

Auditor Labs

GAIA WORLD

LP Wrapper

SMART CONTRACT AUDIT

Made in by auditorlabs.com

Table of contents

1. Disclaimer.....	3
2. About the Project and Company	4
2.1 Project Overview.....	5
3. Vulnerability & Risk Level	6
4. Auditing Strategy and Techniques Applied.....	7
4.1 Methodology	7
4.2 Used Code from other Frameworks/Smart Contracts	8
4.3 Tested Contract Files	9
4.4 Metrics / CallGraph.....	10
4.5 Metrics / Source Lines & Risk.....	11
4.6 Metrics / Capabilities	12
4.7 Metrics / Source Unites in Scope	13
5. Scope of Work	13
5.1 Manual and Automated Vulnerability Test.....	14
5.1.1 Selfdestruct owner	14
5.1.2 Floating compiler versions	15
5.2 SWC Attacks	16
6. Executive Summary.....	20
7. Deployed Smart Contract	20

1. Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Omnisoft LTD (GAIA Everworld). If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (10.12.2021)	Layout
0.2 (13.12.2021)	Test Deployment
0.5 (14.12.2021)	Automated Security Testing Manual Security Testing
0.6 (14.12.2021)	Testing SWC Checks
0.7 (14.12.2021)	Verify Claims
0.9 (14.12.2021)	Summary and Recommendation
1.0 (14.12.2021)	Final document
1.1 (14.12.2021)	Added deployed contract addresses

2. About the Project and Company

Omnisoft LTD
OMC Chambers
Wickhams Cay 1
Road Town, Tortola
British Virgin Islands

Website: <https://gaiaworld.com>

Twitter: <https://twitter.com/GaiaEverWorld>

Medium: <https://medium.com/@gaia-world>

Telegram: <https://t.me/GaiaEverworld>

Discord: <https://discord.gg/EGT7c4RVfs>

Email: contact@gaiaworld.com



2.1 Project Overview

Gaia Everworld blends classic fantasy narratives with state of the art blockchain and NFT technology. In the multi-realm gaming environment, players will be able to use their Gaia Legionnaires to wage campaigns, defend lands, and other immersive activities. Like many other games, like Pokemon, or Clash of Clans, Gaia Everworld allows players to own their characters, and interact in a dynamic environment with other human players all over the world.

The gaming environment allows for players to choose a homeland, which will give their NFT-based Gaia special powers, as well as weaknesses. The game uses a play-to-earn model, so that players have a financial incentive to join and play.

In Gaia Everworld, they offer players the ability to exist in a multi-realm online environment and participate in both PVP Battles and Legion Mode. The game centers on Gaia — a mythical creature that can be bred and owned in the form of an NFT. The underlying goal of the game is to have the strongest collection of Gaia. With these NFT creatures, players can battle other players in the game, and conquer the lands of Gaia Everworld.

Of course, Gaia can be bred and added to a collection of other Gaia — or sold to other players. The two tokens that make the platform work are \$GAIA, which can be staked, and \$GGP, which is needed to breed Gaia.

- Holders of \$GAIA can stake coins to earn \$GAIA.
- Players to earn \$GAIA and \$GGP (Gaia Growth Potion) by playing the game and participating in events and adventures.
- Players to trade or sell their Gaia, Gaia eggs, land and resources in the Gaia Everworld marketplace.
- Players to loan their Gaia to other players in a peer to peer contract. The owner then earns a percentage of the \$GGP earned by the loanees game play.

With NFT based games, players are also the owners of the game, and can control the platform to a much higher level than ever before.

3. 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 as possible.
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

4. 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.

4.1 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 Auditor Labs 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 Auditor Labs 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.

