Generating a Large Web Traffic Dataset

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Generating a Large Web Traffic Dataset

Master Thesis

Benjamin Simmonds

August 31, 2023

Advisors: Prof. Dr. L. Vanbever, Dr. R. Meier

Department of Information Technology and Electrical Engineering, ETH Zürich

Abstract

In the realm of network security, machine learning has emerged as a crucial instrument. Unfortunately, the amount of available training data for network traffic-based models is very limited. This study aimed to bridge this gap. A tool was developed to autonomously gather vast amounts of website network traces. It did so by collecting a large amount of website addresses. Upon accessing these sites, the tool captured the resulting encrypted network traffic. This methodology enabled the collection of 900'000 distinct network traces spanning 220'000 unique website addresses. The data collection process was executed across eight geographically dispersed virtual machines, underscoring the comprehensive and diverse nature of the dataset. The effectiveness of the generated dataset was established by evaluating different Deep Fingerprinting attack scenarios.

Acknowledgement

Firstly, I would like to express my profound gratitude to my supervisor, Roland Meier, for his unyielding support and mentorship throughout the progression of this project. His teachings and dedicated investment of time have been invaluable to me, for which I am deeply appreciative. Further, I extend my heartfelt thanks to both Roland Meier and Laurent Vanbever, for allowing me to work on this project. Lastly, my sincere acknowledgment goes to Armasuisse and its Cyber-Defense Campus for generously supporting this project. Their provision of essential infrastructure has been instrumental in the successful completion of this work.

Contents

Chapter 1

Introduction

Machine learning has emerged as a pivotal instrument in the realm of network security, for defensive measures such as detecting malicious traffic, and offensive measures such as fingerprinting Tor traffic. Training these models necessitates vast amounts of data. Yet, the volume of accessible network traffic data remains scarce. Consequently, researchers often find themselves forced to first construct systems for data collection, diverting their attention from their primary objective: developing advanced network-based machine learning models.

To address this challenge, our goal is to design a tool capable of autonomously generating an extensive dataset centered on encrypted network traffic. This would streamline the data collection process for researchers, enabling them to easily train a variety of models. This study specifically focuses on website traffic, recognizing that websites serve as principal interaction nodes for users and are frequently exposed to a variety of attack vectors. By narrowing the scope to website traffic, the resulting dataset ensures a comprehensive analysis of dominant threats.

Furthermore, labeling the collected website network traces is a crucial step in preparing this data for effective model training. The labeling process ensures that the machine learning model can understand the differences between varying website traffic. However, there are countless methods to label these traces. One common approach is to use the website's address as the label to precisely identify the site being visited. While this method can offer accurate labeling, it unfortunately strongly limits the scope of recognizable websites as it only allows the model to detect those websites that were present in the training set. Thus, exploring other labeling strategies is essential.

Chapter 2

Background

In this chapter, we examine how HTML and HTTP interact together when web browsers load web pages. We also discuss existing website address datasets and clarify why they are not utilized in this study. Finally, we introduce Deep Fingerprinting which will be used to evaluate the quality of the network traffic dataset we produce.

2.1 Web Browsing

Web browsing today, a core function of the World Wide Web [1], is driven by the interaction between HTML (HyperText Markup Language) [2] and HTTP (HyperText Transfer Protocol) [3]. Central to this system, HTML outlines the structural detail of web pages, specifying elements such as headings, paragraphs, links, and multimedia through a uniform markup language. Starting a web browsing session involves the browser sending an HTTP request to a server. In turn, this server responds with the appropriate HTML document. However, a modern webpage typically goes beyond just text; it also includes images, videos, stylesheets, scripts, and various multimedia components. These resources are typically referenced within the HTML using tags like <img>,<link>, and <script>. When the browser encounters such tags, it sends additional HTTP requests to fetch each resource. As these are being retrieved, the browser integrates them into the page, rendering the complete interface users interact with. The collaborative interaction between HTML and HTTP ensures seamless access, presentation, and navigation of all web content.

2.2 Related Work

2.2.1 Existing Network Traffic Datasets

The rise of machine learning has led to its growing application in the network security domain. A crucial aspect of this process involves the collection of extensive datasets for the training of these models. However, a significant obstacle to progress in this field is the lack of data sharing among researchers. The datasets generated during research are rarely shared with the broader academic community, thereby hindering the ability of others to validate or build upon existing work.

The emergence of several publicly available datasets, albeit limited, is a step in the right direction. However, often these datasets are narrowly focused on specific applications, such as IoT denial of service attacks [4], or TCP FIN Flood Attacks [5]. While the availability of such specialized datasets is undoubtedly beneficial for targeted research, their narrow scope restricts their utility across a broader spectrum of applications. This underscores the importance of developing comprehensive datasets and making them available to facilitate a wide array of research inquiries, ultimately driving more advancements in network security.

2.2.2 Existing URL Collections

Existing website collections like Common Crawl [6] have emerged as a valuable resource for researchers from various fields. They span billions of web pages and aim to offer a snapshot of the Internet's vast expanse, ensuring that information is available to all. Captured over diverse periods and languages, the collection presents a valuable resource for data analysis.

Even though this thesis could leverage the URLs (Uniform Resource Locators) from the Common Crawl dataset as a foundational stepping stone, it lacks some data points on the websites that are required for this thesis such as the server's response time (Section 4.1). A potential solution could be to construct an extension of the Common Crawl dataset, by requesting each website of the collection to measure the required server response time. Nonetheless, this solution has its drawbacks. Primarily, dealing with Common Crawl would necessitate downloading an immense volume of data, far exceeding the scale of the collection this thesis is aiming to construct. Further, the organization of data in the Common Crawl dataset is sorted by hostname, which can make the process of data retrieval and management convoluted and time-consuming.

Other platforms like SimilarWeb [7] and Ahrefs [8] offer insights into websites, detailing metrics such as traffic, engagement, demographics, and many more. They aggregate data from diverse sources, including direct measurements and web crawlers. However, they also lack some data points on the websites such as the server response time, and information on pages, other than the home page, hosted on the same domain. Moreover, these platforms typically grant access to only a limited subset of their data, requiring paid subscriptions for full access.

While platforms like Common Crawl, SimilarWeb, and Ahrefs offer valuable insights, they also have their limitations. Rather than relying on these existing datasets, designing and deploying a new, purpose-built web crawler stands out as a more efficient strategy. The crawler can be specifically configured to the unique needs of this work, thereby making it easier and faster to obtain, manage, and analyze the desired website collection.

2.2.3 Deep Fingerprinting

Tor, with its vast user base for anonymous browsing, remains open to website fingerprinting (WF) attacks [9], which can identify encrypted web pages through unique network traffic patterns. Classic WF attacks, using classifiers like SVM, k-NN, and random forests, have reported up to 90% accuracy. To counter such threats, various strategies, including introducing dummy traffic and delaying packet transmission, have been employed. Deep learning, known for its dominance in fields like speech and visual recognition, offers a fresh approach to WF. Deep Fingerprinting [9] harnesses Convolutional Neural Network techniques to gain remarkable accuracy improvements over traditional methods. Deep Fingerprinting, however, relies on having an extensive training set. To address this limitation more effectively, Var-CNN [10] takes a mixed approach. While both Deep Fingerprinting and Var-CNN utilize the directions of network packets, distinguishing between client-to-server and server-to-client communication, Var-CNN goes beyond by integrating additional metrics. It leverages other vital parameters like timing information and metadata. This strategy significantly enhances Var-CNN's efficiency in scenarios constrained by limited data availability.

Chapter 3

Overview

There are numerous unresolved research questions related to the analysis of website network traffic. These range from tracking user activities through Tor networks to inferring information from user requests based solely on a sequence of network packets. To address these questions effectively, a large amount of training data specific to network traffic-based models is necessary. As of now, the availability of such data is limited. Thus, the primary objective of this thesis is to develop a tool capable of generating an extensive network traffic dataset tailored to the requirements of researchers, focusing specifically on website network traffic.

The intended dataset should be constructed using web browser automation and by executing experiments that account for various network characteristics. For instance, the goal is to capture website traffic from diverse global regions while considering different bandwidth constraints. However, before simulating user requests, it is essential to collect a set of active website addresses, as these will be the addresses being requested during the simulation. To achieve this, a web crawler will be developed. Once the network traffic from the identified websites is captured, the robustness and relevance of the dataset will be tested. This process will involve using the captured data to reproduce existing traffic-analysis attacks, such as Web Fingerprinting. In the final stages, alternative data labeling strategies will be explored and tested, going beyond merely using the website's address, as this approach constrains the attack's scope to the size of the training set.

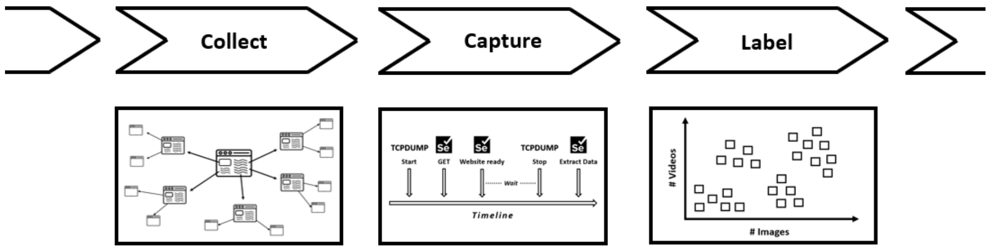


Figure 3.1: A simplified overview of the framework of this study.

Chapter 4

Collecting Website Addresses

To capture an extensive volume of website network traffic that represents user activity on the web around the world, a comprehensive collection of website addresses is necessary.

4.1 Requirements

To ensure efficient capturing of network traffic from hundreds of thousands of websites, it's vital that the sites within the collection respond within a reasonable timeframe, ensuring the entire process does not take too long. There are two primary reasons for delays when requesting websites. First, the websites may no longer be hosted, causing browsers to attempt to retrieve inaccessible sites. Second, some sites may be hosted on slow servers, leading to lengthy response times when attempting to load the web page. Including numerous such sites in the collection can considerably extend the duration of the traffic-capturing process.

Additionally, the collection should include not only the addresses of the main pages of a web domain (e.g. *www.facebook.com/*) but also various resources hosted on the websites. This is essential because the primary content that represents a website isn't always located on its main page. For instance, when accessing Facebook without logging in, users are directed to a login page, which doesn't truly capture the core content of the site. In contrast, accessing pages like MySwitzerland on Facebook (*www.facebook.com/myswitzerland*) offers a perspective that better mirrors actual user requests and associated network traffic. This underlines the importance of a well-curated collection of website addresses.

4.2 Collecting Website Addresses

The HTML structure of a website often incorporates various references to other websites, serving as a resource for generating an extensive set of URLs. The process initiates with the selection of an initial website, from which the URL collection begins. Upon requesting this website, it is scanned for references to external websites. This is accomplished by examining all *href* tags within the HTML code. Subsequently, all identified URLs are extracted and filtered. During this stage, references that are HTML IDs (e.g., #id) or Uniform Resource Identifier (URI) schemes, such as "mailto:", are discarded. The remaining valid addresses are subsequently inserted into the set of discovered URLs. The process then advances by randomly selecting a website from this collection, repeating the URL-gathering process from the newly chosen website.

In most instances, a significant proportion of URLs extracted from a website are addresses belonging to the same domain, with a minority linking to external sites. When these addresses are incorporated into the collection, during subsequent runs, there is a significantly higher probability of visiting a webpage from a previously accessed domain compared to visiting a new one. This scenario typically results in a collection of URLs that do not represent a diverse range of domains.

To mitigate this issue, the collection is organized as a mapping, associating the domain names of the discovered websites with their corresponding subdomains, each of which maps to a list of website addresses specific to that subdomain. This approach enables a balanced selection process among different domain names and their respective subdomains when deciding on the next website to be retrieved.

4.3 Website Collection Quality

The strategies previously discussed enable the generation of an extensive dataset comprising various websites from around the globe. Nonetheless, the quality of the gathered website addresses remains uncertain. There is a possibility that the websites may no longer be responsive, the request might respond with an error (e.g. page not found), or could be hosted on slow-responding servers. These circumstances can significantly compromise network data quality obtained from these websites and extend the web traffic capturing process duration.

Given that website requests are already being made during the process of collecting website addresses, information gained from the requests can be leveraged to identify whether or not a particular URL should be included in the network data-capturing process. If the server responds with just the

```
{
    "google.com": {
        "google.com": [
            "google.com",
            "google.com/maps",
            "google.com/developers"
        ],
        "store.google.com": [
            "store.google.com/de"
        ]
    },
    "linkedin.com": {
        "linkedin.com": [
            "linkedin.com/15822",
            "linkedin.com/65127",
            "linkedin.com/15877",
            "linkedin.com/36985",
            "linkedin.com/11452"
        ]
    },
    "ethz.ch": {
        "ethz.ch": [
            "ethz.ch/en"
        ]
    }
}
```

Figure 4.1: An example website dataset with 3 different domains

HTML code within a specific time frame without receiving an error code, the website is then added to a secondary collection, signifying the websites that remain hosted and responsive within that set period. This polished collection will be utilized during the traffic-capturing process.

It's crucial to acknowledge that there isn't a ”perfect” timeframe that can be universally applied. Some slower servers might still find their way into the set, and conversely, some efficient websites may not be included. However, such imperfections are tolerable. Incorporating a few slower servers into the dataset is acceptable, provided the majority are filtered out. Similarly, missing out on a few good websites isn't detrimental given the vast number of alternative websites available. Evaluating multiple timeframes, a specific span of 5 seconds emerged as the most suitable choice. This period was selected as it effectively detected non-responding and very slow servers, yet still incorporated websites that loaded at a slower pace due to the geographical distance between client and server.

Additionally, the process of gathering website addresses can be significantly accelerated by implementing a multithreaded approach. This method allows multiple threads to concurrently send requests and process responses, thereby reducing the overall time required to compile the polished collection of website addresses for the traffic-capturing process.

Chapter 5

Collecting Network Traffic

In this chapter, we explore methods to automate the simulation of web page loading on real-user devices and detail the process of capturing the resulting network traffic.

5.1 Simulating Browsers

In order to accurately simulate the dynamic interaction between a client and servers when browsing the web, it is crucial to replicate the process a web browser employs to load a website. Selenium [11] is a powerful framework that facilitates web browser automation. With the aid of a web driver, Selenium not only reliably loads websites inclusive of all their dependencies but also grants the ability to interact programmatically with the page, providing a comprehensive and realistic approach to web interaction simulation. During the course of this work, Chrome's web driver was utilized, considering that it is, by a significant margin, the most frequently used browser by users worldwide [12, 13].

5.2 Capturing traffic

To capture encrypted web traffic during website visits, a packet analyzer is employed. Tcpdump [14] was chosen as it is a command-line packet analyzer renowned for its high performance. It allows for the capture of internet packets which are filtered by IP address and then stored in a file for later analysis.

The process to generate the network traffic dataset is as follows: a website is sequentially selected at random from the website address collection. Tcpdump is then initiated with the configuration to only store packets sent towards or from the device running the process. The chosen website is then requested and loaded via Selenium. When Selenium indicates that the page is ready, it is important to note that this does not necessarily mean all resources, such as images, have been fully loaded. Given this, Tcpdump continues to capture for an additional five seconds. This duration was selected after experimenting with multiple values. Five seconds provides sufficient time for capturing traffic related to the lazy loading of the website's resources or ongoing video streaming, without excessively prolonging the total capture time, particularly as thousands of websites will be processed sequentially. After this period, the packet analyzer is halted, resulting in a pcap file that captures the web traffic.

To analyze and label the collected website traces for various web traffic-based models, we store additional metadata related to the request. This metadata includes a website screenshot taken by Selenium, the HTML code, client location details from which the request originated, any encountered errors, and the request timestamp. After extraction, the next website is selected, and the previously mentioned network-capturing process repeats.

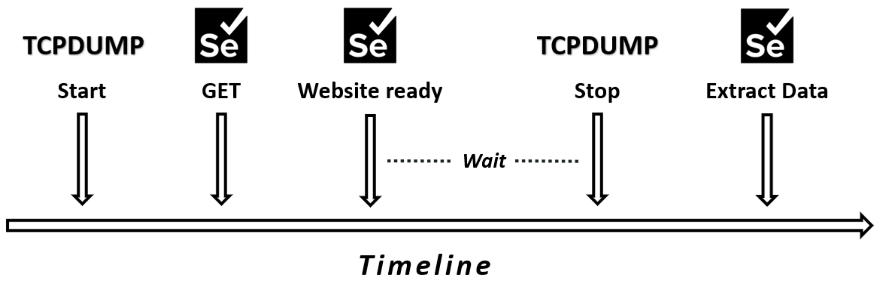


Figure 5.1: The process of capturing the traffic of a single website with Tcpdump and Selenium.

5.2.1 Bandwidth Limitation

To simulate realistic user traffic effectively, it is essential that the bandwidth size is adjustable. Running the collection from large data centers significantly impacts traffic due to their capacity to support considerably larger bandwidths compared to typical user devices. As a solution, Wondershaper [15] is employed to cap both upload and download rates.

To establish appropriate bandwidth constraints, data from major internet service providers was studied. Broadband, one of the leading ISPs in the US, reports an average national download speed of 185.04 Mbps [16]. They also offer upload speeds reaching 20 Mbps. On the other hand, T-mobile indicates average download rates ranging from 72 Mbps to 245 Mbps, and upload rates between 15 Mbps and 31 Mbps [17]. Based on these findings, the download bandwidth limit is set to 180 Mbps, and the upload bandwidth is capped at 20 Mbps for the data collected during this work.

Chapter 6

Labeling Websites

For the successful training of machine learning models utilizing the gathered network traffic data, it is essential that this data is appropriately labeled. While it may initially appear that the website's URL could serve as a logical label, this approach becomes less reliable when working with extensive datasets. Machine learning models might struggle to accurately pinpoint a specific website among thousands based solely on the network traffic data. Moreover, it would not be possible to recognize websites that are not included in the training set. Given these complexities, it becomes clear that alternative labeling strategies should be considered.

6.1 Website Categorization

Websites can be classified into various categories, often sharing a common purpose that may also reflect in their respective website structure and traffic. Utilizing website categories as labels provides a more coarse-grained approach than utilizing the URL. This method broadens the scope, enabling the classification of websites that may not have been part of the initial training set.

A difficulty with this approach is categorizing training data, consisting of thousands of never before seen websites. The immense volume and diversity of these websites underscore the magnitude of the challenge, making traditional manual classification methods impractical and time-consuming. Employing Large Language Models(LLM) can help automate this as these models possess the ability to understand the content and can classify wide amounts of data in a relatively short time.

To achieve this, the HTML code, which is stored during the collection process, is extracted to serve as input. Before feeding this data to the LLM, it is crucial to cleanse the HTML content by removing all tags, scripts, CSS styling, navigation, footer, and any other non-essential element that does not represent the actual content of the website. This cleansing is crucial not only to ensure the model can more accurately and efficiently classify the content but also because these models typically accept a limited input size. Thus, it is vital to maximize the relevance of the content within that constraint. Once the HTML is cleaned, the resulting content is integrated into a prompt. This prompt is designed to request the LLM to categorize the website based on its content. Alongside the request, a set of categories is provided for the model to select from. Once the prompt is presented to the model, the website can be labeled with the determined category.

6.2 Structural labeling

Recognizing the category to which a website belongs based on its encrypted network traffic remains challenging. This is because websites from different categories can appear visually identical, leading to similarities in their traffic behavior.

The patterns of network traffic when rendering websites are predominantly influenced by the resources that browsers need to fetch. These resources include media, CSS, and Javascript files. By utilizing the structural information of websites as labels, machine learning models might find it easier to predict them. Furthermore, this approach would still yield insights into the website's content and distinctive features. A practical metric for this would be the number of images displayed on a website, since the quantity of images directly corresponds to the website's request count and, by extension, the number of packets in its network traffic.

However, labeling the training data is not as straightforward as counting the image tags in the HTML code. Such an approach might inflate the total number of displayed images, especially if the website employs lazy loading. This mechanism might withhold requests for images not currently visible within the viewport. Moreover, relying solely on counting tags can be misleading as images can be loaded via CSS and JavaScript. Hence, to obtain the precise number of loaded images, one must access the browser developer tools. Fortunately, Selenium offers a mechanism to do just that.

Chapter 7

Dataset Evaluation

In this chapter, the dataset collected during the course of this research is examined. The applicability of the data is assessed by replicating existing traffic-analysis attacks. Additionally, the effectiveness of automating website categorization with LLMs is evaluated. Finally, other potential applications of the generated data are explored such as assessing how well Deep Fingerprinting can predict the number of images loaded from a website's network trace.

7.1 Website Fingerprinting

We aim to assess the performance of the Deep Fingerprinting model across diverse website fingerprinting scenarios, leveraging our curated dataset. Initially, we replicate the scenario mentioned in the original paper to validate the utility and reliability of our dataset. Subsequently, we extend our evaluation to explore additional scenarios that may offer a more realistic representation of website fingerprinting challenges.

7.1.1 Data Quality

To assess the quality of the dataset generated, we tested how well our data could reproduce existing traffic-analysis attacks. In our case, we mainly focused on the deep fingerprinting attack [9] and compared our results with those presented by the original authors. Following the methodology outlined in the paper, we collected 1000 network traces for 95 of the most popular websites as listed by Alexa [18] utilizing one of our virtual machines for the task. Employing the same model as the study, our data yielded a 97% accuracy, slightly lower than the 98% reported by the authors. However, when our data was applied to the Var-CNN model [10], it precisely mirrored the 98% accuracy cited in the paper, indicating our ability to largely replicate the results of the WFP attack. The minor discrepancy in the initial comparison may be attributed to variations in our data collection approach compared to that of the original authors. The paper indicates a more or less continuous data collection process, whereas our approach involved gathering website samples over multiple weeks. Web pages often change, which impacts the network traffic generated when loading a page, thereby complicating the model's task of identifying patterns across identical website samples.

Having demonstrated that the data harvested using our tool is effectively applicable to network-based machine-learning attacks, we can shift our focus towards exploring potential applications of the vast amount of data we collected.

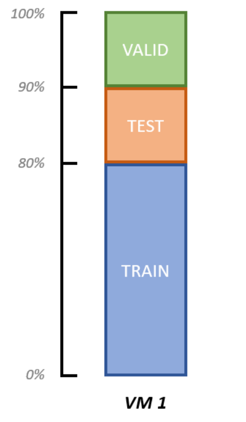
7.1.2 Realistic Attacks

Upon examining the Deep Fingerprinting paper [9], it is clear that the dataset collection (including training, test, and validation sets) was conducted using machines located at a single geographical location (Figure 7.1a). This setup may not properly reflect real-world attack conditions. Typically, attackers would train a model using data from devices under their control, then apply the trained model to data gained from a separate device not under their possession, from which they do not have training set samples and possibly positioned far away from the attacker's devices. Such variations can introduce discrepancies in the attack traces relative to the training dataset, potentially impacting the model's accuracy.

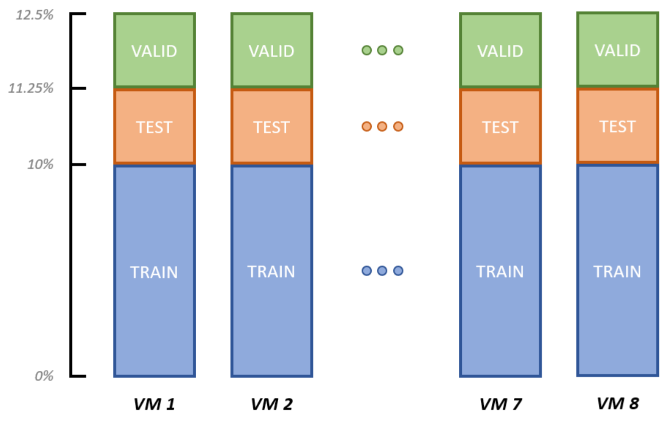
To investigate the impact of varying geographical locations on the Website Fingerprinting (WFP) attack, we systematically gathered traces from the previously mentioned 100 websites across our 8 globally distributed machines. We then divided the gathered samples from each machine into training, testing, and validation sets at an 8:1:1 ratio, as outlined in the paper (Figure 7.1b). After training the model using these sets, we achieved an accuracy rate of 91%. While this confirms the viability of the attack across multiple geographical points, it also suggests a potential decrease in the model's performance.

However, this experiment still does not fully simulate a realistic attack scenario for WFP, as the same devices contributed to both training and testing sets. For a closer approximation to a real-world scenario, we constructed the training and validation sets using samples from 7 of the 8 machines. The samples from the remaining machine were exclusively utilized for the test set (Figure 7.1c). This machine represents the attacker's target. Due to the relative positioning of the target device to the attacker's devices potentially influencing the model's performance, tests were performed for every combination, where one device served as the target and the other 7 as the attacker's training devices.

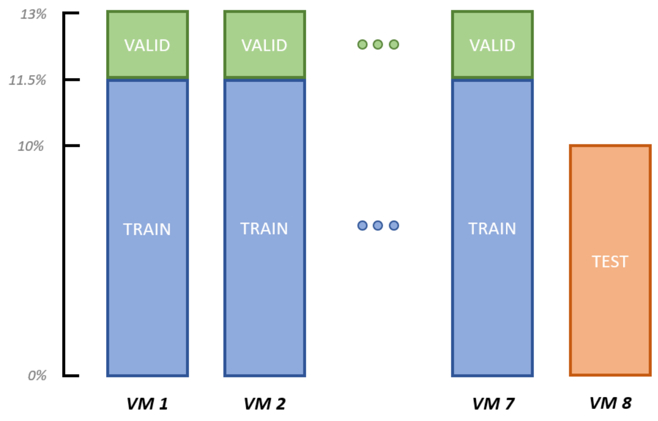
After running the tests the following results could be observed: (Figure 7.2) When the target device was in close proximity to one or more of the attacker's devices, the model's accuracy remained relatively high, reaching up to 89%. However, when the target device was situated at a greater distance, accuracy declined to 78%. This indicates that while the location from which a web page request is made can affect network traces, it does not render them entirely unrecognizable to the model.



(a) The dataset was collected from a single location as described in the Deep Fingerprinting paper.



(b) The dataset is equally distributed across 8 different machines positioned in different locations. Each machine contributes 10% of the entire dataset to the training set and 1.25% to the test and validation sets each.



(c) Training and validation sets are constructed from 7 different machines. The remaining machine contributes samples only to the test set.

Figure 7.1: Distribution of datasets utilized for WFP attacks across one or more machines. In all cases, the data split between training, validation, and test sets adheres to an 8:1:1 ratio. Samples originating from the same website maintain this ratio equal across all machines.

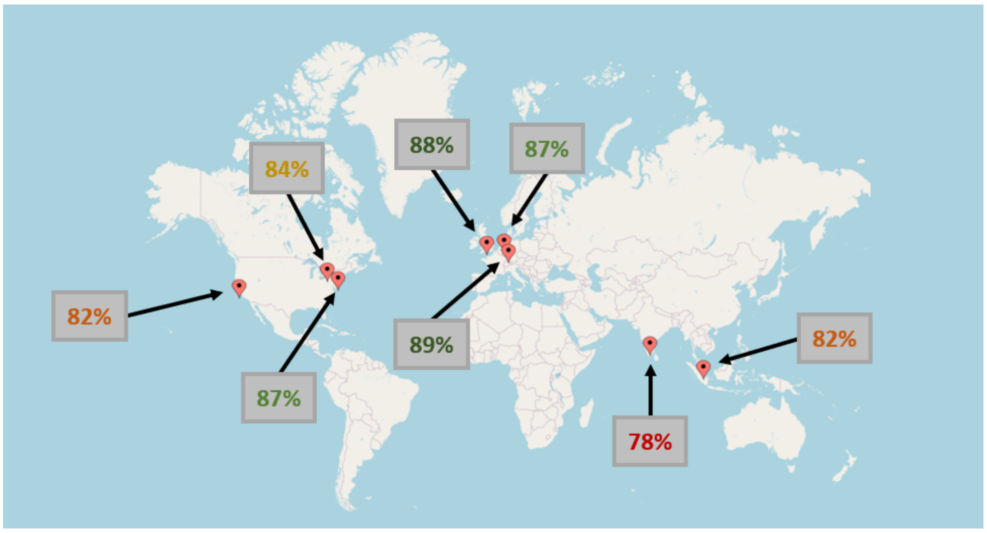


Figure 7.2: Geographical representation of model accuracy based on the position of the target device (test set) relative to all other devices used by the attacker (training and validation sets).

7.2 Categorizing Websites with LLMs

To assess the ability of Large Language Models (LLMs) to categorize websites using content extracted from the HTML code, OpenAI's state-of-the-art language model was evaluated [19]. Through their API [20], the categorization process was automated. The model's prediction accuracy was determined by comparing its outputs with Similarweb's [7] dataset, which contains manual categorizations of popular websites. While Similarweb provides categories for only the top 50 websites for free [21], this volume is sufficient to gauge the model's performance.

For each of the 50 websites, HTML code was obtained and processed as outlined in section 6.1. The refined text was then structured into a coherent prompt, accompanied by the list of website categories provided by Similarweb. Before finalizing and submitting the prompt, its length was verified to ensure compliance with the model's input size limitations. If the content exceeded the allowable token count, it was truncated to fit within the specified length. After submission, the model's categorization was compared to that of Similarweb. OpenAI's "text-davinci-003" model [22] correctly categorized 68% of the websites in alignment with Similarweb's classification.

7.2.1 Limitations

The model's performance may not meet certain expectations for several reasons. A key limitation is the constraint on input size. Specifically, the DaVinci model is restricted to accepting 4096 tokens. Each token is approximately equivalent to four characters, and both the prompt message listing all categories and the response text count toward the 4096 token limit. This limitation led to 24% of the websites exceeding this capacity, compelling them to omit potentially significant data that could have helped to categorize the website.

A further limitation arises when websites are concealed behind login pages. Consequently, the data extracted from these pages will not accurately reflect the website's true content. As websites from all types of categories have login pages, it makes it challenging for a language model to accurately determine the category to which the website belongs.

Lastly, determining the exact category of a website can be challenging in general, as many sites incorporate elements from multiple categories. For instance, while Bing (bing.com) is primarily recognized as a search engine, the language model categorized it under News & Media due to the multitude of news articles on its homepage. Conversely, Naver and Yahoo (naver.com and yahoo.com), both fundamentally News websites, were classified as search engines because they feature a search engine interface at the top of their pages.

In essence, while Language Learning Models (LLMs) offer significant capabilities, they might not match human proficiency in categorizing websites. Humans not only process textual information but can also interpret and utilize visual indicators on a website, providing a comprehensive understanding that often aids in accurate categorization. On the other hand, LLMs, primarily focusing on textual content, might miss these visual details. However, it's worth noting that LLMs can categorize websites at a much faster rate compared to humans.

7.3 Predicting Website Images

In this study, our aim extends beyond merely fingerprinting a select group of known websites. Instead, we aim to predict the characteristics of any website based solely on its internet traffic patterns. We evaluated the success of deep fingerprinting models in predicting the number of images displayed on websites.

7.3.1 HTML Image Tags vs Loaded Images

It is imperative to note that during the data collection phase, the actual number of loaded images was stored (Section 6.2). This is significant because there can be a disparity between the number of images specified in the HTML code and the number that is actually loaded, as illustrated in Figure 7.3. Documenting this disparity is essential, as addressing it may improve the performance of the model. For 35.63% of the analyzed websites, there was a negative delta between these numbers. This suggests that these sites may use techniques such as ”lazy loading” for their images, by setting the ”loading” attribute to ”lazy” in their image tags [23]. On the other hand, the remaining 64.37% of websites either displayed an equal number of images as specified in the image tags or more. This behavior can be attributed to images being loaded through CSS and/or JavaScript. While images can also be lazy-loaded using JavaScript (loading an image as it becomes visible in the viewport), it does not account for the observed variance between the number of image tags and the actual loaded images.

7.3.2 Evaluating Deep Fingerprinting on Image Range Predictions

Given our speculation that the Deep Fingerprinting model might struggle with predicting the exact number of loaded images on a webpage based solely on packet directions, we opted to categorize the website traces based on ranges of image numbers. To illustrate, if the range length is set to 16, the model would attempt to predict whether the webpage has 0-15 images, 16-31 images, 32-47 images, or 48-63 images. This categorization aimed to simplify the model's prediction task. To understand the impact of image

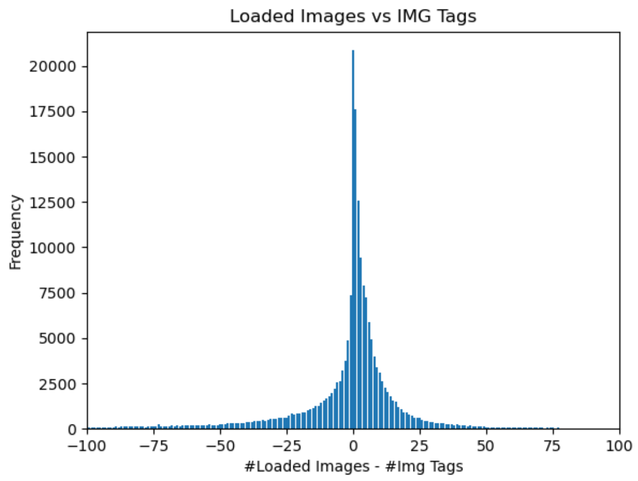


Figure 7.3: Comparison between the count of HTML image tags and the actual images loaded. Negative values signify more 'img' tags than images effectively loaded.

range sizes on the model's accuracy, various range lengths were evaluated for websites with up to 64 loaded images. Considering that the distribution of the number of loaded images declines exponentially (as seen in Figure 7.4), only a specific subset of the dataset was utilized for training to avoid a strong bias toward lower image ranges. For each image range, an equal number of samples were employed for model training.

In testing, the trained Deep Fingerprinting model outperformed the random selection of an image range. However, it struggled to consistently determine the appropriate image range for a website, especially when higher precision was required for shorter image ranges. This suggests that while Deep Fingerprinting can proficiently identify a visited website from a limited set of options, it falls short in extracting additional request information, such as the web page's structural features.

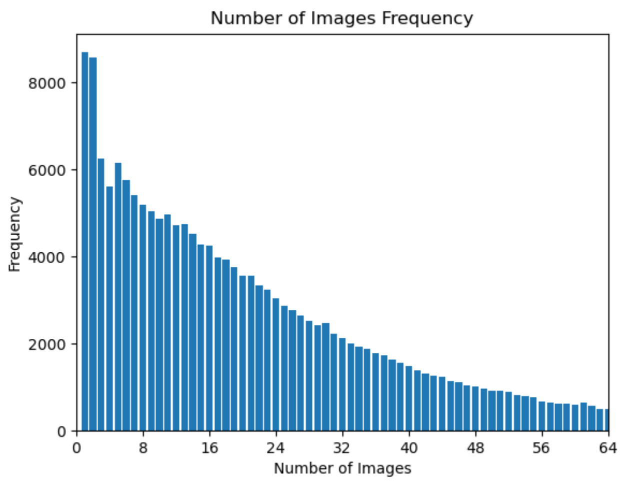


Figure 7.4: Distributation of number of images displayed on each website

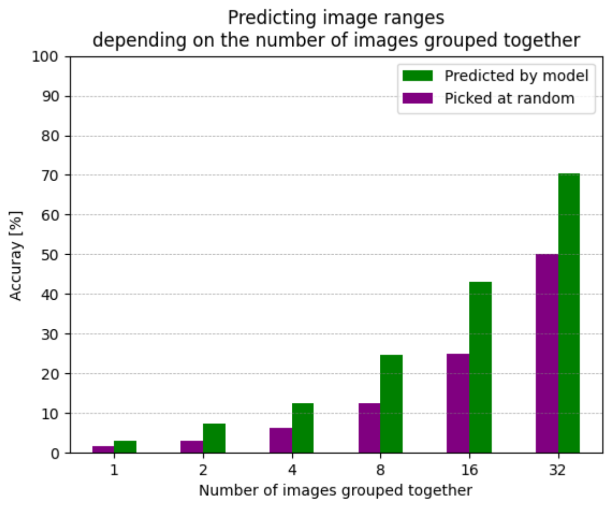


Figure 7.5: Accuracy of the Deep Fingerprinting model in determining the image range for websites, evaluated across various range lengths for web pages with up to 64 loaded images, in comparison to random selection.

7.4 Website Network-Traffic Dataset

During this research, an exhaustive dataset was gathered over a span of 12 weeks. This 6 Terabyte dataset contains 900’000 network traces across 220,000 distinct websites, along with their associated metadata. This was achieved by deploying eight virtual machines globally, positioned in data centers provided by DigitalOcean [24]. These data centers are distributed across key locations, namely New York, San Francisco, Toronto, Amsterdam, London, Frankfurt, Singapore, and Bangalore. This geographic distribution was aimed at capturing a diverse range of network traffic patterns and potential regional discrepancies. For efficient querying and extraction, the data is stored in an Elasticsearch database [25].

Moreover, samples were collected from 100 of the most frequently visited websites, as ranked by Alexa in February 2023 [18]. A minimum of 1,000 samples were uniformly gathered across the eight distinct locations for each website. It's important to note that the exact top 100 websites could not be included in all cases. Legal restrictions in certain countries prevented access to some of these sites, rendering it impossible to obtain traces from those locations. Consequently, substitutions were made with other websites that were accessible across all regions.

Chapter 8

Conclusion

8.1 Conclusion

In this thesis, our primary objective was to generate a comprehensive dataset focused on web traffic. To achieve this, we crafted a specialized tool to capture encrypted traffic across a diverse range of websites. Our strategy incorporated the design of a web crawler, resulting in a broad compilation of website addresses. Following this, we simulated realistic user requests by employing an automated web browser that operated under average bandwidth constraints. This simulation spanned multiple global locations. Using the packet analyzer Tcpdump, we documented the traffic generated from these page loads. A total of 900,000 distinct network traces were obtained, across 220,000 individual website addresses.

To validate the quality and usability of our dataset, we sought to replicate the outcomes from the renowned Deep Fingerprinting traffic-analysis attack. Our exploration spanned various scenarios, especially those focusing on geographical locations. One significant discovery was that the model's accuracy can be notably impacted when both training and applying it across diverse geographical locales.

A crucial element of our research was the labeling strategy. Recognizing that limiting labels to website addresses might constrain the attack's scope to the training set's websites, we ventured into alternative labeling methodologies. This led us to explore a range of strategies, from the categorization of websites using LLMs to the adoption of loaded image counts as a viable metric. However, the Deep Fingerprinting model encountered challenges in precisely predicting the number of images on websites. Additionally, certain complications arose when employing LLMs for website categorization.

In conclusion, this research not only delivers a robust web traffic dataset but also sheds light on the efficacy of current analytical techniques in the domain.

We anticipate that the findings and tools highlighted here will significantly benefit future endeavors in web traffic analysis.

8.2 Future Work

In the course of this work, the emphasis was placed on generating encrypted website traffic, however, the scope of network security is not limited to website network traffic alone, thereby underlining the importance of broadening the dataset and the tool developed in this study to include other types of traffic. The goal should be to create a network traffic dataset that accurately represents all types of network traffic. This completeness is crucial for the training of models designed to handle a wide array of network traffic-based applications. To broaden the dataset, it is imperative to devise diverse methods for generating authentic network traffic, ranging from traffic occurring from file transfers and streaming to Voice over IP. It is essential to recognize that each traffic type necessitates a unique approach for its generation. Furthermore, different types of devices, such as mobile devices and Internet of Things (IoT) devices, should be considered for traffic generation, as they contribute significantly to the overall network traffic and pose unique challenges and patterns that need to be accounted for in the dataset. As the prevalence of IoT devices continues to grow, their impact on network traffic and subsequently network security cannot be overlooked. Therefore, future work should also focus on incorporating traffic from a diverse range of devices to ensure a comprehensive and representative dataset.

Bibliography

[1] J. Powell and A. Clarke, "The www of the world wide web: who, what, and why?" *Journal of Medical Internet Research*, vol. 4, no. 1, p. e4, 2002.

[2] D. Raggett, A. Le Hors, I. Jacobs *et al.*, "Html 4.01 specification," *W3C recommendation*, vol. 24, 1999.

[3] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext transfer protocol–http/1.1," Tech. Rep., 1999.

[4] F. Hussain, S. G. Abbas, M. Husnain, U. U. Fayyaz, F. Shahzad, and G. A. Shah, "Iot dos and ddos attack dataset," 2021. [Online]. Available: https://dx.doi.org/10.21227/0s0p-s959

[5] D. Stiawan, D. Wahyudi, A. Heryanto, S. Sahmin, Y. Idris, F. Muchtar, M. Alzahrani, and R. Budiarto, "Tcp fin flood attack pattern recognition on internet of things with rule based signature analysis," *International Journal of Online and Biomedical Engineering (iJOE)*, vol. 15, p. 124, 04 2019.

[6] Common Crawl. Accessed on 15.06.2023. [Online]. Available: https://commoncrawl.org/

[7] SimilarWeb. Accessed on 03.08.2023. [Online]. Available: https://www.similarweb.com/

[8] Ahrefs. Accessed on 15.08.2023. [Online]. Available: https://ahrefs.com/

[9] Sirinam, Payap and Imani, Mohsen and Juarez, Marc and Wright, Matthew, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *2018 ACM SIGSAC Conference on Computer and Communications Security (CCS '18)*. Toronto, ON, Canada: ACM, October 15–19 2018, pp. 1–16. [Online]. Available: https://doi.org/10.1145/3243734.3243768

[10] S. Bhat, D. Lu, A. Kwon, and S. Devadas, “Var-cnn: A data-efficient website fingerprinting attack based on deep learning,” *arXiv preprint arXiv:1802.10215*, 2018.

[11] Selenium. Accessed on 20.06.2023. [Online]. Available: https://www.selenium.dev/

[12] BrowserStack: Browser Market Share. Accessed on 17.08.2023. [Online]. Available: https://www.browserstack.com/guide/understanding-browser-market-share

[13] Chrome Web Driver. Accessed on 09.08.2023. [Online]. Available: https://chromedriver.chromium.org/

[14] tcpdump. Accessed on 20.06.2023. [Online]. Available: https://www.tcpdump.org/

[15] Wondershaper. Accessed on 07.08.2023. [Online]. Available: https://github.com/magnific0/wondershaper

[16] Broadband Bandwidth Average. Accessed on 07.08.2023. [Online]. Available: https://broadbandnow.com/XFINITY-speed-test

[17] T-mobilr Bandwidth Average. Accessed on 07.08.2023. [Online]. Available: https://www.t-mobile.com/home-internet/faq

[18] Alexa Top 100 websites. Accessed on 07.08.2023. [Online]. Available: https://www.expireddomains.net/alexa-top-websites/

[19] OpenAI. Accessed on 03.08.2023. [Online]. Available: https://openai.com/

[20] OpenAI API. Accessed on 03.08.2023. [Online]. Available: https://openai.com/blog/introducing-chatgpt-and-whisper-apis

[21] SimilarWeb Top Websites. Accessed on 03.08.2023. [Online]. Available: https://www.similarweb.com/top-websites/

[22] OpenAI Models. Accessed on 03.08.2023. [Online]. Available: https://platform.openai.com/docs/models/gpt-3-5

[23] Lazy Loading Images in HTML. Accessed on 08.08.2023. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Performance/Lazy_loading

[24] Digitalocean Datacenters. Accessed on 17.08.2023. [Online]. Available: https://docs.digitalocean.com/products/platform/availability-matrix/

[25] Elasticsearch. Accessed on 20.06.2023. [Online]. Available: https://www.elastic.co