HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Strip Coin
Date: September 3rd, 2021

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

Document

Name	Smart Contract Code Review and Security Analysis Report for Strip Coin.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 token; Vesting; Pre-sale
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Zip archive	strip-contracts-23-8-2021.zip
Technical Documentation	NO
JS tests	NO
Timeline	18 AUGUST 2021 - 03 SEPTEMBER 2021
Changelog	20 AUGUST 2021 - INITIAL AUDIT 03 SEPTEMBER 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 Strip Coin (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contract and its code review conducted between August 18th, 2021 - August 18th, 2021. The second code review conducted on September 3rd, 2021.

Scope

The scope of the project is smart contracts in the repository:

Zip archive:
strip-contracts-23-8-2021.zip
Md5 hash:
83602488872d1ba578383c7680c9391f
Technical Documentation: No
JS tests: No
Contracts:

interfaces/IPresale.sol	(md5: f99ef34b122c3ee1d5ed2bbed588a43d)
interfaces/IStripToken.sol	(md5: 90540632c60d011c6187fe43c6d10318)
interfaces/IVesting.sol	(md5: fa4fce1b654bc2d076eac24b1080bfdf)
Presale.sol	(md5: 46830233ed6d90db38773acc4da8dbf7)
StripToken.sol	(md5: cbff4bad351260169713e3a81f4b52f5)
Vesting.sol	(md5: ff904b94a5545bdb3675129da344d55d)

We have scanned this smart contract 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 ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts are secured.

Insecure	Poor secured	Secured	Well-secured

You are here 

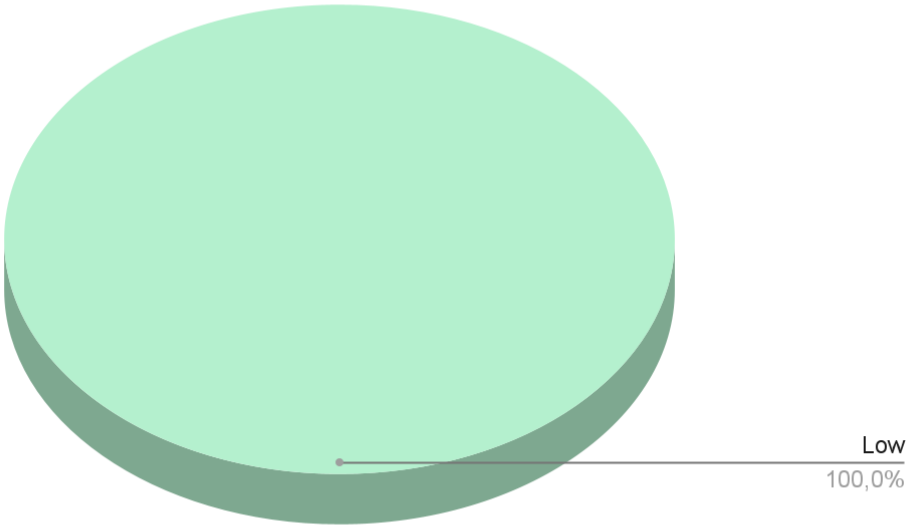
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview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

As a result of the audit, security engineers found **1** high, **1** medium and **14** low severity issues.

After the second review security engineers found that most of the issues were fixed and there are **4** low severity issues left in the code.

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

Graph 1. The distribution of vulnerabilities after the audit.



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

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

Business logic overcome

It is possible to add new recipient by owner without updating unallocatedAmounts and other checkings via **updateRecipient** function.

Recommendation: please either add check that recipient already exists to the **updateRecipient** function or do the same logic as on the **addRecipient** one.

Fixed before the second review.

Medium

Multiple call to the same function

The function getVested is called twice inside the **withdrawToken** function. first call is on the line **156**, and the second one is from the **getWithdrawable** on the line **149.**

Recommendation: please consider saving a gas and storing calculated values as local variable.

Fixed before the second review.

Low

1. Business logic overcome

   It is possible to add new recipient by owner without updating unallocatedAmounts and other checkings via **updateRecipient** function.

   Recommendation: please either add check that recipient already exists to the **updateRecipient** function or do the same logic as on the **addRecipient** one.

   Fixed before the second review.

2. Repeated code

   addRecipients function does the same functionality in the loop which the addRecipient does.

   Recommendation: please consider creating a private function for doing the logic and calling it from both addRecipient and addRecipients.

   Fixed before the second review.

3. Unused import

   Vesting.sol imports Ownable.sol contract which is not used

   Fixed before the second review.

4. Unused import

   Vesting.sol imports IERC20.sol contract which is not used

   Fixed before the second review.

5. Unused import

   Presale.sol imports IERC20.sol contract which is not used

Lines: Presale.sol#5

```
import "@openzeppelin/contracts/token/ERC20/IERC20.sol";
```

6. Unused import

   Presale.sol imports hardhat/console.sol contract which is not used

   Fixed before the second review.

7. Too many digits

   Literals with many digits are difficult to read and review.

   Recommendation: please consider using science notation and ether units, eg: **500e9 ether**

   Fixed before the second review.

8. Unused local variable

   Local variable **data** is not used

   Recommendation: please remove unused local variable.

Lines: Presale.sol#132

```
(bool sent, bytes memory data) = multiSigAdmin.call{value:
weiBalance}("");
```

9. Unused local variable

   Local variable **data** is not used

   Recommendation: please remove unused local variable.

Lines: Presale.sol#175

```
(bool sent, bytes memory data) = multiSigAdmin.call{value:
weiAmount}("");
```

10. Boolean equality

    Boolean constants can be used directly and do not need to be compare to true or false.

    Fixed before the second review.

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

11. Unused state variable

    Vesting.TOTAL_AMOUNT (Vesting.sol#26) is never used

    Fixed before the second review.

12. Missing events access control

    Changing contract's multiSigAdmin doesn't emit an event, which could be hard to track off-chain

Lines: StripToken.sol#69-72

```
function setMultiSigAdminAddress(address _multiSigAdmin) public onlyOwner
{
   require (_multiSigAdmin != address(0x00), "Invalid MultiSig admin
address");
   multiSigAdmin = _multiSigAdmin;
}
```

13. A public function that could be declared external

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

    Fixed before the 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.

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

As a result of the audit, security engineers found **1** high, **1** medium and **14** low severity issues.

After the second review security engineers found that most of the issues were fixed and there are **4** low severity issues left in the code.

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 the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free 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 the security of smart contracts.

Technical Disclaimer

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