Link Prefetching: A Defense Against Website Fingerprinting on Tor *

Vaibhav Sharma
sharm361@umn.edu

Taejoon Byun
taejoon@umn.edu

Elaheh Ghassabani
ghass013@umn.edu

Se Eun Oh
seoh@umn.edu

Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55454

ABSTRACT

This paper is written to get an A in the CSCI5271 course. PERIOD.

This content will be edited later. We plan to explore the area of website fingerprinting in anonymization networks starting with the paper Website Fingerprinting in Onion Routing Based Anonymization Networks.

Keywords

Website fingerprinting, anonymity, encrypted traffic, Tor

1. INTRODUCTION

We did something awesome! *The content of this section will be changed later.* Penchenko et al. [5] were among the first to report website fingerprinting attacks with reasonable accuracy on Tor. This paper provides a sufficient understanding of the feature set and classification framework required for this attack. Some of the team members are familiar with data mining techniques and software packages required for their use. All team members have access to the compute servers provided by the Computer Science and Engineering department and will request access to the other high performance computing resources if required.

2. WEBSITE FINGERPRINTING ON TOR

This section explains the background about Tor, an anonymity network, and the website fingerprinting attack which can neutralize the anonymity that Tor provides.

2.1 Tor

Tor [1] is something.

*This report is submitted as a partial fulfillment of *CSCI5271: Introduction to Security* course.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

2.2 Website Fingerprinting

Figure 1 illustrates the attack model of website fingerprinting.

Figure 1: Website Fingerprinting Attack Model

3. RELATED WORK

There are a decent number of works since 0000 which attempts to deanonymize the Tor network.

3.1 Fingerprinting Attacks

3.2 Defense Mechanisms

3.3 Criticism

4. LINK PREFETCHING AS A DEFENSE

A classifier used in a website fingerprinting attack can distinguish the differnce between two classes of packets when the difference is consisntent between them [**?**]. Thus, intuitively, a classifier would work the best if the difference in the same class is minimal and the difference among heterogeneous classes is high at the same time. However, when variance among fingerprints in the same class is high, a classifier would not be able to characterize it clearly or at least the detection rate using the classifier would be low. This is the basic idea of using prefetching as a defense mechanism; our conjecture is that we can use prefetching to manipulate the number and size of incoming and outgoing packets in order to increase the variance among packets in the same class.

This section explains the concept of link prefetching, discusses its effect on website fingerprints and argue its possible usage as a defense mechanism.

4.1 Link Prefetching

Link prefetching is a HTML syntax that gives the web browser hints about which page the user is most likely to visit in a near future [2, 3]. The pages and resource to pre-fetch are specified in the web page so that the web browser can silently load them (or pre-fetch them) after an idle time. Since the pre-fetch happens only after the page is fully loaded, it does not sacrifice the loading time of the requested web page. Moreover, it can save the loading time for the pre-fetched pages and thus improve the user experience by caching the *future* contents. It was first suggested by Mozilla Foundation in 2003 and supported by most modern browsers nowadays.

The resources to prefetch can be simply specified in *HTML* using a `link` tag [4]. For example, a `link` tag `<link rel="prefetch" href="/page2.html">` tells the browser to prefetch a *html* file named *page2.html*. Resources other than a *HTML* web page can also be pre-fetched similarly using the same syntax. There are also some variations for different types of prefetching – DNS prefetching, which is specified as `<link rel="dns-prefetch" ..>`, is supported by *Mozilla Firefox* and *Google Chrome*, and `<link rel="prerender" ..>` also does the same job as `prefetch` in *Google Chrome* and *Microsoft Internet Explorer*.

Figure 2 illustrates how prefetching actually works in a browser (*Google Chrome*). This page is set up arbitrarily by the authors to demonstrate link prefetching, and contains a link pre-fetch tag that specifies a big image (the image on the left side labeled as *prefetch*). When the prefetching is off, this image shall be requested only when a user puts his mouse cursor on it (mouse over). However, it can be seen on the network timeline (right bottom) that the image is prefetched right after loading the page. This is indicated by a long blue bar on the second row for the file named "*Very-high...*", and it is long because the size of the file is relatively big (*3.5 MB*) that it took a longer time to download. Please also note that the time it took for loading the image when the user actually requested (by putting his mouse on it) was very short, because the image had already been pre-fetched that the browser merely loaded it from the cache (as shown in the *size* column on the fourth row).

4.2 The Effect of Prefetching on Website Fingerprint

Link prefetching obviously affects the traffic by sending additional request for prefetching items, and by prefetching those items after an idle time. However, to what degree it affects the website fingerprint has not been studiedto the best of our knowledge. This subsection first illustrates how the prefetching affects the number of packets being transferred as an example, and then discusses what features of fingerprint prefetching can possibly affect.

