Extending the Exposure Score of Web Browsers By Incorporating CVSS

Fadi Mohsen[1][0000−0003−3876−5781], Adel Shtayyeh[2], Riham Naser[2], Lena Mohammad[2], and Marten Struijk[1]

[1] Information Systems Group,
Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence,
University of Groningen, The Netherlands
Email: f.f.m.mohsen@rug.nl
[2] An-Najah National University, Palestine

Abstract. When browsing the Internet, HTTP headers enable the client and server send extra data with a request or response such as the User-Agent string. This string contains information related to the sender's device, browser, and operating system. Yet its content differs from one browser to another. Despite the privacy and security risks of User-Agent strings, very few works have tackled this problem. A recent work considered giving Internet browsers relative scores to aid users to choose less intrusive ones. The objective of this work is to extend their score by: first, conducting a user study to identify its limitations. Second, extend the exposure score via incorporating data from the NVD. Third, we provide a full implementation, instead of a limited prototype, for assigning scores to users' browsers when they visit our website, suggesting alternative safe browsers, and updating the back-end database with a click of a button. We applied our method to a data-set of more than 52 thousand unique browsers. Our performance and validation analysis show that our solution is accurate and efficient.

1 INTRODUCTION

Web browsers are programs that allow you to search for and view the content of the World Wide Web [7]. Though, these browsers do more than just simply rendering HTML (Hypertext Markup Language) pages and displaying the results. They enable users to use search engines, make online purchases, communicate with each other using social media sites, and much more [19]. However, there are issues related to maintaining the privacy of users and the security of their devices while surfing the web using these programs. These issues can possible result in compromising user's devices and access their personal data such as browser history and auto-fill information [14] [2]. Vulnerable browsers give attackers the opportunity to exploit their security gaps to steal information, delete files, and other malicious activities [21]. Executing such attacks is normally proceeded by collecting detailed information about the target. For example, the version of the operating system, the type of the browser type, and the version of the hardware.

Each browser has its own distinctive User-Agent request header [9]. The User-Agent request header exposes information about the software being used, the operating system and the installation of certain plugins [15]. This information can also be leveraged to track user activities on the web via a process called fingerprinting [12]. Fingerprinting is the process where the combination of fields exposed by the browser leads to an (almost) unique combination [3]. Although most people are aware of the role of cookies in fingerprinting and tracking users across the internet, the use of the User-Agent request header is relatively unknown. Users are not even asked to share this information. The work of Mohsen et al. [15] pointed out the privacy and security risks of User-Agent request header because of the amount and the sensitivity of the information it exposes. They proposed a new technique to quantify the exposure of Internet browsers. The technique is merely based on the information items that exist in the User-Agent request header. As a proof of concept, they implemented a simple prototype to demonstrate how such a technique could be useful in protecting the privacy of users. Thus, in this work, we are primarily extending their technique by incorporating the Common Vulnerabilities and Exposures (CVE) of a browser. CVE is a list of publicly disclosed computer security flaws. We rely on the Common Vulnerability Scoring System (CVSS) to calculate the severity score of each CVE record. We then aggregate the severity scores of all CVE records of a browser. The resulting severity score and the relative score Mohsen et al. [15] are then merged together. The resulting score is the final exposure score of the browser.

The following are the research contributions:

- Conducting a user study to explore the limitation of an existing method for quantifying the exposures of web browsers.
- Extending the technique that was proposed by Mohsen et al. [15] to calculate new exposure score for Internet browsers.
- Conducting a comparative study between the various security and exposure scores.
- Providing a full implementation - online tool that uses the resulting exposure scores to suggest alternative privacy-preserving Internet browsers.
- Conducting an evaluation and validation studies on our tool.

The PHP source code, the database schema, and the final data set are made publicly available [22].

The rest of the paper is organized as follows. In Section 2 we discuss the design and the results of the user study. In Section 3 we go briefly over the methodology of the previous technique then we move to talk about our extension. In Section 4 we discuss the implementation details. In Section 5 we go over the related works. Finally, we conclude the paper with Section 6.

2 User Study

The aim of this user study is to study whether the tool that was proposed by [15] is appealing to the end users, and if not, obtain the list of new features

that need to be added to the tool. As such, this study raises three main research questions:

- Are users aware of the data that is being exposed by their browsers?
- How likely are they willing to use such a score-based system?
- Are there any vital features missing from this system?

