Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Networked Systems
ETH Zürich — seit 2015

Website Fingerprinting in the Data Plane

Semester Thesis

Author: Sunniva Flück

Tutors: Ege Cem Kirci, Roland Meier

Supervisor: Prof. Dr. Laurent Vanbever

March 2021 to June 2021

Abstract

The Internet was not designed with security and privacy in mind. Over the last couple of years, specific protocols and protocol extensions have been introduced to patch the Internet architecture's security and privacy weaknesses. To name a few, HTTPS [14], encrypted DNS (DNS over TLS [25] or DNS over HTTPS [20]), and encrypted SNI [15] (an extension to TLS protocol) focus on making the Internet a more private place by encrypting the sensitive parts of the packets. The goal of all these privacy-preserving protocols and their extensions is to hide the *metadata* that is otherwise visible to the entities monitoring the Internet traffic. However the IP addresses on the packet headers are still readable for everyone and can be used to get information about the accessed domains in a reversed approach. As a first countermeasure Content Delivery Networks [19] impede eavesdropping on IP addresses because they map several websites to the same IP address, so a monitoring entity can no longer tell directly from the IP which page is accessed. However, we can show that the comparison with previously stored page load fingerprints makes it possible to identify domains with a high accuracy. In this thesis, we were able to create a design that detects domains in less than 5 seconds after the opening of a new session with a correct detection rate of 75.1% when multiple domains have the same primary IP address and 92.4% when a single domain maps to one primary IP address.

For the thesis, we generate a data set that contains the page load fingerprints of roughly 50'000 domains. We use it together with a data plane pipeline design of a switch, which is applied to filter traffic specifically and reduces the packet load that needs to be analysed at an early stage. To compare fingerprints and determine accessed websites we develop an analysing script that works in the control plane. A Barefoot Tofino^{TM} Model works as our target to test the implemented design.

Contents

Chapter 1

Introduction

This chapter introduces our work and lays out the structure of the report. In Section 1.1, we discuss the motivation of our work. In Section 1.2, we briefly explain the tasks we cover in our thesis as well as the outlined goals. Finally, Section 1.3 describes how the report is organized.

1.1 Motivation

Since there is no protocol until today, that enables encryption for IP addresses in the header of packets, this poses a privacy weakness that provides attackpoints. It is known from previous papers like Hoang, Niaki, Gill and Polychronakis, 2021 [17] that website page load fingerprinting is a successful way to identify accessed domains, by comparing the IP addresses of the opened sessions with the ones in a pre-populated dataset. Studies show that the theoretical success rate of this kind of attacks can be up to 95%, Patil and Borisov, 2019 [22].

This opens the question how well such an attack would work in a real life scenario and how fast a correct decision could be made. If it is possible to identify accessed domains during the page load this would pose a ground stone for specific blocking of websites and for efficient data mining, even if encryption protocols hide the metadata of packets. The motivation of this thesis is, that we can better assess the risk and efficiency of such an attack. This would make it easier for us to better classify its threat to privacy and to estimate how important it is to find countermeasures.

1.2 Task and Goals

The goal of the thesis is to find out, whether a real time fingerprinting approach is possible with functionality in the switch data plane and evaluation in the control plane. We want to find out how accurate and how fast this kind of attack works and if it is manageable in terms of packet load and data processing. Here we want to take advantage of the speed and early packet processing of the data plane pipeline.
The thesis is divided into several different tasks:

1. **Data Crawling**: The starting point of our work is the acquiring of a reliable dataset. The goal is to get a set that is big enough and thus contains a lot of websites which map onto the same IP addresses. This enables us to test if the fingerprinting approach works or if the primary IP address alone distinguishes the different websites. Another necessity in this task is to get enough data from the page loads, so that a reliable fingerprint can be acquired. This makes it possible to test the $P4_{16}$ code design in a model environment later on.

2. **P4$_{16}$ Code Design**: To create an environment that is close to real life application we need to implement a data plane pipeline on a Tofino^{TM} model in such a way, that this functionality can be used to forward packets to the control plane when it is reasoned. There the incoming packets should undergo continual evaluation, in order to make a good prediction about the website that is accessed at the moment by a given client. This model needs to be expanded to be able to detect traffic from different clients at the same time and consecutive accesses from the same client. One goal is to then find a good tradeoff between fast detection and accuracy.

3. **Control Plane Design:** We want to make continual analysis of the packets that we receive in the control plane possible. For this, we need to implement a Python script that processes incoming IP addresses and compares the so assembled fingerprints with the ones in our data set.

4. **Testing and Evaluation**: The goal of the testing and evaluation part is to provide a scenario that resembles with high accuracy a real life condition. We then want to make a good prediction how well an owner of a switch, who can monitor the traffic, could find out which client is visiting which website at the moment.

1.3 Overview

In Chapter 2, we discuss all background knowledge that is crucial for understanding this thesis. We explain different encryption protocols as well as content delivery networks, bloom filters and fingerprints. In the following Chapter 3, we describe how we acquired our data set and how it is composed. Then we explain our data plane pipeline approach and our fingerprint analysing script in detail. Chapter 4 gives an insight on our results as well as on some interpretations. We then lay out our ideas for future improvements and approaches in Chapter 5. Finally, Chapter 6 gives a short summary and concludes our work.

Chapter 2

Background and Related Work

In this chapter, we describe the concepts that are necessary to understand when reading the thesis, as well as previous work that is related to our approach. In Section 2.1 we explain the principles of TLS, HTTPS, EDNS, SNI, CDN, fingerprints and bloom filters. In Section 2.2 we discuss similarities and assumptions that we can make from related papers.

2.1 Background

2.1.1 Encryption Protocols for Internet Traffic

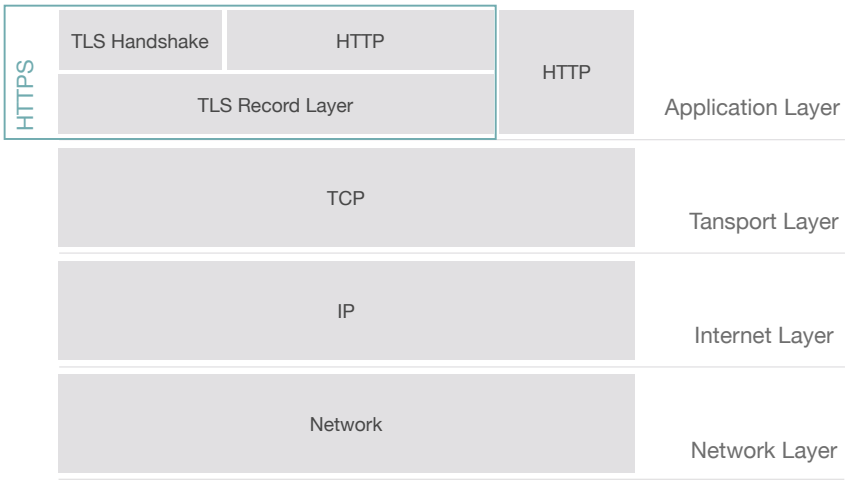


Figure 2.1: Usage of the TLS protocol to encrypt HTTP traffic, together known as HTTPS. Note that the TLS Handshake is not encrypted and HTTP traffic without TLS can coexist with HTTPS.

TLS - Transport Layer Security: TLS [13] is a protocol that provides secure communication over a computer network. It starts each session with a handshake and an exchange of encryption keys, such that a safe data transport is possible. TLS is usually implemented on top of the transport layer and is used to encrypt metadata in application layer protocols such as HTTP or Email.

HTTPS - Hypertext Transfer Protocol Secure: In the past years we have a drastic growth in the usage of HTTP [24] that is transferred over TLS. This so called HTTPS [14] ensures that the

plain text of HTTP is encrypted and can no longer be read by eavesdroppers. Although HTTPS improves security and privacy of network clients it can not fully protect all data that is exchanged. Until this day a lot of DNS queries, as well as the Server Name Indication field (SNI) in the header and all IP addresses, are still in plain text.

EDNS - Encrypted Domain Name System: Since plain text DNS [21] requests are still a high risk for privacy attacks, multiple protocols have been implemented to close this gap. **DoT** [25] (DNS over TLS) and **DoH** [20] (DNS over HTTPS) have been known for quite a long time, but a few providers and browsers managed to introduce these protocols only some years ago to the public. Google for example started in 2018 to integrate DoT in their services. [3] Both protocols work with an encrypted channel to transfer the DNS queries.

