

**Finding Name:** Permissive Cross-Origin Request Handling

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Team** | **Role** | **Project** | **Quality Assurance** | **Is this a re-tested Finding?** |
| Deakin Carr | SCR | Junior Team Member | Ontrack |  |  |
|  |  |  |  |  |  |

|  |
| --- |
| **Was this Finding Successful?** |
| Yes |

**Finding Description**

The Ontrack application's api\_root.rb file configures Cross-Origin Resource Sharing (CORS) to allow requests from any origin, as indicated by the header setting Access-Control-Allow-Origin with a value of '\*'. This overly permissive CORS policy enables web browsers to make requests to the Ontrack API from any domain. While CORS is designed to allow or restrict resources based on the domain of the web pages that are requesting them, setting it to '\*' removes these restrictions, potentially exposing the application to cross-site scripting (XSS) attacks or other cross-origin threats.

**Risk Rating**  
Impact: **Major**  
 Given the nature of the Ontrack application, which deals with sensitive student data and academic records, a vulnerability that could lead to unauthorized access or data manipulation represents a major threat. The impact is such that it could impede regular activity and disrupt the normal operation of the application, affecting both the integrity of academic assessments and the privacy of students.

Likelihood: **Moderate**  
 Considering the application is web-based and potentially accessible by a large number of users, the likelihood of exploiting a permissive CORS policy is moderate. While not certain, there's a reasonable chance that a motivated attacker could identify and exploit this vulnerability, especially if the application's presence and usage are well-known within the university or among external entities.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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**

The primary business impact of this vulnerability is the increased risk of cross-origin attacks, including XSS, where malicious scripts are executed on behalf of the user. This could lead to unauthorized access to sensitive information, session hijacking, or data theft. For an educational tool like Ontrack, which handles student feedback and assessment marking, such vulnerabilities could compromise student privacy, lead to unauthorized access to academic records, and potentially allow manipulation of sensitive data.

**Affected Assets**

The affected asset is the Ontrack application server, particularly the API endpoint configurations within the doubtfire-api/app/api/api\_root.rb script. This vulnerability may affect any data transactions between the AngularJS front end and the Ruby-based back end, where cross-origin requests are involved.

**Evidence**

The following line of code in the api\_root.rb file demonstrates the insecure configuration:

header['Access-Control-Allow-Origin'] = '\*'

This line explicitly sets the CORS policy to allow requests from any origin, illustrating the lack of restriction on cross-origin requests.



**Remediation Advice**

To mitigate this vulnerability, the CORS policy should be configured to only allow requests from trusted origins. This can be done by replacing the wildcard character ('\*') with a list of specific, trusted domains from which cross-origin requests are expected and permitted. For example:

header['Access-Control-Allow-Origin'] = '<https://trusteddomain.com>'

Additionally, it's advisable to review and implement other CORS-related headers, such as Access-Control-Allow-Methods and Access-Control-Allow-Headers, to further tighten cross-origin request handling. Implementing a more restrictive CORS policy will help prevent unauthorized cross-origin access and reduce the risk of cross-site attacks.

**References**

* Mozilla Developer Network, "Cross-Origin Resource Sharing (CORS)," Mozilla Foundation, 2023. [Online]. Available: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>. [Accessed: April 3, 2024].
* OWASP, "Cross-Site Scripting (XSS)," OWASP Foundation, 2023. [Online]. Available: <https://owasp.org/www-community/attacks/xss/>. [Accessed: April 3, 2024].
* ChatGPT, “ChatGPT”, OpenAI [Large language model] Available: [https://chat.openai.com](https://chat.openai.com/) [Accessed: 2/4/2024].

**Contact Details**

Deakin Carr

Carrde@deakin.edu.au

**Pentest Leader Feedback.**

The lead will provide feedback to enact on.