6-30-2023

WebTracker: Real Webbrowsing Behaviors

Daisy Reyes
Loyola University Chicago

Eno Dynowski
Loyola University Chicago

Taryn Chovan
Loyola University Chicago

John Mikos
Loyola University Chicago

Eric Chan-Tin
dchantin@luc.edu

See next page for additional authors

Author Manuscript

This is a pre-publication author manuscript of the final, published article.

Recommended Citation

Reyes, Daisy; Dynowski, Eno; Chovan, Taryn; Mikos, John; Chan-Tin, Eric; Abuhamad, Mohammed; and Kennison, Shelia. WebTracker: Real Webbrowsing Behaviors. 2023 Silicon Valley Cybersecurity Conference (SVCC), , : 1-8, 2023. Retrieved from Loyola eCommons, Computer Science: Faculty Publications and Other Works, http://dx.doi.org/10.1109/SVCC56964.2023.10164930

WebTracker: Real Webbrowsing Behaviors

***Abstract*—With increased privacy concerns, anonymity tools such as VPNs and Tor have become popular. However, the packet metadata such as the packet size and number of packets can still be observed by an adversary. This is commonly known as fingerprinting and website fingerprinting attacks have received a lot of attention recently as a known victim's website visits can be accurately predicted, deanonymizing that victim's web usage. Most of the previous work have been performed in laboratory settings and have made two assumptions: 1) a victim visits one website at a time, and 2) the whole website visit with all the network packets can be observed. To validate these assumptions, a new private webbrowser extension called WebTracker is deployed with real users. WebTracker records the websites visited, when the website loading starts, and when the website loading finishes. Results show that users' browsing patterns are different than what was previously assumed. Users may browse the web in a way that acts as a countermeasure against website fingerprinting due to multiple websites overlapping and downloading at the same time. Over 15% of websites overlap with at least one other website and each overlap was 66 seconds. Moreover, each overlap happens roughly 9 seconds after the first website download has started. Thus, this reinforces some previous work that the beginning of a website is more important than the end for a website fingerprinting attack.**

***Index Terms*—Webbrowsing, Website Fingerprinting, Anonymity, Privacy**

I. Introduction

Privacy is becoming an increasing concern for many. VPNs, Tor, and DuckDuckGo are increasingly being used. Moreover, many companies such as Apple and Brave are touting more private solutions. Users thus want to hide their browsing activities so that they cannot be tracked. This includes clearing browser cache, hiding their IP address and the server's IP address, and installing privacy tools such as ad-blockers. However, nothing is 100% secure or private. Even with all these measures in place, network traffic data can still be collected by a passive adversary. Even though the network traffic is encrypted, the size of each network packet, the number of packets, and the direction of the packets (from client to server or from server to client) can still be seen. Furthermore, if the adversary is at the first hop of the victim, e.g. the Internet Service Provider (ISP), then that adversary knows the identity of the victim. The adversary still needs to determine the identity of the server/destination based only on the packet size and number of packets sent. This is an attack known as website fingerprinting.

Website fingerprinting has been known for a while now [1]. Much of the work in this area has improved on the accuracy and feasibility of the attack. Website fingerprinting defenses have also been proposed and implemented in Tor [2]. However, most of the work have made some assumptions about the user. For example, only one website is visited at a time and it can be determined when a website download starts and ends. Moreover, all the website fingerprinting data have been collected in laboratory settings.

The motivation for this work is that nobody has tried to determine whether website fingerprinting attacks are possible in a realistic setting based on real users' webbrowsing behaviors. The goal of this research is to find real users' webbrowsing behaviors in a privacy-preserving manner. A webbrowser extension called WebTracker was developed to track which websites users are visiting, when a website download starts, and when a website finishes downloading. Participants were then recruited and compensated to fill out an anonymous survey and to install WebTracker.

The results from WebTracker show that almost 16% of websites visited had an overlap with at least one other website. An overlap means that two websites visits are happening at the same time, thus that makes website fingerprinting harder as it cannot be determined whether a network packet is for website1 or for website2. This result shows that it is important to determine real webbrowsing behaviors and use that information to perform more realistic website fingerprinting attacks and to design more efficient defenses.

The contributions of this research are as follows.

- Empirically determine real users' browsing behaviors.
- Develop a privacy-preserving webbrowser extension, called WebTracker, to track user's webbrowsing activities.
- Evaluation shows that a non-trivial part of website downloads overlap with other websites, which could make website fingerprinting harder.

The paper is organized as follows. Section II provides some background information of webbrowsing behaviors and website fingerprinting, along with related work. Section III shows the design of the proposed webbrowser extension WebTracker. The data collection procedure and webbrowsing results are shown in Section IV. Discussion and limitations of our work are outlined in Section V. Section VI provides a summary and some avenues for future work.

II. Background and Related Work

To obtain user's webbrowsing behaviors, real users need to be recruited. Crowdsourcing platforms such as MTurk [3] and Microworkers [4] are used by researchers and companies to obtain real users to perform various tasks, such as completing a survey, testing some website features, or reviewing an application. Anybody can become a user of these crowdsourcing platforms. They may sign up for tasks and are compensated for their time. Users can also be recruited through social media platforms. In our case, we recruited participants from Microworkers to fill out an anonymous survey hosted by Qualtrics and then to install WebTracker on their computer. Since the participants are anonymous, we could not actually identify who installed our extension and compensated everybody who filled out the survey. Qualtrics includes a feature which does not log any IP addresses.

