Researcher: Mohamed Salah

Penetration Testing Report

WAKEY Research Labs

Executive Summary

In the comprehensive assessment of the environment, it was observed that all systems were appropriately patched and updated. However, a critical vulnerability lies in the human factor, where errors can potentially jeopardize the entire system's integrity.

The basic upload function in the facing web application, restricted to PDF and DOCX files, was deemed secure during testing, effectively thwarting even highly skilled threat actors. Nevertheless, a security risk emerges when weaponizing DOCX files by injecting shell code as macros. Exploiting human errors among backend users or administrators through such manipulated files can serve as a gateway for attackers to establish a foothold in the network. Subsequently, this could enable the execution of commands on affected machines, ultimately leading to the compromise of the entire infrastructure after exploiting additional misconfigurations within the organization.

While some low-hanging fruit vulnerabilities were identified, it is crucial to note that these alone cannot result in a standalone full compromise.Vulnerability: Macros in DOCX File

Description:

Macros in DOCX files are pieces of code or scripts that can automate tasks within Microsoft Word documents. These scripts can be written in programming languages like VBA (Visual Basic for Applications). While macros can be useful for automating repetitive tasks, they also pose a significant security risk. Malicious actors may exploit macros to execute harmful actions, such as injecting and running code, when a user opens the document.

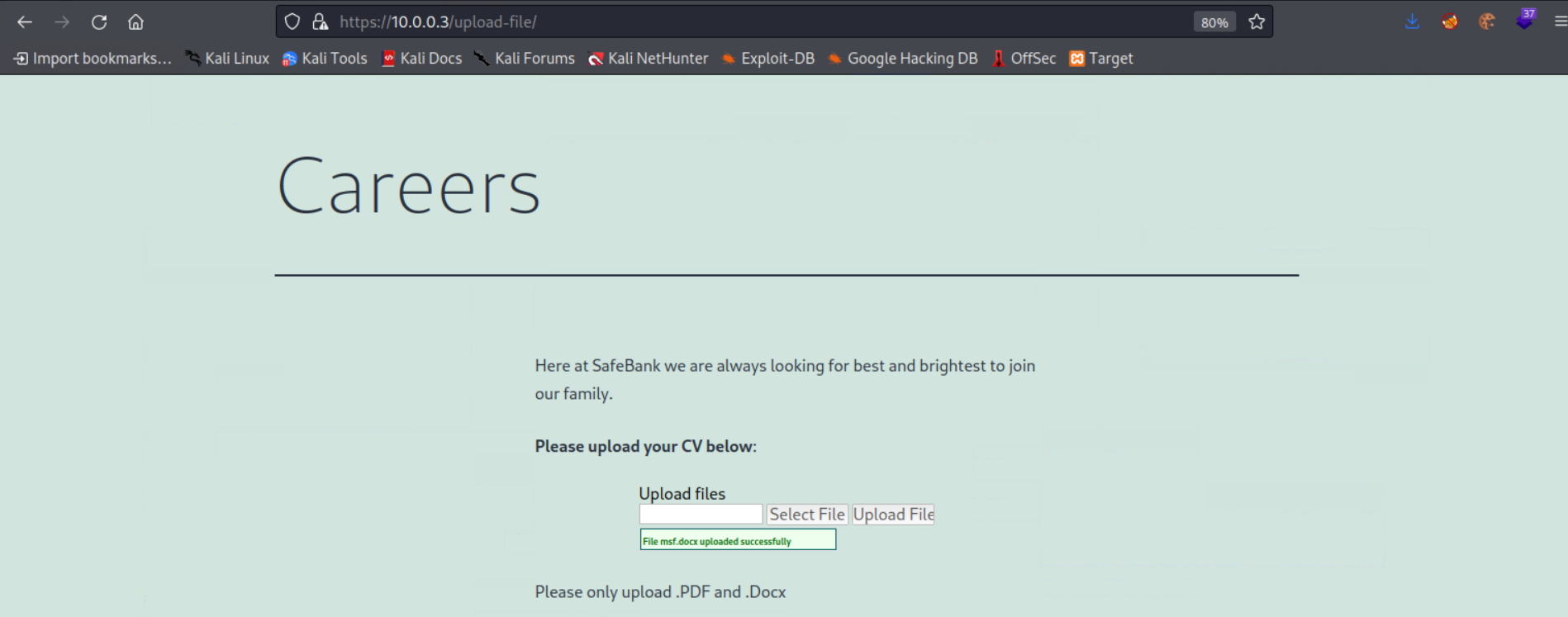
Severity: Critical

Recommendation:

1. Disable Macros by Default: Configure Microsoft Word to disable macros by default. This setting helps prevent the automatic execution of macros when opening a document.
2. Enable Macros Only from Trusted Sources: If macros are necessary for specific tasks, enable them only for documents from trusted sources. Users should exercise caution when enabling macros and verify the authenticity of the document.
3. Use Antivirus Software: Employ robust antivirus software that can detect and block documents with malicious macros. Keep the antivirus definitions up to date for enhanced protection.
4. Educate Users: Provide comprehensive training to users about the potential risks associated with macros. Encourage them to be cautious when opening documents from unknown or untrusted sources.
5. Implement Group Policies: Utilize Group Policies in an enterprise environment to control macro settings centrally. This ensures a consistent and secure configuration across all systems.
6. Regularly Update Software: Keep Microsoft Word and other relevant software up to date with the latest security patches. Software updates often include fixes for known vulnerabilities.
7. Use Advanced Threat Protection: Consider implementing advanced threat protection solutions that can analyze and block malicious documents before they reach end-users.
8. Inspect Outbound Traffic: Monitor and inspect outbound network traffic for suspicious activities, especially if a document with macros attempts to communicate with external servers.
9. Apply Principle of Least Privilege: Limit user privileges to reduce the potential impact of a macro-based attack. Users should only have the permissions necessary for their roles.
10. Regular Security Audits: Conduct regular security audits and assessments to identify and remediate potential vulnerabilities in the organization's document handling processes.

PoC:

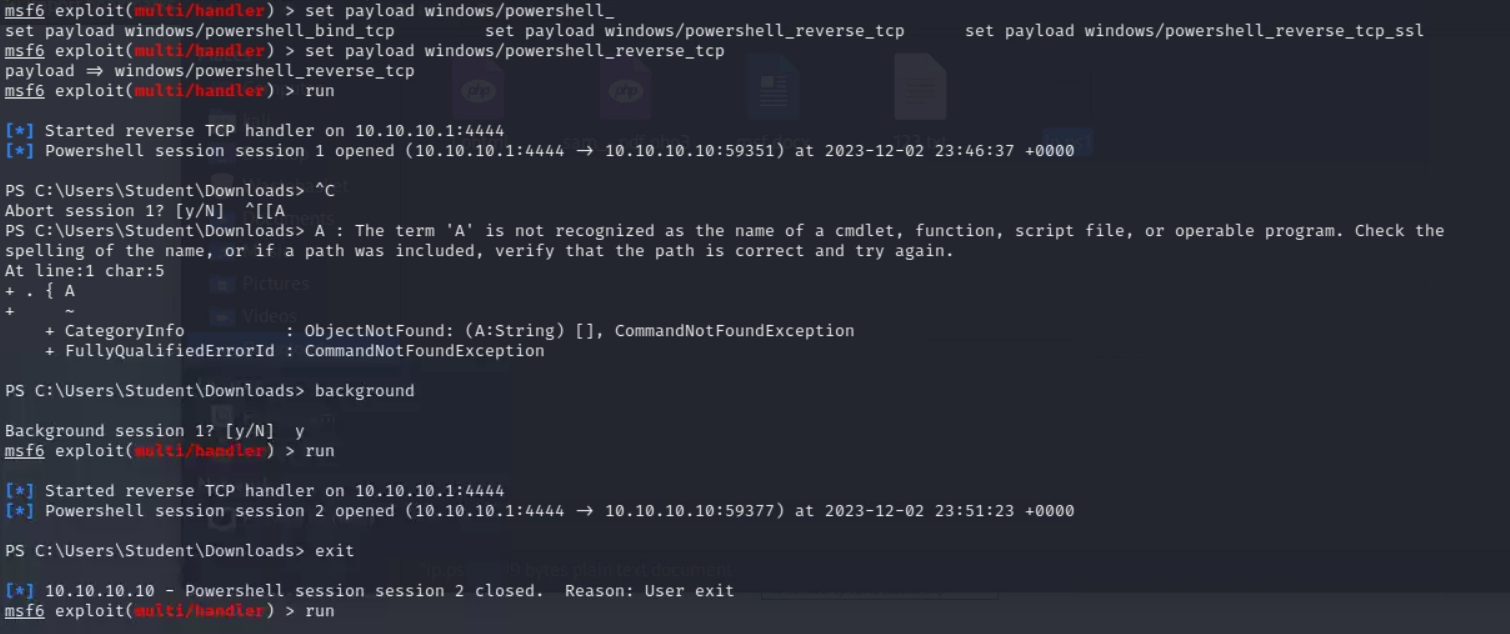
1- Upload form was existing in careers page and only accepting docx and pdf files



