

**File Uploads (with malicious executable code)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name** | **Team** | **Role** | **Project** | **Quality Assurance** | **Is this a re-tested Finding?** |
| Nabiha Masood | PT | PT Team Lead | OnTrack | Roocha Thakkar | Yes |

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

**Finding Description**

OnTrack’s file upload functionality on convenor and student inbox/chat lacks proper validation and filtering mechanisms for screening the uploaded files. Attackers can upload PDF files with embedded JavaScript code because of this security flaw. 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:** 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 a minor form of impact but is not significant enough to be of threat. Can cause some damage but not enough to impede regular activity. | The 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. | The risk that holds a major impact is 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** |
| An 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 happen. | Event occurs at times and/or probably happens a lot. | The event is occurring now and/or happens frequently. |

**Business Impact**

The vulnerability discovered in the OnTrack’s file upload functionality poses a significant security risk to the system and its stakeholders including students and tutors (convenors), with potentially devastating business impacts and severe consequences such as remote code execution, data theft, and compromise of sensitive information.

* **Remote Code Execution (RCE):** Attackers can execute arbitrary commands within the application's environment that can grants attackers unauthorized access and control over the OnTrack resources, allowing them to manipulate, exploit, or compromise the system for malicious purposes.
* **Data Theft:** An attacker with an aim to stealing valuable data assets can craft an advances JavaScript code to embed into a PDF that can be used to intercept and exfiltrate sensitive information stored within the OnTrack’s server or is being accessed by users, leading to unauthorized disclosure of confidential data.
* **Document or File Manipulation:** A JavaScript code within to manipulate documents and files stored within the OnTrack system can allow the adversaries to alter or delete documents such as tasks, submission or resource files by injecting malicious content into files, or modifying file metadata to conceal unauthorized changes, thereby compromising the integrity and reliability of the stored data.
* **Cross-Site Scripting (XSS):** Unlike traditional XSS, the embedded JavaScript code as the payloads within the PDF files can persist within the application database, which is ready to execute upon access by any user. It can trigger cross-site scripting attacks, enabling attackers to hijack user sessions, steal cookies, and perform unauthorized actions on behalf of authenticated users.

**Affected Assets**

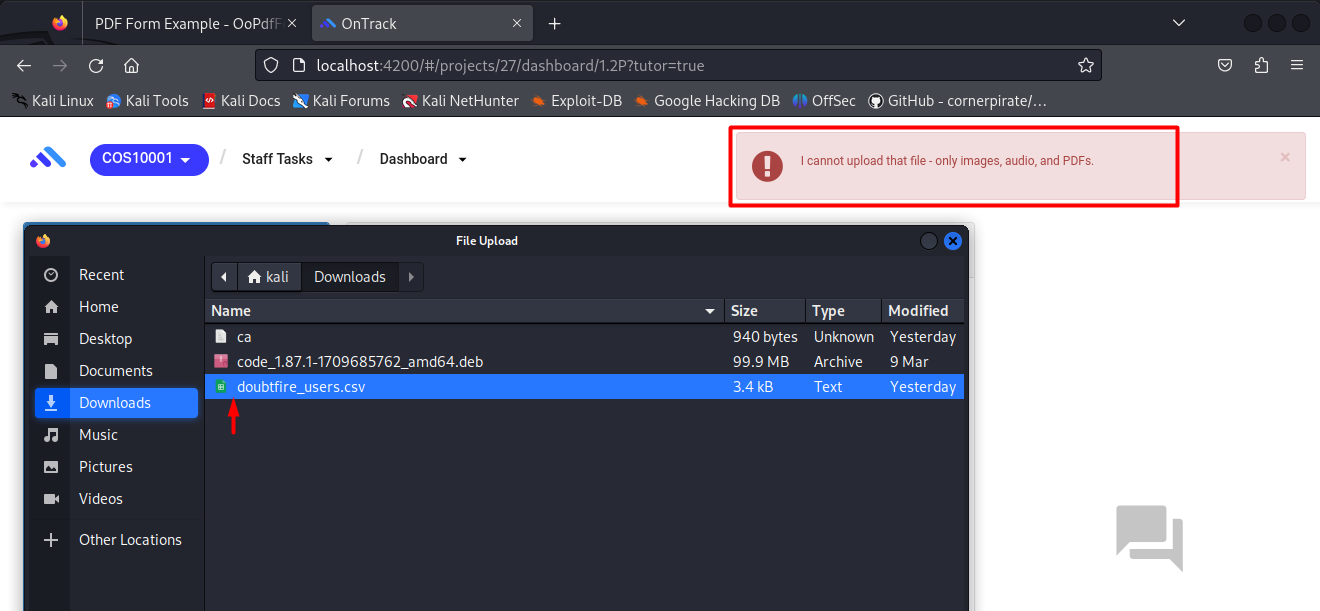
Using this vulnerability, the adversaries can exploit the OnTrack’s web application, and the following assets can be compromised by the attackers:

* 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**

**Step 0: Checking Acceptable File Types**

Upon attempting to upload a random file including a CSV file into the inbox, I discovered that the OnTrack system has implemented whitelisting measures for file uploads. Specifically, only image, audio, and PDF files are deemed acceptable. This limitation presents an opportunity to explore potential exploits within these permitted file formats to probe for vulnerabilities within the system.

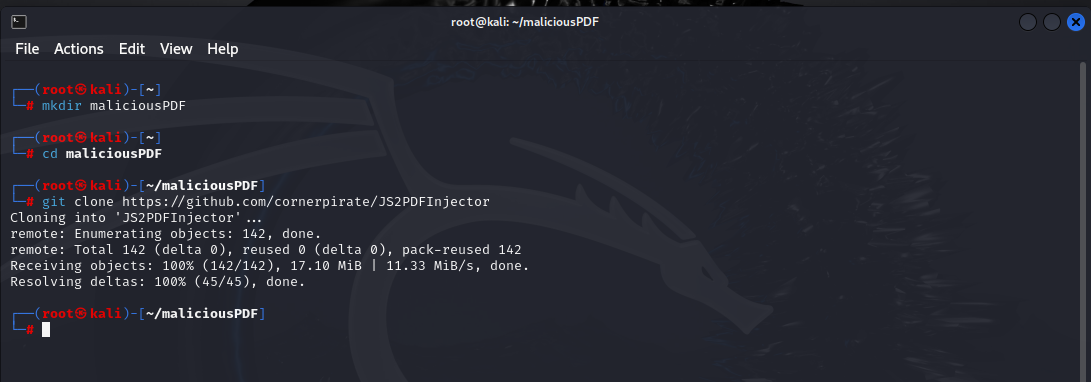


**Step 1: Crafting Malicious PDF**

Download or take any PDF file. I obtained a sample PDF file from <http://foersom.com/net/HowTo/data/OoPdfFormExample.pdf> and saved it as "submission.pdf".

