Visualizing third-party ad-domain mappings on a webpage

Emil Nilsson, *Email: emini757@student.liu.se*
Lukas Rajala, *Email: lukra972@student.liu.se*
Supervisor: Niklas Carlsson, *Email: niklas.carlsson@liu.se*
Project Report for Information Security Course
Linköping University, Sweden

***Abstract*—This report covers the arms race between ad providers and privacy aware users, presenting the efforts that have been made by us to bring attention to this problem in a more visual way for an average user. We cover the different techniques used for detecting ads and what possibilities ad-providers have to prevent or circumvent them. We also cover the difficulties of attributing a visual element on a web page to a specific ad provider.**

I. Introduction

The ad providers are wreaking havoc on the modern web. While some try to be responsible, others do everything they can to maximize their profits. When this involves knowing the users you can be sure that the most ruthless providers will go into any length to gather knowledge about the users on the web. With this in mind the problem being addressed in this report will be how to track who is tracking you, with a focus on sharing this information to users. There are a lot of similarities between ad detection and malware detection. Both in that it's a race between detection and evasion, but also in that the two main ways of detecting it is either by checking against a list of known targets or by analyzing patterns in the behaviour of the targets.

A. Limitations

This project has a limited time budget to match the amount of university credits given. In total for the project each one of us should allocate 80 hours, including practical project work, writing of the report and presentations of progress. This creates a limit in the scope of this work. The final results will therefore leave some important questions and conclusions unanswered. As a consequence of this there are many references to improvements which can be made in future work.

II. Background and related work

Behind this seemingly quite simple issue there are many challenges to be considered. These sections cover what techniques traditional ad blocking involves and the pitfalls of these techniques for this project. This provides context to what has to be further researched and developed for the proposed issue of visualizing ad-domain mappings.

A. Web ads

An advertisement is a piece of media that tries to persuade the viewer of purchasing a product or service. Ads are produced by companies to be able to reach out to more potential customers and persuade them to buy their product or service which in turn leads to greater profits for the company. [9] The company that creates the ad isn't necessarily the same company that provides the ad, here the ad providers step in. [8] Ad providers are companies that specialises in providing ads to customers. By advertising on the web these ad providers can target users better and show ads that are more relevant to the user. They do this by collecting information about people on the internet and then providing ads based on the information collected, the next section will go into more detail on how they collect this information. When discussing ads in this report we are exclusively talking about online ads, that is ads which are displayed to be viewed in a modern web browser.

B. Trackers and Privacy

To collect data of users browsing the web ad-providers use multiple techniques that are classified as trackers. These trackers are embedded into most websites, trackers are mostly not provided by the owner of the website and are therefore commonly referred to as third-party trackers. [10] These third-party trackers can collect the browsing habits of users such as browsing history and even information such as email adresses could be leaked with these trackers. [4] This is obviously a big concern to privacy conscious users as your information can be collected and sold without your consent or knowledge.

C. Browser extensions

Web browser extensions allow a programmer to modify the behaviour of a web browser, this is done using an API provided by the browser. These are distributed to users over the internet so that a user can install these third party extensions in their own browser. An extension is running in a more privileged environment than normal websites, but still have restrictions on what they can accomplish. [3]

D. Traditional ad-blocking

Ad-blockers are a widely used type of web browser extension, with the purpose of blocking resources identified

TABLE I
NUMBER OF EACH RESOURCE TYPE BLOCKED BY FILTER LISTS. [6]

Resource	# Blocked by Filter Lists	Percentage of Total Blocked
Image	11,584	$\approx 11\%$
Script	67,959	$\approx 65\%$
CSS	9,255	$\approx 9\%$
AJAX	8,305	$\approx 8\%$
iFrame	7,745	$\approx 7\%$
Video	23	$\approx 0\%$

as ads that would otherwise be fetched and shown on a webpage. These resources are commonly categorized to images, scripts (JavaScript), style sheets (CSS), asynchronous requests (AJAX), embedded web pages (iFrames) and video elements. An ad will commonly be blocked by preventing a network request for the resource from being run or by finding an element in the rendered HTML document (DOM) and removing or hiding it. [1]