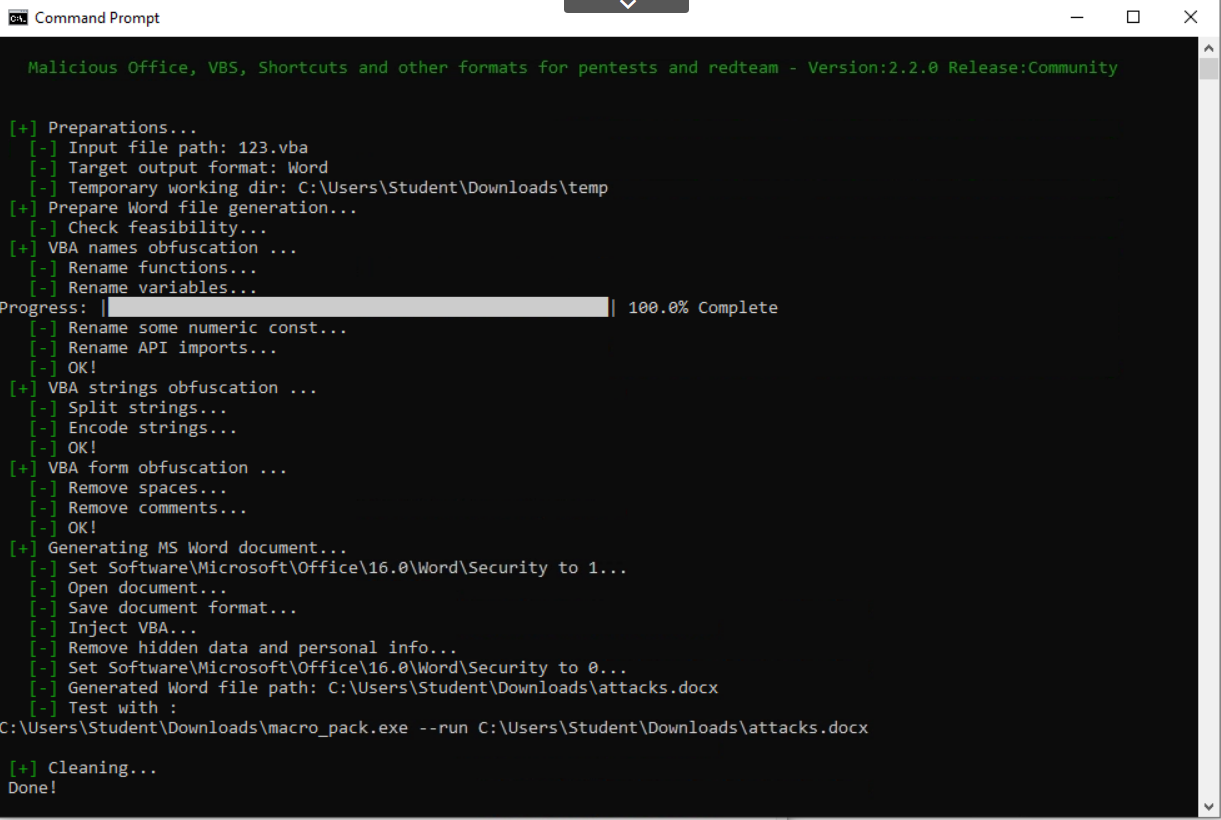
2- It took some time to figure out it need to be a rely on human error and human interaction as someone should be reviewing this files manually.