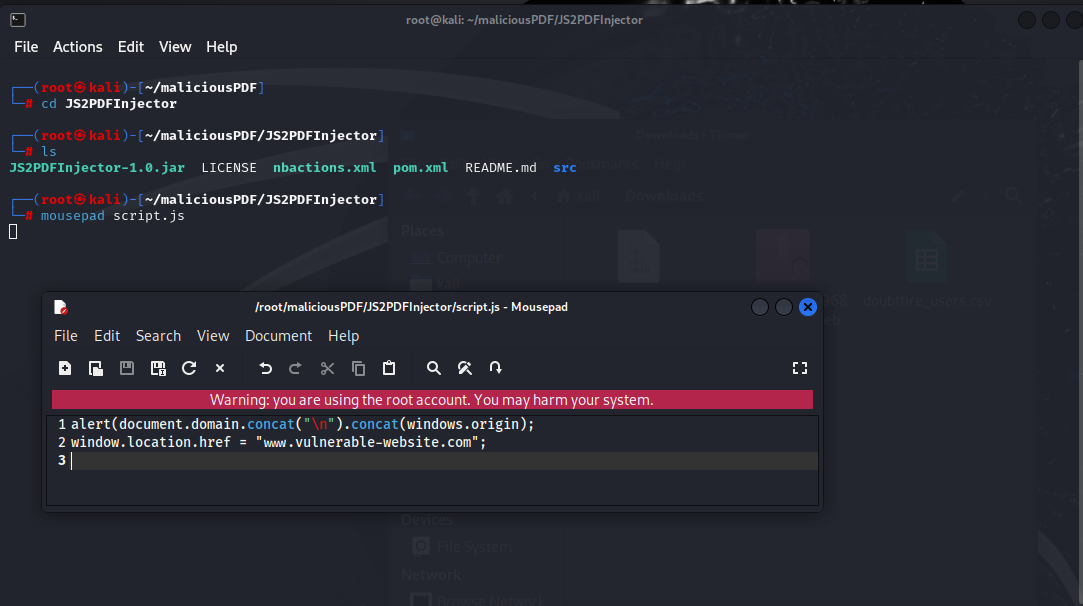
**Step 2. Download *JS2PDFInjector* Tool**

I created a new directory, titled *'maliciousPDF'*, to organize the resources where I cloned the tool *JS2PDFInjector* from the GitHub repository located at <https://github.com/cornerpirate/JS2PDFInjector> . It helps with the injection of JavaScript code into PDF files. Upon execution, the process generates a subfolder named as *'JS2PDFInjector'* within the designated directory i.e., *maliciousPDF*.

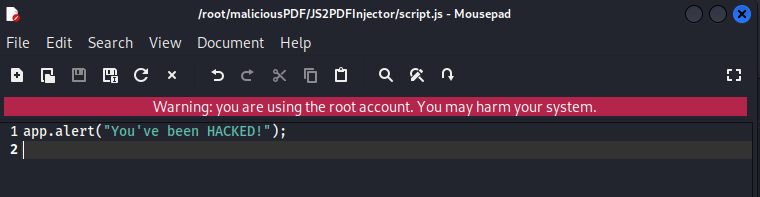


**Step 3. Create a JavaScript file containing malicious code**

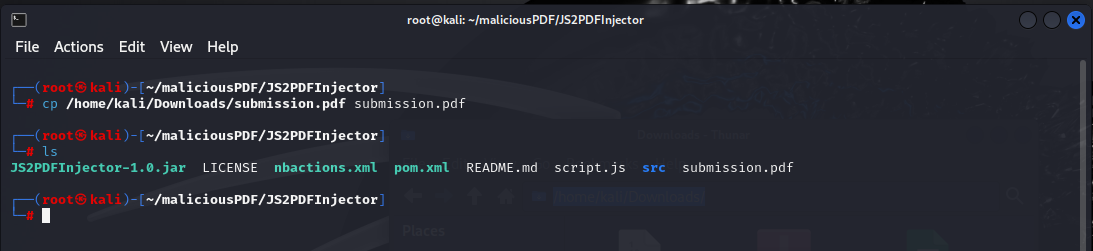
Within ‘JS2PDFInjector’ directory, I proceeded to create a new file named *"script.js"* where I inserted a simple JS code.



A sneak peek of the script here:



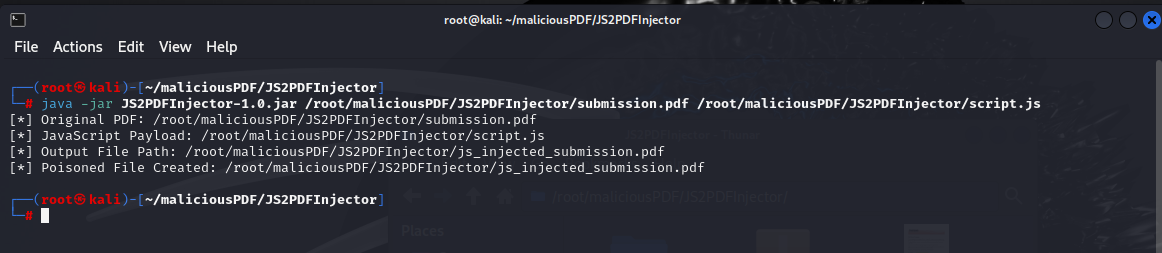
Now, copy our target PDF file i.e., *submission.pdf into* the *JS2PDFInjector* folder to ensure that both the PDF and JavaScript file are in the same directory.



