Capabilities

## Components

| Version | Contracts | Libraries | Interfaces | Abstract |
|---|---|---|---|---|
| 1.1 | 1 | 1 | 1 | 2 |

## Exposed Functions

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

| Version | Public | Payable |
|---|---|---|
| 1.1 | 41 | 0 |

| Version | External | Internal | Private | Pure | View |
|---|---|---|---|---|---|
| 1.1 | 20 | 41 | 1 | 6 | 21 |

## State Variables

| Version | Total | Public |
|---|---|---|
| 1.1 | 30 | 27 |

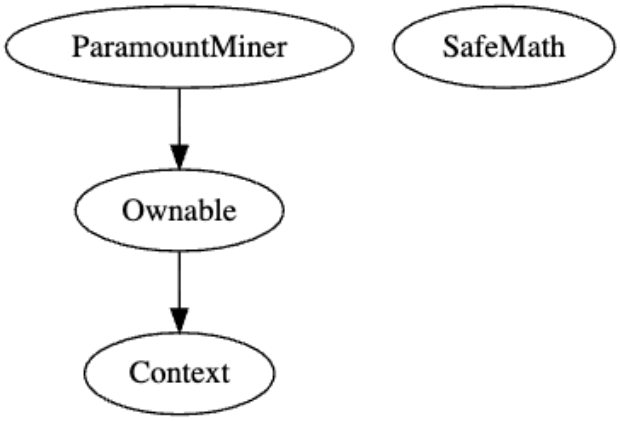
## Capabilities

| Version | Solidity Versions observed | Experimental Features | Can Receive Funds | Uses Assembly | Has Destroyable Contracts |
|---|---|---|---|---|---|
| 1.1 | `^0.8.10` | | | `yes` (1 asm blocks) | |

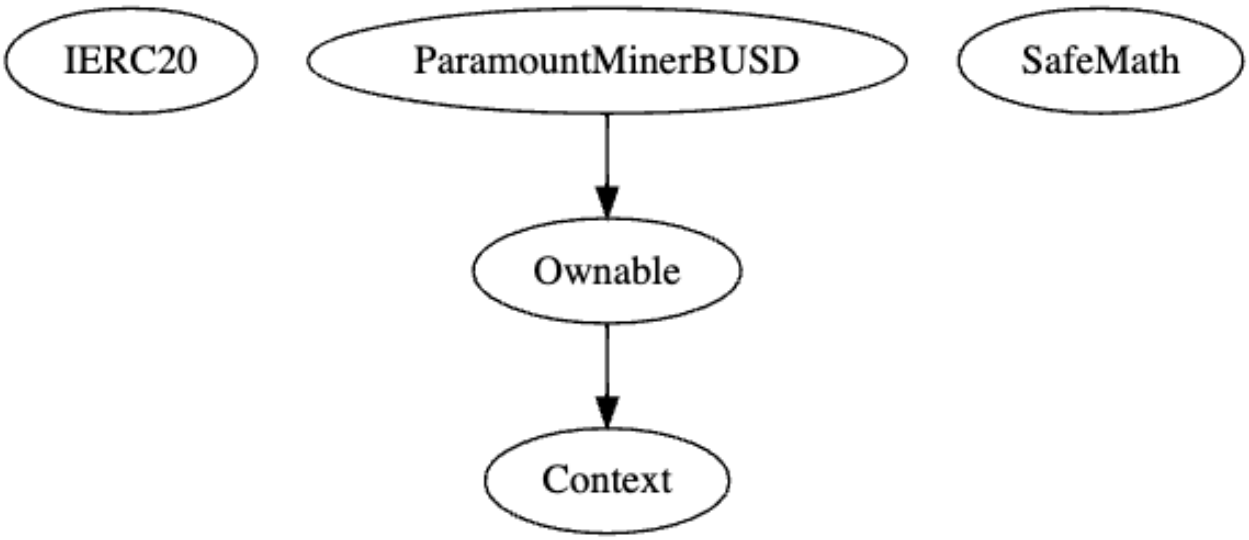
| Version | Transfers ETH | Low-Level Calls | DelegateCall | Uses Hash Functions | EC Recover | New/ Create/ Create2 |
|---|---|---|---|---|---|---|
| 1.1 | yes | | | | | |

