OMNI

Token Upgrade Frontend

Security Assessment Report

Version: 2.0

September, 2025

Contents

Introduction

Sigma Prime was commercially engaged to perform a time-boxed security review of the Omni components in scope.

The review focused solely on the security aspects of these components, 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 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 components in scope.

Overview

The `OMNI` token will be converted to `NOM` tokens (known as Nomina tokens) to support Omni's upcoming rebranding. This review looks at the web frontend, that allows for the conversion between `OMNI` and `NOM`. It will burn a user's `OMNI` tokens and issue them with `NOM` tokens at a ratio of 1:75.

Security Assessment Summary

Scope

The review was conducted on the files hosted on the Omni Nomina Front End repository.

The scope of this time-boxed review was strictly limited to files at commit 601147c.

The fixes of the identified issues were assessed at commit 7ddec7e.

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

Approach

The security assessment covered components written in TypeScript, utilising the React framework.

The manual review focused on identifying issues associated with the business logic implementation of the frontend interface. This includes component interactions, intended functionality and correct implementation.

Additionally, the manual review process focused on identifying vulnerabilities related to known web frontend anti-patterns and attack vectors, including security misconfigurations, unsafe dependencies, and client-side vulnerabilities.

The testing team may use the following automated testing tools:

- FFUF: `https://github.com/ffuf/ffuf`
- BURP: `https://portswigger.net/burp`

Output for FFUF 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 7 issues during this assessment. Categorised by their severity:

- Medium: 1 issue.
- Low: 2 issues.
- Informational: 4 issues.

Detailed Findings

This section provides a detailed description of the vulnerabilities identified within the Omni 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
OTU-01	Incorrect Exchange Rate Quote	Medium	Resolved
OTU-02	Inline Styles Allowed	Low	Closed
OTU-03	Exact React Versions Not Pinned	Low	Resolved
OTU-04	HTTP Response Header Considerations	Informational	Resolved
OTU-05	Syntax Error In Permissions Policy	Informational	Resolved
OTU-06	Hardcoded Test Chain ID	Informational	Resolved
OTU-07	Miscellaneous General Comments	Informational	Resolved

OTU-01	Incorrect Exchange Rate Quote		
Asset	`modal.tsx`		
Status	**Resolved:** See Resolution		
Rating	Severity: Medium	Impact: Low	Likelihood: High

Description

The conversion interface displays an incorrect exchange rate that could confuse users and undermine confidence in the protocol.

There is a token conversion rate displayed at the bottom of the conversion interface:

```
<div className="bg-white/4 rounded-lg border border-border/4 p-4">
  <div className="flex items-center gap-2">
    <span className="text-white text-sm">75 OMNI = 1 NOM</span>
```

This quotes `75 OMNI = 1 NOM`, but the correct value is `75 NOM = 1 OMNI`.

Given the prominent placement of this notice, it could cause uncertainty and doubt for users, and a possible loss of confidence in the protocol.

The likelihood has been rated high as clearly all users would see this notice, and the impact is above informational at low because of the reputational damage.

Recommendations

Invert the exchange rate from `75 OMNI = 1 NOM` to `1 OMNI = 75 NOM`. This will show the actual value users would get in their conversion transactions.

Resolution

The exchange rate was updated.

OTU-02	Inline Styles Allowed		
Asset	`entry.server.tsx`		
Status	**Closed:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Low

Description

There is a possibility for inline styles which, as noted in the developer comments, is better disallowed.

```
'script-src': ["'self'", `'nonce-${nonce}'`, "'strict-dynamic'"],
// TODO: remove 'unsafe-inline' - waiting for this rainbow PR to be merged
// https://github.com/rainbow-me/rainbowkit/pull/2501
'style-src': ["'self'", "'unsafe-inline'", 'https://fonts.googleapis.com'],
'style-src-elem': ["'self'", "'unsafe-inline'", 'https://fonts.googleapis.com'],
// TODO reinstate this when rainbow PR is merged
// ...(isDev
// ? { 'style-src-elem': ["'self'", "'unsafe-inline'", 'https://fonts.googleapis.com'] }
// : {}),
```

Recommendations

The development team already has a plan in place to resolve this issue when a dependant pull request is resolved.

Resolution

As described in the comments, the `unsafe-inline` entry will be removed at a future date.

OTU-03	Exact React Versions Not Pinned		
Asset	`package.json`		
Status	**Resolved:** See Resolution		
Rating	Severity: Low	Impact: Low	Likelihood: Low

Description

Using unpinned React versions exposes the application to potential supply chain attacks and unexpected dependency updates that could introduce vulnerabilities.

