# ECS 152A: Computer Networks Project 2, Part II

Web crawling and HAR file analysis

Professor Zubair Shafiq

Krystal Chau, SID: 920918540 Jacob Feenstra, SID: 921423591

November 21, 2023

## 1 File Submissions

The following files were submitted along with this `proj2` PDF.

1. `crawling.py`
2. `analysis.py`
3. `harFiles.zip`
4. `top-1m.csv`
5. `gpt_part2.py`

## 2 Selenium Implementation

A quick note on the logic of our Selenium implementation.

1. Run Chrome, crawling across the websites indicated in `top-1m.csv`
2. If there is an exception of any kind at a given website, try and serve the website a second time through Selenium. If the second attempt fails, move on to the next website in the list.
3. Successful crawls have all HTTP traffic stored in a HAR file.
4. An individual HAR file for each crawled website. Syntax is:

   `[NUM_IN_top_1m.csv]_[NAME_IN_top-1m.csv].har`.

5. Ensure that the crawling discontinues once we have *successfully* crawled across 1001 websites (there is a different indexer to regulate this).
6. Access each HAR file, and perform desired analytics on third-party cookies in them.

## 3 Third-Party Cookie and Request Analytics

1. For the number of requests made to third-party domains when visiting each site, the `analysis()` function (which handles all HAR parsing, and returns all requested analysis) tallies third-party requests as it parses HAR files. Since each session of `analysis()` is for a particular website, the function simply prints out the tally when it has finished (and indicates the site it is printing the tally out for). Since there are 1001 sites, I will not include all of the analysis, but here is 10 sites visited, and their number of third-party requests.

   (a) amazon.com.br: 0

(b) yale.edu: 0
(c) ryanair.com: 0
(d) webform.org: 82
(e) innovid.com: 20
(f) quantcount.com: 58
(g) eyeota.com: 45
(h) nest.com: 59
(i) ui-dns.com: 2
(j) roblox.com: 2

Our source code, when run, prints the tallies for all 1001 sites.

2. The Top 10 most commonly seen third-parties across all sites are as follows, by their originating domain. Note this based on third-party requests, not the cookies which they send:

   (a) azureedge.net
   (b) azure.com
   (c) azurewebsites.net
   (d) mzstatic.com
   (e) azurefd.net
   (f) dzeninfra.ru
   (g) rzone.de
   (h) azure-dns.com
   (i) ozon.ru
   (j) dzen.ru

   Note these numbers are generated purely from the `domain:` key in requests or responses. Every time one is seen that is deemed third-party (by comparing against the name of the webpage currently being crawled), we increment, until the values above are achieved. So if there is the same cookie from the same domain, but it was used in separate request-responses, it will be counted each time, because it is a measure of "total requests" made to the third-party sites. The more it is requested to, the more it has been "seen".

   Lastly, the most commonly seen third-parties. This differs from originating domain. We look at the domain, to ascertain whether or not it is indeed a third-party cookie, but we must inspect the `name:` key to determine what the precise cookie is. Third-party web servers can produce all sorts of cookies, so it's important to determine what service each performs. Here is the top 10 third-party cookies, along with the domain they belong to, in the format (Domain, Cookie Name, Cookie Count). Some information on what this particular cookie does is below each item. We could only find information on some of these cookies on other sites. We've indicated the source after each informational blurb.

   (a) (doubleclick.net, test_cookie', 1171) Functionality cookie. Allows users interact with features of a site fundamental to the service. Anything from choice of language to maintaining user information (their shopping cart). (Source)
   (b) (rubiconproject.com, khaos, 1171) Store user data in an anonymous form. Includes IP address, location, websites visited, and ads clicked. Tailor ad's based on what a user views in a given ad network (Source)
   (c) (rubiconproject.com, audit, 317) Set to record cookie consent data. I am guessing it stores information on rubicon's server as to whether or not a user session did an opt-in or opt-out for cookies. (Source)

(d) (adsrvr.org, TDID, 257) This domain is owned by TheTradeDesk. The main business activity is: Ad Serving Platform. This cookie carries out information about how the end user uses the website and any advertising that the end user may have seen before visiting the said website. (Source)

(e) (linkedin.com, bcookie, 240) This is a Microsoft MSN 1st party cookie for sharing the content of the website via social media.(Source)

(f) (linkedin.com, lidc, 240) This is a Microsoft MSN 1st party cookie that ensures the proper functioning of this website.(Source)

(g) (yahoo.com, A3, 228) This domain is owned by Yahoo. The main business activity is: Search / Advertising (Source)

(h) (bing.com, MUID, 200) This cookie is widely used my Microsoft as a unique user identifier. It can be set by embedded Microsoft scripts. Widely believed to sync across many different Microsoft domains, allowing user tracking.(Source)

(i) (adnxs.com, uuid2, 188) These cookies are used to deliver adverts more relevant to you and your interests. They are also used to limit the number of times you see an advertisement as well as help measure the effectiveness of the advertising campaign. They are usually placed by advertising networks with the website operator's permission. (Source)

(j) (demdex.net, demdex, 172) This cookie helps Adobe Audience Manger perform basic functions such as visitor identification, ID synchronization, segmentation, modeling, reporting, etc. (Source)

## 4 Addendum

We had some technical issues with this part of the project, and we believe it likely has to do with OS perms and specifications. For one, we did not get nearly as many third-party cookies as we anticipated, even with mobbrowser-proxy set up as per the readme.md instructions (to our knowledge). There was a conspicuous lack of doubleclick.net cookies, although it appears that a healthy amount of our analytics can be attributed to indirect third-party domains owned/operated by doubleclick.net. Our code works, we're just not confident that the HAR files we received reflected unfettered traffic in the top websites of the Internet.

There was one hangnail in the coding of this project. Parsing the HAR file with json tools was not particularly hard in of itself, but it is difficult to figure out an appropriate second-level domain with a general implementation. Take for instance `google.com`, `googleapis.com`, `googlevideo.com`, and `googletagmanager.com`. All of these are in the `top-1m.csv`, and made it in our top 1000 analysis. Provided that we parse `google.com` first, we have an extracted second-level domain that will nominate the rest of the websites indicated above as first-party, if they have relevant cookies on `google.com`. But this cannot be determined without absolute certainty; not without a bit of hard-coding. If 3 or 4 yahoo websites appear, and the extracted second-level domain is something like `analytics.yahoo.com`, then it will use `analytics.yahoo` for it's subsequent comparisons. Other yahoo domains will be viewed as third party. The only way to reliably ensure an appropriate second-level domain is to hardcode this, or use some involved regular expressions (which would also be hardcoded).

This is also not to mention smaller websites, such as `as.com`. As a second-level domain, `as` could be paired with a great many websites, and invalidate them as genuine third-party domains. While we reason that there is a regular expression out there, making use of `.` as delimiters, it would have to be truly encompassing and sensitive to a variety of incoming webpages. It's an interesting problem, but not one that we were able to solve, unfortunately. Because of this, we reason that there is some noise between first-party and third-party cookie declarations. Some might not actually be third-party. Hence why we have an alarming number of azure-like domains

Finally, here is the link to session which created the Chat GPT code in `gpt_part2.py`
ChatGPT-3.5 Session Link