4.2.1 An Example

Figure 3 illustrates a website fingerprint in terms of inter-packet timing, where the *x-axis* represents time in seconds and the *y-axis* represents the number of packets captured during a certain time interval (*500 ms*). The solid line depicts the packets captured while prefetching was enabled, and the dashed line depicts when the pre-fetch was disabled in the browser settings. Both the cases are captured for the same web page, which is the first page of `wired.com`. However, the number of packets are slightly different for each case (4055 for the *off* case while 4218 for the *on* case), because the prefetch-on case obviously requests more resources for caching purpose. The elapsed time for loading the whole page was also different, but this variance is mainly due to the difference in network condition because all the packets are more scattered throughout time compared to the prefetch-off case. When we ignore the speed difference, we can see that the four peaks of the two lines are roughly the same that if we capture the packets multiple times for the same website, we can characterize how the *fingerprint* of a specific website looks like. Please note also that inter-packet timing is only one of many features for characterizing website fingerprint.

Table 1: Experiment design to answer *RQ1*

victim \attacker	prefetch on	prefetch off
prefetch on	(1)	(2)
prefetch off	(3)	(4)

4.2.2 The Effect of Prefetching on Features

Packet lengths Packet length frequency Packet ordering. Interpacket timing

5. EXPERIMENTAL EVALUATION

Experimental result comes here.

5.1 Research Question

1. **RQ1**: Does prefetching itself provide an extra degree of defense?
2. **RQ2**: Can prefetching be used as a browser-side defense mechanism?

1. We speculate that prefetching itself might provide extra defense because of the extra packets. It can also be the case however, prefetching websites are more vulnerable to fingerprinting because of the extra prefetch packets that shows distinct prefix.
2. This case is unlikely since prefetching is on by default. We assume that victims will more likely be using Tor under the default setting.
3. This case is what we are most curious about, whether a victim can confuse an attacker by simply turning prefetching setting off of his browser.
4. This case simulates a situation where a victim is loading any other websites that does not prefetch any resource. This can be used as a comparison case.

5.2 Effects of Pre-Fetching on Fingerprinting

In this section, we will write about our experiments. We are planning to conduct two sets of experiments. If we consider the network traffic, the number of packages go upstream depends on the number of pre-fetching requests, and the number of downstream packages coming depends on the size of resources that should be pre-fetched. Therefore, it is obvious that pre-fetching would affect the fingerprint of the traffic of a particular website. *should be completed.*

