GA GUARDIAN

Animecoin

Anime Claimer

Security Assessment

January 18th, 2025

Summary

Audit Firm Guardian

Prepared By Owen Thurm, Daniel Gelfand, Kose Dogus, Wafflemakr

Client Firm Animecoin

Final Report Date January 18, 2025

Audit Summary

Animecoin engaged Guardian to review the security of their cross-chain token claimer. From the 19th of December to the 24th of December, a team of 4 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Issues Detected Throughout the engagement 3 High/Critical issues were uncovered and promptly remediated by the Animecoin team. Several issues impacted the fundamental behavior of the protocol, following their remediation Guardian believes the protocol to uphold the functionality described for the claimer.

For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

🔗 Blockchain network: **Arbitrum**

✅ Verify the authenticity of this report on Guardian's GitHub: https://github.com/guardianaudits

📊 Code coverage & PoC test suite: https://github.com/GuardianAudits/anime-claimer-1

Table of Contents

Project Overview

Project Summary

Project Name	Animecoin
Language	Solidity
Codebase	https://github.com/chiru-labs-org/anime-claimer-l2-only
Commit(s)	d3092325c9bc879dfbd9c61f2b3d7342ba8246bc

Audit Summary

Delivery Date	January 17, 2025
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	1	0	0	0	0	1
High	2	0	0	0	0	2
Medium	3	0	0	0	0	3
Low	15	0	0	7	0	8

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	🔴 Critical	🟠 High	🟡 Medium
Likelihood: *Medium*	🟠 High	🟡 Medium	🟢 Low
Likelihood: *Low*	🟡 Medium	🟢 Low	🟢 Low

Impact

High Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.

Medium A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.

Low Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.

Medium An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.

Low Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts. Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Findings & Resolutions

ID	Title	Category	Severity	Status
C-01	Claims By The Collector Prevented	Logical Error	Critical	Resolved
H-01	Users Incorrectly Credited WIth NFT Ownership At lzReceive Time	Logical Error	High	Resolved
H-02	Collector Allocations Errantly Validated	Logical Error	High	Resolved
M-01	Insufficient Gas Estimated	Logical Error	Medium	Resolved
M-02	Incompatible Types	Logical Error	Medium	Resolved
M-03	Sanctions Can Be Avoided	Unexpected Behavior	Medium	Resolved
L-01	Missing _logAdminAccess Calls	Events	Low	Resolved
L-01	Last Withdrawn Day Getter Missing	Composability	Low	Acknowledged
L-02	Typo	Typo	Low	Resolved
L-02	View Functions Not Accurate	Typo	Low	Acknowledged
L-03	Incorrect Comment	Documentation	Low	Resolved
L-04	Refund Receiver Is Always The Sender	Unexpected Behavior	Low	Acknowledged
L-04	Delegate Registry On Multiple Chains	Documentation	Low	Acknowledged

Findings & Resolutions

ID	Title	Category	Severity	Status
L-05	Claims Must Always Go To The Claimer	Unexpected Behavior	Low	Acknowledged
L-05	Sanctions Could Be Validated On Arbitrum	Documentation	Low	Acknowledged
L-06	Overallocated Expected Calldata	Optimization	Low	Resolved
L-07	Runbook Typo	Documentation	Low	Resolved
L-08	Misleading Comment	Documentation	Low	Resolved
L-09	recoverCalldata Optimization	Optimization	Low	Resolved
L-10	Unnecessary Ownership Timestamp Check	Optimization	Low	Resolved
L-11	Fixed Term Loans Can Claim Outstanding Vests	Documentation	Low	Acknowledged

C-01 | Claims By The Collector Prevented

Category	Severity	Location	Status
Logical Error	● Critical	AnimeClaimer.sol: 815	Resolved

Description

In the `_getClaimChecksFromVestingConfigs` function the `checks.collectors` array is assigned a length based on the `numCollectors` which is purely the number of configs which satisfy `c.collector = address(0)`.

However the criteria for an address to be added to the `checks.collectors` array is `isForCollector & co.collector = msg.sender`.Therefore the `checks.collectors` array will have empty entries for the configs that are claimed by the collector address themselves.

The `checks.collectors` array with empty entries is then used to verify the claim with the `ClaimChecker` contract on L1. The `_checkCollectorClaim(claimer, collectors[i])` check will trivially fail because no claimer can be approved in the delegate registry for the zero address.

Therefore these claims cannot be verified and cannot be completed.

Recommendation

Shrink the length of the `checks.collectors` array after writing to it or adjust the way the `numCollectors` is calculated so that it is only incremented in the case where `isForCollector && c.collector = msg.sender`.

