

**Finding Name: Malicious PDF Upload**

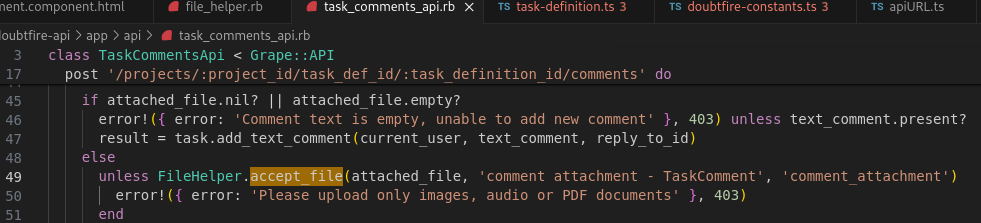
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Team** | **Role** | **Project** | **Quality Assurance** | **Is this a re-tested Finding?** |
| Pushkar Goel | SCR | SCR Team Member | Ontrack | Jaspriya Kaur and Payas Paul |  |
|  |  |  |  |  |  |

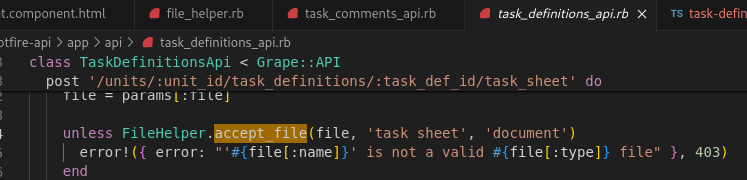
|  |
| --- |
| **Was this Finding Successful?** |
| Yes |

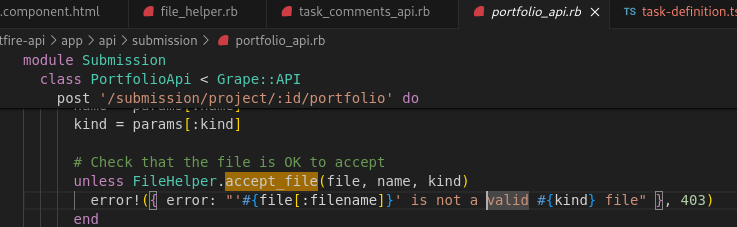
**Finding Description**

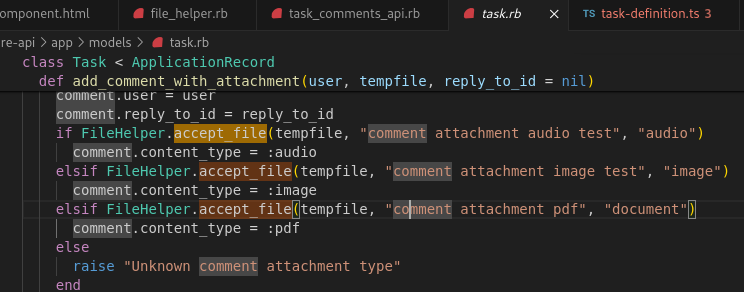
The PDF files being uploaded to the OnTrack portal are not being checked to verify that there is no malicious script inside it. This leads to the possibility of attacker uploading PDFs with malicious javascript scripts inside it to the portal.

This happens due to the fact that the method **file\_helper.accept\_file** just validates the extensions of uploaded file and not the content of the pdf. This method is being used in **task\_comments\_api**, **task\_definitions\_api**, **portfolio\_api** and **task.rb** and thus, allows the same vulnerability across all these modules.

Path: doubtfire-api/app/api/task\_comments\_api.rb

Path: doubtfire-api/app/api/task\_definitions\_api.rb

Path: doubtfire-api/app/api/submission/portfolio\_api.rb

Path: doubtfire-api/app/models/task.rb

This finding poses a critical security risk as it enables attackers to execute arbitrary code within the application's environment, potentially leading to severe consequences such as remote code execution, data theft, and compromise of sensitive information.

**Risk Rating**  
Impact: Major  
Likelihood: Moderate

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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 vulnerability identified in OnTrack’s file upload functionality poses a significant security risk to the system and its stakeholders, including students and tutors (convenors). The potential consequences are severe, ranging from remote code execution to data theft and compromise of sensitive information:

**1. Remote Code Execution (RCE)**:

Attackers can execute arbitrary commands within the application’s environment, granting them unauthorized access and control over OnTrack resources. This allows manipulation, exploitation, or compromise of the system for malicious purposes.