E. Filter lists

Filter lists are the most common way for ad-blockers to detect ads on the internet. These are manually curated and updated by crowd sourcing on the internet. These are used in almost all popular ad blockers on the internet. [16] A single filter is a rule that tells your browser extension which resources and elements to block. Filter lists are a set of rules include URLs to web resources which are previously known to be ads. They also include patterns in HTML markup (described using CSS selectors) which are known to be specific ads. This allows for a shared format where even the user of an extension can define their own custom patterns to filter. [5]

An analysis of filter lists found that most ad-blocking using filter lists is achieved by blocking resources of the script type. [6] See table I for details.

These resources can be categorized to two different categories, visible resources and invisible resources. Images, iFrames and videos are considered visible, and scripts, CSS stylesheets and AJAX requests are considered invisible. Some resources in the visible category may even be invisible to the user. For example a 1x1 pixel image used for tracking purposes. According to table I, only around 18% of resources found by filter lists can be considered visible, but this also includes resources such as tracking pixels. This creates a problem where you can not simply overlay a resource from a filter list as it is not connected to an area on the screen.

To solve this issue a solution is JavaScript attribution, which means being able trace which script produced a certain visible element on a page. There are several solutions to this problem, including stack walking, Chromium AdTracker, AdGraph [6] and PageGraph [12]. The common issue with these solutions is that they for the most part need to be executed at a lower level than an extension has access to. This makes JavaScript attribution a problem which often solved in the source code of the browser, it's rendering engine and JavaScript engine.

The best solution for JavaScript attribution at the browser extension level is stack walking. [6] This technique involves throwing exceptions and following their stack trace for attribution. This works in some use cases but has its limitations that are easy for an ad provider to work around.

F. Ad-blocking Arms Race

With ad blockers becoming more and more popular the ad providers needed a way to to get their ads through the blockers. This resulted in the start of what we call ad-block detection, ad "obfuscation" and ad injection.

A report by Peter Snyder et al. [16] also shows that filter lists keep growing as more new rules get added than old ones get deleted. This results in large lists where only a small subset is actually useful, reducing the performance of ad blocking. The massive dead weight in these lists are a result of the arms race making many old rules irrelevant.

1) Ad-block detection: Ad-block detection is what it sounds like, websites are looking for ad block and if detected they take action by informing the user to turn of ad block or by disabling certain functionality on the website.[11]

2) Ad "obfuscation": Facebook and reddit are two popular websites that let users share and read posts from other users on the platform. Both of these sites are displaying ads to the users by disguising them as posts and adding them into the users feed.

3) Ad injection: Video streaming sites usually try and provide ads in the video player so that users have to watch an ad before viewing the video. When ad-blockers started blocking these ads the ad providers started injecting the ads into the video stream making it harder to detect. Companies are starting to catch on and targeting their audiences through content creators ("influencers") and getting ad time directly in the content.

G. Artificial Intelligence

To solve one of the largest drawbacks of filter lists and to try and stay ahead of the cat and mouse game a good idea would be to use some sort of AI that could recognise ads. In fact there are multiple studies on this with promising results. These studies suggest solutions that could be used for either detecting and blocking ads or using the AI to update the filter lists to try and stay ahead of the ad providers. To do this they extract features from ads such as text, link, layout and style. The idea behind this is that an ad is often easily recognisable by these features. By training a model on these features would make the ad providers lives harder since they would have to change these features to try and trick the model but while still following regulations that require that ads needs to be disclosed.

1) Perceptual ad detection: A possible solution to finding ads is to detect them using perceptual ad detection as described by Storey et al. [17] This would involve making use of what humans can see on the page. A good example of this would be legally required disclosure of ads such as the text "sponsored" or "ad". If you find this text in a certain way you have likely found an ad. This type of ad detection is hard for ad providers to counter as they are legally obliged to disclose the presence of an ad to users.