Resolution

Animecoin Team: The issue was resolved in PR#10.

H-01 | Users Incorrectly Credited WIth NFT Ownership At lzReceive Time

Category	Severity	Location	Status
Logical Error	High	AnimeClaimer.sol: 844	Resolved

Description

In the `_claimBatch` function after the claim has been validated by the l1 state at the timestamp of the `requestClaim` call, the claimer is awarded with the vested amount computed by the `_vested` function.

However the `_vested` function determines the user's vested amount based upon the current `block.timestamp` of the `lzReceive` action. This is however not the same timestamp that the claimer was verified to be authorized to claim at.

The claimer could have sold their NFT to another user after the `block.timestamp` of the `requestClaim` call and thus received vested amounts which should have gone to the new owner of the NFT.

There is a 8 block confirmation threshold currently configured, meaning that roughly 96 seconds will have to occur at minimum between the `requestClaim` call and the `lzReceive` which fulfills the claim.

This delay period could be made even more severe during a DVN outage or if a claim has passed all DVN checks but the `lzReceive` function reverts for some time until the revert is resolved (e.g. claim contract is paused and unpaused, or the daily withdrawal threshold has been met for the day) and then the `lzReceive` function is invoked again successfully passing a significant amount of time later.

Recommendation

Include the timestamp of the claim request in the `ClaimParameters` for the corresponding `claimNonce` if the claimer is shown to be validated for all of the claims then the claims should be carried out up to the stored timestamp of the claim request.

Resolution

Animecoin Team: The issue was resolved in PR#22.

H-02 | Collector Allocations Errantly Validated

Category	Severity	Location	Status
Logical Error	High	AnimeClaimer.sol: 814	Resolved

Description

In the `AnimeClaimer` contract when the claimer is the collector for a collector vest then there is no further validation performed on the vest and the vested amount is granted to the collector address on the L2.

However the owner of the collector address on the L2 may be different then the owner of the collector address on the L1.

For example, a smart contract wallet/multisig could have been transferred from Alice to Bob on the L1, but Alice may have kept ownership of a smart contract wallet/multisig deployed at the same address on the L2.

Recommendation

Ensure that all collector addresses which are awarded are EOAs on both chains. If this is not the case then a more adept verification process is necessary.

Resolution

Animecoin Team: The issue was resolved in PR#14.

M-01 | Insufficient Gas Estimated

Category	Severity	Location	Status
Logical Error	Medium	AnimeClaimer.sol: 308	Resolved

Description

In the `requestClaim` function the `checks.numNFTs` and `checks.numCollectors` values are used to compute the gas that should be provided to the `lzReceive` function on the L2.

However the `checks.numCollectors` value only includes the number of collector claims that need to be verified on the L1, which does not include claims where the collector is the claimer.

These claims will still have to be iterated over in the `lzReceive` function though, and thus should be accounted for in the gas estimation.

Recommendation

Consider basing the gas estimation in the `_lzOptions` function on the `configs.length` instead as this more closely represents the iterations that must be made in the `lzReceive` function.

Resolution

Animecoin Team: The issue was resolved in PR#11.

M-02 | Incompatible Types

Category	Severity	Location	Status
Logical Error	Medium	AnimeClaimer.sol	Resolved

Description

A `VestingConfig` object contains `uint256 streamId` but the `VestingConfigStorage` contains `uint8 streamId`. Consequently, some user configurations will be unclaimable when the `uint256` type is cast to `uint8` within `_saveClaimParameters`, as it will revert with error `Overflow()`.

Recommendation

Consider keeping types consistent or clearly document this behavior.

Resolution

Animecoin Team: The issue was resolved in PR#25.

M-03 | Sanctions Can Be Avoided

Category	Severity	Location	Status
Unexpected Behavior	Medium	AnimeClaimer.sol	Resolved

Description

Currently `checkClaims` only validates sanctions against the claimer: `if (isSanctioned(claimer)) return (claimNonce, false);` However, there's is no validation that the actual owner of the NFT is not sanctioned in the case that the claimer is a delegate.

Recommendation

Consider adding sanctions validation on the NFT owner, otherwise clearly document this behavior.

Resolution

Animecoin Team: The issue was resolved in PR#13.

L-01 | Missing _logAdminAccess Calls

Category	Severity	Location	Status
Events	Low	AnimeClaimer.sol: 460, 467	Resolved

Description

In the `setReadChannel` and `setReadConfirmations` `onlyOwner` functions there is no call to the `_logAdminAccess` function to emit an event for these admin updates.