3- To weaponzie the docx file I had to write a macros that execute a powershell command which deploying a reverse shell to the meterpreter listener I’m on it.

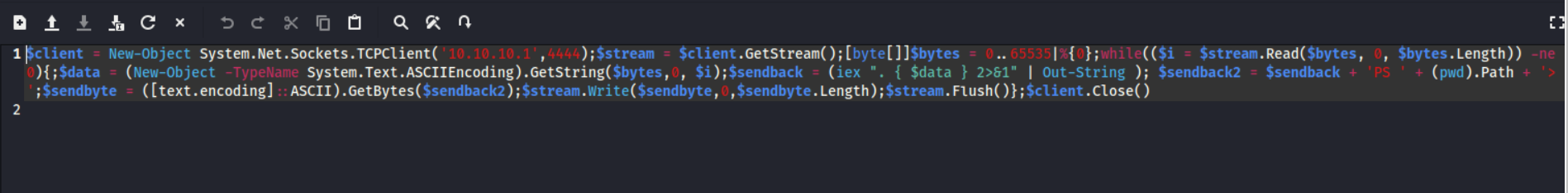
4- Stopped the antivirus and made sure it’s working first between my windows and Linux environment.



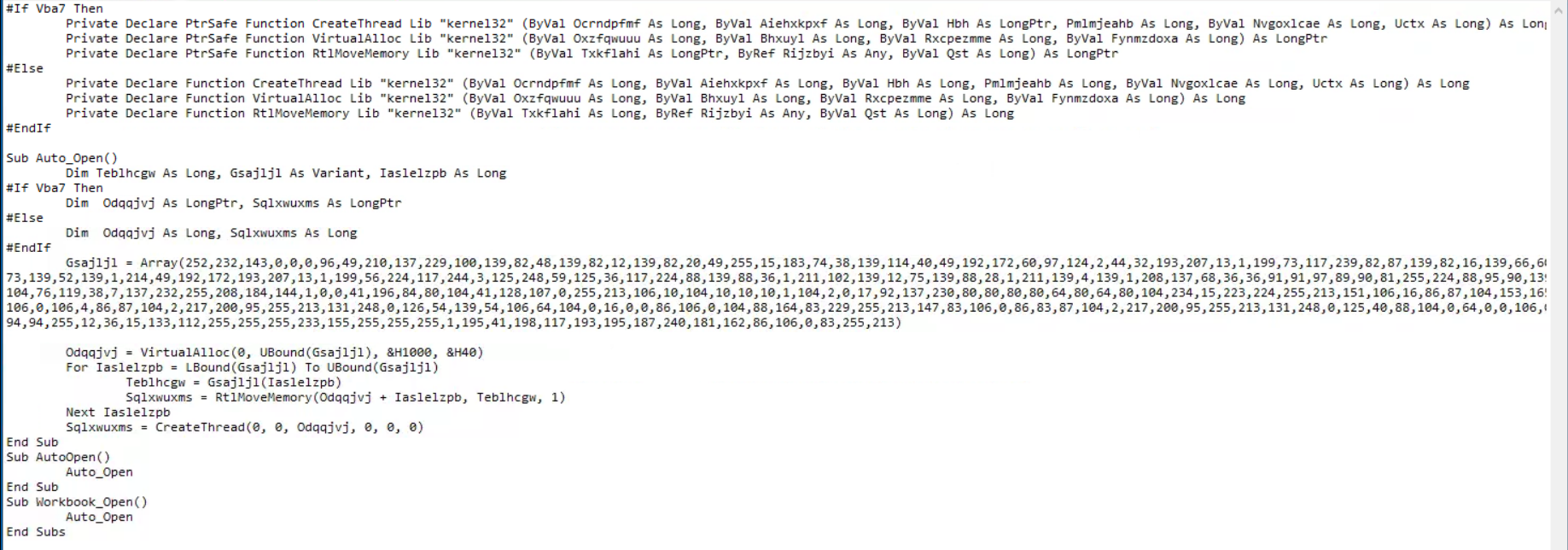
5- then I need to make the macros obfuscated to bypass the security endpoints and the antivirus on the victim machine, used an exploit-kit called macro pack to help obfuscating it.



