Assignment 1

1. **Pick a website**

- Pick a news website **or** an online shop from the lists below.

News websites	Online shops
● https://www.nu.nl/	● https://www.coolblue.nl/
● https://www.ad.nl/	● https://www.ah.nl/
● https://www.telegraaf.nl/	● https://www.zalando.nl/
● https://nos.nl/	● https://www.wehkamp.nl/
● https://www.rtlnieuws.nl/	● https://www.amazon.nl/
● https://www.volkskrant.nl/	● https://www.jumbo.com/
● https://www.nrc.nl/	● https://www.aboutyou.nl/
● https://www.metronieuws.nl/	● https://www.debijenkorf.nl/
● https://www.trouw.nl/	● https://www.hm.com/

Capture the HTTP traffic

For the website you chose:

1. Create a new Chrome/Chromium profile for the assignment
2. Open the Devtools/Network panel
3. Check “Preserve log” (that’ll retain all requests made during a session)
4. Load the website’s homepage; accept all cookies/data processing, dismiss other potential dialogs (permission to send notifications, location access, email signup etc.)
5. Scroll down until the bottom of the page
6. Click on an article or a product page (multiple clicks are okay if you have to). Avoid external links, the inner page should be under the same first-party domain as the homepage
7. Scroll down until the bottom of the second page
8. Save all HTTP request/responses as HAR to a file using the following naming convention: `example.com.har`. No `www.` or other prefixes; just `domain_name.har`.

Capture the HTTP traffic with an adblocker

Now, install uBlock Origin **or** Adblock Plus on Chrome/Chromium. Repeat steps 1-8 **starting again with a fresh profile**, this time with the add-on installed. Name the second HAR file as `domain_name_adblocker.har`. Now you should have two HAR files: one with the adblocker and one without.

Analyze the HAR Data

Write an analysis script as a Jupyter Notebook (.ipynb) or as a standalone Python script (.py) that processes the captured HAR files and outputs the following as two separate JSON files, each containing a (Python) dictionary of results. The overall processing pipeline should look like the following:

- HAR -> analysis -> results dict -> save to JSON

The results dictionary serialized in each JSON should contain the following keys:

- num_reqs: Integer, number of requests (observed in the HAR file)
- num_requests_w_cookies: Integer, number of requests with cookies
- num_responses_w_cookies: Integer, number of responses that set at least one cookie
- third_party_domains: list of distinct third-party domains (eTLD+1)
- cookie_domains: list of distinct cookie domain attributes (using the cookies field)
- xorigin_cookie_domains: list of cookie domains set via HTTP response headers, with SameSite=None, and lifespan >= 90 days
- server_countries: list of distinct server countries (using the serverIPAddress field and the geolocation databases linked below)
- requests: a list of dictionaries, where each dictionary contains the following request/response details:
    - request_domain: String; e.g. example.com
    - server_country: String; e.g. Germany; “unknown” if server IP is unavailable
    - num_request_cookies: Integer
    - num_response_cookies: Integer
    - is_tracker: Boolean; whether the request hostname or domain is listed in EasyList or EasyPrivacy “*just domains”* blocklists
    - url_first_128_char: String; the first 128 characters of the URL; e.g. https://example.com/pixel.gif

Tips:

- The requests list will contain one dictionary for each request-response pair
- You can ignore the blocked requests and responses
- When saving the HAR file you can use one of the following: 1) Export HAR button (⬇) ; 2) Right-click -> Copy-> Copy all as HAR -> paste to an empty file, 3) Right-click -> Save all as HAR with content. Either of these options should work, but in some edge cases there may be character encoding or other unexpected issues
- For server_countries, consider both first and third-party servers
- Unless specified, “*domain”* means eTLD+1
- cookie_domains and xorigin_cookie_domains may include domains with a leading dot (.example.com). You don’t need to do anything about it
- Comment your code when what you do is not obvious
- DRY: Don’t Repeat Yourself. Break your code into reusable small functions
- Avoid deep code indentations
- Use meaningful variable and function names

   a. ✓good: request_domain, response_headers, get_country_by_ip_address
   b. ✗not good: foo, bar, tmp, do_stufff

Practicalities

- Upload a zip file containing the files listed below (a-f). Name the zip file after your student number; e.g. s012345.zip. File names inside the zip archive should look like this:
    a. example.com.har
    b. example.com.json
    c. example.com_adblocker.har
    d. example.com_adblocker.json
    e. s012345.ipynb *OR* s012345.py (analysis script)
    f. requirements.txt: Python packages required to run your script, if any
- You can assume the following files will be available in the same folder as your code. You **do not need to upload** them in your zip file. When testing your code, we will extract your zip file and copy the below files (a-d) to your folder:
    a. easylist-justdomains.txt (link)
    b. easyprivacy-justdomains.txt (link)
    c. dbip-country-lite.mmdb (link)
    d. GeoLite2-Country.mmdb (link): Alternative to (c), requires a MaxMind account
        - Note: using either c or d is acceptable
- Your code should **not** make any calls to online APIs. It should be able to work offline.
- You are free to use publicly available Python packages (e.g. to parse dates).
- You can print log messages from your code (you don't have to)
- Your code should work with Python 3
- Your code should be able to run without any command line parameters
    a. Hard-code the HAR filenames in your code, assume they are in the same folder as the analysis script/notebook
    b. Running "python s012345.py" once should re-generate both JSON outputs with the same content as the submitted JSONs (i.e., the results should be reproducible)
    c. Jupyter Notebook: Should run without any intervention and re-generate the exact JSON outputs