# Inheritance Graph

## v1.0

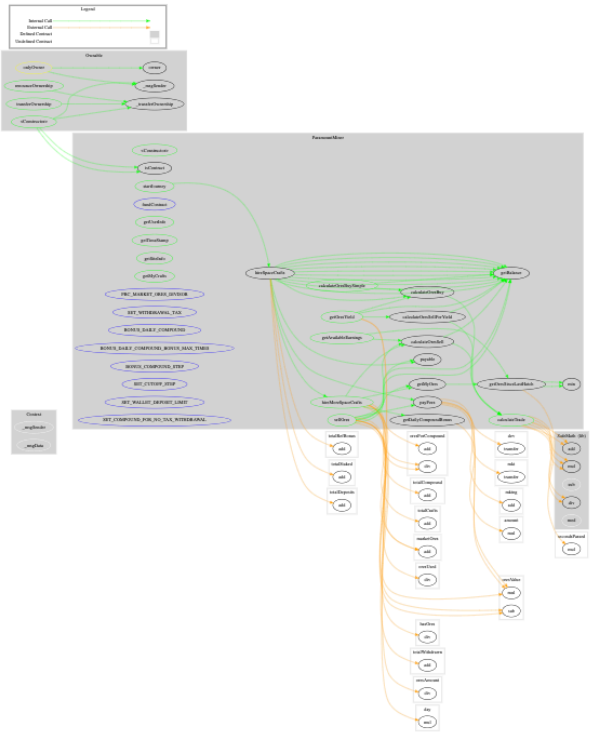



## v1.1

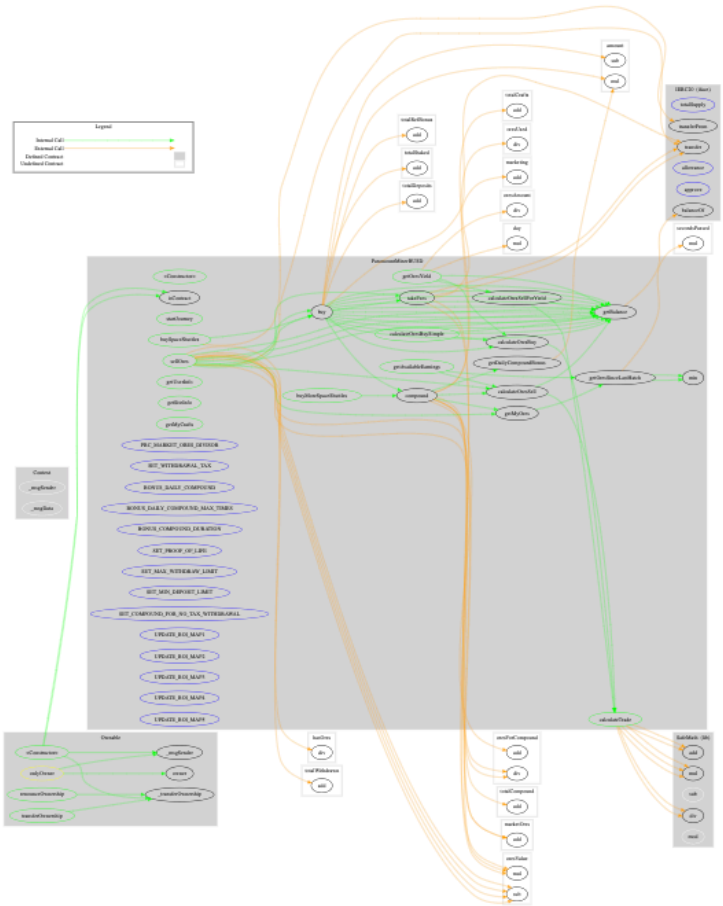


# CallGraph

## v1.0



## v1.1



## Scope of Work/Verify Claims

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Overall checkup (Smart Contract Security)