6- This was the powershell command that will execute to give me the reverse shell

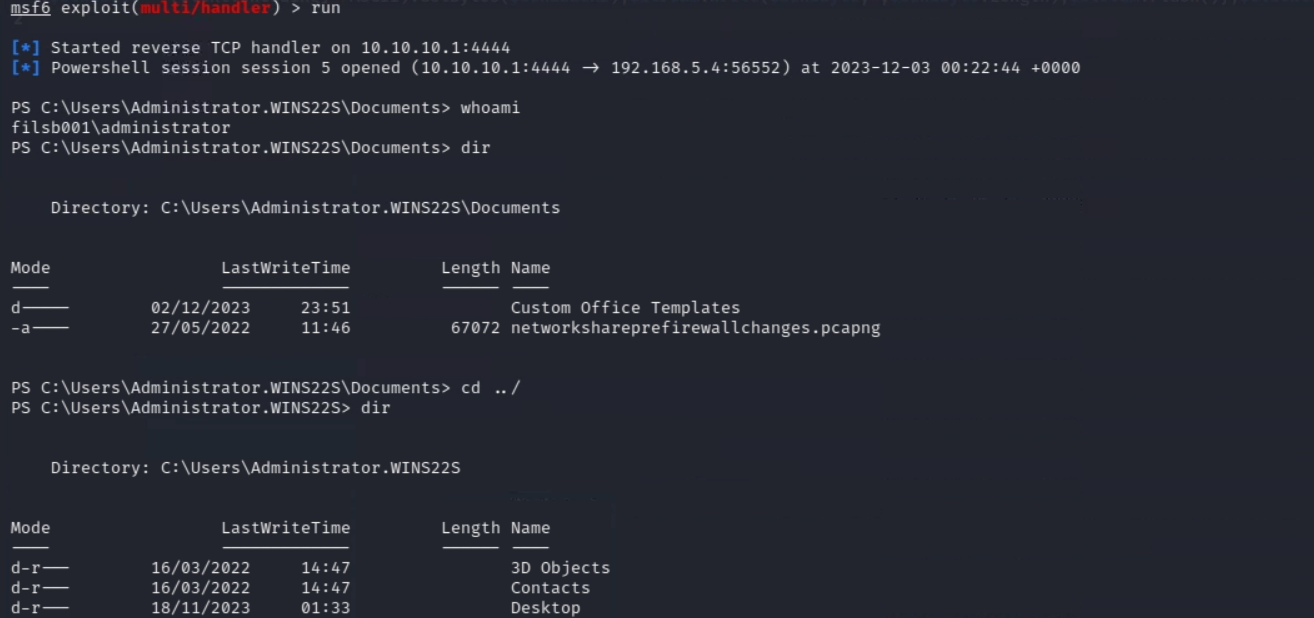


7- Sample of the obfuscated VBA code which will execute once the document opened by the victim.



8- Tested it again and made sure it’s working while antivirus enabled, then supply it on the careers page.

9- After a while once it get opened by the end user, the metepreter listener worked and give me an access to the machine as administrator, which is another infrastructure flaw that this user was logged in as administrator not regular system user.



Vulnerability: Directory Listing

Description:

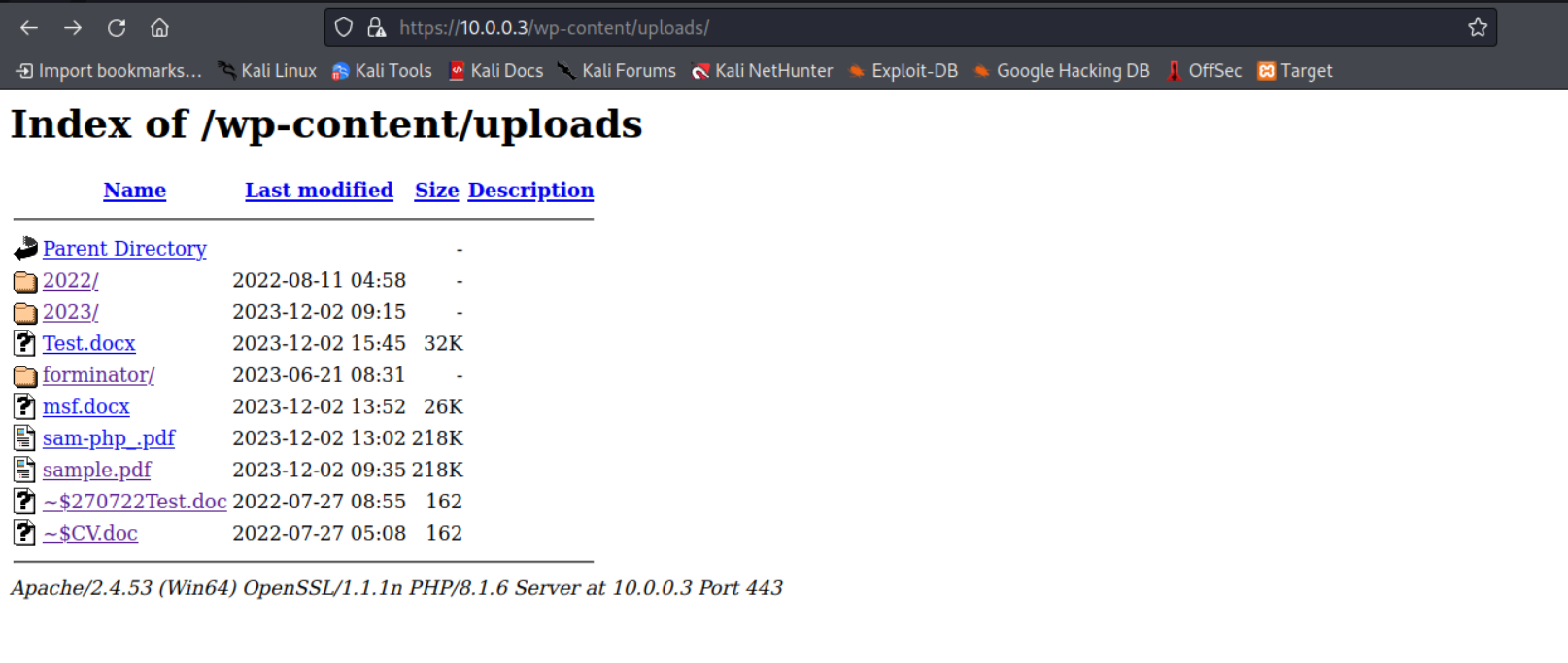
Directory listing is a web server function that displays the directory contents when there is no index file in a specific website directory. It is dangerous to leave this function turned on for the web server because it leads to information disclosure..

Severity: Low

Recommendation:

You should make sure no sensitive information is disclosed or you may want to restrict directory listings from the web server configuration.

PoC:



Vulnerability: Server Banner Disclosure

Description:

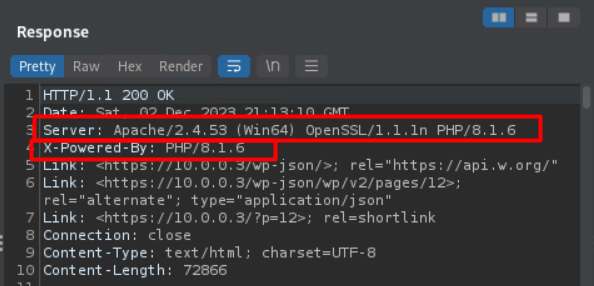
The web server is sending the X-Powered-By: response headers, revealing the PHP version.

Severity: Low

Recommendation:

Configure your web server to prevent information leakage from its HTTP response.

PoC:



Vulnerability: WordPress username enumeration

Description:

In default WordPress installation there are several methods to enumerate author’s username. These WordPress users can then be used in brute-force attacks against WordPress login page to guess passwords.

Severity: Low

Recommendation:

Block requests to sensitive user information at the server using .htaccess file or WAF for example. You should block or redirect all requests made to '/wp-json/wp/v2/users/' and to 'author' parameter (via GET and POST requests).