Autograph

Audit

Presented by:

OtterSec contact@osec.io

Akash Gurugunti Sud0u53r.ak@osec.io

Ajay Kunapareddy d1r3wolf@osec.io

Contents

01 Executive Summary **2**

Overview 2

Key Findings 2

02 Scope **3**

03 Findings **4**

04 Vulnerabilities **5**

OS-ATG-ADV-00 [high] | Improper Implementation Of VRF Requesting 6

OS-ATG-ADV-01 [high] | Improper Selection Of Winners 7

05 General Findings **8**

OS-ATG-SUG-00 | Missing Length Checks 9

OS-ATG-SUG-01 | Unnecessary Metadata Mapping 10

OS-ATG-SUG-02 | Gas Optimizations 11

OS-ATG-SUG-03 | Additional Checks In RequestRandomWords 12

OS-ATG-SUG-04 | Insufficient Validation Of JSON Metadata 13

Appendices

A Vulnerability Rating Scale **14**

B Procedure **15**

01 | Executive Summary

Overview

Autograph engaged OtterSec to perform an assessment of the `smart-contracts-audit-repo` program. This assessment was conducted between March 31st and April 3rd, 2023. For more information on our auditing methodology, see Appendix B.

Key Findings

Over the course of this audit engagement, we produced 7 findings total.

In particular, we found an issue with the selection of random winners from the list of participants (OS-ATG-ADV-01), along with the request and response implementation in the `chainlink` integration that may lead to unexpected behaviours (OS-ATG-ADV-00).

We also made recommendations around several input validation issues (OS-ATG-SUG-04), missing length checks in the `traits` and `values` arrays (OS-ATG-SUG-00), and gas optimization (OS-ATG-SUG-02).

02 | Scope

The source code was delivered to us in a git repository at github.com/Autograph-Core/Smart-Contracts-Audit-Repo. This audit was performed against commit a29a85c.

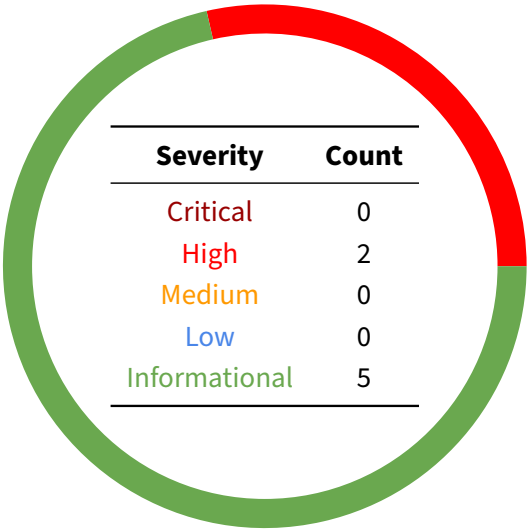
A brief description of the programs is as follows.

Name	Description
Chainlink VRF	Integration of Chainlink VRF to select a random group of winners from a larger group of participants.
Marketplace Filtering	Integration with Opensea Marketplace filterer to filter out blacklisted contracts from interacting with NFTs.
LayerZero	Integration with LayerZero to build cross-chain NFTs that can be transferred between chains.

03 | Findings

Overall, we reported 7 findings.

We split the findings into **vulnerabilities** and **general findings**. Vulnerabilities have an immediate impact and should be remediated as soon as possible. General findings don't have an immediate impact but will help mitigate future vulnerabilities.



Severity	Count
Critical	0
High	2
Medium	0
Low	0
Informational	5

04 | Vulnerabilities

Here, we present a technical analysis of the vulnerabilities we identified during our audit. These vulnerabilities have *immediate* security implications, and we recommend remediation as soon as possible.

Rating criteria can be found in Appendix A.

ID	Severity	Status	Description
OS-ATG-ADV-00	High	Resolved	Improper implementation of Chainlink VRF requesting leads to unexpected behaviour.
OS-ATG-ADV-01	High	Resolved	Improper implementation of selecting winners from the list of participants.

OS-ATG-ADV-00 [high] | Improper Implementation Of VRF Requesting

Description

In `VRFv2Consumer.sol`, `requestRandomWords` is used to request a list of random words from the Chainlink VRF Coordinator. This function should only be called when there are no pending requests needed to be fulfilled. `currentRNGName` and `s_requestId` are used to store the name and ID of the current pending requests.

The current implementation does not check if there is a pending request before sending a new request. This results in overwriting the `currentRNGName` and `s_requestId` variables. As a result, a response to the past request may be fulfilled as a response to the current pending request, which may lead to unexpected behaviours.

