Omni Network

Nomina ERC20

Security Assessment Report

Version: 2.0

August, 2025

Contents

Introduction **2**
- Disclaimer . . . 2
- Document Structure . . . 2
- Overview . . . 2

Security Assessment Summary **3**
- Scope . . . 3
- Approach . . . 3
- Coverage Limitations . . . 3
- Findings Summary . . . 3

Detailed Findings **5**

Summary of Findings **6**
- Inconsistent Zero Address Handling In ERC20 Implementation . . . 7
- Missing Two-Step Mechanism For Updating Minting Authority . . . 8
- Non-Standard Naming Convention And Unnecessary Caching Of Immutable . . . 9

A Vulnerability Severity Classification **10**

Introduction

Sigma Prime was commercially engaged to perform a time-boxed security review of the Omni Network components in scope. The review focused solely on the security aspects of the Solidity implementation of the contract, though general recommendations and informational comments are also provided.

Disclaimer

Sigma Prime makes all effort but holds no responsibility for the findings of this security review. Sigma Prime does not provide any guarantees relating to the function of the components in scope. Sigma Prime makes no judgements on, or provides any security review, regarding the underlying business model or the individuals involved in the project.

Document Structure

The first section provides an overview of the functionality of the Omni Network components contained within the scope of the security review. A summary followed by a detailed review of the discovered vulnerabilities is then given which assigns each vulnerability a severity rating (see Vulnerability Severity Classification), an *open/closed/resolved* status and a recommendation. Additionally, findings which do not have direct security implications (but are potentially of interest) are marked as *informational*.

The appendix provides additional documentation, including the severity matrix used to classify vulnerabilities within the Omni Network components in scope.

Overview

Nomina is an ERC20 token that includes a conversion function between OMNI and Nomina tokens.

Security Assessment Summary

Scope

The review was conducted on the files hosted on the omni repository.

The scope of this time-boxed review was strictly limited to files at commit cb94096. The fixes of the identified issues were assessed at commit 745df3b.

`contracts/nomina/src/token/Nomina.sol`

Note: third party libraries and dependencies were excluded from the scope of this assessment.

Approach

The security assessment covered components written in Solidity.

For the Solidity components, the manual review focused on identifying issues associated with the business logic implementation of the contracts. This includes their internal interactions, intended functionality and correct implementation with respect to the underlying functionality of the Ethereum Virtual Machine (for example, verifying correct storage/memory layout).

Additionally, the manual review process focused on identifying vulnerabilities related to known Solidity anti-patterns and attack vectors, such as re-entrancy, front-running, integer overflow/underflow and correct visibility specifiers.

For a more detailed, but non-exhaustive list of examined vectors, see [1, 2].

To support the Solidity components of the review, the testing team may use the following automated testing tools:

- Aderyn: `https://github.com/Cyfrin/aderyn`
- Slither: `https://github.com/trailofbits/slither`
- Mythril: `https://github.com/ConsenSys/mythril`

Output for these automated tools is available upon request.

Coverage Limitations

Due to the time-boxed nature of this review, all documented vulnerabilities reflect best effort within the allotted, limited engagement time. As such, Sigma Prime recommends to further investigate areas of the code, and any related functionality, where majority of critical and high risk vulnerabilities were identified.

Findings Summary

The testing team identified a total of 3 issues during this assessment. Categorised by their severity:

- Informational: 3 issues.

Detailed Findings

This section provides a detailed description of the vulnerabilities identified within the Omni Network components in scope. Each vulnerability has a severity classification which is determined from the likelihood and impact of each issue by the matrix given in the Appendix: Vulnerability Severity Classification.

A number of additional properties of the components, including optimisations, are also described in this section and are labelled as “informational”.

Each vulnerability is also assigned a **status**:

- ***Open:*** the issue has not been addressed by the project team.
- ***Resolved:*** the issue was acknowledged by the project team and updates to the affected components(s) have been made to mitigate the related risk.
- ***Closed:*** the issue was acknowledged by the project team but no further actions have been taken.

Summary of Findings

ID	Description	Severity	Status
ONE-01	Inconsistent Zero Address Handling In ERC20 Implementation	Informational	**Closed**
ONE-02	Missing Two-Step Mechanism For Updating Minting Authority	Informational	**Resolved**
ONE-03	Non-Standard Naming Convention And Unnecessary Caching Of Immutable	Informational	**Resolved**

ONE-01	Inconsistent Zero Address Handling In ERC20 Implementation
Asset	`Nomina.sol`
Status	**Closed:** See Resolution
Rating	Informational

Description

The Solady implementation of ERC20 permits minting and transferring tokens to and from the zero address, as stated in the contract comments:

```
/// - The ERC20 standard allows minting and transferring to and from the zero address,
///   minting and transferring zero tokens, as well as self-approvals.
///   For performance, this implementation WILL NOT revert for such actions.
///   Please add any checks with overrides if desired.
```

However, within the protocol, a `_DEAD_ADDRESS` is defined with the following comment:

```
/**
 * @notice The address OMNI tokens are sent to on conversion as they cannot be sent to the zero address or burned.
 */
address private constant _DEAD_ADDRESS = address(0xdead);
```

Despite this, the implementation still permits minting or transferring tokens to the zero address, creating an inconsistency in whether the zero address is considered valid or if `_DEAD_ADDRESS` should always be used instead. There are no enforced constraints to prevent sending to the zero address.

```
function mint(address to, uint256 amount) public onlyMinter {
    _mint(to, amount);
}
```

Recommendations

To maintain clarity and enforce consistent logic, it is recommended to implement a constraint that disallows sending tokens to the zero address.

