CSOC 1030: Assignment #3

July 18, 2023

**Prepared By**: *Ashish Kishor Hedau*

Table of Contents

[File Upload misconfiguration in “http”//10.5.30.30” page 2](#_Toc137283759)

[Description 2](#_Toc137283760)

[Impact 2](#_Toc137283761)

[Recommendations 2](#_Toc137283762)

[Steps to Reproduce 3](#_Toc137283763)

# File upload misconfiguration in “http://10.5.30.30” page.

## Description

The assessed web application “<http://10.5.30.30>” was found to have a File Upload type functionality enabling users to upload File type .docx, .doc, .pdf extension. While this feature is intended to be a fun addition, it suffers from insufficient security measures, which creates the file upload vulnerability. This vulnerability poses a significant risk to the web application and its user’s data, potentially leading to malicious attacks and unauthorized access.

## Impact

The website fails to adequately validate the uploaded file’s formats based on their content. As a result, an attacker can upload files with manipulated extension masquerading as innocent document. An attacker can upload files with executable code and gain control over the server and execute arbitrary commands, potentially compromising the entire system.

## Recommendations

To fix this vulnerability, the administrators of the web application need to validate file type and content by performing thorough validation of uploaded files, ensure they match expected formats and do not contain malicious code. They could also utilize antivirus or file scanning tools to identify and block potentially malicious content before storing files on the server. Third, we highly recommend storing uploaded files in a location separate from the web server’s root directory to prevent direct access.

## Steps to Reproduce

The target application was found to have a File Upload functionality where users can upload files with extension .docx, .doc, .pdf, as pictured below:

![A screenshot of a computer

Description automatically generated]()

We started enumerating the website through a directory brute force, to check for directories that are accessible to the client. We found out that “**/uploads**” path was accessible to the user.

![A screen shot of a computer

Description automatically generated]()

We tried to upload a **.php** file to the web app but due to validation it got rejected.

![A screenshot of a computer

Description automatically generated]()

We uploaded the malicious file with altered extension to the web application with the help of a proxy.

![A screenshot of a computer

Description automatically generated]()

From the image below, we can see the malicious file uploaded in the /uploads directory.

![A screenshot of a computer

Description automatically generated]()

To establish a Remote connection, we started a **netcat** listener on port 4242 on the right terminal as seen from the image. We then used the **curl** command to execute the uploaded malicious file and connect it back to our local machine (172.16.1.5) on port 4242. The connection was successfully established, to confirm we ran a command “**whoami**” which gave **www-data** as the output.

![A screen shot of a computer

Description automatically generated]()