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

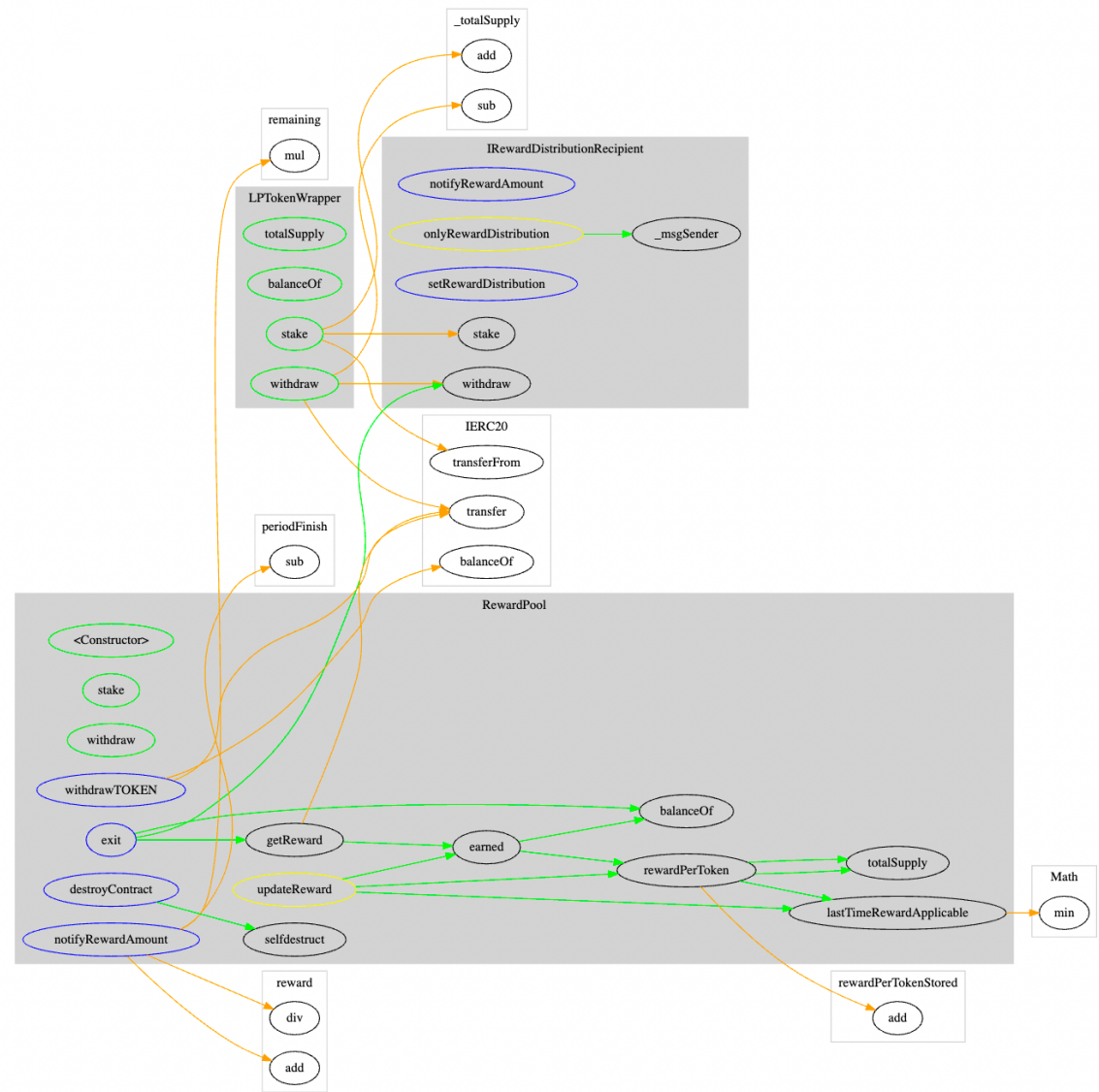
Dependency / Import Path	Source
contracts/interfaces/IUniswapV2Pair.sol	https://github.com/Uniswap/v2-core/blob/master/contracts/interfaces/IUniswapV2Pair.sol
contracts/libraries/SafeMath.sol	https://github.com/Uniswap/v2-core/blob/master/contracts/libraries/SafeMath.sol
contracts/UniswapV2ERC20.sol	https://github.com/Uniswap/v2-core/blob/master/contracts/UniswapV2ERC20.sol
contracts/libraries/Math.sol	https://github.com/Uniswap/v2-core/blob/master/contracts/libraries/Math.sol
contracts/libraries/UQ112x112.sol	https://github.com/Uniswap/v2-core/blob/master/contracts/libraries/UQ112x112.sol
contracts/interfaces/IERC20.sol	https://github.com/Uniswap/v2-core/blob/master/contracts/interfaces/IERC20.sol
contracts/interfaces/IUniswapV2Factory.sol	https://github.com/Uniswap/v2-core/blob/master/contracts/UniswapV2Factory.sol
contracts/interfaces/IUniswapV2Callee.sol	https://github.com/Uniswap/v2-core/blob/master/contracts/interfaces/IUniswapV2Callee.sol
contracts/UniswapV2Pair.sol	https://github.com/Uniswap/v2-core/blob/master/contracts/interfaces/IUniswapV2Pair.sol

