J.BEDEL ON SECURITY

EXPLOITING PHP PATH TRUNCATION [PHP < 5.3]

WHEN

Sometimes you will encounter websites that navigate through their pages using a PHP parameter. Here is an example URL to give you the idea of what your are looking for :

```
http://website.com/index.php?page=page_name
```

It is most likely using this type of PHP code to include the file page_name.php :

```
if(isset($_GET['page'])
{
    include($_GET['page'].".php");
}
```

This code is vulnerable to Local File Inclusion (LFI). Indeed, we are able to include any PHP file on the server and eventually read its content using PHP wrappers.

Even worse, if we manage to escape the .php which is appended at the end of the path we could include any sensitive file on the server (/etc/passwd for example) :

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

This is commonly done by adding a null-byte (%00 in HTTP) at the end of the requested path, escaping the ".php" at the end.

That's why null-byte are usually filtered. PHP path truncation can be exploited to get another way of escaping the file extension.

HOW

In PHP version < 5.3, string maximum length is 4096 characters. If it encounters a string longer than that it will simply truncate it, erasing any character after the maximum length. This is exactly what we want in order to escape the file extension of our LFI vulnerability !

But how to add so much caracters to our path ? We are going to use the way PHP filesystem related functions (especially include()) normalizes path. In fact, in PHP `/etc/passwd/.` will be interpreted as `/etc/passwd` by those functions, allowing us to add `/.` until we reach a path of 4096 characters and escape the file extention.

The idea is here, but in theory it is a bit more complicated than that, and our path have to respect the following conditions :

- must start with an unknown path
- must have an odd number of characters
- must end with a dot

So the requested URL will more likely be as follows :

```
http://website.com/index.php?page=random_path/../etc/passwd/.
[repeated multiples times to reach a path size of 4096 char]/.
```

Live example to train on

As usual I recommand the website root-me.org to train on your hacking skills, here is a challenge on which you can practice this vulnerability exploitation.

hhttps://www.root-me.org/en/Challenges/Web-Serveur/Path-Truncation (https://www.root-me.org/en/Challenges/Web-Server/Path-Truncation)

Sources

Whitepaper by Francecso Ongaro and Guivani Pellerano, with all the theorical details and extensive examples :