Unitags Resolver

Security Review

Cantina Managed review by:
Optimum, Lead Security Researcher
Akshay Srivastav, Associate Security Researcher

February 1, 2024

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Unitags-resolver is Uniswap's ENS offchain resolver implementation using CCIP-read (eip 3668) and ENSIP 10.

From Mar 27th to Apr 29th the Cantina team conducted a review of unitags-resolver on commit hash 802b477c. The team identified a total of **3** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 0
- Low Risk: 1
- Gas Optimizations: 1
- Informational: 1

3 Findings

3.1 Low Risk

3.1.1 `UrlSet` event is not emitted during the construction of the contract

Severity: Low Risk

Context: OffchainResolver.sol#L44

Description: `UrlSet` is defined inside the `UnitagsOffchainResolver` contract to reflect changes made to the `url` storage variable. However, in the current version of the code, the first assignment of value will not be recorded as part of an event.

Recommendation: Consider changing the constructor to call `_setUrl` instead.

Cantina Managed: Fixed in commit 08ae3333 by implementing the auditor's comment.

3.2 Gas Optimization

3.2.1 Code complexity can be reduced by replacing `EnumerableSet` with a mapping

Severity: Gas Optimization

Context: OffchainResolver.sol#L22

Description: `OffchainResolver` currently utilizes OpenZeppelin's `EnumerableSet.AddressSet` datatype to store the signer addresses. Enumerable sets are used in cases where enumeration is needed over the data entities. But as per the `OffchainResolver`'s implementation it never enumerates over the signer addresses.

Hence the use of `EnumerableSet.AddressSet` increases the complexity and gas cost of contract without providing any added functionality.

Recommendation: Consider using a mapping to store the signer addresses:

```
mapping (address => bool) private _signers;
```

Cantina Managed: Fixed in commit 08ae3333 by implementing the auditor's comment.

3.3 Informational

3.3.1 `ChainId` should be included in signature hash

Severity: Informational

Context: SignatureVerifierLib.sol#L25

Description: In case the OffchainResolver contract gets deployed on different EVM chains on same address, then a signer's signature created for one can be replayed on the other chain as well. This may result in unintended signatures getting successfully validated on a different chain than the intended one.

Recommendation: Consider including `CHAIN_ID` in signature hash generation.

```
function makeSignatureHash(address target, uint64 expires, bytes calldata request, bytes memory result)
    internal
    pure
    returns (bytes32)
{
    return keccak256(abi.encodePacked(_PREIMAGE_PREFIX, block.chainid, target, expires, keccak256(request),
↪   keccak256(result)));
}
```

Cantina Managed: Fixed in commit 08ae3333 by implementing the auditor's comment.