psychoPATH – usage scenarios

Table of contents

Hunting LFI (Local File Inclusion aka arbitrary file read)

The *Path traversal* generator can be easily used for hunting Local File Inclusion/arbitrary file reading issues as well - and it's simpler than hunting uploads.

The https://github.com/ewilded/psychoPATH/test_cases/LFI directory contains three vulnerable PHP scripts, reflecting the non-recurrent filter cases broken down in the *Evasive techniques* section of the README.

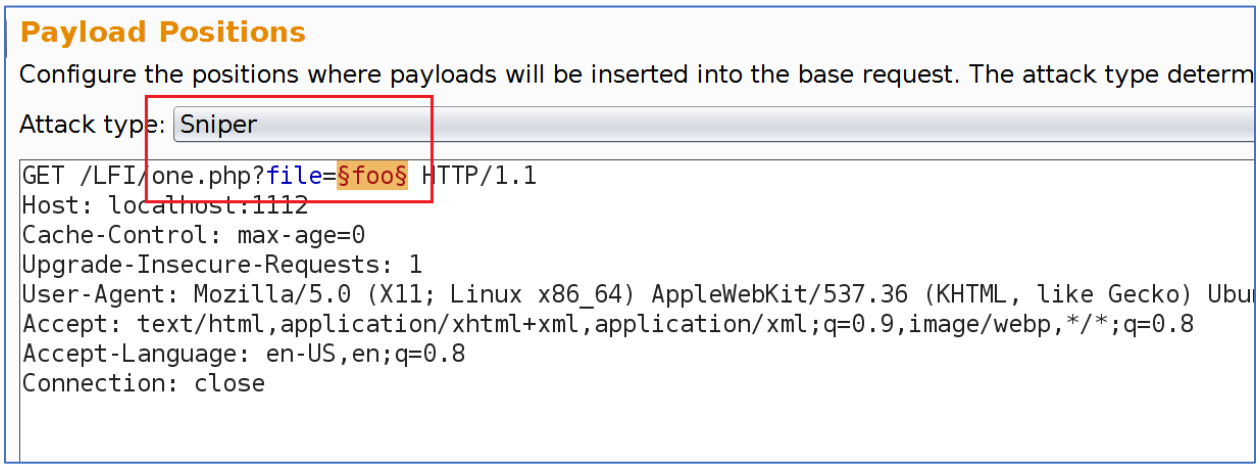
Below is a short presentation on how all three can be quickly detected with the payloads provided by psychoPATH.

test_cases/LFI/one.php:

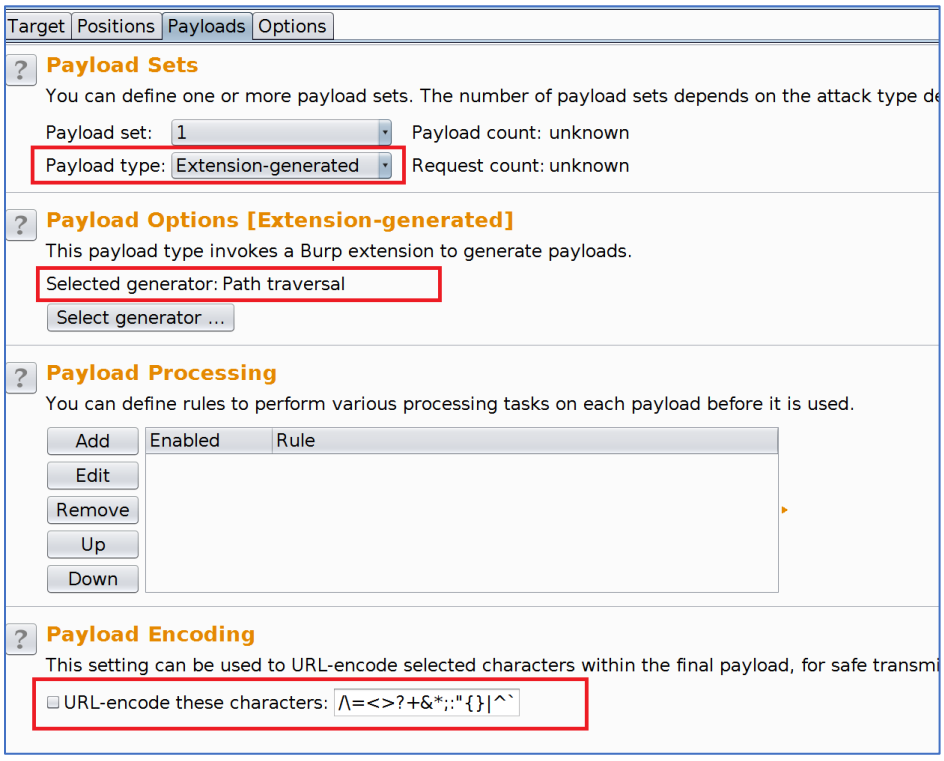
```
1<?php
2
3if(isset($_GET['file']))
4{
5        $file=str_replace('../','',$_GET['file']);
6        echo @file_get_contents('./'.$file);
7}
8#removing only ../
9?>
```

