SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: PhantomDAO
Date: December 30th, 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 PhantomDAO.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	ERC20 tokens
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/Phantom-DAO/phantom-network/
Commit	278e40114243fafe8d08f3b4cc20c1eb49803c39
Technical Documentation	YES
JS tests	YES
Website	phantomdao.xyz
Timeline	23 DECEMBER 2021 - 30 DECEMBER 2021
Changelog	30 DECEMBER 2021 - INITIAL AUDIT

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by PhantomDAO (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 December 23rd, 2021 - December 30th, 2021.

Scope

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

Repository:
https://github.com/Phantom-DAO/phantom-network/

Commit:
278e40114243fafe8d08f3b4cc20c1eb49803c39

Technical Documentation: Yes, https://medium.com/phantom-dao/not-just-another-fork-cb0efaefea1e

JS tests: Yes, in the repository

Contracts:

```
contracts/core/erc20/PHM.sol
contracts/core/erc20/SolmateERC20.sol
contracts/core/erc20/sPHM.sol
contracts/core/erc20/gPHM.sol
contracts/core/PhantomVault.sol
contracts/core/PhantomTreasury.sol
contracts/core/PhantomAdmin.sol
contracts/mixins/PhantomStorageMixin.sol
contracts/storage/PhantomStorageKeys.sol
contracts/storage/PhantomStorage.sol
contracts/level0/bonding/PhantomBonding.sol
contracts/level0/staking/PhantomStaking.sol
contracts/level0/staking/PhantomStakingWarmup.sol
contracts/level0/governance/PhantomExecutor.sol
contracts/level0/finance/routers/PhantomSpiritRouter.sol
contracts/level0/finance/routers/PhantomYearnRouter.sol
contracts/level0/finance/PhantomAllocator.sol
contracts/level0/finance/PhantomPayments.sol
contracts/level1/governance/PhantomGovernor.sol
contracts/level1/finance/PhantomFinance.sol
interfaces/core/IPhantomVault.sol
interfaces/core/erc20/IPhantomERC20.sol
interfaces/core/erc20/IsPHM.sol
interfaces/core/erc20/IgPHM.sol
interfaces/core/erc20/IPHM.sol
interfaces/core/IPhantomTreasury.sol
interfaces/mixins/IPhantomStorageMixin.sol
interfaces/storage/IPhantomStorage.sol
interfaces/storage/IPhantomStorageFactory.sol
interfaces/level0/bonding/IPhantomBonding.sol
interfaces/level0/staking/IPhantomStaking.sol
interfaces/level0/staking/IPhantomStakingWarmup.sol
interfaces/level0/governance/IPhantomGovernor.sol
interfaces/level0/governance/IPhantomExecutor.sol
interfaces/level0/finance/routers/IPhantomSpiritRouter.sol
interfaces/level0/finance/IPhantomAllocator.sol
interfaces/external/IyVault.sol
interfaces/external/ISpiritSwapGauge.sol
interfaces/level1/governance/IPhantomGovernance.sol
interfaces/level1/finance/IPhantomFinance.sol
```

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
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.

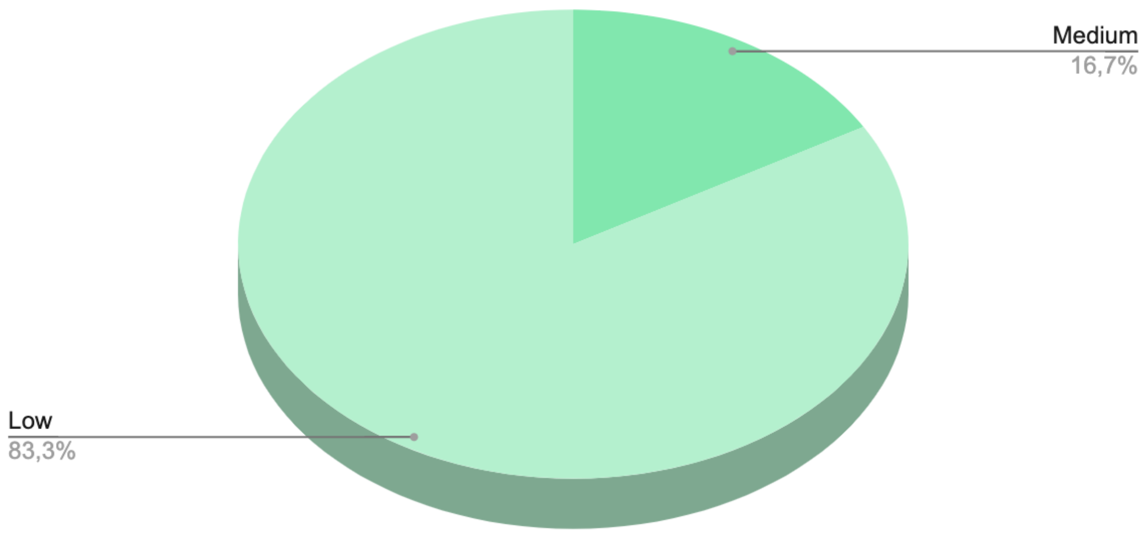
Insecure **Poor secured** **Secured** **Well-secured**

