HTTPS Vulnerability to Fine Grain Traffic Analysis

Bradley Miller

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2012-245
http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-245.html

December 14, 2012

Copyright © 2012, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

HTTPS Vulnerability to Fine Grain Traffic Analysis

by Bradley A. Miller

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor J.D. Tygar
Research Advisor

(Date)

* * * * * * *

Professor Anthony D. Joseph
Second Reader

(Date)

ABSTRACT

In this thesis, we apply the pattern recognition and data processing strengths of machine learning to accomplish traffic analysis objectives. Traffic analysis relies on the use of observable features of encrypted traffic in order to infer plaintext contents. We apply a clustering technique to HTTPS encrypted traffic on websites covering medical, legal and financial topics and achieve accuracy rates ranging from 64% - 99% when identifying traffic within each website. The total number of URLs considered on each page ranged from 176 to 366. We present our results along with a justification of the machine learning techniques employed and an evaluation which explores the impact on accuracy of variations in amount of training data, number of clustering algorithm invocations, and convergence threshold. Our technique represents a significant improvement over previous techniques which have achieved similar accuracy, albeit with the aid of supporting assumptions simplifying traffic analysis. We examine these assumptions more closely and present results suggesting that two assumptions, browser cache configuration and selection of webpages for evaluation, can have considerable impact on analysis. Additionally, we propose a set of minimum evaluation standards for improved quality in traffic analysis evaluations.

1. INTRODUCTION

Learning theory provides powerful techniques for traffic analysis, an attack which uses observable traffic features such as packet size and timing in order to infer the plaintext contents of encrypted traffic. In this thesis, we introduce improved machine learning approaches as a basis for measuring vulnerability of encrypted web traffic to traffic analysis. By treating features extracted from traffic as data to be processed and classified by machine learning techniques, we are able to achieve accuracy rates ranging from 70% - 99% when identifying traffic generated by URLs found in the same webpage. The evaluation used between 176 and 366 URLs from each website, and the websites were either legal, medical or financial in nature. Furthermore, we evaluate our attack with the browser cache and active content enabled, representing a significant improvement over the assumptions held by previously proposed techniques. Our approach also compares favorably against prior efforts with the browser cache disabled and comparing only homepages, as we achieve 95% accuracy over a set of 402 homepages while ignoring all data about packet destination.

The core of our approach is a clustering technique which produces a series of Gaussian Mixture Models (GMMs) for each page we attempt to identify. Each GMM in the series corresponds to a different second level domain seen in traffic while loading a particular URL, where the second level domain associated with a packet is determined by the server's IP address. Rather than using a maximum likelihood approach for fitting the GMMs, we use a Bayesian technique which both facilitates inferring the number of components in each mixture and provides increased resilience against minor variations in traffic. We apply the clustering approach over a two-dimensional feature space representing pairs of outgoing and incoming bursts of traffic seen in traffic to each specific domain. We find that a clustering technique accommodates the effects of caching well by calculating the likelihood of a sample based only traffic which *actually occurs*. This stands in contrast to techniques in which the model specifies a fixed number of times that each value of each feature is expected to occur, and then decreases the likelihood of the sample if feature values occur less often than expected.

Previous work has shown that traffic analysis attacks can achieve high accuracy when used in the web setting, subject to various sets of supporting assumptions. In this thesis, we argue that these assumptions have often been unrealistic, tending to exaggerate traffic analysis vulnerability and inflate the performance of previously proposed attacks. In support of this claim, we present data regarding two of the most common assumptions in order to measure their potential effect on the outcome of traffic analysis evaluations.

The most common assumption that we believe introduces significant bias into results is the disabling of the browser cache. In order to measure the effects of caching we conduct a randomized browsing exercise on the website of a common US bank, repeating the exercise with the cache both enabled and disabled. Our results show that caching both decreases the total amount of traffic and increases variations in the traffic, thereby complicating traffic analysis. We also note that many evaluations attempt to differentiate between a fixed set of popular website homepages known in advance to the attacker. We demonstrate that website homepages generate more traffic than internal pages within websites, thereby simplifying the task of traffic analysis. In further support of this claim, we compare the accuracy of our own technique when identifying traffic from website homepages to accuracy when comparing pages within a single website.

