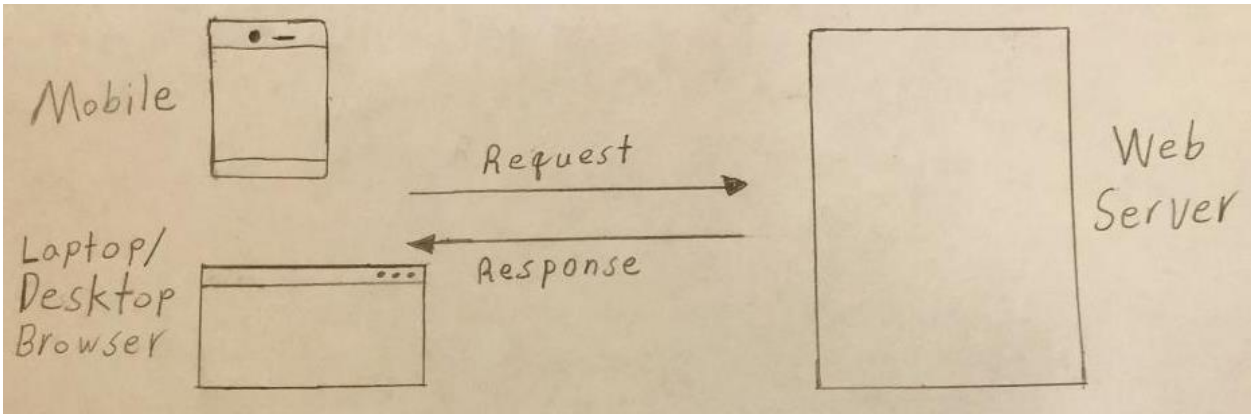
Host A Website for Mobile Users and Desktop/Laptop Users With Apache Web Server

Austin Wolfe

Introduction

This project uses the Apache Web Server to host a website for both mobile users and desktop/laptop through use of the apache modules mod_rewrite and mod_usertrack. Apache is configured to accept HTTP traffic (port 80) to listen for requests and, using mod_rewrite and mod_usertrack, return one of four different websites: A desktop/lap new visitor page, desktop/laptop returning visitor page, mobile new visitor page, and a mobile returning visitor page.

System Architecture



In this project, the webserver takes mobile requests and laptop/desktop requests and responds to them with the appropriate webpage for the type of device an whether they are a returning user. The Web server

Implementation

Materials

Web Server:

- Server Version: Apache/2.4.37 (Unix)
- HTTP version 1 used in this project.
- HTTP version 2 is installed, but not needed fort this project (see headers in access logs in “Testing” section)
- Apache running on a Linux virtual machine: CentOS Linux release 7.6.1810 (Core)

Apache web server modules used for this project:

- mod_rewrite
    - Directives: RewriteEngine, RewriteCond, RewriteRule
- mod_usertrack (compiled and loaded with the apxs command from source code. See methods section)
    - Directives: CookieTracking, CookieName, CookieExpires

Desktop/Laptop Browser (client):

- Browser: Chrome, Version 88.0.4324.182 (Official Build) (64-bit)
- Operating System: Windows 10

Mobile (client):

- Browser: Chrome Mobile Emulator running on a Windows 10 Laptop.

Methods

These are the methods used with the project materials to set up the Apache web server to take requests from the mobile and desktop/laptop clients and return the correct page. First we make sure we have all the modules needed, compiling them if we have to, and then setup the configuration file, being sure to load, define, and use the modules correctly.

Compile the module mod_usertrack with the apxs command

The mod_usertrack module is not yet compiled and loaded. This was done using the apxs commend:

- Find mod_usertrack in the metadata subdirectory which is in the modules subdirectory directory of the apache source code and use the apxs command to compile and load it into the Apache configuration file:
    - **/usr/local/apache2/bin/apxs -cia mod_usertrack.c**

- Search for **LoadModule usertrack_module modules/mod_usertrack.so** in the configuration file and make sure it is there and is uncommented.

