TDDD17 Tracker^2

[trackertracker]
Project report for Information Security Course
Linköpings Universitet, Sweden

Marcus Elander
marel184@student.liu.se

Linus Aarnio
linaa722@student.liu.se

Supervisor: Niklas Carlsson
niklas.carlsson@liu.se

***Abstract*—This project has investigated the possibility to overlay ads with information regarding ad provider and source domain. It was found that connecting an ad to an element on a web page is a hard task in general but possible in some cases. A plugin, based on CatBlock and AdBlock was developed which displays all identified ads on a web page in a list within the plugin popup menu and overlays on a subset of ads on the web page. The report discusses the privacy implications of third party ads, the structure and limitations of the created plugin and alternative approaches.**

I. Introduction

Our task was to create a browser plugin that overlays ads and other third party elements on a web page with the domain name of the source of the element. The goal is to easily see which third party trackers and ad services a user is exposed to on different pages and what content they provide. The plugin is intended to be used in research on web tracking and possibly be extended to use by end users interested in knowing which parties are tracking them and selectively block ad providers.

Using a tool such as this can help a user or researcher find out why a user is shown certain personified ads. Since the requests from each page is sent to the ad provider, it is also possible to know which third party services are gathering the most information about users by finding the most prevalent domains on pages.

II. Background

Ad networks are in many cases setting cookies that identify the user, so that the aggregated data from different sites can be tied to a specific user. [1] If such a cookie is set by a network present of all sites visited by a user, they have access to the users' entire browsing history.

In addition to this aggregated data, ad networks use algorithms to create profiles, which can be very detailed if a lot of data is collected, on users. These profiles are not kept private and might be openly sold to interested parties. [2]

A. Theoretical method

The theory behind identifying third party elements has been extracted from existing ad blocking plugins, such as Catblock [3] and uBlock Origin [4]. We have analyzed the source code and tested elements from a local copy to find the method used to identify an element as an ad.

B. Practical method

Development of the plugin started, as stated in the previous section, with analyzing existing plugins. The development followed a step-for-step plan to enable handling of smaller pieces of a bigger problem. The steps were defined as following.

1) Setup basic plugin
2) Identify ad elements
3) Generate list of all ad source domains
4) Connect source domain to ad element
5) Generate overlay element
6) Place element on existing ad

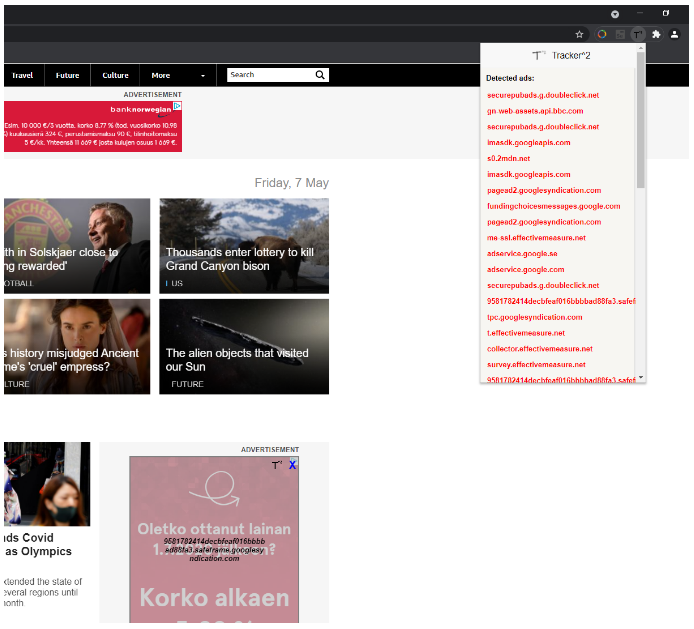
III. Solution and Analysis

This section describes the solution and provides technical details regarding how it works.

The result of this project is a plugin that identifies all source domains of ads on a web page and displays them in a list and on the ads' respective elements as an overlay. This solution was reached through studying existing ad-blockers and building upon Catblock. It enables the user to analyze what ads appear when browsing and where they are provided from.

