Tail Time Defense Against Website Fingerprinting Attacks

JINGYUAN LIANG1, CHANSU YU1, (Senior Member, IEEE), KYOUNGWON SUH2, (Member, IEEE), AND HYOIL HAN2, (Member, IEEE)

1Department of Electrical Engineering and Computer Science, Cleveland State University, Cleveland, OH 44115, USA
2School of Information Technology, Illinois State University, Normal, IL 61790, USA

Corresponding author: Jingyuan Liang (j.liang18@vikes.csuohio.edu)

This work was supported in part by the U.S. National Science Foundation under Grant 2028397.

ABSTRACT In the past few years, many defense mechanisms have been proposed against website fingerprinting attacks. Walkie-Talkie (WT) built on top of the Tor network is known to be one of the most effective defense mechanisms. However, we observed that WT significantly increases the page loading time (time overhead) although the bandwidth overhead is not high. This paper analyzes the cause of the increased page loading time and presents a defending approach called Tail Time (TT), which addresses the problem by limiting the maximum time for which a pending request can block subsequent requests. Our experimental results indicate that the proposed TT defense can significantly reduce the page loading time while keeping the defense performance on par.

INDEX TERMS Website fingerprinting, censorship, tor, machine learning, user perceived experience.

I. INTRODUCTION

Internet censorship is on the rise [1]–[6]. Earlier, it was the surveillance of information in plain text [7], [8]. Then, censors simply blocked encrypted messages when encryption was not widely deployed [9]. Now, censors face the choice of what to do with the data that they cannot fully understand and they usually cannot afford a complete block [10]–[12].

One of the popular approaches adopted by censors is known as website fingerprinting [13]–[15]. It uses features of packets in network communication to infer the contents in the communication, more precisely, to match it to one of the websites targeted for surveillance. The most popular features that have been exploited are packet lengths, packet length frequency, packet ordering, and inter-packet times [16]. Machine learning approaches, such as k-Nearest Neighbors [17] can be used to conduct the fingerprinting attack. Once identified, the censor may decide to block the network connection completely. However, considering the fact that the identification of such connections is not guaranteed to be accurate, it is also possible that the censor may start deteriorating the quality of service by, for example, randomly dropping packets, or throttling the traffic [9], [18].

In this paper, we propose a delay-conscious defense against website fingerprinting attacks, called Tail Time (TT), which is implemented on top of Walkie-Talkie (WT) [19]. WT is known to add little bandwidth overhead compared to other approaches while achieving excellent defense performance. However, WT increases web page loading time substantially, which is already high since it uses the Tor network [20]. Moreover, it is observed that WT results in a large amount of access failures, contributed by the WT’s half-duplex communication and asynchronous nature of web scripts. TT defense addresses both by limiting the maximum time for which a pending request can block subsequent requests.

Contributions of this paper are three-fold: First, most of previous work on Internet censorship focuses on the defense efficiency as a primary performance measure. This paper takes the user experience, more specifically, the page loading time, into consideration when evaluating the defense mechanisms. Our experimental study shows that WT renders the page loading intolerably slow (longer than 120 seconds) in 53% of cases, which is less than 1% in TT. Second, this paper presents an advanced experiment setup, which includes a more accurate cell sequence collection and a future-proof WT/TT implementation. We modified the Tor client such that it emits cell information directly, which is in contrast to the conventional indirect method based on TCP packet capture [19]. Moreover, it uses the webRequest WebExtension
API in Mozilla Firefox instead of patching the codebase of Firefox itself so that it allows other researchers to access and modify easily; Selenium [21] via geckodriver [22] was used to configure and control the Firefox process. Third, over the years, the HTTP request-response model and their underlying transportation mechanisms have gone through major changes, most significantly from HTTP/1.x to HTTP/2. This paper evaluates the impact of HTTP/2 on the website fingerprinting attacks and defenses.

Our experimental study shows that TT reduces the page loading time by 28.0% or more in comparison to WT in majority of websites we tested. TT maintains its advantage under HTTP/2 even though HTTP/2 alleviates the head-of-the-line issue and thus benefits WT more. Note that the defense efficiency of TT is on par with WT.

This paper is organized as follows: Section II describes the background and related work on fingerprinting attacks. Sections III and IV describe the TT defense methodology and implementation and experiment setup respectively. Section V presents our experiment results. Section VI provides conclusions and future work.

II. BACKGROUND
A. TOR DEFENSE
To combat against website fingerprinting attacks, many efforts have been made on improving the Tor network itself or developing ones on top of Tor [23]. This is because Tor itself is already proven to be resilient to website fingerprinting attacks to some degree due to the fact that the Tor network traffic is packed into fixed-length cells and sent over circuits that cannot be predicted from outside. Additionally, various control cells are added to the traffic which makes the traffic vary further [24], [25]. Major media websites, such as BBC News, adopted Tor, to ensure that their contents are available to audiences even in the presence of censorship [6].

B. ATTACKING TOR TRAFFIC
With Tor, the adversary can only see (or infer) a sequence of fixed-length cells and their directions when users are browsing a website. Considering that the cells are encrypted, what the adversary can see are a number of alternating (as in direction) batches, and the number of cells in each batch, as fingerprints. The adversary calculates the fingerprints of such cell sequences for websites to be monitored, and stores the fingerprints in advance. When users are browsing the web, the adversary tries to match the cell sequence observed from users against the data set of known fingerprints, attempting to identify the website users are browsing [24], [25].

C. DEFENSES AGAINST FINGERPRINTING
Effective defenses against website fingerprint attacks are to reschedule the packets (or Tor cells), or to insert padding units to confuse the attacker. Examples are Shmatikov’s adaptive padding [26] and Wright’s traffic morphing [27]. However, rescheduling adds a delay (“time overhead”) because a unit can no longer be sent immediately once generated and needs to wait for scheduling. And padding wastes network bandwidth (“bandwidth overhead”) because of the padded units that do not carry useful information [28]. Recent defenses against website fingerprinting attacks (BuFLO [29], Tamaraw [28], and Walkie-Talkie [19]) use both rescheduling and padding. For example, BuFLO renders accesses to different websites produce exactly the same packet sequence, defeating any possible classifier, but it increases both bandwidth and time overhead.

D. WALKIE-TALKIE DEFENSE
WT [19] is known to be one of the most effective defense algorithms. Two major mechanisms of WT are half-duplex communication and padding. First, it holds all pending data requests until it receives responses of all previous requests. This modification is designed to combine as many requests or responses as possible into a single burst, reducing the total number of bursts. Within each burst, there are only transmissions in the same direction (i.e., either all requests or all responses).

Second, padding is used to make traffic through WT not fingerprintable. It inserts padding units to each burst, attempting to make cell sequences more similar to each other. Since the number of bursts is already reduced and padding is needed only for each burst, the bandwidth overhead is not high compared to Wright’s traffic morphing [27] or BuFLO [29] mentioned above.

E. HTTP/2 AND HTTP/1.X
Website fingerprinting attacks and defenses are affected by underlying transporting mechanisms. HTTP/1.x is known to have several shortcomings such as the limit on the number of connections and the head-of-line blocking. HTTP/2 addresses them and features several new functionalities including multiplexing and server push. The adoption rate of HTTP/2 among websites has increased steadily in recent years, reaching more than 70% in 2021 [30].

Lin et al. compared the defense performance with HTTP/2 and HTTP/1.1 [31]. Their study shows that a plain switch from HTTP/1.1 to HTTP/2 provides no extra defense and no additional vulnerabilities. However, when the server push is used, the k-NN attack accuracy can be reduced from 80% to 74% with no defense applied. If padding is used, which can be achieved naturally in HTTP/2 by pushing unneeded assets using server push, the accuracy could be further reduced to as low as 64%. However, the server push feature is not commonly enabled across the Internet at the moment, and it needs some specific web server configuration.

Note that their research was based on the “modeled” environment [31]. It uses mock websites that replicate the same asset sizes and dependency tree as the actual websites, resulting in similar loading behaviors of the actual websites. This allows them to try different configurations of protocols such as selection of HTTP protocol and server push feature. We adopted the modeled environment in our study.
III. DELAY-CONSCIOUS DEFENSE

Bandwidth overhead is less of a concern in WT because there exists a vast amount of bandwidth available. Rather, time overhead in WT is a concern because it directly affects the quality of service and is easily misinterpreted as service unavailability.

The proposed mechanism, Tail Timeout (TT), is an improvement over WT. It excludes slow resources from WT scheduling so that they do not block other requests for an extended period of time, minimizing the delay caused by such scheduling. TT removes them from the pool of outstanding requests when they have stayed there for a certain tail timeout ($t$). The number of such slow resources is usually less than 10% according to our experiment with Alexa’s top 100 websites[32]. Considering that only a small number of requests trigger such scenarios, we expect that this change will not significantly increase the number of bursts, thus not deteriorating the defense performance as demonstrated in our evaluation in section V-B.

A. HALF-DUPLEX IMPLEMENTATION

As discussed earlier, to ensure that our solution is future-proof and available for other researchers, we use the webRequest WebExtensions API[33]. First, the half-duplex implementation in the originally published WT was made by modifying the Mozilla Firefox source code[19]. The patch no longer applies in current versions of Firefox because the relevant part in Firefox code has been rewritten. Second, this allows extensions to attach event listeners and they receive detailed information about the request and modify or cancel the request during the various stages of making an HTTP request. This add-on keeps track of all requests and blocks requests from being sent, implementing the half-duplex communication without patching the codebase of Firefox itself. The destination server does not need any modification to accommodate the changes at the browser side.

B. PADDING

There needs a padding step to make cell sequences more similar to each other and thus, not fingerprintable, and there are several factors to consider in the padding step such as making a padding decision in real time. For a fair comparison, our implementation is simulated padding using a reference sequence. We pre-select the cell sequence with the largest number of cells as the reference sequence and used it to construct a “padded” data set for all websites and accesses as illustrated in Figure 1.

In other words, we align each cell sequence being padded with the reference sequence by bursts, then for each burst (incoming and outgoing), we add padding cells in the cell sequence to make the number of cells equal to that of the reference sequence at the same position. We add extra bursts at the end of the sequence to match the sizes and directions of the bursts in the reference sequence (see cell sequence A in Figure 1). If the number of cells in a burst in a sequence is larger than that of the reference sequence at the same position, we keep all those cells (see cell sequence B in Figure 1).

C. TAIL TIME

In our TT mechanism implementation, we record the timestamp for each request when a request is released and moved from the blocked queue to the outstanding queue. It is taken off the outstanding queue when it receives a response, or has stayed there for a period of time in the range of $[t − 1, t]$. In other words, all blocked requests can be sent at once in one burst when all previous requests get responses (as in WT) or have waited more than $t − 1$ seconds. Since both the blocked request queue and the outstanding request queue are checked every second in our implementation, with a tail timeout of $t$ seconds, a request may be sent with a delay no more than $t$ seconds. This design introduces further uncertainty into the traffic flow and may defend the traffic against attacks better.

IV. EXPERIMENT SETUP

In our experiment with the defense implementation described above, we accessed the top 100 websites and captured traces of Tor cell sequences during website accesses. We then conducted attacks against the traces to compare the attack accuracy between scenarios with WT or TT with several tail timeout values. We measured the time needed to load those websites in the WT and TT scenarios and compared them with the undefended case (no WT/TT).

A. DATA SET

We fetched Alexa’s top website list[32], which was also used by other researchers working on the same area[19, 34], and removed duplicated sub-domains (such as login.tmall.com and localized domains (such as google.co.in) from the list. We accessed the main page using the HTTP protocol (which redirects to the HTTPS version in many websites) to start the browsing session for each site.

It is noted that we conducted the experiment in April 2019 and reported the results in a conference paper[35]. We conducted another experiment in May 2020 with the following improvements:

- Among the 100 websites, we removed “bad” websites, which triggered errors when accessed even without WT or TT, i.e., accessed with Tor only. We suspect these websites have connectivity issues to the exit nodes of the Tor circuits being used[4]. In our previous experiment, we used “3 errors” out of 100 tries as the criterion of...
being bad and composed a list of 75 “good” websites. We used “1 error” during the new experiment, which resulted in 46 “good” websites. This pre-processing is important because the random behavior of the “bad” websites would give us unreliable results, preventing a fair comparison.

- In the previous experiment, we only accessed each website 50 times, but with an additional 5,000 unmonitored websites, forming an open-world scenario [36]. In the new experiment, we tested the closed-world scenario, where a trace being tested is always known to be from one of the monitored websites, and all those monitored websites have been visited during the training phase. Each website is visited 100 times for a more accurate analysis.

- In the previous experiment, we did not consider the impact of HTTP protocols. In the new experiment, we separate data in order to see the corresponding impact.

- We carefully arranged the website accessing schedule in the new experiment, and reset Tor circuits between accesses, to avoid biases. This was not considered in the previous experiments. In other words, the order of the accesses is determined such that it minimizes the uncertainty due to issues like network fluctuation and avoids biases, by grouping accesses to the same website together and having accesses using different timeout values interleaved.

### B. HTTP/2 AND HTTP/1.X

HTTP/1.x is still used by Internet websites and HTTP/2 is slowly replacing HTTP/1.x if both HTTP server and client’s web browser support it. A small-scale experiment has been conducted to see its impact on attacks and defenses. In Table 1, we present the attacking accuracy of four different attacks using different protocols. Some attacks show similar accuracy in different scenarios, while others show a significant drop in accuracy when HTTP/2 is used, and a further drop with server push enabled. Therefore, we conducted our further research separately by pinning the HTTP version to use, as shown in section V.

Table 1. Accuracy of different attacks on HTTP/1.x or HTTP/2 traffic without extra defense (no WT/TT).

| Attack                     | HTTP/1.x | HTTP/2 | HTTP/2 w/ push |
|----------------------------|----------|--------|-----------------|
| Jaccard [37]               | 99.7%    | 94.3%  | 86.4%           |
| Naive Bayes [37]           | 99.8%    | 87.9%  | 70.8%           |
| Multinomial Naive-Bayes [38]| 99.8%    | 99.1%  | 94.1%           |
| CUMUL [16]                 | 78.7%    | 30.3%  | 34.4%           |

To be more specific, we use the top 100 websites as monitored, with each website visited 100 times. For a total of 10,000 visits, each is done in two scenarios (HTTP/1.x-only and HTTP/2 when available). Considering HTTP/2 server push is rarely used currently, we do not test the HTTP/2 with server push scenario separately. For each scenario, we test the undefended case in which neither WT nor TT is used, the WT case, and TT cases with different timeout values: 1, 2, 3, 4.5, 6, 8, and 11 seconds, totaling 180,000 scenarios.

### C. ATTACKING EXPERIMENT

We ran $k$-Nearest Neighbors ($k$-NN) classifier-based attack [17] on the cell sequences, which is shown to achieve a much better performance compared to other attacks including SVM [13] and CUMUL [16]. During the attack, 1,225 features are extracted from each trace, and initial weights for those features are randomly chosen between 0.5 and 1.5. More recently, researchers have proposed other types of attacks, such as k-fingerprinting [39], p-FP [34] and Deep Fingerprinting [40], which outperform $k$-NN. We additionally used Deep Fingerprinting to confirm the defense efficiency of WT and TT.

To compare the defenses with and without the TT modifications, we accessed websites and logged Tor cell sequences for analysis. To obtain accurate Tor cell sequences, we modified the Tor client such that it emits individual cell information directly. Although this approach is not practical for an attacker in the real world, it would provide the most accurate cell sequences, allowing us an accurate and fair evaluation of fingerprinting attacking and defending algorithms operating on cell sequences. This approach differs from a conventional method based on TCP packet captures [19]. This conventional method reassembles TCP packets into TCP streams, then analyzes the TLS records to infer Tor cells based on the TLS record lengths. A major shortcoming is that, there is no guarantee that the original cells can be correctly reconstructed. Also, our approach works on a higher layer before an outgoing Tor SENDME cell is added and after an incoming
The Tor instance runs inside a Docker [41] container which is also restarted for each access. By running Tor inside a Docker instance and deleting/recreating the instance between each website access, the Tor connection is completely reset, which simplifies controlling and synchronization with Tor.

The controller starts both Tor cell capture in the logger and web browsing to the intended target in Firefox managed by Selenium together at the same time.

The Firefox instance is terminated after 15 seconds and the captured cell sequence is saved to a file as part of the data set.

A new Firefox profile is built, and a WT/TT implementation extension with proper tail timeout value configured for WT/TT accesses, which will be used in the Firefox instance.

The Firefox instance is controlled by Selenium [21] through geckodriver [22] and restarted for each access. Upon each restart, Selenium configures the Firefox instance to use the profile and extension built as mentioned above, and use a SOCKS proxy listened by a modified Tor instance, which emits cell information to a logger.

The controller starts both Tor cell capture in the logger and web browsing to the intended target in Firefox managed by Selenium together at the same time.

The Firefox instance is terminated after 15 seconds and the captured cell sequence is saved to a file as a part of the data set.

D. PAGE LOADING TIME MEASUREMENT

To better measure the user experience on real-world browsing through WT or TT, we use the same overall setup as the attacking experiment as discussed above.

For each access, Selenium starts web browsing to the target website with a page load timeout of 120 seconds, longer than the value used in the attacking experiment.

When the website is loaded, we read window.performance.timing in the website’s context to fetch the page loading timing. If any error occurs, including timeout, we record this error in the result. Otherwise, we record the contents in window.performance.timing as the result into the datastore.

Between accesses, the Firefox instance is terminated and restarted, and the user profile is reset.

In case of defense scenarios (WT/TT), the Firefox instance gets configured to use the Tor instance as a proxy, and have the WT or TT extension installed as a temporary extension with proper timeout value configured. Since we do not support real-time padding on our WT and TT implementations as explained in section III-B, the padding step is not included in this experiment. This does not affect the page loading time.

Also similar to the attacking experiment, we separate the groups of “HTTP/1.x only” and “HTTP/2 when available,” and configure the Firefox extension accordingly.

With all data collected, we process the data in the following way: for each access, we take the timestamps in the fields navigationStart, domLoading, domInteractive, and domComplete. We then calculate the time spent loading, interactive and complete as domLoading − navigationStart, domInteractive − navigationStart, and domComplete − navigationStart, respectively. We then remove all timing data if an error exists in the result, since in these cases, the timing may not reflect the real loading progress of the target website, for example, the time spent to show the error page may be recorded instead. Please refer to [36] for a more detailed explanation on the experiment setup and procedure.

V. RESULTS AND DISCUSSION

This section presents the page loading time (section V-A), and defense effectiveness (section V-B). We further looked into two cases – cnn.com and nytimes.com – to identify the source of the delay and see how they play out differently in WT and TT (section V-C). Note that, during our experiment and study, we observed WT accesses resulted in a large amount of failures, most of which are found to be the 120-second timeout. Detail analysis of those failures is presented in section V-D. In the rest of this section, “WT (*)” denotes the results without such failed accesses, and “WT” show data with such accesses counted as finishing at the 120th second.

A. PAGE LOADING TIME

Page loading time can be considered a direct measure of user experience. Figure 3 shows the page loading time, measured by the domComplete event that fires when the whole page completes loading including all the resources. We made the following observations based on the experiment results.

As expected, both WT and TT increase the page loading time in comparison to undefended Firefox.

More specifically, in Figure 3(a) with HTTP/1.x, WT and TT (tail time of 2 seconds) increase the average page loading time (CDF 0.5) by 51.6% and 31.6%, respectively. Comparing WT and TT, WT clients are forced to wait 20% more time to complete the page loading than TT clients. Similar trends are observed in Figure 3(b) with HTTP/2.

About 20% of websites (CDF 0.8) render WT clients to wait a very long time as shown in both Figures 3(a) and 3(b).

HTTP/2 helps reduce the page loading time of both WT and TT comparing Figure 3(a) and 3(b). This is anticipated because HTTP/2 alleviates the head of line blocking problem of requests for assets. Nonetheless, WT creates blocking in all these scenarios and thus, it still shows a longer page loading time than TT as...
shown in Figure 3(b). Note that a common approach with HTTP/1.1 adopted by many websites is to arrange assets on multiple hosts so browsers can access the assets in parallel by opening multiple connections instead of pipelining the accesses, but HTTP/1.x still shows a higher page loading time.

- For comparison, we showed the page loading time measured by the `domInteractive` event in place of `domComplete`. The `domInteractive` event comes earlier when the necessary information needed to complete the DOM tree is ready, which is usually just the document and synchronous scripts, but without any styling information or media resources. The result is shown in Figure 4. The difference between WT and TT 2 follows the same trend as in Figure 3 but it is not as significant as in Figure 3. This explains that the difference we observed in Figure 3 comes from the interplay in the network.

B. DEFENSE EFFECTIVENESS

We also want to verify that TT does not increase the fingerprintability on the traffic. We ran attacks on the cell sequences using WT and TT with different timeout values, compared with undefended browsing scenarios to measure the accuracy, defined as the percentage of correct attacks among all attacks run in that group. We plot the attack accuracy of k-NN and Deep Fingerprinting in Figures 5 and 6 with HTTP/2 and HTTP/1.x, respectively.

It is shown that both WT and TT result in a significantly lower (27% to 51%) attack accuracy than the undefended case. Note that the absolute values of attacking accuracy are lower than those reported by Wang et al. [19] and Sirinam et al. [40], for both defended and undefended cases. The reason can be that our trace collection procedure is different and more accurate as described in section IV-C. The choice of HTTP/1.x or HTTP/2 does not make a major difference in terms of accuracy.

C. WATERFALL CHARTS

To further identify the source of the delay, we captured the resource-by-resource timing information during page loading. For each request, we capture the following four timestamps during page loading.
FIGURE 5. Attack accuracy of k-NN and Deep Fingerprinting on accesses with HTTP/2 enabled. "Undef." refers to undefended traffic.

FIGURE 6. Attack accuracy of k-NN and Deep Fingerprinting on accesses with HTTP/1.x. "Undef." refers to undefended traffic.

- Request time: the time when the browser initially processes the request and puts the request into the blocked queue.
- Release time: the time when the request is released by the add-on from the blocked queue to the outstanding queue.
- Respond time: the time when the response of the request is received, or the request is otherwise finished (e.g., network error).

The timing information is recorded into logging data in the WT/TT implementation add-on to the Firefox browser when requests are processed.

We tested nytimes.com and cnn.com for page loading, and plotted the timing data (of selected requests) for nytimes.com in Figure 7 and for cnn.com in Figure 8.

We can notice that in Figure 7(a), when no TT is applied, certain requests, such as requests 6 and 7 took a long time to finish, while other requests in the same burst finished within one or two seconds. The requests in the following burst, 22 and 23, are released later at around 14 seconds and completed at around 16 and 17 seconds. In Figure 7(b) with TT, while the request for resource 6 and 7 are still on the way, the following burst of requests including 22 and 23 are released at around 4 seconds and completed at around 5 seconds, which is around 12 seconds earlier than in Figure 7(a). In other words, TT significantly reduces the page loading time by limiting the maximum delay that can be caused by slow requests. We noticed the same trend with cnn.com in Figure 8. A few bottleneck requests may block the whole browsing session in WT.

We further tracked down the slow request 7 on cnn.com. By looking at the page source code of cnn.com, we noticed that request 7 was introduced in the HTML document with a <script> tag with the async attribute set. Request 7 points to https://native.sharethrough.com/assets/sfp-creative-hub-listener.js and Sharethrough is an advertisement platform. As we pointed out before, the resource fetched by the request is exactly for advertisements and should have a relatively lower priority. Web developers of cnn.com correctly specified the lower priority for these resources by using the async attribute, then by design, the browser tends to load these resources at a later time and gives priority to other resources. However, with WT’s half-duplex communication, no other requests can be sent. This situation can be resolved only when the browser finishes the low priority request, probably after an internal timeout, or with the designed TT in the add-on.

In addition to the lower priority requests, there can also be requests to a slow remote server that take a relatively long time. This scenario was not observed in Figures 7 and 8, but we can predict that such scenarios can happen and cause a similar delay for WT.

D. ACCESS FAILURES IN WT

As discussed earlier, WT accesses resulted in an extended delay or access failures. For example, on some popular content-oriented websites such as cnn.com, nytimes.com, and sina.com.cn, the loading times over the Tor network are shown in Table 2. The cnn.com and nytimes.com webpages could not finish loading within 120 seconds when using WT and we terminated page loading at 120 seconds, which are considered access failures, and it is over 75 seconds for sina.com.cn.

Figure 9 shows the number of timeout access failures in WT scenarios with 46 "good" websites over Tor. Overall, the failure rate with WT is 52% on average for HTTP/1.x and 54% for HTTP/2. It is as high as 90% or more in 10 and 11 websites for HTTP/1.x and HTTP/2 respectively. Upon further inspection, most of the access failures are the 120-second timeout, although the average loading time of successful visits are not necessarily close to 120 seconds. In other words, on some websites, WT can complete loading in an acceptable time in one visit but not in another visit. On the user interface, this can be the behavior where the

| Website     | Undef. | TT 1  | TT 2  | TT 3  | TT 4  | TT 5  | TT 6  |
|-------------|--------|-------|-------|-------|-------|-------|-------|
| cnn.com     | 46.758 | >120  | 59.231| 67.160| >120  |
| nytimes.com | 26.262 | >120  | 30.335| 37.370| 41.306|
| sina.com.cn | 35.094 | 75.404| 42.075| 46.530| 66.359|

TABLE 2. Loading time (domComplete event, seconds) of certain websites at some different tail timeouts.
FIGURE 7. Loading nytimes.com main page with original WT and with TT of 2 second. The beginning and the end of a line denote the request time and respond time, respectively. A black or white block shows the time when the request is released, i.e., the release time. Neighboring requests with blocks in the same color, e.g., (2) through (21), are in the same burst. In Figure (a), the burst containing requests (22) through (31) is held by the requests (6) and (7), which is not the case in Figure (b). See [36] for more details including URLs.

FIGURE 8. Loading cnn.com main page with original WT and with TT of 1 second. Request (7) blocks the following burst in Figure (a), which is not the case in Figure (b). See [36] for more details including URLs.

FIGURE 9. Failed page loads in WT due to timeout, i.e., accesses where the web page does not finish loading in 120 seconds. (The failed page loads are very rare, i.e., less than 1% in both TT and undefended cases.)

Other than the possibility that WT makes page loading really slow, another reason is that WT’s half-duplex communication may cause a page loading blocked indefinitely due to the fact that the web scripts are full of asynchronous code/functions. For example, a web page includes a local script (same webserver) that creates a box for an ad placement, and includes a remote script (from some ad platform) that places an ad in the box. In usual cases, the local script loads faster so the box is ready when the remote script tries to place the ad. But when we delay the local script until the remote script request completes, it causes some problem.

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented a new delay-conscious defense mechanism, called Tail Timeout (TT), against website fingerprinting attacks based on WT, one of the most effective defense. We demonstrated that WT could significantly increase the page loading time for many popular websites.
By analyzing the websites experiencing inflated page loading time under WT, we found that the contents marked with low priority, such as online ads could incur such delays in practice. Our experimental results indicate that the proposed TT mechanism can significantly reduce the page loading time while keeping similar defense performance achieved by the original WT. We also presented a future-proof and highly accurate experiment framework which will be useful for other researchers in evaluating fingerprinting attacks and defending algorithms.

In our research, we arbitrarily selected the tail timeout values to test, and we have not identified an approach to determine the optimum values. We are leaving this as future work, as well as investigating the trade-off between the defense effectiveness and user experience. In the future, we also plan to try other attack models, such as naïve Bayes [42] or more trending deep learning-based classifiers [34], or on larger data sets, to evaluate the performance of TT. In addition, we plan to apply TT mechanism to other defenses that potentially increase page load times due to the same head-of-line issue.

REFERENCES

[1] R. Deibert, J. Palfrey, R. Rohozinski, J. Zittrain, and J. G. Stein, Measuring Global Internet Filtering, Cambridge, MA, USA: MIT Press, 2008, pp. 5–27.

[2] C. S. Leberknight, M. Chiang, and F. M. F. Wong, “A taxonomy of sensors and anti-sensors: Part I: Impacts of internet censorship,” Int. J. E-Politics, vol. 3, no. 2, pp. 52–64, Apr. 2012, doi: 10.4018/jep.2012040106.

[3] C. S. Leberknight, M. Chiang, and F. M. F. Wong, “A taxonomy of sensors and anti-sensors: Part II: Anti-censorship technologies,” Int. J. E-Politics, vol. 3, no. 4, pp. 20–35, Oct. 2012, doi: 10.4018/jep.2012100102.

[4] P. Winter and S. Lindskog, “How the great firewall of China is blocking Tor,” in Proc. 2nd USENIX Workshop Free Open Commun. Internet (FOCI). Bellevue, WA, USA: USENIX Association, Aug. 2012, pp. 1–7. [Online]. Available: https://www.usenix.org/conference/fohci2/workshop-program/presentation/winter

[5] Z. Wang, Y. Cao, Z. Qian, C. Song, and S. V. Krishnamurthy, “Your state is not mine: A closer look at evading stateful internet censorship,” in Proc. Internet Meas. Conf. (IMC). New York, NY, USA: Association for Computing Machinery, Nov. 2017, pp. 114–127, doi: 10.1145/3131365.3131374.

[6] BBC News. (Oct. 2019). BBC News Launches ‘Dark Web’ Tor Mirror. [Online]. Available: https://www.bbc.com/news/technology/50150981

[7] J.-P. Verkamp and M. Gupta. “Inferring mechanics of web censor-ship around the world,” in Proc. 2nd USENIX Workshop Free Open Commun. Internet (FOCI). Bellevue, WA, USA: USENIX Association, Aug. 2012, pp. 1–7. [Online]. Available: https://www.usenix.org/conference/fohci2/workshop-program/presentation/verkamp

[8] R. Clayton, S. J. Murdoch, and R. N. M. Watson, “Ignoring the great firewall of China,” in Proc. 6th Int. Conf. Privacy Enhancing Techn. (PET). Berlin, Germany: Springer-Verlag, 2006, pp. 20–35, doi: 10.1007/11597454_2.

[9] S. Aryarn, H. Aryarn, and J. A. Halderman, “Internet censorship in Iran: A first look,” in Proc. 3rd USENIX Workshop Free Open Commun. Internet (FOCI). Washington, DC, USA: USENIX Association, Aug. 2013, pp. 1–8. [Online]. Available: https://www.usenix.org/conference/fohci3/workshop-program/presentation/aryarn

[10] K. Ko, H. Lee, and S. Yang, “The internet dilemma and control policy: Political and economic implications of the internet in North Korea,” Korean J. Defense Anal., vol. 21, no. 3, pp. 279–295, Sep. 2009, doi: 10.1080/13207900903087204.

[11] J. Holowczak and A. Houmansadr, “CacheBrowser: Bypassing Chinese censorship without proxies using cached content,” in Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur. (CCS). New York, NY, USA: Association for Computing Machinery, Oct. 2015, pp. 70–83, doi: 10.1145/2810103.2813696.

[12] C. Brubaker, A. Houmansadr, and V. Shmatikov, “CloudTransport: Using cloud storage for censorship-resistant networking,” in Proc. 14th Int. Symp. Privacy Enhancing Technol. Symp. (PETSy), in Lecture Notes in Computer Science, vol. 8555, E. D. Cristofaro and S. J. Murdoch, Eds. Amsterdam, The Netherlands: Springer, Jul. 2014, pp. 1–20, doi: 10.1007/978-3-319-08506-7_1.

[13] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel, “Website fingerprinting in onion routing based anonymization networks,” in Proc. 10th Annu. ACM Workshop Privacy Electron. Soc. (WPES). New York, NY, USA: Association for Computing Machinery, 2011, pp. 103–114, doi: 10.1145/2046556.2046670.

[14] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, “Peek-a-boo, I still see you: Why efficient traffic analysis countermeasures fail,” in Proc. IEEE Symp. Privacy Secur. (PSS), May 2012, pp. 332–346.

[15] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, “A survey of methods for encrypted traffic classification and analysis,” Int. J. Netw. Manage., vol. 25, no. 5, pp. 355–374, Sep. 2015.

[16] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel, “Website fingerprinting at internet scale,” in Proc. 23rd Netw. Distrib. Syst. Secur. Symp. (NDSS). San Diego, CA, USA: The Internet Society, 2016, pp. 1–15. [Online]. Available: http://www. internetociety.org/nnds/wp-content/uploads/sites/25/2017/09/website-fingerprinting-internet-scale.pdf

[17] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg, “Effective attacks and provable defenses for website fingerprinting,” in Proc. 23rd USENIX Secur. Symp. (USENIX Security). San Diego, CA, USA, Aug. 2014, pp. 143–157. [Online]. Available: https://www. usenix.org/conference/usenixsecurity14/technical-sessions/presentation/wang_tao

[18] R. Ensafi, J. Knockel, G. Alexander, and J. R. Crandall, “Detecting intentional packet drops on the internet via TCP/IP side channels,” in Proc. 15th Int. Conf. Passive Act. Meas. (PAM). Paris, 2014, pp. 109–118, doi: 10.1007/978-3-319-04918-2_11.

[19] T. Wang and I. Goldberg, “Walkie-Talkie: An efficient defense against passive website fingerprinting attacks,” in Proc. 26th USENIX Secur. Symp. (USENIX Security). Vancouver, BC, Canada: USENIX Association, Aug. 2017, pp. 1375–1390. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-tao

[20] R. Dingledine, N. Mathewson, and P. Syverson, “Tor: The second-generation onion router,” in Proc. 13th USENIX Secur. Symp. (USENIX Security). San Diego, CA, USA: USENIX Association, Aug. 2004, pp. 203–220. [Online]. Available: https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router

[21] Software Freedom Conservancy. Selenium. Accessed: Apr. 15, 2019. [Online]. Available: https://www.selenium.dev/

[22] Mozilla Foundation. Geckodriver. Accessed: Apr. 15, 2019. [Online]. Available: https://github.com/mozilla/geckodriver

[23] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright, “Toward an efficient website fingerprinting defense,” in Comp. Security—ESORICS 2016. I. Askoylakis, S. Ioannidis, S. Katsikas, and C. Meadows, Eds. Cham, Switzerland: Springer, 2016, pp. 27–46.

[24] T. Wang and I. Goldberg, “Improved website fingerprinting on Tor,” in Proc. 12th ACM workshop Workshop Privacy Electron. Soc. (WPES). New York, NY, USA: Association for Computing Machinery, Nov. 2013, pp. 201–212, doi: 10.1145/2517840.2517851.

[25] T. Wang and I. Goldberg, “On realistically attacking Tor with website fingerprinting,” Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS). San Diego, CA, USA: The Internet Society, Feb. 2009, pp. 1–14. [Online]. Available: https://www.ndss symposium.org/ndss2009/traffic-morphing-efficient-defense-against-statistical-traffic-analysis

[26] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg, “A systematic approach to developing and evaluating website fingerprinting defenses,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS). New York, NY, USA: Association for Computing Machinery, Nov. 2014, pp. 237–238, doi: 10.1145/2600267.2600362.
JINGYUAN LIANG received the B.S. degree in chemical engineering from Tsinghua University, Beijing, China, in 2014. He is currently pursuing the Ph.D. degree in computer engineering with Cleveland State University, Cleveland, OH, USA. From 2015 to 2021, he was a Graduate Assistant with Cleveland State University. His research interests include cybersecurity and parallel computing. Mr. Liang was a recipient of the Cleveland State University 2020–2021 Graduate Student Research Award.

KYOUNGWON SUH (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Seoul National University, South Korea, in 1991 and 1993, respectively, and the Ph.D. degree from the Department of Computer Science, Rutgers University, New Jersey, in 2000, and the Ph.D. degree in computer science from the University of Massachusetts at Amherst, in 2007. From 1993 to 1998, he was a Research Engineer at LG Electronics, Inc. Since 2007, he has been a Computer Science Professor with the School of Information, Illinois State University, Normal, IL, USA. His research interests include mobile handheld devices, wireless networks, network measurement and inference, network security, and distributed systems.

KYOUNGWON SUH (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Seoul National University, South Korea, in 1991 and 1993, respectively, and the Ph.D. degree from the Department of Computer Science, Rutgers University, New Jersey, in 2000, and the Ph.D. degree in computer science from the University of Massachusetts at Amherst, in 2007. From 1993 to 1998, he was a Research Engineer at LG Electronics, Inc. Since 2007, he has been a Computer Science Professor with the School of Information, Illinois State University, Normal, IL, USA. His research interests include mobile handheld devices, wireless networks, network measurement and inference, network security, and distributed systems.

CHANSU YU (Senior Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from Seoul National University, South Korea, in 1982 and 1984, respectively, and the Ph.D. degree in computer engineering from The Pennsylvania State University, in 1994. From 1984 to 1997, he was a Research Staff at LG Electronics, Inc. From 1997 to 2001, he was an Assistant Professor at ICT, South Korea. Since 2001, he has been a Professor with the Department of Electrical Engineering and Computer Science, Cleveland State University, Cleveland, OH, USA. He has authored/coauthored more than 100 technical articles and numerous book chapters in the areas of hardware security, mobile networking, performance evaluation, and parallel and distributed computing. He holds two patents. Dr. Yu is a member of the ACM and ACM SIGCSE. He is an Editor of the ETRI Journal.

HYOIL HAN (Member, IEEE) received the B.S. degree in electrical engineering from Korea University, in 1984, the M.S. degree in electrical engineering from the Korea Advanced Institute of Science and Technology, South Korea, in 1989, and the Ph.D. degree in computer science and engineering from The University of Texas at Arlington, in 2002. She worked with Samsung Electronics (1984–1985) and Korea Telecom (1989–1993) as a Member of Technical Staff in South Korea, before obtaining her Ph.D. degree. She is currently an Associate Professor with the School of Information Technology, Illinois State University, Normal, IL, USA. She published over 60 technical articles related to her research. Her research interests include machine learning, natural language processing, big data management, and applying AI to security. Dr. Han is a member of the ACM.