Check that mod_rewrite is uncommented in the configuration file

This module is already loaded into the configuration file just search for it and make sure it is uncommented:

- Search for **LoadModule rewrite_module modules/mod_rewrite.so**

Set up the mod_rewrite and mod_usertrack directives to handle requests from client devices.

These requests will be handled so that: when a mobile request comes in for the first time, the server responds with the page **new_visitor_m.html** and stores a cookie on the computer so that when that mobile user returns and requests that page again, the **returning_visitor_m.html** page is returned.

Similarly, the first webpage request from laptop/desktop client, the webserver returns **new_visitor.html** and stores a cookie onto the device, and when that client returns and requests that page again, the **returning_visitor.html** page is sent in the response from the server.

In this implementation, the directives for the modules used to accomplish the desired responses to the web server are set up in a virtual host container that listens to all incoming traffic on port 80 (http).

```
<VirtualHost _default_:80>
    DocumentRoot /usr/local/apache2/htdocs
    RewriteEngine on
    CookieTracking on
    CookieName returninguser
    CookieExpires "3 weeks"


    RewriteCond %{HTTP_COOKIE} returninguser
    RewriteCond %{HTTP_USER_AGENT} "android|iphone" [NC]
    RewriteRule ^/index.html /returning_visitor_m.html
    RewriteCond %{HTTP_COOKIE} returninguser
    RewriteRule ^/index.html /returning_visitor.html

    RewriteCond %{HTTP_COOKIE} !returninguser
    RewriteCond %{HTTP_USER_AGENT} "android|iphone" [NC]
    RewriteRule ^/index.html /new_visitor_m.html
    RewriteCond %{HTTP_COOKIE} !returninguser
    RewriteRule ^/index.html /new_visitor.html

</VirtualHost>
```

- The **DocumentRoot** directive points to the directory that holds the webpages the rewrite rules that are written will refer to (using mod_rewrite and mod_usertrack):
- The **RewriteEngine** directive for mod_rewrite must be set to "on" along with the **CookieTracking** directive for mod_usertrack. Give the cookie a name and an expiration time with the directives **CookieName** and **CookieExpires**.
- The **RewriteCond** directives set the conditions for which **RewriteRule** directives rewrite the requested url to the new url and return the new url to the client (Fox & Hao, 2018, p.360).

Create simple webpages and place them in the correct directory.

Since, inside the configuration file, the DocumentRoot directive is pointing to /usr/local/apache2/htdocs, the webpages in this implementation of the project will be placed in that directory (just make sure the webpages are in the directory pointed to by the DocumentRoot directive).

Make the four simple websites inside the correct directory:

1. **new_visitor.html** with the content “New Visitor Page”
2. **returning_visitor.html** with the content “Returning Visitor Page”
3. **new_visitor_m.html** with the content “New Mobile Visitor Page”
4. **returning_visitor_m.html** with the content “New Mobile Visitor Page”

Testing and Analysis

The testing phase of the project is used to test whether the server is correctly responding to each type of client (mobile/new, mobile/returning, desktop/laptop/new, desktop/laptop/returning).

Testing the Request and Response Pages In the Chrome Browser

Desktop User test:

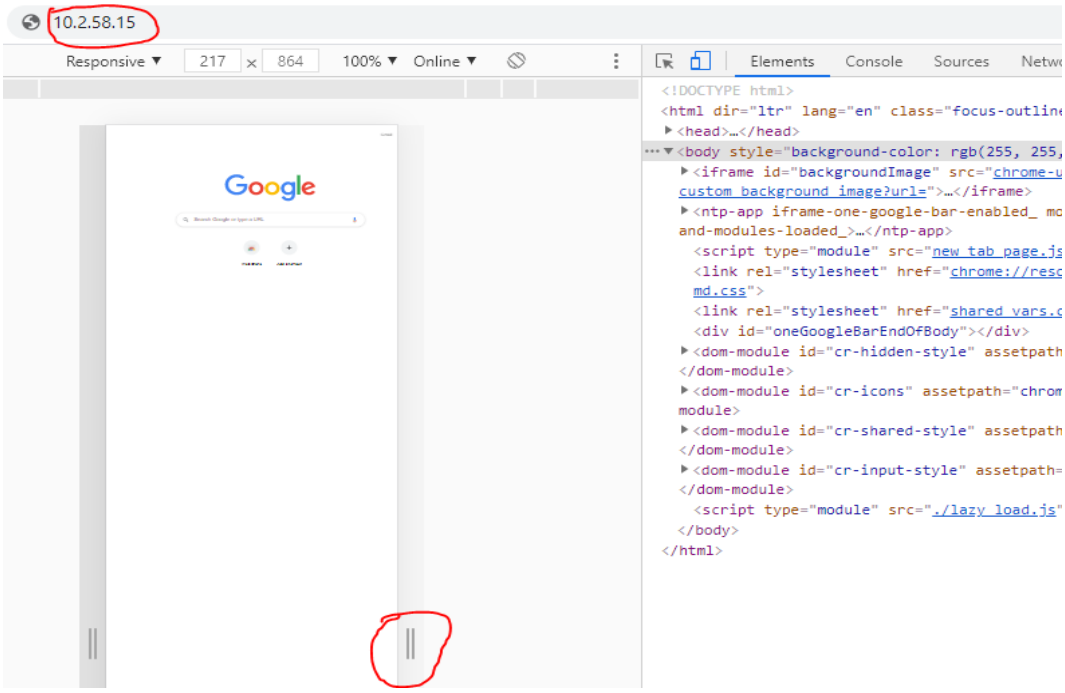
Connect to the server for the first time by putting the IP address in the search bar. The new visitor page should be returned first.

Enter the IP address to the server again and the mod_rewrite and mod_usertrack will let the server know to return the returning visitor page.

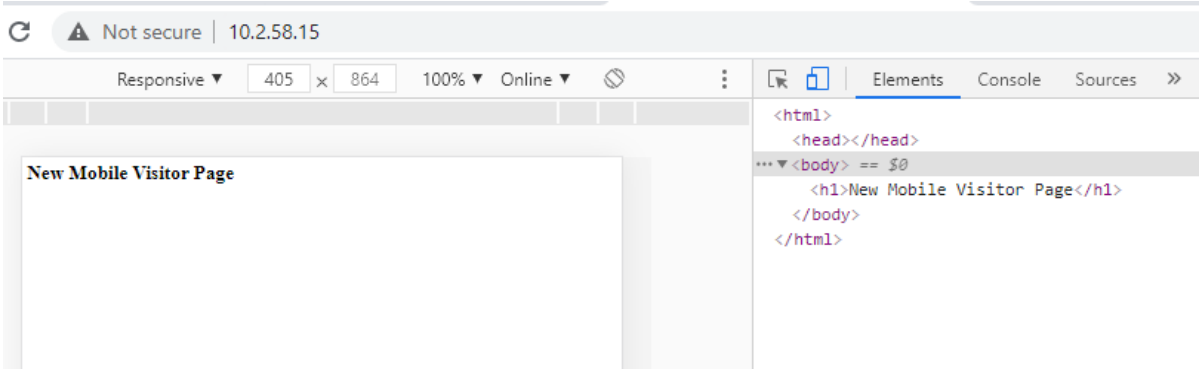
Mobile User Test:

Clearing the cookies in Chrome first, so to start fresh when testing the mobile server settings, we now use Chromes mobile emulation:

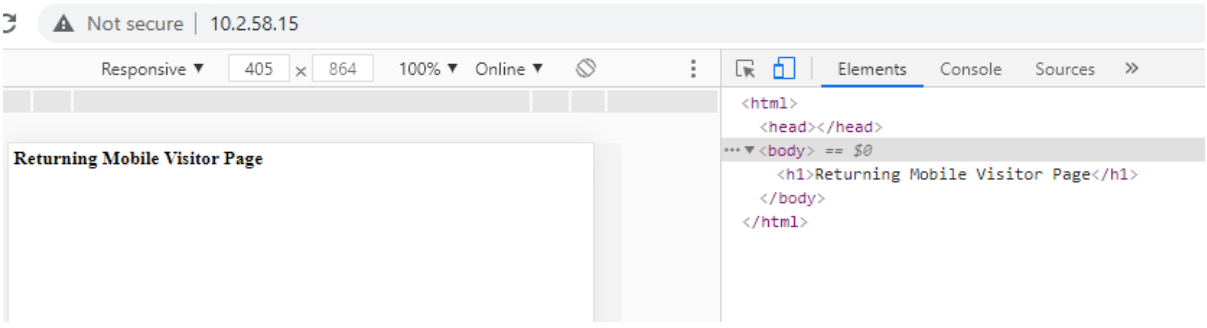
1. In Chrome use Ctrl+Shift+I and then adjust the window size to be smaller to emulate a mobile screen, then enter the ip address.



2. Once entered, the new mobile visitor page should be returned.



3. Enter to IP address again to get the returning mobile visitor page.



Looking at the Apache Log Entries for Each of the Requests Coming From a Client

Log Entries:

These are four log entries from the file **access_log**, inside the directory /usr/local/apache2/logs. Each start with the client IP address (the device contacting the server) and the date (in red).

The first two are the new and returning desktop visitor, the second (and last) two are the new and returning mobile visitor.

The error logs show, from left to right:

1. the remote hostname
2. the time that the request was received
3. the first line of the request
4. the status code of the request
5. the number of bytes sent
6. the "User-Agent" header sent by the client.

```
172.31.111.149 [21/Feb/2021:03:04:20 -0500] "GET / HTTP/1.1" 200 52 "Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.
4324.182 Safari/537.36"
172.31.111.149 [21/Feb/2021:03:04:27 -0500] "GET / HTTP/1.1" 200 58 "Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.
4324.182 Safari/537.36"
172.31.111.149 [21/Feb/2021:03:04:47 -0500] "GET / HTTP/1.1" 200 59 "Mozilla/5.0
(Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/88.0.4324.182 Mobile Safari/537.36"
172.31.111.149 [21/Feb/2021:03:04:51 -0500] "GET / HTTP/1.1" 200 65 "Mozilla/5.0
(Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko)
 Chrome/88.0.4324.182 Mobile Safari/537.36"
```

1. Full Table (see next two tables for a detailed, zoomed in, view of the logs):

Remote Host	Time Request Received	First Line of Request	Status Code of Request	Bytes Sent	Client User-Agent Header
172.31.111.149	[21/Feb/2021:03:04:20 -0500]	GET / HTTP/1.1	200	52	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.182 Safari/537.36
172.31.111.149	[21/Feb/2021:03:04:27 -0500]	GET / HTTP/1.1	200	58	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.182 Safari/537.36
172.31.111.149	[21/Feb/2021:03:04:47 -0500]	GET / HTTP/1.1	200	59	Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.182 Mobile Safari/537.36
172.31.111.149	[21/Feb/2021:03:04:51 -0500]	GET / HTTP/1.1	200	65	Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.182 Mobile Safari/537.36