First screenshot shows us the response of the script when a benign string *foo* is provided under the *file* variable. No content is returned:

We send the request to Intruder and mark the injection point:



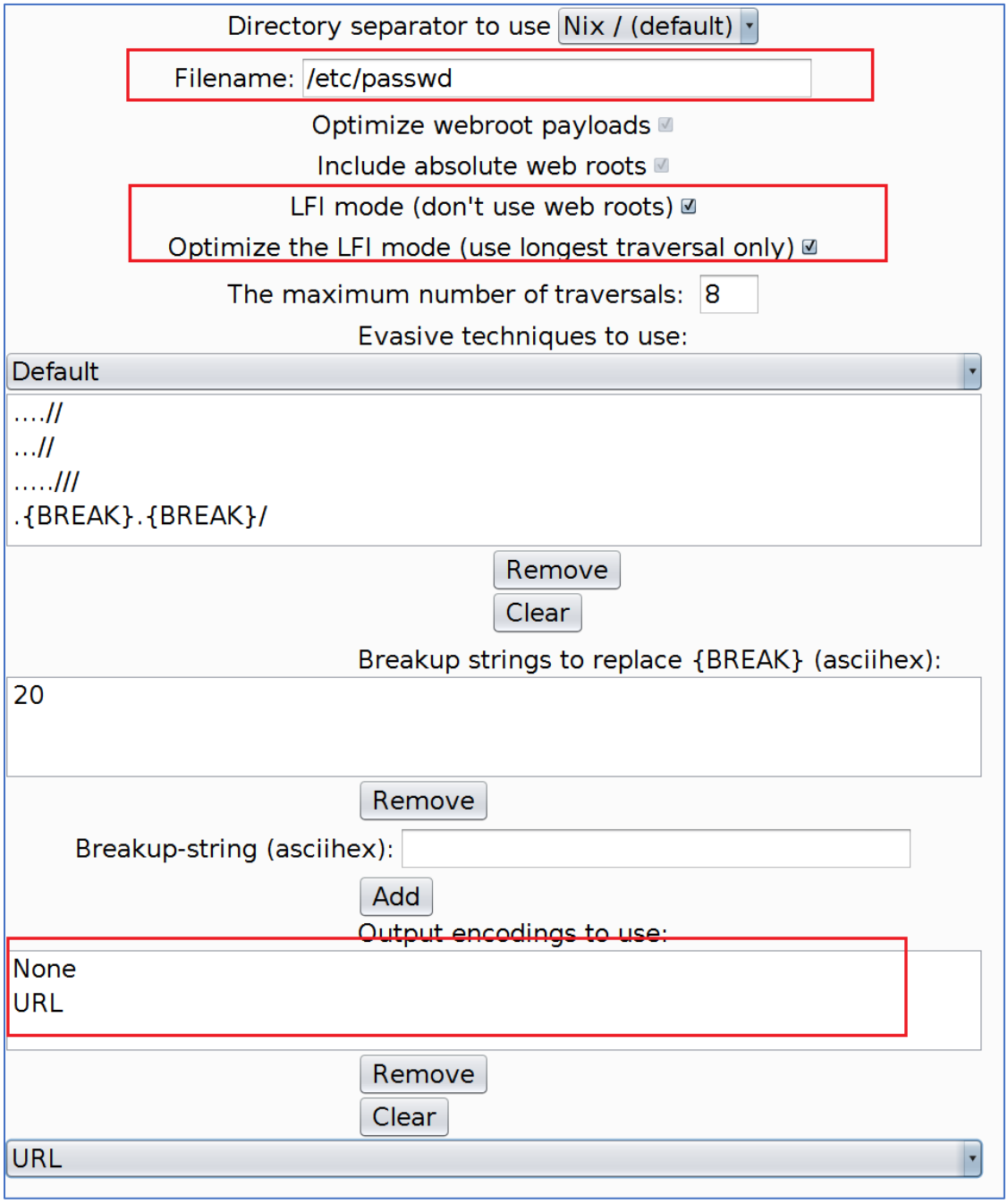
In payload options, we choose *Extension generated -> Path traversal* payload type:



Please remember to uncheck the *URL-encode these characters* box – the output encoding is handled by the plugin itself - and customizable.

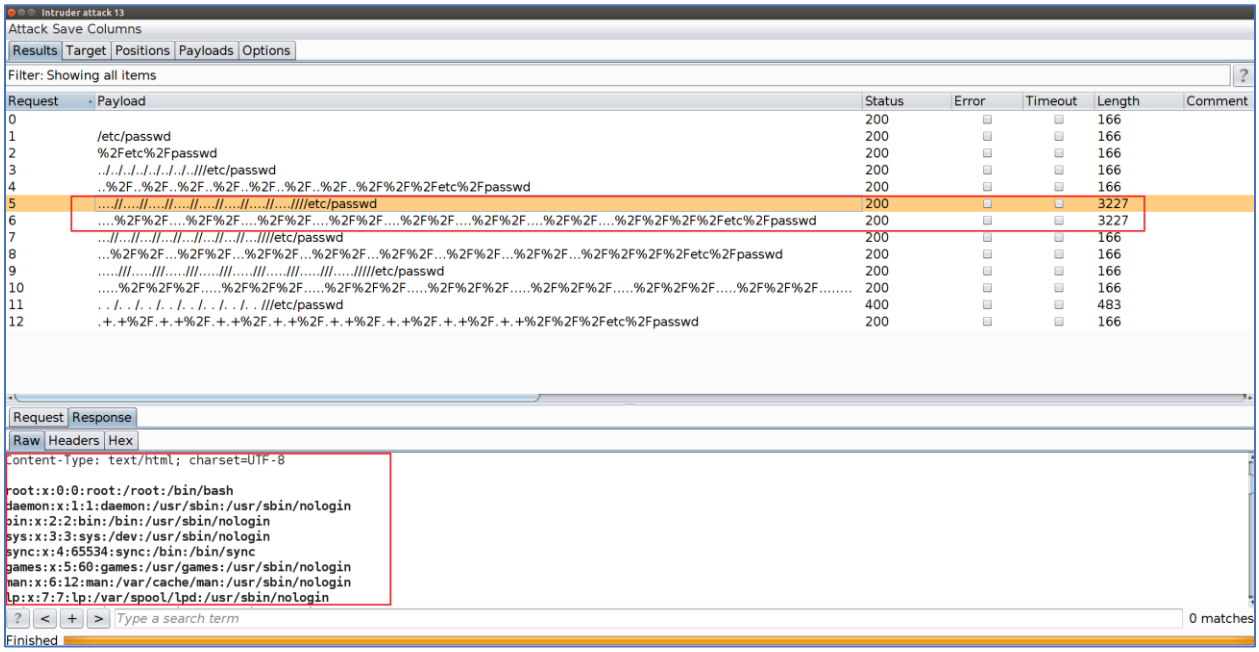
Now let's move to the psychoPATH configuration panel. We choose the file name we would like to read, for instance /etc/passwd.

We turn the *LFI mode* on, so no payloads involving web roots will be used. We can also enable *LFI optimization* to only use the longest traversal payloads (reasonable when the file we are requesting is an absolute path, like /etc/passwd):



Also, it is safe to choose two output encoding modes; *None* and *URL-encoding*.

We proceed to Intruder, simply run *Start attack* and watch:



As it can be seen in Intruder's output, two variants (URL-encoded and raw) of the non-recurrent evasive payload ….// hit the file.