The plugin works by utilizing Adblock and by extension filter lists to identify ad elements when a request is made from the ad provider. The filter lists contain known patterns that have been reported as ads. The patterns are found in urls, css-classes and styles. Every request made by the browser is intercepted and checked with regex for a match against the different patterns, mostly url patterns. Adblockers also use css matching, e.g. an element with css class containing ”right-col-ad-600” would be matched from EasyList [5]. However, by the time this matching is made there is usually no connection to the serving domain, which leads to it being unusable in this application.

The intercepting/blocking functionality in itself is using the existing API from the *chrome.webRequest* library. [6] By defining a listener and attaching it to *onBeforeRequest* one can access the request and some of it's data. Then, through the previously mentioned filtering, the handler function can return *true* for the requests that should be blocked and the listener can be defined to perform this blocking. See below for code example.



Tracker^2

Detected ads:

securepubads.g.doubleclick.net
gn-web-assets.api.bbc.com
securepubads.g.doubleclick.net
imasdk.googleapis.com
s0.2mdn.net
imasdk.googleapis.com
pagead2.googlesyndication.com
fundingchoicesmessages.google.com
pagead2.googlesyndication.com
me-ssl.effectivemeasure.net
adservice.google.se
adservice.google.com
securepubads.g.doubleclick.net
9581782414decbfeaf016bbbbad88fa3.safef
tpc.googlesyndication.com
t.effectivemeasure.net
collector.effectivemeasure.net
survey.effectivemeasure.net
9581782414decbfeaf016bbbbad88fa3.safef

Fig. 1. An example of the popup display window and an overlayed ad on BBC homepage

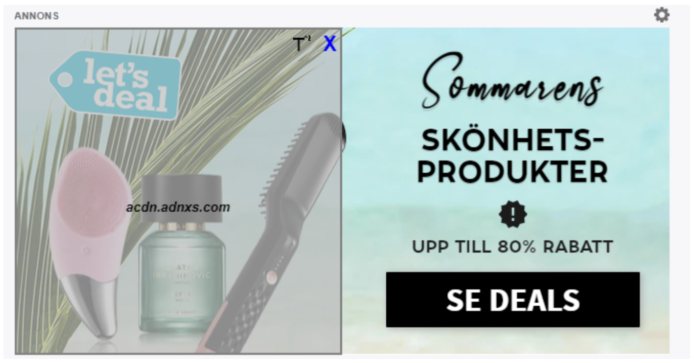


Fig. 2. An example of an overlayed ad on Aftonbladet homepage

Listing 1. Example code of blocking requests with a handler function.

```
chrome.webRequest.onBeforeRequest
    .addListener(onBeforeRequestHandler,
    { urls: ["http://*/*",
             "https://*/*",
             "ws://*/*",
             "wss://*/*"]},
    ["blocking"]
  );
```

Our solution uses the existing code from Catblock/Adblock to intercept and match the requests for ads, but is modified to not block the requests but instead store the matched urls in a list. The identified ads are then displayed in the plugin's popup window accessible by the toolbar. The existing plugin popup window from Catblock was remade to display this list instead of the data it originally displayed. An example of the popup window showing the ads found on bbc.co.uk is found in Figure 1.

The overlays generated by the plugin works by identifying the position of the element that made the request and then placing a new partially transparent element with the same position and size. This new element can then display the url the request is made to. An example of such an overlay on the homepage of Aftonbladet is shown in Figure 2. The problem with this approach is that only some types of elements have a known position and size at the time the request is made. Table I shows the resource types that can make requests, and indicates those where we can determine size and position of the element with green. The most common type of resource to request an ad resource are scripts. [7] They perform the request before creating the HTML element which makes our approach unable to overlay the ad element.

TABLE I
LIST OF REQUEST TYPES WITH INDICATION OF WHETHER THE SIZE AND POSITION OF THE ELEMENT CAN BE DETERMINED IN THE REQUEST

🟥 main_frame	🟩 sub_frame	🟥 stylesheet
🟥 script	🟩 image	🟥 font
🟩 object	🟥 xmlhttprequest	🟥 ping
🟥 csp_report	🟥 media	🟥 websocket
🟨 other		

A. Ad types

This section discusses identified types of ads such as ads provided by the host domain, third-party ads and *"one pixel ads"*.