## Write functions of contract

### v1.0

1. BONUS_COMPOUND_STEP
2. BONUS_DAILY_COMPOUND
3. BONUS_DAILY_COMPOUND_BONUS_MAX_TIMES
4. PRC_MARKET_ORES_DIVISOR
5. SET_COMPOUND_FOR_NO_TAX_WITHDRAWAL
6. SET_CUTOFF_STEP
7. SET_WALLET_DEPOSIT_LIMIT
8. SET_WITHDRAWAL_TAX
9. fundContract
10. hireMoreSpaceCrafts
11. hireSpaceCrafts
12. renounceOwnership
13. sellOres
14. startJourney
15. transferOwnership

### v1.1

startJourney
buyMoreSpaceShuttles
sellOres
buySpaceShuttles
PRC_MARKET_ORES_DIVISOR
SET_WITHDRAWAL_TAX
BONUS_DAILY_COMPOUND
BONUS_DAILY_COMPOUND_MAX_TIMES
BONUS_COMPOUND_DURATION
SET_PROOF_OF_LIFE
SET_MAX_WITHDRAW_LIMIT
SET_MIN_DEPOSIT_LIMIT
SET_COMPOUND_FOR_NO_TAX_WITHDRAWAL
UPDATE_ROI_MAP1
UPDATE_ROI_MAP2
UPDATE_ROI_MAP3
UPDATE_ROI_MAP4
UPDATE_ROI_MAP5

## Overall checkup (Smart Contract Security)

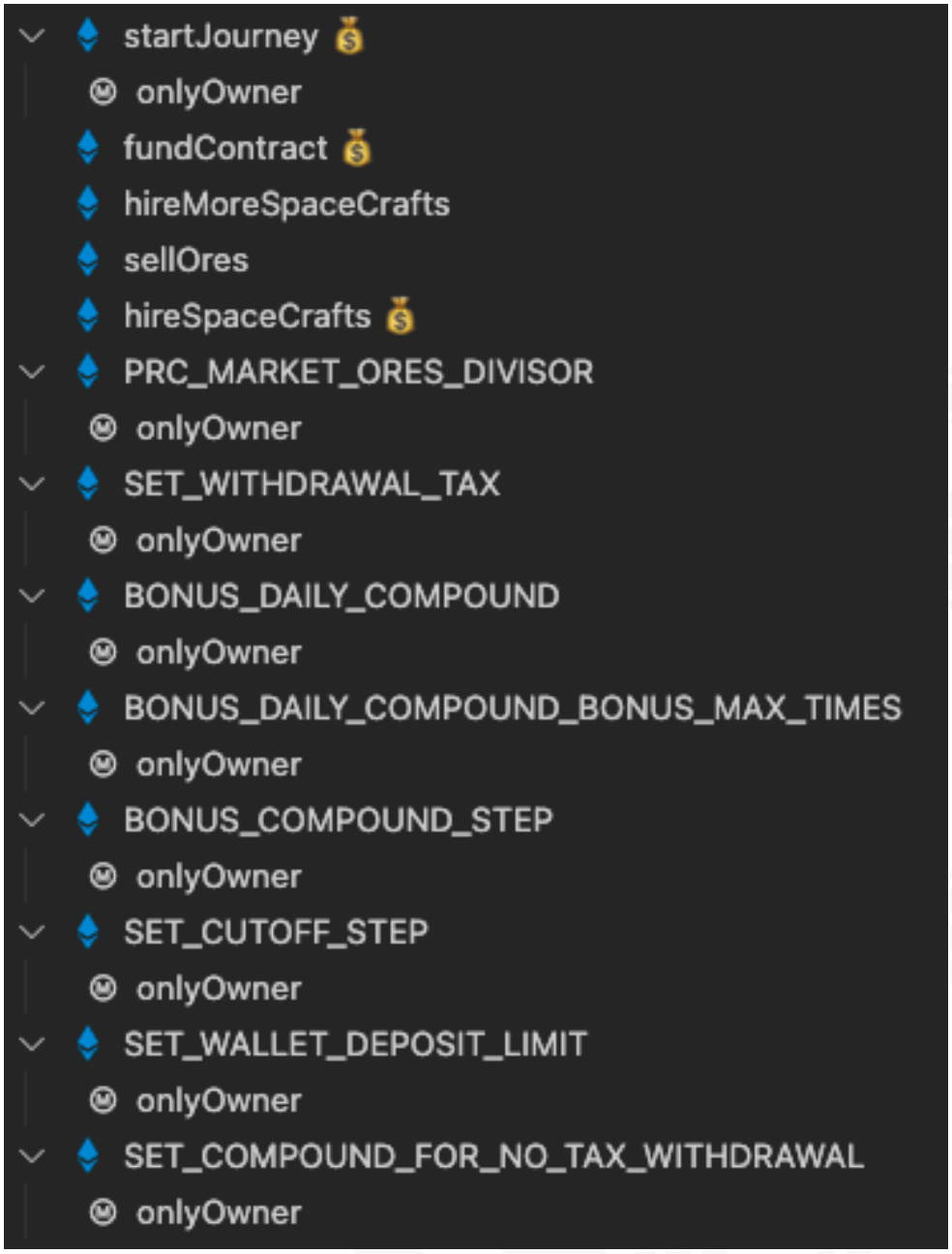
| Tested | Verified |
| --- | --- |
| ✓ | ✓ |