H. Existing legislation

Grant Storey et al. [17] propose an approach to ad-blocking where instead of using traditional filter lists, existing legislation is used. This is done as a way to combat the arms race of advertising providers trying to avoid detection by filter lists, challenging the assumption that the arms race between ad detection and ad providers will escalate indefinitely. This is described as perceptual ad blocking which ignores traditional methods to completely focus on information that would be visible to a human because of legislation.

This shows a key difference in malware detection and ad detection, which is that as ads are legal, they most likely follow legal frameworks. Although this might be useful, we can assume that the worst offenders when it comes to malicious ads are the ones who would be less likely to follow legislation.

III. Methods

The goal of the project is to create a web browser extension to visualize ad and tracking information on web pages. The ad information shall be mapped to the individual ads so that a user of the extension gets a quick overview of the source of an ad. The extension shall primarily be developed for running in the Firefox web browser, although the technical framework behind the browser extension is not in focus in this report. The main challenges with this project are detecting which parts of websites are ads, and which ad provider has provided a specific ad. This problem comes with two main challenges. It may for example be trivial to find that a script from a certain ad provider is being loaded, but this does not give any information of what visual elements the script will produce. On the contrary, finding an element and deducing which script created it is also difficult.

To address the goal of this project, we will firstly discuss our basic prototype, and secondly offer theoretical insights in how this prototype could be improved by discussing different methods for further development.

A. Prototype

To solve this problem we have decided to create a prototype extension, named B AdBoy, as a proof of concept for solving this issue. The practical method that is applied in our prototype is using filter lists of known ad patterns. This is done by using common filter lists and filter list parsing to find relevant ads. To be able to quickly make use of these filter lists, with an efficient parser, the core components of existing open source ad blocking software are used. After searching the web for relevant open source projects we decided to use the AdBlock Plus Core, by forking the webext-sdk repository. [2]

We decided to use two common filter lists in the prototype, EasyList and Peter Lowe's list. These lists were chosen based on their popularity and have not been thoroughly evaluated or compared to others. The filter lists are added using AdBlock Plus Core's built in filter list subscription functionality.

As the blocking part of the AdBlock Plus Core is not of interest in this application we disabled the network request blocking and element hiding code. This was then replaced with our own code for overlaying ads. The overlaying part

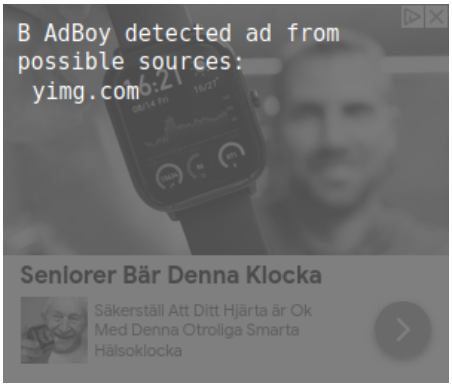

Fig. 1. Example of an overlaid ad with source attribution.

gets a large amount of CSS selectors from AdBlock Plus Core after the page has loaded and queries the document for these using the built in browser APIs. This gives a small amount of target elements that are classified as ads. To avoid double overlaying, we remove target elements that have any other target element as an ancestor node in the DOM. Resulting in only the outermost target elements being kept. The querying is periodically rerun to find newly added ads, for example if a webpage has infinite scrolling or if a new page is loaded in a single page application.

The remaining set of targeted elements each get a CSS class name and also a child element for the overlay and its contents. We then add mutation observers to all the targeted ads, which detect any changes to the HTML of the target or its descendants.

The ad overlay gets the detailed information about ad sources by searching all of its descendants for their "src" attributes, which results in a list of URLs to external content such as images, iframes and scripts. These urls are further parsed to only include the main part of the domain name, excluding subdomains, according to Mozilla's Public Suffix List [14], and duplicates are removed. The overlay is recreated any time the ad mutates, by listening to the mutation observers.

