Week 14 Homework: 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 Architecture

2. **What are the different parts of an HTTP request?**
   Request Line, Request Headers, Request Body

3. **Which part of an HTTP request is optional?**
   Request Body

4. **What are the three parts of an HTTP response?**
   Headers, Status Line, Body (Optional)

5. **Which number class of status codes represents errors?**
   400 Codes

6. **What are the two most common request methods that a security professional will encounter?**
   GET and POST requests

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

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:

What are the advantages of using curl over the browser?

It is a free command line tool which is very flexible and versatile.

Can be utilized to manage and test HTTP Requests/Responses in an automated manner that allows for adjustments on the fly.

Able to support many different protocols even though a UI may not be not available.

It can be used to complete complex tasks such as authentication, HTTP post, SSL connections, proxy support and FTP uploads.

Also users are able to do simple things with curl, such as download web pages and web images.

Which curl option is used to change the request method?

-X

Which curl option is used to set request headers?

-H

Which curl option is used to view the response header?

-I

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

Options

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:

Which response header sends a cookie to the client?

Set-Cookies

```
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: cart=Bob
```

Which request header will continue the client's session?

```
GET /cart HTTP/1.1
Host: www.example.org
Cookie: cart=Bob
'Connection: keep-alive'
```

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
```

What is the request method?

POST

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

Upgrade-Insecure-Requests: 1

Does the request have a user session associated with it?

No, the Session is not established yet. This is evident as there is no “cookie” displayed.

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

Login credentials

Example HTTP Requests and Responses

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

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]
```

What is the response status code?

200

What web server is handling this HTTP response?

Apache

Does this response have a user session associated to it?

SessionID=5

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

Page configuration details

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

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

<u>Monoliths and Microservices</u>

Answer the following questions about monoliths and microservices:

What are the individual components of microservices called?

Clients
Identity Providers
API Gateway
Messaging Formats
Databases
Static Content
Management
Service Discovery

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

An API (Application Programming Interface) is a software intermediary that allows two applications to talk to each other. In other words, an API is the messenger that delivers your request to the server of a provider that you're requesting information from and then delivers the response back to you.

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

Load Balancing. This is the process of distributing network traffic across multiple servers. This ensures no single server bears too much demand. By spreading out the work evenly, load balancing improves application responsiveness and increases availability of applications/websites for users.

Deploying and Testing a Container Set

Answer the following questions about multi-container deployment:

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.

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

YAML configuration file

Databases

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

SELECT * FROM Customers;

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.)

INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);

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

If you leave off the where clause, all records will be deleted!

You should write:

```
delete from "tablename"

where "columnname"
  OPERATOR "value"
[and|or "column"
  OPERATOR "value"];

[ ] = optional
```

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 cookie and sessions within the WordPress app.

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

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.

Navigate to localhost:8080/wp-admin/

On the left-hand toolbar, hover over Users and click Add New.

Enter the following information to create the new user named Amanda.

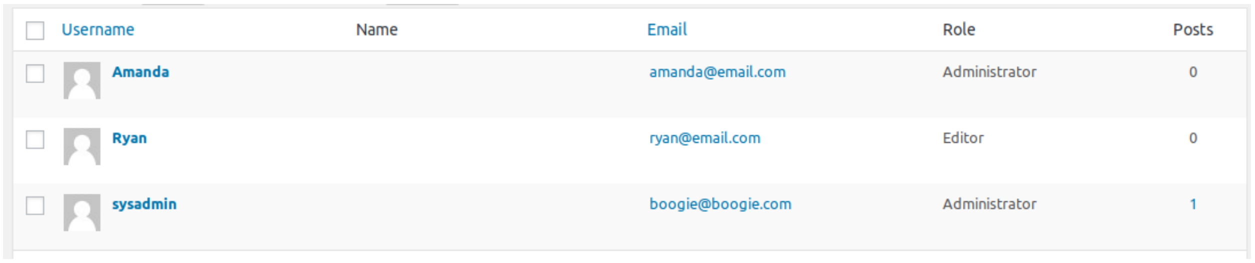
Username: Amanda
Email: amanda@email.com
Skip down to password:

Password: password
Confirm Password: Check the box to confirm use of weak password.
Role: Administrator

Create another user named Ryan.