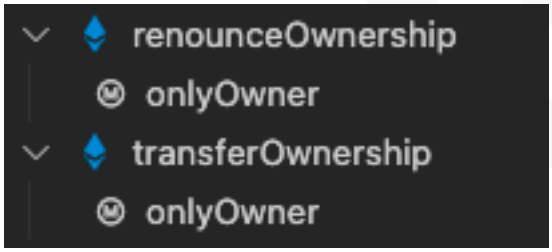
### Legend

| Attribute | Symbol |
| --- | --- |
| Verfified / Checked | ✓ |
| Partly Verified | ⚑ |
| Unverified / Not checked | ✗ |
| Not available | - |

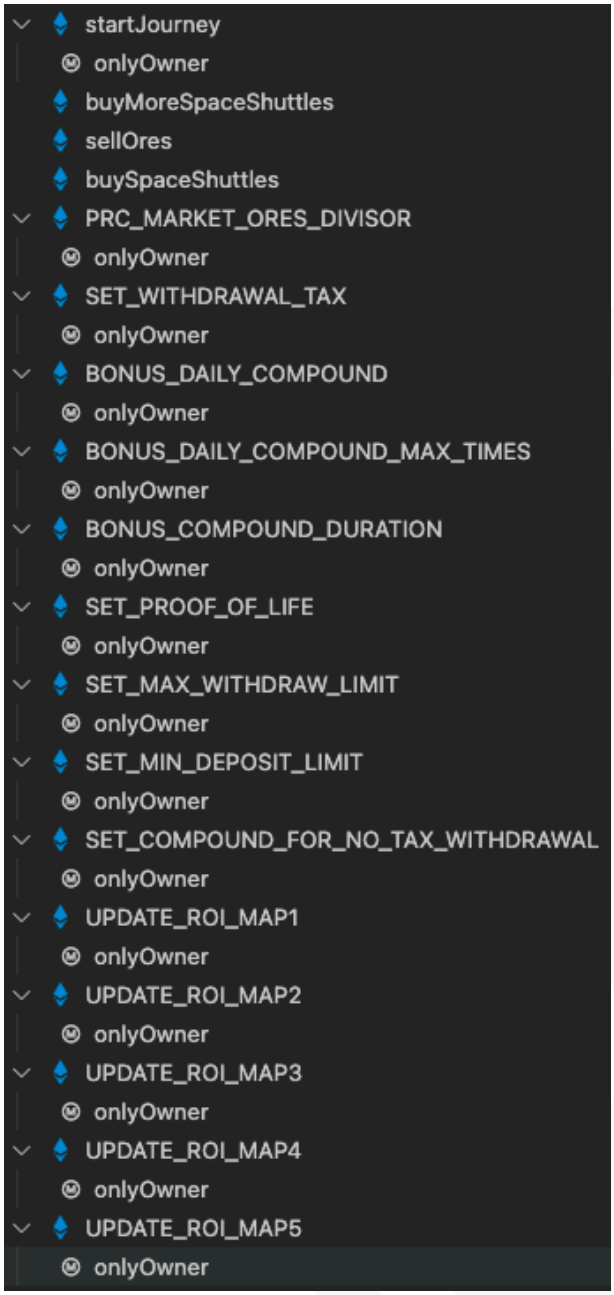
## Modifiers and public functions

### v1.0



## v1.1




# Comments

## v1.0

- *Existing Modifiers*
  - onlyOwner

- There are several authorities which are authorized to call some functions, that means, if the owner is renounced, another address is still authorized to call functions
  - Be aware of this
- Owner can
  - Set withdrawal tax to 80%

