GUARDIAN

Animecoin

Anime Claimer #3

Security Assessment

January 18th, 2025

Summary

Audit Firm Guardian

Prepared By Owen Thurm, Daniel Gelfand, windhustler

Client Firm Animecoin

Final Report Date January 17, 2025

Audit Summary

Animecoin engaged Guardian to review the security of its review of their cross-chain token claimer. From the 14th of January to the 16th of January, a team of 3 auditors reviewed the source code in scope. All findings have been recorded in the following report.

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 https://github.com/chiru-labs-org/anime-erc20
Commit(s)	anime-claimer-l2-only: e86b5e02aaafcbbdefac7031407c2ecdd6f2555e anime-erc20: 57fe3c92c71d72700f1f3c66259ab511baab2fe5

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	0	0	0	0	0	0
● Medium	0	0	0	0	0	0
● Low	12	0	0	7	1	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
L-01	Unused Import	Best Practices	● Low	Resolved
L-02	Unused Errors	Optimization	● Low	Resolved
L-03	isForNFT Optimization	Optimization	● Low	Acknowledged
L-04	Unnecessary Cast	Optimization	● Low	Acknowledged
L-05	Compromised Signer Griefing Attack	Griefing	● Low	Acknowledged
L-06	Delayed Claim Warning	Warning	● Low	Partially Resolved
L-07	Lacking Zero Configs Validation	Validation	● Low	Resolved
L-08	Anime Coin Trapped Ether	Trapped Ether	● Low	Acknowledged
L-09	Unused Contract	Best Practices	● Low	Acknowledged
L-10	Misleading Comment	Best Practices	● Low	Resolved
L-11	Merkle Tree Entries	Validation	● Low	Acknowledged
L-12	Incompatible Clock Mode	Compatibility	● Low	Acknowledged

L-01 | Unused Import

Category	Severity	Location	Status
Best Practices	● Low	AnimeClaimer.sol	Resolved

Description

The `EVMCallComputeV1` struct is imported in the `AnimeClaimer` however it is unused.

Recommendation

Consider removing the `EVMCallComputeV1` import.

Resolution

Animecoin Team: Resolved.

L-02 | Unused Errors

Category	Severity	Location	Status
Optimization	● Low	AnimeClaimer.sol	Resolved

Description

The InvalidClaimerType and ClaimIsZeroAddress errors in the AnimeClaimer contract are unused.

Recommendation

Remove the InvalidClaimerType and ClaimIsZeroAddress errors.

Resolution

Animecoin Team: Resolved.

L-03 | isForNFT Optimization

Category	Severity	Location	Status
Optimization	● Low	AnimeClaimer.sol	Acknowledged

Description

In the _validateRequestClaim function the isForNFT variable is declared outside of the UUID case, however it is only used inside of the UUID case. Therefore an optimization for non-UUID cases is to move the isForNFT declaration inside of the UUID case.

Recommendation

Move the isForNFT inside of the UUID case in the _validateRequestClaim function.

Resolution

Animecoin Team: Acknowledged.

L-04 | Unnecessary Cast

Category	Severity	Location	Status
Optimization	● Low	AnimeClaimer.sol: 365	Acknowledged

Description

In the _claimBatch function the s.withdrawn variable is cast to a uint256 type when calculating the withdrawAmount. However the s.withdrawn variable is already a uint256 type and therefore does not need to be cast.

Recommendation

Remove the unnecessary cast for the s.withdrawn variable.

Resolution

Animecoin Team: Acknowledged.

L-05 | Compromised Signer Griefing Attack

Category	Severity	Location	Status
Griefing	● Low	AnimeClaimer.sol: 766	Acknowledged

Description

In the _validateRequestClaim for UUID related claims the uuidToPackedNftID and nftToUUID mappings are written to and validate that the corresponding UUID to tokenId pairing is the only one used for either the UUID or tokenId in future claims.

This is to prevent a compromised UUID signer from being able to steal more than the amount of the unrevealed elemental's vests. However now that the uuidToPackedNftID mapping has been introduced a compromised signer gains another potentially harmful attack.

Consider the following scenario:
• Normal Azuki with ID 1 has the highest allocation out of all NFTs of 50 Million $ANIME tokens
• Alice does not own the Azuki with ID 1
• Alice gains access to the UUID signer
• Alice forges a signature that shows that Azuki with ID 1 corresponds to a UUID claim with only 10,000 $ANIME tokens
• Alice submits a requestClaim call, the nftToUUID and uuidToPackedNftID mappings are written with the errant pairing
• Alice's claim verification fails the claimChecker, since she does not own the Azuki with ID 1
• However Alice's requestClaim transaction was successful, so the mapping values remain
• The actual owner of Azuki ID 1 cannot make their normal claim and thus loses out on 49,990,000 $ANIME tokens

The compromised signer may repeat this attack for many of the largest token allocations, until low allocation UUID related claims run out.