Username: Ryan
Email: ryan@email.com
Skip down to password:

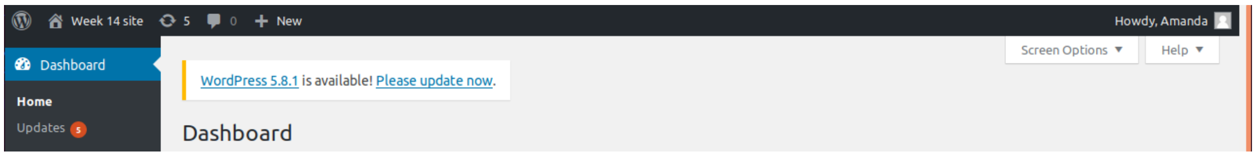
Password: 123456
Confirm Password: Check the box to confirm use of weak password.
Role: Editor



Username	Name	Email	Role	Posts
Amanda		amanda@email.com	Administrator	0
Ryan		ryan@email.com	Editor	0
sysadmin		boogie@boogie.com	Administrator	1

Log out and log in with the following credentials:

Username: Amanda
Password: password

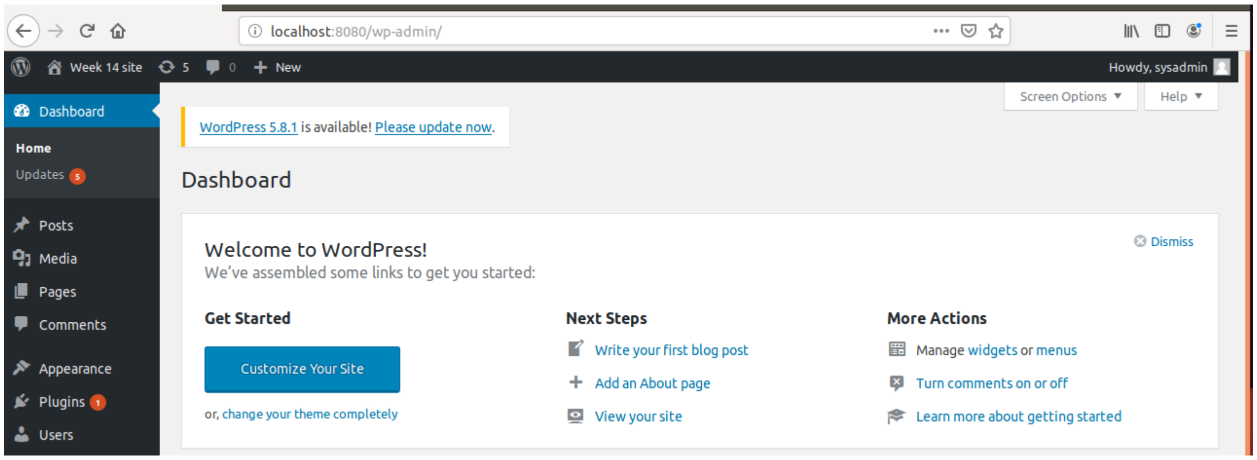


WordPress 5.8.1 is available! Please update now.

