

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, Header, Request Body(optional)

3. Which part of an HTTP request is optional?

Request Body(optional)

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

Status line, response headers, response body

5. Which number class of status codes represents errors?

400-499 are client errors and 500-599 indicate server errors

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

Get and Post

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

Post

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?

Allows for use through a container that has no user interface and be able to quickly test HTTP requests in a way that can be automated, but also allows them to make adjustments

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

-X --request

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

-H --header

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

-i --include

14. 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?

HTTP/1.1 200 OK

Content-type: text/html

15. Set-Cookie: cart=Bob

Answer: Set-Cookie

Which request header will continue the client's session?

GET /cart HTTP/1.1
Host: www.example.org

16. Cookie: cart=Bob

Answer: Cookie

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

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

Content-Type: application/x-www-form-urlencoded

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

User agent yes, User session no

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

Username and Password login credentials

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?

HTTP/1.1 200 OK

22. What web server is handling this HTTP response?

Server: Apache

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

Set-Cookie: SessionID=5

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

Content-Type: text/html

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

Monoliths and Microservices

There are many security headers in this example, but the one covered in class would have been Strict-Transport-Security, which tells the client that it should only be accessed over HTTPS and not HTTP

Answer the following questions about monoliths and microservices:

26. What are the individual components of microservices called?

Front-end server, back-end server, and a database

When a monolith's components are separated by functions, they are called services

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

“The back-end server is an application programming interface(API): the implementation of new protocols or features onto an existing application to alter the way it is used or accessed”

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

Containers like docker

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 (the slide with th octopus and the bricks)

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

“Docker Compose uses YAML to define the containers, their networking configurations, and where you want to copy files from your host machine into your container.”

Databases

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

SELECT * FROM customers

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.)

INSERT INTO (the topic of each section, then VALUE, and don't forget the semicolon in the right places)

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

DELETE FROM would remove entire table and delete all of the entries

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.

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

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.
 - **Question:** Did you see any obvious confirmation of a login? (Y/N)
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.
3. Read the contents of the `ryancookies.txt` file.
 - **Question:** How many items exist in this file?

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

Step 4: Log in 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`.
 - **Question:** Is it obvious that we can access the Dashboard? (Y/N)
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.

- **Question:** Look through the output where 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.

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>.

- **Question:** What happens this time?