Resolution

The issue has been closed as invalid. A deliberate design decision was made to allow minting and transferring tokens to the zero address for performance reasons. The use of the `_DEAD_ADDRESS` is intended for specific cases where tokens cannot be sent to the zero address or burned, that is OMNI tokens which are based on OpenZeppelin's ERC20 implementation.

ONE-02	Missing Two-Step Mechanism For Updating Minting Authority
Asset	`Nomina.sol`
Status	**Resolved:** See Resolution
Rating	Informational

Description

The minter authority can be updated directly in a single transaction. As only the current minter authority is permitted to update this address, any mistake, such as setting it to an incorrect or inaccessible address, would render the minting functionality problematic.

```
function setMintAuthority(address _mintAuthority) public onlyMintAuthority {
    mintAuthority = _mintAuthority;
    emit MintAuthoritySet(_mintAuthority);
}
```

Recommendations

It is recommended to implement a two-step process for updating the minter authority address. This typically involves the current authority proposing a new address, which the new address must then accept. This helps prevent accidental or malicious misconfiguration.

Resolution

The issue has been resolved in commit 745df3b by implementing a two-step process for updating the minting authority.

ONE-03	Non-Standard Naming Convention And Unnecessary Caching Of Immutable
Asset	`Nomina.sol`
Status	**Resolved:** See Resolution
Rating	Informational

Description

The contract contains two code quality issues related to the `omni` immutable variable that deviate from standard Solidity conventions and best practices.

First, the immutable variable uses lowercase naming instead of the standard convention of capitalised constants and immutables:

```
address public immutable omni; //@audit Should be OMNI per naming conventions
```

Second, the `convert` function unnecessarily caches the immutable variable to a stack variable:

```
function convert(address to, uint256 amount) public {
    address _omni = omni; //@audit Unnecessary caching for immutable
    if (amount == 0) return;
    if (to == address(0)) revert ZeroAddress();
    if (_omni == address(0)) revert ConversionDisabled();

    _omni.safeTransferFrom(msg.sender, _DEAD_ADDRESS, amount);
    _mint(to, amount * CONVERSION_RATE);
}
```

Since immutable variables are embedded directly into the contract bytecode at deployment, accessing them does not incur storage read costs and therefore caching provides no gas benefit. This pattern is typically used for storage variables where each read costs gas.

Recommendations

Consider renaming the immutable variable to follow standard conventions and remove unnecessary caching:

```
address public immutable OMNI;

function convert(address to, uint256 amount) public {
    if (amount == 0) return;
    if (to == address(0)) revert ZeroAddress();
    if (OMNI == address(0)) revert ConversionDisabled();

    OMNI.safeTransferFrom(msg.sender, _DEAD_ADDRESS, amount);
    _mint(to, amount * CONVERSION_RATE);
}
```

Resolution

The recommendation has been implemented in commit 745df3b.

Appendix A Vulnerability Severity Classification

This security review classifies vulnerabilities based on their potential impact and likelihood of occurance. The total severity of a vulnerability is derived from these two metrics based on the following matrix.

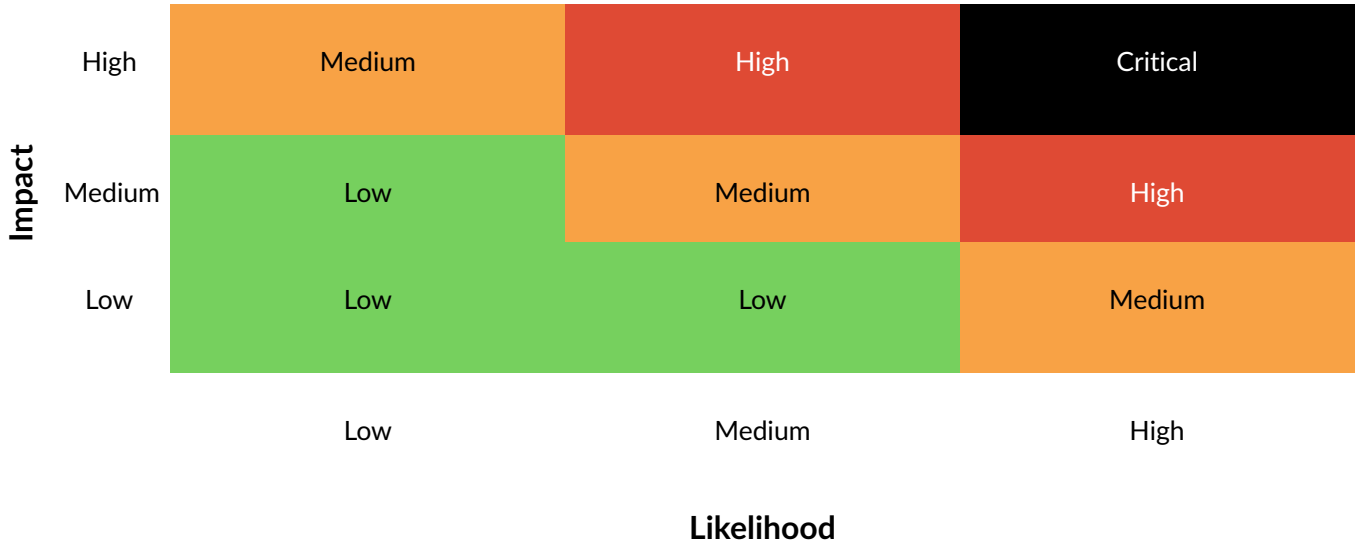

Table 1: Severity Matrix - How the severity of a vulnerability is given based on the *impact* and the *likelihood* of a vulnerability.

References

[1] Sigma Prime. Solidity Security. Blog, 2018, Available: https://blog.sigmaprime.io/solidity-security.html. [Accessed 2018].

[2] NCC Group. DASP - Top 10. Website, 2018, Available: http://www.dasp.co/. [Accessed 2018].