4.3 Tested Contract Files

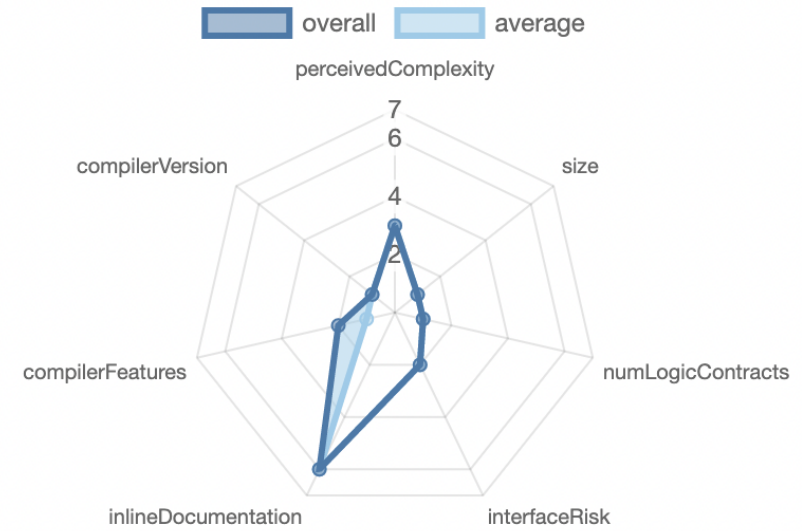
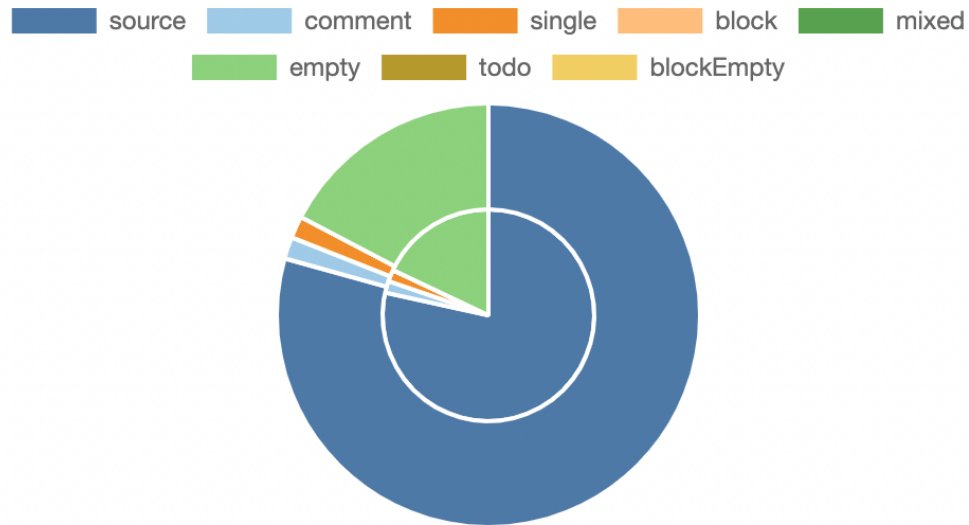
The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5)
LPWrapper.sol	300b8674c4b765a639474f0e424fdc34












4.4 Metrics / CallGraph



4.5 Metrics / Source Lines & Risk





4.6 Metrics / Capabilities


Solidity Versions observed		 Experimental Features		 Can Receive Funds		 Uses Assembly		 Has Destroyable Contracts			
<input type="text" value="0.5.0"/>				<input type="text"/>		<input type="text"/>		<input type="text" value="yes"/>			
 Transfers ETH		 Low-Level Calls		 DelegateCall		 Uses Hash Functions		 ECRecover		 New/Create/Create2	
<input type="text" value="yes"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>		<input type="text"/>	
 TryCatch		Σ Unchecked									

Exposed Functions





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

 Public	 Payable			
<input type="text" value="17"/>	<input type="text" value="0"/>			
External	Internal	Private	Pure	View
<input type="text" value="6"/>	<input type="text" value="12"/>	<input type="text" value="0"/>	<input type="text" value="0"/>	<input type="text" value="5"/>