Additionally, in the `OAppCore` contract there is a `setDelegate` `onlyOwner` function which is not overridden and thus will not emit a `AdminAccessed` event.

Recommendation

Consider adding a `_logAdminAccess` invocation to these functions.

Resolution

Animecoin Team: The issue was resolved in PR#15.

L-01 | Last Withdrawn Day Getter Missing

Category	Severity	Location	Status
Composability	Low	AnimeClaimer.sol	Acknowledged

Description

The AnimeClaimer includes many getters to query the state of the system but is missing a getter for `lastWithdrawnDay`.

Recommendation

Consider adding a getter for `lastWithdrawnDay`.

Resolution

Animecoin Team: Acknowledged.

L-02 | Typo

Category	Severity	Location	Status
Typo	Low	AnimeClaimer.sol: 288	Resolved

Description

In the comment for the `requestClaim` function the `_lzReceive` function is referred to as `_lsReceive`.

Recommendation

Correct this to `_lzReceive`.

Resolution

Animecoin Team: The issue was resolved in PR#16.

L-02 | View Functions Not Accurate

Category	Severity	Location	Status
Typo	Low	AnimeClaimer.sol	Acknowledged

Description

Function `dailyTotalWithdrawn()` aims to return the current `dailyTotalWithdrawn`, however the value returned may be stale since the current day be different from the `lastWithdrawnDay` but `_resetDailyTotalWithdrawnIfNewDay` hasn't been triggered yet.

This can be problematic for integrators reading this value and assuming the current day has already had withdraws when it hasn't.

Recommendation

Consider adding logic within `dailyTotalWithdrawn()` to reflect the start of a new day.

Resolution

Animecoin Team: Acknowledged.

L-03 | Incorrect Comment

Category	Severity	Location	Status
Documentation	Low	AnimeClaimer.sol: 144	Resolved

Description

In the comment for the `uuidSigner` value in the `AnimeClaimerStorage` struct it is mentioned that this value `SHOULD be configured`, however this value must be configured along with the other values that are documented as such.

This is because the `uuidSigner` must be configured to a nonzero address for the `readyForClaim` function to return true.

Recommendation

Correct the comment to indicate that the `uuidSigner` qualifies as a variable which `MUST be configured`.

Resolution

Animecoin Team: The issue was resolved in PR#17.

L-04 | Refund Receiver Is Always The Sender

Category	Severity	Location	Status
Unexpected Behavior	Low	Global	Acknowledged

Description

The `requestClaim` function assigns the `msg.sender` as the refund receiver for a native refund.
As a result, Smart Contracts which receive a collector allocation, do not have the functionality to delegate, and do not have a receive function cannot claim their vest.

Recommendation

This scenario is unlikely to occur, especially for a contract which is able to claim from the claimer. However if this issue is desired to be solved, or additional utility should be added, a `refundReceiver` parameter can be added to and used within the `requestClaim` function.

Resolution

Animecoin Team: Acknowledged.

L-04 | Delegate Registry On Multiple Chains

Category	Severity	Location	Status
Documentation	Low	Global	Acknowledged

Description

Currently delegation is validated solely through the `DelegateRegistry` contract deployed on Ethereum within function `checkClaims`. Consequently, delegations through the `DelegateRegistry` contract deployed on Arbitrum will not allow for the receipt of allocations.

Recommendation

Clearly document this behavior to users.

Resolution

Animecoin Team: Acknowledged.

L-05 | Claims Must Always Go To The Claimer

Category	Severity	Location	Status
Unexpected Behavior	Low	AnimeClaimer.sol	Acknowledged

Description

In the previous version of the `AnimeClaimer` contract there was a `to` address which allowed users to `claimBatch` to their desired address on Arbitrum.

However in the new `AnimeClaimer` contract the allocations are always sent to the claimer who calls the `requestClaim` function.

This may decrease the simplicity of integrations/user interactions with the `AnimeClaimer`, especially for claims that are made on behalf of another collector using the delegate registry.

Recommendation

Consider adding a `to` parameter to the `requestClaim` function so that a claim can be sent to the configured address rather than always to the claimer.

Resolution

Animecoin Team: Acknowledged.

L-05 | Sanctions Could Be Validated On Arbitrum

Category	Severity	Location	Status
Documentation	Low	Global	Acknowledged

Description

The `ChainAnalysis` oracle for sanctioned addresses is also deployed on Arbitrum. Consequently, sanctions can be verified at the start of a `requestClaim` to prevent the message from even being sent, as well as `_lzReceive` to account for sanctions list changes in the case of delayed receipt (e.g. if there is a DVN outage).