The versions of `react`, `react-dom` and `react-router` allow automatic minor updates:

```
"react": "^18.3.1",
"react-dom": "^18.3.1",
"react-router": "^7.7.1",
```

Recommendations

Remove the `^` prefixes to lock in specific exact versions.

Security Benefits:

- **Supply chain attack prevention** - Eliminates risk of malicious updates being auto-installed
- **Dependency confusion mitigation** - Prevents attackers from publishing higher versions of packages to hijack installs
- **Reproducible builds** - Ensures identical dependency trees across environments
- **Vulnerability control** - Explicitly choose when to update, allowing security review first

Resolution

The `^` prefixes were removed.

OTU-04	HTTP Response Header Considerations
Asset	`entry.server.tsx`
Status	**Resolved:** See Resolution
Rating	Informational

Description

Several HTTP response header configurations could be optimised for enhanced security, though current settings provide adequate protection.

Whilst the response headers are generally securely configured, there are a few points to consider:

Add X-Frame-Options: DENY

This provides defence-in-depth against clickjacking, but is optional given that modern browser support for CSP (which defends against clickjacking) is now very high.

Cross-Origin-Embedder-Policy : require-corp

If users report problems with wallet integrations, changing this setting to `credentialless` may help, although `credentialless` may have some issues with browser support.

Recommendations

Monitor user reports of wallet integration problems and consider modifying the `Cross-Origin-Embedder-Policy` if issues arise.

Resolution

The `X-Frame-Options: DENY` header was added.

OTU-05	Syntax Error In Permissions Policy
Asset	`entry.server.tsx`
Status	**Resolved:** See Resolution
Rating	Informational

Description

There is a missing comma in the `Permissions-Policy` header.

```
responseHeaders.set(
  'Permissions-Policy',
  'geolocation=(), camera=(), microphone=() clipboard-write=(self)',
)
```

There should be a comma after `microphone=()`.

This omission will only affect some browsers, but it is best practice to use the syntax as in the specification unless otherwise advised.

Recommendations

Add the relevant comma.

Resolution

The relevant comma was added.

OTU-06	Hardcoded Test Chain ID
Asset	`modal.tsx` `use-approve.ts` `use-balances.ts` `use-convert.ts`
Status	**Resolved:** See Resolution
Rating	Informational

Description

Hardcoded chain IDs increase deployment risk and maintenance overhead, as each instance must be manually updated when deploying to different networks.

In many locations in the code, the Holesky chain ID is hard coded instead of a generalised chain ID configuration variable. This makes deployment more risky, as all instances of the hardcoded chain ID will need to be manually modified on deployment.

```
const approve = async () => {
  if (!address || !amount) return
  approvedRef.current = amount
  writeContract({
    address: omni,
    abi: erc20Abi,
    functionName: 'approve',
    chainId: holesky.id,
    args: [nom, parseUnits(amount, 18)],
  })
}
```

Recommendations

Replace the hardcoded chain Ids for Holesky with a configuration variable, allowing the chain to be modified in one location.

Resolution

The recommended change was implemented, replacing the Holesky Id with `chain[env].id`.

OTU-07	Miscellaneous General Comments
Asset	All files
Status	**Resolved:** See Resolution
Rating	Informational

Description

This section details miscellaneous findings discovered by the testing team that do not have direct security implications:

1. **Conversion Errors Provide No Information To Users**

   Related Asset(s): modal.tsx

   If there is an error for any reason during the submission of a conversion transaction, the modal component simply prints a retry button.

   ```
   // error
   if (isConvertError || isBatchConvertError)
     return {
       disabled: false,
       label: 'Retry',
       onClick: () => convert(),
     }
   ```

   Both `useConvert` and `useBatchConvert` hooks use the `wagmi` module's `useWriteContract`, `useWaitForTransactionReceipt`, and `useSendCalls`, which return objects containing human readable error descriptions and boolean values classifying the error type. It would therefore be possible to provide users with more guidance as to the cause of their transaction failure, which would result in a better user experience and increased confidence in the protocol.

   Currently, the modal only accesses the boolean `isError` flags but ignores the rich error data.

   Print human readable error messages during convert transactions to improve user experience.

Recommendations

Ensure that the comments are understood and acknowledged, and consider implementing the suggestions above.

Resolution

Updates to the code implemented significantly more informative messages in error cases.

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.

Impact	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Low	Low	Medium
		Low	Medium	High
			Likelihood	

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