**2. Data Theft**:

Crafty attackers can embed advanced JavaScript code into PDFs. When accessed by users, this code intercepts and exfiltrates sensitive information stored on the OnTrack server. Unauthorized disclosure of confidential data becomes a real risk.

**3. Document or File Manipulation**:

Malicious JavaScript code within the OnTrack system can alter or delete documents (such as tasks, submissions, or resource files). Adversaries inject harmful content into files or modify file metadata, compromising data integrity and reliability.

**4. Cross-Site Scripting (XSS)**:

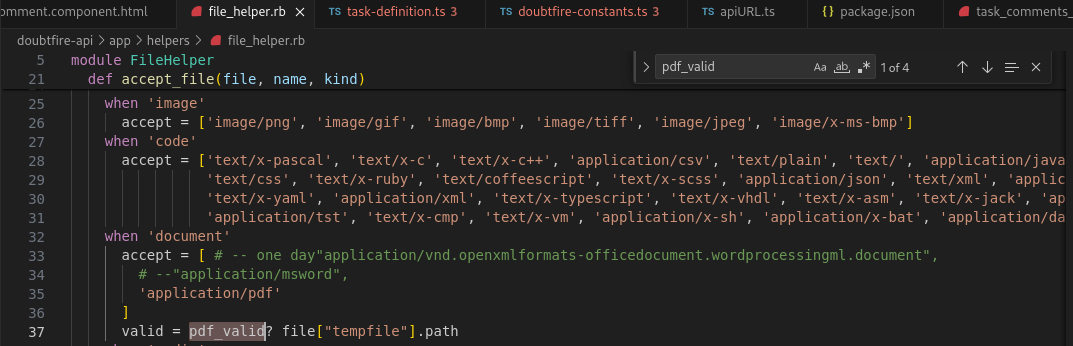
Unlike traditional XSS, embedded JavaScript payloads within PDF files persist in the application database. Upon user access, they trigger cross-site scripting attacks. Attackers can hijack user sessions, steal cookies, and perform unauthorized actions on behalf of authenticated users.

**Affected Assets**

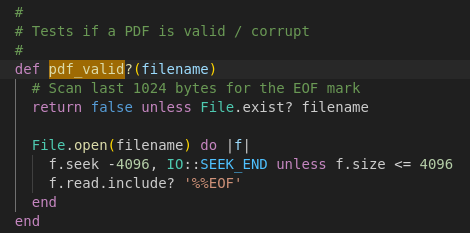
* User credentials (e.g., usernames, passwords) or profile information (e.g., email addresses, contact details)
* Uploaded files (e.g., images, documents, PDFs) from the users
* Session tokens, cookies
* OnTrack’s codebase (e.g., source code files, scripts)

**Evidence**

The accept\_file method checks the extension of the document to ascertain its a pdf, and discards other file formats.

Path: doubtfire-api/app/helpers/file\_helper.rb

However, to check the validity of the pdf, it calls the function pdf\_valid which only checks the size of the file, and does not verify that the content is safe.



This improper validation gives rise to the vulnerability allowing for Malicious file upload

**Remediation Advice**

* The PDF content validation should be enhanced to ensure that any malicious urls/scripts inside the code are sanitized or the file is rejected
* The Ontrackis only supposed to accept PDFs for assignment, and therefore do not need support JS scripts in the PDF, and therefore the feature should be disabled to prevent this vulnerability. It can be done using a library like **pdfnet-node.** The code for that would be like:

const { PDFNet } = require('pdfnet-node');

async function removeJavaScript(inputPath, outputPath) {

const doc = await PDFNet.PDFDoc.createFromFilePath(inputPath);

doc.initSecurityHandler();

doc.lock();

const jsActions = await doc.getJavaScriptActions();

jsActions.forEach((jsAction) => {

doc.removeJavaScript(jsAction);

});

**References**

Stokes, P 2019, *Malicious PDFs | Revealing the Techniques Behind the Attacks*, SentinelOne.

Pdfnet-node : https://www.npmjs.com/package/@pdftron/pdfnet-node

**Contact Details**

Pushkar Goel – s222162584@deakin.edu.au

**Secure Code Review Leader Feedback.**

The lead will provide feedback to enact on.