Final Project

INTRODUCTION TO APPLICATION SECURITY – ATTACKS & DEFENSE

VS

LFI

Alex Shleymovich

The Academic College of Tel Aviv-Yaffo

208439844

About the website

I built a simple online store – Alex Shop – where user can choose between 2 available items (2 T-shirts). The store was built with Bootstrap 4 and PHP v8.0.11. For this project I used XAMPP to run Apache server on my localhost (Apache v2.4.51). Alex Store has the following pages:

Index. html – landing page, from this page user can choose an item and proceed to the product page. **Note**: About, Contact Us and Products nav-items are all pointing to Index.html
products.php – a page that contains the PHP functions as well as some HTML code. In this page there are 3 PHP functions with the same functionality – each of them includes the product page that was chosen in Index.html with the help of include(). Upon running the wrong function is being called which cause the LFI vulnerability. However, correct functions are also appearing there but aren't called.
product1.html – simple html page that contains content about T-Shirt #1. Located inside includes folder
product2.html – simple html page that contains content about T-Shirt #2. Located inside includes folder
style.css – a CSS file which contains all the style configurations for all the project

In this website I exploit the LFI vulnerability. By saying this I mean that my website uses a file path as an input and the website treats that input as trusted and safe. This can lead to "Directory Traversal" attacks, where an attacker will try to find and access files on the web server to gain access to sensitive data etc.

How to exploit Local File Inclusion in Alex Shop?

I divided a website's code into directories, multiple files, etc. to make everything neat and readable. In order for the browser to find these files I was required to designate the correct file path and then pass it to a function. The function then opens the file in question and includes it inside the document for the browser to be able to see is as valid code.

When the user lands on the website he receives the following page:

http://localhost:8080/Final%20Project/

After choosing one of the represented items the customer ends up on the product page of the item:

http://localhost:8080/Final%20Project/products.php?product1=product1

http://localhost:8080/Final%20Project/products.php?product2=product2

By replacing product1/product2 the vulnerability allows the attacker to use Directory Traversal attacks which allow to navigate through files inside and outside of the web application directory.

For example, we can use the following sequence "../index" to get back to the index.html page:

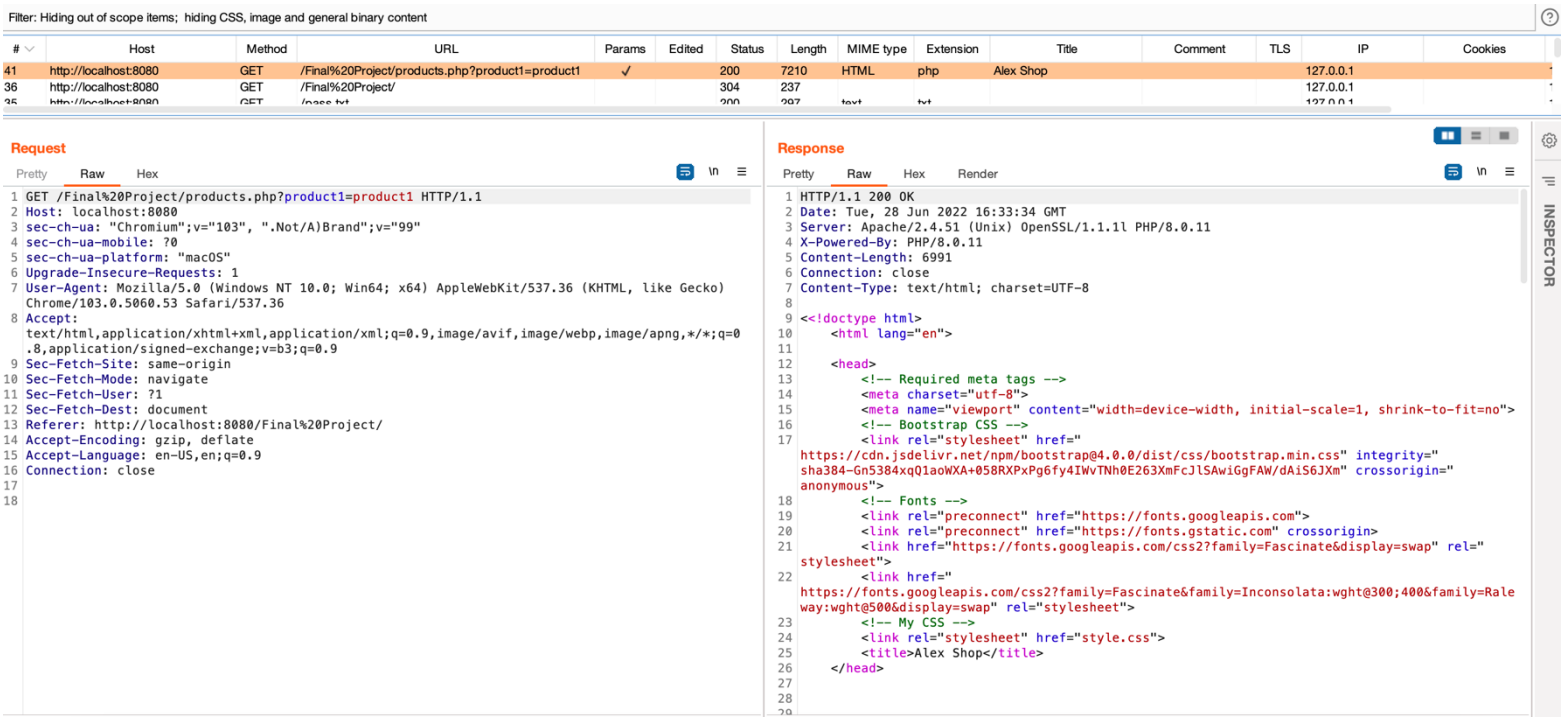
http://localhost:8080/Final%20Project/products.php?product2=../index

This vulnerability is exposed because of the wrongly written PHP function which_product_to_show_wrong() in products.php file. The fucntion receives user input and uses it as an argument in the include(). This allows the attacker to adjust the $GET variable.