2.1 Survey design

We created a custom survey that prompts participants to answer a set of questions before and after showing them the exposure scores of their browsers. The custom survey collects also the User-agent strings of participants after taking their consent. The survey's questions were also split into multiple pages to make participation less overwhelming. The survey starts with an introduction followed by the demographic questions. After that comes in a set of questions concerning information privacy and Internet browsers. The exposure score report is then shown to the user alongside questions pertaining the exiting scoring system and ideas to improve it.

During the survey, participants get to see their parsed User-agent string and the exposure score of their browsers, a list of alternative browsers, and a unique token. The unique token is connected to their survey answers in case they decided to have them removed from the data set.

2.2 Demographics

Nearly, half of the users who visited the study URL managed to complete it. In total, 115 participants has answered all the questions successfully. The average time of completing the survey was eight minutes. More than 85% of participants were aged between 15 and 25: 15-20 (29.6%) and 20-25 (55.8%). About 32.2% of participants were female and 67.0% were male. The education and technical experience of the participants can be noted as relatively high. The distribution of participants over the age groups, education, and technical skills should be taken in consideration before generalizing the results.

2.3 User Awareness

The majority of participants disagreed with the statement that their browser did not share any data about them. However, they indicated not to know what data their browser shared. After showing them their parsed User-agent strings, participants were asked if they were aware such data was being exposed. Our analysis showed that the technical skills of a participant plays an important role in her awareness of the data exposure. Participants with high technical skills were less surprised by the exposed data in comparison to participants with low technical skills.

2.4 Likely to Use

The results show that users are not very likely to use the current score-based system. Yet, there seems to be no overwhelming majority for either side, which gives us a window to improve the current system. Moreover, some participants find that the current score-based system uses technical terms that is hard for some users to understand. Others have stressed the need to add more information as to how the score is calculated.

2.5 Key Features

Our analysis shows that the browser UI and page loading speed are the most important features for users when choosing a browser. The plugin support, startup speed and privacy comes after that. The Least important feature was if a browser was pre-installed on the machine. Although people do consider privacy, it is not the most important feature that people will base their browser choice on.

2.6 Identifiable

Out of the 115 User-agent strings that we collected from participants, 53% of the users had a unique User-agent string. The User-agent string that appeared the most was generated by Chrome browsers running on windows 10. They appeared 19 times. Closely followed by Safari running on iOS 14.4.2 and Firefox on windows. They both appeared 11 times. A close look at the collected User-agent strings, we found the followings:

- Android devices add the product code of the device being used to the User-Agent.
- IPhone's generally generates the same User-agent string regardless of the phone's type as long as they are on the same iOS version and use Safari.
- Instagram and other mobile apps use an in-app browser with a custom User-agent header (for both iOS and Android).

Participants who are using an Android device had a unique User-agent string because the product code of the device is embedded in the User-agent string. Both Chrome and the default browser for Samsung expose this information. Firefox and Brave do not.Why such detailed information is included is unclear. It resulted in 29 android agent strings in being completely unique. iOS devices do not suffer from this issue.

3 Methodology

In this section, we will discuss our methodology in generating a final exposure score for Internet browsers. The final score is a combination of two scores; thus, we will start by briefly talking about the first one, the relative score. After that, we will discuss the second score, CVSS score.

3.1 Relative Score

The relative score of Mohsen et al. [15] is based on the amount of information that are revealed by the browser's User-agent string in relative to the other browsers in the seed data set. The equation shown in equation 1 shows the exposure score of browser i based on all j's, the attributes that were contained in the User-agent string. For each element j, the sensitivity and visibility scores and constants are calculated based on other equations, which we do not show here. For further information about this equation, we refer to [15].

$$EXP(i) = \sum_j EXP(i,j) = \sum_j (S(j) + S_j) \times (V(i,j) + V_j) \quad (1)$$

3.2 CVSS Score

The National Vulnerability Database (NVD) is currently supporting CVSS Version 3.x [17] and CVSS Version 2.0 [18] [1]. Thus, a particular vulnerability can have at most two scores. However, since the CVSS 3.0 is not yet complete, many vulnerabilities will only have one score, the CVSS version 2.0. As such, we decided that the CVSS score of a vulnerability is the average of two scores if both existed, otherwise, it is the CVSS 2.0 score.