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** medium and **5** low severity issues.

Graph 1. The distribution of vulnerabilities after the audit.



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

Audit overview

■ ■ ■ ■ Critical

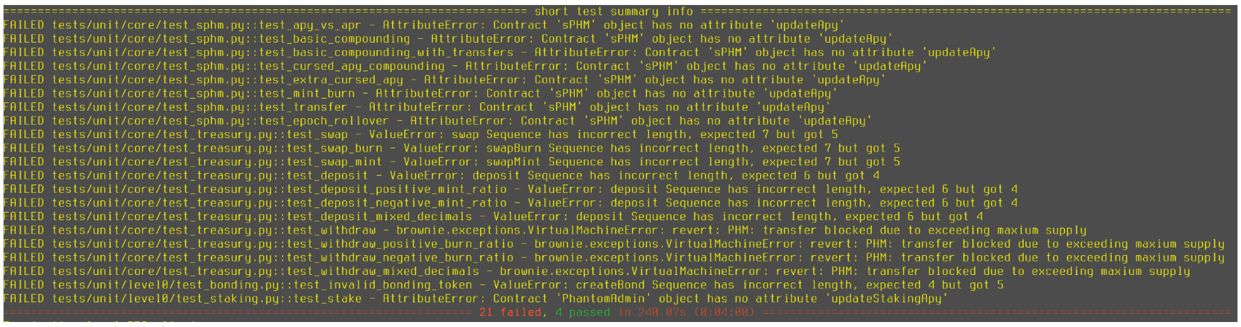
No critical issues were found.

■ ■ ■ High

No high severity issues were found.