- Prevent that MARKET_ORES_DIVISOR can set to 0 because it will revert
    - “hireMoreSpaceCrafts” function
    - “sellOres” function

## v1.1

- Owner can set
    - ROI_MAP
        - We recommend to Implement a limitation downwards for ROI_MAP2-ROI_MAP5 because of the behaviour of the contract.
- MARKET_ORES_DIVISOR should not be set to 0
- Functions were refactored and called by single functions
    - buyMoreSpaceShuttles
    - etc
- All payable functions were removed

**Please check if an OnlyOwner or similar restrictive modifier has been forgotten.**

# Source Units in Scope

## v1.0

| Type | File | Logic Contracts | Interfaces | Lines | nLines | nSLOC | Comment Lines | Complex. Score | Capabilities |
|---|---|---|---|---|---|---|---|---|---|
| | contracts/paramountminer.sol | 4 | ———— | 433 | 431 | 338 | 17 | 285 | |
| | **Totals** | **4** | ———— | **433** | **431** | **338** | **17** | **285** | |

## v1.1

| Type | File | Logic Contracts | Interfaces | Lines | nLines | nSLOC | Comment Lines | Complex. Score | Capabilities |
|---|---|---|---|---|---|---|---|---|---|
| | contracts/BUSD miner.sol | 4 | 1 | 486 | 455 | 367 | 11 | 317 | |
| | **Totals** | **4** | **1** | **486** | **455** | **367** | **11** | **317** | |

### Legend

| Attribute | Description |
|---|---|
| Lines | total lines of the source unit |
| nLines | normalized lines of the source unit (e.g. normalizes functions spanning multiple lines) |
| nSLOC | normalized source lines of code (only source-code lines; no comments, no blank lines) |
| Comment Lines | lines containing single or block comments |
| Complexity Score | a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...) |

# Audit Results

## AUDIT PASSED

## Critical issues

**No critical issues**

## High issues

**No high issues**

## Medium issues

**No medium issues**

## Low issues

| Issue | File | Type | Line | Description |
|---|---|---|---|---|
| #1 | Main | Contract doesn't import npm packages from source (like OpenZeppelin etc.) | - | We recommend to import all packages from npm directly without flatten the contract. Functions could be modified or can be susceptible to vulnerabilities |
| #2 | Main | A floating pragma is set | 2 | The current pragma Solidity directive is „"^0.8.10"". |
| #3 | Main | State variable visibility is not set | 112, 113 | It is best practice to set the visibility of state variables explicitly |
| #4 | Main | Missing Events Arithmetic | 396, 411 | Emit an event for critical parameter changes |

## Informational issues

| Issue | File | Type | Line | Description |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| #1 | Main | State variables that could be declared constant (constable-states) | 89, 90, 88, 112, 113, 87 | Add the `constant` attributes to state variables that never change |
| #2 | Main | Error message is missing | 385, 390, 395, 400, 405, 410, 415, 420, 425, 430, 435, 440, 445, 450, 483 | Provide an error message for require statement |
| #3 | Main | Unnecessary calculation | 349 | getBalance().add(amount).sub(amount is equal to getBalance()<br><br>e.g.<br>Amount = 10<br>Balance 100<br><br>Balance + Amount - Amount <=> 100 + 10 - 10 <=> 100 |
| #4 | Main | Rename variables | 324, 112, 113 | We recommend you to name the variables in a more understandable way, instead of some letters. |

## Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information https://docs.soliditylang.org/en/v0.5.10/natspec-format.html) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variables, functions etc. do.