Dashboard

Step 2: Baselining

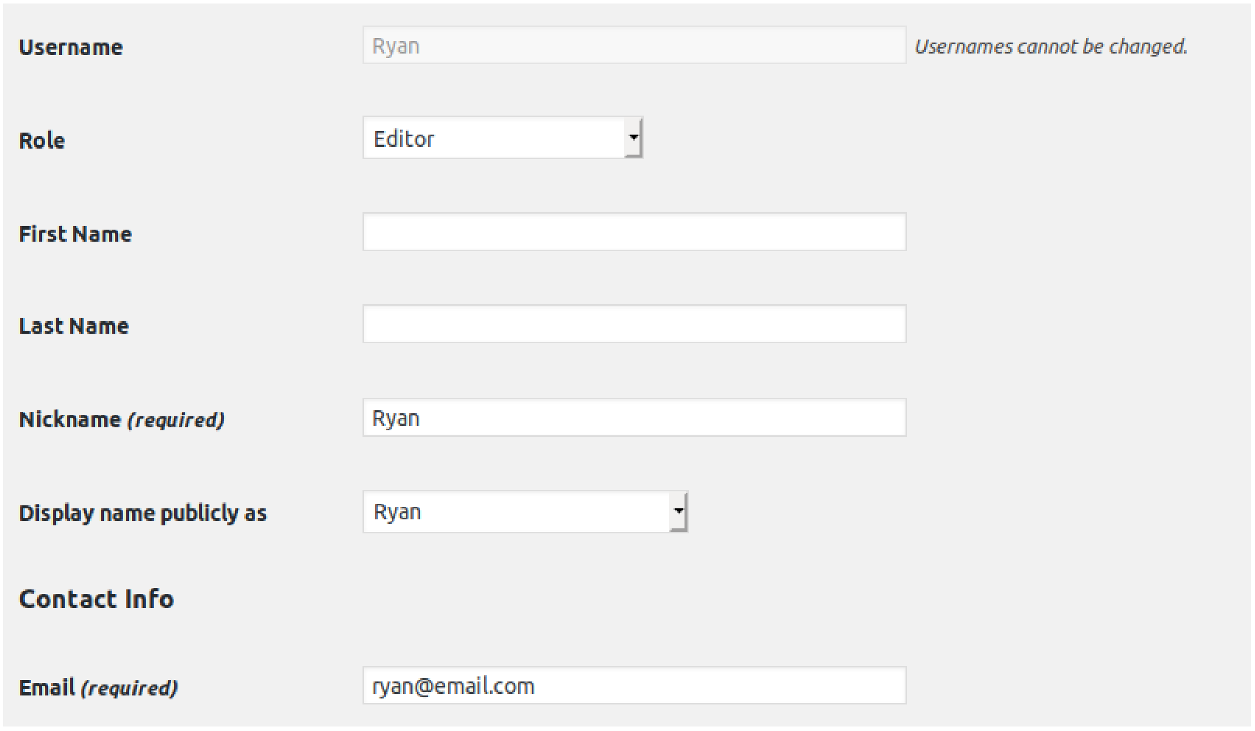
For these "baselining" steps, you'll want to log into two different types of accounts to see how the WordPress site looks at the localhost:8080/wp-admin/users.php page. We want to see how the Users page looks from the perspective of an administrator, vs. a regular user.



WordPress 5.8.1 is available! Please update now.

Welcome to WordPress!
We've assembled some links to get you started:

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.