SNI [12] (Server Name Indication) was used as a workaround for name-based virtual hosting servers to co-host many websites that support HTTPS. During the TLS Handshake the the SNI field in the header tells the specific website the request belongs to. This functionality is needed such that the server can respond with the respective TLS certificate for the given domain name. Until TLS 1.2 this step takes place before the actual handshake, so it is unencrypted and readable for eavesdroppers. TLS 1.3 intends to solve this problem by introducing an encryption technique of the SNI field - **ESNI** [15]. This, however, is only a real privacy and security improvement if it is used with DoT or DoH such that the exchange of keys can happen in a secure manner. Since the two protocols are still not the standard in internet communication, ESNI is also merely on the rise.

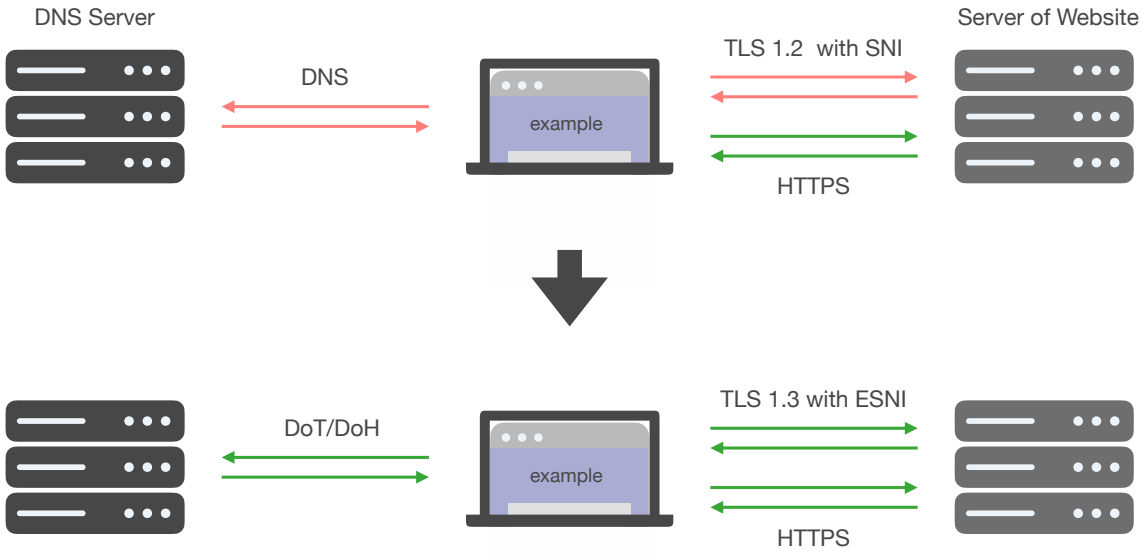


Figure 2.2: Transition from unencrypted (red arrows) to fully encrypted (green arrows) with the usage of the new protocols TLS 1.3 and DNS over TLS or DNS over HTTPS.

If all three protocols are fully integrated into the internet traffic it impedes privacy attacks. In this thesis, we evaluate the case that all clients use EDNS such that we can only read the IP addresses to make a prediction about the domain name.

2.1.2 CDN - Content Delivery Network

Content Delivery Networks [19] are proxy servers that are geographically distributed. They are provided in order to make low latency and high availability all over the world possible. CDN providers host a lot of different websites on their servers, several of them can then be disguised behind one IP address. The SNI field tells the respective website that is requested. This means instead of having a DNS request and a corresponding IP address on the exact website, the IP address which we get from the DNS request belongs to the CDN which makes its own domain to data mapping to get the requested website. Like this, it is not possible anymore to make a reverse lookup of an IP address to be completely sure which website got accessed.

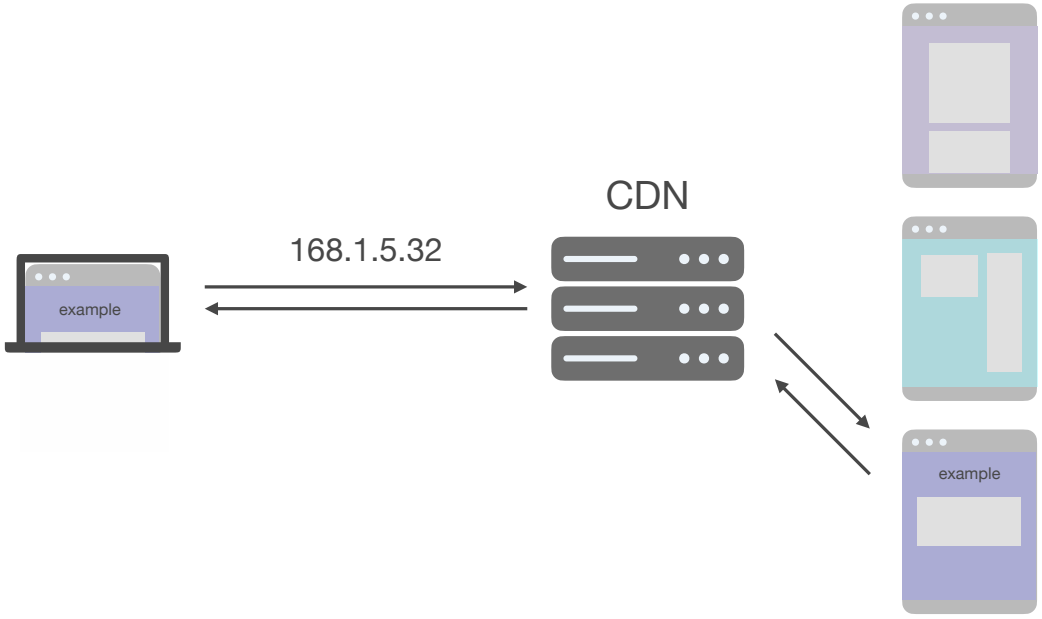


Figure 2.3: CDNs hide different websites behind the same IP address. This makes the websites harder to identify when eavesdropping on the IP addresses of packets.

Additionally to this disguise it is possible that CDNs also map their websites to a set of different IP addresses. This is applied to distribute a high rate of requests to several servers for better load balancing. So it is possible that the same website we accessed a couple of minutes before, now uses another IP address. This is also true if we access websites from different places on earth. In this case, our requests are sent to servers that are geographically near to our current position.

2.1.3 Fingerprints

As a fingerprint of a website, we understand the set of IP addresses where content is loaded from during a page load, in combination with the primary IP to which the main domain belongs to. Normally if we access a website the *index.html* script is located on a server with a specific IP address, the *primary IP*. But additional scripts, pictures and other functionalities may be located on different servers. So when we try to load a web page, the initial HTML document gets parsed and we automatically open different sessions to other IP addresses to load all the content. This fingerprint is unique in many cases since the number of different combinations is endless. Therefore it should be possible to determine a specific domain in a reversed manner if we can collect its fingerprint, even if the primary IP is not only used for this website. As mentioned in Section 2.1.1, the IP address in the header of packets is still in plain text and can be read by everyone who can monitor the traffic. The attack in which one figures out the targeted website by previously storing and then evaluating the fingerprint is therefore called **Website Fingerprinting** [28].

