**File Inclusion:**

File Inclusion is a vulnerability in which an attacker is allowed to include a file to a website usually through a script on the web server. There are a couple types of file inclusion: Remote File Inclusion, and Local File Inclusion.

Remote File Inclusion: The vulnerability occurs due to the use of user-supplied input without proper validation.

 Local File Inclusion: is similar to a *Remote File Inclusion* vulnerability except instead of including remote files, only local files i.e. files on the current server can be included. The vulnerability is also due to the use of user-supplied input without proper validation.

Successful Remote File Inclusion attacks allow malicious users to run their own PHP code on a vulnerable website. The attacker is allowed to include his own (malicious) code in the space provided for PHP programs on a web page.

To mitigate this risk, set **allow\_url\_fopen** and **allow\_url\_include** to "**Off**" in:

/etc/php5/cli/php.ini

/etc/php5/cgi/php.ini

To commit the changes, restart the httpd daemon:

**Apache:**

* **service httpd restart** or
* **/etc/init.d/httpd restart** or
* **/etc/init.d/apache2 restart**

Whatever works for your distro...

RFI Detection:

Just a few commands to find out who's knocking. Feel free to expand on what's here as new RFI attacks are discovered.

**grep -hr '\.\.\/\.\.\/' /var/log/lighttpd/\*.log | sed 's/ - \[.\*"GET / "/' | awk '{print $1" - "$2" - "$3}' > /tmp/rfi-attacks.log**

**grep -hr '\/etc\/passwd' /var/log/lighttpd/\*.log | sed 's/ - \[.\*"GET / "/' | awk '{print $1" - "$2" - "$3}' > /tmp/rfi-attacks.log**

Real World Output (target domain omitted to protect the innocent):

**Remote File Inclusion** (also known as RFI) is the process of including remote files through the exploiting of vulnerable inclusion procedures implemented in the application. This vulnerability occurs, for example, when a page receives, as input, the path to the file that has to be included and this input is not properly sanitized, allowing external URL to be injected. Although most examples point to vulnerable PHP scripts, we should keep in mind that it is also common in other technologies such as JSP, ASP and others.

The most effective solution to eliminate file inclusion vulnerabilities is to avoid passing user-submitted input to any filesystem/framework API. If this is not possible the application can maintain a white list of files, that may be included by the page, and then use an identifier (for example the index number) to access to the selected file. Any request containing an invalid identifier has to be rejected, in this way there is no attack surface for malicious users to manipulate the path.

Almost all web application frameworks support file inclusion. File inclusion is mainly used for packaging common code into separate files that are later referenced by main application modules. When a web application references an include file, the code in this file may be executed implicitly or explicitly by calling specific procedures. If the choice of module to load is based on elements from the HTTP request, the web application might be vulnerable to RFI.

An attacker can use RFI for:

* Running malicious code on the server: any code in the included malicious files will be run by the server. If the file include is not executed using some wrapper, code in include files is executed in the context of the server user. This could lead to a complete system compromise.
* Running malicious code on clients: the attacker's malicious code can manipulate the content of the response sent to the client. The attacker can embed malicious code in the response that will be run by the client (for example, Javascript to steal the client session cookies).

PHP is particularly vulnerable to RFI attacks due to the extensive use of "file includes" in PHP programming and due to default server configurations that increase susceptibility to an RFI attack ([4,5]).