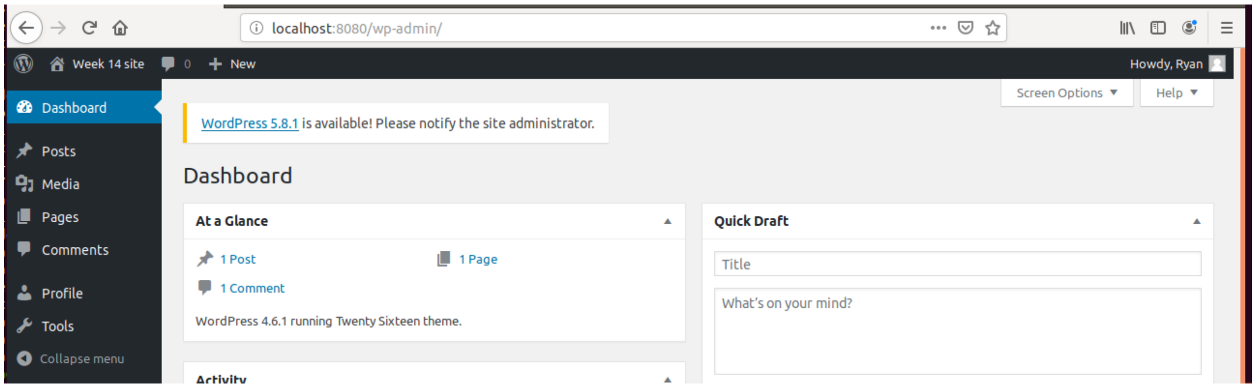


Username	Ryan	Usernames cannot be changed.
Role	Editor	
First Name		
Last Name		
Nickname *(required)*	Ryan	
Display name publicly as	Ryan	

Contact Info

Email *(required)*	ryan@email.com

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.

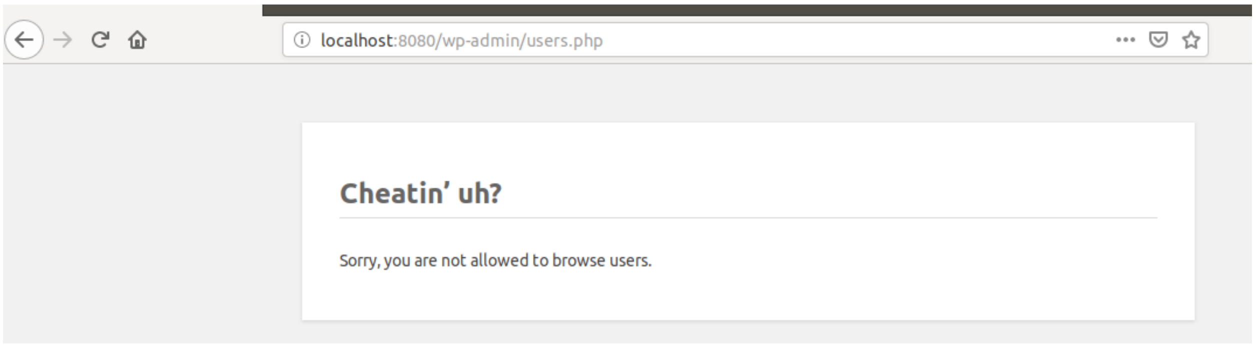


WordPress 5.8.1 is available! Please notify the site administrator.

Dashboard

WordPress 4.6.1 running Twenty Sixteen theme.

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



Cheatin' uh?

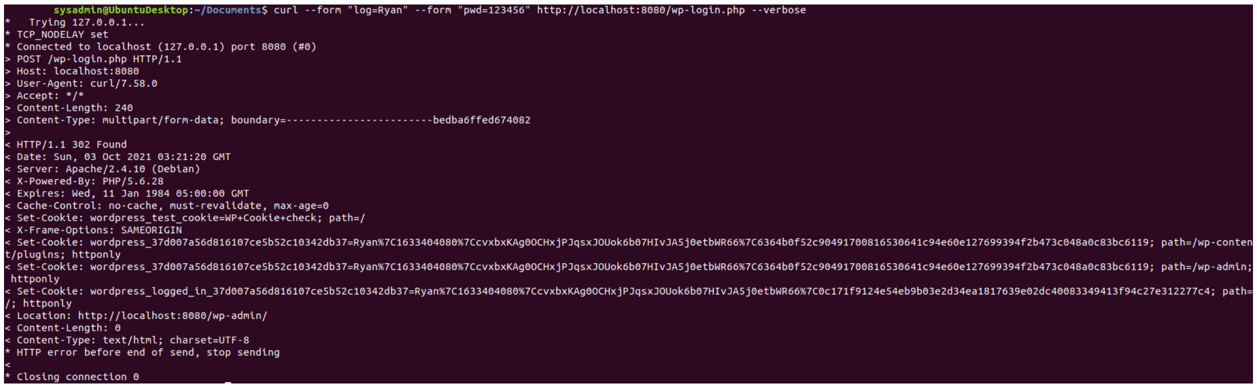
Sorry, you are not allowed to browse users.