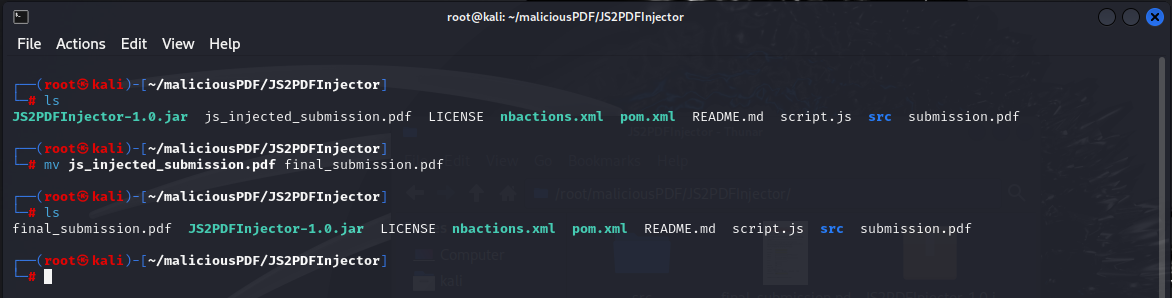
**Step 4: Embedding the malicious code into PDF**

The command *java -jar JS2PDFInjector-1.0.jar <Path to target PDF file> <Path to JavaScript file>* helps injection of the JS into the PDF file, where:

* ***java:*** Starts the Java Virtual Machine (JVM), which makes it possible to run Java program.
* ***-jar:*** Tells the JVM to run a Java program that's been compressed into a JAR file.
* ***JS2PDFInjector-1.0.jar:*** This JAR file contains the JS2PDFInjector tool that performs the injection process.