3.3 Final Exposure Score

In Equation 2 we show our methodology in calculating the final exposure score for browsing software. The final exposure score equals the normalized relative score plus the CVSS score divided by 2. The CVSS score is the average of the two scores. In case the CVSS version 3.x is missing, the CVSS score becomes the CVSS 2.0 score. Originally, the relative score has no maximum value. In order to combine it with the CVSS score, we had first to normalize it.

$$FIN\_SCORE(i) = NORM(REL\_SCORE(i) + [AVG(CVSS2, CVSS3)|(CVSS3)]/2 \quad (2)$$

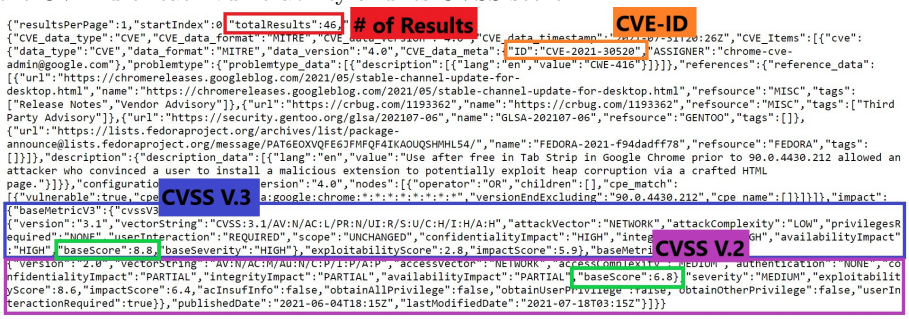
3.4 Dataset

We used the data set of [15], which contained over a million User-agent string. Our evaluation of this data set revealed two important points. First, the data set contained a lot of duplicated records. Second, a good number of these records were for old browser versions. Thus, we removed all duplicates and then looked online for newer data sets, that contain more recent versions of the existing browsers. The result is a new data set, that contains 52,000 unique browsers. Each record in the final data set is composed of 51 columns. Forty-seven columns are for the different attributes that we can possibly retrieve from the User-agent string. Two columns are for the CVSS and the relative scores. The last two columns are for keeping track of when was the two scores were updated.

3.5 Final Data set Summary

Nearly, 89% of the User-Agent strings are coming from a Browser, the other 11% are divided among other software types such as Application 6%, Bot/Crawler 3%, Email Client 2%. As per the device types, 45% mobile phones, 33% desktop, 19% tablets, and the other 3% are unknown devices. Regarding the distribution of the platform makers, it comes as no surprise that the top three most used platforms were Google, Microsoft and Apple, since 45% of the devices are mobile phones. Finally, the distribution of the used rendering engines shows that Blink is the most used, with 54% share, WebKit 19%, Gecko 13%, Trident 3%, on Presto Opera 3%, and UCWeb 3%.

Fig. 1. A snippet of an NVD Json file. It contains the number of vulnerabilities, and the CVE id of each vulnerability and its CVSS score.



4 Implementation

As a first step, we calculated the relative scores for all the records in the final data set according to [15]. We then calculated the CVSS scores, which took a lot of time because we had to crawl the information from the NVD website. Finally, the final score, exposure score, is derived from the previous two scores. The following is an overview of the score-based system that we developed. It uses the aforementioned scores to recommend safer browsing options to the users. The following are the steps that make up the whole procedure:

- Extract and process the User-Agent string
- Calculate the relative score of user's browser.
- Calculate the CVSS score of user's browser.
- Present the report including the scores and alternative browsers.
- Update the final data set.

We will go through these measures in significant detail.

Extracting and processing the User-agent string In this step, the User-agent string is extracted from the requests coming into our system. Then, the User-agent string is parsed to extract the 47 features. Next, the final data set is searched to find a match based on the 47 features. The result of this search can be summarized in the following cases:

- Best-case scenario: a match is found, and both scores are present.
- Average-case scenario: a match is found, but one of the scores is missing.
- Worst-case scenario: a match is not found.

