GA GUARDIAN

Animecoin

Anime Claimer #2

Security Assessment

January 18th, 2025

Summary

Audit Firm Guardian

Prepared By Owen Thurm, Daniel Gelfand, Kose Dogus

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 29th of December to the 9th of January, a team of 3 auditors reviewed the source code in scope. All findings have been recorded in the following report.

Issues Detected Throughout the engagement 1 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)	3ab039b4011ce2b11d379999ce28749e52c35ce2

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	0	0	0	0	0	0
High	1	0	0	0	0	1
Medium	3	0	0	0	1	2
Low	8	0	0	4	0	4

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
H-01	OApp Frozen Due To Ownership Revert		High	Resolved
M-01	Vested Amounts Stolen After Ownership Transfer		Medium	Partially Resolved
M-02	setReadChannel Always Assigns lzReadChannel	Logical Error	Medium	Resolved
M-03	ERC721 Delegations With Rights Are Missed	Logical Error	Medium	Resolved
L-01	Dangerous NFT To UUID Update		Low	Acknowledged
L-02	Unused DelegateCheckerLib		Low	Resolved
L-03	Claim Requests Allowed When At Daily Limit		Low	Acknowledged
L-04	Duplicate DVN Configured		Low	Resolved
L-05	Lacking Maximum Configs Validation		Low	Resolved
L-06	Dangerous Reentrancy Guard		Low	Acknowledged
L-07	Rights Key Does Not Follow Convention		Low	Resolved
L-08	Contracts Are Whitelisted For All Claims		Low	Acknowledged

H-01 | OApp Frozen Due To Ownership Revert

Category	Severity	Location	Status
	High	ClaimChecker.sol: 72	Resolved

Description

In the `_checkNFTClaim` function the owner of the `tokenId` is checked with the `ownerOf` function, however this function will often revert for many NFT collections and thus cause a DoS for the OAPP preventing all subsequent reads.

This occurs because the DVNs cannot process the read request, producing an error of `UnresolvableCommand` which prevents the DVN from verifying their response for the nonce.

When DVNs have not verified the response for a nonce, validation in the `EndpointV2` contract prevents any subsequent messages for the OApp from being processed in `lzReceive`:

https://github.com/LayerZero-Labs/LayerZero-v2/blob/7da76840e41dc593d3c2007ce35b911b1d816b4b/packages/layerzero-v2/evm/protocol/contracts/MessagingChannel.sol#L138

This produces a block of read messages on the chain which the invalid read request was triggered from. The delegate of the OApp can call the skip function on the endpoint in order to skip the affected nonce which cannot be verified and resume `lzReceive` processing.

However a malicious actor can continue to trigger read requests which will revert and place the OApp in a locked state.

Recommendation

Wrap the call to `ownerOf` in a try/catch and return false if the `ownerOf` call reverts as a validation response to indicate that the claim is not valid.

Resolution

Animecoin Team: Resolved.

M-01 | Vested Amounts Stolen After Ownership Transfer

Category	Severity	Location	Status
	Medium	AnimeClaimer.sol	Partially Resolved

Description

Because lzRead receptions can revert initially and be retried at a later time it is possible for malicious actors to steal vested amounts after an NFT has already been transferred to a new user.

For example, consider the following chain of events:
• Alice has Azuki #7 with 10 ANIME tokens vested on day 10
• The day 10 withdrawal limit has already been met on Arbitrum
• Alice intentionally submits a claim request that will fail upon lzReceive due to the withdrawal limit, but validates Alice as the owner of Azuki #7 up to block.timestamp of day 10, allowing Alice to claim 10 ANIME tokens
• On day 11 Alice lists their Azuki #7 for sale under the pretense of the buyer being able to claim the 10 vested ANIME tokens
• Azuki #7 is sold to Bob on day 11
• Alice retries their lzReceive on Arbitrum by invoking the lzReceive function through the endpoint themselves. The read result is still in the message channel so this action is allowed.
• Alice claims the 10 vested ANIME tokens that had accrued up to day 10
• Bob purchased Azuki #7 under the pretense that he would also receive these 10 vested ANIME tokens, however after his purchase Alice took those 10 ANIME tokens from him

Recommendation

Retries of lzRead results can be disabled by simply returning without claiming tokens in the lzReceive function. This would require users to re-submit a requestClaim transaction when their read request fails due to the withdrawal limit, contract pause, etc. However L2 transaction costs + lzRead costs are cheap so this may be acceptable.