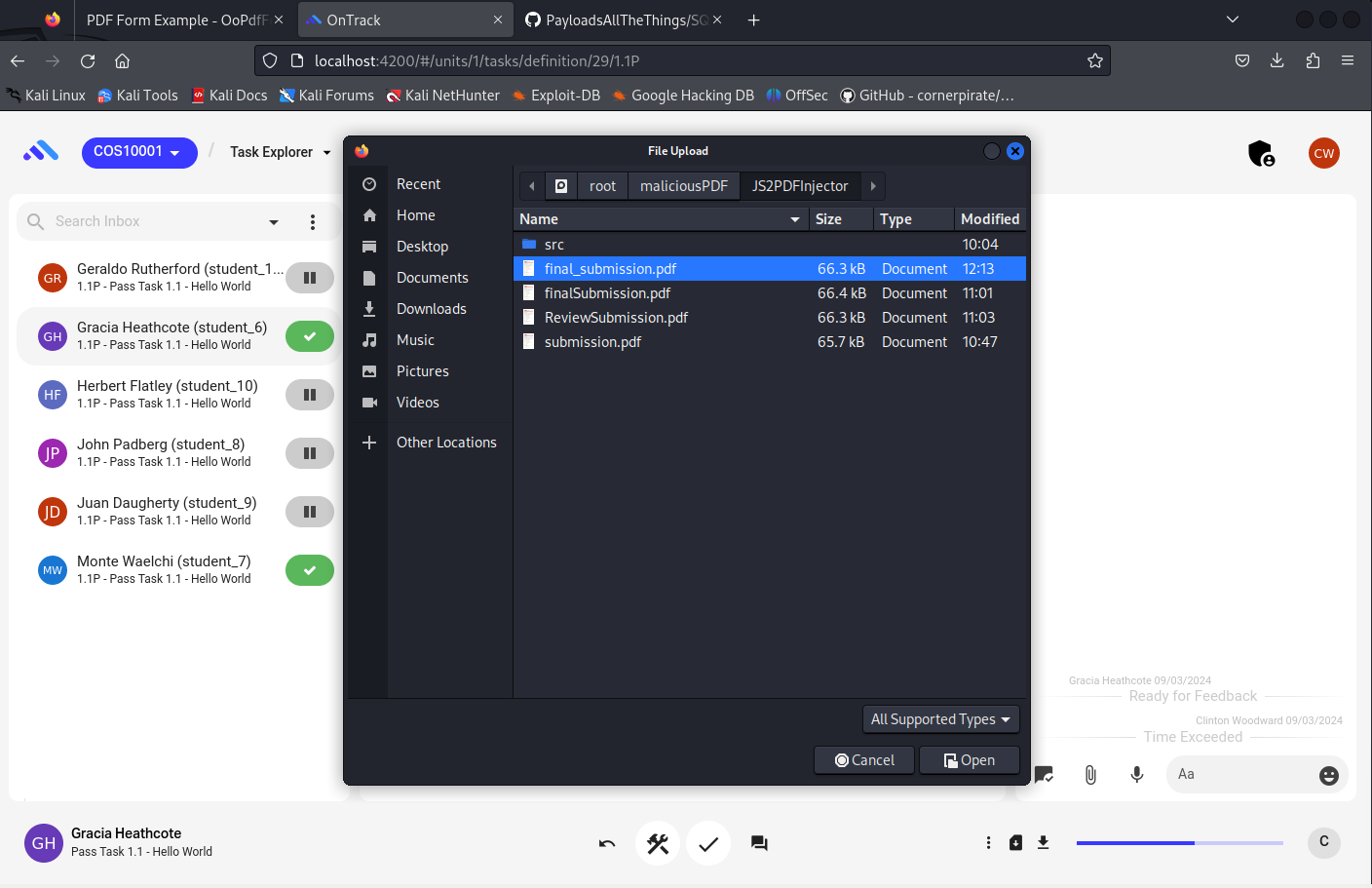


It can be seen from the above screenshot that a poisoned file named *js\_injected\_submission.pdf* has been generated in the same directory. In order to trick the user, we can rename the file to something logical such as *final\_submission.pdf.*

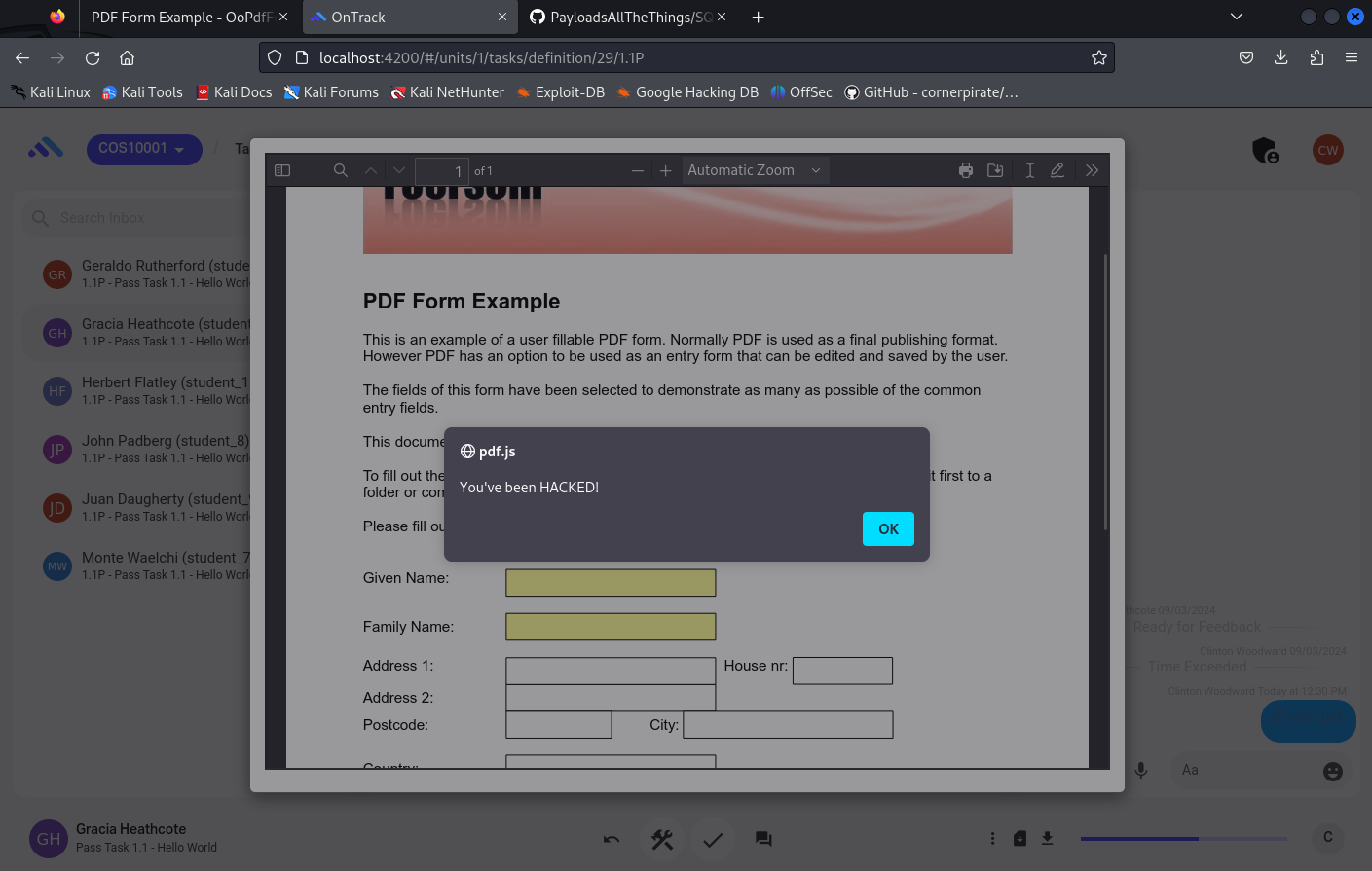


**Step 5: Upload Malicious PDF to inbox/chat box**

Now upload the poisoned file into the inbox.



The file is uploaded to the inbox successfully without any warnings. However, when the file is opened, the embedded JS code runs and performs unexpected actions i.e., in this case, is an alert.



**Remediation Advice**

There are various ways to prevent any impact from this vulnerability by implementing security measures in place and using the right technologies. A few of the remedies are mentioned here:

* **Implement Proper File Validation:** Before enabling uploaded files to be processed or stored through inbox, it is crucial to validate the uploaded file to confirm integrity, format, and content to make sure they satisfy predetermined criteria. This can include examining file signatures and performing content analysis to find and remove harmful information, such as JavaScript encoded in PDF files.
* **Setting up Content Disposition Headers**: It will request the browser to ask the user to download the file rather than running it immediately within the browser environment. It can be achieved by using suitable content disposition headers, such as "Content-Disposition: attachment".
* **Store files in an external directory:** The upload files should be stored into external directories i.e., outside the webroot to prevent attackers from executing malicious files like these through a website URL.

**References**

* Cornerpirate (Dec 2019) - JS2PDFInjector. GitHub: <https://github.com/cornerpirate/JS2PDFInjector>
* BePractical (2023). Unknown [Video]. YouTube. <https://www.youtube.com/watch?v=IJn6_PmXm7E>
* How to prevent file upload vulnerabilities. Available at: <https://webdevolutions.blob.core.windows.net/blog/pdf/how-to-prevent-file-upload-vulnerabilities.pdf> (Accessed: 27 March 2024).

**Contact Details**

**Name:** Nabiha Masood

**Email:** [**s222137149@deakin.edu.au**](mailto:s222137149@deakin.edu.au)

**Pentest Leader Feedback**

**The Finding is well document and is verified on my end. Happy with work.**