

**Finding Name: Prototype Pollution.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Team** | **Role** | **Project** | **Quality Assurance** | **Is this a re-tested Finding?** |
| Jaspriya Kaur | SCR | Senior Lead | Ontrack |  |  |
|  |  |  |  |  |  |

|  |
| --- |
| **Was this finding successful?** |
| Yes |

**Finding Description**

The Minimist library, versions prior to 1.2.6 and 0.2.4, contain a critical vulnerability due to improper handling of user-supplied input. The issue arises from a lack of proper validation and sanitization of input passed to the setKey() function located in the index.js file (lines 69-95). This function is susceptible to Prototype Pollution attacks, allowing an attacker to manipulate the prototype of objects and introduce or modify properties in unexpected ways.

**Risk Rating**

Impact: Major

Likelihood: High

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Impact values** | | | | |
| **Very Minor** | **Minor** | **Significant** | **Major** | **Severe** |
| Risk that holds little to no impact. Will not cause damage and regular activity can continue. | Risk that holds minor form of impact, but not significant enough to be of threat. Can cause some damage but not enough to impede regular activity. | Risk that holds enough impact to be somewhat of a threat. Will cause damage that can impede regular activity but will be able to run normally. | Risk that holds major impact to be of threat. Will cause damage that will impede regular activity and will not be able to run normally. | Risk that holds severe impact and is a threat. Will cause critical damage that can cease activity to be run. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Likelihood** | | | | |
| **Rare** | **Unlikely** | **Moderate** | **High** | **Certain** |
| Event may occur and/or if it did, it happens in specific circumstances. | Event could occur occasionally and/or could happen (at some point) | Event may occur and/or happens. | Event occurs at times and/or probably happens a lot. | Event is occurring now and/or happens frequently. |

**Business Impact**

1. Remote Code Execution (RCE): An attacker could exploit this vulnerability to execute arbitrary code within the context of the application, leading to complete compromise of the system.
2. Data Tampering: By manipulating object prototypes, an attacker could modify critical data structures within the application, leading to data corruption, integrity violations, or unauthorized access to sensitive information.
3. Denial of Service (DoS): The ability to inject unexpected properties into objects could lead to resource exhaustion, resulting in a denial of service condition, rendering the application unavailable to legitimate users.

**Affected Assets**

1. Data assets processed, stored, or transmitted by systems using the vulnerable Minimist library are at risk.
2. Infrastructure components, including servers, databases, and cloud services, relying on the vulnerable library may be vulnerable.
3. Applications developed with versions of Minimist prior to 1.2.6 and 0.2.4 are susceptible to the identified vulnerability.
4. Services provided by affected applications or systems could experience disruptions or unauthorized access.
5. Client endpoints accessing vulnerable services or applications may be indirectly impacted.
6. Networks hosting vulnerable systems could facilitate lateral movement or unauthorized access.
7. Critical business processes relying on data integrity or availability may face disruptions or compromise.

**Evidence**

**Step 1: Lines : 69-95**



**Path: workspace/package-lock.json**

**Remediation Advice**

* Upgrade: Update the Minimist library to version 1.2.6 or 0.2.4, which contain the necessary patches to address the vulnerability.
* Monitor: Regularly monitor for security advisories and updates from the Minimist project and other dependencies. Stay informed about potential security risks and apply patches promptly.
* Input Validation: Implement strict input validation and sanitization mechanisms to prevent malicious input from reaching vulnerable functions or components.
* Defense in Depth: Employ additional security measures such as input/output filtering, least privilege principles, and code reviews to bolster the overall security posture of the application.

**References**

https://github.com/minimistjs/minimist/issues/11

**Contact Details**

Jaspriya Kaur

S222196102@deakin.edu.au

**Pen test Leader Feedback.**

The lead will provide feedback to enact on.