2. Without Client User-Agent Header

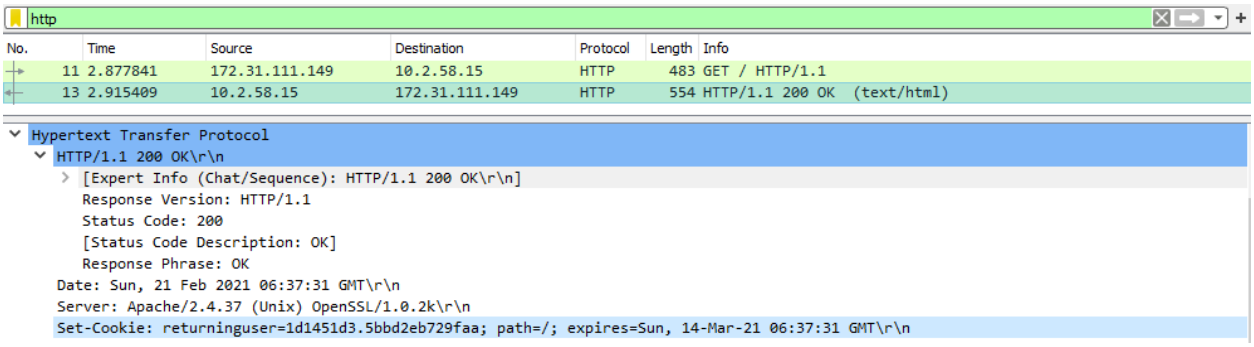
Remote Host	Time Request Received	First Line of Request	Status Code of Request	Bytes Sent
172.31.111.149	[21/Feb/2021:03:04:20 -0500]	GET / HTTP/1.1	200	52
172.31.111.149	[21/Feb/2021:03:04:27 -0500]	GET / HTTP/1.1	200	58
172.31.111.149	[21/Feb/2021:03:04:47 -0500]	GET / HTTP/1.1	200	59
172.31.111.149	[21/Feb/2021:03:04:51 -0500]	GET / HTTP/1.1	200	65

3. With Client User-Agent Header

Remote Host	Time Request Received	Client User-Agent Header
172.31.111.149	[21/Feb/2021:03:04:20 -0500]	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.182 Safari/537.36
172.31.111.149	[21/Feb/2021:03:04:27 -0500]	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.182 Safari/537.36
172.31.111.149	[21/Feb/2021:03:04:47 -0500]	Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.182 Mobile Safari/537.36
172.31.111.149	[21/Feb/2021:03:04:51 -0500]	Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.182 Mobile Safari/537.36

Observing the Cookie Exchange from Server to Client in Response to a Client Request.

Wireshark Cookie Exchange Capture:



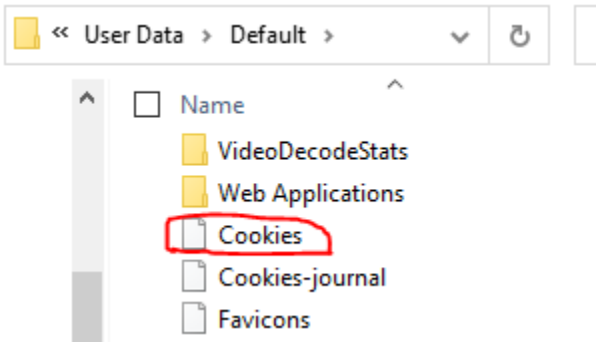
The capture shows data from frame No. 13, the server addresses being 10.2.58.15, setting the cookie (highlighted in light blue at the bottom) for the client at address 172.31.111.149.

This information in Wireshark can be found by filtering for http traffic and looking in the http section in the details pane (highlighted in darker blue). The cookie, “returninguser”, is able to be seen since http traffic is not encrypted.

How Cookies are Stored to a Device

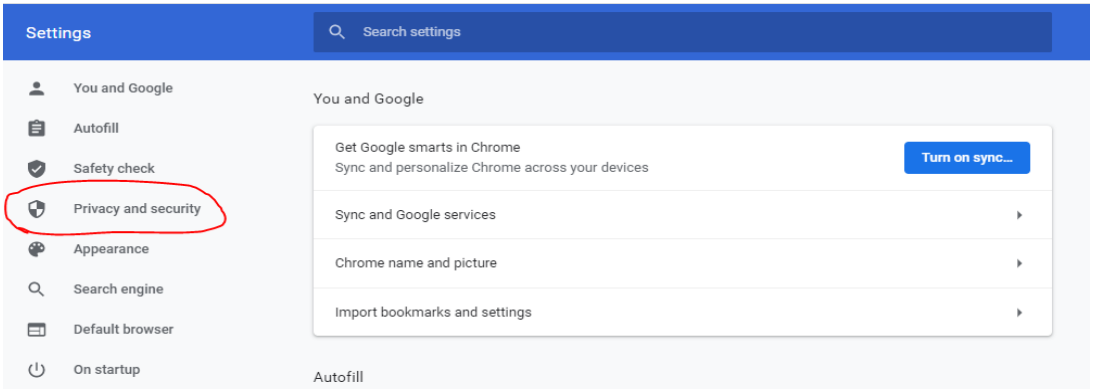
The Cookies File on the User Client:

Chrome stores cookies inside the Cookies file located at C:\Users\username\AppData\Local\Google\Chrome\User Data\Default (where “username” should be replaced by an actual username).