Having presented evidence of the potential bias which common assumptions can introduce into evaluation results, we also propose criteria for more accurate evaluations. Although we do maintain that fully realistic simulations or reconstructions of actual internet traffic are ultimately intractable, this does not mean we should make no efforts to harmonize the discrepancies we are aware of between simulated traffic used in evaluations and actual real world traffic. For example, evaluations should reflect the traffic diversity caused by user specific content and by differences in browser software and configurations, such as caching and common types of plug-ins and active content. Additionally, evaluations should focus on objectives which are relevant to the privacy technology and threat model being considered. For example, an adversary analyzing any system (such as SSH tunnels, VPNs, etc.) that tunnels all web traffic through a single connection must not assume that all traffic comes from a fixed set of pages. In many cases, it may not even be appropriate to assume that all traffic is web traffic.

In Section 2 we present the evaluation methodologies of previous work, and in Section 3 we critique these methodologies. In Section 4 we suggest minimum standards for evaluation of traffic analysis techniques. In Section 5 we present a new traffic analysis technique in light of our proposed minimum standards, and in Section 6 we evaluate our approach. Section 7 concludes and discusses future work.

2. EVALUATIONS IN RELATED WORK

In this section we present the evaluation methodologies and results of previous work. Correct interpretation and under-

Author	Privacy Technology	Page Set Scope	Page Set Size	Accuracy (%)	Cache	Single Site vs. Homepages	Analysis Primitive	Active Content	Software Stack
Hintz [9]	SSL proxy	Closed	5	100	?	Homepages	Request	?	IE5
∗Sun [12]	SSL proxy	Open	2,000 100,000	75 (TP) 1.5 (FP)	Off	Mixed	Request	Off	IE5
†Cheng [4]	HTTPS	Closed	71	94	Off	Single Site	Request	Off	Netscape
Danezis [6]	HTTPS	Closed	?	89	n/a	Single Site	Request	n/a	n/a
Herrmann [8]	SSH tunnel	Closed	775	97	Off	Homepages	Packet	?	Linux, FF 2.0
‡Dyer [7]	SSH tunnel	Closed	775	91	Off	Homepages	Packet	?	Linux, FF 2.0
Liberatore [10]	SSH tunnel	Closed	1000	75	Off	Homepages	Packet	Flash	Linux, FF1.5
Bissias [3]	SSH tunnel	Closed	100	23	?	Homepages	Packet	?	Linux, FF1.0
Herrmann [8]	Tor	Closed	775	3	Off	Homepages	Packet	?	Linux, FF2.0
Panchenko [11]	Tor	Closed	775	55	Off	Homepages	Packet	Off	Linux, FF3
Panchenko [11]	Tor	Open	5 1,000	56-73 (TP) .05-.89 (FP)	Off	Homepages	Packet	Off	Linux, FF3
Coull [5]	Anonymous Trace	Open	50 100	49 .18	On	Homepages	NetFlow	Flash & Scripts	FF

∗Sun gathered pages from a link database and hence likely used both homepages and internal pages from a variety of websites.
†Cheng set the cache to 200KB, which we regard as *effectively* off. The analyzer trained on 489 pages, but only 71 were actually tested.
‡Dyer evaluated his attack using datasets released by Herrmann and Liberatore, focusing on Herrmann's dataset.

Table 1: Summary of related works. Note that "?" indicates the author did not specify the property, "n/a" indicates the property does not apply to the author, and "FF" indicates Firefox.

standing of the results requires appreciating the environment in which those results were obtained. Therefore, any accuracy measurements must be regarded as *relative* to a set of underlying assumptions. Note that the degree to which an assumption is reasonable depends on the privacy technology and the threat model being considered in the evaluation.

