ASSURE DEFI®
THE VERIFICATION **GOLD STANDARD**

Security Assessment:
Optimus Pool Manager

December 22, 2023

- Audit Status: **Pass**
- Audit Edition: **Advance**

VERIFIED BY ASSURE DEFI®
INTEGRITY * TRUST * CREDIBILITY

Project Overview

Token Summary

Parameter	Result
Address	0x
Name	Optimus
Token Tracker	Optimus (OPTIMUS)
Decimals	9
Supply	100,000,000
Platform	Ethereum
compiler	v0.8.21+commit.d9974bed
Contract Name	PoolManager
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://etherscan.io/address/0x#code
Payment Tx	Corporate

Main Contract Assessed
Contract Name

Name	Contract	Live
Optimus	0x	No

TestNet Contract Assessed
Contract Name

Name	Contract	Live
Optimus	0x4298C180d1608dCd2B1b10962d09903F6Cb52119	No

Solidity Code Provided

SolID	File Sha-1	FileName
OPPM	752e6bf48c4750b3f0045e9f3104e6ace12e7f9c	PoolManager.sol
OPPM	1944aa7c692c5753879a1a44999791cf0aeeeef8	IPoolManager.sol
OPPM	d743d6021e0d1600f19950f08da20708a01d285f	StakingPool.sol
OPPM	938244c87506d5ead576a9ebca884495989da992	NativeRewards.sol
OPPM	42757e89638f1f385f21fc91deadcbc8135f464e	IStakingPool.sol
OPPM	85f074df7ec3466a8c6083790f516ad794f04e4b	TokenRewards.sol

Smart Contract Vulnerability Checks

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) while overlaying a wide range of weakness variants that are specific to smart contracts.

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	PoolManager.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	PoolManager.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	PoolManager.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	PoolManager.sol	L: 2 C: 0
SWC-104	Pass	Unchecked Call Return Value.	PoolManager.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	PoolManager.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	PoolManager.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	PoolManager.sol	L: 0 C: 0
SWC-108	Low	State variable visibility is not set..	PoolManager.sol	L: 18 C: 7,L: 25 C: 35
SWC-109	Pass	Uninitialized Storage Pointer.	PoolManager.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	PoolManager.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	PoolManager.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	PoolManager.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	PoolManager.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	PoolManager.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-115	Pass	Authorization through tx.origin.	PoolManager.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	PoolManager.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	PoolManager.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	PoolManager.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	PoolManager.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randonmness.	PoolManager.sol	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	PoolManager.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	PoolManager.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	PoolManager.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	PoolManager.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	PoolManager.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	PoolManager.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	PoolManager.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	PoolManager.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	PoolManager.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U +202E).	PoolManager.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	PoolManager.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	PoolManager.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	PoolManager.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	PoolManager.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	PoolManager.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	PoolManager.sol	L: 0 C: 0

We scan the contract for additional security issues using MYTHX and industry-standard security scanning tools.

Smart Contract Vulnerability Details

SWC-103 - Floating Pragma.

CWE-664: Improper Control of a Resource Through its Lifetime.

References:

Description:

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation:

Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.

References:

Ethereum Smart Contract Best Practices - Lock pragmas to specific compiler version.

Smart Contract Vulnerability Details

SWC-108 - State Variable Default Visibility

CWE-710: Improper Adherence to Coding Standards

Description:

Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

Remediation:

Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.
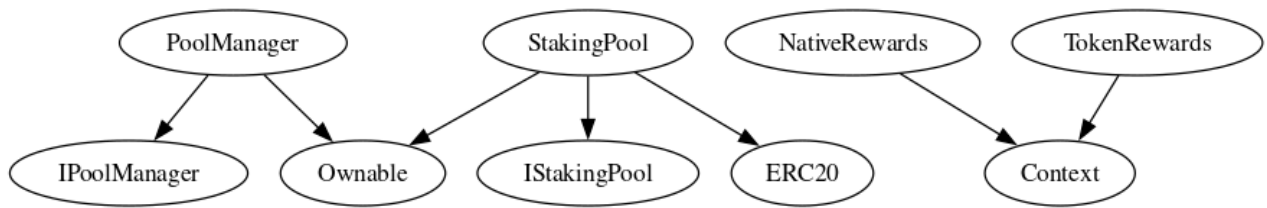
References:

Ethereum Smart Contract Best Practices - Explicitly mark visibility in functions and state variables

Inheritance

The contract for Optimus has the following inheritance structure.

The Project has a Total Supply of 100,000,000

Privileged Functions (onlyOwner)

Please Note if the contract is Renounced none of this functions can be executed.