To record webbrowsing, a webbrowser extension was developed as it is lightweight and can access the internals of a webbrowser without being a full blown application. Any user of a computer can easily (un)install a webbrowser extension.

Website fingerprinting [1], [2], [5]–[35] is an attack that attempts to identify the website visited based only on network traffic. Other fingerprinting attacks [36] exist but do not rely on network traffic and are orthogonal to this research. The adversary in a website fingerprinting attack sits on the first hop of the victim, e.g. the ISP or a Tor [37] entry relay. Knowing the victim, the goal of the adversary is to determine the website visited given that the IP address is hidden, e.g. using Tor or a VPN. Thus, only the size of each network packet, the number of packets, the direction of each packet (from server to client or from client to server), and the timing information of each packet are observed. Using only this information and machine learning algorithms such as k-nearest neighbor (K-NN), SVM, or deep learning, it has been shown that a website can be predicted with an accuracy over 94%.

The dataset used by the majority of previous work has been in a laboratory setting, where the authors collect the network traffic from a pre-determined set of websites using their lab machines. This is not a realistic setting as regular users browse the web differently than going to website1, wait a certain amount of time, close the tab, and then go to website2, wait a certain amount of time, close the tab, and then go to website3, etc. More recent work [38] has looked at real browsing traffic on Tor – the authors showed that the accuracy decreases significantly when monitoring more than 25 websites. This work complements that paper by looking at how users browse the web to determine if website fingerprinting is still practical.

Much of the work on website fingerprinting has collected data from one website at a time, which means a user would browse the web visiting one website at a time. Some of the previous work also clears the webbrowser cache between website visits. Although some research [39], [40] has relaxed some of these assumptions by looking at multiple tabs being opened and overlapping website visits, there has been little work on how users actually browse the web. This research looks at real users' webbrowsing behaviors, how many websites are visited, and how often they visit each website. This research will impact the practicality and feasibility of website fingerprinting attacks and defenses.

There have been previous work on monitoring user's webbrowsing behaviors [41]–[46]. Although previous work focused on creating a user browsing model or predicting which websites users will visit or determining the websites visited, this work focuses on whether users browse multiple websites at the same time. While designing the experimental data collection, [47] advice was followed.

III. Design

To track real users' webbrowsing behaviors, a privacy-preserving webbrowser extension called *WebTracker* was developed. WebTracker tracks when a user visits a new webpage and when the webpage finishes downloading. An overview of WebTracker is given in Figure 1. The tool tracks 1) when a user visits a new webpage, either by typing a URL and clicking *Enter* or by clicking on a URL link; 2) when a webpage finishes downloading; and 3) when a webbrowser tab is closed. Item #3 is not important for tracking users' webbrowsing behaviors. The data collected are.

- userID: a number that is randomly generated when WebTracker is installed. The number stays the same even if WebTracker is enabled/disabled. The number will be regenerated if the user uninstalled and reinstalled WebTracker. This is to track each user.
- tabID: the tab number that an action was triggered on.
- URL: the second-level domain of the webpage.
- status: start downloading (0), finish downloading (1), or tab closed (2).
- timestamp (in UTC).

Every time an action is triggered, the information above is concatenated as a comma-separated string and sent as a POST request to our webserver. We emphasize that the tool does not store any data locally. Every action is sent as an encrypted POST packet to our webserver. An eavesdropper could potentially determine that a user has installed WebTracker but this does not reveal the website visited.

To measure the start/finish of a page download, the $chrome.tabs$ API for Chrome and the $browser.tabs$ API for Firefox is used. That API has a method *onUpdated*, which is called whenever a tab is updated in any way. The method has a type, *changeInfo*, with sub-type "status" which returns the status of the tab. For our purposes, the relevant return values were "loading" and "complete" which are codes of "1" and "2" respectively. The timestamp is also provided – this way we were able to determine the amount of time taken for the tab to go from "loading" to "complete". Whenever a tab is closed, the $chrome/browser.tabs$ API fires an event *onRemove* – the event has a type *TabID*, which is used to track which tab was closed. We would check to ensure that the tab being closed was actually a website, and not just a blank tab.

To protect the privacy of the users, the webserver is set to not record any IP address. Thus, the only information recorded is the information sent from the webbrowser. To further protect the privacy of the users, each URL is hashed. The webserver is only publicly accessible through port 443.

WebTracker was implemented as an extension for two webbrowsers: Google Chrome and Mozilla Firefox. Since they are extensions, they utilize HTML ($\sim$40 lines of code), CSS ($\sim$100 lines of code), and JavaScript ($\sim$200 lines of code). The implementation is straightforward and contains a simple ON/OFF switch for users to choose when they want to have their webbrowsing behavior tracked or not. jQuery, a JavaScript library, is used for the switch to work. To avoid

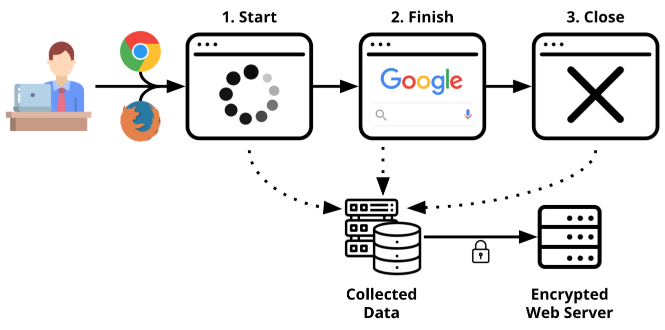


Fig. 1: Overview of WebTracker.

accidental downloads of the WebTracker, we did not post the extension in the official Chrome web store or Firefox add-on extension site. To install the webbrowser extension in either Firefox or Chrome, a user needs to manually download the extension from our webserver and manually install it.