■ ■ Medium

Some tests are failing



```
=========================================================================== short test summary info ===========================================================================
FAILED tests/unit/core/test_sphm.py::test_apy_vs_apr - AttributeError: Contract 'sPHM' object has no attribute 'updateApy'
FAILED tests/unit/core/test_sphm.py::test_basic_compounding - AttributeError: Contract 'sPHM' object has no attribute 'updateApy'
FAILED tests/unit/core/test_sphm.py::test_basic_compounding_with_transfers - AttributeError: Contract 'sPHM' object has no attribute 'updateApy'
FAILED tests/unit/core/test_sphm.py::test_cursed_apy_compounding - AttributeError: Contract 'sPHM' object has no attribute 'updateApy'
FAILED tests/unit/core/test_sphm.py::test_extra_cursed_apy - AttributeError: Contract 'sPHM' object has no attribute 'updateApy'
FAILED tests/unit/core/test_sphm.py::test_mint_burn - AttributeError: Contract 'sPHM' object has no attribute 'updateApy'
FAILED tests/unit/core/test_sphm.py::test_transfer - AttributeError: Contract 'sPHM' object has no attribute 'updateApy'
FAILED tests/unit/core/test_sphm.py::test_epoch_rollover - AttributeError: Contract 'sPHM' object has no attribute 'updateApy'
FAILED tests/unit/core/test_treasury.py::test_swap - ValueError: swap Sequence has incorrect length, expected 7 but got 5
FAILED tests/unit/core/test_treasury.py::test_swap_burn - ValueError: swapBurn Sequence has incorrect length, expected 7 but got 5
FAILED tests/unit/core/test_treasury.py::test_swap_mint - ValueError: swapMint Sequence has incorrect length, expected 7 but got 5
FAILED tests/unit/core/test_treasury.py::test_deposit - ValueError: deposit Sequence has incorrect length, expected 6 but got 4
FAILED tests/unit/core/test_treasury.py::test_deposit_positive_mint_ratio - ValueError: deposit Sequence has incorrect length, expected 6 but got 4
FAILED tests/unit/core/test_treasury.py::test_deposit_negative_mint_ratio - ValueError: deposit Sequence has incorrect length, expected 6 but got 4
FAILED tests/unit/core/test_treasury.py::test_deposit_mixed_decimals - ValueError: deposit Sequence has incorrect length, expected 6 but got 4
FAILED tests/unit/core/test_treasury.py::test_withdraw - brownie.exceptions.VirtualMachineError: revert: PHM: transfer blocked due to exceeding maxium supply
FAILED tests/unit/core/test_treasury.py::test_withdraw_positive_burn_ratio - brownie.exceptions.VirtualMachineError: revert: PHM: transfer blocked due to exceeding maxium supply
FAILED tests/unit/core/test_treasury.py::test_withdraw_negative_burn_ratio - brownie.exceptions.VirtualMachineError: revert: PHM: transfer blocked due to exceeding maxium supply
FAILED tests/unit/core/test_treasury.py::test_withdraw_mixed_decimals - brownie.exceptions.VirtualMachineError: revert: PHM: transfer blocked due to exceeding maxium supply
FAILED tests/unit/level0/test_bonding.py::test_invalid_bonding_token - ValueError: createBond Sequence has incorrect length, expected 4 but got 5
FAILED tests/unit/level0/test_staking.py::test_stake - AttributeError: Contract 'PhantomAdmin' object has no attribute 'updateStakingApy'
=============================================================== 21 failed, 4 passed in 240.07s (0:04:00) ===============================================================
```

Recommendation: Please update tests and make sure that you have recommended coverage(minimum 95% for branches, 100% for the main logic contracts)

■ Low

1. The function could become inoperable

   Contracts: PhantomBonding.sol

   Functions: redeemBonds

   Function iterates over nonces and the number of repetitions is not limited. If the number is bigger than some value, the function will be inoperable.

   Recommendation: add(optional) *limit* parameter that limits nonces included in the loop during a single call or implement in-function gas management.

2. Missing event for changing registered contracts, debug mode, staking compounding periods per year, token bonding list, bonding type, bonding multiplier

   Contracts: PhantomStorage.sol, PhantomAdmin.sol

   Functions: registerContract, unregisterContract, enableDebugMode, disableDebugMode, updateStakingCompoundingPeriodsPerYear, addTokenToBondingList, addMultipleTokensToBondingList, removeTokenFromBondingList, removeMultipleTokensToBondingList, addBondType, removeBondType, setBondingMultiplierFor

   Changing critical values should be followed by the event emitting for better tracking off-chain.

   Recommendation: Please emit events on the critical values changing.

3. Boolean equality

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

   Contracts: PhantomStorage.sol

   Modifier: onlyRegisteredContracts

   Recommendation: remove the equality to the boolean constant.

4. A public function that could be declared external.

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

   Contracts: gPHM.sol, SolmateERC20.sol, sPHM.sol, PhantomTreasury.sol

   Functions: approve, transfer, transferFrom, permit, getCurrentVotes, delegate, delegateBySig, getPriorVotes, balanceFromPHM, balanceToPHM, rewardRate, externalValueOf, sumReserves

   Recommendation: Use the **external** attribute for functions never called from the contract.

5. State variables that could be declared constant

   Constant state variables should be declared constant to save gas.

   Contracts: sPHM.sol, PhantomPayments.sol, PhantomStorageMixin.sol

   Variables: _rewardRate, SECONDS_PER_YEAR, phantomStorage

   Recommendation: Add the constant attributes to state variables that never change.

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** medium and **5** low severity issues.

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.