Additionally, consider including validation that the withdrawal limit is not met for the current day in the requestClaim function to disallow users from potentially wasting gas + the lzRead fee when this scenario arises. This also allows for a quicker feedback loop on the withdrawal limit error.

Resolution

Animecoin Team: Partially Resolved.

M-02 | setReadChannel Always Assigns lzReadChannel

Category	Severity	Location	Status
Logical Error	Medium	AnimeClaimer.sol: 710	Resolved

Description

In the `setReadChannel` function the `active` parameter determines whether the channel is being activated or deactivated. However the `$.lzReadChannel = channelId;` assignment is always made no matter the `active` value.

Therefore when a read channel is deactivated it is assigned as the active `lzReadChannel`. This will put the app in an unintentional DoS'd state since the peer is no longer assigned for this channel.

Recommendation

Do not assign the `lzReadChannel` value if `active` is false.

Resolution

Animecoin Team: Resolved.

M-03 | ERC721 Delegations With Rights Are Missed

Category	Severity	Location	Status
Logical Error	Medium	ClaimChecker.sol: 90	Resolved

Description

In the `_checkNFTClaim` function the following boolean condition is used to determine if the claimer is authorized to initiate the ANIME claim for the `nftOwner`.

`DelegateCheckerLib.checkDelegateForERC721(claimer, nftOwner, nftContractAddress, tokenId) || _checkDelegateV2Rights(claimer, nftOwner)`

However the `_checkDelegateV2Rights` function does not check for ERC721 specific delegations, it checks for global delegations for the `nftOwner` address.

Therefore if the `nftOwner` issues an ERC721 token delegation to the claimer address with the specific `SUBDELEGATION_RIGHTS_KEY` the delegation will not be seen as a valid authorization of the claimer.

Recommendation

Create a new `_checkDelegateV2RightsForNft` function which uses `checkDelegateForERC721`.

Resolution

Animecoin Team: Resolved.

L-01 | Dangerous NFT To UUID Update

Category	Severity	Location	Status
	Low	AnimeClaimer.sol	Acknowledged

Description

The `setNFTToUUID` function allows the owner to update the `nftToUUID` value thus changing the UUID which an NFT corresponds to.

This implies that it may be possible that a signature is provided which shows that UUID X is associated with NFT A at one point in time and then subsequently after correction a signature is provided that shows that UUID X is associated with NFT B at another time.

If this is the case, then a malicious actor may retrieve the original signature which shows that UUID X is associated with NFT A and use this signature for their own NFT A to claim the vest of NFT B.

Also be aware that updating the NFT to UUID mapping allows for more than the original vesting allocation to be claimed, since the vesting state storage slot which stores the `withdrawnAmount` is unique to the nft and `tokenId`, not the UUID.

Recommendation

Be aware of these risks when making any updates with the signatures and `nftToUUID` mapping. To avoid the risk of stale signature replay in these scenarios consider assigning a new UUID signer after every update so that previous signatures cannot be used.

Otherwise every affected NFT can be force assigned the most recent correct UUID in the `nftToUUID` mapping. To resolve the `withdrawnAmount` discrepancy, consider including an admin function to update the `withdrawnAmount` of a user's claim.

Otherwise be sure to update the merkle proof accordingly to reduce the user's allocation by the already withdrawn amount.

Resolution

Animecoin Team: Acknowledged.

L-02 | Unused DelegateCheckerLib

Category	Severity	Location	Status
	Low	ClaimChecker.sol: 101	Resolved

Description

In the `ClaimChecker` contract the `_DELEGATE_REGISTRY_V2` is called directly in the `_checkDelegateV2Rights` function without using the `DelegateCheckerLib`.

However the `DelegateCheckerLib` performs a more efficient call, supporting the bytes32 rights parameter with the overloaded `checkDelegateForAll` function.

Recommendation

Consider using the `DelegateCheckerLib` for the `SUBDELEGATION_RIGHTS_KEY` check as well.

Resolution

Animecoin Team: Resolved.

