

**Finding Name:** SQL Injection

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

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

**Finding Description**

In an SQL injection attack, the user can submit an SQL query directly to the database, gaining access without providing appropriate credentials. The code snippet in this finding allows unsanitised input from the joins[] parameter into the SQL query, which attackers can manipulate to execute arbitrary SQL commands

**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**

This vulnerability could put the entire application and the organisation at a high risk. It can lead to unauthorised access to private information, make the clients lose trust in the company, cause financial loss that results from affected data, harm the company's reputation and possibly lead to legal action.

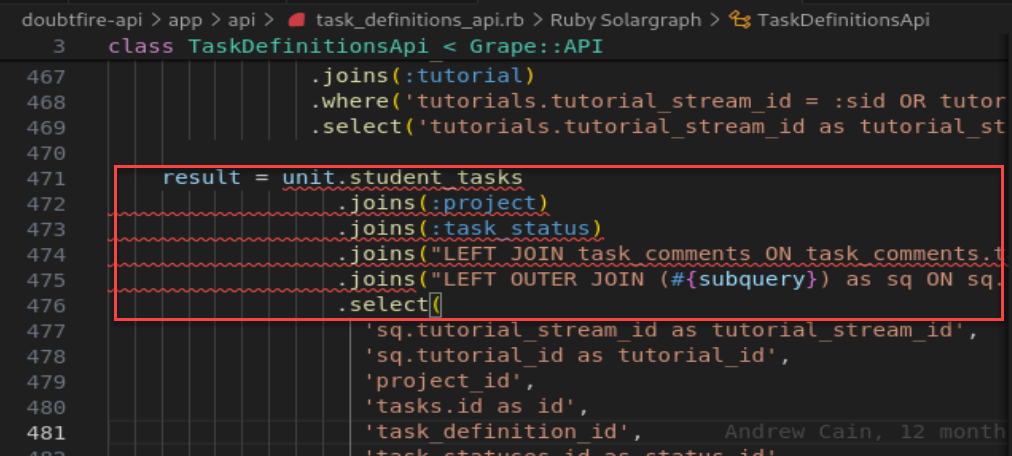
**Affected Assets**

Database containing sensitive information

User data and credentials stored in the database

**Evidence**

The vulnerable code is located in doubtfire-api/app/api/task\_definitions\_api.rb



This snippet receives unsanitised input from the following lines into “joins[]”

A screen shot of a computer program

Description automatically generated

**Remediation Advice**

Rewrite the code so that it uses parameterised queries instead of writing SQL queries using string concatenation. This ensures user input is seen as just data and not executable code.

Validate input for only expected characters and patterns into the joins[] parameter. Avoid using special characters that may be used in SQL injection attacks.

**References**

SQL injection (SQLi). (n.d.). Retrieved April 10, 2024, from Snyk.io: https://learn.snyk.io/lesson/sql-injection/?loc=ide

SQL Injection Prevention. (n.d.). Retrieved April 10, 2024, from https://cheatsheetseries.owasp.org/: https://cheatsheetseries.owasp.org/cheatsheets/SQL\_Injection\_Prevention\_Cheat\_Sheet.html

What is SQL Injection (SQLi) and How to Prevent It. (n.d.). Retrieved April 10, 2024, from www.acunetix.com/: https://www.acunetix.com/websitesecurity/sql-injection/

**Contact Details**

Natalia Khobotova

s223011356@deakin.edu.au

**Pentest Leader Feedback.**

Great work.