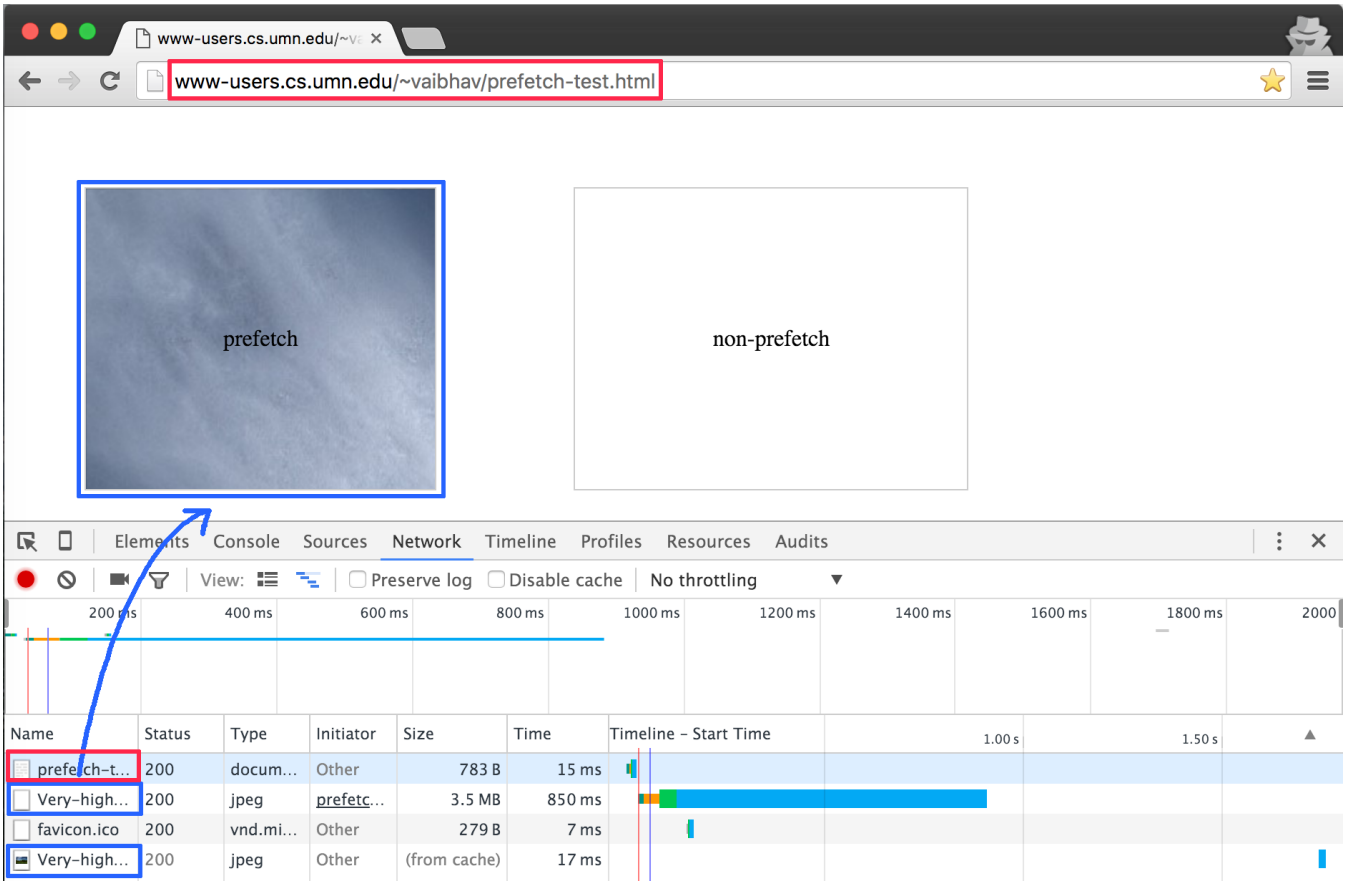

Figure 2: Network timeline showing pre-fetch

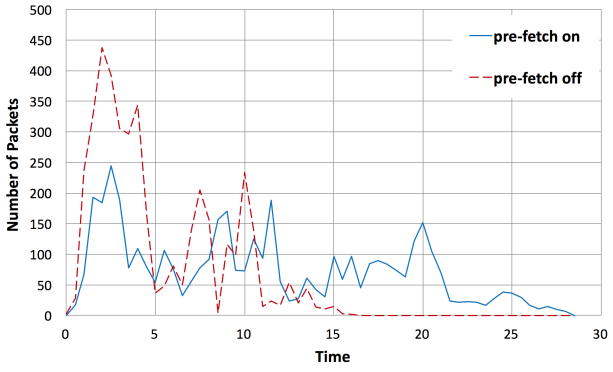

Figure 3: A website fingerprint for pre-fetch on/off cases

5.3 Investigate Pre-Fetching Effects on top 60 Popular Websites

We are running experiments to see how pre-fetching affects the websites' fingerprints. After doing some search on top popular websites, we put together a small crawler by which we learnt that only around 60 of all 6000 websites are use pre-fetching mechanism. We are capturing traffic of these websites in two different modes: 1) with enabled pre-fetching, and 2) with disabled pre-fetching. We are working on feature extraction, and about to decide which classifiers to use for the learning phase. Ultimately, we plan to conduct two sets of experiments. One sort of experiment is to compare two series of the captured packets and find the accuracy number with the help of a classifier, by which our goal is to provide an evidence to see if pre-fetching really affects fingerprints of websites. So, if the result will be positive, we will perform another set of experiment, which kind of simulates a sub set of those 60 websites. Then, we will see how (altering) the size of pre-fetching affects fingerprinting attacks/ defense mechanisms.

5.4 Effect of Pre-Fetching Packets Size on Fingerprinting Attacks

Here, we will explain our second experiment. We will simulate a sub set of webpages we investigated in the previous experiment. Then, we will equip them with a mechanism so that they can affect the downstream traffic and finally their fingerprint. Then, we will analyze the result to see how this idea contributes to the effectiveness of attacks and defense techniques.

6. DISCUSSION

We'll find what to discuss later on after finishing the experiment section.

7. CONCLUSIONS

What we have done in this work is so cool and awesome. What you may criticize will all be put here as "future work".

This section will conclude the result of our experiments. Finally we will provide some evidence to show how prefetching affect fingerprinting attacks. Based on our result, we are planing to suggest some defense mechanisms.

8. ACKNOWLEDGMENTS

The authors appreciate Professor Stephen McCamant for telling geeky jokes in classes all the time. This is a research project for CSCI5271, University of Minnesota.

9. REFERENCES

[1] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[2] D. Fisher. Link prefetching FAQ, 2003.

[3] D. Fisher and G. Saksena. Link prefetching in mozilla: A server-driven approach. In *Web content caching and distribution*, pages 283–291. Springer, 2004.

[4] M. Nottingham. Rfc5988: Web linking. Technical report, 2010.

[5] 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*, pages 103–114. ACM, 2011.