### 21. May 2022:

- Read whole report for more information

## SWC Attacks

| ID | Title | Relationships | Status |
|---|---|---|---|
| SWC-136 | Unencrypted Private Data On-Chain | CWE-767: Access to Critical Private Variable via Public Method | PASSED |
| SWC-135 | Code With No Effects | CWE-1164: Irrelevant Code | PASSED |
| SWC-134 | Message call with hardcoded gas amount | CWE-655: Improper Initialization | PASSED |
| SWC-133 | Hash Collisions With Multiple Variable Length Arguments | CWE-294: Authentication Bypass by Capture-replay | PASSED |
| SWC-132 | Unexpected Ether balance | CWE-667: Improper Locking | PASSED |
| SWC-131 | Presence of unused variables | CWE-1164: Irrelevant Code | PASSED |
| SWC-130 | Right-To-Left-Override control character (U+202E) | CWE-451: User Interface (UI) Misrepresentation of Critical Information | PASSED |
| SWC-129 | Typographical Error | CWE-480: Use of Incorrect Operator | PASSED |
| SWC-128 | DoS With Block Gas Limit | CWE-400: Uncontrolled Resource Consumption | PASSED |

| | | | |
|---|---|---|---|
| SWC-127 | Arbitrary Jump with Function Type Variable | CWE-695: Use of Low-Level Functionality | PASSED |
| SWC-125 | Incorrect Inheritance Order | CWE-696: Incorrect Behavior Order | PASSED |
| SWC-124 | Write to Arbitrary Storage Location | CWE-123: Write-what-where Condition | PASSED |
| SWC-123 | Requirement Violation | CWE-573: Improper Following of Specification by Caller | PASSED |
| SWC-122 | Lack of Proper Signature Verification | CWE-345: Insufficient Verification of Data Authenticity | PASSED |
| SWC-121 | Missing Protection against Signature Replay Attacks | CWE-347: Improper Verification of Cryptographic Signature | PASSED |
| SWC-120 | Weak Sources of Randomness from Chain Attributes | CWE-330: Use of Insufficiently Random Values | PASSED |
| SWC-119 | Shadowing State Variables | CWE-710: Improper Adherence to Coding Standards | PASSED |
| SWC-118 | Incorrect Constructor Name | CWE-665: Improper Initialization | PASSED |
| SWC-117 | Signature Malleability | CWE-347: Improper Verification of Cryptographic Signature | PASSED |

| | | | |
|---|---|---|---|
| SWC-116 | Timestamp Dependence | CWE-829: Inclusion of Functionality from Untrusted Control Sphere | PASSED |
| SWC-115 | Authorization through tx.origin | CWE-477: Use of Obsolete Function | PASSED |
| SWC-114 | Transaction Order Dependence | CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition') | PASSED |
| SWC-113 | DoS with Failed Call | CWE-703: Improper Check or Handling of Exceptional Conditions | PASSED |
| SWC-112 | Delegatecall to Untrusted Callee | CWE-829: Inclusion of Functionality from Untrusted Control Sphere | PASSED |
| SWC-111 | Use of Deprecated Solidity Functions | CWE-477: Use of Obsolete Function | PASSED |
| SWC-110 | Assert Violation | CWE-670: Always-Incorrect Control Flow Implementation | PASSED |
| SWC-109 | Uninitialized Storage Pointer | CWE-824: Access of Uninitialized Pointer | PASSED |
| SWC-108 | State Variable Default Visibility | CWE-710: Improper Adherence to Coding Standards | NOT PASSED |
| SWC-107 | Reentrancy | CWE-841: Improper Enforcement of Behavioral Workflow | PASSED |
| SWC-106 | Unprotected SELFDESTRUCT Instruction | CWE-284: Improper Access Control | PASSED |

| | | | |
|---|---|---|---|
| SWC-105 | Unprotected Ether Withdrawal | CWE-284: Improper Access Control | PASSED |
| SWC-104 | Unchecked Call Return Value | CWE-252: Unchecked Return Value | PASSED |
| SWC-103 | Floating Pragma | CWE-664: Improper Control of a Resource Through its Lifetime | NOT PASSED |
| SWC-102 | Outdated Compiler Version | CWE-937: Using Components with Known Vulnerabilities | PASSED |
| SWC-101 | Integer Overflow and Underflow | CWE-682: Incorrect Calculation | PASSED |
| SWC-100 | Function Default Visibility | CWE-710: Improper Adherence to Coding Standards | PASSED |


**Blockchain Security | Smart Contract Audits | KYC**

MADE IN GERMANY