The functionality begins with the application recognizing if the extension is on or off, as shown in Figure 1. If it is off, it does nothing. If it is on, the extension utilizes JavaScript listener events to listen for when Chrome or Firefox fires a trigger that something changed on the browser. Then the data $userID, tabID, URL, status, timestamp$ is collected and then sent encrypted to our webserver. The webserver creates a log of all that information that is sent.

IV. Evaluation

A. Data Collection

WebTracker was active, collecting data, from June 1, 2020 to December 31, 2020. Since we did not know who installed WebTracker, we could not ask the participants to uninstall the extension. Thus, on January 1, 2021, WebTracker automatically disabled itself. Users were recruited from Microworkers [4] crowdsourcing platform from June 1, 2020 to September 1, 2020. This allowed for four months of data recorded for the participants recruited on September 1 and seven months of data recorded for the participants recruited on June 1. Participants could disable or uninstall WebTracker at any time. IRB (Institutional Review Board) approval was obtained before the experiments were conducted.

Participants recruited from Microworkers had to be over 18 years old, had to understand English and can be from any country. We could not verify this information since we do not know who the workers were, but these were options selected on the Microworkers site. Every participant was asked to fill out a brief survey. The survey described the experiment and asked for their worker ID. Before filling out the worker ID, participants were asked to install WebTracker. They were provided installation and uninstallation instructions for both Mozilla Firefox and Google Chrome. Each participant was compensated with $\$1.50$ after completion of the survey. Due to the privacy features implemented in WebTracker, it cannot be determined whether a participant installed the extension. A total of 238 participants were recruited.

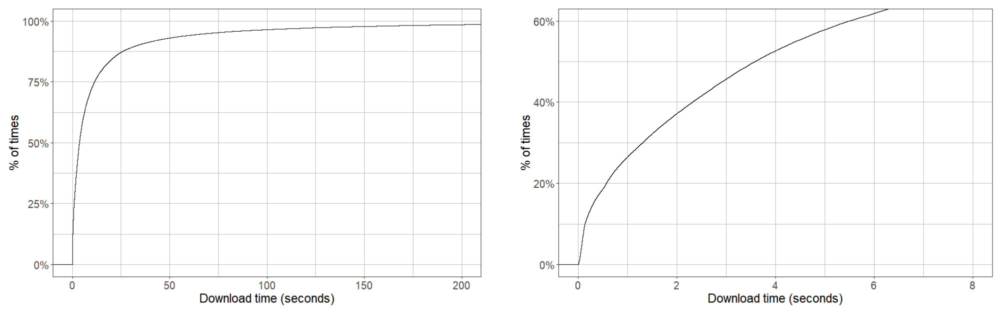
B. WebTracker Results

Browsing Results.

Recall that the goal of WebTracker is to privately collect participants' web browsing behavior in order to garner insight on real-world website visit overlaps, which can be used further in website fingerprinting research. The data collected includes a unique and random userID so that each individual user can be monitored, the tabID in case a user visits the same website using different tabs, the URL, the status so that we know when website download starts and stops, and the timestamp of that event. Users can enable/disable WebTracker at anytime. Moreover, webbrowsers could crash or be closed before all websites have finished downloading. Finally, WebTracker could have been installed while other website visits are in progress. It is highly unlikely that a user will install WebTracker on a fresh webbrowser. Thus, our data collection was conservative: 1) we excluded any URLs that did not have a start downloading or finish downloading status; 2) we excluded any website visits that took longer than 10 minutes to download. It is atypical for a webpage to take more than several seconds to finish downloading. That large download time could be because it was a visit to the same URL on the same tab but at a later time. As an example, a participant could start to visit a website on the first tab and then disable WebTracker. The website finishes downloading. Some time later, the same participants visits the same website on the first (same) tab, then enables WebTracker. The extension will log when that website finishes downloading. The time difference for that URL will be big since WebTracker only knew about the start of the first visit and the end of the second visit; 3) we excluded data that came from the *microworkers.com* website. The reason is that our participants were recruited from that website and likely visited it often, potentially skewing the data.

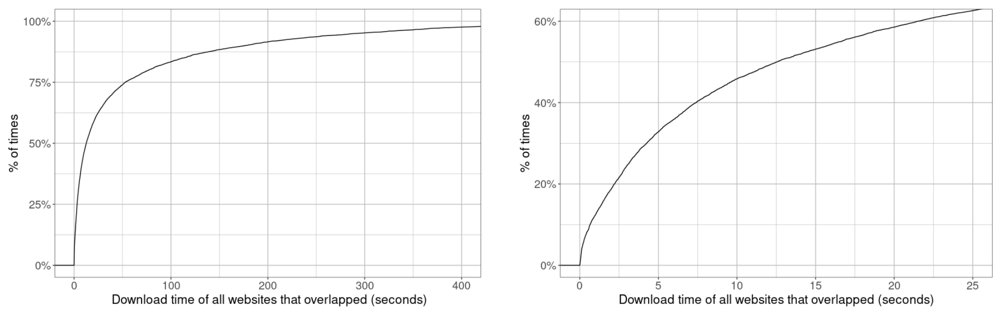
Webtracker was installed a total of 83 times. This is lower than the 283 users recruited because not all users installed the extension. Due to the way the experiment was setup, it was not possible to determine which user installed the extension. Moreover, some users uninstalled the extension soon after install because we only obtained data from some userIDs for only a few minutes. The total time elapsed between the start of our data collection and the end is about 211 days (almost 7 months from June 1 to December 31). The total number of websites counted in this entire dataset is $57,097$ websites, or roughly about 270 websites visited a day on average. Out of that time, users spent 50.88% of that time downloading data, or about 107 days. Furthermore, out of all the time spent downloading, 15.66% of downloaded websites were overlapping, with 2.11 websites in an overlap on average.