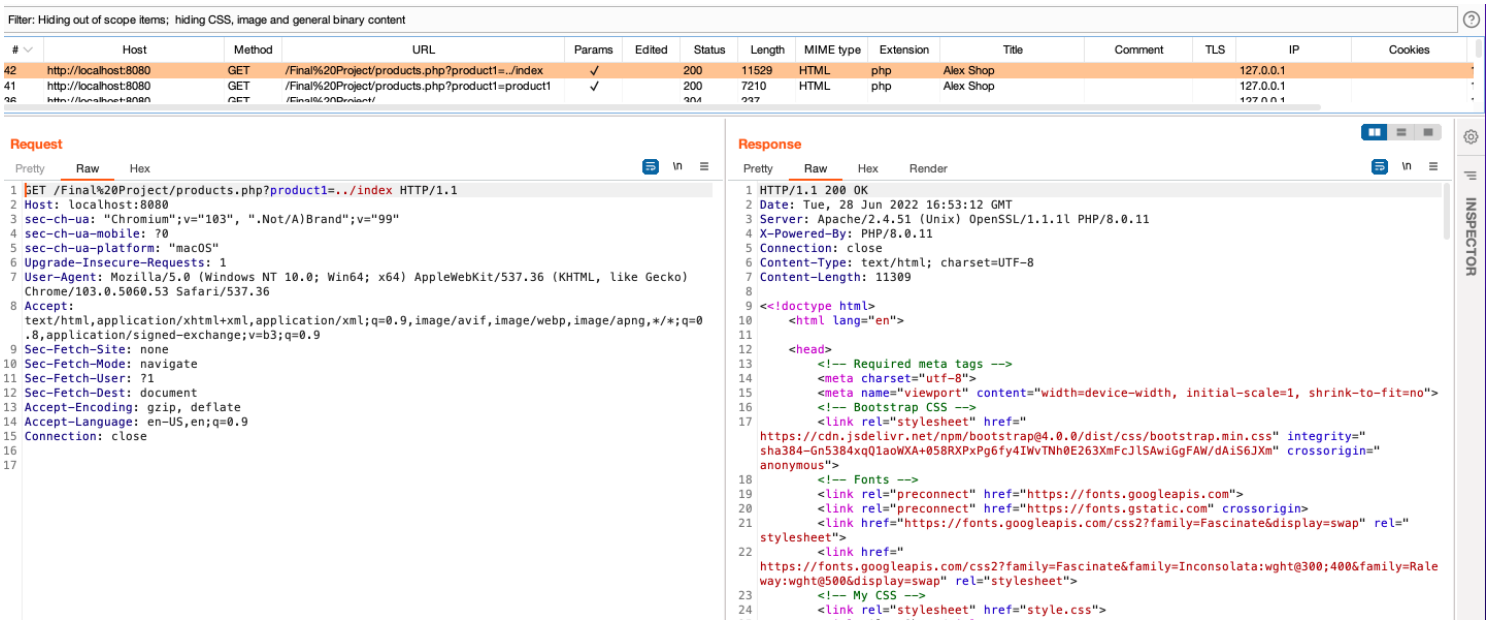
```
function which_product_to_show_wrong()
{
    if (isset($_GET['product1']) || isset($_GET['product2'])) {

        if (isset($_GET['product1'])) {
            include('includes/' . $_GET['product1'] . '.html');
        } elseif (isset($_GET['product2'])) {
            include('includes/' . $_GET['product2'] . '.html');
        }
    }
}
```

Below is the demonstration of how such attack looks in Burp. Burp will catch every package and show us the server's response:



In the above screen capture can be seen a correct request and response. Thus 200 received and the requested page showed. But when we changed the product1 to "../index" we will still receive a 200 response and the logic of the web application remains the same. In the SS below I caught a package with LFI and the server returned 200 as expected even though we used path traversal technique.

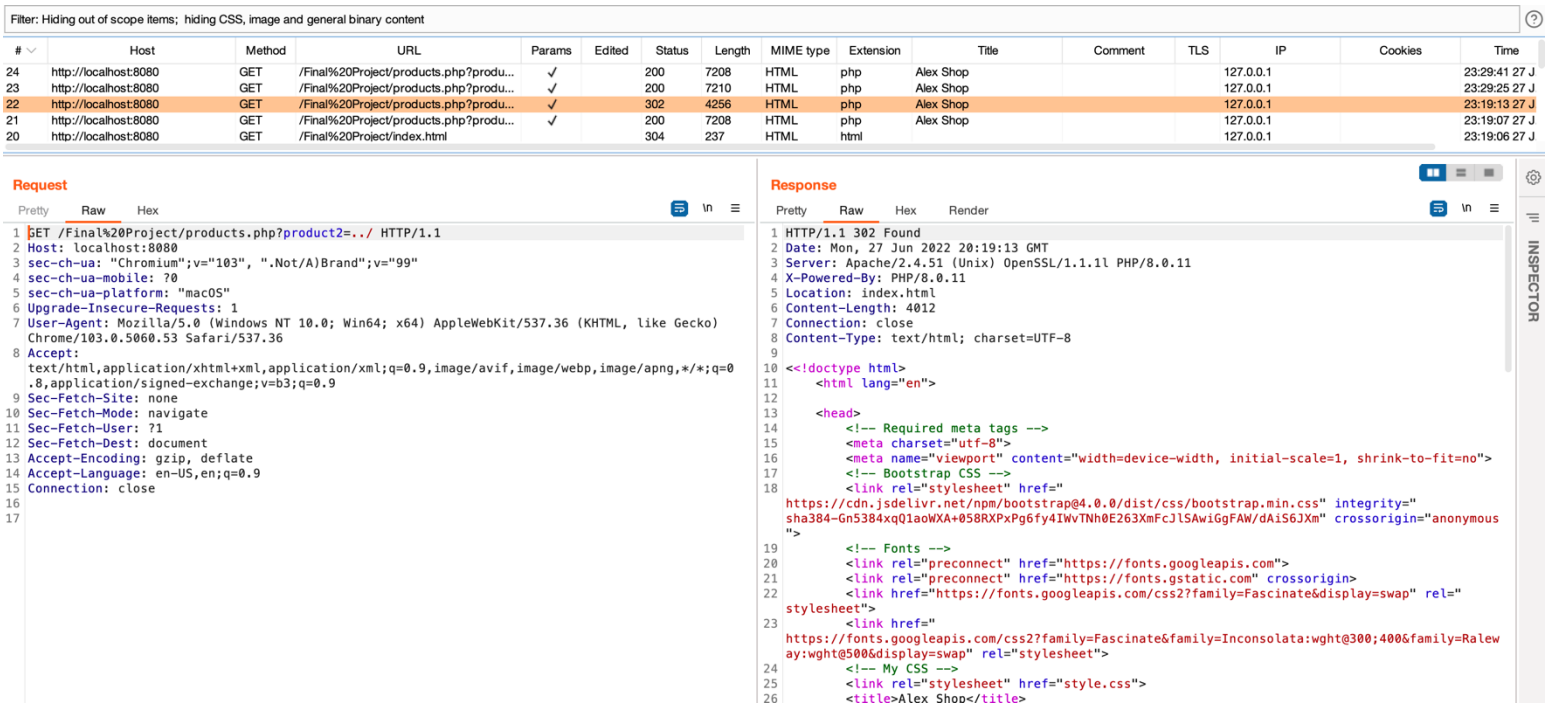


How to prevent Local File Inclusion in Alex Shop?

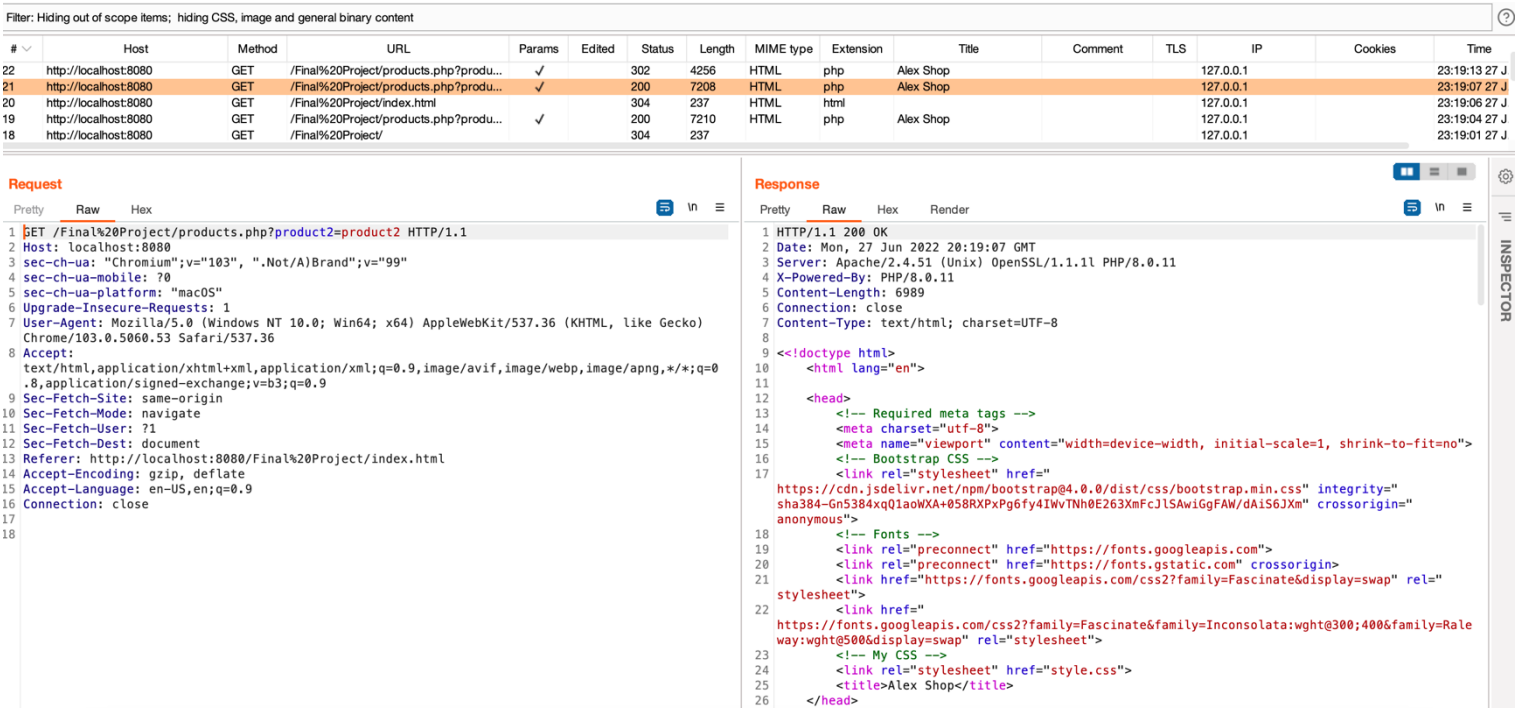
One of the best techniques to avoid LFI vulnerability is to hardcode all the files that theoretically could be included as could be seen in the PHP function which_product_to_show_correct_1() in products.php file. Basically, that means to validate the user input and allow to include files that passed the validation.

```
function which_product_to_show_correct_1()
    {
        if (isset($_GET['product1']) || isset($_GET['product2'])) {
            if (isset($_GET['product1']) and $_GET['product1'] == 'product1') {
                include('includes/' . $_GET['product1'] . '.html');
            } elseif (isset($_GET['product2']) and $_GET['product2'] == 'product2')
{
                include('includes/' . $_GET['product2'] . '.html');
            }
            else{
                header('Location:index.html');
            }
        }
    }
    ?>
```

Below could be found a screen capture from Burp which shows how the web application reacts to LFI when which_product_to_show_correct_1() is used:



As could be seen we changed the input to "../" and received HTTP error code 302 which means that we were redirected. This happens for all the inputs that are not product1 or product2. Below is shown a SS of an acceptable HTTP request which passed the validation and received 200:



Another useful solution to prevent LFI is whitelisting - use verified and secured whitelist files/URLs and ignore everything else. which_product_to_show_correct_2() in products.php file uses this technique by creating an array with all the whitelisted URLs if the current URL in the array it will load the requested page, otherwise redirect to index.html:

```
<?php
        function which_product_to_show_correct_2()
        {
            $available_links = [

'http://192.168.64.2/','http://192.168.64.2/Final%20Project/','http://localhost:8080','http://localhost:8080/Final%20Project/',

'http://localhost:8080/Final%20Project/index.html','http://192.168.64.2/Final%20Project/index.html','http://localhost:8080/Final%20Project/#',
```

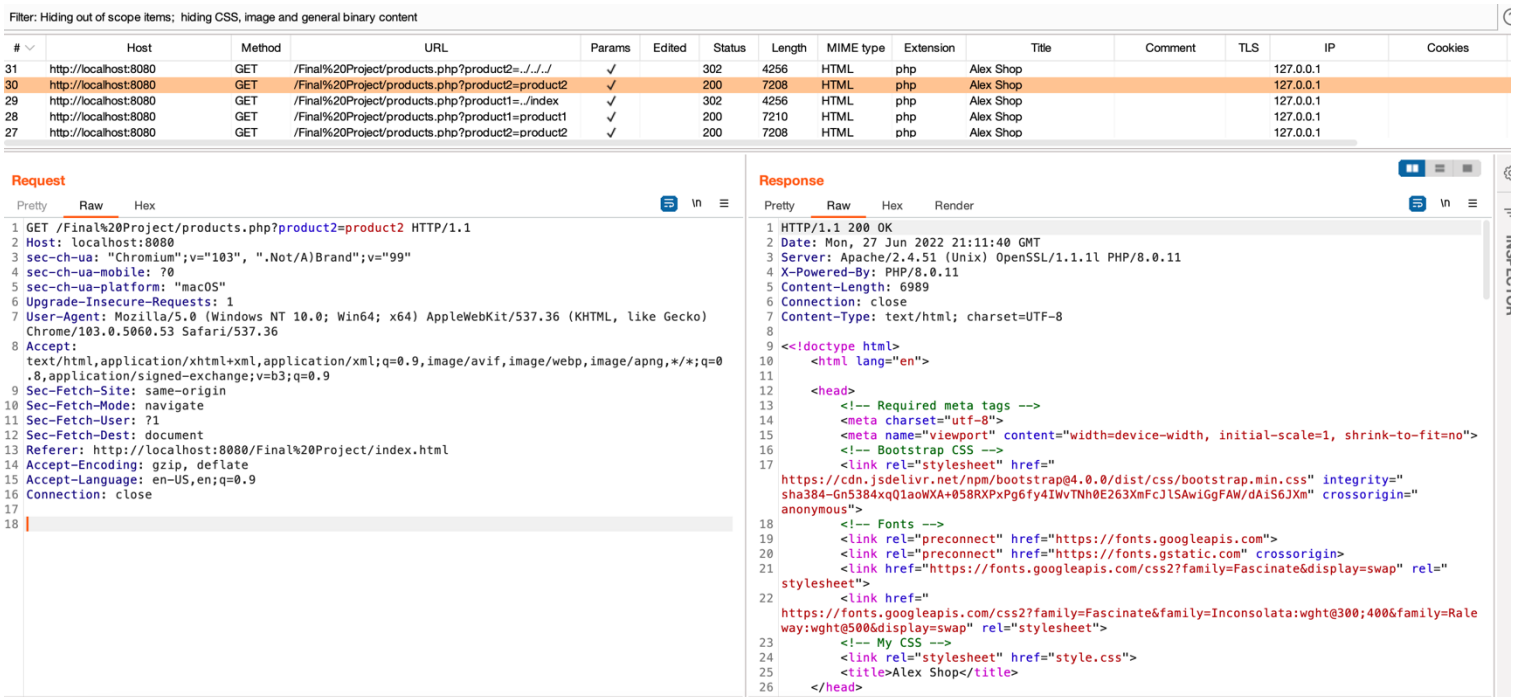
```
'http://192.168.64.2/Final%20Project/#','http://192.168.64.2/Final%20Project/products.php?product1=product1','http://localhost:8080/Final%20Project/products.php?product1=product1',

'http://localhost:8080/Final%20Project/products.php?product2=product2','http://192.168.64.2/Final%20Project/products.php?product2=product2'
            ];

            $actual_link = "http://$_SERVER[HTTP_HOST]$_SERVER[REQUEST_URI]";
            if (in_array($actual_link, $available_links))
            {
                if (isset($_GET['product1']) || isset($_GET['product2'])) {

                    if (isset($_GET['product1'])) {
                        include('includes/' . $_GET['product1'] . '.html');
                    } elseif (isset($_GET['product2'])) {
                        include('includes/' . $_GET['product2'] . '.html');
                    }
                }
            }
            else
            {
              header('Location:index.html');
            }
        }
        ?>
```

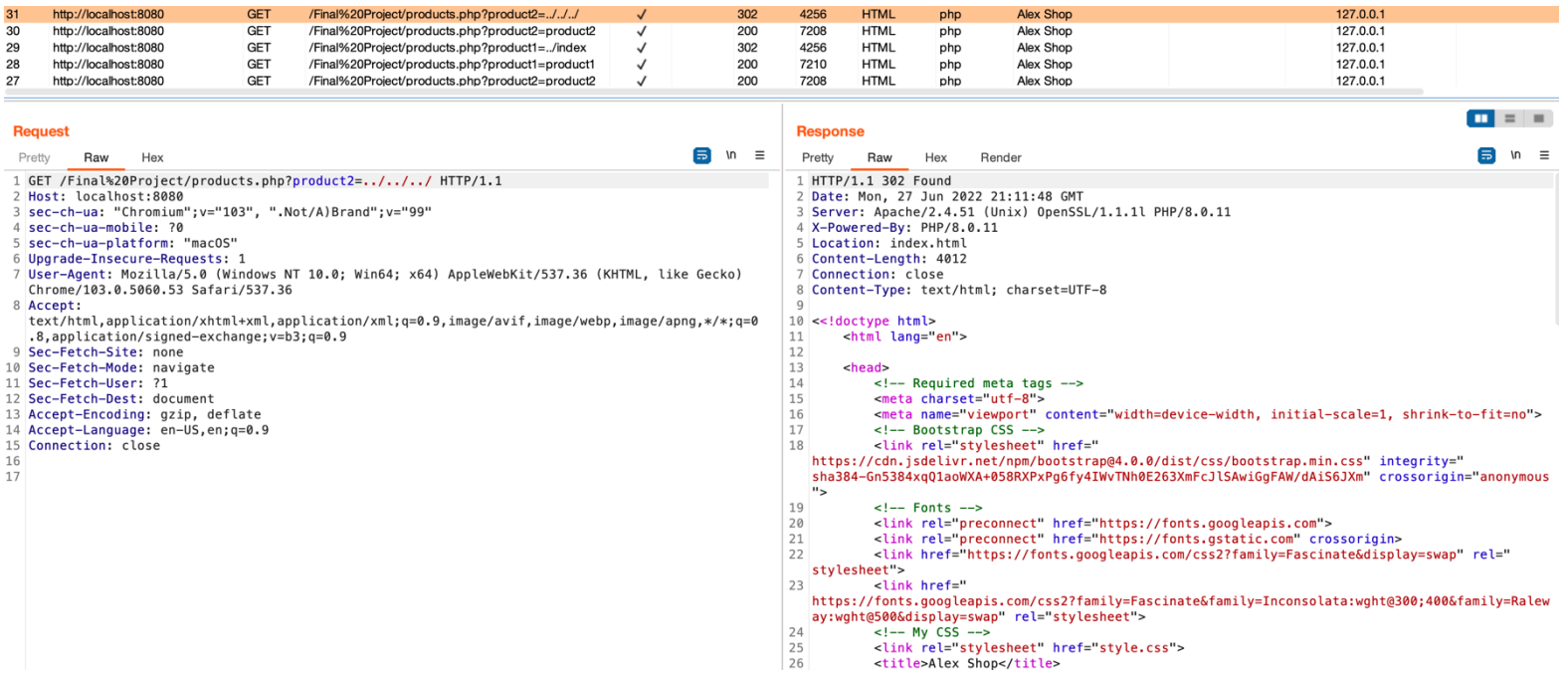
In the SS below, we can see a good request with the right parameters that appears in the whitelisted array of links and thus we received a 200 response.



Filter: Hiding out of scope items; hiding CSS, image and general binary content

#	Host	Method	URL	Params	Edited	Status	Length	MIME type	Extension	Title	Comment	TLS	IP	Cookies
31	http://localhost:8080	GET	/Final%20Project/products.php?product2=../../../	✓		302	4256	HTML	php	Alex Shop			127.0.0.1	
30	http://localhost:8080	GET	/Final%20Project/products.php?product2=product2	✓		200	7208	HTML	php	Alex Shop			127.0.0.1	
29	http://localhost:8080	GET	/Final%20Project/products.php?product1=../index	✓		302	4256	HTML	php	Alex Shop			127.0.0.1	
28	http://localhost:8080	GET	/Final%20Project/products.php?product1=product1	✓		200	7210	HTML	php	Alex Shop			127.0.0.1	
27	http://localhost:8080	GET	/Final%20Project/products.php?product2=product2	✓		200	7208	HTML	php	Alex Shop			127.0.0.1	

But once we try to manipulate the input that is being used latter on in the include() function and replace it with "../../../" we will see a 302 response code – meaning we were redirected. And as can be seen in the SS below we were redirected to index.html page.



31	http://localhost:8080	GET	/Final%20Project/products.php?product2=../../../	✓	302	4256	HTML	php	Alex Shop		127.0.0.1
30	http://localhost:8080	GET	/Final%20Project/products.php?product2=product2	✓	200	7208	HTML	php	Alex Shop		127.0.0.1
29	http://localhost:8080	GET	/Final%20Project/products.php?product1=../index	✓	302	4256	HTML	php	Alex Shop		127.0.0.1
28	http://localhost:8080	GET	/Final%20Project/products.php?product1=product1	✓	200	7210	HTML	php	Alex Shop		127.0.0.1
27	http://localhost:8080	GET	/Final%20Project/products.php?product2=product2	✓	200	7208	HTML	php	Alex Shop		127.0.0.1

Request

Pretty Raw Hex

```
1 GET /Final%20Project/products.php?product2=../../../ HTTP/1.1
2 Host: localhost:8080
3 sec-ch-ua: "Chromium";v="103", ".Not/A)Brand";v="99"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "macOS"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/103.0.5060.53 Safari/537.36
8 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0
  .8,application/signed-exchange;v=b3;q=0.9
9 Sec-Fetch-Site: none
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15 Connection: close
16
17
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 302 Found
2 Date: Mon, 27 Jun 2022 21:11:48 GMT
3 Server: Apache/2.4.51 (Unix) OpenSSL/1.1.1l PHP/8.0.11
4 X-Powered-By: PHP/8.0.11
5 Location: index.html
6 Content-Length: 4012
7 Connection: close
8 Content-Type: text/html; charset=UTF-8
9
10 <!doctype html>
11     <html lang="en">
12
13     <head>
14         <!-- Required meta tags -->
15         <meta charset="utf-8">
16         <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
17         <!-- Bootstrap CSS -->
18         <link rel="stylesheet" href="
   https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css" integrity="
   sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous
   ">
19         <!-- Fonts -->
20         <link rel="preconnect" href="https://fonts.googleapis.com">
21         <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
22         <link href="https://fonts.googleapis.com/css2?family=Fascinate&display=swap" rel="
   stylesheet">
23         <link href="
   https://fonts.googleapis.com/css2?family=Fascinate&family=Inconsolata:wght@300;400&family=Raleway:wght@500&display=swap" rel="stylesheet">
24         <!-- My CSS -->
25         <link rel="stylesheet" href="style.css">
26         <title>Alex Shop</title>
```

Limitations and obstacles

In order to exploit the path traversal technique in web applications where LFI vulnerability was found it is required to use Apache 2.4.49 (CVE-2021-41773) or Apache 2.4.50 (CVE-2021-42013). The Apache version for this project v2.4.51 didn't allow to use path traversal technique and prevented from reaching files/folders outside of the web application folder.

Another technique that is being used in LFI is Null byte injection (%00), however, this issue has been fixed in PHP 5.3.4 (which is already an old and unsupported PHP version).