We have identified three main classes of elements picked up by the plugin. The first one is ads where the host domain itself or a variation of it is the one serving the ad. It might still be an ad for a third party product or service and link out to another domain on click, but our plugin picks up only the server. The second class are visible third party-ads, where the whole ad is fetched as a resource from a third party domain. The third class are elements which cause a request to a third party domain but do not result in a visible element on the page. We refer to these as one pixel ads.

From a privacy standpoint, we can see differences between the types in how much data the trackers can collect about a user and the knowledge a user has about the tracking. The ads served from the same domain can collect data about the user, but only while he/she browses on the domain. Even if the ad inserts tracking cookies, a domain that only serves ads on its' own pages can't aggregate data about the user across the internet. First-party ads tend to be visible, and the user also expects the same domain to know about their browsing habits on the site so we consider the knowledge about tracking to be good in this kind of ad. A third party ad may be able to gather more data about the user. If the ad is served from an ad network with presence on many sites, and the network places tracking cookies in the users browser, it can have information about a users entire browsing history. Otherwise, they can gather the same amount of data as first-party ads. Ads are usually clearly visible, but since it might not be obvious that the request is made to a third party domain instead of the same domain as in the first class, the detectability of the tracking is lower.

The third class is one pixel ads, where an element of only one pixel is inserted on the page and performs a request to a third party domain. This can collect data about the user in the same way as third party ads, but has much lower detectability. A tech-savvy user might see a third party ad and be aware they are being tracked, but it doesn't help to know about the tracking capabilities of one pixel ads since they can't be seen on the page without a plugin pointing them out.

B. Evaluation and Comparison

This section will discuss whether the solution proved to be effective and why. Alternative solutions will also be discussed and how they could potentially change the outcome of the project.

1) AI based ad identification: Another alternative solution is to make use of AI instead of filter lists to identify ads on the web page. Possible advantages of choosing such a solution could be a more detailed identification of the ads, the possibility to categorize based on content and other parameters as well as not having to rely on the filter lists being up to date and comprehensive. Drawbacks to this solution are the specialized knowledge required and the time to implement such a solution and to train the AI to perform well. There are existing efforts to use machine learning for detecting ad resources.[7] They build a graph connecting HTML elements with requests and JavaScript behaviors. This graph could possibly be used to place overlays even on the types of elements which our current approach is unable to overlay.

IV. Conclusion

This section will conclude the report with final remarks on the project and its outcome as well as summarize the report.

The projects results partially fulfills the predefined goal of overlaying information on ad elements on a web page. The main drawback identified is that most ads are not possible to overlay. This in combination with the hidden tracker element of a web site serves as a threat to the users privacy and integrity. As long as one considers these drawbacks, the product developed in this project could be used as a base for future research regarding ad servers and the identification of different ad types a user is exposed to during browsing since all ads are shown in the popup list. Future development to the plugin produced in the project could be to sort the generated list based on ad type or generate statistics based on browsing sessions to inform the user of recurring ad types and source domains. To overlay all ads on all pages, another approach than filter list matching is probably better.

References

[1] J. Estrada-Jimenez, A. Rodriguez-Hoyos, J. Parra-Arnau, and J. Forne, "Measuring online tracking and privacy risks on ecuadorian websites." *2019 IEEE Fourth Ecuador Technical Chapters Meeting (ETCM), Technical Chapters Meeting (ETCM),2019 IEEE Fourth Ecuador*, pp. 1 – 6, 2019.

[2] I. Ullah, R. Boreli, and S. S. Kanhere, "Privacy in targeted advertising: A survey." 2020.

[3] Catblock. [Online]. Available: https://getcatblock.com/

[4] ublock origin. [Online]. Available: https://github.com/gorhill/uBlock/

[5] Easylist blocking filters. [Online]. Available: https://easylist-downloads.adblockplus.org/easylist.txt

[6] chrome.webrequest. [Online]. Available: https://developer.chrome.com/docs/extensions/reference/webRequest/

[7] U. Iqbal, P. Snyder, S. Zhu, B. Livshits, Z. Qian, and Z. Shafiq, "Adgraph: A graph-based approach to ad and tracker blocking," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 763–776.