



Smart contracts security assessment

Final report

[Tariff: Standard](#)

Sh!tcoin Miner - Polygon Land

September 2023



0xguard.com



hello@0xguard.com

Contents

1. Introduction	3
2. Contracts checked	3
3. Procedure	4
4. Known vulnerabilities checked	4
5. Classification of issue severity	5
6. Issues	6
7. Conclusion	8
8. Disclaimer	9

Introduction

The report has been prepared for **Sh!tcoin Miner - Polygon Land**.

The audited contract is a Ponzi staking with rewards in the same coin with a progressive ROI and capped total yield without explicitly defined source of such rewards.

5% of deposited coins goes to the owner's addresses, 2% (default value) - to the referrer.

The SHA-1 hashes of audited files are:

scm-final.sol 2d895ca9c057a7565f63f9d7a7e4a04fdd5afa89.

Update. The updated contract has SHA-1 of 29662c1d78649a718cf6b0c9b49d53bc9dad8fb6 (scm_v3_2.sol) and was deployed to the [0xF2E46FB5Eb359A4e0029642fFF4C5040b440E027](https://polygonscan.com/address/0xF2E46FB5Eb359A4e0029642fFF4C5040b440E027) address in the Polygon Mainnet with an ERC1967 upgradeable proxy at [0xd4E40A4b1F9F46144209FF0496209858D30fc820](https://polygonscan.com/address/0xd4E40A4b1F9F46144209FF0496209858D30fc820). The contract code has not been verified. We couldn't ensure its matching with the audited version.

Name	Sh!tcoin Miner - Polygon Land
Audit date	2023-09-18 - 2023-09-21
Language	Solidity
Platform	Polygon Network

Contracts checked

Name	Address
SCM	0xF2E46FB5Eb359A4e0029642fFF4C5040b440E027

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
<u>Unencrypted Private Data On-Chain</u>	passed
<u>Code With No Effects</u>	passed
<u>Message call with hardcoded gas amount</u>	passed
<u>Typographical Error</u>	passed
<u>DoS With Block Gas Limit</u>	passed
<u>Presence of unused variables</u>	passed
<u>Incorrect Inheritance Order</u>	passed
<u>Requirement Violation</u>	passed
<u>Weak Sources of Randomness from Chain Attributes</u>	passed
<u>Shadowing State Variables</u>	passed

<u>Incorrect Constructor Name</u>	passed
<u>Block values as a proxy for time</u>	passed
<u>Authorization through tx.origin</u>	passed
<u>DoS with Failed Call</u>	passed
<u>Delegatecall to Untrusted Callee</u>	passed
<u>Use of Deprecated Solidity Functions</u>	passed
<u>Assert Violation</u>	passed
<u>State Variable Default Visibility</u>	passed
<u>Reentrancy</u>	passed
<u>Unprotected SELFDESTRUCT Instruction</u>	passed
<u>Unprotected Ether Withdrawal</u>	passed
<u>Unchecked Call Return Value</u>	passed
<u>Floating Pragma</u>	passed
<u>Outdated Compiler Version</u>	passed
<u>Integer Overflow and Underflow</u>	passed
<u>Function Default Visibility</u>	passed

Classification of issue severity

High severity

High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.

Medium severity

Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.

Low severity

Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

Issues

High severity issues

1. Source of rewards (SCM)

Status: Open

The staking offers a positive ROI percent but there's no source of rewards to harden this reward rate. If there will be not enough rewards, the withdraw and compound functions become unusable as token transfer will be reduced to 0.

```
function handleWithdraw(uint256 rewards) private {
  if(getBalance() < rewardsValue) {
    rewardsValue = getBalance();
  }
  uint256 payout = rewardsValue.sub(fees);
  payable(_msgSender()).transfer(payout);
  ...
}
```

Medium severity issues

1. Doubtful math (SCM)

Status: Fixed

The `calculateSell` calculations are different from the `getYield` estimation. Actual rewards amounts are times several greater than estimated. Without documentation it's impossible to check where the error is, but we suppose it's in the `calculateTrade` function.

Also, the `getUserDailyYieldControl` function performs many unnecessary calculations.

Recommendation: Check the math.

Low severity issues

1. Typos (SCM)

Status: Fixed

Typo in 'REFELCTION', 'Mininum'.

2. Inconsistent comment (SCM)

Status: Fixed

There's a comment about constructor in initializable contracts:

```
///  
@dev no constructor in upgradable contracts. Instead we have initializers
```

In fact, OpenZeppelin's Initializable contract suggest implementing a constructor section with the `_disableInitializers` function.

Recommendation: Follow the OpenZeppelin's guide.

3. Initialization of the constants (SCM)

Status: Fixed

No need to initialize variables that are constant it its nature:

```
uint256 private _NOT_ENTERED;  
uint256 private _ENTERED;  
  
function initialize(address _dev1, address _dev2, address _dev3) public initializer {  
    _NOT_ENTERED = 1;  
    _ENTERED = 2;  
}
```

Recommendation: Use the ReentrancyGuard contract.

Conclusion

Sh!tcoin Miner - Polygon Land SCM contract was audited. 1 high, 1 medium, 3 low severity issues were found.

1 medium, 3 low severity issues have been fixed in the update.

The audited contract is designed to be deployed with a proxy, and therefore can be upgraded by the owner.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without OxGuard prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts OxGuard to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

OxGuard retains exclusive publishing rights for the results of this audit on its website and social networks.