Figure 2 shows that up to 75% of websites take less than 10 seconds to download. Some websites do take longer to download. This could be possible because websites that users visit could include downloading files such as photos, videos,



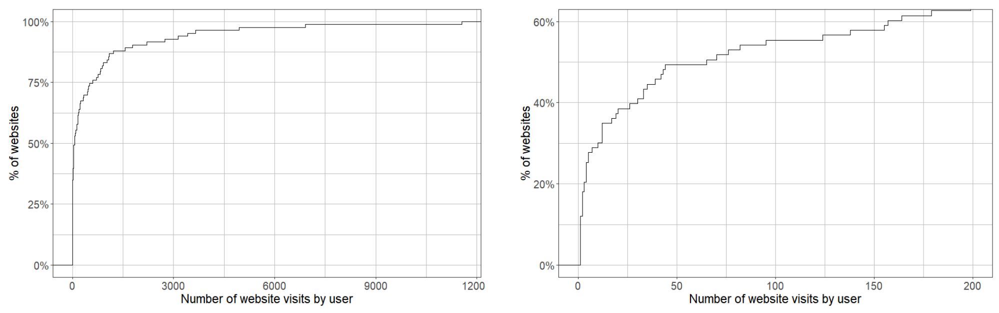
(a) (b)

Fig. 2: Empirical Cumulative Distribution Function (CDF) plot of time taken to download a website. (a) shows the CDF for download times up to 200 seconds and (b) shows a zoomed-in figure with the CDF percentage up to 60%.



(a) (b)

Fig. 3: Empirical cumulative distribution function (CDF) plot of download times of only websites that have overlapped. (a) shows the CDF for loading times up to 200 seconds and (b) shows a zoomed-in figure with the percentage up to 60%.



(a) (b)

Fig. 4: Empirical cumulative distribution function (CDF) plot of number of websites visited per user. (a) shows the CDF for up to 1,200 website visits (b) shows a zoomed-in figure with the CDF percentage up to 60%.

and music, which are often large in file size and take longer to download. Moreover users that participated in this research might have come from different countries, meaning some may not have access to high internet speeds, which may result in a longer download time. The average download time was 16.68 seconds, and the median download time was 3.57 seconds with a standard deviation of 47.95 seconds. This indicates that the data itself varies greatly, but the low median suggests most websites do not take long to download, with larger websites bringing up the average. A larger variance in website downloads potentially indicates that websites are more unique. If a website took a longer time to download than usual, its packet data could point to it belonging to a media website, such as one with videos or pictures. Figure 3 shows the download time only for websites that are overlapping with other websites. As shown in the figure, the median download time is 12.5 seconds, which is much higher than the median download time of 3.57 seconds for all websites. This makes sense as websites that take longer to download (for example, downloading a movie) are more likely to overlap with other websites as the user opens a different tab. Furthermore, Figure 4 includes the websites visits of the 83 users that participated in this research. Some

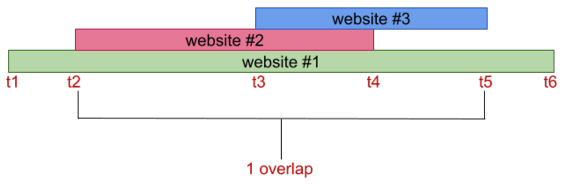


Fig. 5: Diagram demonstrating how an overlap is counted. It is not contingent on the number of websites present, but rather on when all websites that are overlapping have finished. Therefore, though there are more than two websites, the entire bundle of websites is counted as one overlap. The overlap time is thus t5 - t2. The elapsed time before overlap is t2 - t1.

users only visited a few sites while using WebTracker, but the greater proportion visited tens or hundreds of websites, with 50% of users visiting at least roughly 50 websites.

Overlapping Results.

An overlap occurs when two or more websites are being downloaded at the same time. Determining when an overlap ends is more involved. Figure 5 shows a diagram of a hypothetical webbrowsing history for a user. The user visits three websites: Website1 starts at t1 and ends at t6, Website2 starts at t2 and ends at t4, and Website3 starts at t3 and ends at t5. The overlap time in the figure is t5-t2 and encompasses all three websites. Though more than two websites are overlapping, the entire overlap time is taken as the elapsed time when all involved websites are overlapping each other. Therefore, Website2 and Website3 are not counted as their own overlap, given that they overlap already with Website1 – the three websites are counted as one overlap. However, not all websites were downloading across the entire overlap, such as Website3 was not present for time t2 to t3. We instead averaged the number of active websites in the overlap: first, there were 2 websites in the overlap between t2 and t3, then 3 websites between t3 and t4, then 2 websites between t4 and t5. Therefore, the number of websites in this overlap is $(2+3+2)/3 = 2.33$. Another important metric is the elapsed time before overlap occurs, which in the diagram is t2-t1. This shows the time when Website1 is downloading before other websites start. This is important as a longer t2 - t1 time means a possibly easier prediction of Website1. The last metric is download time. The download time for Website1 is t6 - t1. The download time for Website2 is t4 - t2. The download time for Website3 is t5 - t3.

Figure 6 shows how our algorithm works in calculating overlaps. The active list keeps track of all websites that are still downloading (status “1”). Websites are removed from the active list when they are finished downloading; those will have a status of “2”. If a website starts downloading and the active list is empty, then the start of a new overlap is found. If a new website starts downloading and the number of websites in the active list is greater than 1, then an overlap is occurring and is measured accordingly. When a website finishes downloading and the number of websites in the active list decreases back to 1, then overlapping has stopped.

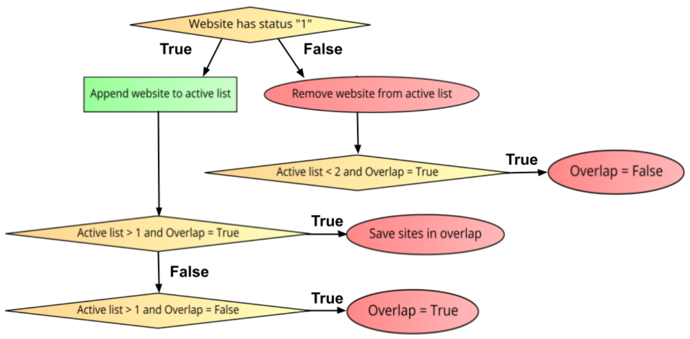

Website has status "1"
True
False
Append website to active list
Remove website from active list
Active list < 2 and Overlap = True
True
Overlap = False
Active list > 1 and Overlap = True
True
Save sites in overlap
False
Active list > 1 and Overlap = False
True
Overlap = True

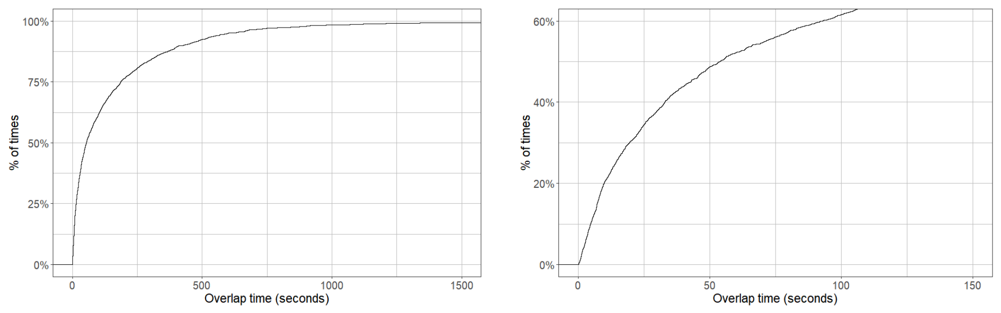
Fig. 6: Flowchart showing how our algorithm works in calculating overlaps.

Figure 7 shows the CDF for the overlap times, with the average length of an overlap being 223.19 seconds and the median time being 66.68 seconds with a standard deviation of 836.07 seconds. This indicates that the data for the overlap time varies significantly. This suggests that the overlap length is dependent on user activity, whether they are downloading a large file while browsing other websites, watching multiple videos, streaming, or performing multiple small tasks at the same time. Long overlaps could potentially produce enough noise in the data to make it difficult to fingerprint the website. If the overlap is long, it would suggest that the websites are overlapping during the duration of their entire downloading period, rather than a small overlap where websites overlap momentarily. Figure 9 show the elapsed download time of the first website before overlap occurs. 25% of the time, the overlap occurs after 4 seconds of download. The average amount of time a website downloads before overlapping occurs is 24.06 seconds with a median of 8.738 seconds and standard deviation of 45.62 seconds. Additionally, Figure 8 demonstrates the total elapsed download time of the first website in an overlap. The average amount of time it takes for the first website to download is 97.92 seconds, with median 41.46 seconds with a standard deviation of 127.24. The long elapsed times from Figure 9 indicate it generally takes some time before the first website overlaps with another website. That can pose an issue as most website fingerprinting algorithms look at the whole website network trace rather than just the first few packets or seconds. Therefore, an earlier overlap means it might be more difficult to perform a website fingerprinting attack. As can be seen from Figure 8 and Figure 9, it takes almost 9 seconds before overlap occurs. Moreover, the first website, when overlap occurs, takes 41 seconds to download. This means that 22% of the first website is not overlapping with any other website.

Summary

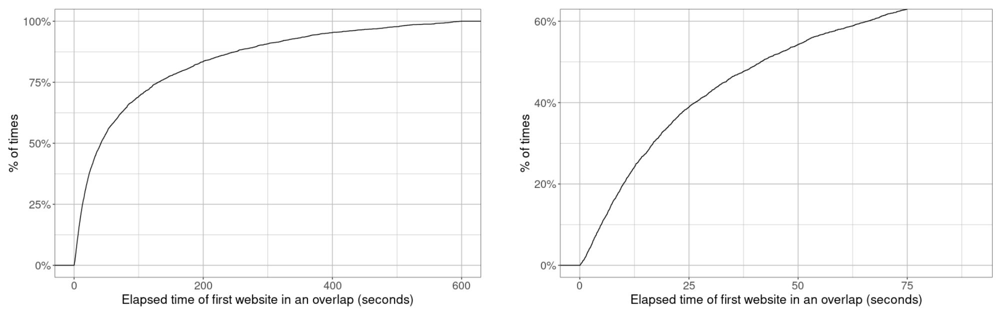
The results can be summarized as follows:

- 15.66% of websites were overlapping with at least one other website.



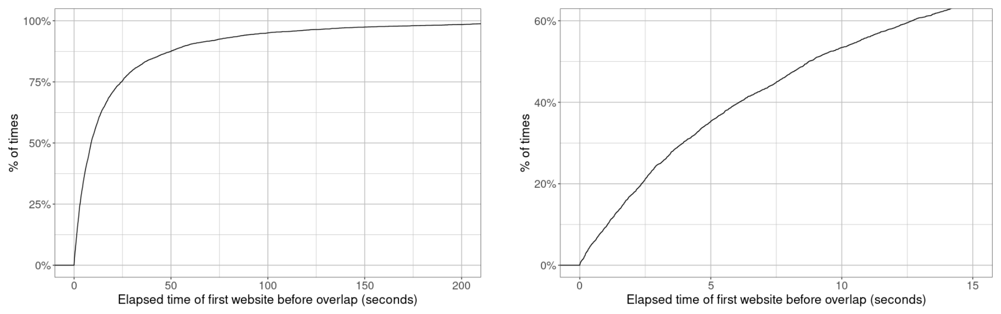
(a) (b)

Fig. 7: Empirical cumulative distribution function (CDF) plot for overlap times. (a) shows the CDF for overlap times up to 1,500 seconds and (b) shows a zoomed-in figure with the CDF percentage up to 60%.



(a) (b)

Fig. 8: Empirical cumulative distribution function (CDF) plot of time it takes for the first website in an overlap to download. (a) shows the CDF for loading times up to 200 seconds and (b) shows a zoomed-in figure with the percentage up to 60%.



(a) (b)

Fig. 9: Empirical cumulative distribution function (CDF) plot of time it takes for the first website in an overlap to download before the overlap occurs. (a) shows the CDF for loading times up to 600 seconds and (b) shows a zoomed-in figure with the percentage up to 60%.

- The median download time for non-overlapping website was 3.57 seconds.
- The median download time for overlapping websites was 12.5 seconds.
- The median time of an overlap was 66.68 seconds.
- The median time before an overlap starts for a website was 8.738 seconds.

The results indicate that overlapping could have an effect on website fingerprinting attacks, especially if the models considers the whole website instead of the first few network packets. This would make webbrowsing safer and more private. The second website will not be distinguishable in an overlapping download. However, the first 9 seconds of the first website are still vulnerable to website fingerprinting attacks – future work could address whether that is enough time to perform an accurate website fingerprinting attack.

V. Discussion

The results show that users' browsing habits are different than assumed. We found that occasionally even four websites can download at the same time. This adds a layer of complexity to determining what website(s) a network trace belongs to. Additionally, overlaps, though varying greatly in length, tended to be long, with a median of 66 seconds. Long overlaps could be a potential countermeasure against website fingerprinting attacks as overlapping websites are more difficult to identify. However, users only spent 15.66% of their downloading time in overlap, meaning that a majority of users' website visits could potentially be identified. Additionally, we found that websites download a few seconds of data before overlap occurs, with a median of 8.738 seconds elapsed time. Previous research have devised an algorithm that can accurately predict a website with only the first few seconds of its packet data with about 85% accuracy [48]. This is because websites are easier to identify from the beginning of the trace rather than the end of the trace. Therefore, though overlaps tend to be long, some websites may still be able to be identified if they are the first website in an overlap but the second website will be harder to identify. Based on this result, website fingerprinting models that utilize the whole network traffic are likely not reliable.

All the webbrowsing data collected are through a Firefox or Chrome extension. We did not measure whether the traffic is through a VPN or through the Tor network or whether the participants were using the Tor browser. Most web users do not use Tor, thus we doubt that any of the data collected is through the Tor network. This could be future research to measure only webbrowsing behavior on the Tor browser.

We claim our WebTracker to be "privacy-preserving"; however, it is still a tracker. We attempted to remove any identifiable information such as IP address and URL. Due to privacy issues, we did not collect the network traffic. This could have provided more insight on who is browsing how. Due to the lack of URLs and the lack of actual packet metadata, we could not do a website fingerprinting attack on the data we collected. It would have been interesting to determine the website fingerprinting accuracy on the overlapped websites.

We do not expect that the data collection sent to the server interferes with the data collection. The data sent is small (around 5KB including the TLS handshake) and is not recorded by the WebTracker extension.

There are some limitations. The first one is that participants were aware that their browsing activities were being monitored, thus the webbrowsing behavior might not be natural. Participants could also turn off/on the webbrowser extension at any time. Another limitation is that WebTracker did not have a mobile version and only recorded webbrowsing habits of desktop users of Firefox and Chrome. Therefore, users' mobile browsing habits are not known, nor is it known if browsing through apps rather than only through a dedicated browser affects overlaps. Moreover, there were only 83 installs of WebTracker. Due to the longer than normal download times of a few seconds, it could be that many of the participants had slower Internet connection or were not from the US or using a VPN. More users from more diverse locations need to be recruited. Future work might also explore requiring participants to upload a screenshot showing the extension has been installed before compensation.

VI. Conclusion and Future Work

This research showed two main results: 1) privately track real users' webbrowsing behaviors such as how many websites they visit, how long each website takes to download, and how many websites are visited at the same time; 2) show that website fingerprinting attacks make a strong assumption that users browse one website at a time and the adversary have access to all the network packets for each website visit. Actually, over 15% of all website visits overlapped with at least one other website. Moreover, each overlap lasted about 66 seconds. Only the first 9 seconds of a website, on average, are clean and are not overlapping with other website visits.

Our results show that more work is needed to determine the practicality of website fingerprinting attacks. Moreover, with the way the participants in this research browsed the web, there were overlaps which in some way act as a countermeasure against website fingerprinting. Overlaps tend to decrease the accuracy of website fingerprinting attacks. Unfortunately, we could not record the number of packets and thus, could not determine the number of packets during the first non-overlapping 9 seconds of website download.

As future work, more participants and from more diverse locations need to be recruited and their webbrowsing behaviors analyzed. This work could also be extended to mobile devices such as smartphones and tablets. Recording webbrowsing behaviors on the Tor browser could also be future work. No website fingerprinting attack is performed – the next step would be to collect data by simulating the webbrowsing behavior observed and determine the effectiveness of website fingerprinting attacks.

References

[1] A. Hintz, "Fingerprinting websites using traffic analysis," in *Proceedings of 2Nd International Conference on Privacy Enhancing Technologies*, ser. PET'02. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 171–178.

[2] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, "Toward an efficient website fingerprinting defense," in *ESORICS*, 2016.

[3] Amazon Mechnical Turk, https://www.mturk.com/, Accessed 2021.

[4] Microworkers, https://www.microworkers.com/, Accessed 2021.

[5] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, "Statistical identification of encrypted web browsing traffic," in *Proceedings of 2002 IEEE Symposium on Security and Privacy*, ser. SP '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 19–.

[6] G. D. Bissias, M. Liberatore, D. Jensen, and B. N. Levine, "Privacy vulnerabilities in encrypted http streams," in *Proceedings of the 5th International Conference on Privacy Enhancing Technologies*, ser. PET'05. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 1–11.

[7] M. Liberatore and B. N. Levine, "Inferring the source of encrypted http connections," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 255–263.

[8] L. Lu, E.-C. Chang, and M. C. Chan, *Website Fingerprinting and Identification Using Ordered Feature Sequences*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 199–214.

[9] D. Herrmann, R. Wendolsky, and H. Federrath, "Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier," in *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, ser. CCSW '09. New York, NY, USA: ACM, 2009, pp. 31–42.

[10] M. Perry, "Experimental defense for website traffic fingerprinting," https://blog.torproject.org/blog/experimental-defense-website-traffic-fingerprinting, 2011.

[11] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, “Website fingerprinting in onion routing based anonymization networks,” in *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, ser. WPES ’11. New York, NY, USA: ACM, 2011, pp. 103–114.

[12] X. Gong, N. Borisov, N. Kiyavash, and N. Schear, “Website detection using remote traffic analysis,” in *Proceedings of the 12th International Conference on Privacy Enhancing Technologies*, ser. PETS’12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 58–78.

[13] T. Wang and I. Goldberg, “Improved website fingerprinting on tor,” in *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, ser. WPES ’13. New York, NY, USA: ACM, 2013, pp. 201–212.

[14] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, “Touching from a distance: Website fingerprinting attacks and defenses,” in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ser. CCS ’12. New York, NY, USA: ACM, 2012, pp. 605–616.

[15] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, “Effective attacks and provable defenses for website fingerprinting,” in *Proceedings of the 23rd USENIX Conference on Security Symposium*, ser. SEC’14. Berkeley, CA, USA: USENIX Association, 2014, pp. 143–157.

[16] R. Nithyanand, X. Cai, and R. Johnson, “Glove: A bespoke website fingerprinting defense,” in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, ser. WPES ’14. New York, NY, USA: ACM, 2014, pp. 131–134.

[17] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt, “A critical evaluation of website fingerprinting attacks,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’14. New York, NY, USA: ACM, 2014, pp. 263–274.

[18] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, “A systematic approach to developing and evaluating website fingerprinting defenses,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’14. New York, NY, USA: ACM, 2014, pp. 227–238.

[19] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, “Website fingerprinting at internet scale,” in *Proceedings of the 23rd Internet Society (ISOC) Network and Distributed System Security Symposium (NDSS 2016)*, 2016.

[20] X. Cai, R. Nithyanand, and R. Johnson, “Cs-buflo: A congestion sensitive website fingerprinting defense,” in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, ser. WPES ’14. New York, NY, USA: ACM, 2014, pp. 121–130.

[21] T. Wang and I. Goldberg, “On realistically attacking tor with website fingerprinting,” in *Privacy Enhancing Technologies Symposium (PETS)*, 2016.

[22] R. Spreitzer, S. Griesmayr, T. Korak, and S. Mangard, “Exploiting data-usage statistics for website fingerprinting attacks on android,” in *Proceedings of the 9th ACM Conference on Security &#38; Privacy in Wireless and Mobile Networks*, ser. WiSec ’16. New York, NY, USA: ACM, 2016, pp. 49–60.

[23] J. Hayes and G. Danezis, “k-fingerprinting: A robust scalable website fingerprinting technique,” in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016, pp. 1187–1203.