Finally the extension ads some CSS styles to the head of the page, targeting the ad and overlay. To improve user experience the ad gets a grayscale and opacity filter to move user focus from the ad to the overlay. The result of this is shown in figure 1.

B. Theoretical

As a follow up to the prototype we also aim to theoretically analyze the effectiveness of our prototype with the following questions in consideration.

- How relevant is the content of the filter lists to our project?
- How can we detect more ads?
- Are the sources shown in the overlay accurate?
- How does our prototype perform in the arms race?
- What is the user experience for a user of the extension like?

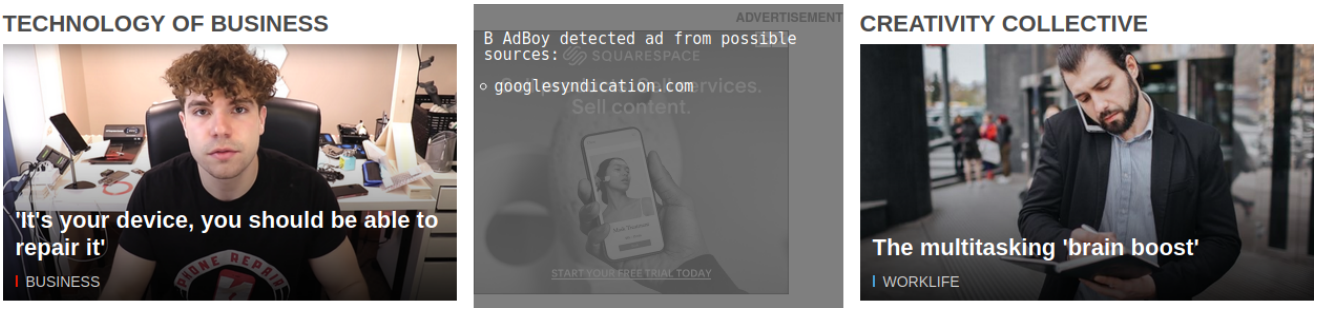

Fig. 2. Showcasing the presence of an overlaid ad in the news feed of bbc.com.

IV. Result and analysis

This section aims to evaluate our results, including what our prototype does well, what could be improved and what technological limitations affect the prototype. We will connect these results to the theoretical questions described in the methods.

The B AdBoy Prototype has shown to be effective at overlaying ads on many web pages. Without optimizing the tool for a specific website or ad provider it can detect different types of ads on different websites. An example of it in action on a news website can be seen in figure 2.

With ad detection based on filter lists no manual work has to be done to keep the extension up to date, as it automatically fetches current filter lists using AdBlock Plus Core. This provides a solution that will stay updated without requiring additional work from us as developers. It is versatile enough to display ads served both from third parties and first parties (as seen in figure 3).

Fig. 3. Ad served from a first party host on reddit.com.

A. Evaluation of Filter Lists

As the AdBlock Plus Core and the filter lists are designed to block ads, the workings of the forked project are not completely in line with what we want. Many filters from the filter lists are targeting scripts, which we simply ignore. While filter lists have been proven to be very useful for completely blocking ads, they have their drawbacks in allowing visualization for the user. This is because many ads on today's web have made great efforts to avoid the most simple forms of detection. This has resulted in a majority of ads being distributed as scripts, while very few ads are just a simple image or iFrame embedded in a page, as seen in table I. A script can easily be blocked from running at all, but make it a lot more difficult to analyze what ads these scripts actually produce. This makes current filter lists very useful for detecting which ads exist, but not very useful for detecting the behaviour of these ads.

When we disabled script blocking, we noticed that the elements that the script produced were also elements that the filter lists target. This proved to be useful as many ads could still be found this way. Although this was the case, it catches far from every ad out there. This enforces our belief that filter lists could be better optimized for ad visualization instead of ad blocking, by creating purpose built filter lists.

1) Filter list optimization: With the drawbacks of the current filter lists, one approach would be to generate new filter lists specialized for the purpose of visualizing ads instead of blocking them. This could be done by curating filter lists for this purpose by crowd sourcing, as is already done by popular ad blocking filter lists.

Another promising solution would be using artificial intelligence to generate more optimized filter lists. This has been done by Sjösten et al [15] for the purpose of serving localized filter lists to underserved regions of the world. They combine usage of PageGraph for brower instrumentation with an ad classifier into their filter generation pipeline to generate new filter lists to complement existing filter lists. This research might prove useful in the future for creating better filter lists optimized for visualizing ads.

An important aspect to consider is the performance of large filter lists. As some websites have shown to slow down with B AdBoy enabled, importance should be put in removing unused filter rules. This has previously been discussed by Snyder et al. [16]. Besides measuring the usage of specific filter rules across the web and removing the most uncommonly applied filters. Filter lists could also be pruned, to only include the rules which are compatible with the ad visualization technology in use, to avoid unnecessary performance overhead.

B. Evaluation of ad detection

After manually testing our extension on popular websites we see that most ads are indeed overlaid correctly and many also provide a possible source for the ad. However some ads

are missing the sources. We have also found that our extension misses ads to overlay meaning that we don't have a perfect detection rate of ads. The magnitude of this issue is hard to measure since we can't use any prebuilt tests for ad-blocking and we would need to build a new test suite for testing our solution.

C. Evaluation of Displayed Sources

As seen in figure 4 the quality of the displayed sources may vary. Here there are three different domains serving external resources inside of the ad. In this case the logo images are hosted on imgur.com, a tracking pixel is from stackoverflow.com and a hidden iframe is served from googlesyndication.com.

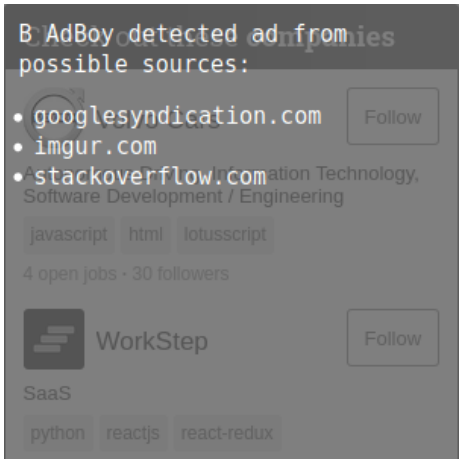

Fig. 4. Ad on stackoverflow.com displaying multiple sources.

For our source finding algorithm it is impossible to know which one of these domains, if any, is serving the actual ad.

This algorithm for finding the source of an ad is a severe flaw in our implementation that would require additional work in the future. This is due to many ads not having descendants with a "src" attribute corresponding to the provider. Instead the "src" could be for example an image, which is displayed in the ad, but unrelated to the provider of the ad, as seen with imgur.com in figure 4. The browser API api also blocks the traversal of iframes, due to security concerns, resulting in the inability traverse into an iframe. This can be solved by running a script from the extension inside all iframes as well. By doing this the sub documents can send information to the main document through the background task of the extension. Some ad providers use iframes without providing a "src" attribute, making our extension fail to find a connected domain. From our experience Google does this with some of their ads. This issue would be solved by further researching JavaScript attribution instead of working with the assumption that ads would serve some external files under the provider's domain.

JavaScript attribution using stack walking could be implemented in the browser extension in the future, but has been proven to be easily defeated by ad providers looking to avoid detection.

Many of the cutting edge research papers have chosen to go beyond the level of a what a web browser extension is capable of through the browser's provided extension API. [6] [7] [4] This is done to achieve a more optimal result than has previously been accomplished by web browser extensions. While going outside the scope of a web browser extension does not help with the final delivered product in this project, it could prove to be very useful for delivering additional data to our extension.