In considering previous work, it is helpful to separate related work into two separate categories based on the type of privacy technology which is used in the evaluation. The first category includes systems which function below the application layer and are designed to not require support from the website or browser. This category includes systems such as SSL, Wi-Fi, VPNs and SSH tunnels. The second group includes systems such as the Tor browser bundle that make modifications at the application layer (i.e. within the browser). Common modifications include disabling the cache and active page content, as these increase state, functionality and binary vulnerabilities within the browser, all of which could be exploited to compromise user privacy. Ironically, although these measures address immediate concerns of direct privacy violations, these modifications also increase vulnerability to traffic analysis by reducing variations in user traffic.

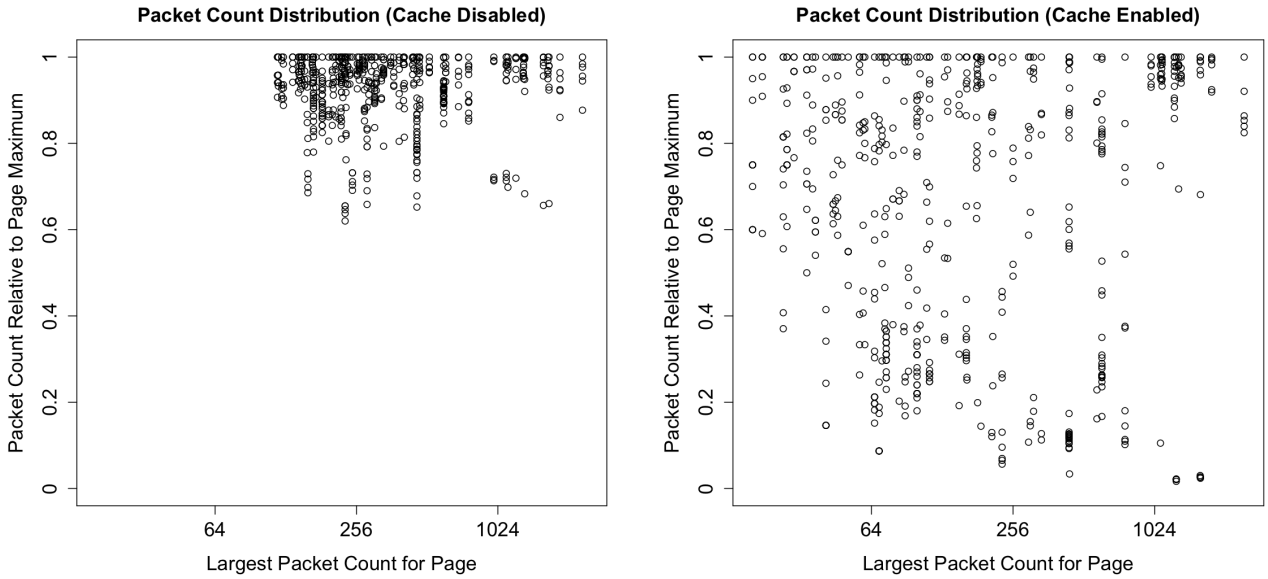
Although browser and application layer modifications can powerfully increase user privacy, many of these modifications are poorly understood and unlikely to be pursued by non-expert users. Rather, we believe that users are only likely to use these features when the privacy technology they are using includes an expertly configured browser. Hence, problems arise when researchers apply modifications made in expertly configured application layer systems to studies of systems which run below the application layer. Although users may plausibly deploy these application layer modifications while using privacy technologies which function below the application layer, this is unlikely. Most likely, the end effect has been that the introduction of these assumptions biases results towards appearing more severe, while introducing assumptions which researchers and practitioners easily identify as being somewhat unrealistic.

While exaggerating vulnerability may appear *safe* from a security standpoint, we argue that in this instance the opposite is true. Although some works have reported success rates in excess of 90% using basic heuristics and machine learning techniques applied to mundane features, encryption mechanisms remain widely deployed which include no built-in defenses against traffic analysis attacks. The doubt cast on traffic analysis results by generous assumptions has actually increased security vulnerability as researchers and developers decline to widely deploy traffic analysis defense mechanisms. This doubt, evidenced by the security community's response, persists even in the face of publications concluding that encrypted channels do little to prevent an attacker from learning the pages requested by a user [7].

