



## NWI-IMC074 Online Tracking and Privacy / Assignment 2

 Grade weight: **3.5/10** of the final grade  
 Due: **12 April 2024** (23:59, Nijmegen time)

### Objectives:

- **Develop a web crawler** using [Playwright](#) that satisfies the requirements listed below
- Run two separate crawls of 50 [Dutch government websites](#):
  - **Crawl-allow**: Accept all cookies, do not block trackers
  - **Crawl-block**: Accept all cookies, block requests to tracker domains listed in [Disconnect's blocklist](#)
- Analyze the crawl data (see below)
- Prepare a report based on your analyses

### Crawler requirements:

- Should accept the following as command line options/parameters:
  - **--block-trackers**: when this option is provided, the crawler should block trackers
  - **-u, -l**: list of websites or domains to be crawled, as a single URL (-u), or a list of URLs in a file (-l). For example:
    - `python crawl.py --block-trackers -u example.com`
    - `python crawl.py -l sites-list.txt`where sites-list.txt contains a list of URLs, each on a separate line.
- Should be able to accept cookies (and personal data processing) on the GDPR consent dialogs after page load.
  - You can benefit from the [priv-accept](#) project to identify the “Accept” buttons on consent dialogs. You may need to extend the list of keywords.
- To evade bot-detection scripts that expect a user-like interaction, the crawler should scroll down until to the bottom of the page in multiple steps (after accepting cookies). See [this example code](#) from OpenWPM (a web privacy measurement tool) for inspiration, but feel free to use [Playwright's internal methods](#).
- Should implement the following waits between crawling steps:
  - Page-load -> Wait 10s -> Click accept (if any) -> Wait 3s -> Scroll down -> Wait 3s -> Close the page
- Should record HTTP requests and responses to a HAR file. You can use Playwright's internal HAR recording feature.
- Should take screenshots of the page, before and after accepting cookies.
- Should record a [video](#) of each visit.

## Analyses & the Report:

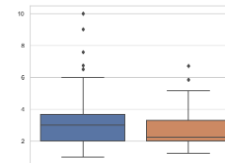
Prepare a report that contains the following analyses. You do not need to comment on the figures and tables, but you should provide self-explanatory captions for each table and figure.

1. Add a table on the number of failures encountered during each crawl, similar to the following:

Error type	Crawl-accept	Crawl-block
Page load timeout	5	6
Consent click failure	3	2

2. Compare data from the two crawls using a series of box plots, where X axis is the crawl name (*Crawl-accept*, *Crawl-block*), and Y axis is one of the metrics given in (a) to (e). For instance, the box plot for *Number of requests* will be based on *the distribution of number of requests per website* in each crawl. You will draw a separate box plot (similar to the figure below) for each of the following metrics:

- a. Page load time (based on [load event](#))
- b. Number of requests
- c. Number of distinct third-party domains
- d. Number of distinct tracker domains
- e. Number of distinct third-party domains that set a cookie with *SameSite=None*, and without the *Partitioned* attribute (no need to check the expiry date)



3. Using the metrics from 2a-e, compare data from two crawls in a table such as the following:

Metric	Crawl-allow			Crawl-block		
	Min	Median	Max	Min	Median	Max
Page load time (s)	2.3	20.4	45.9	2.0	20.3	70.6
Number of requests	2	100	300	...	...	...
...						

4. Add a table of ten most prevalent third-party domains (based on the number of distinct websites where the third party is present), indicating whether the domain is classified as a tracker or not by Disconnect.

Third-party domain	Number of distinct websites (Crawl-allow)	Number of distinct websites (Crawl-block)	isTracker?
google-analytics.com	30	15	Yes
someotherdomain.net	25	12	No
...			

5. Add a frequency table of HTTP methods (such as GET, POST, ..) for each crawl.

HTTP Method	Number of requests with this method (Crawl-allow)	Number of requests with this method (Crawl-block)
GET	3000	2000
POST	...	...
...	...	...

6. Analyze the [Permissions-Policy](#) headers encountered in the crawls and make a list of websites that disable access to camera, geolocation or microphone for all parties (including first and third). In total, you should make 6 separate website lists: 2 crawls x 3 permissions.
7. Analyze the [Referrer-Policy](#) headers encountered in the crawl data, and make a list of websites that use no-referrer or unsafe-url values (separately). In total, you should make 4 separate lists: 2 crawls x 2 Referrer-Policy values.
8. Analyze the [Accept-CH](#) headers encountered in the crawl data, and make a list of 3 **high-entropy client hints** that are requested on most websites:



Client Hint	Number of websites (Crawl-allow)	Number of websites (Crawl-block)
Sec-CH-UA-Platform-Version	14	12
Sec-CH-UA-WoW64	...	...
...	...	...

9. For each crawl, identify the 3 most prevalent (by distinct websites) cross-domain [HTTP redirection](#) (source, target) pairs. Cross-domain redirection means the source and target of the redirection has different eTLD+1's.

Source domain	Target domain	Number of distinct websites
google-analytics.com	doubleclick.net	10
...		

10. In a few sentences:
- What were the most challenging parts of the assignment?
  - What were the findings that surprised you, if any?

### Tips:

- In order to detect tracking-related requests or tracker domains, use [Disconnect's blocklist](#) and (optionally) [Mozilla's trackingprotection-tools library](#).
- Websites without consent dialogs should still be included in the results
- You can use Playwright via Python or NodeJS.
- You are free to use publicly available Python or NodeJS packages, except packages that use Playwright under the hood (feel free to ask on Brightspace when in doubt). You can take inspirations from the following fully equipped web privacy focused crawlers:
  - <https://github.com/duckduckgo/tracker-radar-collector>
  - <https://github.com/openwpm/OpenWPM>
- Add titles for X and Y axes in the plots (e.g. Y-axis title: *Page load time(s)* ). Add captions for figures and tables in the report.
- Use a Github/Gitlab repository for development and issue tracking. The repository can be private.
- Avoid running browsers in parallel because this may get you blocklisted.
- Unless specified, "*domain*" means eTLD+1.
- Use meaningful variable and function names
  -  good: request\_domain, response\_headers, get\_country\_by\_ip\_address
  -  not good: foo, bar, tmp, a, do\_stuff, get\_data
- Your code should **not** make any calls to online APIs.

## Submission

- Upload a zip file containing the data, source code and the report. Name the zip file after your group name on Brightspace. The zip file contents should be organized as follows (📁=folder):
  - a. report.pdf
  - b. 📁 crawler\_src
    - README.md (instructions on how to install (when applicable) and run your crawler)
    - *[Crawler source code. Can be multiple files and/or folders.]*
  - c. 📁 analysis
    - *[Analysis source code. Can be multiple files and/or folders. Only Python scripts or Jupyter Notebooks.]*
  - d. 📁 crawl\_data\_allow
    - example.com\_allow.har
    - example.com\_allow\_pre\_consent.png (*screenshot before clicking Accept*)
    - example.com\_allow\_post\_consent.png (*screenshot after clicking Accept*)
    - example.com\_allow.webm (*captured video*)
    - [same files for the other websites....]
  - e. 📁 crawl\_data\_block
    - same files as crawl\_data\_allow, but for the block crawl (i.e. we should have "block" in the filenames instead of "allow".)
  - f. requirements.txt or package.json: Python or node packages required to run your script, if any.

## Help

- Ask your questions on this Brightspace discussion list:  
<https://brightspace.ru.nl/d2l/le/427013/discussions/topics/110332/View>

🍀 Good luck! 🍀