An option which shows a lot of potential for future work is combining these solutions. For example by using B AdBoy together with PageGraph as a web crawler [13]. This would create an environment where B AdBoy could use working JavaScript attribution to automatically build a database of relations between sources and DOM elements. This database could then be accessed by B AdBoy users to recive more accurate source attribution.

D. Evaluation of the Arms Race

Unfortunately our solution isn't the solution to the arms race. However we can see some benefits and drawbacks with our solution. The benefits are that ads are not actually blocked and that ad providers don't need to take up arms against our extension since their ads are still displayed to the users. Another benefit is that our solution isn't targetable by traditional Ad block detection. However there are some obvious drawbacks to our prototype the main drawback is that ad providers are in constant fights with ad blockers and therefore also in fight with our filter lists that are used in the prototype. This means that we rely on filter lists that need to be updated every so often. Also if the ad providers really wanted to they could easily detect our overlaying and try and interfere with it. So as we can see our prototype is far from ending the arms race, our solution is a way for ads to exist on the web but also show a behind the scenes view of all the providers involved in bringing you the ad. This would hopefully encourage users to make better decisions and for ad providers to take a better path in providing ads without stalking the user all over the web. The following subsections will cover the future improvements in this area.

1) Proposing new legislation: As described in the background ads often have to be disclosed according to existing legislation. This makes it possible to for example search for the word "sponsored" or "ad" in the webpage. These legislations are a good start to providing transparency for ads on the web, but can be considered not enough. New legislation could be brought forward so that users could browse the web freely without being tracked, or at least be fully aware of the tracking.

Although this report will not go into the area of proposing new legislation it is still a valid option. However a potential drawback is that a feasible assumption would be that the ad providers who badly follow legislation would be the ones who are most likely the most malicious or intrusive providers. The website might also follow other local laws than the user so while the European Union or United States of America might have strong laws that a lot of websites follow, many websites are hosted for audiences in other parts of the world.

E. Evaluation of User Experience

A normal ad-blocker partially aims to increase performance by removing network requests and stopping script execution. Our prototype does not in any way increase performance, as it does not block these things. This means that if anything it is an added performance penalty for the user. The evaluation of how large the decrease in performance is will not be done in this report, but it could prove to be significant on a webpage with a complex DOM tree.

Applying a gray scale filter to the ads will bring the users attention away from the ads and to the actual content of the website. We see this as an improvement as it doesn't take away focus from the website that you are visiting and will let you focus on whats important first.

In the case where there is a false positive overlay or an ad a user actually wants to see, the overlay should be dismissable. This has not been implemented in the prototype as it would take resources from more important work. In the scenario where this prototype should be developed into an extension distributed to users, this requirement should get increased priority. The ability to permanently remove a specific overlay from a website should then also be included.

False positives are not as sensitive for our extension compared to an ad blocker, since no content is being blocked or hidden. In the worst case of getting a false positive, the overlay from our extension would be dismissable.

Our extension is also missing a graphical user interface (GUI), as this has been downprioritized. If the browser extension were to be distributed, a user should using a GUI be able to configure options such as filter lists, filter exceptions, and visuals of the overlay. The GUI should also show the user hidden trackers on the web page that can not be overlaid as they are not visible on the page.

V. Conclusions

There are many ways to detect the presence of ads on a web page. But many of those do not take into account where on the page an ad is displayed. This makes for a challenging problem to solve. Our prototype solves the task of visualising some of the ads and ad-providers on the web. It does this by using the traditional filter lists to find elements which are considered ads. We then dig deeper into the elements and their children to try and find the sources of who has distributed them.

This technique is not flawless and relies heavily on filter lists. In this report we have discussed improvements that could be made. Some are possible to solve by continuing purely using an extension. While others are outside of the scope for an extension, but could be useful for creating external data sources for the extension.

The browser extension is still an early prototype, lacking functionality in many areas such as testing, optimization, ad detection, source attribution and user experience. Although this is lacking, the report aims to provide insights in how all of these areas can be improved in future revisions of the extension.