Calculating Relative Score This step is needed in the worst-case scenario. It is also needed in the average_case scenario when the relative score is missing. For the latter case, the relative score is calculated and the browser's record in the final data set is updated accordingly. However, calculating the relative score for a requesting browser that does not exist in the final data set is a quite cumbersome process. This is because it depends on all browsers in the database. It entails updating some of the terms and constants in the original equation [15] such as n and $|R^j|$. Moreover, it requires updating the relative scores of all browsers in the database. For simplicity and efficiency reasons, the current score-system temporarily updates the terms and constants to calculate the relative score of the requesting browser. The relative scores of existing browsers are not updated. The relative and CVSS scores are calculated for the new browser. The new browser will also be added to the final data set including the scores.

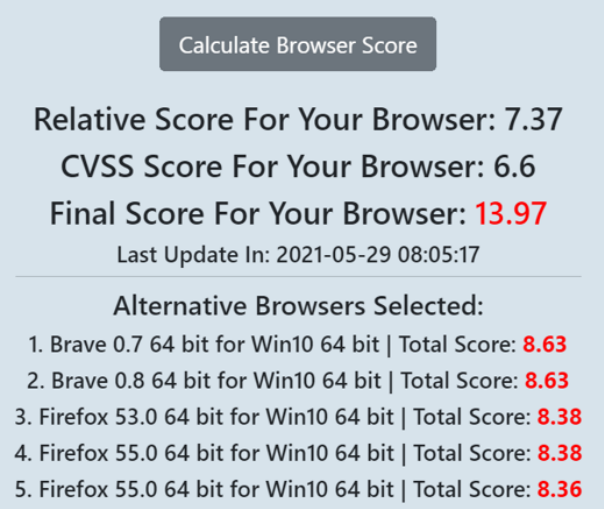
Calculating the CVSS Score This step is needed in the worst-case scenario. It is also needed in the average_case scenario when the CVSS score is missing. The CVSS score is calculated by first searching the NVD website for related vulnerabilities. The search is conducted based on three keys: the browser name, the browser version, and the platform. The NVD website sends back the results as a JSON file. The file contains several information, most notably the number of vulnerabilities, the ID of each vulnerability and the base metric that contains the CVSS score in its two versions. The file shown in Figure 1 was returned after searching for the following keys: Chrome, 90.0, and Win10. It shows that there are 46 distinct vulnerabilities linked to this particular browser. The CVSS scores for one of these vulnerabilities are highlighted.

The Json file is then parsed into two dimensional array. For each vulnerability, the final CVSS score is calculated by either averaging both scores or considering one of them if the other is missing. The final CVSS score of a browser would then be the average of all these final CVSS scores.

Calculating the Final Exposure Score The final exposure score for each browsing software in our final data set is calculated using Equation 2 of Section 3.3. The maximum possible value for this score is 20. The relative and CVSS scores contributes evenly to this score with 10 each. Browsers with lower

scores are better for users because they reveal less information and has less vulnerabilities. In order to understand the relationship between the CVSS score and the relative score of a browser, we calculated the correlation coefficient. It meant to measure the degree of linear association between the two continuous variables. The correlation value was 0.18, which is considered weak. Thus, merging these two scores is considered advantageous as it gives us a better representative score. Otherwise, one of these scores would've been enough.

Fig. 2. The current implementation returns a list of alternative browsers with lower final exposure scores that also fit the user's device specifications. Due to the space limit, we could not show the entire GUI.



Displaying Scores and Suggestions In this step, a report is displayed to the user pertaining her browsing software. The report contains information such as a relative score, CVSS score, final score, last updated, and alternative browsers. It also shows all the attributes that the current browser reveals. Additionally, it provides a description of the scores and their privacy implications.

In Figure 2, the final exposure score of the user's browser, which is *Chrome 90.0 on Windows 10 64bit*, is 13.97 out of 20, the relative score is 7.37 out of 10, and the CVSS score is 6.6 out of 10. The browsing software reveals numerous attributes such as platform, device type and device name.

4.1 Update: Admin Portal

There are three reasons to why we need to occasionally update the final data set. First, to update the CVSS scores due to the discovery of new vulnerabilities

that get added to the NVD database. Second, to add new User-agent strings to the data set. Third, to update the relative scores. The update of the relative scores is necessarily due to the changes occur on the values of the terms and constants in the original equation [15]. The changes in these values happen due to the addition of a new browser to the final data set as explained in Section 4. However, executing any of these processes is computationally expensive. Which means that they should not be executed frequently, like every time a new request comes in. As such, we added an admin portal to the score-based system. The portal provides three manual options pertaining to the aforementioned points. The first two options though can be reprogrammed to run automatically at a specified time and date.