Recommendation

Consider querying the Arbitrum oracle as well and accounting for any necessary gas usage increases, otherwise document this behavior.

Resolution

Animecoin Team: Acknowledged.

L-06 | Overallocated Expected Calldata

Category	Severity	Location	Status
Optimization	Low	AnimeClaimer.sol: 278	Resolved

Description

In the `AnimeClaimer` constructor the `expectedCalldataSize` which is denominated in calladata bytes is assigned as 256.

This is ultimately used as the calldata size option in the lz options blob passed to the send function. This is meant to estimate the cost of the additional calldata which is provided from the result of the read function on the L1 chain.

However the `checkClaims` function result will not return a payload of 256 bytes of calldata to the `lzReceive` function, it returns a payload of 64 bytes of calldata to the `lzReceive` function. One 32 byte word for the `claimNonce` and one padded 32 byte word for the `authorized` boolean.

This results in overpaying the executor for the additional calldata that is estimated to be required for the lzRead response payload on every claim.

Recommendation

Update the `expectedCalldataSize` to 64 bytes.

Resolution

Animecoin Team: The issue was resolved in PR#24.

L-07 | Runbook Typo

Category	Severity	Location	Status
Documentation	Low	Runbook	Resolved

Description

The provided runbook contains a typo in the instructions for deploying the `ClaimChecker` contract on Ethereum. The first step mentions `Only select “ethereum”, unselect “ethereum” and others`. However it should read `Only select “ethereum”, unselect “arbitrum” and others`.

Recommendation

Correct the runbook step to `Only select “ethereum”, unselect “arbitrum” and others`.

Resolution

Animecoin Team: Resolved.

L-08 | Misleading Comment

Category	Severity	Location	Status
Documentation	Low	AnimeClaimer.sol: 415, 421	Resolved

Description

In the comment for both implementations of the `quoteForClaim` function it is mentioned that the purpose of the function is to:

`Returns the amount of ETH in wei that needs to be passed into claimBatch`. However the `quoteForClaim` function is intended to return the amount of ETH in wei that needs to be passed into the `requestClaim` function.

Recommendation

Correct the comment above both implementations of the `quoteForClaim` functions to:

`Returns the amount of ETH in wei that needs to be passed into requestClaim.`

Resolution

Animecoin Team: The issue was resolved in PR#23.

L-09 | recoverCalldata Optimization

Category	Severity	Location	Status
Optimization	Low	AnimeClaimer.sol: 734	Resolved

Description

In the `_validateRequestClaim` function the ECDSA library `recover` function is used. However the config object with the `signature` to be verified is declared as `calldata`, therefore the `recoverCalldata` function should be used to save a couple hundred gas.

Recommendation

Use `recoverCalldata` instead of `recover`.

Resolution

Animecoin Team: The issue was resolved in PR#26.

L-10 | Unnecessary Ownership Timestamp Check

Category	Severity	Location	Status
Optimization	Low	ClaimChecker.sol: 70	Resolved

Description

In the `ClaimChecker` contract the `_checkNFTClaim` function implements specific logic for Azuki NFTs which asserts that ownership of the NFT being verified has not begun at this `block.timestamp`.

This is to prevent NFT flash loans from being used to flash loan an NFT and claim it's outstanding vest amount. However it is impossible to use an intra-transaction flash loan to prove ownership of an NFT because the DVNs will not check the state of the L1 at an intermediate transaction state.

Therefore this check for Azuki NFT ownership is no longer necessary and is instead preventing users from claiming their vest at the timestamp of when they actually began their ownership. This case is unlikely to affect UX, however it can be removed for simplicity.

Recommendation

Consider removing the specific logic for Azuki NFTs in the `_checkNFTClaim` function.

Resolution

Animecoin Team: The issue was resolved in PR#29.

L-11 | Fixed Term Loans Can Claim Outstanding Vests

Category	Severity	Location	Status
Documentation	Low	Global	Acknowledged

Description

There exist platforms such as Blur which support fixed term loans of NFTs which will be awarded allocations, such as Azukis.

For Azukis which have accrued an unclaimed vest allocation a fixed term loan can be made to allow the loaner to claim the outstanding vested amount even though they did not hold ownership over the NFT for the previous vesting period.

Recommendation

Be aware of this risk and document it for users who have their NFTs available for loan.

Resolution

Animecoin Team: Acknowledged.

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/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 and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits