

Web Development

Overview

In this homework, we will review the many of the concepts and tools covered in the Web Development unit. If needed, refer to the reference sheets provided to you.

- HTTP Reference Sheet
 - curl Reference Sheet
-

Questions:

Before you work through the questions below, please create a new file and record your answers there. This will be your homework deliverable.

HTTP Requests and Responses

Answer the following questions about the HTTP request and response process.

1. What type of architecture does the HTTP request and response process occur in?

[Client_Server based Architecture](#)

2. What are the different parts of an HTTP request?

[Request line](#)

[Headers](#)

[Request body \(optional\)](#)

3. Which part of an HTTP request is optional?

[Request body](#)

4. What are the three parts of an HTTP response?

[Status line](#)

[Headers](#)

[Response body](#)

5. Which number class of status codes represents errors?

400 codes indicate client error

500 codes indicate server errors

6. What are the two most common request methods that a security professional will encounter?

GET

POST

7. Which type of HTTP request method is used for sending data?

Post: Sends data *to* a source, often changing or updating a server.

8. Which part of an HTTP request contains the data being sent to the server?

Request body

9. In which part of an HTTP response does the browser receive the web code to generate and style a web page?

Response body

Using curl

Answer the following questions about curl:

10. What are the advantages of using curl over the browser?¹

Transmit data without user interaction.

A. Ability to manage HTTP Requests / Responses in a Repeatable , Programmatic way.

B. Ability to quickly test HTTP Requests in a way that can be automated.

C. Allows ability to make adjustments as the security professional works.

D. Ability to support numerous protocols even if a UI is not present.

¹https://www.google.com/search?q=What+are+the+advantages+of+using+curl+over+the+browser%3F&sxsrf=AOaemvLZPQbWsOcK3rWclXxV70oIDjV6Uw%3A1639473201313&ei=MWC4YcO-ErLC0PEPoLyeiAo&ved=0ahUKEwiDm6HO-eL0AhUyITQIHSCeB6EQ4dUDCA4&uact=5&oq=What+are+the+advantages+of+using+curl+over+the+browser%3F&gs_lcp=Cgdnd3Mtd2l6EAMyBAgjECcyBggAEBYQHjoHCCMQ6gIQJ0oECEYYAEoECEYYAFDZDFjZDGD7FWgBcAJ4AIABUIgBUJIBATGYAQCgAAQGgAQKwAQrAAQE&sclient=gws-wiz

11. Which curl option is used to change the request method?

`curl --request https://httpbin.org/ (or -X)`

12. Which curl option is used to set request headers?

`curl -v https://httpbin.org/`

13. Which curl option is used to view the response header?

`curl --I https://httpbin.org/`

14. Which request method might an attacker use to figure out which HTTP requests an HTTP server will accept?

`Options, Get, Post, Head, Put, Delete, Connect`

Sessions and Cookies

Recall that HTTP servers need to be able to recognize clients from one another. They do this through sessions and cookies.

Answer the following questions about sessions and cookies:

15. Which response header sends a cookie to the client?

HTTP/1.1 200 OK

Content-type: text/html

Set-Cookie: cart=Bob

[The Set-Cookie HTTP response header](#)

Set-Cookie: cart=Bob

16. Which request header will continue the client's session?

GET /cart HTTP/1.1

Host: www.example.org

Cookie: cart=Bob

[Cookie: cart=Bob](#)

Example HTTP Requests and Responses

Look through the following example HTTP request and response and answer the following questions:

HTTP Request

```
POST /login.php HTTP/1.1
Host: example.com
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 34
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/80.0.3987.132 Mobile Safari/537.36
username=Barbara&password=password
```

17. What is the request method?

POST

Example: POST /login.php HTTP/1.1

18. Which header expresses the client's preference for an encrypted response?

The HTTP Upgrade-Insecure-Requests request header

Example: Upgrade-Insecure-Requests: 1

19. Does the request have a user session associated with it?

No

20. What kind of data is being sent from this request body?

username=Barbara&password=password

HTTP Response

HTTP/1.1 200 OK

Date: Mon, 16 Mar 2020 17:05:43 GMT

Last-Modified: Sat, 01 Feb 2020 00:00:00 GMT

Content-Encoding: gzip

Expires: Fri, 01 May 2020 00:00:00 GMT

Server: Apache

Set-Cookie: SessionID=5

Content-Type: text/html; charset=UTF-8

Strict-Transport-Security: max-age=31536000; includeSubDomains

X-Content-Type: NoSniff

X-Frame-Options: DENY

X-XSS-Protection: 1; mode=block

[page content]

21. What is the response status code?

The response code is 200. The 200 codes indicate success.

22. What web server is handling this HTTP response?

The web server handling the response is: Apache.

23. Does this response have a user session associated to it?

This response does have a user session associated with it.

Example:Set-Cookie: SessionID=5

24. What kind of content is likely to be in the [page content] response body?

log on page

25. If your class covered security headers, what security request headers have been included?

Strict-Transport-Security: max-age=31536000; includeSubDomains

X-Content-Type: NoSniff

X-Frame-Options: DENY

X-XSS-Protection: 1; mode=block

Monoliths and Microservices

Answer the following questions about monoliths and microservices:

26. What are the individual components of microservices called?

Front-end HTML server

Back-end application server

Database

27. What is a service that writes to a database and communicates to other services?

API

28. What type of underlying technology allows for microservices to become scalable and have redundancy?

Docker Containers allow microservices to become scalable and redundant. Containers aren't resource heavy and are easy to deploy. The fact that they are separated from each other allows one to continue working if another container goes down.²

Deploying and Testing a Container Set

Answer the following questions about multi-container deployment:

29. What tool can be used to deploy multiple containers at once?

Docker Compose is a tool for defining and running multi-container Docker applications. In Compose, you use a YAML file to configure your application's services. Then, you create and start all the services from your configuration by running a single command.³

2

https://search.yahoo.com/search;_ylt=Aw9lmGzD7lh60oArE1XNyoA;_ylc=X1MDMjc2NjY3OQRfcgMyBGZyA21jYWZlZQRmcjIDc2ltdG9wBGdwcmlkA2l0QTZRdnBpVFZtanVidHI3VHJXdeEEbl9yc2x0AzAEbl9zdWdnAzAEb3JpZ2luA3NIYXJjaC55YWVhby5jb20EcG9zAzaEcHFzdHIDBHBxc3RybAMwBHFzdHJsAzk5BHF1ZXJ5A1doYXQIMjB0eXBjITlw2YIMjB1bmRlcmx5aW5nJTlwdGVjaG5vbG9neSUyMGFsbG93cyUyMGZvciUyMG1pY3Jvc2VydmljZXMIIMjB0byUyMGJlY29tZSUyMHNjYWxhYmxlJTlwYW5kJTlwaGF2ZSUyMHJlZHVuZGFuY3kiM0YEdF9zdG1wAzE2Mzk1MTgxNzQ-?p=What+type+of+underlying+technology+all+ows+for+microservices+to+become+scalable+and+have+redundancy%3F&fr2=sb-top&fr=mcafee&type=E211US105G0

3

https://www.google.com/search?q=What+tool+can+be+used+to+deploy+multiple+containers+at+once%3F&sxsrf=AOaemvL-fbPFVCQnPUdRDKdTgK1xWrRDQw%3A1639531884283&source=hp&ei=bEW5YZTMDpOv0PEPmpiy0Aw&iflsig=ALs-wAMAAAAAYbITfHxGMnvQHY--H_0mK8A786GwqqOU&ved=0ahUKEwiUsruc1OT0AhWTFzQIHRqMDMoQ4dUDCAk&uact=5&oq=What+tool+can+be+used+to+deploy+multip

30. What kind of file format is required for us to deploy a container set?

docker-compose YAML file

A YAML- file provides a concise format for specifying the instance settings. You learn how to: Configure a YAML file. Deploy the container group.

Databases

31. Which type of SQL query would we use to see all of the information within a table called customers?

The SQL SELECT Statement

Example: SELECT * FROM *table_name*;

32. Which type of SQL query would we use to enter new data into a table? (You don't need a full query, just the first part of the statement.)

The SQL INSERT INTO statement

Example: INSERT INTO *table_name*

33. Why would we never run DELETE FROM <table-name>; by itself?

Be careful when **deleting** records in a table! Notice the **WHERE** clause in the **DELETE** statement. The **WHERE** clause specifies which record(s) should be deleted. If you omit the **WHERE** clause, all records in the table will be deleted!⁴

Bonus Challenge Overview: The Cookie Jar

For this challenge, you'll once again be using curl, but this time to manage and swap sessions.

⚠ **Heads Up:** You'll need to have WordPress set up from the Swapping Sessions activity from Day 1 of this unit. If you have not done it or it is improperly set up, please refer to the Day 1 student guide and the Swapping Sessions activity file.

If you recall, on Day 1 of this unit you used Google Chrome's Cookie-Editor extension to swap sessions and cookies. For this homework challenge, we'll be using the command-line tool curl to practice swapping cookies and sessions within the WordPress app.

It is important for cybersecurity professionals to know how to manage cookies with curl:

le+containers+at+once%3F&gs_lcp=Cgndnd3Mtd2l6EAMyBAGAEb4yBQgAEIYDUABYAGCzBmgAcAB4AIABUlgBUJIBATGYAQCGAQKgAQE&scient=gws-wiz

⁴ https://www.w3schools.com/sql/sql_delete.asp

- Web application security engineers need to regularly ensure cookies are both functional and safe from tampering.
 - For example, you might need to request a cookie from a webpage and then test various HTTP responses using that cookie. Doing this over and over through the browser is tedious, but can be automated with scripts.
- The same concept applies for penetration testers and hackers: curl is used to quickly save a cookie in order to test various exploits.
 - For example, an HTTP server may be configured so that, in order to POST data to specific pages, clients need to have cookies or authentication information set in their request headers, which the server will verify.

Revisiting curl

Recall that you used curl to craft different kinds of requests for your curl activity, and that you saw how to use the Chrome extension Cookie-Editor to export and import cookies and swap sessions.

There will be many systems in which you will need to test requests and cookies that will not connect to a browser or browser extension.

curl not only allows users to look through headers, send data, and authenticate to servers, but also to save and send cookies through two curl options: --cookie-jar and --cookie.

These two options work exactly like Cookie-Editor, but on the command line.

- --cookie-jar allows a curl user to save the cookies set within a response header into a text file.
- --cookie allows a user to specify a text file where a cookie is saved, in order to send a request with the cookies embedded in the request header.

Let's look at how we can create a curl command that will log into a web page with a supplied username and password, and also save the server's response that should contain a cookie.

Logging In and Saving Cookies with Curl

If we want to use the curl command to log into an account, Amanda, with the password password, we use the following curl options:

- `curl --cookie-jar ./amandacookies.txt --form "log=Amanda" --form "pwd=password" http://localhost:8080/wp-login.php --verbose`
- `curl`: The tool that we are using.
- `--cookie-jar`: Specifies where we will save the cookies.
- `./amandacookies.txt`: Location and file where the cookies will be saved.
- `--form`: Lets us pick the login username and password forms that we set in our user info earlier. In this case it's our username.
- `log=Amanda`: How WordPress understands and accepts usernames.
- `--form`: Lets us pick the login username and password forms that we set in our user info earlier. In this case it's our password.
- `pwd=password`: How WordPress understands and accepts passwords.
- `http://localhost:8080/wp-login.php`: Our WordPress login page.
- `--verbose`: Outputs more specific description about the actions the command is taking.

Run the command: `curl --cookie-jar ./amandacookies.txt --form "log=Amanda" --form "pwd=password" http://localhost:8080/wp-login.php --verbose`

If the site confirms our credentials, it will give us a cookie in return, which curl will save in the cookie jar file `./amandacookies.txt`.

Now let's look at how to use that saved cookie on a page that requires us to be logged in.

Using a Saved Cookie

To use a saved cookie, we use the following curl syntax:

- `curl --cookie ./amandacookies.txt http://localhost:8080/wp-admin/users.php`
 - `curl`: The tool that we are using.
 - `--cookie`: Precedes the location of our saved cookie that we want to use.

- `./amandacookies.txt`: Location and file where the cookies are saved.
- `http://localhost:8080/wp-admin/users.php`: A page that requires authentication to see properly. Note that we are not going to the login page, because supplying a cookie in this instance assumes that we are already logged in.

Now that we know how to use the curl cookie jar, let's look at what we need to do for this challenge.

Bonus Challenge Instructions: The Cookie Jar

First, using Docker Compose, navigate to the Day 1 WordPress activity directory and bring up the container set:

- `/home/sysadmin/Documents/docker_files`

Using curl, you will do the following for the Ryan user:

- Log into WordPress and save the user's cookies to a cookie jar.
- Test a WordPress page by using a cookie from the cookie jar.
- Pipe the output from the cookie with grep to check for authenticated page access.
- Attempt to access a privileged WordPress admin page.

Step 1: Set Up

Create two new users: Amanda and Ryan.

1. Navigate to `localhost:8080/wp-admin/`
2. On the left-hand toolbar, hover over **Users** and click **Add New**.
3. Enter the following information to create the new user named Amanda.
 - Username: Amanda
 - Email: `amanda@email.com`

4. Skip down to password:
 - Password: password
 - Confirm Password: Check the box to confirm use of weak password.
 - Role: Administrator
5. Create another user named Ryan.
 - Username: Ryan
 - Email: ryan@email.com
6. Skip down to password:
 - Password: 123456
 - Confirm Password: Check the box to confirm use of weak password.
 - Role: Editor
7. Log out and log in with the following credentials:
 - Username: Amanda
 - Password: password

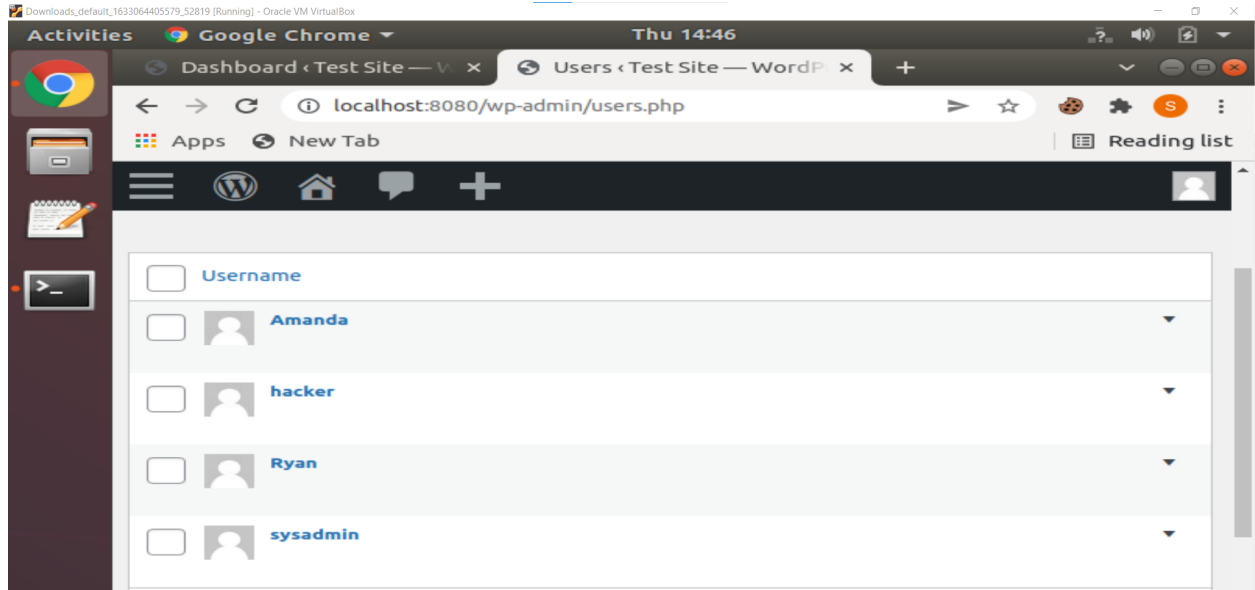
Step 2: Baselineing

For these "baselineing" steps, you'll want to log into two different types of accounts to see how the WordPress site looks at the localhost:8080/wp-admin/users.php page. We want to see how the Users page looks from the perspective of an administrator, vs. a regular user.

1. Using your browser, log into your WordPress site as your sysadmin account and navigate to localhost:8080/wp-admin/users.php, where we previously created the user Ryan. Examine this page briefly. Log out.

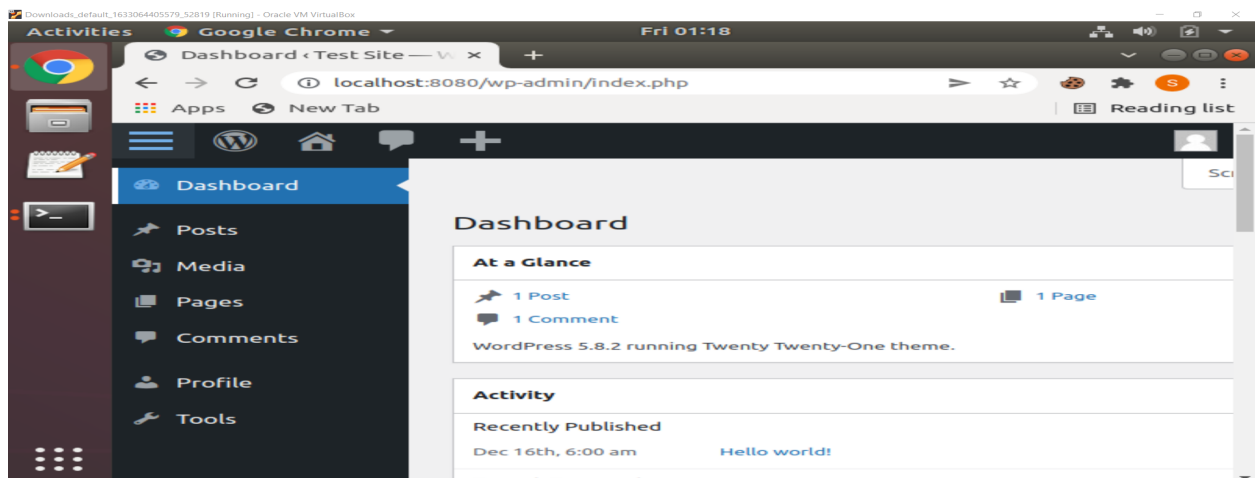
localhost:8080/wp-admin/users.php

<-----(admin can see users)



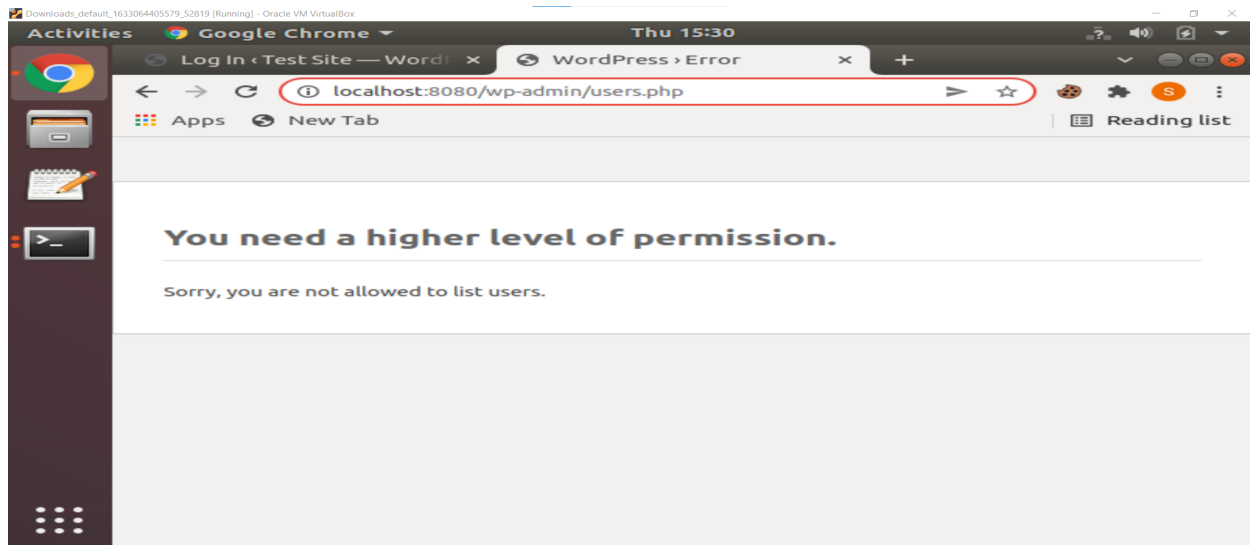
2. Using your browser, log into your Ryan account and attempt to navigate to localhost:8080/wp-admin/index.php. Note the wording on your Dashboard.

<localhost:8080/wp-admin/index.php> <----- (Ryan has no users tab under Dashboard)



3. Attempt to navigate to localhost:8080/wp-admin/users.php. Note what you see now.

<localhost:8080/wp-admin/users.php> <----- (Ryan is not allowed to list users)



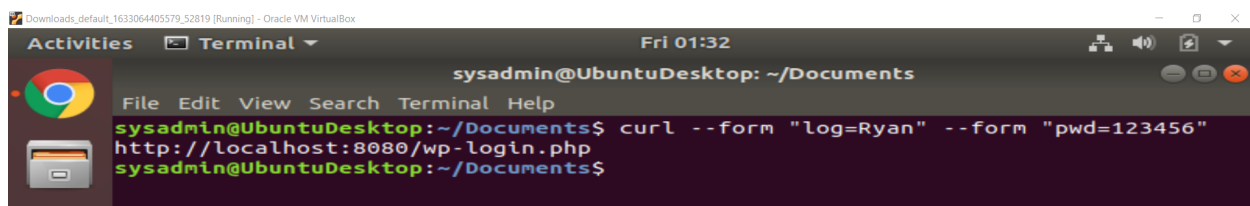
Log out in the browser.

Step 3: Using Forms and a Cookie Jar

Navigate to ~/Documents in a terminal to save your cookies.

1. Construct a curl request that enters two forms: "log={username}" and "pwd={password}" and goes to <http://localhost:8080/wp-login.php>. Enter Ryan's credentials where there are placeholders.

```
curl - -form "log=Ryan" - -form "pwd=123456" http://localhost:8080/wp-login.php
```

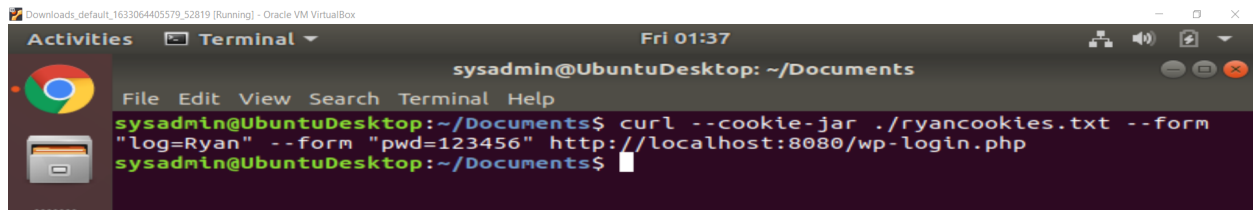


-Question: Did you see any obvious confirmation of a login? (Y/N)

No

2. Construct the same curl request, but this time add the option and path to save your cookie: --cookie-jar ./ryancookies.txt. This option tells curl to save the cookies to the ryancookies.txt text file.

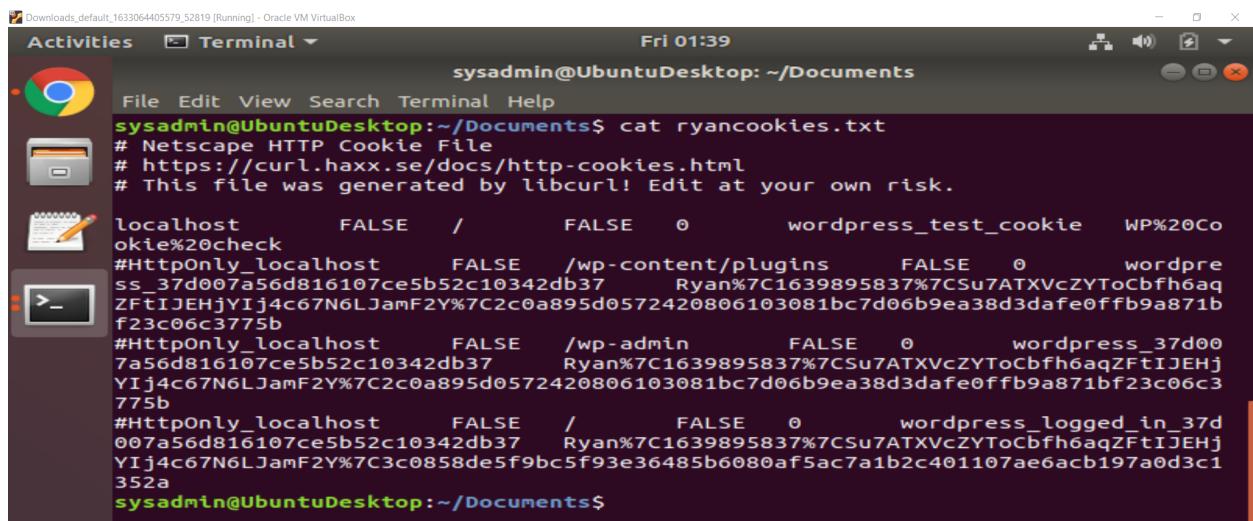
`curl --cookie-jar ./ryancookies.txt --form "log=Ryan" --form "pwd=123456" http://localhost:8080/wp-login.php`

A terminal window titled 'sysadmin@UbuntuDesktop: ~/Documents' showing the execution of a curl command. The command is: `curl --cookie-jar ./ryancookies.txt --form "log=Ryan" --form "pwd=123456" http://localhost:8080/wp-login.php`. The terminal output shows the command being executed and the prompt returning to the user.

```
sysadmin@UbuntuDesktop: ~/Documents
sysadmin@UbuntuDesktop:~/Documents$ curl --cookie-jar ./ryancookies.txt --form
"log=Ryan" --form "pwd=123456" http://localhost:8080/wp-login.php
sysadmin@UbuntuDesktop:~/Documents$
```

3. Read the contents of the ryancookies.txt file.

`cat ryancookies.txt`

A terminal window titled 'sysadmin@UbuntuDesktop: ~/Documents' showing the output of the 'cat ryancookies.txt' command. The output displays three cookies generated by libcurl, each with its domain, path, and expiration date. The cookies are for 'wordpress_test_cookie', 'wordpress_37d007a56d816107ce5b52c10342db37', and 'wordpress_logged_in_37d007a56d816107ce5b52c10342db37'.

```
sysadmin@UbuntuDesktop: ~/Documents
sysadmin@UbuntuDesktop:~/Documents$ cat ryancookies.txt
# Netscape HTTP Cookie File
# https://curl.haxx.se/docs/http-cookies.html
# This file was generated by libcurl! Edit at your own risk.

localhost      FALSE /      FALSE 0      wordpress_test_cookie WP%20Co
okie%20check
#HttpOnly_localhost FALSE /wp-content/plugins FALSE 0      wordpre
ss_37d007a56d816107ce5b52c10342db37 Ryan%7C1639895837%7CSu7ATXVcZYToCbfh6aq
ZFtIJEHjYIj4c67N6LJamF2Y%7C2c0a895d0572420806103081bc7d06b9ea38d3dafe0ffb9a871b
f23c06c3775b
#HttpOnly_localhost FALSE /wp-admin FALSE 0      wordpress_37d00
7a56d816107ce5b52c10342db37 Ryan%7C1639895837%7CSu7ATXVcZYToCbfh6aqZFtIJEHj
YIj4c67N6LJamF2Y%7C2c0a895d0572420806103081bc7d06b9ea38d3dafe0ffb9a871b
f23c06c3775b
#HttpOnly_localhost FALSE / FALSE 0      wordpress_logged_in_37d
007a56d816107ce5b52c10342db37 Ryan%7C1639895837%7CSu7ATXVcZYToCbfh6aqZFtIJEHj
YIj4c67N6LJamF2Y%7C3c0858de5f9bc5f93e36485b6080af5ac7a1b2c401107ae6acb197a0d3c1
352a
sysadmin@UbuntuDesktop:~/Documents$
```

Question: How many items exist in this file?

Three

Note that each one of these is a cookie that was granted to Ryan after logging in.

Step 4: Login Using Cookies

1. Craft a new curl command that now uses the --cookie option, followed by the path to your cookies file. For the URL, use http://localhost:8080/wp-admin/index.php.

`curl -b cookie-jar ./ryancookies.txt http://localhost:8080/wp-admin/index.php`

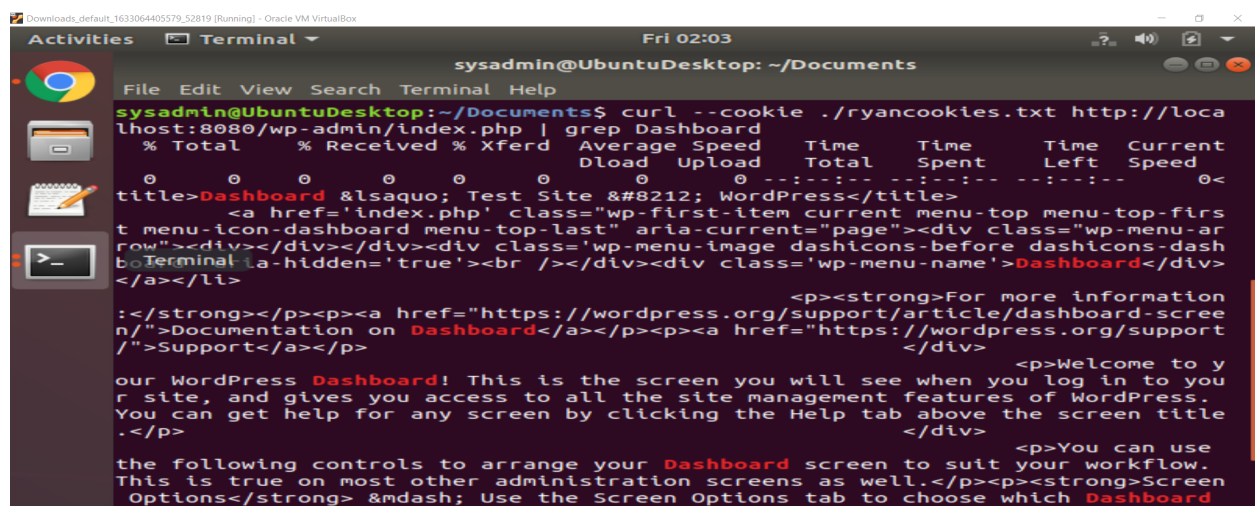
```
sysadmin@UbuntuDesktop:~/Documents$ curl --cookie ./ryancookies.txt http://localhost:8080/wp-admin/index.php
<!DOCTYPE html>
<!--[if IE 8]>
<html xmlns="http://www.w3.org/1999/xhtml" class="ie8 wp-toolbar" lang="en-US">
<![endif]-->
<!--[if !(IE 8) ]><!-->
<html xmlns="http://www.w3.org/1999/xhtml" class="wp-toolbar" lang="en-US">
<!--<![endif]-->
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Dashboard &lsquo; Test Site &#8212; WordPress</title>
<script type="text/javascript">
addLoadEvent = function(func){if(typeof jQuery!="undefined")jQuery(document).ready(func);else if(typeof wpOnload
'function'){wpOnload=func;}else{var oldonload=wpOnload;wpOnload=function(){oldonload();func();}}};
var ajaxurl = '/wp-admin/admin-ajax.php',
    pagenow = 'dashboard',
    typenow = '',
    adminpage = 'index.php',
    thousandsSeparator = ',',
    decimalPoint = '.',
    isRtl = 0;
</script>
<meta name="viewport" content="width=device-width,initial-scale=1.0">
<link rel='dns-prefetch' href='//s.w.org' />
<style type="text/css">
img wp-smiley
```

-Question: Is it obvious that we can access the Dashboard? (Y/N)

Yes

2. Press the up arrow on your keyboard to run the same command, but this time, pipe | grep Dashboard to the end of your command to return all instances of the word Dashboard on the page.

`curl -b cookie-jar ./ryancookies.txt http://localhost:8080/wp-admin/index.php | grep Dashboard`



```
sysadmin@UbuntuDesktop:~/Documents$ curl --cookie ./ryancookies.txt http://localhost:8080/wp-admin/index.php | grep Dashboard
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           % Done    0     0     0    0         0      0      0      0
0         0     0     0    0         0      0      0      0
title>Dashboard &lsquo; Test Site &#8212; WordPress</title>
<a href='index.php' class="wp-first-item current menu-top menu-top-firs
t menu-icon-dashboards menu-top-last" aria-current="page"><div class="wp-menu-image dashicons-before dashicons-dashb
b</div><div class="wp-menu-name">Dashboard</div>
</a></li>
<p><strong>For more information
:</strong></p><p><a href="https://wordpress.org/support/article/dashboard-scre
n/">Documentation on Dashboard</a></p><p><a href="https://wordpress.org/support
/">Support</a></p>
<p>Welcome to y
our WordPress Dashboard! This is the screen you will see when you log in to you
r site, and gives you access to all the site management features of WordPress.
You can get help for any screen by clicking the Help tab above the screen title
.</p>
<p>You can use
the following controls to arrange your Dashboard screen to suit your workflow.
This is true on most other administration screens as well.</p><p><strong>Screen
Options</strong> &mdash; Use the Screen Options tab to choose which Dashboard
```

-Question: Look through the output where the Dashboard is highlighted. Does any of the wording on this page seem familiar? (Y/N) If so, you should be successfully logged in to your Editor's dashboard.

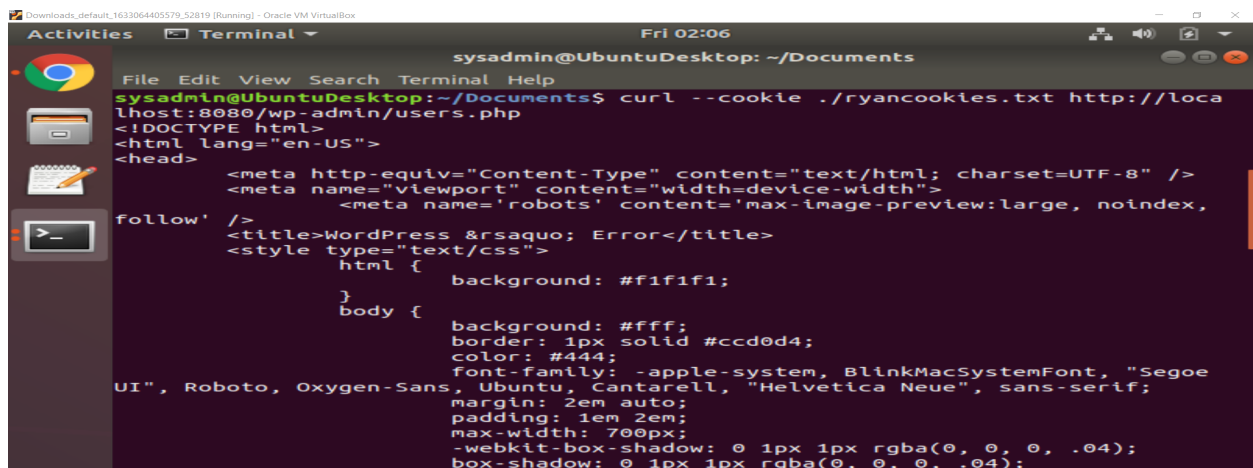
Yes. There is info displayed on the Dashboard

Step 5: Test the Users.php Page

1. Finally, write a curl command using the same --cookie ryancookies.txt option, but attempt to access <http://localhost:8080/wp-admin/users.php>.

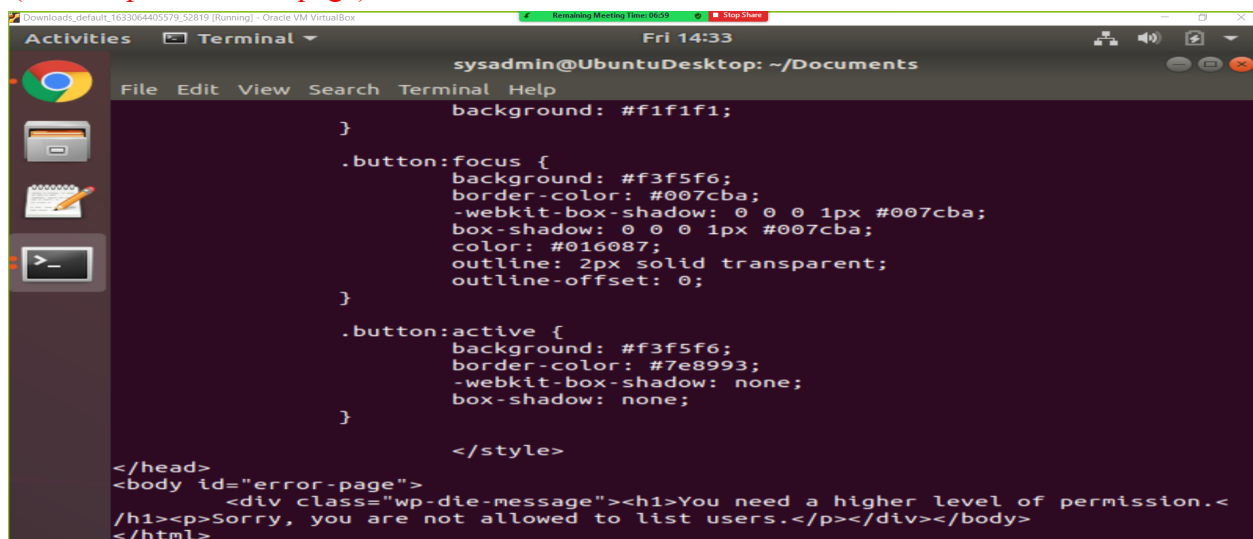
`curl -b ryancookies.txt http://localhost:8080/wp-admin/users.php`

(Top part of result page)



```
sysadmin@UbuntuDesktop: ~/Documents
sysadmin@UbuntuDesktop:~/Documents$ curl --cookie ./ryancookies.txt http://localhost:8080/wp-admin/users.php
<!DOCTYPE html>
<html lang="en-US">
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <meta name="viewport" content="width=device-width">
  <meta name="robots" content="Max-Image-Preview:large, noindex, follow" />
  <title>WordPress &rsquo; Error</title>
  <style type="text/css">
    html {
      background: #f1f1f1;
    }
    body {
      background: #fff;
      border: 1px solid #ccd0d4;
      color: #444;
      font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", Roboto, Oxygen-Sans, Ubuntu, Cantarell, "Helvetica Neue", sans-serif;
      margin: 2em auto;
      padding: 1em 2em;
      max-width: 700px;
      -webkit-box-shadow: 0 1px 1px rgba(0, 0, 0, .04);
      box-shadow: 0 1px 1px rgba(0, 0, 0, .04);
    }
  </style>
</head>
<body id="error-page">
  <div class="wp-die-message"><h1>You need a higher level of permission.</h1><p>Sorry, you are not allowed to list users.</p></div>
</body>
</html>
```

(Bottom part of result page)



```
    }
    .button:focus {
      background: #f3f5f6;
      border-color: #007cba;
      -webkit-box-shadow: 0 0 1px #007cba;
      box-shadow: 0 0 1px #007cba;
      color: #016087;
      outline: 2px solid transparent;
      outline-offset: 0;
    }
    .button:active {
      background: #f3f5f6;
      border-color: #7e8993;
      -webkit-box-shadow: none;
      box-shadow: none;
    }
  </style>
</head>
<body id="error-page">
  <div class="wp-die-message"><h1>You need a higher level of permission.</h1><p>Sorry, you are not allowed to list users.</p></div>
</body>
</html>
```

“Sorry, you are not allowed to list users.”