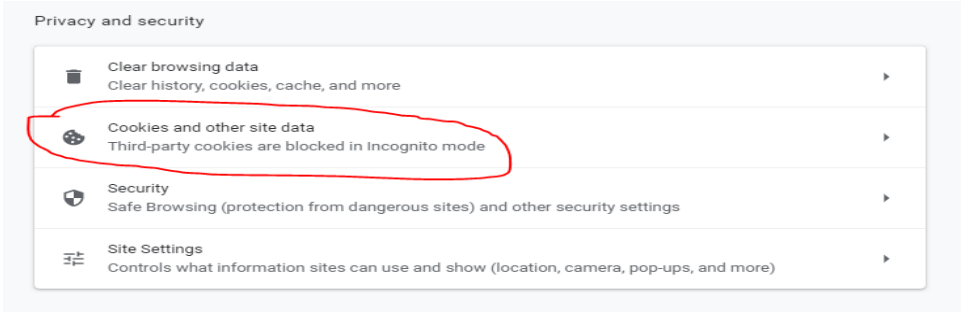


This file is compiled and not human readable, so the chrome browser is needed to view the cookies:

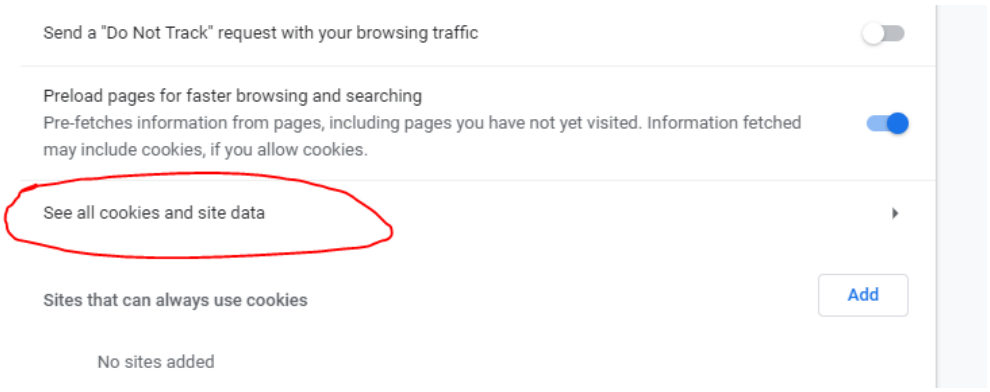
1. In Chrome, inside settings, click “Privacy and security”.



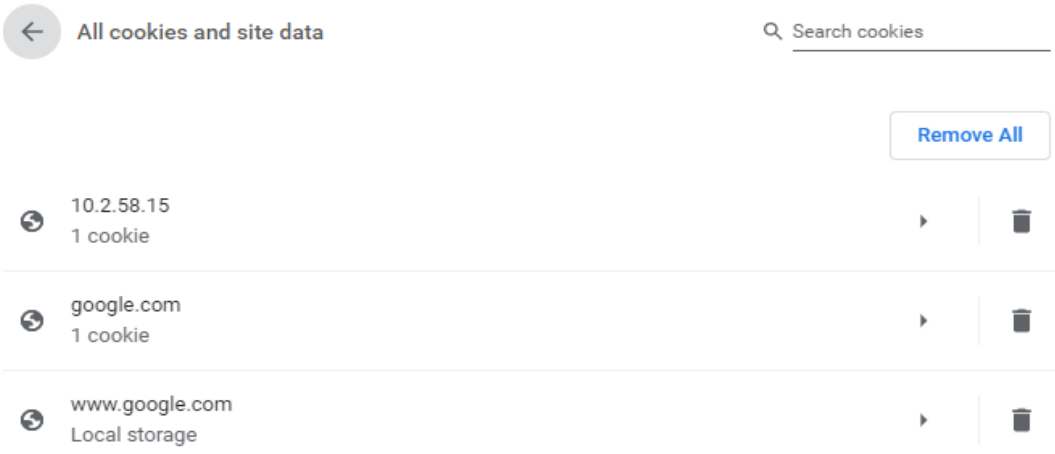
2. Click “Cookies and other site data”.



