Lab: File path traversal, traversal sequences blocked with absolute path bypass

**1. Summary**

**Bug Title:** a path traversal vulnerability in the display of product images in /image?filename= parameter.

**Severity:** High.

**Description**: A Path Traversal vulnerability was identified in the /image?filename= parameter of the application. This vulnerability allows an attacker to manipulate the filename parameter to access files outside the intended directory. By exploiting this, an attacker could retrieve sensitive files from the server, such as configuration files, credentials, or other confidential data.

**Date Discovered:** 17/8/2024.

**Status:** Solved.

**2. Bug Details**

**Vulnerability Type:** Path Traversal.

**Affected URL/Endpoint:** /image?filename= .

**Description:** The application contains a Path Traversal vulnerability in the /image?filename= parameter, which is designed to serve product images to users. Path Traversal, also known as Directory Traversal, occurs when an attacker manipulates input to navigate directories on the server's file system outside of the intended directory structure. This vulnerability is particularly dangerous as it can allow unauthorized access to sensitive files on the server, leading to data exposure, system compromise, or further attacks. The vulnerability arises due to insufficient validation and sanitization of the filename parameter in the URL. When a user requests an image, the application dynamically generates a file path based on the value provided in this parameter. In a secure implementation, the application should only serve files from a designated directory (e.g., /var/www/html/images/). However, if the filename parameter is not properly validated, an attacker can modify it to include directory traversal sequences, such as ../, to escape the intended directory and access files in other parts of the file system. However this lab filter the escapping sequences well **we can bypass it via absolute path**

**Steps to Reproduce:**

1. Go to Lab URL: <https://portswigger.net/web-security/file-path-traversal/lab-absolute-path-bypass> and click access lab

2. Click on any product and intercept the request using any interception proxy (i used burp) and forward the requests until you hit the request for fetching the image as below



3. modify filename parameter with this payload /etc/passwd and send the request then examine the response you will see that we get passwd file conent



**Proof of Concept (PoC):**



**Impact:** Exploiting this vulnerability can lead to unauthorized access to sensitive information, potentially exposing critical system files, user data, or configuration files. In a worst-case scenario, it may allow an attacker to gain further access to the system or escalate their privileges..

**3. Recommendations**

1. **Input Validation:** Ensure that the input for the filename parameter is restricted to valid and expected file paths only. Avoid accepting user input that contains directory traversal characters like ../.
2. **Sanitize User Input**: Strip out or encode special characters and patterns that could lead to directory traversal. Alternatively, resolve the input to a canonical path and ensure it stays within the intended directory.
3. **Use Whitelisting:** Only allow filenames that match a predefined whitelist of acceptable values or extensions.
4. **Access Control:** Ensure that file permissions and web server configurations are set up to restrict access to sensitive directories and files.

**4. Conclusion**

**Summary:** The identification of the Path Traversal vulnerability in the /image?filename= parameter highlights a significant security flaw that could have serious consequences for the application's security posture. Left unaddressed, this vulnerability can be exploited by attackers to gain unauthorized access to sensitive files on the server, potentially leading to data breaches, system compromise, and further malicious activities. The importance of fixing this bug cannot be overstated. By resolving this issue through proper input validation, sanitization, and secure coding practices, the organization can prevent unauthorized file access, protect sensitive information, and maintain the integrity and confidentiality of its systems. Addressing this vulnerability is crucial not only for safeguarding the application but also for preserving the trust of users and ensuring compliance with data protection regulations.

**5. Appendices**

**Tools Used:**  Burp Suite.

**References:**

* <https://portswigger.net/web-security/file-path-traversal>
* <https://owasp.org/www-community/attacks/Path_Traversal>
* <https://learn.snyk.io/lesson/directory-traversal/>