Log out in the browser.

Step 3: Using Forms and a Cookie Jar

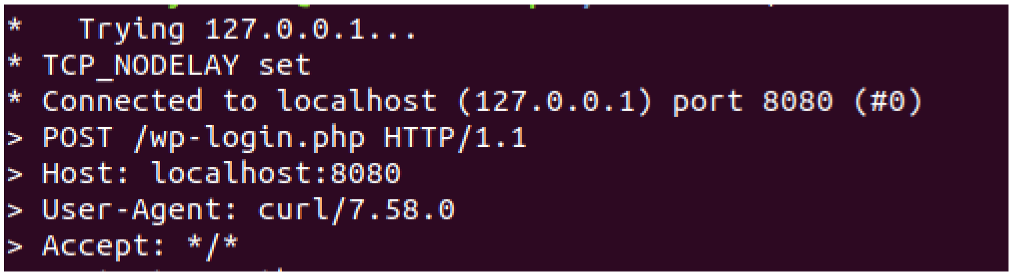
Navigate to ~/Documents in a terminal to save your cookies.

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.



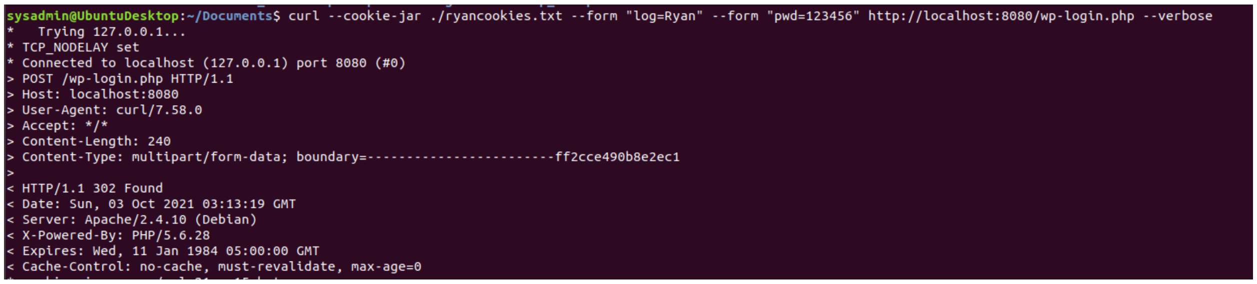
```
sysadmin@UbuntuDesktop:~/Documents$ curl --form "log=Ryan" --form "pwd=123456" http://localhost:8080/wp-login.php --verbose
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /wp-login.php HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.58.0
> Accept: */*
> Content-Length: 240
> Content-Type: multipart/form-data; boundary=------------------------bedba6ffed674082
>
< HTTP/1.1 302 Found
< Date: Sun, 03 Oct 2021 03:21:20 GMT
< Server: Apache/2.4.10 (Debian)
< X-Powered-By: PHP/5.6.28
< Expires: Wed, 11 Jan 1984 05:00:00 GMT
< Cache-Control: no-cache, must-revalidate, max-age=0
< Set-Cookie: wordpress_test_cookie=WP+Cookie+check; path=/
< X-Frame-Options: SAMEORIGIN
< Set-Cookie: wordpress_37d007a56d816107ce5b52c10342db37=Ryan%7C1633404080%7CcvxbxKAg0OCHxjPJqsxJOUok6b07HIvJA5j0etbWR66%7C6364b0f52c9049170081653064 1c94e60e127699394f2b473c048a0c83bc6119; path=/wp-conten
t/plugins; httponly
< Set-Cookie: wordpress_37d007a56d816107ce5b52c10342db37=Ryan%7C1633404080%7CcvxbxKAg0OCHxjPJqsxJOUok6b07HIvJA5j0etbWR66%7C6364b0f52c90491700816530641c94e60e127699394f2b473c048a0c83bc6119; path=/wp-admin;
 httponly
< Set-Cookie: wordpress_logged_in_37d007a56d816107ce5b52c10342db37=Ryan%7C1633404080%7CcvxbxKAg0OCHxjPJqsxJOUok6b07HIvJA5j0etbWR66%7C0c171f9124e54eb9b03e2d34ea1817639e02dc40083349413f94c27e312277c4; path=
/; httponly
< Location: http://localhost:8080/wp-admin/
< Content-Length: 0
< Content-Type: text/html; charset=UTF-8
* HTTP error before end of send, stop sending
<
* Closing connection 0
```