4.2 Testing Environment

The system is tested on a local web server using XAMPP v3.2.4 for Windows 10 Pro v20H2. The system is created using PHP v8.0.0, the final data set is stored in MySQL 8.0.0, and phpMyAdmin v5.0.4. The experimental platform is based on an Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 6 Cores and 12 Threads with 16GB memory, and the GPU 1660ti. The PHP source code, the database schema, and the final data set are made available online [22] for researchers

5 Related Works

Our work is considered an improvement over the work of Mohsen et al. [15]. In their work, a new formula was introduced to measure the privacy exposure of web browsers. The formula considered only the information that are included in the User-agent string. In addition, their implementation was basic and incomplete, and their data set contained numerous duplicate entries. Thus, in this work, we first extended the exposure formula to take into account the browsers' vulnerabilities, cleaned the data set and added new records, and provided full implementation. The aim of this work and Mohsen et al. is to counter user identification and tracking through device fingerprinting. Device fingerprinting was first studied by Peter's [3]. In his work, modern web browsers were tested in order to determine whether they can be fingerprinted or not using the information that they disseminate while browsing the Internet. One of their key findings was that browsers reveal too much information, which makes them trackable. Takeda proposed a number of techniques to identify the owner of a digital device [20]. One of these techniques was based on analyzing the browsers' fingerprints such as: HTTP Accept Header, Browser Plugins, System Fonts and Screen size and color depth. Yen et al. [23] carried on a large-scale study on month-long anonymized datasets that were collected by the Hotmail web-mail service and the Bing search engine. Their results show that User-Agent strings can effectively be used to identify hosts on the Internet. The identification accuracy can also be significantly improved if combined with the IP address of the host. Kaur et al. [11] proposed a web browser fingerprinting technique that works despite the security devices and

measures that are normally deployed at the corporate network boundary such as VPNs, proxy servers and NAT. Laperdrix et al. [13] demonstrated the effect of the recent innovations in HTML5 on increasing the accuracy of fingerprinting. They also showed that browser fingerprinting on mobile devices is highly possible and effective similar to personal computers. On the other hand, Hupperich et al. [10] found that existing tracking techniques do not perform well on mobile devices; thus, they proposed several features that tracking systems could leverage to fingerprint mobile devices. Martin et al. [16] was able to identified web browsers using the underlying JavaScript engine. As far as the preventive measures, Laperdrix et al. [13] explained different ways to reduce the possibility of fingerprinting, such as removing plugins and using regular HTTP headers. Martin et al. [16] leveraged their proposed browser fingerprinting technique to prevent session hijacking attacks. In addition, there were a number of proposals to counter the privacy threat of browser fingerprinting and tracking users [4] [5] [8]

6 Conclusion and Future work

In this paper, we first conducted a user study to identify the limitations of an existing method for calculating relative scores for web browsers. We then extended the method by incorporating the browser's vulnerabilities data that are extracted from the National Vulnerability Database (NVD). We also provided a full web implementation for our approach, in which the relative score of the user's browser is calculated on the fly, then a list of alternative safe browsers is shown to the user. our implementation is based on a seed data set of over 52 thousand unique browsers along with their vulnerabilities. Furthermore, we updated the data set constantly based on the visits and the changes that happen on the NVD records. Our validation and performance analysis of our approach showed that it is accurate and efficient. For instance, the time to get a score takes 0.85 seconds if this browser is existing in our database. If this browser is new, it takes 6.16 seconds to calculate the final score. And updating the entire data set takes 1.82 minutes. We plan to improve our method for future works by incorporating the CVSS temporal metrics and the CVSS environmental metrics [6]. The PHP source code, the database schema, and the final data set are made publicly available [22].

