OVERVIEW

PROJECT SUMMARY

Project	**EthLlamasNFT**
Platform	N/a
Language	Solidity

AUDIT SUMMARY

Date	08-03-2022
Audit Type	Static Analysis, Manual Review
Audit Result	**PASSED**
Auditor	**Jarmo van de Seijp** https://tinyurl.com/Jvdseijp

RISK SUMMARY

Risk Level	Total	Found	Pending	Solved	Acknowledged	Objected
Critical	0	0	0	0	0	0
Major	0	0	0	0	0	0
Medium	1	1	0	0	1	0
Minor	3	3	0	2	1	0
Informative	5	5	0	3	2	0
Discussion	0	0	0	0	0	0

FINDINGS

Function Default Visibility

SWC-ID: SWC-100

Relationship:
CWE-710: Improper Adherence to Coding Standards

Description:
Functions that do not have a function visibility type specified are public by default. This can lead to a vulnerability if a developer forgot to set the visibility and a malicious user is able to make unauthorized or unintended state changes.

Relevance:

getEthllamas, transferOwnership, renounceOwnership should be declared external

Category	Risk Level	Number of Findings	Status
SWC-100	informational	**3**	Solved

Constable State

Relationship:
CWE-710: Improper Adherence to Coding Standards

Description:

Constant state variables should be declared constant to save gas.

Relevance:

_ownerWallet should be declared constant

Category	Risk Level	Number of Findings	Status
SWC-108	informational	**1**	Solved

Unused Code

SWC-ID: SWC-131

Relationship:
CWE-1164: Irrelevant Code

Description:
Unused variables are allowed in Solidity and they do not pose a direct security issue.
It is best practice though to avoid them as they can:

- cause an increase in computations (and unnecessary gas consumption)

- indicate bugs or malformed data structures and they are generally a sign of poor code quality

- cause code noise and decrease readability of the code

Relevance:
_totalReserved is never used

Category	Risk Level	Number of Findings	Status
SWC-131	Informational	**1**	Solved

Note:
a lot of the inherited from libraries contain unused code. I can give you the full list, which is quite big, but that's only relevant if you intend to change the imports.

Missing Events

Description:
The contract may change significant state variables in the contract, but does not emit these changes in events. This may result in lack of transparency or 3rd party applications being unable to properly register the contract's current state

Relevance:

setCost and **setMaxMintAmount** should emit an event (expecially **setCost**)

Category	Risk Level	Number of Findings	Status
Missing events	minor	**2**	Solved

Risk Of Centralization

Description:
The owner of the contract has the power to significantly change the economics from within the contract. All Fees, Taxes and Swap&Liquify are controlled without governance. Losing access to the owner account would result in permanent loss of control of these functions

Category	Risk Level	Number of Findings	Status
Control Flow	Minor	**1**	Acknowledged

Access Control

Description:
The owner of the contract is transferred to an arbitrary address without checking if the recipient is able to accept ownership, or is a contract address with no method of controlling the ownership functions. In case of a mistakenly transferred ownership, it would be lost permanently

Category	Risk Level	Number of Findings	Status
Push-Over-Pull	Medium	**1**	Acknowledged

AUDIT RESULT

Basic Coding Bugs

1. Constructor Mismatch

o Description: Whether the contract name and its constructor are not identical to each other.

o Result: PASSED

o Severity: Critical

Ownership Takeover

o Description: Whether the set owner function is not protected.

o Result: PASSED

o Severity: Critical

Redundant Fallback Function

o Description: Whether the contract has a redundant fallback function.

o Result: PASSED

o Severity: Critical

Overflows & Underflows

Description: Whether the contract has general overflow or underflow Vulnerabilities

o Result: PASSED

o Severity: Critical

Reentrancy

o Description: Reentrancy is an issue when code can call back into your contract and change state, such as withdrawing ETHs.

o Result: PASSED

o Severity: Critical

MONEY-Giving Bug

o Description: Whether the contract returns funds to an arbitrary address.

o Result: PASSED

o Severity: High

Blackhole

- *Description: Whether the contract locks ETH indefinitely: merely in without out.*
- *Result: PASSED*
- *Severity: High*

Unauthorized Self-Destruct

- *Description: Whether the contract can be killed by any arbitrary address.*
- *Result: PASSED*
- *Severity: Medium*

Revert DoS

- *Description: Whether the contractis vulnerable to DoSattack because of unexpected revert.*
- *Result: PASSED*
- *Severity: Medium*

Unchecked External Call

- *Description: Whether the contract has any external call without checking the return value.*
- *Result: PASSED*
- *Severity: Medium*

Gasless Send

- *Description: Whether the contractis vulnerable to gasless send.*
- *Result: PASSED*
- *Severity: Medium*

Send Instead of Transfer

- *Description: Whether the contract uses send instead of transfer.*
- *Result: PASSED*
- *Severity: Medium*

Costly Loop

o Description: Whether the contract has any costly loop which may lead to Out-Of-Gas exception.

o Result: PASSED

o Severity: Medium

(Unsafe) Use of Untrusted Libraries

o Description: Whether the contract use any suspicious libraries.

o Result: PASSED

o Severity: Medium

(Unsafe) Use of Predictable Variables

o Description: Whether the contract contains any randomness variable, but its value can be predicated.

o Result: PASSED

o Severity: Medium

Transaction Ordering Dependence

o Description: Whether the final state of the contract depends on the order of the transactions.

o Result: PASSED

o Severity: Medium

. Deprecated Uses

o Description: Whether the contract use the deprecated tx.origin to perform the authorization.

o Result: PASSED

o Severity: Medium