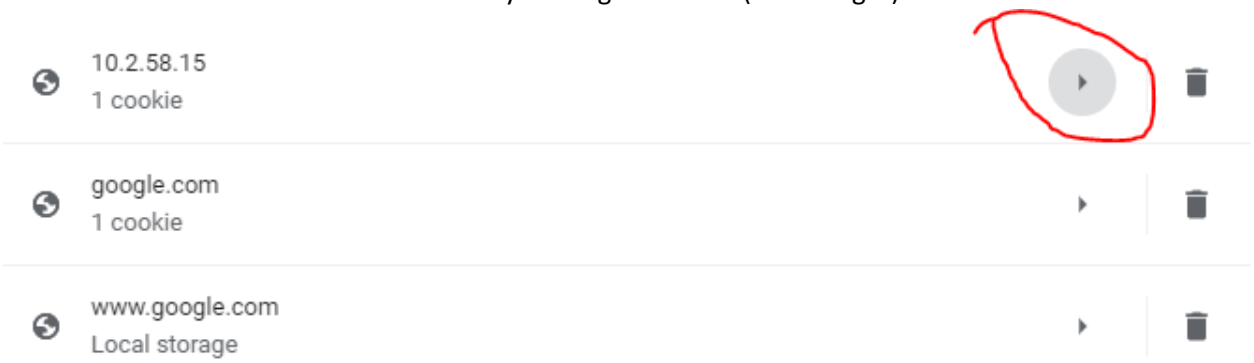
3. Scroll until you see (then click) “See all cookies and data”.



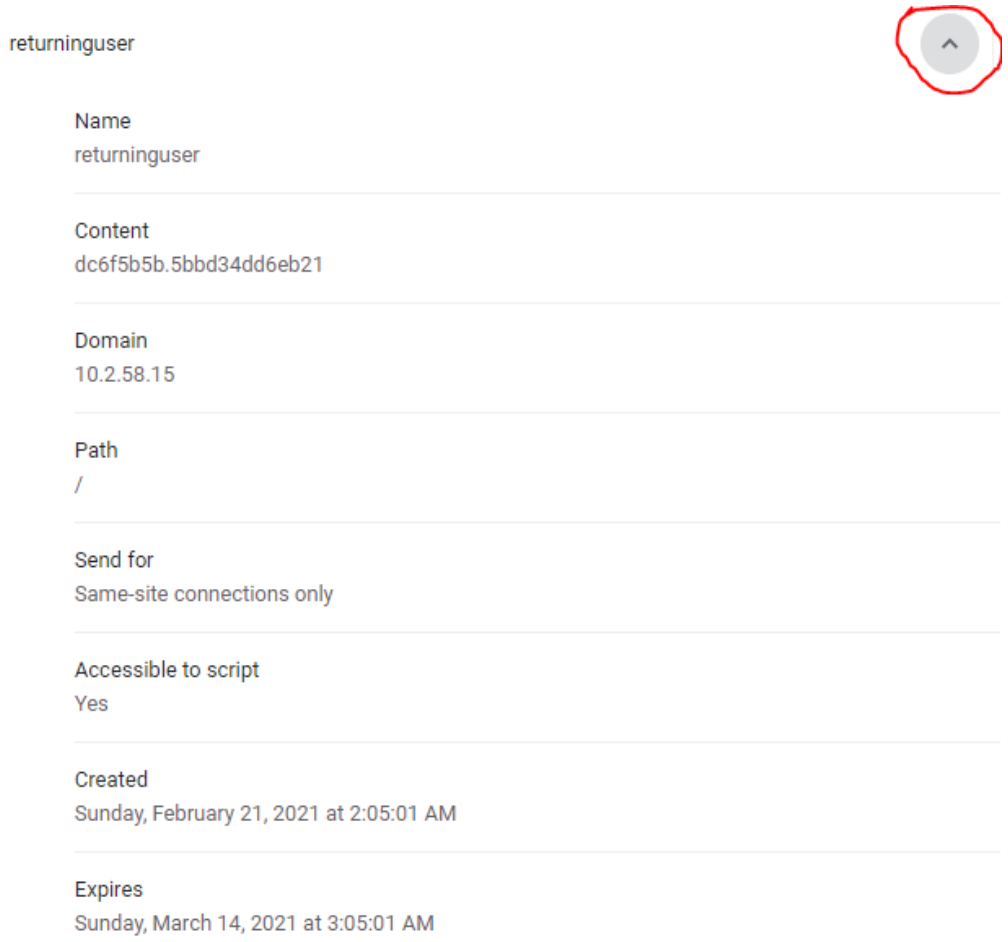
4. Here, the cookies Chrome stores on the computer can be seen. Notice the cookie from 10.2.58.15, which is the machine running the apache server set up for this lab.