StateVariables

Total	 Public
<input type="text" value="12"/>	<input type="text" value="10"/>

4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	LPWrapper.sol	3	_____	176	164	131	3	125	
	Totals	3	_____	176	164	131	3	125	

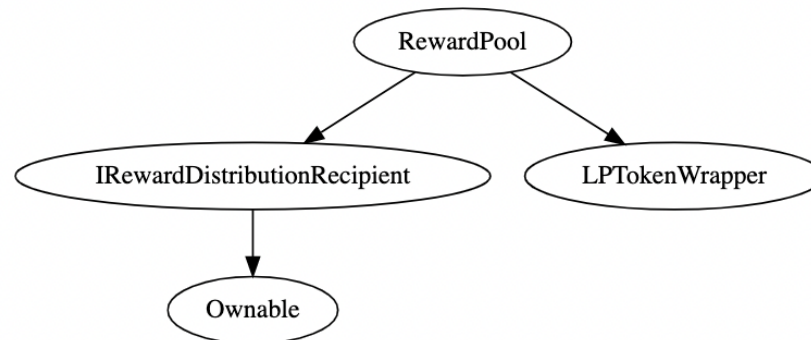
5. Scope of Work

The GAIA Everworld Team provided us with the files that needs to be tested. The scope of the audit is the LP Wrapper contract.

The team put forward the following assumptions regarding the security, usage of the contracts:

- The smart contract is coded according to the newest standards and in a secure way

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.



5.1 Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, Auditor Labs experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES

5.1.1 Selfdestruct owner

Severity: HIGH

Status: ACKNOWLEDGED

File(s) affected: LPWrapper.sol

Attack / Description	Code Snippet	Result/Recommendation
Use of selfdestruct: Can block calling contracts unexpectedly. Be especially careful if this contract is planned to be used by other contracts (i.e. library contracts, interactions). Selfdestruction of the callee contract can leave callers in an inoperable state.	Line: 172 - 174 <pre>function destroyContract() external onlyOwner { selfdestruct(msg.sender); }</pre>	The function destroyContract can be called only by the Owner, but once called destroys the contract and can cause problems for current staker. Consider removing that function or make sure the owner is controlled by a multisig, to prevent single point of failure such as hacks or phishing attacks.

MEDIUM ISSUES

During the audit, Auditor Labs experts found **no Medium issues** in the code of the smart contract.

LOW ISSUES

During the audit, Auditor Labs experts found **no Low issues** in the code of the smart contract.

INFORMATIONAL ISSUES

5.1.2 Floating compiler versions

Severity: INFORMATIONAL

Status: ACKNOWLEDGED

Code: SWC-103

File(s) affected: ALL



Attack / Description	Code Snippet	Result/Recommendation
The current pragma solidity directive is floating. It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.	<code>pragma solidity ^0.5.0;</code>	<p>It is recommended to follow the latter example, as future compiler versions may handle certain language constructions in a way the developer did not foresee. i.e. Pragma solidity 0.5.0</p> <p>See SWC-103: https://swcregistry.io/docs/SWC-103</p>

5.2 SWC Attacks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✓
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✓
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✓
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✓
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✓
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✓
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✓
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✓

ID	Title	Relationships	Test Result
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✓
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✓
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✓
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✓
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	✓
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✓

ID	Title	Relationships	Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✓
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✓
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✓
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✓
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✓
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✓
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✓
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✓

ID	Title	Relationships	Test Result
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	X
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	X
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	

6. Executive Summary

Two (2) independent Auditor Labs experts performed an unbiased and isolated audit of the smart contract codebase. The final debriefs took place on the December 14, 2021.

The main goal of the audit was to verify the claims regarding the security of the smart contract. During the audit, no critical issues were found, after the manual and automated security testing and the claim have been successfully verified. Only the selfdestruct function can have an impact on the contract availability and can be further secured with a multisig.

7. Deployed Smart Contract

VERIFIED

GAIA X MATIC - <https://polygonscan.com/address/0xc746d3226af9ecb6b0eb80c1799e8bc22a1114e5#code>

GAIA X DAI - <https://polygonscan.com/address/0xaBE3AB72b608237d80bE59854bD9aD74c35F5b4F#code>

GAIA X BNB <https://bscscan.com/address/0xfbd25f0e7943f7b0d101e59e37337cdf37ec9676#code>