**Technical Writeup**

**(1)Vulnerability:**

Directory traversal is a vulnerability that occurs when an application allows an attacker to access files and directories that are outside the intended directory.

The vulnerability lies in the ‘get\_req.php’ file, specifically in the handling of the ‘cont’ parameter passed via a GET request. It directly uses this parameter to open a file without proper validation, allowing potential directory traversal.

1. **Input Handling:**

* The PHP script ‘get\_req.php’ accepts input through the ‘cont’ parameter obtained from the GET request ($\_GET['lang']) without validating or sanitizing it.

1. **File Operation without Validation:**

* The vulnerable part of the script uses the received ‘cont’ parameter directly within the ‘fopen()’ function without proper validation or sanitation.
* The script intends to read the content of a file using fread() specified by the user-provided cont’ and displays it on the page.

1. **Potential Exploitation:**

* An attacker can manipulate the ‘cont’ parameter to navigate through directories using traversal sequences like ../ or ../../ to move up parent directories.

1. **Impact of Successful Exploitation:**

* If the traversal is successful and the server allows access to sensitive files, an attacker might access files outside the intended directory structure.
* For instance, accessing system configuration files (/etc/passwd on Unix-like systems) might reveal sensitive information to the attacker.

1. **Directory Traversal Risk:**

* Since there's no validation or restriction, an attacker can manipulate the ‘cont’ parameter to navigate through directories on the server, potentially accessing sensitive files outside the intended directory.

**Steps to replicate the vulnerability:**

1. Access the vulnerable get\_req.php script, through a web browser.
2. Use the 'cont' parameter to perform directory traversal by inputting something like .../../../../../../etc/passwd or similar traversal strings to access files outside the intended directory.

**<http://localhost/get_req.php?name=Pranjali&email=pranjalimailaddress&lang=../../../../../../etc/passwd>**

1. If the directory traversal is successful, the script will attempt to open the specified file (e.g., /etc/passwd on Unix-like systems), and its content may be displayed on the web page.  
     
     
   My selenium script has some version issues so if not able to run it in some other system, here is another vulnerability.

**(2)Vulnerability:**

This vulnerability involves exploiting the HTTP PUT method to upload a malicious PHP script onto the server. The script, when executed, allows an attacker to issue commands on the server, potentially leading to unauthorized access to server information or control. This specific vulnerability demonstrates how an unsecured web server can be manipulated to execute arbitrary system commands via a web interface.

The described vulnerability is a type of Remote Code Execution (RCE) and potentially a Path Traversal attack.

The server allows the PUT method, which is not typically used in standard web applications and should be disabled or properly secured. Additionally, the server fails to sanitize inputs or restrict the types of files and commands that can be uploaded and executed.

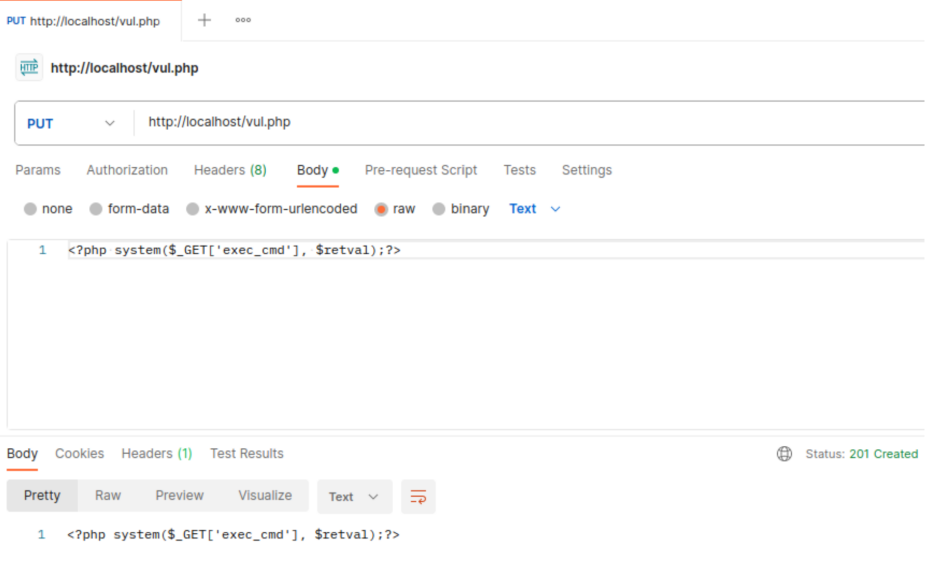
**Impact of Successful Exploitation:**

If exploited, an attacker can gain insights into the server's operating system and architecture. More maliciously crafted commands could lead to further exploitation, data theft, or unauthorized control over the server.

**Steps to replicate the vulnerability:**

1. Uploading the Malicious PHP Script:

This PHP script uses the system() function to execute a command from the exec\_cmd GET parameter. The script is uploaded to the server using the PUT method via POSTMAN.



1. Executing Commands via the Uploaded Script:

The system() function in the PHP script executes the provided command (uname - mrs), which returns information about the server's operating system, machine type, and kernel.