Table 1 presents an overview of the evaluation techniques used in prior work and the associated results. Note the prevalence of non-default software configurations used with systems which function below the application layer. The columns are as follows:

Privacy Technology: The encryption or protection mechanism analyzed for traffic analysis vulnerability. Note that some authors considered multiple mechanisms, and hence appear twice.

Page Set Scope: *Closed* indicates the evaluation used a fixed set of pages known to the attacker in advance. *Open* indicates the evaluation used traffic from pages both of interest and unknown to the attacker. For example, a closed evaluation is appropriate for a setting where the attacker can identify the traffic destination and corresponding website, and can therefore enumerate all pages which exist on that website. Open is appropriate for settings such as Tor



(a) (b)

Figure 1: Figures 1a and 1b were each generated by three random browsing sessions which loaded 72 pages within a bank website multiple times. Figure 1a shows the cache disabled and Figure 1b shows the cache set to 1MB, 8MB and 64MB for one session each. Each point represents the number of data packets in an individual page load as a fraction of the maximum number of packets for that page on the same plot.

or SSH tunnels, where all web traffic is sent through a single channel. Since the attacker is not given the destination of the traffic, he must assume that the channel could contain traffic to arbitrary webpages.

Page Set Size: For closed scope, the number of pages used in the evaluation. For open scope, the number of pages of interest to the attacker and the number of background traffic pages, respectively.

Accuracy: For closed scope, the percent of pages correctly identified. For open scope, the true positive (TP) rate of correctly identifying pages of interest and false positive (FP) rate of mistaking background traffic for interesting pages.

Cache: *Off* indicates browser caching was disabled. *On* indicates caching was enabled to default size.

Single Site vs. Homepages: *Single Site* indicates all pages used in the evaluation came from a single website. *Homepages* indicates all pages used in the evaluation were the homepages of different websites.

Analysis Primitive: The basic unit on which traffic analysis was conducted. *Request* indicates the analysis operated on the size of each object (e.g. image, style sheet, etc.) loaded for each page. *Packet* indicates meta-data observed from TCP packets. *NetFlow* indicates network traces anonymized using NetFlow.

Active Content: Indicates whether Flash, JavaScript, Java or any other plugins were enabled in the browser.

Software Stack: The OS and browser used in the evaluation.

Several authors require discussion in addition to Table 1. Danezis evaluated his technique using HTTP server logs rather than generated traffic. Since these logs record information at HTTP request granularity, Danezis' accuracy rating describes success at identifying individual HTTP requests as opposed to entire page loads. To appreciate this distinction, suppose there are three pages that each have a unique HTML file and share an image. Furthermore, suppose that the browser cache contains all HTML files but not the shared image. If the user loads any of the pages, the only request will be for the uncached image. Even if the request is identified with 100% accuracy, there is still no way to know which page was actually loaded.

Although Herrmann evaluated his technique with caching disabled, he also describes a "preliminary" evaluation in which he enabled caching. Aided by several assumptions increasing traffic vulnerability, Herrmann achieved 92% accuracy. Herrmann increased the browser cache size from the 50MB Friefox default to 2GB, thereby preventing cache evictions. Herrmann also loaded all pages in the same order during both training and testing, further stabilizing the state of the cache. Herrmann's evaluation also considered only the homepages of popular websites. These pages tend to be content rich, generating more traffic and offering increased resilience to caching and consequently increased vulnerability to traffic analysis. Many homepages (e.g. news sites) may also disable caching of many objects to ensure users get timely content. In contrast to Herrmann, our evaluation

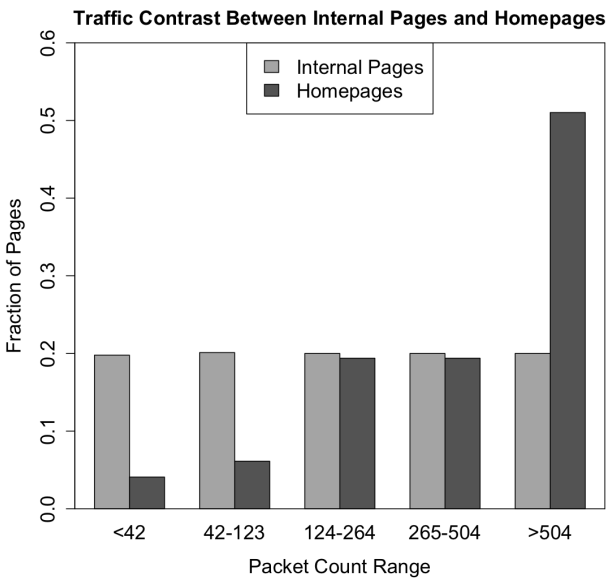


Figure 2: Difference in total number of data packets seen while loading website homepages and internal pages within a website. Notice that five ranges of packet sizes have been determined such that they evenly partition the range of packet counts for internal pages.

uses randomized browsing and varied cache sizes in order to robustly simulate diverse cache states. Our evaluation also uses pages from a single website, generating a smaller traffic footprint than a comparison of homepages.

3. EVALUATION TECHNIQUE ANALYSIS

In this section we review the potential effects of the evaluation assumptions and techniques described in Section 2. To facilitate our discussion, we categorize the assumptions into two distinct groups. *Explicit* assumptions are those designed to make traffic more vulnerable to attack, and influence factors such as the pages selected for analysis and the configuration of the system generating traffic. *Implicit* assumptions arise as artifacts of the evaluation infrastructure, and include design aspects such as using the same software stack, browser cookies, or IP address to generate all traffic. We find that the broad scope of assumptions casts doubt on the true effectiveness of previously proposed traffic analysis techniques.

We begin our discussion with the explicit assumptions. Most significantly, *all* evaluations we have surveyed that utilize actual packet traces have assumed that the browser cache has been either fully disabled, set so small (200 KB) as to be *effectively* disabled [4], or set so large (2GB) as to be relatively stable [8]. Coull et al. consider a reasonable cache configuration, although their evaluation is restricted to a relatively small set of the Alexa Top 50 website homepages [5]. Danezis' use of HTTP server logs does reflect the effects of caching, but his evaluation attempts to identify individual HTTP requests rather than entire page loads and assumes that individual object lengths are available [6]. The assumption that the browser cache is disabled is only appropriate for privacy systems which include an expertly configured application level component. For example, the Firefox distribution in the Tor Browser Bundle disables caching by default, although even Tor guidelines allow users to enable the memory cache [1]. There is no reason to believe that users browsing the web over HTTPS will disable their browser cache.

Figures 1a and 1b present the effects of caching on the traffic fingerprint of a website. Data for each figure was gathered by visiting 72 unique pages within the secure portion of the website of a common US bank. The 72 pages were spread over nine separate sections of the website. Although the ordering of page visits within each of the nine sections remained constant as this was often dictated by the link structure of the website, the order in which the sections themselves were visited was randomized. Note that some pages were visited multiple times within each section as necessitated by link structure.

Each section was visited twice during each sample in order to allow for potential effects of caching. The Firefox parameters `browser.cache.disk.capacity` and `browser.cache.memory.capacity` were used to set the size of the disk and memory caches respectively. In order to gather data for caching disabled, the cache size was set to 0. In order to gather data for caching enabled, we varied the size of the cache to simulate a range of cache states. Both the memory and disk caches were set to 1MB, 8MB and 64MB for one sample each.

We gathered data using Firefox 13.0.1 running on Ubuntu 12.04 using a virtual machine provided by VirtualBox 4.1.18. Since Firefox came with Flash installed, the only other browser extensions we used were Greasemonkey version 0.9.20 and Firebug version 1.9.2. Traffic was recorded for each page by invoking and terminating TCPDUMP in synchronization with the page loadings conducted by Firefox. This was accomplished by making requests to a local server which controlled TCPDUMP appropriately.

