Understanding and Addressing Concept Drift in Website Fingerprinting

Anatoly Shusterman, Roie David, Yossi Oren

Ben-Gurion University of the Negev

Abstract

Website fingerprinting attacks let attackers determine which websites a user visits, posing a significant risk to online privacy. Website fingerprinting attacks have been demonstrated both in the network-based setting, where the adversary is able to observe the network traffic between the user and the secure network, and in the cache-based setting, where the adversary injects malicious JavaScript code into the user's browser and observes memory activity. In both of these settings, previous research has demonstrated that state-of-the-art website fingerprinting attacks can succeed even in highly-constrained environments, for example when attacking the privacy-enhanced Tor browser. One known limitation of website fingerprinting attacks, however, is their sensitivity to *concept drift* — as the time difference between the training period and the actual attack grows, the accuracy of the model used in the fingerprinting attack degrades due to changes in webpage content, network conditions, and the software implementation of the browser. This phenomenon is a known challenge in website fingerprinting.

This study provides a quantitative analysis of the effect on website fingerprinting attacks of concept drift in both the network-based and cache-based settings, based on a website fingerprinting trace dataset collected over a period of several months. It examines the effect on accuracy of changes both to the browser version and to the website content. It then investigates multiple approaches for addressing concept drift, assuming the attacker can add a limited amount of fresh data to his training dataset.

Our evaluation shows that using advanced machine learning techniques can significantly reduce the effect of concept drift on website fingerprinting attacks, and that, specifically, an incremental learning approach nearly maintains the model's accuracy over time, while requiring only 2% of the data needed for full retraining. This finding demonstrates the practicality of long-term website fingerprinting attacks in the real world.

Keywords: Communication Network Protocols, Network Security and Privacy, Side Channel Analysis

1. Introduction

The importance of private and secure web browsing is evident. Given access to a user's browsing activities, an adversary can easily learn about the user's affiliations, sexual orientation, or political views. Exposing this information could be dangerous to the user, and even risk the user's life in some regions of the world. As the amount of sensitive information being transmitted over the Internet grows, it is crucial that users have confidence in the privacy of their web browsing activity.

One significant development in protecting user privacy has been the upgrade from HTTP to HTTPS, which utilizes Transport Layer Security (TLS) to encrypt data, and thus prevents on-path adversaries from reading and modifying browsing sessions. Even the identity of source and the destination of browsing traffic, which was traditionally available to on-path eavesdroppers [30], has been recently become harder to detect due to the growing use of Encrypted Server Name Indication (ESNI) [17], virtual private networks (VPNs) and the Tor network [47].

Nevertheless, even in the absence of explicit knowledge of the data being exchanged, and of the server being accessed, an adversary can still indirectly infer the user's browsing activity by analyzing *side-channel metadata*, such as the statistical characteristics of network traffic, or the activity of the CPU cache on the user's computer. This approach is called *website fingerprinting*, and has been shown to be effective in identifying the

websites a user visits, even in the presence of the most advanced network security and privacy defenses [15]. Many studies have explored fingerprinting approaches, and have suggested defenses against this attack [42, 18, 37].

1.1. Threat Model

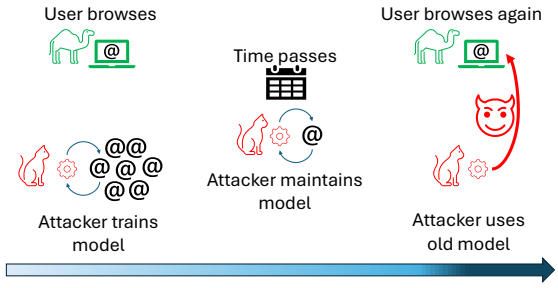


Figure 1: The attacker monitors the user's side-channel metadata and attempts to discover the user's browsing activity. As time passes, the performance of the attack degrades due to concept drift.

The two actors in the website fingerprinting threat model are the **user** and the **attacker**. The user browses the web over a secure network, and may potentially access sensitive websites. The attacker's objective is to learn which websites the user is currently visiting. It is assumed that the attacker can neither directly observe the content of the user's network traffic, due to the user of techniques such as TLS and ESNI, nor can the attacker run malicious software on the user's computer. Instead, the attacker must indirectly observe the user's browsing activity by collecting side-channel metadata, such as network traffic characteristics or cache activity, and then use these traces to infer the user's browsing activity.

As discussed further in Section 2, there are two main approaches to website fingerprinting: *network-based website fingerprinting (NBWF)* and *cache-based website fingerprinting (CBWF)*, which differ in the sources of side-channel data available to the attacker. Regardless of the chosen approach, the attacker's general strategy is to collect side-channel traces related to the loading and rendering of various websites, use these traces to train a machine-learning classifier, and then later apply this classifier to previously-unseen traces to try and detect the website currently being visited by the user. As already observed by Juarez et al. [18], one main challenge on the attacker's side is the problem of *concept drift*. As indicated in Figure 1, as the time difference between the data used for training and the data used for testing grows, the classifier's accuracy degrades, due to changes both in the content of the websites being visited, and in the software implementation of the browser. We capture this time difference formally as Δ_{TT}, defined as the time in weeks between the training and testing of the classifier.

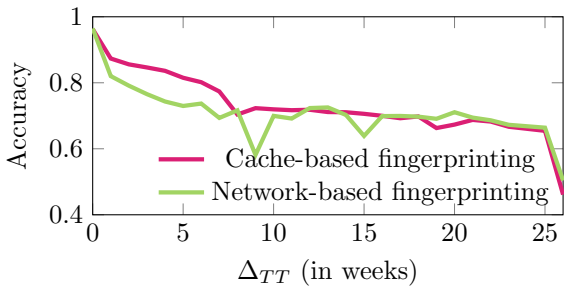


Figure 2: Website fingerprinting accuracy degrades over time due to concept drift.

The attacker's capabilities, knowledge and objective are summarized in Table 1.

1.2. Motivation

Figure 2 summarizes an experiment further described in section 4. The figure illustrates how both CBWF and NBWF accuracy degrades from over 94% when $\Delta_{TT} = 0$ to around 50% (above a base rate of 1%) as Δ_{TT} increases. To deal with concept drift, an attacker must either settle for a significant decline in accuracy over time, or periodically collect fresh side-channel traces and retrain the classifier. Constantly retraining the classifier poses a formidable data gathering burden on the adversary, and may call into question the very practicality of website fingerprinting, as noted by Juárez et al. [18], who commented that "[this] strong effect of time in the classifier's accuracy poses a challenge to the adversary who will need to train the classifier on a regular basis. Depending on the size of the world that he aims to cover, the cost of training may exceed the time in which data provides reasonable accuracy rates ". Understanding the concept drift effect, and considering methods for addressing it, is the main focus of this paper.

While concept drift has not been addressed previously in the specific context of website fingerprinting, it is a well-known challenge in other fields which use machine learning to make predictions. Many methods have studied concept drift and its effects,

Property	Cache-based Website Fingerprinting	Network-based Website Fingerprinting
Capability	Collect cache activity traces (via injected JavaScript)	Collect network traffic traces (via on-path observation)
Knowledge	Victim's computer configuration, closed list of potential websites.	
Objective	Identify the website currently being visited by the user.	
Limitations	Cannot directly access website contents or identify server address.	

Table 1: Website fingerprinting threat model summary.

and have proposed methods to address it in other domains such as spam detection, intrusion detection, and price forecasting [25]. In our work, we set out to understand the effect of concept drift on website fingerprinting attacks, and to propose a methodology for dealing with it effectively.

Our work has two primary goals. The first goal of our work is to analyze the effect of the two factors which contribute to concept drift in the website fingerprinting attack scenario: web content drift, and browser version drift. We do this by analyzing data from both website fingerprinting attack domains – cache-based website fingerprinting and network-based website fingerprinting. Our second goal is to evaluate state-of-the-art approaches for addressing concept drift, in the specific context of website fingerprinting. In particular, we study how these different approaches allow a trade-off between the *fresh data budget* available for retraining the model and the model's performance over time.

More specifically, in this paper we answered the following research questions:

How does concept drift affect the accuracy of website fingerprinting attacks? How do changes to website content and to browser versions contribute to this effect? Can advanced machine-learning methods be used to mitigate this effect with a reasonable re-training budget?

To answer these questions, we collected a unique website fingerprinting dataset that includes network and cache traces for 100 popular websites, collected weekly over a period of 26 weeks and spanning 16 consecutive versions of the popular Chrome browser. We then evaluated the effect of concept drift on the accuracy of website fingerprinting, and compared the robustness of both cache side-channel data and network side-channel data to this effect. We discovered that the main cause of concept drift is the change in the content of the websites being visited, while changes in browser version had a lesser effect. Next, we investigated three modern approaches for addressing concept drift with partial retraining: incremental learning, transfer learning, and N-shot learning. We quantitatively evaluated the effectiveness of these approaches using linear regression analysis. Our results show that incremental learning can practically maintain the model's accuracy over time, while requiring only a small fraction of the data needed for full retraining.

Document structure: The rest of this paper is organized as follows: section 2 provides background information on website fingerprinting attacks, cache side-channel attacks, and concept drift. section 3 reviews related work on website fingerprinting attacks and concept drift. section 4 describes our methodology for collecting and analyzing website fingerprinting traces, and for evaluating the effect of concept drift. section 5 presents the results of our evaluation, and section 6 discusses the implications of these results. Finally, section 7 concludes the paper and suggests directions for future work.

2. Background

2.1. Website Fingerprinting

The first to use the term *fingerprinting* was Hintz [15], who demonstrated the feasibility of such an attack. As Hintz observed, the unique Document Object Model (DOM) structure of webpages, in which objects such as text, images, videos and scripts are activated in a deterministic order, results in side-channel information which is similar for multiple visits to the same webpage, but varies for visits to different webpages. Thus, by collecting and statistically analyzing this metadata, an attacker can identify the website, or more precisely, the exact webpage, being accessed, even without direct access to the content of the communication or to the computer under attack.

The website fingerprinting attack consists of several stages:

First, the attacker collects a dataset containing several leakage traces for each website the attacker wants to identify. Next, the attacker trains a machine learning classifier on the collected traces. This can either be done through classical machine learning approaches, where the attacker first extracts features from the data, or through a deep learning pipeline which inputs the raw trace data. Finally, at some later date, the adversary applies the trained classifier to leakage traces collected from the victim's machine, and uses it to infer the website the victim is visiting.

2.1.1. Network-Based Website Fingerprinting

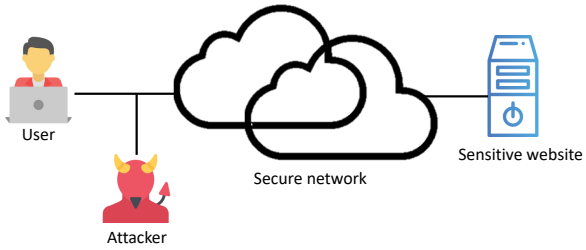


Figure 3: The user visits a website over a secure network. The attacker observes the traffic between the user and the secure network.