Recommendation

Consider restricting UUID claims to only Azuki Elemental collection claims to limit this griefing attack vector from affecting large allocations from other NFT collections.

Otherwise be aware of this secondary exploit that can happen with a compromised UUID signer. The contract has the sufficient owner methods to repair this griefing attack if it were to take place.

Resolution

Animecoin Team: Acknowledged.

L-06 | Delayed Claim Warning

Category	Severity	Location	Status
Warning	● Low	AnimeClaimer.sol	Partially Resolved

Description

The maximum age for a read response to be executed allowed in the AnimeClaimer contract is 1 hour. This significantly limits the risk of stale read requests being executed far past the time at which ownership/delegation was verified at. However there is still some risk that should be communicated with users.

For example, consider the following scenario:
• Alice has Azuki #7 with 10 ANIME tokens vested on day 10 at 11:30pm
• 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 at 11:30pm, allowing Alice to claim 10 ANIME tokens
• On day 11 at 12:10am 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 at 12:20am
• Alice retries her lzReceive on Arbitrum by invoking the lzReceive function through the endpoint herself. 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 at 11:30pm
• 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

This finding serves merely to document this risk for users.

Resolution

Animecoin Team: Partially Resolved.

L-07 | Lacking Zero Configs Validation

Category	Severity	Location	Status
Validation	● Low	AnimeClaimer.sol	Resolved

Description

In the `requestClaim` function there is no validation that the configs array is a nonzero length.

Recommendation

To avoid any unexpected behavior, consider validating that the configs array is a nonzero length.

Resolution

Animecoin Team: Resolved.

L-08 | Anime Coin Trapped Ether

Category	Severity	Location	Status
Trapped Ether	● Low	Animecoin.sol: 53	Acknowledged

Description

The registerTokenOnL2 function accepts ether but does not explicitly use all of the msg.value sent nor refund additional msg.value to the caller. If excess ether is sent it will be trapped in the Animecoin contract .

Recommendation

Consider validating that exactly the valueForGateway and valueForRouter is sent in the registerTokenOnL2 or refund additional ether to the caller. Alternatively consider adding an owner function to withdraw ether from the contract.

Resolution

Animecoin Team: Acknowledged.

L-09 | Unused Contract

Category	Severity	Location	Status
Best Practices	● Low	AnimeClaimer.sol: 26	Acknowledged

Description

`AnimeClaimer` contract inherits from `OAppOptionsType3` but does not use anything from it.

Recommendation

Consider removing the `OAppOptionsType3` inherited contract.

Resolution

Animecoin Team: Acknowledged.

L-10 | Misleading Comment

Category	Severity	Location	Status
Best Practices	● Low	AnimeClaimer.sol: 138	Resolved

Description

The expectedCalldataSize in the AnimerClaimer contract is set to 128, although the size of the return data from ClaimChecker:checkClaims is 64 bytes.

The comment above the expectedCalldataSize storage variable states that "Any extra overallocated is returned to the sender anyways".

The calldata parameter is used inside the Executor to determine the amount to pay for the LZ message and if it is larger than the expected size, the extra gas is not returned to the sender.

Recommendation

The additional gas cost is negligible but consider changing the comment to reflect the actual behavior.

Resolution

Animecoin Team: Resolved.

L-11 | Merkle Tree Entries

Category	Severity	Location	Status
Validation	● Low	AnimeClaimer.sol	Acknowledged

Description

The AnimeClaimer contract stores vesting state using a hash key:

```
bytes32 hash = EfficientHashLib.hash(uint160(nft), tokenId, uint160(collector), streamId);
```

When creating multiple allocations in the Merkle tree for:
• Same NFT/tokenId pair, or
• Same collector address

Each allocation must use a unique streamId to prevent storage collisions in the vesting state.

Recommendation

When generating the Merkle tree, ensure unique streamId values for each allocation to the same recipient (NFT or collector).

Resolution

Animecoin Team: Acknowledged.

L-12 | Incompatible ClockMode

Category	Severity	Location	Status
Compatibility	● Low	Animecoin.sol	Acknowledged

Description

The Animecoin contract on Arbitrum is an ERC20Votes token, however uses the default clock mode of block.number. This means the voting system will rely on checkpoints based on L1 blocks rather than L2 blocks or timestamps.

This can lead to integration issues with the governance system that integrates with the ERC20Votes token.

Recommendation

The recommended clock mode for Arbitrum governance tokens is timestamp.

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