L-03 | Claim Requests Allowed When At Daily Limit

Category	Severity	Location	Status
	Low	AnimeClaimer.sol	Acknowledged

Description

In the `requestClaim` function there is no validation that prevents users from requesting claims that will trivially fail when the daily withdrawal limit has been reached.

Recommendation

Consider adding validation which prevents users from requesting claims when the daily withdrawal limit has already been reached to save users from potentially wasting gas and LayerZero fees.

Resolution

Animecoin Team: Acknowledged.

L-04 | Duplicate DVN Configured

Category	Severity	Location	Status
	Low	layerzero.config.ts: 42	Resolved

Description

The `LayerZero` DVN is unnecessarily repeated as a required DVN and an optional DVN, however for the desired behavior of 2/3 DVNs with one of them always being `LayerZero` then the `LayerZero` DVN should be included only once in the `requiredDvns` list with the optional DVNs list being the other DVNs with an `optionalDVNThreshold` of 1.

Furthermore, currently there are 4 total unique DVNs included in the config file, meaning that one of the non-LayerZero DVNs should be removed from the optional DVNs list to make it a 2/3 total verification.

Recommendation

Remove the `LayerZero` DVN from the `optionalDVNs` list as well as another of the DVNs from the `optionalDVNs` list. Then reduce the `optionalDVNThreshold` to 1 to achieve a 2/3 total DVNs verification where one DVN must always be `LayerZero`.

Resolution

Animecoin Team: Resolved.

L-05 | Lacking Maximum Configs Validation

Category	Severity	Location	Status
	Low	AnimeClaimer.sol: 302	Resolved

Description

In the `AnimeClaimer` contract the `requestClaim` function does not include any validation on the maximum length of the `config` list.

This may be a good validation to include to avoid any unexpected behaviors while providing DVNs with requests that include an unnecessarily large amount of `tokenIds` to claim for.

It's not clear what the upper limit is for return data that the executor and/or DVNs can process and given the implications of a potentially stuck OApp like in C-01 it may be best to avoid this undefined behavior altogether by limiting the amount of configs provided to an individual `requestClaim` call to a reasonable amount.

Recommendation

Consider validating that the configs.length is within a reasonable limit.

Resolution

Animecoin Team: Resolved.

L-06 | Dangerous Reentrancy Guard

Category	Severity	Location	Status
	Low	ClaimChecker.sol: 35	Acknowledged

Description

The `checkClaims` function in the `ClaimChecker` contract uses a `nonReentrant` modifier to prevent any potentially malicious contracts from re-entering into the `checkClaims` function during the DVN simulation.

However if a malicious contract somehow does reenter into the `checkClaims` function and trigger the `nonReentrant` modifier to revert this will cause the OApp to enter a stuck state as described in C-01.

Recommendation

No pathway for this to occur has been identified, however out of an abundance of caution instead of using a `nonReentrant` modifier, consider using an `ENTERED` storage variable and setting it as true at the beginning of the `checkClaims` function.

Instead of reverting if a call reenters the `checkClaims` function, simply return false to indicate that the claim could not be verified. This allows a malicious actor to DoS a single claim, but they cannot freeze the OApp and DoS all subsequent read messages if this behavior is implemented.

Resolution

Animecoin Team: Acknowledged.

L-07 | Rights Key Does Not Follow Convention

Category	Severity	Location	Status
	● Low	ClaimChecker.sol: 20	Resolved

Description

The `SUBDELEGATION_RIGHTS_KEY` in the `ClaimChecker` contract is declared as the bytes of a string, however typically rights keys are hashed strings with keccak256. An example of this is the upcoming Shadow rights for APE chain: https://x.com/0xQuit/status/1869499764464906373

Recommendation

Consider using a hash of the `"AN_CLAIMER_PERMS"` string instead of the bytes of the string directly.

Resolution

Animecoin Team: Resolved.

L-08 | Contracts Are Whitelisted For All Claims

Category	Severity	Location	Status
	Low	ClaimChecker.sol: 81, 102	Acknowledged

Description

The `explicitContractClaimers` mapping is used both for NFT claims and collector claims. Therefore if a user explicitly designates a contract claimer then that claimer is able to claim all NFT claims and collector level claims for the user.

This may be unexpected for the user as they may expect to be able to designate a contract claimer as the claimer for an individual NFT or just NFT claims in general.

Recommendation

Be aware of this behavior, if intended then be sure to document it clearly to users.

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