5. You can see more details about a cookie by clicking the arrow (on the right).



Clicking the arrow in the next page will bring down a dropdown box with details about the cookie:



Websites put cookies on a computer's hard drive when someone visits a website for the first time. A cookie has a unique ID and is used to track someone's session while visiting the website, keeping track of certain things they do while they visit, and in our case, in this lab, to keep track of returning website visitors.

Conclusion

This project is accomplished by using an apache server that is setup for http traffic, compiling and using the correct modules (mod_rewrite, mod_usertrack) with the directives needed to create a cookie, rewrite conditions, and rewrite rules. These are placed inside of a virtual host container setup to listen to http requests and return the correct webpages that were made and pointed to by the DocumentRoot directive inside this virtual host container; these pages being determined by the type of device (desktop/laptop or mobile) and whether they are new visitors or returning visitors (if the web server has

stored a cookie on the device yet). If the device is visiting for the first time, a cookie is stored onto the device.

Personal impressions and feelings I have about this project are ones of appreciation and relief, since I now have a clearer understanding of how apache serves requests. I think the project is well made, since the instructions are well explained. I had no trouble understanding what the lab wanted me to accomplish, allowing me to learn the material faster.

References

Kobayashi, KobayashiKobayashi 1, Puru vermaPuru verma 8733 bronze badges, SRIDHARANSRIDHARAN 88811 gold badge1313 silver badges2727 bronze badges, George SmithGeorge Smith 15111 silver badge44 bronze badges, Pandian_Snklpandian_Snkl 43655 silver badges1616 bronze badges, & Some programmer dudeSome programmer dude 357k3030 gold badges337337 silver badges536536 bronze badges. (1964, May 01). Where does Chrome store cookies? Retrieved February 22, 2021, from https://stackoverflow.com/questions/31021764/where-does-chrome-store-cookies

Buckler, C. (2017, April 10). How to use mobile emulation mode in chrome. Retrieved February 22, 2021, from https://www.sitepoint.com/how-to-simulate-mobile-devices-with-device-mode-in-chrome/

Apache module MOD_REWRITE. (n.d.). Retrieved February 22, 2021, from http://httpd.apache.org/docs/current/mod/mod_rewrite.html

Apache module mod_usertrack. (n.d.). Retrieved February 22, 2021, from https://httpd.apache.org/docs/2.4/mod/mod_usertrack.html

Fox, R., & Hao, W. (2018). *Internet infrastructure: Networking, web services, and cloud computing*. Boca Raton, FL: CRC Press.