



Secure Use Case

Preventing LFI & RFI

Three key ways to prevent RFI attacks are:

- Never use arbitrary input data in a literal file include request
- Use a filter to thoroughly scrub input parameters against possible file inclusions
- Build a dynamic whitelist

Like all code injection attacks, LFI and RFI are results of allowing unsecure data into a secure context.

The best way to prevent an RFI attack is to never use arbitrary input data in a literal file include request.

Taking the example from earlier, a more secure way of implementing this demo is by using an whitelist to prevent LFI and RFI:

lfi.php

```
<?php
```

```
$page='pages/home.php';
```

```
if (isset($_GET['page']))
```

```
{
```

```
switch($_GET['page'])
```

```
{
```

```
case 'home' :
```

```
case 'login': $page='pages/'.$_GET['page']; break;
```

```
}
```

```
}
```

```
?>
```

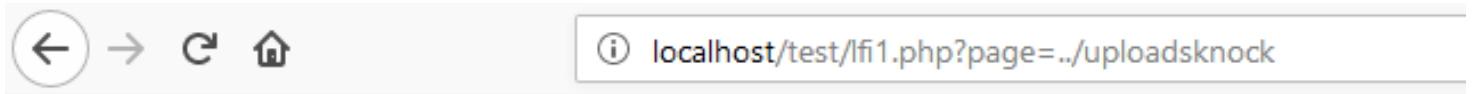


```
<a href="?page=home.php">Home</a>&nbsp;&nbsp;&nbsp;<a href="?page=login.php">Login</a>
```

 ProActive Control for Software Security... Home  Concept overview of Logging and

include (\$page);

In this case, even upload the malicious script successfully, it still can't be run.



[Home](#) - [Login](#) Wellcome to home