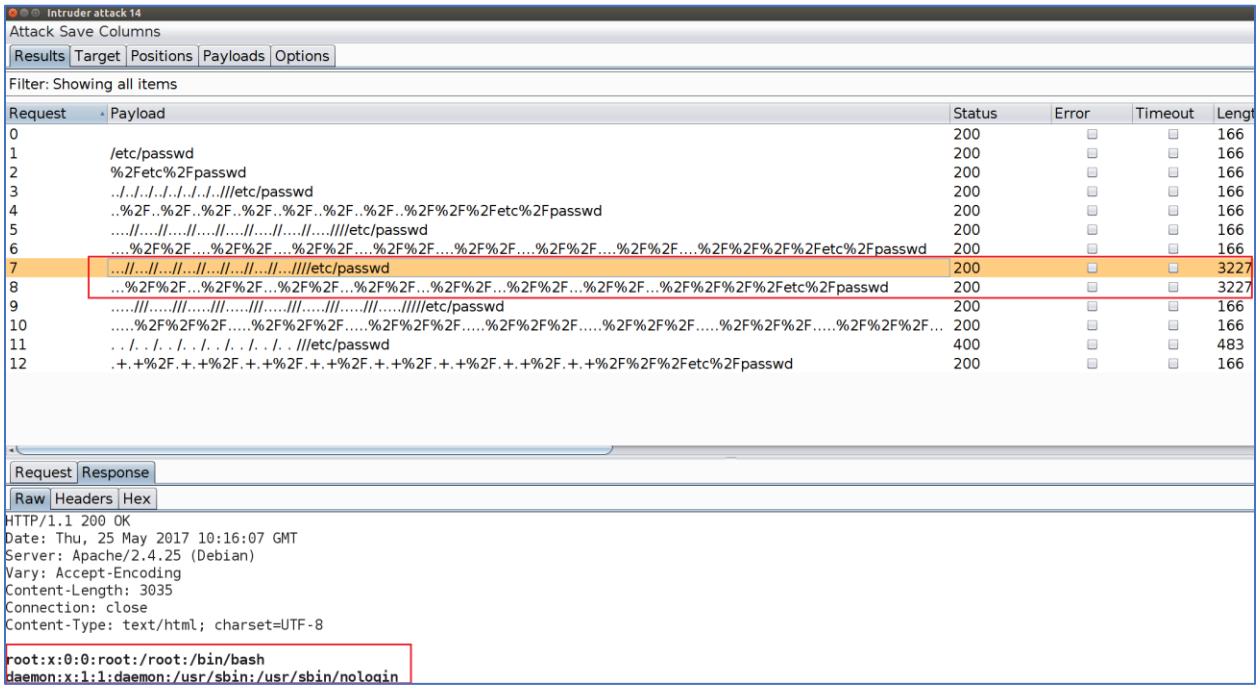
Let's move on to another example.

two.php:

```php
<?php

if(isset($_GET['file']))
{
        $file=str_replace('./','',$_GET['file']);
        echo @file_get_contents('./'.$file);
}
#removing only ./
?>
```

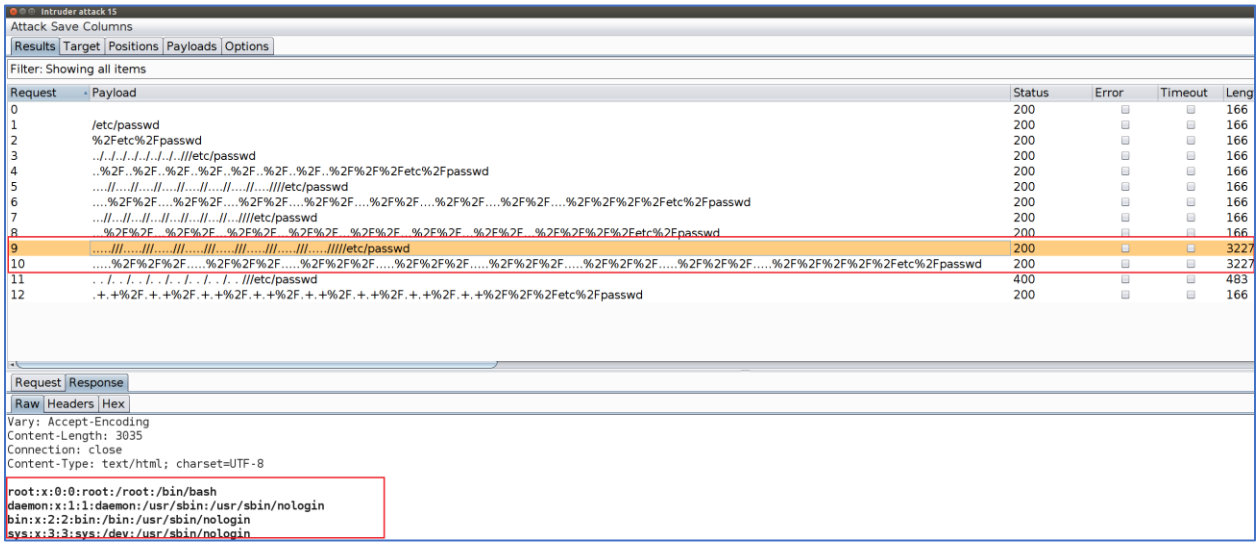
With the exactly same configuration, the Intruder attack results are as follows:



three.php:

```php
<?php

if(isset($_GET['file']))
{
        $file=str_replace('../','',$_GET['file']);
        $file=str_replace('./','',$file);
        echo @file_get_contents('./'.$file);
}
#removing ../ then ./
?>
```

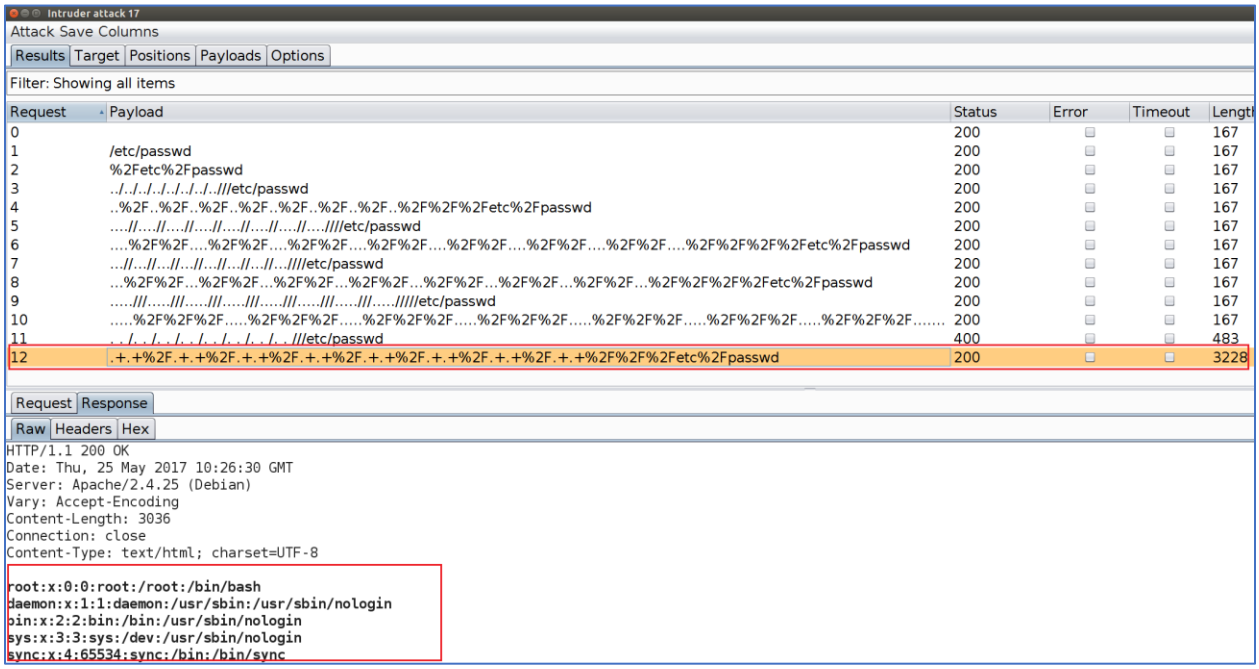


four.php:

```php
<?php

if(isset($_GET['file']))
{
        $file=str_replace('..','',$_GET['file']);
        $file=str_replace(' ','',$file); // removing white spaces
        echo @file_get_contents('./'.$file);
}
#removing .. then white spaces
?>
```

And the winner is:



Hunting uploads in the dark

The interface contains several configurable lists of elements used to build permutations of all potentially valid web root paths:

TBD