Function Name	Parameters	Visibility
setTimelockSeconds		external
resetWalletStakedTime		external
removePool		external
createPool		external
setPercentages		external
setPercentages		external

OPTIMUS-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	ℹ Informational	PoolManager.sol: NativeRewards L: 38 C: 11,L: 19 C: 11 PoolManager L: 18 C: 7, L: 25 C: 35	🗎 Detected

Description

The following functions are declared as public and are not invoked in any of the contracts contained within the projects scope:

Function Name	Parameters	Visibility
_rewardsPerShare		internal
setShare	address _wallet,uint256 _balUpdate,bool _removing	public
_totalPercentages		internal
_raffleUserIdx		internal

The functions that are never called internally within the contract should have external visibility

Remediation

We advise that the function's visibility specifiers are set to external, and the array-based arguments change their data location from memory to calldata, optimizing the gas cost of the function.

References:

external vs public best practices.

OPTIMUS-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	⬥ Low	PoolManager.sol: NativeReward L: 38, C: 14, PoolManagers L: 176 C: 14, TokenRewards L: 127 C:14,StakingPools L:88 C:14, L:69 C:14	▤ Detected

Description

The given input is missing the check for the non-zero address.

The given input is missing the check for the missing required function.

Remediation

We advise the client to add the check for the passed-in values to prevent unexpected errors as below:

```
    ...
    require(receiver != address(0), "Receiver is the zero address");
    ...
    ...
    require(value X limitation, "Your not able to do this function");
    ...
```

We also recommend customer to review the following function that is missing a required validation. missing required function.

OPTIMUS-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	Low	PoolManager.sol: PoolManager L: 182, C: 14, L: 176 C: 14,L: 191 C: 14,L: 216 C: 14,L: 224 C: 14, TokenRewards L: 127 C:14, StakingPools L:88 C:14, L:69 C:14	Detected

Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.The linked code does not create an event for the transfer.

Remediation

Emit an event for critical parameter changes. It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Technical Findings Summary
Classification of Risk

Severity	Description
🔴 Critical	Risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
🟠 High	Risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
🟡 Medium	Risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform
✦ Low	Risks can be any of the above but on a smaller scale. They generally do not compromise the overall integrity of the Project, but they may be less efficient than other solutions.
ⓘ Informational	Errors are often recommended to improve the code's style or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

Findings

Severity	Found	Pending	Resolved
🔴 Critical	0	0	0
🟠 High	0	0	0
🟡 Medium	0	0	0
✦ Low	2	2	0
ⓘ Informational	1	1	0
Total	3	3	0

Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/OptimusAI_Token	Pass
Other	Hello@optimustoken.io	Pass
Website	https://www.optimustoken.io/	Pass
Telegram	https://t.me/OptimusAI_ETH	Pass

We recommend to have 3 or more social media sources including a completed working websites.

Social Media Information Notes:

Auditor Notes: undefined

Project Owner Notes:

ASSURE
DEFI

OFFICIAL
PARTNER

Audit Result

Final Audit Score

Review	Score
Security Score	87
Auditor Score	87

The Following Score System Has been Added to this page to help understand the value of the audit, the maximun score is 100, however to attain that value the project most pass and provide all the data needed for the assessment. Our Passing Score has been changed to 85 Points for a higher standard, if a project does not attain 85% is an automatic failure. Read our notes and final assessment below.

Audit Passed

Assessment Results

Important Notes:

• No vulnerabilities or exploits were detected at the time of the audit.ı

• This is the master contract import TokenRewards, NativeReWards and StakingPools. Each Contract has been reviewed during our analysis.ı

• Consider adding reentrancy to all the contracts with external functions for users.ı

• This audit contains data accross all the contracts.

Auditor Score =87
Audit Passed

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that actagainst the nature of decentralization, such as explicit ownership or specialized access roles incombination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimalEVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on howblock.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functionsbeing invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that mayresult in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to makethe codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code,such as a constructor assignment imposing different require statements on the input variables than a setterfunction.

Coding Best Practices

ERC 20 Conding Standards are a set of rules that each developer should follow to ensure the code meet a set of creterias and is readable by all the developers.

Disclaimer

Assure Defi has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocation for the Project, and users relying on this report should not consider this as having any merit for financial advice in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided 'as is, and Assure Defi is under no covenant to audited completeness, accuracy, or solidity of the contracts. In no event will Assure Defi or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report.

The assessment services provided by Assure Defi are subject to dependencies and are under continuing development. You agree that your access or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The services may access, and depend upon, multiple layers of third parties.

ASSURE DEFI™
THE VERIFICATION **GOLD STANDARD**