There is no data indicating how the prototype performs in the amount of ads found and the reliability of the source attribution. In the case where further development of the extension is to be done, we recommend testing and comparing the amount of ads successfully found to that of an ad blocker, and that the source attributions are compared to those of a tool such as PageGraph.

While there is obviously much left to do before the product is perfect, we believe that what we have achieved are great first steps into the area of ad visualization web browser extensions.

References

[1] *Ad blocking*. URL: https://en.wikipedia.org/wiki/Ad_blocking (visited on 04/27/2021).

[2] *Ad Blocking Web Extension SDK*. URL: https://gitlab.com/eyeo/adblockplus/webext-sdk (visited on 05/04/2021).

[3] *Browser Extensions*. URL: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions (visited on 04/27/2021).

[4] Steven Englehardt and Arvind Narayanan. "Online Tracking: A 1-Million-Site Measurement and Analysis". In: CCS '16. Vienna, Austria: Association for Computing Machinery, 2016, pp. 1388–1401. ISBN: 9781450341394. DOI: 10.1145/2976749.2978313. URL: https://doi.org/10.1145/2976749.2978313.

[5] *How to write filters*. URL: https://help.eyeo.com/en/adblockplus/how-to-write-filters (visited on 04/27/2021).

[6] Umar Iqbal et al. "AdGraph: A Graph-Based Approach to Ad and Tracker Blocking". In: *2020 IEEE Symposium on Security and Privacy (SP)*. 2020, pp. 763–776. DOI: 10.1109/SP40000.2020.00005.

[7] B. Li et al. "JSgraph: Enabling Reconstruction of Web Attacks via Efficient Tracking of Live In-Browser JavaScript Executions". In: *NDSS*. 2018. DOI: 10.14722/NDSS.2018.23319.

[8] Zhou Li et al. "Knowing Your Enemy: Understanding and Detecting Malicious Web Advertising". In: CCS '12. Raleigh, North Carolina, USA: Association for Computing Machinery, 2012, pp. 674–686. ISBN: 9781450316514. DOI: 10.1145/2382196.2382267.

[9] Rima Masri and Monther Aldwairi. "Automated malicious advertisement detection using VirusTotal, URLVoid, and TrendMicro". In: *2017 8th International Conference on Information and Communication Systems (ICICS)*. 2017, pp. 336–341. DOI: 10.1109/IACS.2017.7921994.

[10] Johan Mazel, Richard Garnier, and Kensuke Fukuda. "A comparison of web privacy protection techniques". In: *Computer Communications* 144 (2019), pp. 162–174. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2019.04.005. URL: https://www.sciencedirect.com/science/article/pii/S0140366418300604.

[11] M. Mughees et al. "A First Look at Ad-block Detection: A New Arms Race on the Web". In: *ArXiv* abs/1605.05841 (2016).

[12] *PageGraph*. URL: https://github.com/brave/brave-browser/wiki/PageGraph (visited on 04/27/2021).

[13] *pagegraph-crawl*. URL: https : / / github . com / brave - experiments/pagegraph-crawl (visited on 04/27/2021).

[14] *Public Suffix List*. URL: https://publicsuffix.org/ (visited on 05/04/2021).

[15] Alexander Sjösten et al. "Filter List Generation for Underserved Regions". In: *Proceedings of The Web Conference 2020*. WWW '20. Taipei, Taiwan: Association for Computing Machinery, 2020, pp. 1682–1692. ISBN: 9781450370233. DOI: 10.1145/3366423.3380239. URL: https://doi.org/10.1145/3366423.3380239.

[16] Peter Snyder, Antoine Vastel, and Benjamin Livshits. *Who Filters the Filters: Understanding the Growth, Usefulness and Efficiency of Crowdsourced Ad Blocking*. 2020. arXiv: 1810.09160 [cs.CR].

[17] Grant Storey et al. *The Future of Ad Blocking: An Analytical Framework and New Techniques*. 2017. arXiv: 1705.08568 [cs.CR].