Website fingerprinting was originally proposed for a *man-in-the-middle* (MITM) network attacker [15]. Such an attack, illustrated in fig. 3, assumes an *on-path* adversary, who is able to observe the network traffic between the user and the secure network. The side-channel information retrieved from the monitored traffic includes the source IP address, which identifies the victim's machine, as well as packet characteristics such as packet direction, length, and timing. Note that the adversary cannot read the content of the packets, as it is encrypted, nor can he see the destination addresses of the traffic, which is assumed to be hidden by the use of Tor, a VPN, or other secure network techniques. Under this threat model, state-of-the-art works using deep learning techniques have shown that website fingerprinting attacks can achieve accuracies approaching 100% [37].

2.1.2. Cache-Based Website Fingerprinting

In parallel to the network-based approach, another line of works, starting with Clark et al. [6], investigated the use of local side-channel information, collected on the user's computer, to perform website fingerprinting. As illustrated in fig. 4, in this case the attacker is not located on the network, but is rather running unprivileged code on the user's machine which can indirectly observe the user's browsing activity. One well-explored vector for delivering this code is through a web page which contains malicious JavaScript code that is downloaded to the user's browser and run in the background.

Shusterman et al. [41], in particular, showed how the cache side-channel can be used to successfully perform an website fingerprinting attack in this setting. The next subsection provides more details on cache side-channel attacks and their use in the web context.

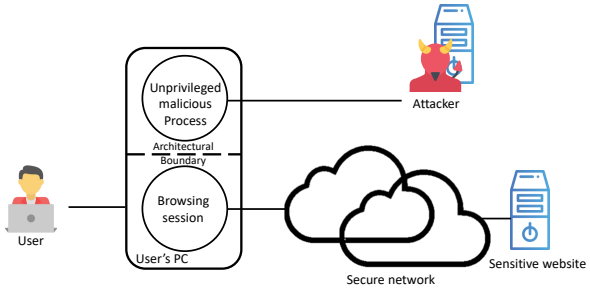


Figure 4: The remote adversary injects malicious JavaScript code into a browser running on the target machine

2.2. Cache Structure and Operation

The cache is a data structure used to temporarily store frequently-accessed data. It is typically smaller and faster than the computer's main random-access memory (RAM). By storing the most-recently used data in cache memory stores, CPU prevents itself from having to obtain data from the slower RAM, significantly improving performance. Caches typically form a hierarchy of progressively larger and slower cache levels, with the last-level cache (LLC) being shared among all cores, threads, processes, and even virtual machines running on a given CPU chip [60], as shown in fig. 5. The LLC is divided into multiple cache sets, each of which is responsible for a subset of the computer's memory. When the processor needs to access data from the memory, it first checks the relevant sets in the different caches for the data. If the data is found in any of the caches, an event referred to as a *cache hit*, the processor can quickly access the data [14].

If the data is not found in the cache, the processor must access data from the slower main memory, an event referred to as a *cache miss*.

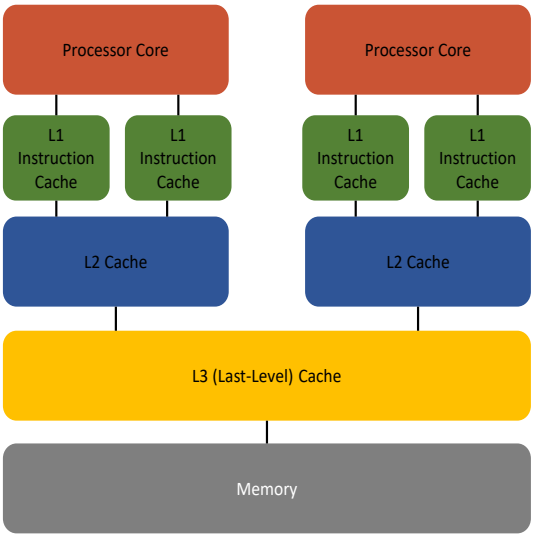


Figure 5: Last level cache (LLC) architecture

2.2.1. The Prime+Probe Attack

When multiple processes run on a single processor, they share the use of the processor's resources, including the cache. This sharing may result in unintended communication channels between programs [16]. Prime+Probe [29] is an attack that exploits unintended communication through the cache. Using this attack, an adversary can use a process to steal confidential information from other processes running on the same processor, including highly-protected information such as cryptographic keys, as shown in [1, 29, 49]. The Prime+Probe attack consists of three main steps: Prime, Idle and Probe, as demonstrated in figure 6.

1. **Prime.** In the Prime step, an adversary fills one or more cache sets with its own code or data.
2. **Idle.** In the idle step, the adversary waits for the victim to use the cache.
3. **Probe** In the final Probe step, the adversary stops waiting and measures the time it takes to load each set of data or code that they primed.

Because of the difference in timing between memory and cache access, a longer access time during the probe step indicates that the victim used the cache and evicted the adversary, while a shorter

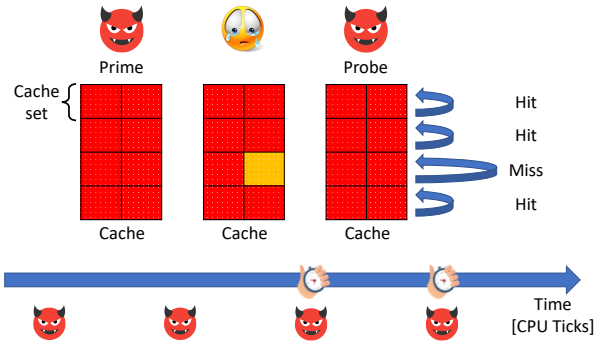


Figure 6: Prime+Probe flow

access time indicates that the victim did not access the cache. As a result, the adversary will be able to determine which cache sets the victim accessed. Liu et al. [23] presented an effective implementation of the Prime+Probe side-channel attack against the LLC. Concurrently, Maurice et al. [27] proposed C5, a cross-VM covert channel which does not measure contention in specific cache sets, but instead measures contention over the whole cache.

2.2.2. Browser-Based Cache Side-Channel Analysis

Variants of the Prime+Probe and C5 attacks have been demonstrated in the web context [28, 62]. This variant is particularly dangerous due to its minimal permission requirements, as no special privileges are needed for its execution. Additionally, this technique can be used to attack a variety of micro-architectures [43, 13], such as those used by Intel, ARM, and Apple, without the need to tailor the attack to each architecture. This attack can even be successfully performed when JavaScript is completely disabled on the victim's device [43].

In order to perform browser-based cache side-channel analysis, an attacker first needs to create malicious web content designed to probe the memory activity of the victim's device. This content can be a website, an advertisement, or even a seemingly harmless image or video file. Once the malicious content has been created, the attacker can use various means to deliver it to the victim's device, for example, by sending it via email or by hosting it on a malicious website. Once the victim's device loads the malicious content, the attacker can continuously monitor its LLC activity over time. The resulting side-channel trace can be used to compromise the victim's privacy, just like the network activity trace is used in network-based website fin-

gerprinting attacks.

2.3. Concept Drift

In many real-world phenomena, the distribution of data changes over time. Examples include weather forecasts, which change over the year, customer buying preferences, which follow changing trends and fashions, and even the popularity of different words and phrases in language, which change as the language evolves over time. When a machine learning model is trained on data from one time period and then tested on data from a different time period, the model's accuracy may degrade due to these changes in the data distribution. This phenomenon is known as *concept drift* [50]. Works on concept drift in academic literature attempt to develop frameworks and algorithms for concept drift detection, to understand the source of the drift to model its influence on the data source, and to devise methods to adjust the machine learning model to minimize the drift [25].

Concept drift is a significant problem in the website fingerprinting domain. Websites change over time as their structure and the objects they contain (e.g., like photos, videos, and scripts) change. In addition, the browser itself changes over time as its software is updated, resulting in differences in the way it fetches and renders web content. Due to both of these reasons, the side-channel traces generated by browser activity change over time, leading to a drift between the current traces and the ones used to train the website fingerprinting machine learning model. This causes the accuracy of the existing model to decrease, since it cannot match the leakage trace to the target class. Previous works in the network domain measured the degradation of the classification model's accuracy over time [18, 37]. They found that within a few weeks, the model's accuracy percentages dropped dramatically. A preliminary experiment on the cache-based fingerprinting scenario was performed in [44]. The results of this experiment showed that the cache-based fingerprinting is also significantly affected by concept drift. To date, however, there has been no attempt to characterize and address concept drift in the website fingerprinting domain using state-of-the-art methods explored in other domains [12, 59, 53, 10, 8, 9].

3. Related Work

3.1. Website Fingerprinting

Research has shown that network-based website fingerprinting attacks are feasible, even on the highly-secure Tor network [31, 54, 18, 32]. Those studies have shown that it is feasible to manually extract features from the collected traffic traces and use machine learning algorithms to classify future traces. As shown in [18], in many cases, the choice of features could be more important than the choice of an algorithm. Sirinam et al. [46] proposed a new website fingerprinting attack based on a convolutional neural network (CNN). Their attack outperformed previous attacks on the Tor browser and was evaluated against state-of-the-art defenses (i.e., WTF-PAD [19], and Walkie-Talkie [55]).

3.1.1. Website Fingerprinting on Secured Networks

Rimmer et al. [36] investigated the use of deep learning techniques for website fingerprinting and performed the first systematic exploration of state-of-the-art deep learning algorithms applied to website fingerprinting. As part of their study, the authors collected a dataset comprised of over three million network traces. They were the first to automate the feature extraction step for classification using deep learning.

Several studies looked at the possibility of performing website fingerprinting based on local side-channel information, particularly through the LLC [28, 44]. Oren et al. [28] explored the possibility of fingerprinting via malicious JavaScript code. The authors presented a Prime+Probe cache attack that can be launched from within sandboxed JavaScript in a browser and showed how to infer visited websites and track the user's mouse activity. They also extended the attack from [23] to LLC in the more common case of 4KB-sized pages.

Other studies proposed cache-based attacks from web browsers that employ the Prime+Probe technique. Shusterman et al. [44] designed and implemented a cache occupancy side-channel attack through the LLC. Unlike Oren et al. [28], they used the low-timer resolution supported in modern JavaScript engines, which was a countermeasure for cache side-channel attacks. The attack designed by Shusterman et al. [44] measures contention over the whole cache, similar to the covert channel proposed by Maurice et al. [27].

Their attack consists of collecting traces, called *memorygrams* of cache occupancy when the

browser downloads and renders websites; deep neural networks are used to analyze and classify the collected traces. In contrast to Oren et al. [28], Shusterman et al. [44] used one-dimensional memorygrams instead of a two-dimensional array representing multiple individual sets or groups of sets at each point in time inside the cache.

Figure 7 provides examples of memorygrams. As can be seen, each website has its unique signature in the cache. The CBWF attack takes advantage of this phenomenon and creates one-dimensional memorygrams.

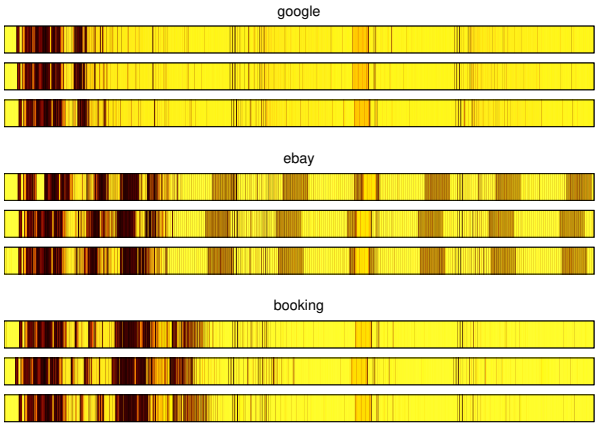


Figure 7: Examples of memorygrams

3.2. Concept Drift in Website Fingerprinting

The challenge of concept drift in the website fingerprinting domain was first recognized by Juarez et al. [18] in 2014. When discussing the limitations of website fingerprinting methods, the authors mentioned the cost associated with updating the machine learning classifier; then, they performed a concept drift experiment with their classifier and found that the accuracy of the classifier dropped sharply to 50% when it was evaluated on data collected 10 days after the date in which their training data was collected. The authors also indicated that maintaining a model to keep it up-to-date is one of the real-world challenges attackers have to overcome in order to successfully perform a website fingerprinting attack.

Rimmer et al. [37] evaluated their proposed deep learning models on data collected over a period of several weeks. Their classifier's performance decreased by 25% over two months due to concept drift.

Aiming to solve the concept drift in website fingerprint problem, Attarian et al. [2] addressed the problem as an online-learning with continuously updating stream data. The authors proposed AdaWFPA, an adaptive online website fingerprinting attack based on adaptive stream mining algorithms. They used the data used by Wang et al. in [56]. Their method used the Adaptive Hoeffding Tree model. Although their classifier achieved a perfect score, the dataset used to evaluate the attack was collected over a short period of time and there was no evidence of any influence by the concept drift. It is thus difficult to compare their results directly to ours. For our study, we created a unique dataset consisting of traces collected over a significant amount of time, allowing concept drift to be properly evaluated.

3.3. Addressing Concept Drift

Various methods have been proposed for dealing with concept drift. In this section, we discuss the methods we have chosen suited to our problem.

3.3.1. Transfer Learning

Transfer learning is a method in which information obtained by training a model in a particular domain is used to solve a problem from a different domain [57]. Transfer learning can be used when there are two similar tasks, and there is a trained model for one of them. The existing model architecture and its weights can be used to train a model that fits the second task, even when there is an insufficient amount of data.

Transfer learning differs from traditional machine learning because it uses a pre-trained model as the springboard for the initiation of a secondary task. Previous studies have shown that the layers and parameters of deep architectures can capture the underlying domain-invariant data representation [24].

An example use of this technique in concept drift problems was presented by García et al. [12]. In their work, they explored how transfer learning can enhance malware detection systems facing concept drift, where the characteristics of malware change over time. Additional researches in the field which use this method to adapt to concept drift include Xie et al. and Wang et al. [59, 53]. Recent research has also demonstrated the effectiveness of integrating transfer learning and concept drift detection methods for wind power forecasting [52].

3.3.2. Incremental Learning

Incremental learning methods are used to train models gradually over time. This type of learning retains the knowledge acquired from old instances to classify new instances. Prior studies developed incremental learning techniques by leveraging linear classifiers, an ensemble of weak classifiers, nearest neighbor classifiers, and similar methods. [4, 34, 21].

The advances in deep learning seen in the last decade have been accompanied by studies on the use of incremental learning in deep neural network models. Rebuffi et al. [35] proposed a method which is used to select a small number of instances from each class. Xiao et al. [58] presented a training method that grows a network hierarchically as new training data are added. Rusu et al. [38] proposed the use of progressive neural networks for reinforcement learning which increases the number of layers in the network to handle new coming data.

One usage of incremental learning in concept drift is shown by Sheu et al. [40]. They focus on spam filtering and used decision trees for classification. The incremental learning model assumed regular updates of their dataset to detect concept drift in new emails. Incremental learning is widely used to detect concept drift by other authors, including Elwell et al. and Ditzler et al. [10, 8, 9].

3.3.3. N-Shot Learning and Triplet Loss Networks

In multi-class classification tasks, deep learning models are often used to assign input feature vectors to one of several predefined categories. These models apply nonlinear transformations to the data via the network architecture and adjust the weights of the network using a cross-entropy loss function to minimize the uncertainty in the predicted probabilities. Despite the ability of the model to discriminate between different classes, feature vectors from the same class may be assigned different embeddings in the output vectors layer. Similarly, feature vectors from different classes may receive similar embeddings that are still distinct enough for accurate classification. The stability and consistency of these embeddings can impact the model's overall accuracy, and regularization techniques and diverse training data can be used to improve the model's performance.

Triplet loss networks [39] offer an alternative method for achieving stability and consistency. This method receives three input vectors: an anchor, positive examples, and negative examples, and its goal is to optimize the network so the anchor and positive examples will obtain similar and different embeddings for the anchor and the negative example. Thus, the output of this network is a vector embedding rather than a classification. This feature vector representation aims to extract features for classes from the same domain, even if the domain was not used to train the classifier. There is no need to retrain the classifier for every new class; instead, a few feature vectors can be collected, and their embeddings can be classified using a simple classifier like k-nearest neighbors.

This network's advantage in concept drift problems stems from the fact that the embeddings of the extracted vectors do not depend on the label or the time at which it was collected. The N-shot learning method was the subject of previous studies on website fingerprinting and demonstrated good results [5]. It was also used for concept drift detection by Yu et al. and other authors [61].

3.4. Adaptive Deep Learning in Adversarial Settings

A similar problem to concept drift is the problem of adversarial attacks on deep learning models. In adversarial settings, especially in the online setting, attackers can observe the model's predictions, and subsequently adapt their attacks to cause the model to misclassify their inputs. For example, spammers can observe the behavior of a spam filter, and change the format their spam messages to bypass the filter [11, 7], and malware creators can change the binary format of their malware to bypass a deep learning-based antivirus program [20]. The methods used by the defender in this case are similar to those used to address concept drift: detect this change in the attacker's behavior, adapt the model to correctly classify this new behavior, and, ultimately, attempt to build a robust model that is not affected by the attacker's changes [63, 33, 26].

4. Methods

4.1. Motivation

Our experiment setup is designed to answer the research questions posed in section 1. Specifically, we aim to understand the effect of concept drift on the accuracy of website fingerprinting attacks, to isolate the effects of website content changes and browser software changes, and to compare the robustness of cache side-channel data and network

side-channel data to this effect. We also aim to evaluate the effectiveness of modern machine learning methods for mitigating the effect of concept drift on website fingerprinting attacks.

To achieve these goals, we first describe our system setup, then describe our dataset collection methodology, and finally describe the methods we used to analyze the dataset and evaluate the effect of concept drift.

4.2. System Setup

Our experimental setup is designed to collect cache and network side-channel traces generated while a user is browsing the web. To collect the data, we set up machines with an Intel Core i5-10500 CPU at 3.1GHz with a 12MB last-level cache, 16GB of RAM, and running the Ubuntu v20.04 operating system. These machines ran scripts which looped over a set of websites, using the Google Chrome browser controlled by the Selenium driver. In our experiment we captured network and cache side-channel trace data simultaneously. To capture network traces, we connected both machines to the Internet through a man-in-the-middle machine, following the methodology of Shusterman et al. [44]; To acquire cache contention data, we opened a JavaScript attack page on the machine under test, in parallel with regular browsing. This attack page, similarly based on Shusterman et al. [44], probes the whole LLC every two milliseconds for a period of 30 seconds, producing a one-dimensional vector of 15,000 values representing cache contention over time.

4.3. Dataset Collection

The goal of our dataset collection was to capture the effect of concept drift on website fingerprinting attacks, and in particular to isolate the effect of website content changes from the effect of browser software changes. To do so, we set up two identical machines using the setup of section 4.2. The version of Chrome on the first machine was fixed at version 92.0. The data collected on this machine is therefore affected only by webpage content changes. The second machine was updated weekly with the latest version of the Google Chrome browser, spanning versions from 92.0 to 99.0 over the course of the experiment. The data collected on this machine simultaneously followed the natural drift in two vectors – content updates and browser version updates. To prevent a case where a single dataset contained traces from multiple browser versions, we checked to see whether a browser update was available only at the beginning of each week. The exact list of versions of the Google Chrome browser used in the experiment is provided in the Appendix.

Over a period of 26 weeks, we used both machines to collect cache and network side-channel traces while browsing 100 popular websites. While choosing a larger number of websites could have been more representative of the diversity of the web, we chose to limit ourselves to 100 websites to we could finish a single iteration of the experiment over the course of a single week, while making sure we collected a reasonable amount of traces for each website during each iteration. The list of websites was chosen from the (now deprecated) Alexa Top-100 listing for May 2021, as listed in the Appendix. Recognizing that the original list comprises numerous variations of similar websites, such as amazon.com and amazon.co.uk, we excluded duplicates and supplemented the dataset with next in popularity URLs to ensure a diverse representation. Each website was loaded 100 times, resulting in a weekly total of 10,000 samples for each machine and each type of attack. At the final week of the experiment, we collected a third dataset which included 10 samples from each of the top 100 Alexa websites, collected using 16 different versions of the Google Chrome browser.

Since the entire experiment was completed in less than a week, we implicitly assumed that the webpage fingerprint does not change significantly during this time. This experiment was designed to highlight the effect of browser updates on accuracy, while limiting the amount of variability in the website content itself. To summarize, we collected three main datasets, where each dataset contains both cache and network traces: in the first, the browser version was kept constant while the website content was updated; in the second, both the browser and the website content were dynamically updated; and in the third, the browser version was updated while the website content was kept constant.

4.4. Machine Learning Models Evaluated

Our evaluation includes the three machine learning models described in section 3.3, as well as two baseline models: a No-Retrain model that is trained once in the beginning of the campaign and does not address concept drift, and a model that is fully retrained from scratch every week. The full retraining strategy should offer the highest accuracy, but

at the cost of requiring the most data to be collected and managed. The No-Retrain model, on the other hand, should offer the lowest accuracy, as it does not address concept drift at all, but it has the advantage of requiring the least data to be collected and managed. The three more advanced models, transfer learning, incremental learning, and N-shot learning, offer a tradeoff between accuracy and data collection requirements. Each of these models is evaluated using two different retraining budgets: one where the model is retrained using 10% of the size of the full dataset (corresponding to 10 traces per label), and one where the model is retrained using 2% of the size of the full dataset, corresponding to 2 traces per label. The models we evaluated are summarized in table 2 below.

Model	**Retraining Budget**
No retraining	0%
Transfer Learning	10% and 2%
Incremental Learning	10% and 2%
N-Shot Learning	10% and 2%
Full retraining	100%

Table 2: Machine learning models evaluated

All of our machine learning models are based on the LSTM neural network architecture proposed by Shusterman et al. [44]. This architecture contains two blocks which perform local feature extraction and dimensionality reduction tasks, followed by a third block which performs the classification.

Each of the first two blocks consists of a convolutional layer and a max-pooling layer; the convolutional layer extracts local features from the input trace, and the max-pooling layer sub-samples the block output to reduce its dimensions. The third block contains an LSTM layer that extracts temporal features from the local features extracted by the first two blocks and a dropout layer that performs regularization on the output to avoid over-fitting. The last layer is a fully-connected layer whose size is based on the number of websites it will classify. During training, the model weights were optimized using the Adam optimizer while minimizing categorical cross-entropy criteria. This architecture was used as-is for the No-Retrain and Full-Retrain models, and was adapted for the transfer learning, incremental learning, and N-shot learning models, as described below. For the *transfer learning* method, The attacker stores a pre-trained model which serves as a base for future attacks. Although the model itself has a basic knowledge of general feature extraction from webpage fingerprinting traces, the patterns it learns for specific webpages are considered outdated, as this model may have been trained long time before the actual attack. In order to execute an effective attack, the attacker collects a relatively small amount of recent fingerprinting traces and uses them to fine-tune the base model. This method was chosen for its low maintenance costs, as the attacker only needs to store a single base model and then fine-tune it using a small amount of traces collected a few days before the attack.

For the *incremental learning* method, as in the previous method, the attacker trains an initial base model. In contrast to the transfer learning method, however, in this method the attacker maintains the model periodically by continuously fine tuning it with small quantities of fresh data. This method provides a ready-to-use model with a smaller budget of fresh traces than the transfer learning method, but it must be constantly maintained. Compared to the transfer learning method, this method requires less fresh data per training session, but more data overall. In addition, as we show below, this method is the most effective in maintaining the model's accuracy over time.

The *N-shot learning* model requires a more advanced approach. Here, the goal is to create a robust embedding that maximizes the KL-divergence between different webpage labels and minimizes it between similar webpage labels. As the embedding approaches this goal, different webpage labels become more *linearly separable*, allowing the embeddings to be discriminated by a simple classifier such as k-nearest neighbors.

In our setting, the data for this method was acquired as follows: To train the embedding model, we kept a ratio of 90 traces for each of 100 webpages. In contrast to the transfer learning and incremental learning settings, however, in this setting we added an additional 30 traces per 100 webpages every two weeks, an addition which made the data more diverse, contributing to the model robustness.

The model training was divided into two parts: First, a model with a similar architecture to previous methods was trained on the first 60 traces * 100 webpages. Next, we replaced the last layer of the LSTM classifier with a dense layer with a linear activation function, and trained it while minimizing the triplet semi-hard loss function on the rest of the data.

During the attack period, we collected additional 2% or 10% of the data, embedded it using the embedding model and trained a KNN model on the data embeddings. In the evaluation part, we embedded the evaluation set traces and predicted their label using the KNN model.

Unlike the previous methods, the N-Shot learning base model operates as an embedding model rather than a classification model. This unique characteristic suggests its potential as a foundational model for identifying even unseen webpages, leveraging minimal examples during the classification phase. While acknowledging this capability, it is important to note that exploring such functionalities falls beyond the scope of this paper.

4.5. Data Preprocessing

Before training a machine learning model, the raw data traces must be preprocessed. The cache contention vector has 15,000 floating-point values ranging from 2 to 8, denoting the time in milliseconds it takes to probe the LLC. We applied standardization on these vectors to optimize the neural network's training sessions.

The size of the raw network traces varied; therefore, we padded the short traces with zeros and limited the longer traces to a constant number of 1,500. To process the network traces, we followed the methodology of Juarez et al. [18], representing every outgoing packet as 1 and every incoming packet to -1. As a final preprocessing step, we standardized each vector.

4.6. Assumptions

Our website fingerprinting setup follows several assumptions commonly held in similar works in the field [18].

Closed-World. Website fingerprinting is commonly evaluated in two scenarios. The first one is the *closed-world* scenario, and the second and more general is the *open-world* scenario. In the main part of this research, we evaluated our attacks on the closed world scenario, which will allow us to analyze and compare the performance of classification approaches. In this scenario, the victim only browses a pre-known list of sites. As a result, the attacker can train a model on data collected when browsing those sites. Equipped with the knowledge that in the future, the victim would only visit sites the model was trained on, the model is tasked with predicting the site the user visited in our evaluation. This contrasts with the open-world scenario, in which there is a possibility that the victim may visit an unknown, unmonitored site. In this setting, the model must have another possible classification value representing an unknown website, which is not in the list of monitored websites. In Section 5.5.1 we perform a limited evaluation of the open-world scenario, and show that it is similarly affected by concept drift, and that the methods we propose for handling concept drift are also effective in this scenario.

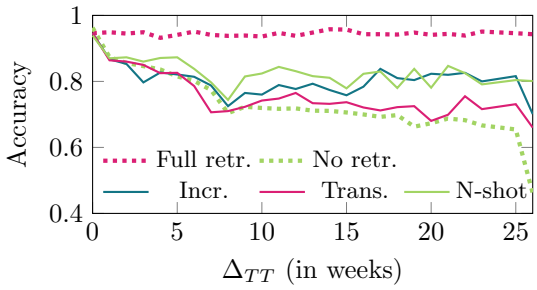
One Tab. Like other website fingerprinting studies, our attack model assumes that the victim browses just one webpage at any given time. The only other open tab is the page containing the malicious JavaScript code. That is, two tabs are open on the victim's machine, one of which measures the cached content and sends it to the attacker. We do not, however, assume that both tabs are open on the same browser; in fact, the victim may be using two different browsers.

No Background Traffic. Our final assumption, widely adopted in the literature, is not trivial – we assume that the attacker has the ability to isolate the network traffic associated with the connection to the server from traffic associated with other applications on the target machine. This is a complex assumption, since it is possible to open a connection over a Tor or VPN connection to several services simultaneously, resulting in a scenario where a man-in-the-middle attacker cannot assign every message passing through the network to a particular connection.

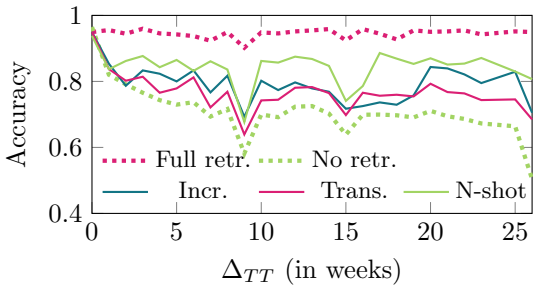
5. Results

In this section we assess how well the machine learning model performs under various maintenance methods. Each maintenance approach is allocated a budget, which is used to ensure the model stays fresh. This budget represents a percentage of the full-size database required to train a new model from scratch. It is important to note that the budget allocated to model maintenance relates to each weekly fine-tuning step, and not to the total number of traces used throughout the campaign's history. Here is a breakdown of the budget indicators:

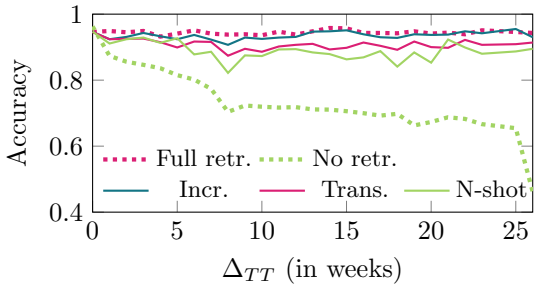
- For the full-retraining method, the budget is set at 100% of the training set.



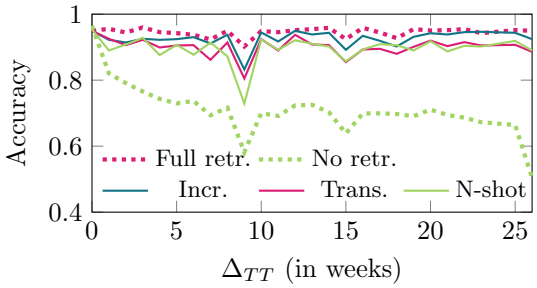
(a) Cache contention-based classifier using 2% of fresh data



(b) Network-based classifier using 2% of fresh data



(c) Cache contention-based classifier using 10% of fresh data



(d) Network-based classifier using 10% of fresh data

Figure 8: Accuracy comparison – Dynamic browser version and Dynamic web content

- In the incremental learning method, the budget is refreshed weekly, as this method requires weekly fine-tuning of the model.
- Transfer and N-shot learning methods allocate the budget for a single fine-tuning session, conducted in the same week as the evaluation on the test dataset, and consume no data in all other weeks.
- The no-retrain method operates with a budget of 0%.

When comparing different methods, we must consider the trade-off between the effort required to collect fresh data and their resulting performance.

5.1. Evaluation Metrics

When comparing methods for model maintenance, the accuracies of a multiple testing experiments need to be combined into a single indicator that represents the ability of the method to mitigate concept drift. As an initial step, we apply the Kolmogorov-Smirnov test, which is used to compare continuous distributions of two samples or to test whether a sample comes from a particular distribution. Using this test, we will check whether the Static and the Dynamic browser versions come from different distributions. Next, to compare between multiple maintenance methods, we decided to use the slope of a linear regression curve fit. A large negative number for the slope indicates a stronger trend of accuracy degradation over time. Using this method allows us to see beyond individual accuracy values, allowing us to compare different methods with initially-different accuracy levels.

5.2. Dynamic Browser Version, Dynamic Web Content

This experiment captures the default behavior of concept drift by measuring the impact of both noise vectors: the dynamic web content and the constantly-changing browser version. Due to the combination of these two factors, we hypothesized that this setting will be the hardest for all maintenance methods. Figures 8a, 8b, 8c and 8d show CBWF and NBWF classifier performance when comparing different maintenance methods and settings. In the figures, the x-axis indicates the week in which the test set was acquired, and the y-axis

	Cache					Network				
	No retr.	N-shot	Transfer	Incr.	Full retr.	No retr.	N-shot	Transfer	Incr.	Full retr.
Static Browser Version, 10% training budget	-0.625	-0.089	-0.014	**0.151**	0.023	-0.984	-0.017	-0.092	**0.107**	0.056
Static Browser Version, 2% training budget	-0.625	-0.110	-0.490	**-0.047**	0.023	-0.984	-0.241	-0.482	**-0.207**	0.056
Dynamic Browser Version, 10% training budget	-1.091	-0.140	-0.055	**0.049**	-0.006	-0.690	0.004	-0.054	**0.058**	0.018
Dynamic Browser Version, 2% training budget	-1.091	-0.295	-0.672	**-0.228**	0.006	-0.690	**-0.087**	-0.350	-0.272	0.018

Table 3: Slopes of the regression lines derived from the models' classification accuracy for the methods examined

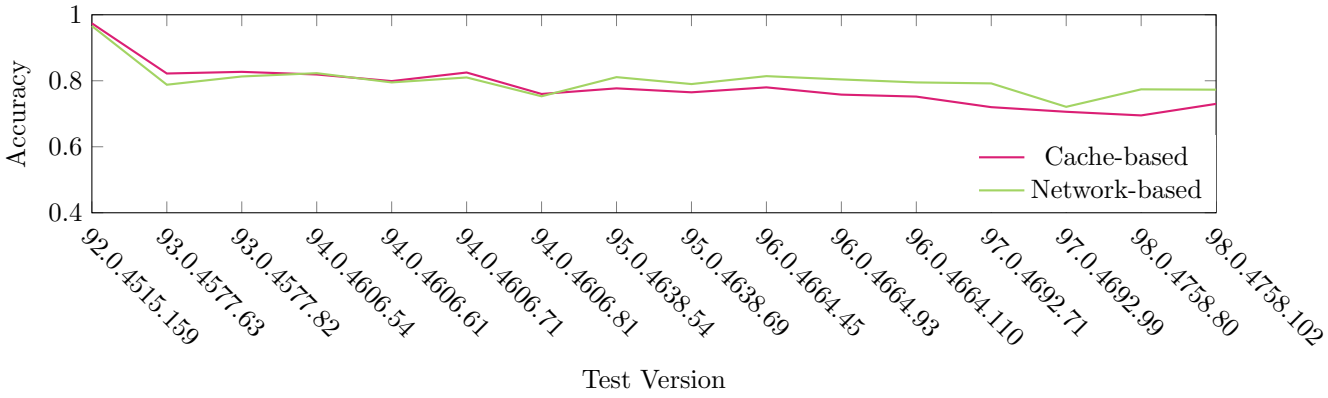


Figure 9: Effect of Google Chrome version drift on classification accuracy (trained on version 92.0)

presents the classifier's accuracy on the test set. We also report on the significance of these results using a Kolmogorov-Smirnov one sided test with significance level of 0.01, When training and testing data were both captured during the same week, over 95% accuracy was observed for both the network and cache contention classifiers. Over the 26-week period, however, it decreases up to 60%. In a setting with a 10% training budget, we observe that all maintenance methods perform well, with slight differences in accuracy. Table Section 5 shows that in this setting, the incremental and transfer learning methods on CBWF model have slopes of 0.049 and -0.055, which are better than the N-shot learning method which obtains a slope of -0.140, with all differences being statistically significant. For the NBWF model, the the incremental learning method is significantly better than the other two, while the difference between the N-shot and transfer learning models is not significant. When moving to the 2% training budget settings, for the CBWF model the N-shot and incremental suffer a minor degradation, to slopes of -0.295 and -0.228, respectively, which are statistically indistinguishable. The transfer learning method, however, degrades to a slope of -0.672, which is significantly worse than the other two methods. In the NBWF model with a 2% training budget, the N-shot learning method is significantly better than the other two, while the difference between the incremental and transfer learning methods is not significant.

5.3. Dynamic Browser Version, Static Web Content

In this experiment, we measured how a model trained on data from an early browser version performed when presented with traces collected from newer browser versions. To minimize the effect of changes to the content of the webpages themselves on accuracy, we ran this experiment over a short time period. This required us to collect a smaller portion of data for every test version. The entire experiment finished in less than two weeks, during which we were able to collect 10 traces for each of

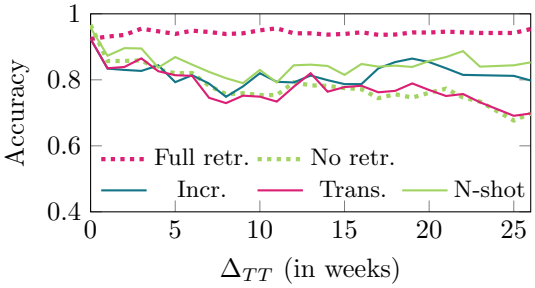


Figure 10: Cache-based, 2% fresh data

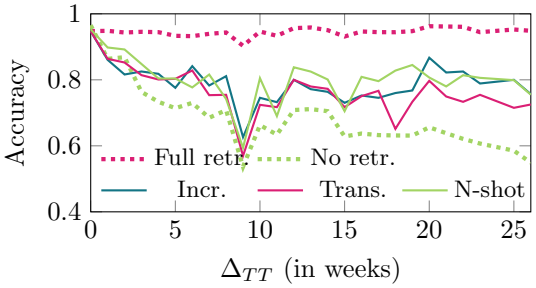


Figure 11: Network-based, 2% fresh data

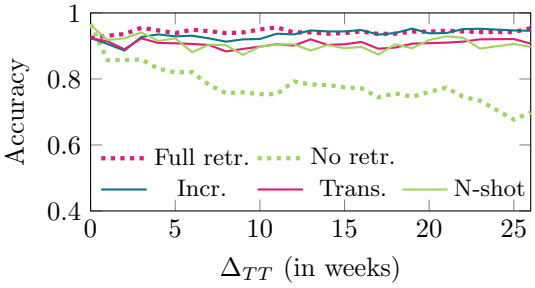


Figure 12: Cache-based, 10% fresh data

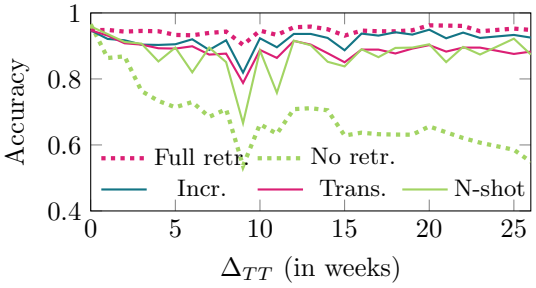


Figure 13: Network-based, 10% fresh data

Figure 14: Accuracy comparison – Static browser version and Dynamic web content

the 100 webpages for each browser version we evaluated. Our experimental data captured 16 different Chrome versions, including 6 major version updates (e.g., from 92.0 to 98.0) and an additional 10 minor version updates. As in previous experiments, this experiment was conducted on both CBWF and NBWF models, and their data was collected simultaneously on the same machine. Figure 9 shows the effect on accuracy of a model trained on the prior version (92.0) of Google Chrome data while predicting on traces collected on 15 later versions. The results in the figure show that in both models there is a sharp accuracy drop after changing the 1st version – from 92.0.4515.159 to 93.0.4577.63, after which the accuracy degradation becomes more moderate over versions. After the 1st major version change, which reduced both models accuracy by 15%, additional updates had a smaller, but still noticeable, marginal effect on accuracy. This was observed both for major and for minor version updates. In conclusion, the browser version changes have a significant effect on the accuracy of the website fingerprinting models, but we note that the effect is not as severe as the effect of changes in the content of the webpages themselves.

To further investigate the effect of browser version changes on the accuracy of the website fingerprinting models, we analyzed the results of the two previous experiments, and conducted a Kolmogorov-Smirnov statistical test on the accuracy results of the dynamic content–static version and the dynamic content–dynamic version experiments. For the CBWF experiment, the test concluded that browser version change significantly contributes to concept drift (p=0.98e-6). For the NBWF experiment, in contrast, we cannot make this claim, due to a p-value of 0.021.

5.4. Static Browser Version, Dynamic Web Content

In the static browser experiment we measured the base classifier's performance degradation over 26 weeks, where the parameter that varies is the website content. Figures 10, 11, 12 and 13 present the static browser version experiment results for the cache contention-based and network-based classifiers. In general, we see that the different maintenance methods show similar behaviors to the ones observed in the previous experiment which models dynamic browser versions.

5.5. Additional Evaluations

5.5.1. Open World Scenario

In addition to the closed-world scenario, we also performed a limited evaluation of the model in an open-world scenario. In this scenario, the model is trained on a set of sensitive webpages, but is also expected to detect non-sensitive webpages that were not part of the training set. We simulated this scenario using 70 sensitive webpages and 30 non-sensitive webpages, each having 100 traces. We evaluated this use-case in both network and cache domains. The model structure remained the same, except for the output softmax layer, which now has a size of 70. While predicting the classes, we set a threshold that determined whether the model was confident enough to recognize a website. Below-threshold confidence values over all websites were interpreted as a Non-Sensitive label. In this scenario, we compared the No-Retrain method to the Transfer learning method. We chose to evaluate this scenario with transfer learning, because it is well-suited for scenarios where the attacker has limited opportunities to collect fresh data monthly from the target environment. This approach significantly reduces the data collection burden while still enabling the model to adjust to recent changes in website content. We also experimented on static browser versions in both cache and network domains, to isolate the temporal drift factor from the browser drift factor. Here we compared the F2-Score of the No-Retrain use case with the Transfer-Learning method. The F2-Score prioritizes recall, making it more sensitive to false negatives, We used this score because we prioritize finding the sensitive labels, rather than predicting non-sensitive ones. In the figures 15 and 16, we can see that in both Cache and Network models, the Transfer-Learning method has a small but significant accuracy advantage of around 3%, compared to the No-Retrain method. This shows that our transfer learning method is effective in adapting to changes in the target environment, even in the open-world scenario.

5.5.2. Countermeasures

Several works have proposed countermeasures to mitigate the effectiveness of website fingerprinting attacks [19, 55, 3, 22]. The common approach of many of these countermeasures is to *add noise* to the traces collected by the attacker in the online phase of the attack, either by perturbing the network, or by creating artificial traffic on the network. Since this noise is not available to the attacker at training time, this approach effectively reduces the accuracy of the attack. While there is an existing body of work on countermeasures which are effective against website fingerprinting attacks assuming $\Delta_{TT} = 0$, no previous works considered the combined effect of concept drift and countermeasures. To evaluate the impact of concept drift on the effectiveness of countermeasures, we simulated two types of countermeasures: a cache-based countermeasure and a network-based countermeasure.

Cache Countermeasure: We simulated the cache countermeasure 17 as Gaussian noise added to cache evaluation traces, while the model was trained on clean data. To evaluate the impact of this countermeasure on static-browser cache data, Gaussian noise with a *mean* of 0 was added, and three settings of *standard deviation* (std) were tested: 0.1, 0.2, and 0.3.

In the No-Retrain case, when applying the countermeasure, we observed an accuracy drop between 10% and 50% as the standard deviation increased from 0.1 to 0.3. The impact of concept drift itself remained a moderate decline in accuracy, whereas the countermeasure was responsible for the most significant accuracy drop.

Network Countermeasure: We simulated the network countermeasure by randomly adding packets to the network trace. In this experiment, we considered two noise parameters: the probability of adding a packet and the standard deviation (std) of the added packet size. For simplicity, we increased both the added packet probability (P) and the size std together to amplify the noise effect. Figure 18 shows that when std is 300 and P is 0.1, the degradation in accuracy is minimal. However, when these parameters are increased to std=600 and P=0.3, the accuracy drops by 50%.

Having established a baseline for the No-Retrain method, we compared the performance of this approach with the transfer learning method, assuming an attacker budget of 10% of the traces collected weekly and a noise power of 0.1. As Figure 18 shows, transfer learning significantly slows down the accuracy degradation caused by the noise countermeasure, when combined with concept drift. When, however, the noise power is increased to 0.2, we observed that the transfer learning method does not noticeably improve accuracy, and that the accuracy drops to a level similar to that of the No-Retrain method. This can be intuitively explained by the fact that the perturbation introduced by the

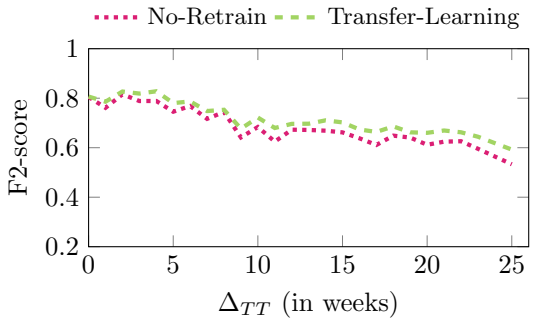


Figure 15: Cache-based classifier Open-World, confidence threshold set to 0.35

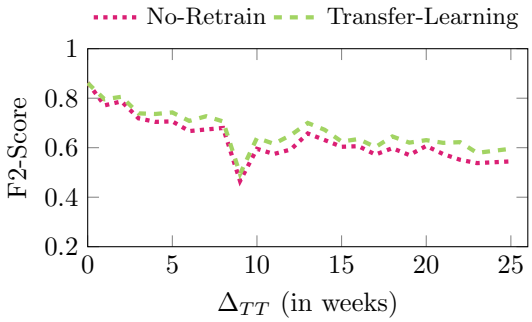


Figure 16: Network-based Open-World, confidence threshold set to 0.35

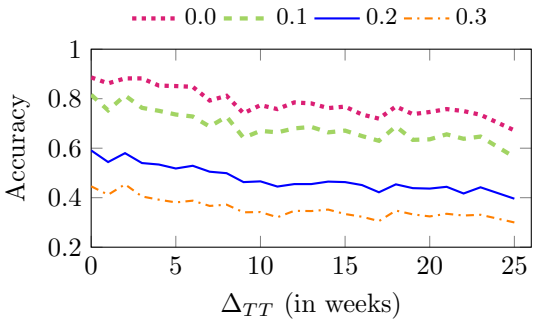


Figure 17: Cache-based No-Retrain classifier coutermeasure

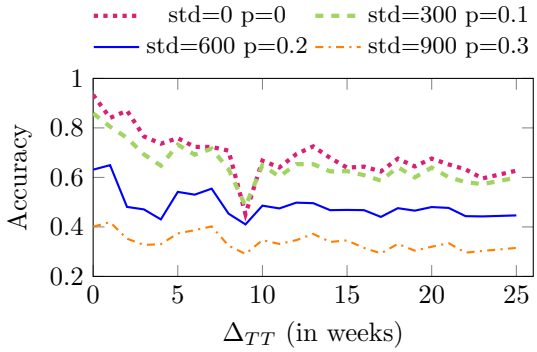


Figure 18: Network-based No-Retrain classifier countermeasure

countermeasure at such a high level of noise overwhelms the accuracy gain offered by transfer learning.

6. Discussion

In this study we investigated the use of different methods for dealing with concept drift in the context of website fingerprinting attacks. We examined three ways of dealing with concept drift: transfer learning, incremental learning and N-shot learning, with each method requiring the collection of new instances every few weeks. We analyzed each method's performance over time, using a dataset collected over 26 weeks that included drift. Our dataset will be available at `https://orenlab.sise.bgu.ac.il/p/ConceptDrift`, along with the scripts for data collection, models training, experiments code with examples. We found that by collecting a relatively small amount of fresh data, it is possible to maintain the model's stability over time.

Our results indicate that incremental learning is the superior method. We note, however, that incremental learning learns with the help of the many examples collected over the 26-week period, while the other methods rely on a significantly smaller amount of traces. In cases in which weekly model maintenance is an issue, or if there is a limited trace budget, the best method is N-shot learning. Transfer learning, on the other hand, was not shown to outperform the other two methods in these respects.

The findings of our research, in which we tackled one of the challenges associated with website fingerprinting attacks, pave the way toward the successful implementation of such attacks in the real world.

Surprisingly, no sharp decay in accuracy was observed due to changes in the version of the Chrome browser. One of our hypotheses at the beginning of the study was that the accuracy would decrease significantly as time went on. While this did happen, it occurred at a more gentle pace than we expected. This suggest that the updates to the Chrome browser during our evaluation period did

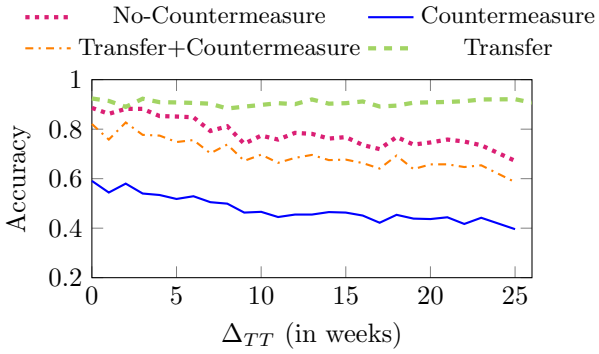


Figure 19: Cache-based No-Retrain VS Transfer learning classifier coutermeasure

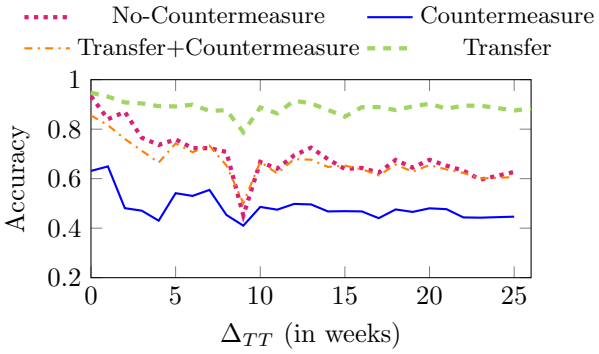


Figure 20: Network-based No-Retrain VS Transfer learning classifier coutermeasure

not have a major impact on the way the Chrome browser's engine uses RAM as it renders the websites. Since we only experimented on the Chrome browser, and only for a fixed period, we cannot determine whether this behavior generalizes to other browsers.

6.1. Limitations

While our results suggest that long-term fingerprinting campaigns can be designed to deal with concept drift, there are some limitations with the methods we presented in this study. One main limitation is the need for continuous data collection. Although we aimed to develop methods that significantly reduce the amount of data that must be collected, in order to maintain a database that results in high accuracy, the attacker must constantly collect data and retrain the machine learning model. Collecting data continuously every few weeks may not always be possible. For example, in the case of NBWF attacks, there may be attackers who intercept the victim's traffic using a man-in-the-middle attack for a limited time (for example, because they have one-time physical access to the network of the attacked computer). In these cases, the attacker cannot collect data continuously as we assumed in this study. The attack will be feasible, however, fi the attacker has persistent access to the victim's machine, such as in the case of a malware infection, or if the attacker has control over the victim's network, such as in the case of an overly-curious employer. This limitation is added to the non-trivial list of criticisms aimed at the general premise of website fingerprinting attacks, as identified by Juarez et al. [18], including assumptions on the user, the browser, and the network.

6.2. Generalizability to Additional Websites

Our study focused on a limited set of 100 websites, which may not fully represent the diversity of real-world web traffic. Future work, especially when considering the open-world setting, should consider expanding the dataset to include a broader range of websites, such as the top-1k Alexa sites, to better understand the generalizability of our findings across different browsing contexts. Larger datasets typically require the use of more advanced machine learning models which are better able to capture the differences between websites. Intuitively, these models should be even more sensitive to concept drift.

6.3. Cache vs Network

In this research we examined the differences between CBWF and NBWF attacks in the context of concept drift. Our evaluation revealed that both of them are influenced by web content drift. In contrast, we were only able to show significant influence of browser version changes on accuracy in the CBWF setting. One possible explanation for this was the fact that cache-based observations are more closely tied to the browser's code execution patterns. Another explanation is simply that there was too much noise present in the network-based observations, which made the effect of browser version changes difficult to detect. In general, while attacks on the network were more accurate, the general behavior was similar to that seen in attacks on the cache. Therefore, the attacker can perform either CBWF or NBWF. Since each method has its advantages and disadvantages, the attacker can chose whichever method they prefer. If attacker can regularly inject JavaScript code into a browser

on the machine (e.g., through advertisements), they can perform a CBWF attack, and if the attacker is on the same network as the victim, they can perform an NBWF attack.

6.4. Browser Updates

One of our hypotheses was related to browser updates. We expected that the Google Chrome browser updates (both major and minor) would influence the concept drift due to small changes in the browser's engine. This hypothesis was found to be significant only for CBWF. In NBWF, the drift in the static website content experiment was not significantly greater than the drift that happened over time.

6.5. Advantages and Disadvantages of Different Maintenance Methods

Previous works suggested dealing with concept drift in website fingerprinting by periodically retraining the model from scratch. Our work introduces three more advanced methods for maintaining the model's accuracy over time: transfer learning, incremental learning, and N-shot learning. All of these methods use a dramatically smaller amount of fresh data than the full retraining method, at the price of some decrease in accuracy. Each method has its own strengths and weaknesses that have to be considered when choosing the most method suited for a particular attack scenario.

The *incremental* method achieves higher accuracies in most configurations, but it requires the attacker to collect data and fine-tune the model every week. This model is therefore best suited for attackers who can maintain a constant data collection process, such as those who have access to the victim's machine or network for an extended period of time.

The *transfer* method is more flexible, as it allows the attacker to fine-tune the model with a single set of fresh data, without the need for constant data collection. This flexibility, however, comes at the cost of lower accuracy than the incremental method. This model is therefore best suited for attackers who can only collect a single set of fresh data, but do not have the ability to maintain constant data collection access.

Finally, the *N-shot* method is the most robust to small amounts of fresh data, allowing the attacker to maintain a high level of accuracy even with a very small training budget. This suggests that the N-shot method forms a better generalization of the website classification task than the other methods, allowing it to maintain accuracy even as the update data gets smaller. The N-shot method is therefore best suited for attackers who can only collect a small amount of fresh data.

The comparative benefits of each method are summarized in Table 4.

6.6. Future Work

This work, which is the first to investigate the effect of concept drift on website fingerprinting attacks, opens up several avenues for future research.

Follow-up research could examine the impact of concept drift on *additional browsers*, such as Firefox or Tor. In particular, it would be interesting to see how the effect of periodic updates to the "Gecko" renderer used by Firefox and Tor, or to the Webkit engine used by Safari, is similar to the one we observed in Chrome.

It would also be interesting to observe the impact of concept drift on *additional web-based attacks*, including traffic characterization and application classification [45, 51, 48]. We hypothesize that these types of data are less subject to change than constantly-evolving webpages, and that the impact of concept drift on these attacks would be therefore less severe.

A final future research direction could focus on *countermeasures*. As we showed in this study, traditional noise-based countermeasures, when properly applied, can cause a significant drop in the accuracy of the attacker, even if the attacker attempts to use transfer learning or incremental learning to mitigate concept drift. Indeed, a countermeasure that degrades the attacker's accuracy in the complete absence of concept drift is just as effective when concept drift is added, since concept drift appears to the classifier as a form of additional noise. These countermeasures, however, are not always practical, since they cause significant degradation to the user experience [3]. It would be interesting, therefore, to consider lightweight countermeasures which are only effective for a non-zero Δ_{TT}, explicitly leveraging concept drift to confuse the adversary. For example, if a website periodically changes the order of the elements on each page, while making sure the rendered output is semantically and visually identical to the original, this may mislead an adversary trained on the original order of elements, while effectively adding no additional burden to the user or the network. Similar transformations may also

Method	Accuracy	Data Collection	Use Case
Full Retraining	Optimal	100% weekly	Full network access
Incremental Learning	Best	10% weekly	Full network access, limited trace budget
Transfer Learning	Better	10% once	Limited network access, limited trace budget
N-shot Learning	Good	2% once	Limited network access, extremely limited trace budget

Table 4: When to Use Each Maintenance Method

be applied to images, scripts, remote resources and other elements of the website. This form of countermeasure, which assumes the existence of concept drift as part of its underlying design, is an interesting topic for future work.

7. Conclusion

In this study, we examined the effect of concept drift on the accuracy of website fingerprinting attacks. We presented three methods for dealing with this drift and maintaining the model's accuracy over time: transfer learning, incremental learning, and N-shot learning. Our results show that incremental learning is the best method for maintaining the model's accuracy over time if the attacker has the opportunity to constantly maintain the model. If fine-tuning can be performed only once, the attacker will benefit more from the n-shot learning method. We also found that the Chrome browser updates significantly influence the model's accuracy for cache-based fingerprinting, while we could not demonstrate the same effect for network-based fingerprinting. Our results show that it is possible to maintain the model's accuracy over time with a relatively small amount of fresh data, providing evidence that long-term website fingerprinting campaigns are feasible, despite the effect of concept drift. To facilitate further research in this area, we make available the dataset used in this study, as well as the scripts for data collection, model training, and experiments.

Appendix A. Supplementary Data

Version	**Release Date**
92.0.4515.159	2021-08-16
93.0.4577.63	2021-08-31
93.0.4577.82	2021-09-13
94.0.4606.54	2021-09-21
94.0.4606.61	2021-09-24
94.0.4606.71	2021-09-30
94.0.4606.81	2021-10-07
95.0.4638.54	2021-10-19
95.0.4638.69	2021-10-28
96.0.4664.45	2021-11-15
96.0.4664.93	2021-12-06
96.0.4664.110	2021-12-13
97.0.4692.71	2022-01-04
97.0.4692.99	2022-01-19
98.0.4758.80	2022-02-01
98.0.4758.102	2022-02-14

Table A.5: Chrome Versions Under Study

google.com	imgur.com	twitch.tv
thestartmagazine.com	youtube.com	xhamster.com
whatsapp.com	amazonaws.com	facebook.com
wordpress.com	pornhub.com	tianya.cn
baidu.com	apple.com	yahoo.co.jp
xnxx.com	wikipedia.org	booking.com
csdn.net	rakuten.co.jp	qq.com
xinhuanet.com	alipay.com	chase.com
taobao.com	adobe.com	naver.com
spotify.com	yahoo.com	pinterest.com
microsoft.com	espn.com	tmall.com
dropbox.com	livejasmin.com	craigslist.org
amazon.com	bongacams.com	aliexpress.com
zhihu.com	twitter.com	babytree.com
bing.com	soundcloud.com	sohu.com
tumblr.com	ebay.com	discordapp.com
live.com	so.com	github.com
jianshu.com	jd.com	cnblogs.com
tribunnews.com	nicovideo.jp	vk.com
quora.com	stackoverflow.com	daum.net
instagram.com	bbc.com	okezone.com
medium.com	sina.com.cn	force.com
office.com	nytimes.com	weibo.com
salesforce.com	xvideos.com	cricbuzz.com
reddit.com	pixnet.net	msn.com
1688.com	360.cn	ettoday.net
paypal.com	stackexchange.com	yandex.ru
cnn.com	bilibili.com	popads.net
linkedin.com	onlinesbi.com	hao123.com
nih.gov	blogspot.com	roblox.com
imdb.com	mediafire.com	netflix.com
aparat.co	t.co	globo.com
fandom.co	indeed.com	sogou.com
researchgate.net		

Table A.6: Top 100 Alexa sites, as of May 2021

References

[1] Onur Aciiçmez, Billy Bob Brumley, and Philipp Grabher. New results on instruction cache attacks. In Stefan Mangard and François-Xavier Standaert, editors, *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, volume 6225 of *Lecture Notes in Computer Science*, pages 110–124. Springer, 2010. doi: 10.1007/978-3-642-15031-9_8. URL https://doi.org/10.1007/978-3-642-15031-9_8. 5

[2] Reyhane Attarian, Lida Abdi, and Sattar Hashemi. Adawfpa: Adaptive online website fingerprinting attack for tor anonymous network: A stream-wise paradigm. *Comput. Commun.*, 148:74–85, 2019. doi: 10.1016/j.comcom.2019.09.008. URL https://doi.org/10.1016/j.comcom.2019.09.008. 7

[3] Tapan Basak, Reza Curtmola, Yossi Oren, and Hai Phan. Flipstress: Noise injection defenses against cpu-cache-based web attacks. In *21st EAI International Conference on Security and Privacy in Communication Networks (SecureComm)*, Xiangtan, China, July 2025. EAI. 15, 18

[4] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. *Advances in neural information processing systems*, 13, 2000. 8

[5] Mantun Chen, Yongjun Wang, Hongzuo Xu, and Xiatian Zhu. Few-shot website fingerprinting attack. *Comput. Networks*, 198:108298, 2021. doi: 10.1016/j.comnet.2021.108298. URL https://doi.org/10.1016/j.comnet.2021.108298. 8

[6] Shane S. Clark, Hossen Asiful Mustafa, Benjamin Ransford, Jacob Sorber, Kevin Fu, and Wenyuan Xu. Current events: Identifying webpages by tapping the electrical outlet. In *ESORICS*, volume 8134 of *Lecture Notes in Computer Science*, pages 700–717. Springer, 2013. 4

[7] Sarah Jane Delany, Padraig Cunningham, Alexey Tsymbal, and Lorcan Coyle. A case-based technique for tracking concept drift in spam filtering. *Knowl. Based Syst.*, 18(4-5):187–195, 2005. 8

[8] Gregory Ditzler and Robi Polikar. Incremental learning of concept drift from streaming imbalanced data. *IEEE Trans. Knowl. Data Eng.*, 25(10):2283–2301, 2013. doi: 10.1109/TKDE.2012.136. URL https://doi.org/10.1109/TKDE.2012.136. 6, 8

[9] Ryan Elwell and Robi Polikar. Incremental learning of variable rate concept drift. In Jón Atli Benediktsson, Josef Kittler, and Fabio Roli, editors, *Multiple Classifier Systems, 8th International Workshop, MCS 2009, Reykjavik, Iceland, June 10-12, 2009. Proceedings*, volume 5519 of *Lecture Notes in Computer Science*, pages 142–151. Springer, 2009. doi: 10.1007/978-3-642-02326-2_15. URL https://doi.org/10.1007/978-3-642-02326-2_15. 6, 8

[10] Ryan Elwell and Robi Polikar. Incremental learning of concept drift in nonstationary environments. *IEEE Trans. Neural Networks*, 22(10):1517–1531, 2011. doi: 10.1109/TNN.2011.2160459. URL https://doi.org/10.1109/TNN.2011.2160459. 6, 8

[11] João Gama, Indre Zliobaite, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46 (4):44:1–44:37, 2014. doi: 10.1145/2523813. URL https://doi.org/10.1145/2523813. 8

[12] David Escudero García, Noemí DeCastro-García, and Ángel Luis Muñoz Castañeda. An effectiveness analysis of transfer learning for the concept drift problem in malware detection. *Expert Syst. Appl.*, 212: 118724, 2023. doi: 10.1016/J.ESWA.2022.118724. URL https://doi.org/10.1016/j.eswa.2022.118724. 6, 7

[13] Qian Ge, Yuval Yarom, David Cock, and Gernot Heiser. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. *J. Cryptogr. Eng.*, 8(1):1–27, 2018. doi: 10.1007/s13389-016-0141-6. URL https://doi.org/10.1007/s13389-016-0141-6. 5

[14] Mark D. Hill and Alan Jay Smith. Evaluating associativity in CPU caches. *IEEE Trans. Computers*, 38 (12):1612–1630, 1989. doi: 10.1109/12.40842. URL https://doi.org/10.1109/12.40842. 4

[15] Andrew Hintz. Fingerprinting websites using traffic analysis. In *International workshop on privacy enhancing technologies*, pages 171–178. Springer, 2002. 2, 3, 4

[16] Wei-Ming Hu. Lattice scheduling and covert channels. In *1992 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, USA, May 4-6, 1992*, pages 52–61. IEEE Computer Society, 1992. doi: 10.1109/RISP.1992.213271. URL https://doi.org/10.1109/RISP.1992.213271. 5

[17] Christian Huitema and Eric Rescorla. Issues and Requirements for Server Name Identification (SNI) Encryption in TLS. RFC 8744, July 2020. URL https://www.rfc-editor.org/info/rfc8744. 1

[18] Marc Juárez, Sadia Afroz, Gunes Acar, Claudia Díaz, and Rachel Greenstadt. A critical evaluation of website fingerprinting attacks. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA, November 3-7, 2014*, pages 263–274. ACM, 2014. doi: 10.1145/2660267.2660368. URL https://doi.org/10.1145/2660267.2660368. 2, 6, 7, 11, 17

[19] Marc Juárez, Mohsen Imani, Mike Perry, Claudia Díaz, and Matthew Wright. Toward an efficient website fingerprinting defense. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26-30, 2016, Proceedings, Part I*, volume 9878 of *Lecture Notes in Computer Science*, pages 27–46. Springer, 2016. doi: 10.1007/978-3-319-45744-4_2. URL https://doi.org/10.1007/978-3-319-45744-4_2. 6, 15

[20] Bojan Kolosnjaji, Ambra Demontis, Battista Biggio, Davide Maiorca, Giorgio Giacinto, Claudia Eckert, and Fabio Roli. Adversarial malware binaries: Evading deep

learning for malware detection in executables. In *EUSIPCO*, pages 533–537. IEEE, 2018. 8

[21] Ilja Kuzborskij, Francesco Orabona, and Barbara Caputo. From n to n+ 1: Multiclass transfer incremental learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3358–3365, 2013. 8

[22] Haipeng Li, Nan Niu, and Boyang Wang. Cache Shaping: An Effective Defense Against Cache-Based Website Fingerprinting. In *CODASPY 2022 - Proceedings of the 12th ACM Conference on Data and Application Security and Privacy*, 2022. doi: 10.1145/3508398.3511500. 15

[23] Fangfei Liu, Yuval Yarom, Qian Ge, Gernot Heiser, and Ruby B. Lee. Last-level cache side-channel attacks are practical. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 605–622. IEEE Computer Society, 2015. doi: 10.1109/SP.2015.43. URL `https://doi.org/10.1109/SP.2015.43`. 5, 6

[24] Mingsheng Long, Jianmin Wang, Yue Cao, Jia-Guang Sun, and Philip S. Yu. Deep learning of transferable representation for scalable domain adaptation. *IEEE Trans. Knowl. Data Eng.*, 28(8):2027–2040, 2016. doi: 10.1109/TKDE.2016.2554549. URL `https://doi.org/10.1109/TKDE.2016.2554549`. 7

[25] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, João Gama, and Guangquan Zhang. Learning under concept drift: A review. *CoRR*, abs/2004.05785, 2020. URL `https://arxiv.org/abs/2004.05785`. 3, 6

[26] Xingjun Ma, Yuhao Niu, Lin Gu, Yisen Wang, Yitian Zhao, James Bailey, and Feng Lu. Understanding adversarial attacks on deep learning based medical image analysis systems. *Pattern Recognit.*, 110:107332, 2021. 8

[27] Clémentine Maurice, Christoph Neumann, Olivier Heen, and Aurélien Francillon. C5: cross-cores cache covert channel. In Magnus Almgren, Vincenzo Gulisano, and Federico Maggi, editors, *Detection of Intrusions and Malware, and Vulnerability Assessment - 12th International Conference, DIMVA 2015, Milan, Italy, July 9-10, 2015, Proceedings*, volume 9148 of *Lecture Notes in Computer Science*, pages 46–64. Springer, 2015. doi: 10.1007/978-3-319-20550-2_3. URL `https://doi.org/10.1007/978-3-319-20550-2_3`. 5, 6

[28] Yossef Oren, Vasileios P. Kemerlis, Simha Sethumadhavan, and Angelos D. Keromytis. The spy in the sandbox: Practical cache attacks in javascript and their implications. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 1406–1418. ACM, 2015. doi: 10.1145/2810103.2813708. URL `https://doi.org/10.1145/2810103.2813708`. 5, 6, 7

[29] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: the case of AES. *IACR Cryptol. ePrint Arch.*, 2005:271, 2005. URL `http://eprint.iacr.org/2005/271`. 5

[30] Venkata N Padmanabhan and Lakshminarayanan Subramanian. An investigation of geographic mapping techniques for internet hosts. In *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 173–185, 2001. 1

[31] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In Yan Chen and Jaideep Vaidya, editors, *Proceedings of the 10th annual ACM workshop on Privacy in the electronic society, WPES 2011, Chicago, IL, USA, October 17, 2011*, pages 103–114. ACM, 2011. doi: 10.1145/2046556.2046570. URL `https://doi.org/10.1145/2046556.2046570`. 6

[32] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. Website fingerprinting at internet scale. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016. URL `http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2017/09/website-fingerprinting-internet-scale.pdf`. 6

[33] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy*, pages 582–597. IEEE Computer Society, 2016. 8

[34] Robi Polikar, Lalita Upda, Satish S Upda, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE transactions on systems, man, and cybernetics, part C (applications and reviews)*, 31(4):497–508, 2001. 8

[35] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5533–5542. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.587. URL `https://doi.org/10.1109/CVPR.2017.587`. 8

[36] Vera Rimmer, Davy Preuveneers, Marc Juárez, Tom van Goethem, and Wouter Joosen. Automated feature extraction for website fingerprinting through deep learning. *CoRR*, abs/1708.06376, 2017. URL `http://arxiv.org/abs/1708.06376`. 6

[37] Vera Rimmer, Davy Preuveneers, Marc Juárez, Tom van Goethem, and Wouter Joosen. Automated website fingerprinting through deep learning. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018. URL `http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_03A-1_Rimmer_paper.pdf`. 2, 4, 6, 7

[38] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016. URL `http://arxiv.org/abs/1606.04671`. 8

[39] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 815–823. IEEE Computer Society, 2015. doi: 10.1109/CVPR.2015.7298682. URL `https://doi.org/10.1109/CVPR.2015.7298682`. 8

[40] Jyh-Jian Sheu, Ko-Tsung Chu, Nien-Feng Li, and Cheng-Chi Lee. An efficient incremental learning mechanism for tracking concept drift in spam filtering. *PloS*

one, 12(2):e0171518, 2017. 8

[41] Anatoly Shusterman, Lachlan Kang, Yarden Haskal, Yosef Meltser, Prateek Mittal, Yossi Oren, and Yuval Yarom. Robust website fingerprinting through the cache occupancy channel. In Nadia Heninger and Patrick Traynor, editors, *28th USENIX Security Symposium, USENIX Security 2019, Santa Clara, CA, USA, August 14-16, 2019*, pages 639–656. USENIX Association, 2019. URL https://www.usenix.org/conference/usenixsecurity19/presentation/shusterman. 4

[42] Anatoly Shusterman, Lachlan Kang, Yarden Haskal, Yosef Meltser, Prateek Mittal, Yossi Oren, and Yuval Yarom. Robust website fingerprinting through the cache occupancy channel. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 639–656, 2019. 2

[43] Anatoly Shusterman, Ayush Agarwal, Sioli O'Connell, Daniel Genkin, Yossi Oren, and Yuval Yarom. Prime+probe 1, javascript 0: Overcoming browser-based side-channel defenses. In Michael Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 2863–2880. USENIX Association, 2021. URL https://www.usenix.org/conference/usenixsecurity21/presentation/shusterman. 5

[44] Anatoly Shusterman, Zohar Avraham, Eliezer Croitoru, Yarden Haskal, Lachlan Kang, Dvir Levi, Yosef Meltser, Prateek Mittal, Yossi Oren, and Yuval Yarom. Website fingerprinting through the cache occupancy channel and its real world practicality. *IEEE Trans. Dependable Secur. Comput.*, 18(5):2042–2060, 2021. doi: 10.1109/TDSC.2020.2988369. URL https://doi.org/10.1109/TDSC.2020.2988369. 6, 7, 9, 10

[45] Anatoly Shusterman, Chen Finkelstein, Ofir Gruner, Yarin Shani, and Yossi Oren. Cache-based characterization: A low-infrastructure, distributed alternative to network-based traffic and application characterization. *Comput. Networks*, 200:108550, 2021. doi: 10.1016/j.comnet.2021.108550. URL https://doi.org/10.1016/j.comnet.2021.108550. 18

[46] Payap Sirinam, Mohsen Imani, Marc Juárez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1928–1943. ACM, 2018. doi: 10.1145/3243734.3243768. URL https://doi.org/10.1145/3243734.3243768. 6

[47] Paul Syverson, Roger Dingledine, and Nick Mathewson. Tor: The secondgeneration onion router. In *Usenix Security*, pages 303–320, 2004. 1

[48] Davide Tammaro, Silvio Valenti, Dario Rossi, and Antonio Pescapè. Exploiting packet-sampling measurements for traffic characterization and classification. *Int. J. Netw. Manag.*, 22(6):451–476, 2012. doi: 10.1002/nem.1802. URL https://doi.org/10.1002/nem.1802. 18

[49] Yukiyasu Tsunoo, Teruo Saito, Tomoyasu Suzaki, Maki Shigeri, and Hiroshi Miyauchi. Cryptanalysis of DES implemented on computers with cache. In Colin D. Walter, Çetin Kaya Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, volume 2779 of *Lecture Notes in Computer Science*, pages 62–76. Springer, 2003. doi: 10.1007/978-3-540-45238-6_6. URL https://doi.org/10.1007/978-3-540-45238-6_6. 5

[50] Alexey Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106(2):58, 2004. 6

[51] Petr Velan, Milan Cermák, Pavel Celeda, and Martin Drasar. A survey of methods for encrypted traffic classification and analysis. *Int. J. Netw. Manag.*, 25(5):355–374, 2015. doi: 10.1002/nem.1901. URL https://doi.org/10.1002/nem.1901. 18

[52] Jingdi Wang, Yundong Cao, Huaikang Jin, Shouyuan Fan, Bin Wang, and Huaiping Jin. An adaptive deep learning method integrating transfer learning and concept drift detection for wind power forecasting. In *2024 43rd Chinese Control Conference (CCC)*, pages 6275–6280. IEEE, 2024. 7

[53] Pingfan Wang, Nanlin Jin, Duncan Davies, and Wai Lok Woo. Model-centric transfer learning framework for concept drift detection. *Knowl. Based Syst.*, 275:110705, 2023. doi: 10.1016/J.KNOSYS.2023.110705. URL https://doi.org/10.1016/j.knosys.2023.110705. 6, 7

[54] Tao Wang and Ian Goldberg. Improved website fingerprinting on tor. In Ahmad-Reza Sadeghi and Sara Foresti, editors, *Proceedings of the 12th annual ACM Workshop on Privacy in the Electronic Society, WPES 2013, Berlin, Germany, November 4, 2013*, pages 201–212. ACM, 2013. doi: 10.1145/2517840.2517851. URL https://doi.org/10.1145/2517840.2517851. 6

[55] Tao Wang and Ian Goldberg. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In Engin Kirda and Thomas Ristenpart, editors, *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, pages 1375–1390. USENIX Association, 2017. URL https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/wang-tao. 6, 15

[56] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. Effective attacks and provable defenses for website fingerprinting. In Kevin Fu and Jaeyeon Jung, editors, *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014*, pages 143–157. USENIX Association, 2014. URL https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/wang_tao. 7

[57] Karl R. Weiss, Taghi M. Khoshgoftaar, and Dingding Wang. A survey of transfer learning. *J. Big Data*, 3:9, 2016. doi: 10.1186/s40537-016-0043-6. URL https://doi.org/10.1186/s40537-016-0043-6. 7

[58] Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, Yuxin Peng, and Zheng Zhang. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In Kien A. Hua, Yong Rui, Ralf Steinmetz, Alan Hanjalic, Apostol Natsev, and Wenwu Zhu, editors, *Proceedings of the ACM International Conference on Multimedia, MM '14, Orlando, FL, USA, November 03 - 07, 2014*, pages 177–186. ACM, 2014. doi: 10.1145/2647868.2654926. URL https://doi.org/10.1145/2647868.2654926. 8

[59] Ge Xie, Yu Sun, Minlong Lin, and Ke Tang. A selective transfer learning method for concept drift adaptation.

In Fengyu Cong, Andrew Chi-Sing Leung, and Qinglai Wei, editors, *Advances in Neural Networks - ISNN 2017 - 14th International Symposium, ISNN 2017, Sapporo, Hakodate, and Muroran, Hokkaido, Japan, June 21-26, 2017, Proceedings, Part II*, volume 10262 of *Lecture Notes in Computer Science*, pages 353–361. Springer, 2017. doi: 10.1007/978-3-319-59081-3_42. URL https://doi.org/10.1007/978-3-319-59081-3_42. 6, 7

[60] Yuval Yarom, Qian Ge, Fangfei Liu, Ruby B. Lee, and Gernot Heiser. Mapping the intel last-level cache. *IACR Cryptol. ePrint Arch.*, page 905, 2015. URL http://eprint.iacr.org/2015/905. 4

[61] Hang Yu, Tianyu Liu, Jie Lu, and Guangquan Zhang. Automatic learning to detect concept drift. *CoRR*, abs/2105.01419, 2021. URL https://arxiv.org/abs/2105.01419. 8

[62] Mojtaba Zaheri, Yossi Oren, and Reza Curtmola. Targeted deanonymization via the cache side channel: Attacks and defenses. In Kevin R. B. Butler and Kurt Thomas, editors, *31st USENIX Security Symposium, USENIX Security 2022, Boston, MA, USA, August 10-12, 2022*, pages 1505–1523. USENIX Association, 2022. URL https://www.usenix.org/conference/usenixsecurity22/presentation/zaheri. 5

[63] Wei Emma Zhang, Quan Z. Sheng, Ahoud Alhazmi, and Chenliang Li. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Trans. Intell. Syst. Technol.*, 11(3):24:1–24:41, 2020. 8