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

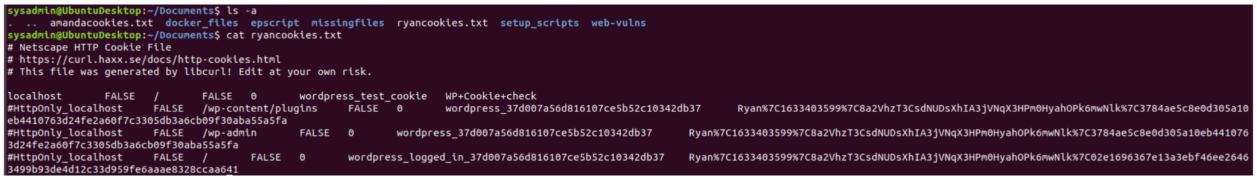


```
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /wp-login.php HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.58.0
> Accept: */*
```

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. Read the contents of the ryancookies.txt file.



```
sysadmin@UbuntuDesktop:~/Documents$ curl --cookie-jar ./ryancookies.txt --form "log=Ryan" --form "pwd=123456" http://localhost:8080/wp-login.php --verbose
*   Trying 127.0.0.1...
* TCP_NODELAY set
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /wp-login.php HTTP/1.1
> Host: localhost:8080
> User-Agent: curl/7.58.0
> Accept: */*
> Content-Length: 240
> Content-Type: multipart/form-data; boundary=------------------------ff2cce490b8e2ec1
>
< HTTP/1.1 302 Found
< Date: Sun, 03 Oct 2021 03:13:19 GMT
< Server: Apache/2.4.10 (Debian)
< X-Powered-By: PHP/5.6.28
< Expires: Wed, 11 Jan 1984 05:00:00 GMT
< Cache-Control: no-cache, must-revalidate, max-age=0
```



