# TP3 Web tracking Technologies

The amount of stars "*" corresponds to the difficulty of the exercise: the more stars it has, the more difficult is the exercise.

### 1* Cookie access control

(Folder: ex2) Assume a web browser's cookie storage contains the following cookies: example.com: eid = '123', tracker.com: tid = '456', another.com: aid = '789'.
Analyse the source code of the cookieAccess.html and the embedded scripts and iframes. At every call to `document.cookie`, which cookies will be read and why? Analyse each HTTP request that is send by this application to `http://tracker.com` and list all the cookies that this server will receive. If no cookies are sent at some HTTP request, justify why.

### 2* Third-party cookie and HTML5 localStorage testing

(File: integrator.html) Choose your favourite web browser. Disable third-party cookies/third-party data in the browser preferences. In integrator.html set up your own third-party domain, and write the code for gadget.html to test whether your browser

1. Automatically sends third-party cookies in an HTTP header whenever they are present in the browser,
2. Allows a JavaScript program that runs in a third-party origin to
   - set third-party cookies;
   - read third-party cookies;
   - set third-party HTML5 localStorage;
   - read third-party HTML5 localStorage.

Does your web browser implement the blocking of third-party cookies properly? Does it also disable the usage of the third-party HTML5 localStorage as expected? Justify your answers.

### 3*** Advanced cookie stealing

(No files) Consider a web application consisting of 4 pages:

- main.html located at `google.com`
- cart.html located at `play.google.com`. The user has already visited this page, and the cookie is now stored in the user's browser.
- apps.html located at `play.google.com`. To access some DOM elements at main.html, the script on apps.html changes its effective origin from `play.google.com` to `google.com` by `document.domain` DOM API.
- events.html located at `events.google.com`. This domain is controlled by the attacker.

How an attacker can steal the cookie associated to the `play.google.com` domain stored in the user's browser?

### 4** Respawning cookies by HTML5 localStorage

(File: respawning.html) Assume that the web application's server is located at `example.com`, and the server sets a new cookie with the name `id` every time when it responses to the HTTP request without any cookie named `id`. Check the source of the file respawning.html and write code for the script respawn.js that implements respawning of cookies via HTML5 localStorage even if the user has deleted them from his browser.

Is such tracking first-party or third-party? Does it support tracking over multiple sites (for example if respawn.js script was present on the other site at `another.com`) or only on site `example.com`?

### 5** Tracking via Etag header

(No file) Implement a web server that provides tracking via a Etag header. Give examples of the corresponding HTTP request and response headers and explain how this tracking technique works. Provide an example of a web application that uses this technique and allows tracking across multiple cites.

### 6* Cookie syncing

(No file) Implement a cookie syncing on a server located at `A.com` that allows a server at `B.com` to synchronise `A.com`'s cookies with `B.com`'s cookies.

### 7** Fingerprinting browser extensions

(No file) Implement a JavaScript program that is able to detect the following browser extensions in either FireFox or Google Chrome browsers: AdBlock Plus, Disconnect, Privacy Badger, Ghostery, Do Not Track Me, Lightbeam. If for some extensions it is impossible, explain why.