Proof of Concept

Consider the following scenario:

1. A request (request1) has been raised by calling `requestRandomWords`, requesting random words to select five winners from a list of 20 participants.
2. Another request (request2) has been raised while the first request is still pending, requesting random words to select three winners from a list of five participants.
3. The response for the first request is fulfilled, while the second request is considered as the current pending request.
4. Since the `requestId` is not validated in `fulfillRandomWords`, the `randomWords` of length five is used to calculate winners from the list of five participants, resulting in all participants becoming the winners.

Remediation

Validate whether the name and ID of the current pending request, such as, `currentRNGName` and `s_requestId`, have zero values before sending the request in `requestRandomWords`. Additionally, set these variables to zero values after consuming the `randomWords` in `fulfillRandomWords`. Also, validate whether the `requestId` parameter is equal to the `s_requestId` in `fulfillRandomWords`.

Patch

Fixed in 8004a0c.

OS-ATG-ADV-01 [high] | Improper Selection Of Winners

Description

In `VRFv2Consumer.sol`, `fulfillRandomWords` is triggered by the `vrfCoordinator` to fulfill the request for random words. The `randomWords` is then used to select a random list of participants as `winners`.

This leads to two primary issues:

1. Storing the indexes of the winners from the list of participants is not a preferred method. As calculating `randomWords[i] % _RNGInstance.participants.length` may result in duplicate indexes, which then results in duplicate members in the list of winners. Additionally, the removal of duplicates result in a fewer number of winners than expected.
2. The process for which the participants are deleted from the list is faulty, as `delete` sets the array item to its default value and does not reduce the length of the array.

The combination of the two issues provided may lead to unexpected behaviours. For example, the list of the `winners` may contain empty strings when `getRNGWinners` is called.

Remediation

Directly store the chosen participant value in the `winners` list instead of the index. Additionally, properly remove the chosen participant from the `participants` array by swapping it with the last element, then utilizing the pop method to remove the last element.

Patch

Fixed in ac66a86.

05 | General Findings

Here, we present a discussion of general findings during our audit. While these findings do not present an immediate security impact, they represent antipatterns and could lead to security issues in the future.

ID	Description
OS-ATG-SUG-00	Missing length checks for traits and values arrays while setting metadata.
OS-ATG-SUG-01	Unnecessary mapping for metadata when only one key is used.
OS-ATG-SUG-02	Recommendations for gas optimization.
OS-ATG-SUG-03	Implement additional checks in `requestRandomWords` to prevent unnecessary requests.
OS-ATG-SUG-04	Insufficient validation of metadata may break JSON encoding.

OS-ATG-SUG-00 | Missing Length Checks

Description

In `AutographDynamicNFTOptionalMarketplaceFilteringNonUpgradeable.sol`, `constructor` and `setMetadataForAll` take `traits` and `values` arrays as inputs and store them in the `metadata` variable. These inputs are then used as keys and values in `tokenURI`.

However, it is important to note that the lengths of the `traits` and `values` arrays must be asserted to be equal. Otherwise, it is possible that they may have different lengths, leading to inconsistency in `tokenURI`'s data.

Remediation

Add a constraint to check if the lengths of `traits` and `values` are equal in `constructor` and `setMetadataForAll` to prevent inconsistency in `tokenURI`'s data.

Patch

Fixed in 8004a0c by adding the length checks.

OS-ATG-SUG-01 | Unnecessary Metadata Mapping

Description

In `AutographDynamicNFTOptionalMarketplaceFilteringNonUpgradeable.sol`, `metadata` is used to store the metadata for a token with `tokenId` as the key and its `metadata` as the value.

However, there is no functionality to insert or update the metadata for any `tokenId`s. As a result, only the default metadata, such as, `metadata[0]`, is inserted and used. This deems the mapping unnecessary since the default metadata may be directly stored and used without implementing the mapping.

Remediation

If the intended behaviour is to only use the default metadata for the tokens, then store the default metadata directly without using the mapping. This reduces the complexity of the contract and promotes readability and understanding.

On the other hand, if the intended behaviour is to store different metadata for each `tokenId`, then implement a function that can insert or update metadata for any `tokenId`. This function should only be accessible by the `upgrader` or `admin` to ensure proper data consistency.

Additionally, `tokenURI` must be updated such that it uses the metadata for the `tokenId` if it exists, and uses the default metadata if it does not. This ensures that the correct metadata is returned for each token while preventing any unexpected behaviours or inconsistencies.

Patch

Fixed in 8004a0c by removing mapping for metadata and using a single instance.