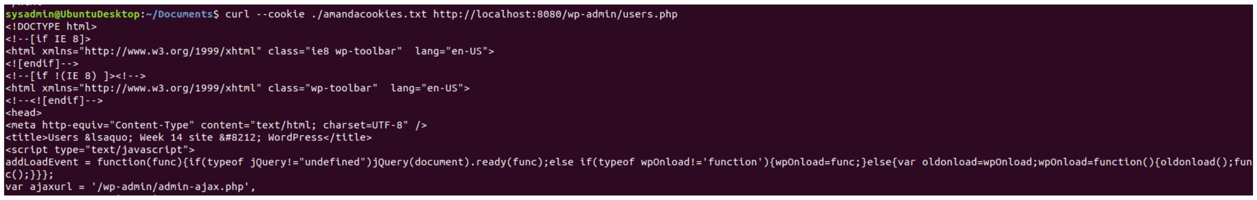
```
sysadmin@UbuntuDesktop:~/Documents$ ls -a
.  ..  amandacookies.txt  docker_files  epscript  missingfiles  ryancookies.txt  setup_scripts  web-vulns
sysadmin@UbuntuDesktop:~/Documents$ cat ryancookies.txt
# Netscape HTTP Cookie File
# https://curl.haxx.se/docs/http-cookies.html
# This file was generated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

localhost       FALSE   /       FALSE   0       wordpress_test_cookie   WP+Cookie+check
#HttpOnly_localhost     FALSE   /wp-content/plugins     FALSE   0       wordpress_37d007a56d816107ce5b52c10342db37      Ryan%7C1633403599%7C8a2VhzT3CsdNUDsXhIA3jVNqX3HPm0HyahOPk6mwNlk%7C3784ae5c8e0d305a10
eb4410763d24fe2a60f7c3305db3a6cb09f30aba55a5fa
#HttpOnly_localhost     FALSE   /wp-admin       FALSE   0       wordpress_37d007a56d816107ce5b52c10342db37      Ryan%7C1633403599%7C8a2VhzT3CsdNUDsXhIA3jVNqX3HPm0HyahOPk6mwNlk%7C3784ae5c8e0d305a10eb441076
3d24fe2a60f7c3305db3a6cb09f30aba55a5fa
#HttpOnly_localhost     FALSE   /       FALSE   0       wordpress_logged_in_37d007a56d816107ce5b52c10342db37    Ryan%7C1633403599%7C8a2VhzT3CsdNUDsXhIA3jVNqX3HPm0HyahOPk6mwNlk%7C02e1696367e13a3ebf46ee2646
3499b93de4d12c33d959fe6aaae8328ccaa641
```

Question: How many items exist in this file? 3

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

Step 4: Log in Using Cookies

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.



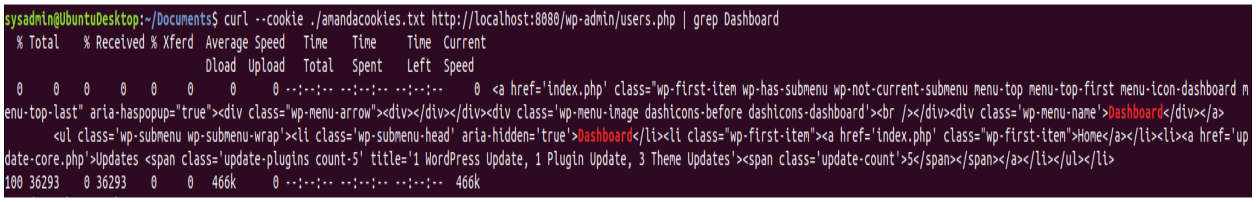
```
sysadmin@UbuntuDesktop:~/Documents$ curl --cookie ./amandacookies.txt http://localhost:8080/wp-admin/users.php
<!DOCTYPE html>
<!--[if IE 8]>
<html xmlns="http://www.w3.org/1999/xhtml" class="ie8 wp-toolbar"  lang="en-US">
<![endif]-->
<!--[if !(IE 8) ]><!-->
<html xmlns="http://www.w3.org/1999/xhtml" class="wp-toolbar"  lang="en-US">
<!--<![endif]-->
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Users &lsaquo; Week 14 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();fun
c();}}};
var ajaxurl = '/wp-admin/admin-ajax.php',
```

Question: Is it obvious that we can access the Dashboard? (Y/N) No

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.

Question: Look through the output where Dashboard is highlighted. Does any of the wording on this page seem familiar? (Y/N) Yes