[24] G. Cherubin, J. Hayes, and M. Juárez, “Website fingerprinting defenses at the application layer,” *PoPETs*, vol. 2017, no. 2, pp. 186–203, 2017. [Online]. Available: https://doi.org/10.1515/popets-2017-0023

[25] S. E. Oh, S. Li, and N. Hopper, “Fingerprinting keywords in search queries over tor,” *PoPETs*, vol. 2017, 2017.

[26] G. Cherubin, “Bayes, not naive: Security bounds on website fingerprinting defenses,” *Proceedings on Privacy Enhancing Technologies*, no. 4, pp. 135–151, 2017.

[27] K. Kohls, D. Rupprecht, T. Holz, and C. Pöpper, “Lost traffic encryption: Fingerprinting lte/4g traffic on layer two,” in *Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WiSec ’19. New York, NY, USA: ACM, 2019, pp. 249–260.

[28] V. Rimmer, D. Preuveneers, M. Juárez, T. van Goethem, and W. Joosen, “Automated feature extraction for website fingerprinting through deep learning,” *25th Symposium on Network and Distributed System Security (NDSS 2018)*, 2018.

[29] P. Sirinam, M. Imani, M. Juarez, and M. Wright, “Deep fingerprinting: Undermining website fingerprinting defenses with deep learning,” in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’18. New York, NY, USA: ACM, 2018, pp. 1928–1943.

[30] M. S. Rahman, P. Sirinam, N. Mathews, K. G. Gangadhara, and M. Wright, “Tik-tok: The utility of packet timing in website fingerprinting attacks,” *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 3, pp. 5–24, 2020.

[31] T. Pulls and R. Dahlberg, “Website fingerprinting with website oracles,” *Proceedings on Privacy Enhancing Technologies*, vol. 2020, no. 1, pp. 235–255, 2020.

[32] W. De la Cadena, A. Mitseva, J. Hiller, J. Pennekamp, S. Reuter, J. Filter, T. Engel, K. Wehrle, and A. Panchenko, “Trafficsliver: Fighting website fingerprinting attacks with traffic splitting,” in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 2020, pp. 1971–1985.

[33] T. Wang, “High precision open-world website fingerprinting,” in *2020 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, may 2020, pp. 231–246.

[34] P. Sirinam, N. Mathews, M. S. Rahman, and M. Wright, “Triplet fingerprinting: More practical and portable website fingerprinting with n-shot learning,” in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1131–1148.

[35] C. Wang, J. Dani, X. Li, X. Jia, and B. Wang, “Adaptive fingerprinting: Website fingerprinting over few encrypted traffic,” in *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, 2021, pp. 149–160.

[36] S. Karami, P. Ilia, and J. Polakis, “Awakening the web’s sleeper agents: Misusing service workers for privacy leakage,” in *Network and Distributed System Security Symposium*, 2021.

[37] Tor, https://www.torproject.org/, 2021.

[38] G. Cherubin, R. Jansen, and C. Troncoso, “Online website fingerprinting: Evaluating website fingerprinting attacks on tor in the real world,” in *31st USENIX Security Symposium (USENIX Security 22)*. Boston, MA: USENIX Association, Aug. 2022, pp. 753–770.

[39] Y. Xu, T. Wang, Q. Li, Q. Gong, Y. Chen, and Y. Jiang, “A multi-tab website fingerprinting attack,” in *Proceedings of the 34th Annual Computer Security Applications Conference*, ser. ACSAC ’18. New York, NY, USA: ACM, 2018, pp. 327–341.

[40] W. Cui, T. Chen, C. Fields, J. Chen, A. Sierra, and E. Chan-Tin, “Revisiting assumptions for website fingerprinting attacks,” in *ACM Asia Conference on Computer and Communications Security*, ser. AsiaCCS ’19. ACM, 2019.

[41] M. A. Awad and I. Khalil, “Prediction of user’s web-browsing behavior: Application of markov model,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 4, pp. 1131–1142, 2012.

[42] T.-P. Liang and H.-J. Lai, “Discovering user interests from web browsing behavior: An application to internet news services,” in *Proceedings of the 35th annual Hawaii international conference on system sciences*. IEEE, 2002, pp. 2718–2727.

[43] J. J. Lee and M. Gupta, “A new traffic model for current user web browsing behavior,” *Intel corporation*, 2007.

[44] S. Goel, J. Hofman, and M. Sirer, “Who does what on the web: A large-scale study of browsing behavior,” in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 6, no. 1, 2012.

[45] H. Weinreich, H. Obendorf, E. Herder, and M. Mayer, “Not quite the average: An empirical study of web use,” *ACM Transactions on the Web (TWEB)*, vol. 2, no. 1, pp. 1–31, 2008.

[46] T. Takahashi, C. Kruegel, G. Vigna, K. Yoshioka, and D. Inoue, “Tracing and analyzing web access paths based on {User-Side} data collection: How do users reach malicious {URLs}?” in *23rd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2020)*, 2020, pp. 93–106.

[47] J. F. Duncan and L. J. Camp, “Conducting an ethical study of web traffic,” in *Proceedings of the 5th USENIX Conference on Cyber Security Experimentation and Test*, ser. CSET’12. USA: USENIX Association, 2012, p. 7.

[48] W. Cui, T. Chen, and E. Chan-Tin, “More realistic website fingerprinting using deep learning,” in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2020, pp. 333–343.