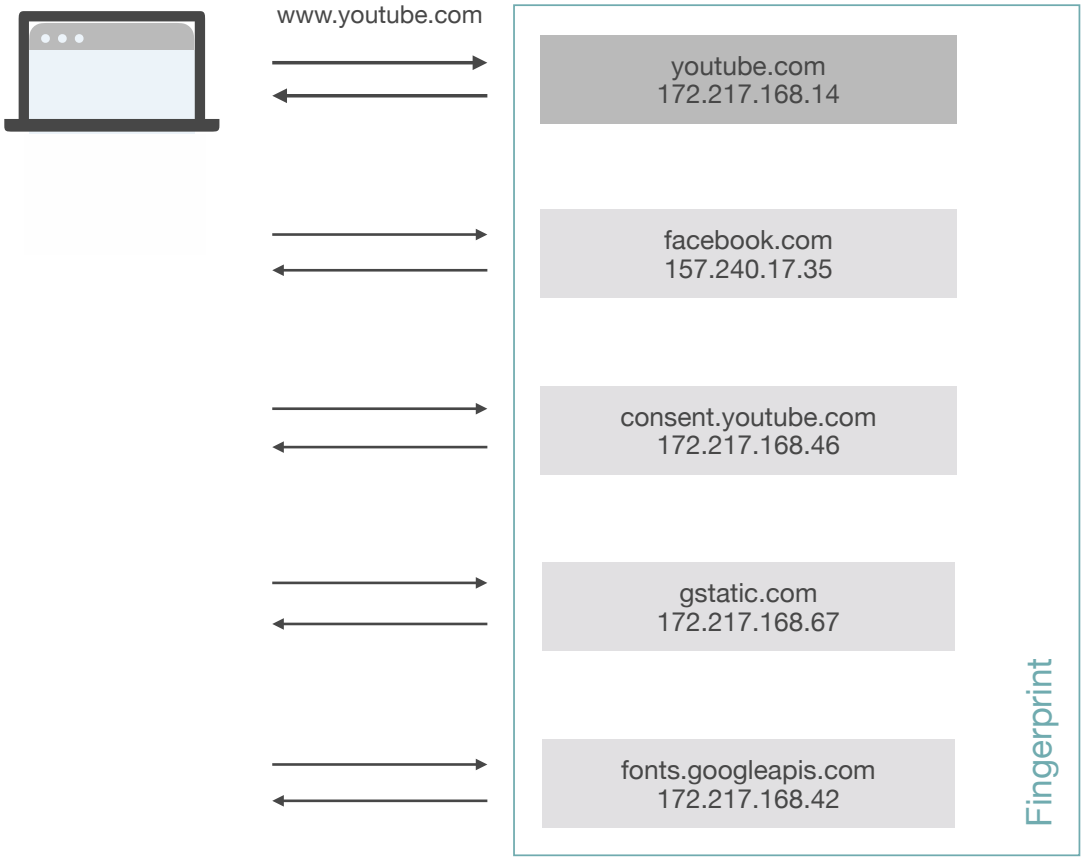


Figure 2.4: Example fingerprint during a website access of *youtube.com*, with the primary IP marked in darker grey and the additional loaded resources in lighter grey. Note that as a subresource *facebook.com* is accessed which is by itself also a main domain. This shows the difficulty of identifying the right main domain when doing page load fingerprinting attacks.

2.1.4 Bloom Filter

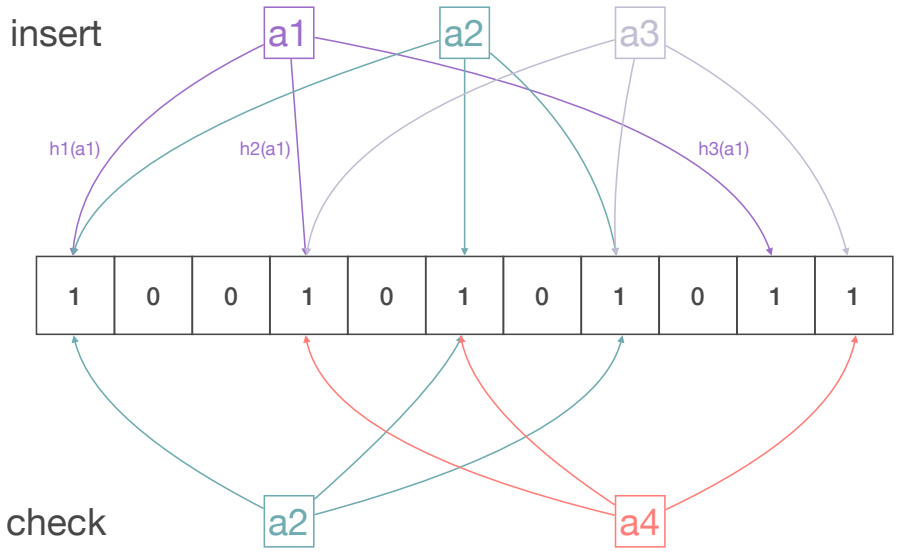


Figure 2.5: Insert of three values (*a1-a3*) with three different hash functions (*h1-h3*) into a bloom filter and check of two values (*a2* and *a4*) with a false positive result of *a4*.

If one wants to be able to check in a space-efficient, probabilistic way if an element is a member of a set, a good solution is to use bloom filters. The concept is that one hashes every one of the N elements with K different hash algorithms to an array of the size M. The places in the array where the element hashes to, the value is set to one. To check if an element is present, the same hash algorithms are applied and the respective place in the array is checked, if they return one, the element is most certainly here. If they return 0, it is sure that they are not. Elements can be added to the set, but for the normal bloom filter they can not be removed. Bloom filters make the most sense if in the application it is appreciated to have no false negative values because if the element is in the set it will always match. The rate of false positive values behaves according to the formula:

$$FPR = (1 - (1 - 1/M)^{K*N})^K \tag{2.1}$$

Especially on switches, routers and other electronic devices with limited storage space, it makes sense to translate large sets to bloom filters. Also, the access speed improves, the search complexity reduces from *O(n)* to *O(1)* [11].

In this thesis, we constructed bloom filters as P4 registers in the data plane. The inserted elements were IP addresses that got hashed by 6 different algorithms onto the register. Like that we can filter the whole network traffic to get only packets from IP addresses that are important to the analysis. It is an optimal situation to use bloom filters since we use a lot of data and never have to delete an entry.

2.1.5 P4$_{16}$ - Barefoot Tofino™ Model

P4$_{16}$ [8] is a language for programming network devices, specifically to program the data plane of a target. All control plane functionalities we need to implement in another language. P4$_{16}$ is built upon the earlier versions of the language P4 and P4$_{14}$. It is used to define on a high level how packets are processed while passing a switch. In this thesis, the P4$_{16}$ code works on top of a **Tofino™ Native Architecture (TNA)** [9] which is a type of a Protocol Independent Switching Architecture [18] (PISA). The target in our implementation is a **Barefoot Tofino™ Model**, since a real hardware switch would be out of scope of the thesis.

2.2 Related Work

Several papers studied the relevance and possibilities of website fingerprinting before. Especially the fingerprint gathering and analysis part of this thesis is very closely related to earlier work from Patil and Borisov, 2019 [22], Hoang, Niaki, Gill and Polychronakis, 2021 [17] as well as from Hayes and Danezis [16]. There are different aspects where we can directly derive findings to our thesis and make assumptions how this would influence future aspects of our work:

Time Dependency: Hoang, Niaki, Gill and Polychronakis 2021 [17] included longitudinal dependencies in their website fingerprinting study. They found, that after two months the successful identification rate decreases from 91% to 70% with the same data set. Since we use a very similar approach in gathering fingerprint information, equal assumptions can be made about our success rate, even if we did not include different sets at different times in our study.

Relevance of Caching: Since caching would result in less sessions that are opened and less packets with trackable IP addresses that are exchanged, the success of website fingerprinting depends on it. Earlier studies from Hoang, Niaki, Gill and Polychronakis, 2021 [17] had a success rate of their website fingerprinting technique with caching of around 80%, compared to one without of 91%. There is no reason why this would behave differently in our model. In our evaluation, however, we made sure that caching is disabled and did not focus on its impact.

Place Dependency: In our study, we accessed all target websites from the same vantage point at ETH in Zürich. But the success of website fingerprinting depends also on the place where different DNS requests were made. Since they deliver different IP addresses depending on where you want to access a website, it would be crucial to be at the same place as the host who is getting observed. But like Patil and Borisov, 2019 [22] state in their work, if the whereabouts of a host are known it is easy to simulate the responses to their DNS requests with tools like zdns [7].

Ad Blocking: Hoang, Niaki, Gill and Polychronakis, 2021 [17] studied the effect of ad blocking on website fingerprinting with help of the Brave Browser [1]. They found, that the efficiency of their fingerprinting technique dropped from 91% to 76%. It needs to be said, that their approach on fingerprinting relies substantially on the order of the access to sub-resources. Since ad blocking considerably affects the order, they did a non ordered approach as well where the efficiency dropped only to 80%. This would better match the fingerprinting approach that was made in this thesis. So it could be assumed that the usage of ad blockers would affect our approach in a similar way.

Chapter 3

Design

This chapter lays out the overall design of our website page load fingerprinting approach. Section 3.1 explains how we gather our data set and which previous adjustments are necessary to make testing possible. In Section 3.2, we describe in detail our data plane approach and what functionalities we implement. Finally, Section 3.3 gives an insight into how the detected packets are processed in the control plane.

3.1 Data Set and Preprocessing

To be able to access websites in an automated fashion it is necessary to implement a web crawler. For this purpose, we developed a Python [4] script which works on the basis of Chrome Driver [2]. This is an open source tool that implements the WebDriver standard for Google Chrome. It gives us the possibility to automatically open websites one after another, on a virtual machine, in a headless manner. During the data crawl we deactivate sandboxing, caching, cookies and ad blocking in our script.

Page Load Fingerprint Collecting: First, we acquire 50'000 domain names from the list of the million most popular domains in the world wide web on the Majestic website [6]. With the Selenium Wire [5] module our program opens consecutive subpages of the Majestic Million site where the mentioned list is located. It parses the HTML [26] script for the domain names, stores them in an array and writes it to a JSON [27] file.[1] Then each of the 50'000 front pages of the domains are opened by the Chrome Driver and load for a maximum of 5 seconds with no pause between the access of two following websites. If the page load is shorter, the driver stops earlier and goes to the next one. Sites, where the main domain can not be resolved, are not respected in further crawling. This leads to a total of 48'497 domains, which are all accessed three separate times with an interval of approximately three days. We do this to have a bigger chance of getting the correct primary IP address, if a website maps to multiple ones. Of each website, all IP addresses from opened sessions, as well as their size and timestamp are automatically stored in *json* format by using the functionalities of the Selenium Wire tool. This includes multiple primary IPs for the same main domain with their respective fingerprint if they are alternating in the repeated accesses.

Test Data Collection: In order to be able to simulate traffic, additionally to the stored requests during each website access, the script listens to all incoming and outgoing packets with the

[1] As an additional functionality we included an option in our script to leave it open that domain names can also be read from a normal text file.

Scapy [23] module on an additional thread. We then store these packets for every targeted website as an own packet capture (pcap) file.

Since the unrepeated data crawl takes approximately 56 hours already, we made several different attempts to speed up the process. Threading increases the speed of the crawler by 300%, but we see that the performance is limited due to the single web interface. It is no longer possible to assign packets to their respective domains when the access of following websites overlaps. This makes it impossible for us to prepare the pcaps for testing. However several threads can be used for repetitions where the simultaneous catching of pcaps is no longer necessary. Another very limiting factor is the slowness of some page loads.

Preprocessing: By using the stored page load fingerprints our script filters the pcap data to the essential traffic, that also got recognized by Selenium Wire as a valid request from the specific site. After that, we change the client IP of the pcap to one of n randomly generated client IPs from a similar prefix as the original one. This makes it possible for us to simulate the traffic to come from different hosts, but to make sure that the host address does not unintentionally match an accessed IP of a website. To integrate real life conditions into our testing, we write the timestamp of the original packet to the MAC destination field, such that we can read it in the data plane and interpret it like it is real traffic. This is necessary since sending packets in the Tofino model is not as fast as actual traffic on the web. These pcap files are then ready to use for traffic replay in the switch and analysing in the control plane.

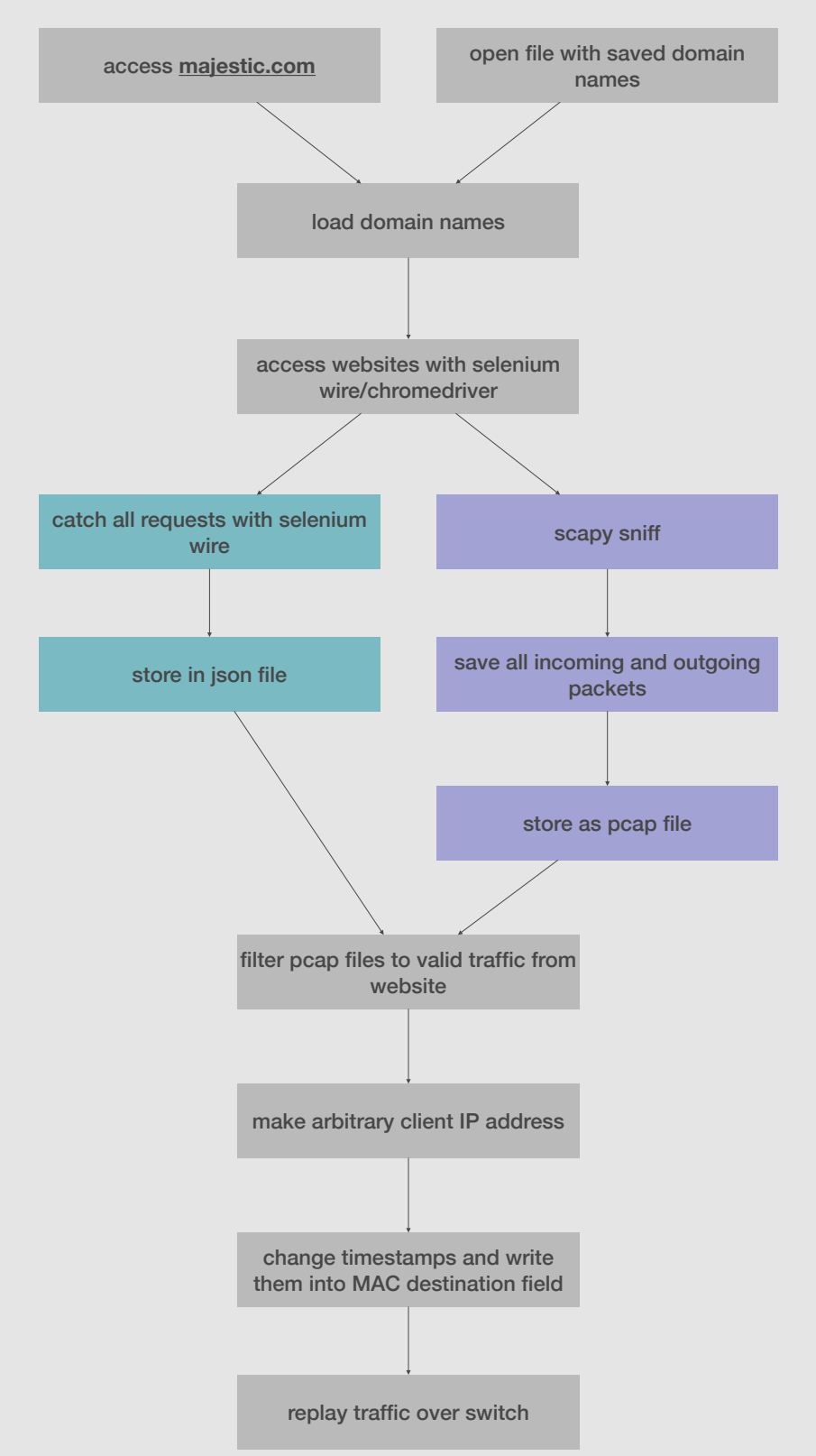
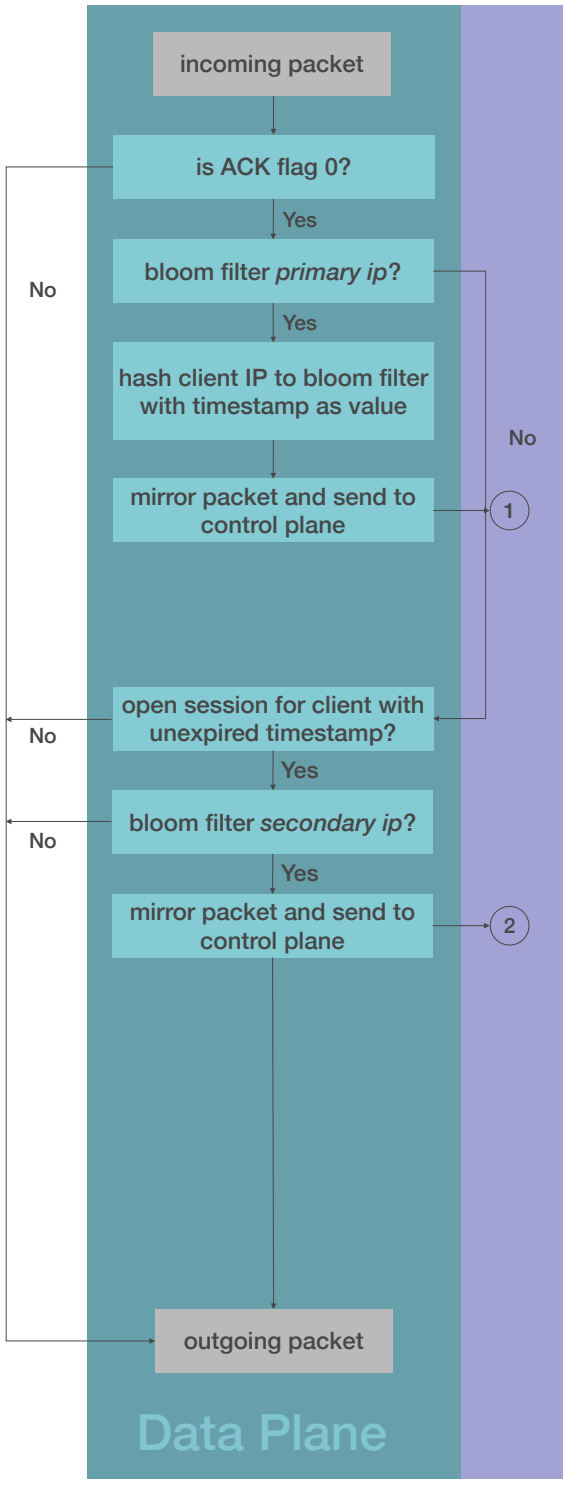


Figure 3.1: The workflow of data collection, processing and traffic replay. First, we download the domain list from the Majestic Million website. We then create the fingerprint data set at the same time as we collect the pcap files for testing. The latter get processed such that they better match real life scenarios.

3.2 Data Plane Design



Filtering Based on Header Fields: The goal of the functionality in the data plane is to reduce the traffic that is sent to the control plane to a processable amount, such that it is more easy to analyse and interpret it. For this to be fulfilled, we filter the traffic with the help of a $P4_{16}$ script, such that it can only mirror and forward valid IP/TCP packets. The traffic is further filtered to all the packets that have the *TCP Flag ACK* set to 0. This means that the packet is the first one from a new session to a server. This makes it possible that we only forward one packet from each flow, which reduces the amount of packets to analyze to about 3.3% of the original amount.[a]

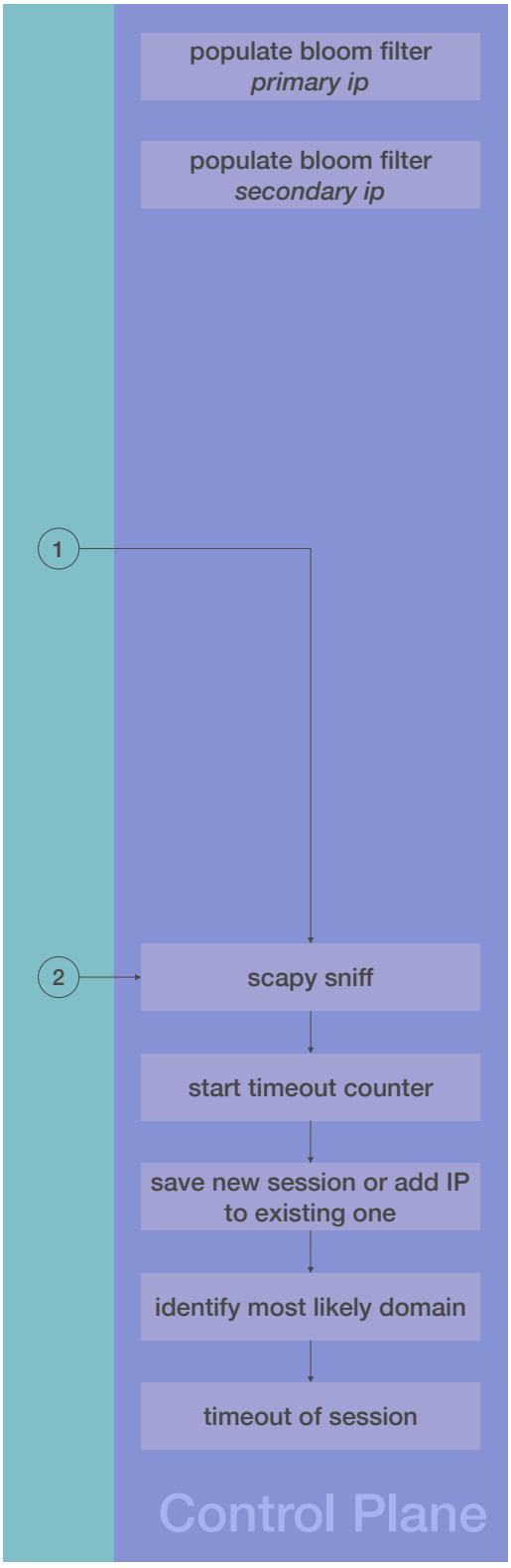
Bloom Filters: If a packet passes the first two conditions, we hash its destination IP (which represents the IP of the website server) to a bloom filter, which checks if this IP address is stored as a primary IP from our web crawl. If this is true, we mirror the packet and forward the clone. Additionally, we hash the source IP (the IP of the host who wants to access the website) to a bloom filter, which represents all clients that have a session that is open. To improve this functionality the timestamp of the first packet of the specific client session is written into the register such that we can see how long the session is already open. When the next packet from the same client arrives, we check the time difference to the timestamp, if it is bigger than a specified timeout the session counts as expired and the packet is not forwarded.

If the destination IP hash did not match the primary IP bloom filter but a session for the given client IP is already open and not expired, we hash the destination IP to the secondary IP bloom filter. There all the values for all known IP addresses in our fingerprints are stored. If it matches on this bloom filter and satisfies the other conditions, we mirror it and forward the clone to the control plane as well.

[a] Mean of 100 pcap files

3.3 Fingerprint Analysis in the Control Plane

Population of Bloom Filters and Mirroring Functionality: Registers and tables that are used in the data plane are populated with a controller. We need to do this for the crawled data of the primary IPs and the secondary IPs, to construct the respective bloom filters before we can test sending packets. We also need to specify, with the help of our controller, how the packets that undergo mirroring according to our data plane logic get forwarded.



Processing the Incoming Packets: The Python analyser in the control plane constantly sniffs for incoming packets. As soon as there is one on the respective interface and it comes from a known primary IP, the analyser opens a new session for the given host and starts a timeout counter. With all the packets from the same client, it creates a fingerprint and compares it with the fingerprints of all the domains that come into question. For the first two seconds of a new session, the analyser listens to additional primary IPs, such that websites that first load an additional resource like an ad manager do not get mistaken as one. If there is still only one domain option after 2 seconds, the analyser immediately decides on this domain. Otherwise, as soon as the similarity between the fetched fingerprint and a given fingerprint from the data crawl reaches a threshold value, the domain is marked as most likely and is written to a file.

We compute the **threshold** value by finding the maximal similarity between two fingerprints that have the same primary IP. It is shown in Figure 4.3, that in our main dataset the maximum is 100%. In smaller datasets it is lower which results in a decrease of the average decision time.

If the script can not make a decision on a domain because the similarity never reaches the threshold, it automatically closes a session after a specified decision time. It then decides on the domain that has the most similar fingerprint to the fetched one, in terms of same IPs that are accessed and similarity of the fingerprint size.

Chapter 4

Evaluation

This chapter attempts to give an overview of all findings that we could make during the evaluation of the data set and our design. In Section 4.1, we explain the analysis of our crawled data set, which gives us some perspective on the necessity of the page load fingerprinting approach. Section 4.2 shows the results of the accuracy and speed of page load fingerprinting with our data plane design.

4.1 Fingerprint Analysis

To be able to test our data later on in a sensible manner and to check if fingerprinting makes sense, we first needed to evaluate on the dataset. We found that of the 48'497 tested domains, 33'904 (roughly 70%) have a one-to-one domain to IP mapping, as you can see in Figure 4.1. The anonymity set size is defined as the number of domains that map to the same IP, so one-to-one mapped domains represent the highest bar on the left. These domains should be recognised solely by knowing the correct primary IP address. The other 30% have more than one domain that maps to the same IP address, here website page load fingerprinting is necessary and could be an advantage. The highest amount of websites that match to the same IP is 133. We can assume that the number of n-to-one mapped domains gets even bigger if more data is crawled since Patil and Borisov, 2019 [22] observed more than 50% of n-to-one mapped IP addresses with a data set of roughly a million domains. This shows us that it makes sense to use the intended approach and distinguish between the websites by using fingerprinting.

The analysis of the fingerprint size reveals that on average every website fetches, additionally to their primary IP, 6.58 IP addresses. The biggest fingerprint that we found has the size of 50 (including the primary IP). However, 4728 websites fetch no additional resources that are located on servers with different IP addresses. 1667 of them are from domains that do not map to a unique IP address, which means it is a major challenge to identify their correct domain by website fingerprinting because they do not technically have one.

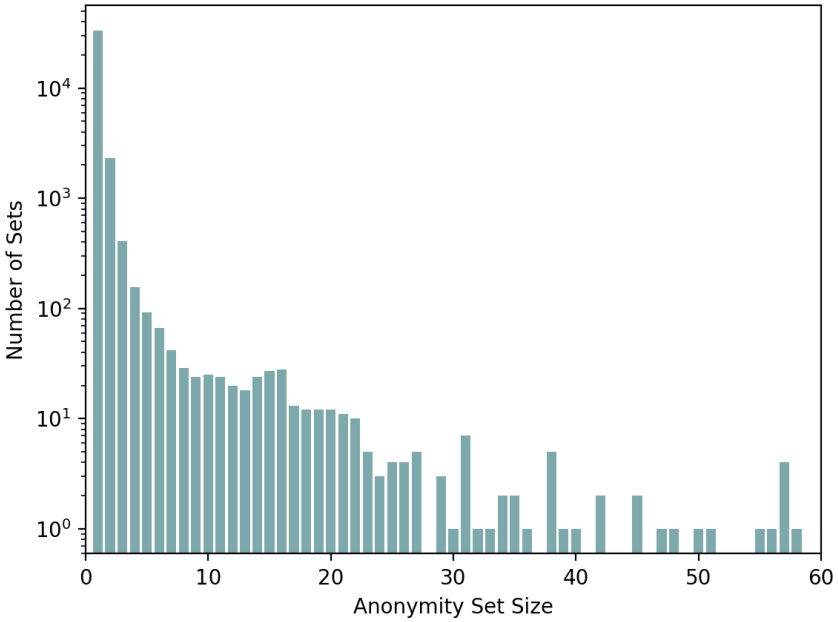


Figure 4.1: As the anonymity set size we understand the number of domains that map to the same IP address. We can see in the figure, that around 70% of the accessed domains have a unique primary IP. The other 30% share their IP address with up to 132 other domains from our data set.

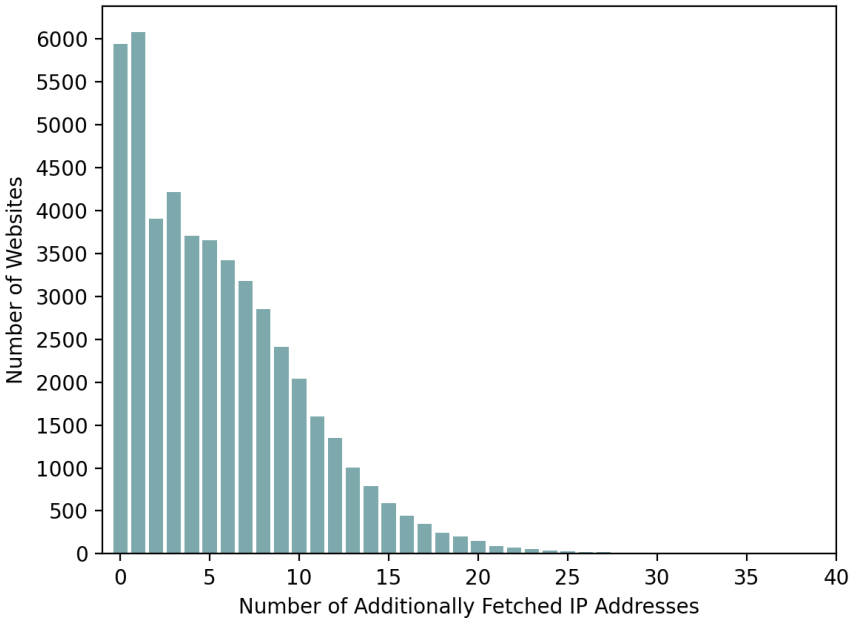


Figure 4.2: In average websites fetch additionally 6.58 IP address, however 4728 domains do not fetch any subresources from servers with other IPs at all.

We now analyse the similarity of fingerprints with the same primary IP according to Equation 4.1. It can be stated, that half of them have a similarity of smaller or equal to 6.5% (Figure 4.3). These should be more straightforward to distinguish and should not be mistaken for the wrong domain if we have stored the right primary IP in our dataset. However, according to our data set 3.5% of all found fingerprints that share their primary IP address with at least one other domain have the exact same fingerprint as another one. This implies solely that the same IP addresses were accessed, the websites could still be distinguished by differentiating in access duration, order of opened sessions or amount of loaded content from each address. This approach was not pursued in our thesis but was done in more detailed fingerprinting studies like Hayes and Danezis, 2016 [16].

$$Similarity = \frac{Fingerprint\,A \cap Fingerprint\,B}{Fingerprint\,A \cup Fingerprint\,B} \tag{4.1}$$

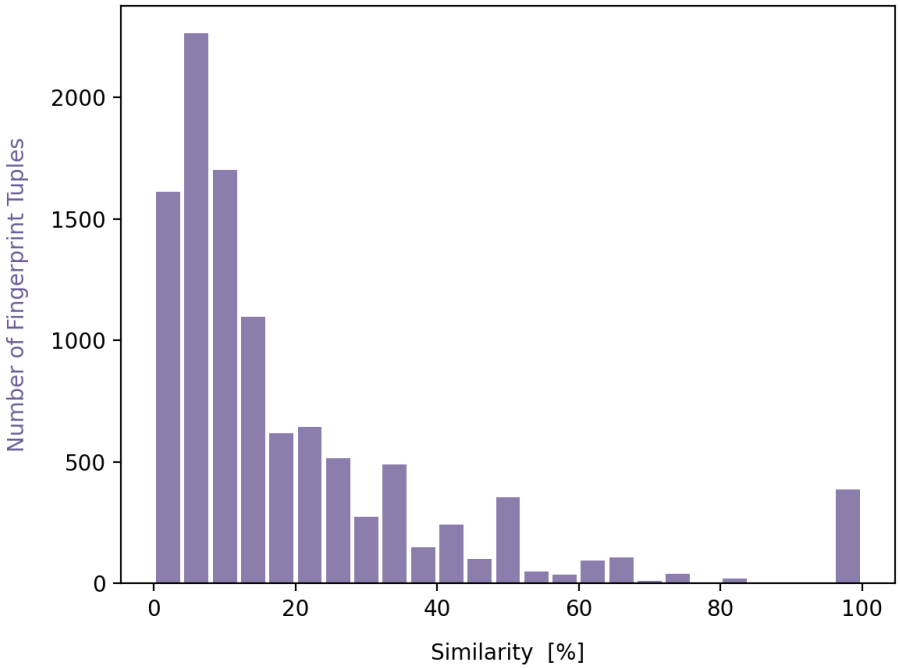


Figure 4.3: This figure shows the similarity of two fingerprints that belong to the same primary IP address. We can see that half of the fingerprints have a similarity that is smaller than 6.5%. However, 3.5% of the fingerprints that belong to a n-to-one mapped domain have the exact same fingerprint as another one.

As described in Section 3 the domains were accessed multiple times such that we are better able to resolve domains that map to alternating IP addresses. 64% of all the domains that were accessed had every time the same IP address, 28% of them had two different ones and 8% had three. This makes future primary IP addresses in real life scenarios less predictable, impedes fast data collection and decreases the correct page load fingerprinting rate.

4.2 Real Time Domain Catching

In this section we present and discuss our performance results from sending pcap files over the data plane pipeline and analysing them in the control plane.

4.2.1 Pre-Evaluation to Determine Parameters

We studied the optimal parameters to find a good tradeoff between being as accurate as possible in deciding on the right domains, while not suffering big losses in terms of speed and false positive decisions in a smaller pre-evaluation. For that, we used a set of 1000 domains split into two distinct sets, one contained all one-to-one mapped domains and one contained all n-to-one mapped domains. Note that for the pre-evaluation we accessed every domain only a single time to populate the data set.

Forced Decision Time: As forced decision time we understand the specified time from the opening of a session until the script automatically closes a client session and evaluates which domain has the highest similarity to the populated fingerprint. Since we limited crawling to a maximum of five seconds per client session, we decided to vary the forced decision time between 3.5 and 5 seconds, see Figure 4.4 and 4.5. We found that the percentage of correct domains does not vary more than 2.5% over this timespan, but the number of false positive decisions increases drastically if we reduce the forced decision time. This is due to the fact that clients open several sessions to stored primary IPs while accessing a website. If the decision is made after 3 seconds, it is probable that an additional falsely session gets opened, since a lot of pcaps replay traffic that our script sensed over a timespan up to 5 seconds. We found that a forced decision time of **4.75 seconds** gives us good results in terms of correctly interpreted domains while keeping the false positive decisions low.

Arrival Time Span for the Primary IP: While doing analysis on incorrectly detected websites with a unique primary IP, we could see that the main reasons for wrong detection is an alternating IP with a non overlapping set of IPs stored in our data set and a too early decision on the wrong primary IP. We could reduce the influence of the second problem by enabling the script for a defined time, we call this primary decision time, to interpret incoming primary IPs of a session as a possible option. While doing tests on different primary decision times we found that the longer we wait, the more websites get correctly identified. But obviously, the average decision time increased, so we set the primary decision time to **2.25 seconds**.

Number of Page Loads: Like we mentioned above, a lot of websites had the wrong primary IP stored. We address this problem by triple accessing all domains for the performance evaluation of our main data set. This number is limited due to the long duration of the webcrawling. Since we did not intend to make a longitudinal study, we wanted to collect alternating primary IP addresses from approximately the same point in time.

We can see in the pre-evaluation that n-to-one mapped domains, that rely on website page load fingerprinting, have a chance of around 78% of being recognized (Figure 4.4). Whereas one-to-one mapped domains have an approximately 10% higher detection rate (Figure 4.5). In terms of the decision time, we can expect an average of around 4 seconds. False positive decision are relatively common, we did not further pursue a solution to this since the importance of reducing the false positive rate is strongly dependent on the area of application and it would make sense to address this problem in context to it.

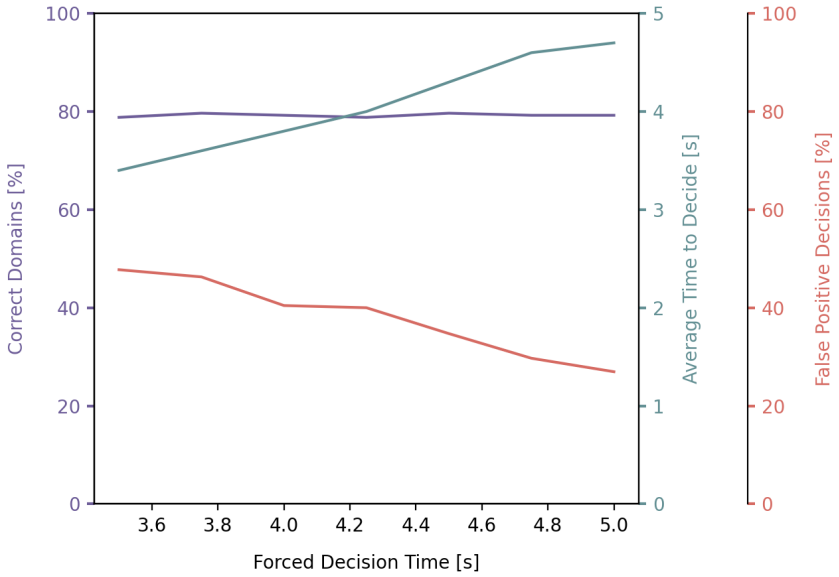


Figure 4.4: Influence of different forced decision times with **n websites mapped to one IP address**. Whereas the percentage of correct domains stays nearly the same when we reduce the forced decision time, the rate of false positive decisions increases drastically while the average decision time decreases.

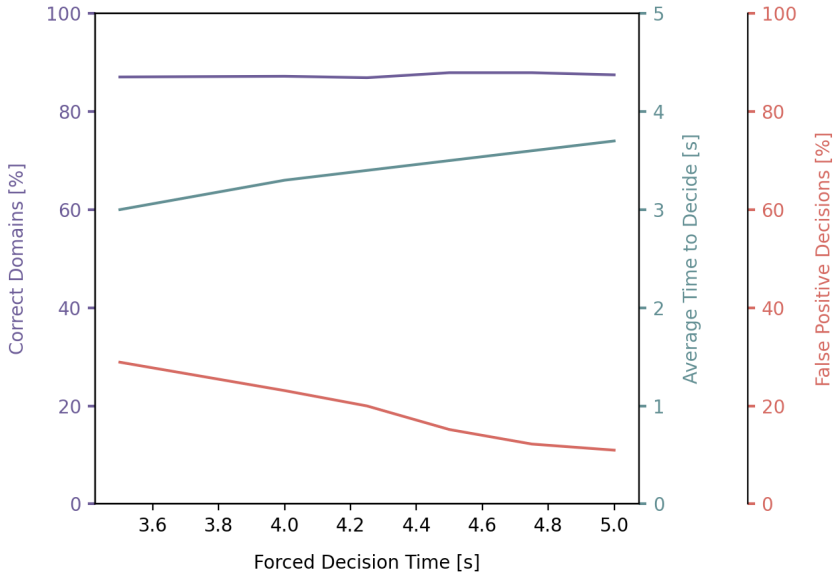


Figure 4.5: Influence of different session timeouts with **one website mapped to one IP address**. It can be seen that the correct detection rate is higher than with n-to-n mapped domains, while the average decision time is lower.

4.2.2 Performance Evaluation

As we saw in Section 4.1 a large part of our crawled domains is one-to-one mapped. Since we can clearly determine the error sources that can happen when trying to identify them, and because they are less dependent on page load fingerprinting, we decided to test only a subset of these domains. We can expect a similar behaviour for the remaining ones.
Therefore we tested 1000 pcap files and analysed them with our main data set with triple access and with the data set with only one access. From these thousand domains, 936 have a non-empty pcap file. With the triple access data set out 865 get correctly identified by our analysing script, this results in a correct detection rate of 92.4%. Which is slightly higher than we predicted in the pre-evaluation. Additional to the 865 correctly detected domains, the script decided on 169 wrong ones. The average time to make a decision was 4.03 seconds. This is an approximately 0.6% higher success rate than without triple access, where the script detected 859 correct domains. In this evaluation, the average decision time was 4.06 seconds. From these results, we can derive that even slightly higher correct decision rates could be possible if the domains are accessed further times.

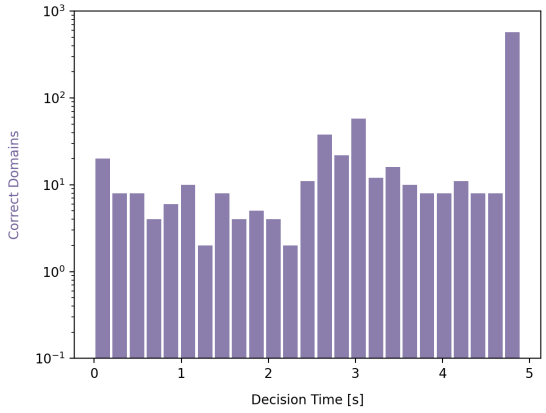


Figure 4.6: Decision time of 1000 **one-to-one** mapped domains. We see here that one-to-one mapping result in more early detection. However most domains are decided at the forced decision time.

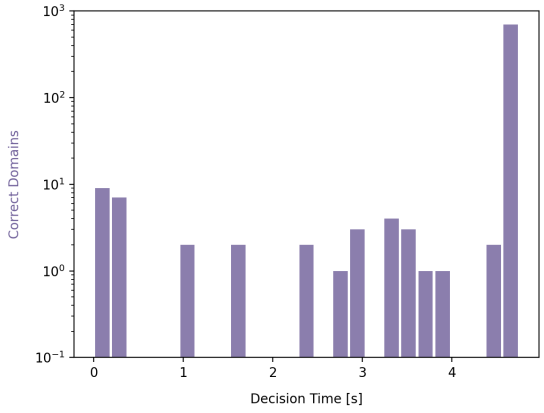


Figure 4.7: Decision time of 1000 **n-to-one** mapped domains. We can see in this figure that due to the prolonged time to decide on the primary IP and the threshold of 1, early detection in n-to-one mapped domains is rare.

Testing of 16'478 n-to-one mapped pages resulted in 15'763 non-empty pcap files that were sent. From them, 11'838 were correctly identified domains. This is a detection rate of around 75.1%. The time to accurately identify a domain took us 4.52 seconds on average. We assume that the slightly lower detection rate and the higher average decision time come from the fact, that the script had to decide between more options and could make less early decisions. If we look at all targeted websites in our data set combined (n-to-one as well as one-to-one), we can record an overall correct detection rate of **87.2%**.

Table 4.1: Performance evaluation of our main data set. While the correct detection rate in one-to-one mapped domains stays high compared to the pre-evaluation, we had some loss in the one of n-to-one mapped domains.

Mapping	**Correct Domains** [%]	**False Positive Decisions** [%]	**Average Decision Time** [s]
One-to-One	92.4	16.3	4.03
N-to-One	75.1	28.1	4.52

4.2.3 Discussion

We can see that n-to-one mapped domains facilitate privacy when exposed to website page load attacks, compared to one-to-one mapped ones. This supports the assumption that CDNs represent an advantage in terms of user privacy. We can therefore say that for privacy enhancing it would be preferable to host more websites over CDNs.

However, even if we now think of a scenario where all websites are hosted on CDNs we can still see that a correct identification rate of 75.1% poses a risk for the privacy of clients. Especially if we consider, that this attack can be enhanced with more specific data crawling and by using more complex analysing algorithms.

If we now assume the case where the page load fingerprinting attack is utilised for **data mining**, so to store information about website accesses of different clients, the decision time should be short enough to detect all websites. We can assume that the timespan from typing in a domain name until closing a page again takes normally longer than 4.5 seconds. This also if we are curious about the user's interest, it makes more sense to catch the websites they do not close straight again. However we can discuss if it makes sense make data mining in a real time approach because there are not many applications where it is necessary to know the accessed websites right away.

A more conceivable approach would be that we use the page load fingerprinting attack as a tool to **block websites** at real time. Here, a page load of 4.5 seconds before closing would sure be enough to prevent the user from getting all needed information from a website. Preferably it would make more sense to detect at an earlier point in time which website is accessed before the whole page is displayed for several seconds. However, it can be assumed that in a real life scenario with actual web traffic the decision time would take less time since it is now strongly influenced by how fast a page was loaded with the Chrome Driver tool. Also, if we rely on a smaller set of websites we want to block, we can perfect the fingerprints and decide faster on them.

Chapter 5

Outlook

Since the time of the thesis was limited there are still a lot of aspects about website fingerprinting in the data plane that can be studied and improved. We discuss in the following a selection that we think would be worth following:

Hardware application: Our whole work is based on the Tofino model, this means that it is closely related to real life application but it is limited to some point. Especially testing of traffic could be done in a less artificial way. For future work, it would make sense to implement the design directly on the hardware target to verify the results for real scenarios.

Fine tuning of the design and additional testing: In a future study one could apply a more sophisticated fingerprinting analyzing algorithm, this could involve timestamps, packet size, traffic density and many more. With this approach, a more accurate and sensible website fingerprinting could be achieved. Another point that was once intended to solve in this thesis but was discontinued because of difficulties in the artificial testing environment would be to expand the fingerprint analysis to be able to track multiple website accesses at the same time from the same client.

In data crawling several improvements could be made. One could access the websites with more repetitions, which could positively influence the number of correctly recognised primary IP addresses as mentioned in section 4.2. Since we expect for real life applications a more specified need for page load fingerprinting, where the focus lays on fewer interesting websites, it would be easy to access them in advance as many times as it provides additional IP data. To be better able to recognize fingerprints one could also discuss if it makes sense to access subpages of websites additional to their main page. This would more accurately describe real web browsing.

Countermeasures: We now showed that it is nowadays possible to recognize around 88% of the accessed websites in a span of less than five seconds with our design. But with privacy in mind, one should be alarmed by this rate and it would be essential to find possible countermeasures to solve this data leakage in the future. Further encryption, recursive access of resources, more CDN hosting and less outsourced resources could be a starting points to closing this gap.

Comparison with a full control plane approach: To defend our approach to a full control plane analysis combined with a router protocol like NetFlow [10] one could do further investigation on their possibilities in filtering and forwarding. This would make it possible to compare the $P4_{16}$ approach and better confirm its validity.

Chapter 6

Summary

In the range of this thesis, we acquired a data set of roughly 50'000 domains with their respective page load fingerprints. At the same time, we collected and prepared pcap files of the same domains to provide a good testing environment. In order to then filter the produced traffic efficiently while traversing a switch inside a testing environment we implemented a $P4_{16}$ that worked on top of a Barefoot Tofino^{TM} Model in the data plane. As a counterpart in the control plane, we implemented a Python script that works as a real time analyser and collector of page load fingerprints. Like that we were able to decide in less than 5 seconds after the beginning of a page load on a website with an accuracy of at least 75.1%. This shows us that real time page load attacks that can be used for example as a censoring application are theoretically possible and pose a risk for user privacy, even if all metadata in packets is encrypted..

Bibliography

[1] Brave Browser. https://brave.com/de/.

[2] Chrome Driver. https://chromedriver.chromium.org/. (Accessed: Apr 2021).

[3] Google Public DNS now supports DNS-over-TLS. https://security.googleblog.com/2019/01/google-public-dns-now-supports-dns-over.html.

[4] Python. https://www.python.org/. (Accessed: Apr 2021).

[5] Selenium Wire. https://pypi.org/project/selenium-wire/. (Accessed: Apr 2021).

[6] The Majestic Million. https://majestic.com/. (Accessed: Jun 2021).

[7] ZDNS. https://github.com/zmap/zdns.

[8] P416 language specification. https://p4lang.github.io/p4-spec/docs/P4-16-v1.0.0-spec.pdf, 2017. (Accessed: June 2021).

[9] $P4_{16}$ Intel® Tofino Native Architecture. Tech. Rep. 631348-0001, Intel® Corperation, Mar. 2021.

[10] B. Claise. Cisco Systems NetFlow Services. https://datatracker.ietf.org/doc/html/rfc3954.html, 2004. RFC 3954.

[11] Blustein, J., and El-Maazawi, A. Bloom filters - a tutorial, analysis, and survey. Tech. rep., Faculty of Computer Science Dalhousie University, 2002.

[12] D. Eastlake. SNI - Server Name Indication. https://datatracker.ietf.org/doc/html/rfc6066#section-3, 2011. RFC 6066.

[13] D. Eastlake. TLS - Transport Layer Security. https://datatracker.ietf.org/doc/html/rfc5246, 2011. RFC.

[14] E. Rescorla. HTTPS - Hypertext Transport Protocol Secure. https://datatracker.ietf.org/doc/html/rfc2818, 2000. RFC 2818.

[15] Eric Rescorla and Kazuho Oku and Nick Sullivan and Christopher Wood . ESNI. https://datatracker.ietf.org/doc/draft-ietf-tls-esni/, 2021. Draft.

[16] Hayes, J., and Danezis, G. *k-fingerprinting: A Robust Scalable Website Fingerprinting Technique*. University College London, 2016.

[17] Hoang, N. P., Niaki, A. A., Gill, P., and Polychronakis, M. *Domain Name Encryption Is Not Enough: Privacy Leakage via IP-based Website Fingerprinting.* Stony Brook University and University of Massachusetts, Amherst, 2021.

[18] Kim, C. Programming the network data plane: What, how, and why? Slides, Barefoot Networks, pp. 9–12.

[19] L. Peterson and B. Davie and R. van Brandenburg. CDN - Content Delivery Network. `https://datatracker.ietf.org/doc/html/rfc7336`, 2014. RFC 7336.

[20] P. Hoffman and P. McManus. DoH - DNS over HTTP. `https://datatracker.ietf.org/doc/html/rfc8484`, 2018. RFC 8484.

[21] P. Mockapetris. DNS - Domain Name Service. `https://datatracker.ietf.org/doc/html/rfc1035`, 1987. RFC 1035.

[22] Patil, S., and Borisov, N. *What Can You Learn from an IP.* Universitiy of Illinois at Urbana-Champaign, Montreal, QC, Canada, 2019.

[23] Philippe Biondi. Scapy. `https://scapy.net/`, 2021. (Accessed: Apr 2021).

[24] R. Fielding and J. Gettys and J. Mogul and H. Frystyk and L. Masinter and P. Leach and T. Berners-Lee. HTTP - Hypertext Transport Protocol. `https://datatracker.ietf.org/doc/html/rfc2616`, 1999. RFC 2616.

[25] S. Dickinson and D. Gillmor and T. Reddy. DoT - DNS over TLS. `https://datatracker.ietf.org/doc/html/rfc7858`, 2018. RFC.

[26] T. Berners-Lee and D. Connolly. HTML - Hypertext Markup Language. `https://datatracker.ietf.org/doc/html/rfc1866`, 1995. RFC 1866.

[27] T. Bray. JSON - JavaScript Object Notation. `https://datatracker.ietf.org/doc/html/rfc7159`, 2014. RFC 7159.

[28] Website Fingerprinting. Website fingerprinting. `https://datatracker.ietf.org/doc/html/draft-irtf-pearg-website-fingerprinting-00`. (Accessed: Apr 2021).

Appendix A

Appendix

Here, we present an example of a crawled webpage fingerprint.


```
“index”: 9,
"primary_domain": "googletagmanager.com",
"primary_ip": "172.217.168.8",
"domain_list": [
  {
    "domain": "abs.twimg.com",
    "ip": "152.199.21.141",
    "timestamp": "06/09/2021, 08:27:14.161198",
    "content-length (bytes)": "207840"
  },
  {
    "domain": "googletagmanager.com",
    "ip": "172.217.168.8",
    "timestamp": "06/09/2021, 08:27:14.199407",
    "content-length (bytes)": "1555"
  },
  {
    "domain": "www.google.com",
    "ip": "172.217.168.4",
    "timestamp": "06/09/2021, 08:27:14.311784",
    "content-length (bytes)": "6327"
  },
  {
    "domain": "www.google.com",
    "ip": "172.217.168.4",
    "timestamp": "06/09/2021, 08:27:14.340139",
    "content-length (bytes)": "3170"
  },
  {
    "domain": "abs.twimg.com",
    "ip": "152.199.21.141",
    "timestamp": "06/09/2021, 08:27:14.386956",
    "content-length (bytes)": "4668"
  }
]
```


Figure A.1: Example of a crawled fingerprint