OS-ATG-SUG-02 | Gas Optimizations

Description

There are several recommendations to promote gas optimization in `VRFv2Consumer.sol` and `AutographDynamicNFTOptionalMarketplaceFilteringNonUpgradeable.sol`:

1. In `VRFv2Consumer.sol`, `fulfillRandomWords`'s `randomNumber` field in `CompletedRandomNumberGeneratorRollWinner` should be changed to `randomWords[i]` to emit the random number that leads to the selection of the respective winner. Instead of calculating the value of `randomWords[i] % _RNGInstance.participants.length` multiple times, store the value in a variable for utilization instead.
2. In `AutographDynamicNFTOptionalMarketplaceFilteringNonUpgradeable.sol`'s `generateMetadata`, the if case `if (t != metadata[0].traits.length - 1) {` may be removed from the loop by running the loop just up to `metadata[0].traits.length - 1` and adding the last element separately outside the loop.

Remediation

Integrate the recommendations provided to optimize gas usage.

OS-ATG-SUG-03 | Additional Checks In RequestRandomWords

Description

In `VRFv2Consumer.sol`, the input parameters in `requestRandomWords` are insufficiently validated. Additional checks may be added to prevent unnecessary requests from being sent to `chainlink`.

The value of `numOfWinners` is not validated to be greater than zero, while the value of `rngName` is not asserted to be a non-empty string.

Remediation

Validate the `numOfWinners` to be greater than zero and the value of `rngName` to be a non-empty string.

Patch

Fixed in 8004a0c by validating the `numOfWinners` and `rngName` to be non-zero values.

OS-ATG-SUG-04 | Insufficient Validation Of JSON Metadata

Description

In `AutographDynamicNFTOptionalMarketplaceFilteringNonUpgradeable.sol`, `setMetadataForAll` is used by the `upgrader` to update the default metadata for the `ERC721` tokens. The input arguments are stored in the `metadata` state variable, which is later utilized to generate a `tokenURI` in `tokenURI`'s JSON encoding.

However, since the inputs are not properly validated, this may potentially lead to the breaking of the JSON encoding. This is because there are several characters such as `"` and `.` that need to be escaped before inserting them into JSON strings by concatenation.

Remediation

Validate the inputs for `setMetadataForAll` and ensure that there are no characters in the strings that may break the JSON encoding.

A | Vulnerability Rating Scale

We rated our findings according to the following scale. Vulnerabilities have immediate security implications. Informational findings can be found in the General Findings section.

Critical	Vulnerabilities that immediately lead to loss of user funds with minimal preconditions Examples: • Misconfigured authority or access control validation • Improperly designed economic incentives leading to loss of funds
High	Vulnerabilities that could lead to loss of user funds but are potentially difficult to exploit. Examples: • Loss of funds requiring specific victim interactions • Exploitation involving high capital requirement with respect to payout
Medium	Vulnerabilities that could lead to denial of service scenarios or degraded usability. Examples: • Malicious input that causes computational limit exhaustion • Forced exceptions in normal user flow
Low	Low probability vulnerabilities which could still be exploitable but require extenuating circumstances or undue risk. Examples: • Oracle manipulation with large capital requirements and multiple transactions
Informational	Best practices to mitigate future security risks. These are classified as general findings. Examples: • Explicit assertion of critical internal invariants • Improved input validation

B | Procedure

As part of our standard auditing procedure, we split our analysis into two main sections: design and implementation.

When auditing the design of a program, we aim to ensure that the overall economic architecture is sound in the context of an on-chain program. In other words, there is no way to steal funds or deny service, ignoring any chain-specific quirks. This usually requires a deep understanding of the program's internal interactions, potential game theory implications, and general on-chain execution primitives.

One example of a design vulnerability would be an on-chain oracle that could be manipulated by flash loans or large deposits. Such a design would generally be unsound regardless of which chain the oracle is deployed on.

On the other hand, auditing the implementation of the program requires a deep understanding of the chain's execution model. While this varies from chain to chain, some common implementation vulnerabilities include reentrancy, account ownership issues, arithmetic overflows, and rounding bugs.

As a general rule of sum, implementation vulnerabilities tend to be more "checklist" style. In contrast, design vulnerabilities require a strong understanding of the underlying system and the various interactions: both with the user and cross-program.

As we approach any new target, we strive to get a comprehensive understanding of the program first. In our audits, we always approach targets with a team of auditors. This allows us to share thoughts and collaborate, picking up on details that the other missed.

While sometimes the line between design and implementation can be blurry, we hope this gives some insight into our auditing procedure and thought process.