References

1. Aksu, M.U., Dilek, M.H., Tatlı, E., Bicakci, K., Dirik, H., Demirezen, M.U., Aykır, T.: A quantitative cvss-based cyber security risk assessment methodology for it systems. In: 2017 International Carnahan Conference on Security Technology (ICCST). pp. 1–2 (2017). https://doi.org/10.1109/CCST.2017.8167819
2. Barona, R., Anita, E.A.M.: A survey on data breach challenges in cloud computing security: Issues and threats. In: 2017 International Conference on Circuit ,Power and Computing Technologies (ICCPCT). pp. 1–8 (2017). https://doi.org/10.1109/ICCPCT.2017.8074287

3. Eckersley, P.: How unique is your web browser? In: International Symposium on Privacy Enhancing Technologies Symposium. pp. 1–18. Springer (2010)
4. FaizKhademi, A., Zulkernine, M., Weldemariam, K.: Fpguard: Detection and prevention of browser fingerprinting. pp. 293–308 (07 2015). https://doi.org/10.1007/978-3-319-20810-7$_2$1
5. Fiore, U., Castiglione, A., Santis, A.D., Palmieri, F.: Countering browser fingerprinting techniques: Constructing a fake profile with google chrome. In: 2014 17th International Conference on Network-Based Information Systems. pp. 355–360 (Sep 2014). https://doi.org/10.1109/NBiS.2014.102
6. First: first.org, https://www.first.org/cvss/user-guide
7. GCFGlobal: Internet basics - using a web browser, https://edu.gcfglobal.org/en/internetbasics/using-a-web-browser/1/
8. Gómez-Boix, A., Frey, D., Bromberg, Y.D., Baudry, B.: A Collaborative Strategy for mitigating Tracking through Browser Fingerprinting. In: MTD 2019 - 6th ACM Workshop on Moving Target Defense. pp. 1–12. London, United Kingdom (Nov 2019). https://doi.org/10.1145/3338468.3356828, https://hal.inria.fr/hal-02282591
9. Hoffman, C.: What is a browser's user agent?, https://cutt.ly/DW77C6v
10. Hupperich, T., Maiorca, D., Kührer, M., Holz, T., Giacinto, G.: On the robustness of mobile device fingerprinting: Can mobile users escape modern web-tracking mechanisms? In: Proceedings of the 31st Annual Computer Security Applications Conference. pp. 191–200 (2015)
11. Kaur, H., Zavarsky, P., Jaafar, F.: Unauthorised data leakage from corporate networks through web browser fingerprinting vulnerability
12. Laperdrix, P., Bielova, N., Baudry, B., Avoine, G.: Browser fingerprinting: A survey. CoRR **abs/1905.01051** (2019), http://arxiv.org/abs/1905.01051
13. Laperdrix, P., Rudametkin, W., Baudry, B.: Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. In: 2016 IEEE Symposium on Security and Privacy (SP). pp. 878–894. IEEE (2016)
14. Matteson, S.: 5 common browser security threats, and how to handle them, https://www.techrepublic.com/article/5-common-browser-security-threats-and-how-to-handle-them/
15. Mohsen, F., Shehab, M., Lange, M., Karastoyanova, D.: Quantifying information exposure by web browsers. In: Arai, K., Kapoor, S., Bhatia, R. (eds.) Proceedings of the Future Technologies Conference (FTC) 2020, Volume 3. pp. 648–667. Springer International Publishing (2021)
16. Mulazzani, M., Reschl, P., Huber, M., Leithner, M., Schrittwieser, S., Weippl, E., Wien, F.: Fast and reliable browser identification with javascript engine fingerprinting. In: Web 2.0 Workshop on Security and Privacy (W2SP). vol. 5. Citeseer (2013)
17. NIST: nvd.nist.gov, https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator
18. NIST: nvd.nist.gov, https://nvd.nist.gov/vuln-metrics/cvss/v2-calculator
19. Scientific, F.: Introduction to browsing the web, https://www.freedomscientific.com/SurfsUp/Introduction.htm
20. Takeda, K.: User identification and tracking with online device fingerprints fusion. pp. 163–167 (10 2012). https://doi.org/10.1109/CCST.2012.6393552
21. Wikipedia: Web browser, https://en.wikipedia.org/wiki/Web$_b rowser$
22. xxx: Source code, https://drive.google.com/file/d/1prsenyhYNa0zK8rxY0$_N DMk8OeL-ysFn/view?usp = sharing$
23. Yen, T.F., Xie, Y., Yu, F., Yu, R.P., Abadi, M.: Host fingerprinting and tracking on the web: Privacy and security implications. In: NDSS. vol. 62, p. 66 (2012)