TorFA: Tor Fingerprinting Attack

Nathan Dautenhahn Wooyoung Chung George Karavaev Prateek Mittal

University of Illinois at Urbana-Champaign

{dautenh1,wchung8,karavae1,mittal2}@illinois.edu

Abstract

Tor, a privacy enhancing technology, provides anonymous communications by providing several layers of encryption and a complex routing scheme called *onion routing*. The developers of Tor claim that Tor defends against a non-global adversary. We examine methods of breaking Tor's anonymity by employing a local passive traffic analysis attack. Traffic analysis attempt to identify the contents of Web communications by statistically profiling a set of websites, and then comparing an unknown sample of a website download against the profiles.

We introduce a novel method that applies *relative entropy* measurements to classify Tor traffic based upon packet counts. We achieved up to 25% success rate, a $10x$ increase of previous works. We also focus on a real world deployment of our attack, which includes identifying several issues that impede the development of a complete attack. We also provide detailed descriptions of how to make this attack deployable in the real-world.

1 Introduction

Several industries, including banking, commerce, social networking, and cloud computing, are providing end users with the ability to access personal data via the Internet. As the number of online services increases, more personal and private data are being pushed onto the Web, necessitating online identification and authentication. Coupled with the increased usage of Internet services is the fact that the Internet is an extremely open and accessible medium (e.g. wireless hotspots) enabling tracking of a large volume of user data. For example, every search in Google is stored without anonymization for a period of nine months [7]. Mayer-Schönberger [11] suggests that the trend of capturing and storing information will create an environment in which everything a person does will permanently recorded. Under such circumstances, it is important to have a capability to anonymously send and receive information over the Internet without being tracked.

In response to the growing need for online privacy technologies, the Onion Routing project has developed Tor [2], a free anonymization service for Internet communications. Tor accomplishes the goal of end user privacy by employing a scheme called *onion routing* [3], which uses multiple layers of encryption and route redirection to provide anonymous communications for a limited set of web applications (e.g., browser and chat clients). The goal of Tor is to obfuscate an end user's communication such that an adversary cannot associate any Internet request with that user or identify the contents of the user's Internet traffic. Tor uses the term *Onion Router* (OR) to identify routers in the network that provide packet routing, and *Onion Proxy* (OP) to signify a client issuing an Internet request. Tor aims to provide this protection against a non-global adversary that can observe partial traffic in the network, manipulate Tor traffic, enable ORs of its own, and compromise existing ORs.

It has been suggested [2, 8] by the Tor developers that Tor has the potential of being vulnerable to a website fingerprinting attack, which is generally described as a contextual traffic analysis attack [13], or a profiling attack [10]. In a general fingerprinting attack, an adversary creates a signature of a website (based upon what the adversary can view as an eavesdropper) and builds a statistical profile of the website from the observable characteristics of the traffic. The fingerprint is then stored to a database, which is used by the attacker to classify unknown data in order to identify the content a given user is viewing. The primary goal of these attacks is to correlate a given Internet request with the end user who initiated the request, or, in other terms, to attempt to obtain information about the content someone is viewing on the web.

Current statistical profile attacks on encrypted streams are limited in three ways. First, they have made assumptions that make the attacks *infeasible* in the real world. These assumptions allow them to gain successful results only outside the context that an attacker in the real world would have to face. Second, only one of these attacks [4] has been attempted on Tor traffic, and in that case had only a 2.96% average success rate. The developers of these attacks have not provided a strategy for implementing their attacks successfully against Tor traffic [5, 1, 10, 4]. Third, the previous classifiers were correlation metrics that depend upon independence between packet counts, which is not realistic in the real world.

Contributions: This paper introduces TorFA, a novel fingerprinting attack that:

- Uses Kullback-Leibler (K-L) divergence as a method of performing classification of unknown data sources.
- Is the first to perform an-in depth analysis of the lower-level implementation details of Tor from the perspective of an attacker.
- Focuses on real-world implementation and feasibility issues of the fingerprint attack.
- Identifies key future steps to creating a fully implementable fingerprint attack.
- Discusses countermeasures to the Tor attack and primary hurdles of implementing these defense mechanisms.
- Fits the attacker model that Tor claims to successfully defend against, thus identifying key security vulnerabilities in the Tor implementation.
- Improves upon previous attacks [4] obtaining a success rate of 24%.

In this paper we identify the key elements necessary to implement a real world fingerprinting attack on the Tor network. We also provide an attack that fits within the Tor threat model. The remaining sections of this paper are structure as follows: Section 2 discusses related works and the primary advances that TorFA has provided. Section 3 describes a theoretical fingerprint attack. In section 4 we describe our novel fingerprinting attack TorFA, its subsequent implementation, and conclude with experimentation and results. In section 5 we provide key future work, in section 6 countermeasure analysis, and conclude in section 7.

2 Related Work

Traffic analysis is the science of inferring information using communication patterns. Our focus on this work is on a particular kind of traffic analysis attack called website fingerprinting, and we will first describe related work in this area.

2.1 Website Fingerprinting

Sun et al. [14] were the first to perform a website fingerprinting attack on encrypted data. They showed that the number of objects in a webpage, and the size of those objects, uniquely identify a webpage. Similarly, Hintz [5] showed a proof-of-concept attack on the Safe Web anonymizing proxy, which also utilized information about object sizes. In addition, the order of the objects downloaded was also considered; for example, the main HTML page is downloaded first. We note that it is not possible to apply these attacks directly on the Tor protocol, since the attacks assume that object sizes are known, and which are not revealed by the Tor protocol.

Bissias et al. [1] proposed the first website fingerprinting attack that uses IP packet lengths to infer the content of encrypted communications. They showed that the distribution of IP packet lengths as well as inter-arrival times can uniquely identify the downloaded webpage. Observe that the use of inter-arrival time distribution requires the attacker to be local to the targeted user in the data-profiling phase. Liberatore and Levine [10] propose a variant that uses only packet length information to infer the downloaded website. This is advantageous, as the attack is location-neutral while generating profiles. The authors propose the use of Jaccard's coefficient and Naïve Bayes classifier for classification during fingerprinting. However, Liberatore and Levine's attack has not been tested on the Tor network. While our attack is also based on using only the packet length information, it differs from [10] in its use of the KL divergence metric for classification, as well as in its focus on studying the effectiveness of the fingerprinting attack on the real Tor network.

The Jaccard's coefficient [16] ignores the packet frequencies completely, and the Naïve Bayes classifier relies on absolute frequency values. Recently, Herrmann et al. [4] propose a state-of-the art multinomial Naïve Bayes classifier that dramatically improves the success rate of the website fingerprinting attack to greater than 95% for OpenSSH encrypted traffic. However, the multinomial naive Bayes classifier did not work as well on the real Tor network, and the success percentage was only 3%. In contrast, our attack achieved a success rate of greater than 24%.

2.2 Other Attacks

In general, low latency anonymous communication systems like Tor are vulnerable to end-to-end timing correlation [12], where an adversary observing the connection from the client as well as the destination endpoints is able to link them together.

Recent work has focused heavily on exploiting the heterogeneous nature of distributed systems to compromise user anonymity. Exploitable features include node congestion, latency, and reliability.

3 General Fingerprint Attack

In this section we discuss the design of a general-purpose website fingerprinting attack. We outline the types of considerations an attacker should keep in mind. A general fingerprint attack consists of two activities: a training phase, during which data are collected for fingerprint creation, and a classification phase, during which unknown website signatures are compared with those in the fingerprint database for identification. These two phases are described in detail below.

Training: During the training phase, the attacker collects statistical information about the observable characteristics of a given website transfer. Then the attacker creates a profile from the unique contents of the information. Steps include:

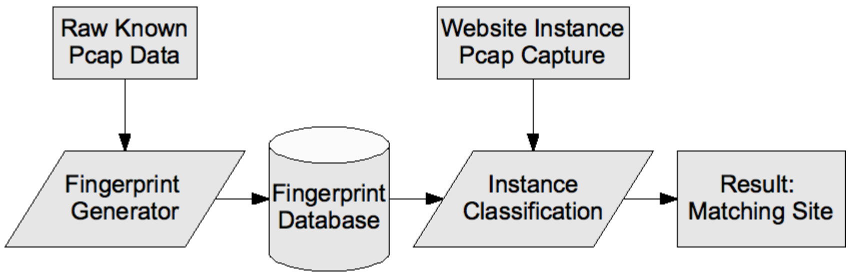

Figure 1: Program data and information flow

- Feature Selection: Identify attributes of the encrypted data that are unique to a given website. This requires the attacker to perform analysis of the low-level communications that are being compromised. The primary goal of this phase is to perform traffic analysis as discussed by Raymond [13].
- Identify the type of classifier that will be used to compare the fingerprints to the unknown samples. The type of classifier will have a large effect upon the data storage format. Previous efforts [4, 10] have used classifiers such as the Multinomial Naïve Bayes Classifier, Jaccard's Classifier, and Naïve Bayes Classifier with Kernel Density.
- Define a data format based upon the attributes and classifier technique.
- Compute and store fingerprints in a database for classification.

Unknown Classification: Steps include the design an algorithmic approach to performing the attack, and specific steps to accomplish that algorithm.

- Establish eavesdropping implementation/method.
- Record encrypted web traffic.
- Apply the classification routine to identify the website the user is viewing.

4 Fingerprint Attack Case Study

In this section, we discuss the design, implementation, and experimentation of our fingerprint attack. Section 4.1 section 4.3 describes the implementation details pertinent to meeting the attacker model and other important design choices, and section 4.4 describes our experimentation setup and results.

4.1 Fingerprint Attack Design

In this section we describe the feature selection, data classification model, and fingerprint data format. Section 4.2 describes the adversary threat model, and contrasts our approach to previous research; section 4.2.1 describes our selected features; section 4.2.2 discusses the mathematical model and tools we selected to perform classification of unknown website instances; section 4.2.3 describes the subsequent format for data values,

The general flow of information in TorFA can be seen in Figure 1. Raw known data are input into the fingerprint creator, which produces a fingerprint that is stored to a database. Then an unknown data item is entered into the instance classifier, which outputs the fingerprint match. Here we define the term *instance* to mean an instance of traffic data obtained by observing communications for one website download.

4.2 Adversary Threat Model

We assume an adversary with the following capabilities:

- The attacker can eavesdrop on the communications between the OP and the OR. This allows the attacker to identify the IP address of the client he or she is targeting, and thus the end user of the machine.
- The user is limited to downloading one website at a time. This is important because HTTP streams in Tor are multiplexed across one connection (e.g., if a user clicks two links at the same time in a browser, they will be downloaded concurrently, and Tor will put them into the same encrypted pipe). We will refer to this as the *multi-stream problem*, and discuss potential solutions in section 5.4.

In addition to assuming those capabilities, we make the following weak assumptions:

- The attacker is able to extract Tor traffic from the encrypted stream. The problem with extracting Tor traffic from the encrypted stream will be discussed further in section 5.3, but in order to be successful the attacker must not only know the specific port the OP is using, but also be able to filter out any traffic that uses TLS for its port number (e.g., email using SSL, automatic updating software, or chat clients). In section 5.3 we provide a solution to this problem.
- The victim's browser has the same configuration as the browser used in the training phase by the attacker. Due to the specific nature of how a browser downloads data, each browser can produce a different signature for the same website. Additionally, the browser configuration has the potential to greatly affect the fingerprint of a website. This problem manifests itself primarily in the decision on whether the browser is configured with or without cache enabled. The browser configuration assumption can be mitigated if the attacker creates per-browser fingerprints, rather than just per-website fingerprints. However, that option has the drawback that the more fingerprints there are in the database, the less likely it is that a match will be found.

All of the assumptions have been made by prior efforts on website fingerprinting [5, 1, 10, 4]. The reason why we identify weak assumptions here is that the focus of our attack is on working in a real-world environment, and thus our design and implementation are built to mitigate these assumptions. The weaker assumptions are issues that have trivial solutions. We contrast our approach to previous work to show the difficulties in implementing a real-world fingerprint attack.

It is important to note that we have relaxed some primary assumptions taken by prior work [1, 10, 4], namely, that the attacker has created a fingerprint for each site the user visits, the attacker knows which browser the victim is using, and the victim's browser is configured the same as the attacker's (e.g., the attacker creates profiles based upon no cache, and then assumes that the user has a browser with the same settings). By eliminating this assumption we, opened up the opportunity to use different classifier functions, which will be discussed in section 4.2.2. Another important assumption we remove is that we do not require that packet sizes are independent of one another.

4.2.1 Feature Analysis and Selection

The first step in creating fingerprints is to identify the features or specific attributes that are unique to a given download. The focus is to expose any information that might distinguish one website from another. Raymond [13] discusses several such attributes that can be used in this type of attack: packet timing, ordering, size, and direction.

Our analysis shows that Tor is resilient to timing attacks due to the dynamic nature of the Tor network. This dynamism is an attribute of the constant change in Tor circuits, as well as the option a client is given to choose any router other than the entry node along its circuit to be the exit node. The timing variation makes it hard to produce accurate results that rely on timing, but it may be possible to use timing as a secondary feature enabling more accurate results for the attack. The description of these capabilities is provided in section 5.2.

Based upon our analysis of Tor, we identified packet counts as the most consistent feature of website downloads, and attribute to implement in this case study. By neglecting both timing and ordering of packets, we provide a lower bound on the full capability of the attack. We revisit the feature selection problem in section 5.2 in discussing the relevant methods for including timing and ordering.

4.2.2 Classifier

The selection of the classifier is key to developing a successful fingerprint attack. Previous work has used Jaccard's coefficient [10], Naïve Bayes Classifier, Multinomial Naïve Bayes Classifier [4], and cross correlation [1]. These classifiers have several drawbacks. First, they require an assumption be made that the probabilities assigned to each packet size is independent of other packet sizes, and second, if the attacker removes the assumption that he or she knows all websites visited by a victim [4] the success rate goes from 98% to 40% [1]. We wanted to be able to produce a result that did not depend on knowing the victim's website preferences and thus researched new methods.

As the goal of our project was to develop a more realistic fingerprinting attack we decided upon the use of Kullback Leibler (K-L) divergence. In a related field, protocol identification, Hjelmvik [6] showed that K-L divergence [9] is successful at properly classifying protocol instances. Furthermore, K-L divergence does not require the independence assumption between packets.

K-L divergence is a measure in statistics that quantifies the difference between two probability distributions. It is also referred to as *relative entropy*. The following equation is used to measure the difference of a probability distribution Q from another probability distribution P:

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

The greater $D_{KL}(P||Q)$ is, the less Q is like P. In other words, the closer the K-L divergence measure is to zero, the more accurately Q matches P. With respect to our fingerprinting attack, P is the probability distribution of the known fingerprints, and the distribution Q is a sample of data captured while listening to client and entry router communications.

4.2.3 Fingerprint Format

In order to perform classification, it is necessary to adopt and enforce a standard data format throughout the entire attack process. The fingerprint format is heavily influence by the selected classifier, which is K-L

[1]The success rates described here refer to attacking OpenSSH not Tor

Packet Size	1	2	3
Frequency Count	10	5	5
Probability Vector	.5	.25	.25

Table 1: Sample probability frequency count and subsequent probability distribution vector. The sizes selected here do not have any bearing on the sizes used in our attack.

divergence. One key requirement of the K-L divergence is that our probability vectors must not have a zero value for any vector. This is important when defining the method by which we track frequency data in our implementation.

To create a probability vector of the packet counts, an attacker collects statistical data with respect to the feature being used for comparisons. This statistical information is represented as a frequency count of the number of times a specific packet size has been viewed by the attacker. An example is shown in Table 1[2].

In the attack, as data are collected (either in the learning phase or in the testing phase), the attacker increments the frequency counter for each packet size observed. These counts are considered a frequency distribution on the packet sizes. Once data have been collected, the frequency distribution is converted into a probability distribution by taking each count and normalizing it by the total number of objects sampled. These probability vectors are the data types that are described in section 4.2.2.

4.3 Implementation

In this section, we discuss lower-level implementation details that drive the experimentation. There are three primary phases to the attack as implemented: data collection, fingerprint generation, and classification. We discuss implementation choices we made and how they affect the expected results of the attack.

Data Collection: Specific issues related to data collection include: browser selection, traffic observation, filtering out non-Tor traffic, and whether or not to use browser cache.

Browser selection is important because one of the weak assumptions requires that the attacker uses the same browser as the victim. The reason why this assumption is made is that different browsers produce very different fingerprints for the same site. In order to quantify this effect, we have implemented data collection from two browsers: Safari and wget. Each is configured not configured to use cache. The primary reason for selecting wget is that it is a command-line utility, and as such provides much faster data collection and greater noise reduction due to the limited number of connections opened for each download. This is in contrast to a browsers such as *Safari* or Firefox, which may have several plugins installed and other updating utilities automatically connecting web services via the HTTPS protocol. In practice this intuition proved false as the Safari test results achieved the highest success rate (this is shown in section 4.4.2). We perform most experiments with wget, but have included Safari in order to quantify the difference between itself and the wget samples. There are a few potential problems in using wget to discuss as well. A primary issue with wget is that it may not accurately download downloading entire Flash objects. It will successfully download all files linked to in the static HTML, but if a Flash object has links to other sites, wget will miss this content. An additional configuration option used for wget, was to configure it so that it would not be considered a bot by web hosts, but our efforts were insufficient because one website, craigslist.com, denied any requests coming from our set of IP addresses.

In order to filter traffic, we decided to use tcpdump. The key to using this method of data collection is

[2]The columns correspond to packet sizes in arbitrary terms.

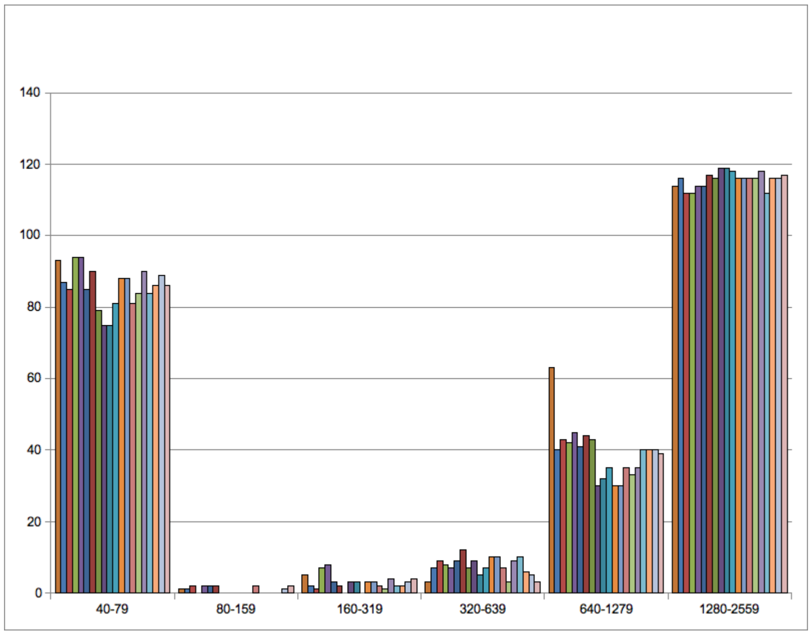

Figure 2: Histogram plot of packet size distribution. The y-axis represents packet counts, and the x-axis represent the size of the packets.

that tcpdump must be properly configured to record only Tor traffic. In order to accomplish this task, we established filters to listen on ports that are common to Tor. That introduces a problem, because each OR is allowed to select the port number it uses [15], and thus defining a static set of ports to listen to has the potential of resulting in missed traffic. If an OR uses a port we have not listened to, the tcpdump output file will have zero packets captured. We take advantage of this in fingerprint generation and instance comparison by not including any captures with zero packets. Section 5.3 discusses how this issue is avoided by taking advantage of Tor network topology knowledge to capture packets that are sent to and from ORs rather than the port number.

Generating Fingerprints: This section describes the format of the fingerprint and important design issues our its selection. The primary implementation issues related to fingerprint generation are the need to develop an algorithm for combining samples, and the constraint that all probability vectors must be non-zero due to K-L divergence properties. The reason the probability vectors cannot have any empty bins is related way K-L divergence is calculated. In the K-L diversion equation, the known distribution is divided by the unknown distribution, and thus the unknown distribution cannot have zero values, or else the equation would be undefined. This forced us to analyze the packet size distribution of a normal website download in order to identify ranges of packets that we could assume would have packets with a high probability.

Based upon our analysis of Tor traffic, we determined that packet sizes are clustered around four primary locations. After experimenting with several ranges, it was decided that to guarantee fulfilment of the requirement, we would need to select four ranges centered about the sizes that are most frequent. A sample packet frequency of specific ranges can be seen in Figure 2. The analysis resulted in teh conclusion that we should use a probability vector that included packet counts in the size ranges: $[0, 599]$, $[600, 799]$, $[800, 1449]$, and $[1450, 4500]$. These ranges were selected because frequency analysis showed that there were localized packet sizes that dominated the frequency counts, and thus the ranges were selected so that each one of these sizes was in a different range bin.

In order to create a fingerprint, which is a combination of several known website instance downloads, we calculate the sample mean of all packet counts and then create the probability vector from that data. In order to

Classification: A major issue with respect to classification is that the distribution of one website's fingerprint has the potential to match that of a different website with a similar probability distribution, but different overall packet frequency counts. For example, consider the two frequency distributions: $[200, 200, 200, 200]$ and $[1, 1, 1, 1]$. Clearly, these two websites are different, but the K-L divergence measure between the two is 0 because each probability distribution is: $[.25, .25, .25, .25]$, denoting an exact match. In order to account for this possibility our classification mechanism includes a filter that uses standard deviation to exclude any matches that are absolutely not in the set. Once the standard deviation has filtered the potential matches, the fingerprint with the lowest K-L divergence is selected as the match. It is important to notice that the standard deviation check can filter out all potential matches, and in this case TorFA indicates that there is no match. This is useful when we compare website samples that have not been included in the set of websites fingerprinted. In section **??**, we discuss the potential use of robust classifier combinations.

4.4 Experimentation

In the following sections we discuss the experimental setup of data collection, experiments implemented, and results. We show that the K-L divergence classifier is an improvement upon previous work [4], in that we achieve a success rate of 24%, while earlier efforts achieved only 2.96%. We show that smaller data sets produce higher success rates, at the cost of high false positive rates. We also show that the assumption that a different browser will produce much different data sets as we compare results from Safari on a Mac to wget on Ubuntu with an outcome of *zero* matches.

We designed our experiments not only to identify the basic success/failure rates of our classification engine, but also to display important characteristics about our methods and how to develop a realistic attack. The primary focus of the attack was to develop an approach that would become more realistic than previous approaches, which these experiments do.

4.4.1 Experimental Setup

In order to evaluate TorFA, we collected data using wget from 106 URLs with approximately $51,000$ unique downloads. All data captures were obtained by recording Tor traffic with tcpdump on the Ethernet card of the victim machine. As discussed previously, this is assumed to be an accurate assumption because anywhere along the route between the Ethernet card and the OR can be used to observe the traffic. We automated data collection by developing a ruby shell script that opens a tcpdump capture, downloads a selected website with wget, and then repeats. We collected data on six different machines, with three distinct setups. The machines used to collect the data are as follows:

- `Mac OSX Snow Leopard Darwin 10.2.0 Darwin Kernel Version 10.2.0: root:xnu-1486.2.11~1/RELEASE_I386 i386`
- `Linux 2.6.31-17-generic #54-Ubuntu SMP  x86_64 GNU/Linux`
- `Linux 2.6.31-17-generic-pae #54-Ubuntu SMP i686 GNU/Linux` (There are five instances of this machine running the script.)

Browser	Machine	Success Rate
wget	Ubuntu-M	24%
wget	Ubuntu-0	13%
wget	Ubuntu-1	10%
wget	Ubuntu-2	15%
wget	Ubuntu-3	3%
wget	Ubuntu-4	11%
wget	Mac	8%
Safari	Mac	25%

Table 2: Here we compare instances of sites against the fingerprint database that represents the same machine and browser.

4.4.2 Results

The results are comprised of performing three experiments. Let $W = \{w | w \text{ is a website in the fingerprint database}\}$. We identify an instance of a website download as w_i, where w_i represents a unique instance of a download from website w for $i = 1, 2, 3, \ldots$. For each experiment we compare w_is that are randomly selected the fingerprint and non-fingerprint sets. It is important to note that the number of site instances collected by machine Ubuntu-M is approximately six hundred instances. This is compared to the approximately twelve hundred instances collected by the other Ubuntu machines (each), and one hundred and fifty samples that were collected with Safari on the Macintosh. Additionally note that we created fingerprints using twenty randomly selected samples for each of the experiments described below.

4.4.3 Experiment 1

In this experiment we classify w_i against the database of all fingerprinted websites. We select w_i such that for each website the chosen w_is are evenly distributed and selected at random from the set of instances that comprise W's fingerprint and those instances of w that are not in the fingerprint. We perform this *ten* times and compute the average success rate. We also include a comparison of data collected with different browsers and machines against the primary machine Ubuntu-M. In these results we see that Ubuntu-M produces the best results. The reason why the results are better is due to less noise in the data collection phase. The other Ubuntu machines include more data, and the random nature of selecting the fingerprints to add into the database includes a greater potential for the poor choices to pull the results down. In order to avoid this we can use methods as discussed in section 5.1.

Table 2 displays the results for each set of sites being compared against the fingerprints created by the same machine and browser combination. Table 3 depicts the cross comparison of each set of sites (marked by browser and download) against one machine, Ubuntu-M. One thing to note is that the noise distribution across each of these machines will be different, causing some discrepancies. Some important things to note are: Safari did not match a single site with the same site using wget on Ubuntu-M, and that Safari performed well when matching against itself. These results suggest that due to the nature of the GUI web browser they require more packet downloads, and as such make it easier to identify one site from another.

Browser	Machine Instances	Success Rate
wget	Ubuntu-0	7%
wget	Ubuntu-1	6%
wget	Ubuntu-2	7%
wget	Ubuntu-3	3%
wget	Ubuntu-4	7%
wget	Mac	6%
Safari	Mac	0%

Table 3: Results from experiment 1. We compare site instances from each machine versus the fingerprint database of machine Ubuntu-M

Browser	Machine Instance	Fingerprint Set	Success Rate	False Positive Rate
wget	Ubuntu-0	News Sites	40%	78%
wget	Ubuntu-0	Retail Sites	43%	84%

Table 4: Here we created fingerprint sets out of a subset of the websites we collected data for. We compare the full set of data in this class with the same browser and machine versus the smaller database. In order to mimic a real attack we have selected websites that would compete against each other. The first fingerprint set includes news sites, and the second includes retail sites.

4.4.4 Experiment 2

In this experiment we classify w_i site instances that are not in the fingerprint set. The focus of the experiment is to display the false positive rates of TorFA. The result is that we see a failure rate of 99%.

4.4.5 Experiment 3

In this experiment we create a fingerprint database comprised of only a select few sites. The purpose of this experiment is to look at how TorFA performs given a smaller set of fingerprints to match data to. Due to the nature of the probability distribution the success rates should increase because there are less potentially false matches within the database.

Figure 4 displays the results of this experiment. Our intuition is correct in that the successful number of matches is high, but the costs for this high success rate is increased false positives.

5 Discussion and Future Work

The analysis of TorFA and subsequent results opened up a great deal of insight into the issues related to statistical analysis attacks on the Tor network. In this section we explore techniques that can be used to improve the initial TorFA results. The entire approach is developed to provide understanding about the real issues impeding this form of attack in the real world, and thus we provide here analysis and specific future steps to solving many of the major problems. In section 5.1 we provide methods of improving and filtering out various sources of noise during the data collection phase. Section 5.2 discusses the addition of new features to base the classification on, section 5.3 discusses specific implementation issues with Tor traffic capturing with tcpdump. Section refsec:multistream discusses the assumption that the end user downloads only one website instance at a time. Section 5.5 describes the mechanisms that are used to create a better classification engine.

5.1 Data Collection: Noise Reduction

In this section we discuss several issues related to noise sources in the data collection phase. The sources of noise in our collection data include: Tor control traffic, other TLS traffic, and Flash problems. We also discuss how to filter this noise out in a general purpose way.

Tor Control Messages: In Tor the term circuit denotes a logical route in the Tor network that an OP uses to send information on. Often times in an attack the client will build a new circuit, tear down a circuit, or change the circuit that is actively participating in a download. When this occurs the collection of data will be skewed by each of new Tor control packet. There are two solutions to this problem. First, analyze the Tor message handshaking protocol and filter these transactions out. It has been observed that whenever an OP initializes a new circuit it uses special control and "Client Hello" TLS messages to the OR, which can be used to identify the presence of non-site related packets. More analysis can be done to develop an understanding of the exact number of packets and what flow is associated with a given circuit setup action. Thus, whenever an initialization packet is seen by the classifier it can remove some of the packets being observed as though they were a part of the circuit setup packets. Second, if the attacker has either compromised an entry OR or has established his own OR, then the attacker will be able to differentiate between control and flow traffic. It was seen that whenever the OP creates a new circuit a TLS "Hello" packet is observed, which can be exploited by the attacker to determine when this form of Tor traffic is being processed concurrently with the attack.

Flash sites: It was observed that sites implemented primarily in Flash created poor fingerprints with high standard deviations. One example is *pandora.com*, which had a standard deviation of 325.9 total packets. Including these poor fingerprints into the database for a given set of sites will only degrade the overall success rate. Therefore, analysis can be done to take out any sites that have extremely high standard deviations [3].

Filter Anomalies: In any statistical approach there exist the potential for anomalies to exist in a given set of experiments. In our current version of the attack we perform no advanced filtering on the set of site instances that were recorded. To fix this problem a filter can be created to calculate a site's standard deviation and then this can be compared to all instances of that site, deleting any captures that are statistical outliers.

Syn/Ack Packet Removal: Our approach has included the use of syns and acks, but it can be seen that these packets do not hold much information related to the site in consideration. The primary characteristic of the site in consideration will be the larger data packets being sent from the destination to the OP. The syns and acks may not provide useful analysis with respect to these packets.

5.2 Traffic Feature Additions

One of the shortcomings of this attack is a lack of distinct attributes for each website. The limiting factor is that TorFA only uses packet counts. In our analysis we have identified several other features that could be pertinent in providing more successful classification. We identify that the potential features: timing, bandwidth analysis, and message protocol. The use of timing is severely limited due to the dynamic nature of the Tor network, but what can be used is relative timing analysis. Certain websites may have specific

[3] Remember that we are focusing on the attacker's perspective, so as this would be poor for comparability of results in a research environment it is exactly what an attacker would want to do.

response times that differ from others. Another feature to analyze is bandwidth. Websites that have more bandwidth capabilities can theoretically push more data. Further inspection should analyze the change in bandwidth measurements for a given site across multiple circuits and time frames. Lastly, an in depth inspection should consider the ordering of packets associated with each site instance. The analyst should observer if there is any order with respect to when certain sized messages are sent to and from the OP.

5.3 Tor Traffic Capturing

One of the primary assumptions we made is that the attacker can extract all Tor traffic from a given stream, as discussed in Section 4.2. In this section we consider potential methods to eliminate this assumption.

The problem with Tor traffic extraction is that the end user's system can use secure encrypted communications concurrently with Tor. Thus by using only port numbers to filter traffic the attacker observes non-Tor traffic mixed in with Tor traffic. Furthermore, the problem includes the potential that an OP has used a different port than the attacker is listening to. In order to solve this problem the attacker can obtain a list of all the active OPs and filter the traffic based upon those rather than only ports. This is possible due to the nature of the way the Tor protocol works. When an OP is activated it establishes a connection to a Tor directory server, which then send it a list of all the ORs. The attacker could use this functionality to obtain information about all of the ORs in a given region and use that to filter the traffic. Implementing this into TorFA would allow for a more realistic fingerprinting attack.

5.4 Multi-stream Parsing

Another primary assumption that we have made is that the end user will only download one webpage at a time. This is a naive assumption, that is made worse by the fact that tor multiplexes these requests over one circuit. There is one potential ways of reducing this assumption, which is to develop a method to detect when the user is performing simultaneous downloads. Research can be done into the use of Fourier transforms and advanced signals analysis to extract this information. If it can be identified that there are more than one download in a stream then the attacker can neglect that data sample. once this step is obtained, it might be possible to use packet-inter arrival times to distinguish between different flows in a stream.

5.5 Classification Improvements

There are several ways that the classification technique can be improved including: clustering algorithms, and multiple classifier selection.

5.5.1 Classifying Against Sets of Web Sites

A clustering algorithms would work in a way that would allow the attack to declare whether or not an unknown sample belongs to a specific class of sites. This would allow the attacker to implement clustering algorithms on the probability distributions, and ultimately create a scenario where the attacker can identify which site the user is not viewing. For example, if the attacker wants to know if a user is viewing *google.com*, and the algorithm provides a result that the instance is not a match with the class of *google.com* then the attacker has identified his primary concern.

5.5.2 Multiple Classifiers

The attacker can potentially increase his success rates if he uses a more robust mechanism to perform classification. We propose that one such robust mechanism is to use multiple mathematical classifiers to classify unknown instances. by using several classifiers on the data set, the classifier would benefit from the aggregate knowledge received from using multiple mathematical models to perform instance classification. Primarily, we use standard deviation to filter out those sites that were similar in probability distribution, but were much different in terms of total packet count. A methodology could be implemented using mechanisms in random such as Jaccard's classifier providing the first pass and K-L divergence as the final comparison. The best approach would be to implement several mechanisms and track performances to gain insight into the best combinations for success rates.

6 Countermeasures

In this section we briefly discuss the types of countermeasures that would successfully defend against our attack. In general, in order to defend against an attack that uses statistical methods is to make all data appear the same from a statistical standpoint. When every component that is being compared and classified appears the same then there is no possibility to identify one instance of an object versus another. The goal of defending against these attacks is to alter the visible probability distribution so that the statistical counters appear different than they are.

One method of defending against our attack is the use of adaptive traffic padding. In this method the entry OR adds *dummy* packets into the stream of messages sent to the OP with the idea that the attacker will not be able to identify whether or not a packet is real traffic or simply a *dummy* traffic. Tor already has this functionality built into the specification, which would make implementation easy. This method has an important finer point though; if the adaptive padding is done in any deterministic way the attacker will not have to change the attack in order to account for the padding because the extra packets will be included in the attackers data collection and thus built into his fingerprints. Even more important is the fact that randomized padding techniques produce statistically average output if not made complex enough. This type of attack must be done with extreme attention to understanding how the mechanism will be observed by the attacker's collection and fingerprint classification perspective.

In the Tor network implementing adaptive padding is extremely costly due to the excess bandwidth incurred, which is bad because the Tor servers are provided for free by end users. If the costs of keeping servers up increases too high hosts will have the option of taking it down. So in considering these defenses it is essential to also establish the cost of each implementation through the use of simulation or real network experiments.

7 Conclusion

In this paper we show that it is possible to observe content inside of a tor protected stream of data while adhering to the tor threat model. we were able to accomplish this by developing a novel attack TorFA that uses a K-L divergence classifier. The K-L divergence classifier is much more robust than previous work [4]. We are the first to provide an in depth analysis of tor traffic as it relates to the fingerprint attack, and identified specific obstacles to implementing this attack in the real world. We have specified direct actions that can be taken to remove these obstacles.

8 Acknowledgments

The authors would like to thank all of our anonymous reviewers. Additionally, we thank Robin Berthier for his excellent advice and direction, and Yih-Chun Hu for his ideas and support. We would also like to thank Audrey Dautenhahn and Jenny Applequist for there timely and wonderful editorial support.

References

[1] George Dean Bissias, Marc Liberatore, David Jensen, and Brian Neil Levine. Privacy vulnerabilities in encrypted http streams. In *In Proceedings of Privacy Enhancing Technologies Workshop (PET 2005)*, pages 1–11, 2005.

[2] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *In Proceedings of the 13th USENIX Security Symposium*, pages 303–320, 2004.

[3] US Naval Research Lab's Center for High Assurance Computer Systems (CHACS). Onion routing. `http://www.onion-router.net/`.

[4] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *CCSW '09: Proceedings of the 2009 ACM workshop on Cloud computing security*, pages 31–42, New York, NY, USA, 2009. ACM.

[5] Andrew Hintz. Fingerprinting websites using traffic analysis. In *Workshop on Privacy Enhancing Technologies*, 2002.

[6] Erik Hjelmvik. Statistical protocol identification with spid: Preliminary results. Swedish National Computer Networking Workshop, 2009.

[7] Google Inc. Privacy faq. `http://www.google.com/privacy_faq.html`, 2010.

[8] The Tor Project Inc. Tor volunteer. `http://www.torproject.org/volunteer.html.en`.

[9] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[10] Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted http connections. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 255–263, New York, NY, USA, 2006. ACM.

[11] Viktor Mayer-Schonberger. Useful void: The art of forgetting in the age of ubiquitous computing. April 2007.

[12] Steven Murdoch and George Danezis. Low-cost traffic analysis of tor. IEEE Symposium on Security and Privacy, 2005.

[13] Jean-Franois Raymond. Traffic analysis: Protocols, attacks, design issues and open problems. In *PROCEEDINGS OF INTERNATIONAL WORKSHOP ON DESIGN ISSUES IN ANONYMITY AND UNOBSERVABILITY*, pages 10–29. Springer-Verlag New York, Inc., 2001.

[14] Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkata N. Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *SP '02: Proceedings of the 2002 IEEE Symposium on Security and Privacy*, page 19, Washington, DC, USA, 2002. IEEE Computer Society.

[15] TheOnionRouter. Torfaq. `http://wiki.noreply.org/noreply/TheOnionRouter/TorFAQ#FirewalledClient`.

[16] C.J. van Rijsbergen and Ph. D. Information retrieval, 1979.