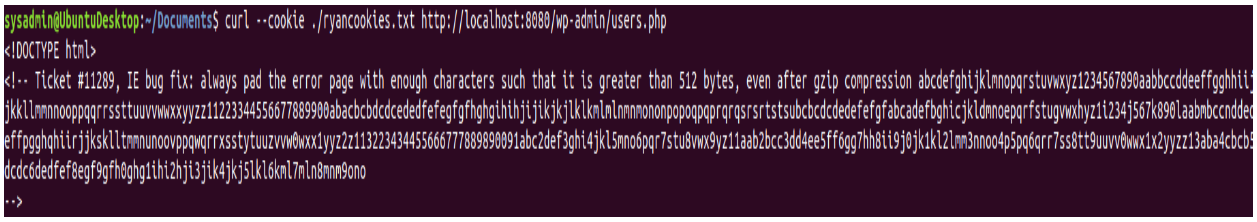
If so, you should be successfully logged in to your Editor's dashboard.



```
sysadmin@UbuntuDesktop:~/Documents$ curl --cookie ./amandacookies.txt http://localhost:8080/wp-admin/users.php | grep Dashboard
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0  <a href='index.php' class="wp-first-item wp-has-submenu wp-not-current-submenu menu-top menu-top-first menu-icon-dashboard m
enu-top-last" aria-haspopup="true"><div class="wp-menu-arrow"><div></div></div><div class='wp-menu-image dashicons-before dashicons-dashboard'><br /></div><div class='wp-menu-name'>Dashboard</div></a>
        <ul class='wp-submenu wp-submenu-wrap'><li class='wp-submenu-head' aria-hidden='true'>Dashboard</li><li class="wp-first-item"><a href='index.php' class="wp-first-item">Home</a></li><li><a href='up
date-core.php'>Updates <span class='update-plugins count-5' title='1 WordPress Update, 1 Plugin Update, 3 Theme Updates'><span class='update-count'>5</span></span></a></li></ul></li>
100 36293    0 36293    0     0   466k      0 --:--:-- --:--:-- --:--:--  466k
```

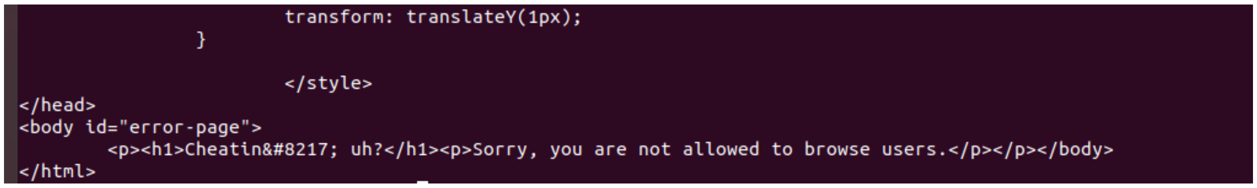
Step 5: Test the Users.php Page

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



```
sysadmin@UbuntuDesktop:~/Documents$ curl --cookie ./ryancookies.txt http://localhost:8080/wp-admin/users.php
<!DOCTYPE html>
<!-- Ticket #11289, IE bug fix: always pad the error page with enough characters such that it is greater than 512 bytes, even after gzip compression abcdefghijklmnopqrstuvwxyz1234567890aabbccddeeffgghhiij
jkkllmmnnooppqqrrssttuuvvwwxxyyzz11223344556677889900abacbcbdcdcededfefegfgfhghgihihjijikjkjlklkmlmlnmnmononpopoqpqprqrqsrsrtstsubcbcdcdedefefgfabcadefbghicjkldmnoepqrfstugvwxhyz1i234j567k890laabmbccnddeo
effpgghqhiirjjksklltmmnunoovppqwqrrxsstytuuzvvw0wxx1yyz2z113223434455666777889890091abc2def3ghi4jkl5mno6pqr7stu8vwx9yz11aab2bcc3dd4ee5ff6gg7hh8ii9j0jk1kl2lmm3nnoo4p5pq6qrr7ss8tt9uuvv0wwx1x2yyzz13aba4cbcb5
dcdc6dedfef8egf9gfh0ghg1ihi2hji3jik4jkj5lkl6kml7mln8mnm9ono
-->
```

Question: What happens this time?



```
                        transform: translateY(1px);
                }

                        </style>
</head>
<body id="error-page">
        <p><h1>Cheatin&#8217; uh?</h1><p>Sorry, you are not allowed to browse users.</p></p></body>
</html>
```