The figures concentrate on packet count since packets are the most basic unit that is commonly observable in encrypted traffic and the unit that most traffic analysis attacks operate on. Notice that the variation in traffic for each page is larger in Figure 1b than in Figure 1a, in effect blurring the traffic fingerprint. Also, notice that the total amount of traffic decreases with caching enabled, making the traffic fingerprint smaller. This demonstrates that the effects of caching have high potential to frustrate traffic analysis.

In addition to assumptions about browser cache settings, assumptions about active content and plugins are likely to have influenced results. Some authors have assumed that active content such as JavaScript and Flash are disabled [4, 11, 12], while others have not specified either way [3, 7, 8, 9]. Some common uses of active contents include increasing interaction with the user, loading advertising, and customizing the page to the user's browser environment. These are all likely to increase the traffic generated by the page and complicate traffic analysis attacks.

Separate from assumptions about browser configuration, assumptions about traffic parsing have also influenced evalua-

tion results. Several authors have also designed traffic analysis techniques that utilized object sizes as a feature [4, 6, 9, 12]. These authors have recovered individual object sizes using either web server logs, unencrypted HTTP traffic or browsers which don't pipeline requests. Since individual object sizes cannot be recovered from encrypted traffic issued by a modern web browser, it is unclear if these techniques would be effective in practice.

The selection of websites for evaluation has also biased results. Cheng intentionally chose a static site with a mix of HTML and images [4], while others have compared the homepages of different websites [3, 5, 7, 8, 9, 10, 11]. Given the difficulty associated with identifying "typical" or "average" websites for the purposes of traffic analysis evaluations, selecting sites from rankings of the most popular internet websites does seem reasonable. That said, evaluations must also include more than website homepages as interactions with the homepage alone ostensibly constitute a minority of user interactions with any given website.

In order to quantify the effects of comparing website homepages rather than many pages from within a single website, we randomly selected 100 of the Alexa top 1,000 websites, loaded the homepage of each site, and then browsed through nine additional links on the site at random. The only restriction applied to link selection was that the link could not lead to either the root or index page (i.e. the homepage) of any website. This restriction served to separate traffic generated by website homepages from traffic generated by pages within a site. Data was gathered using the Chickenfoot extension to Firefox running on OS X with caching and active content enabled.

Figure 2 summarizes the contrast between website homepages and pages deeper within websites. By partitioning the total count of data packets transferred in the loading of pages internal to a website into five equal size buckets, we see that there is a clear skew towards homepages being larger and more content rich. Although part of this effect is due to caching, this shows that an evaluation which enables the cache and visits only homepages of different websites is still utilizing traffic which has unrealistically high volume, thereby producing more content for traffic analysis and machine learning techniques.

Results have also been influenced by the scope of pages considered in the evaluation. Recall that evaluations may consider either a closed or open scope. In a closed scope, the evaluation is restricted to a fixed set of webpages known in advance to the attacker. In an open scope, the attacker may encounter arbitrary traffic while observing a connection and is interested in identifing traffic only from a fixed set of webpages, treating the remaining traffic as background traffic. Closed models are only realistic in settings where the attacker is monitoring either traffic from a direct connection between the endpoints (such as HTTPS) or traffic which has already left any proxy system. In settings where the attacker monitors an encrypted channel to a proxy (e.g. Tor or an SSH connection), the attacker cannot rely on the user only visiting webpages from a fixed set. Thorough evaluations in open world settings must vary the size of the set of interesting pages as evaluations have shown larger sets can increase the false positive rate [11].

Explicit assumptions aside, implicit assumptions can arise as consequences of the evaluation infrastructure. Although web traffic is increasingly diverse due to user specific content and differences in browser software and plugins, few evaluations mention any measures taken in the experimental design to account for these differences. Hintz trained and evaluated his attack on separate machines, although he only analyzed five pages [9]. Danezis trained and evaluated his attack using server logs [6]. Absent any reflection of user diversity, using the same OS, browser, plugins, IP address and cookies for all training and evaluation will reduce traffic diversity.

Infrastructure has also influenced evaluations by automatically isolating individual page loads, allowing the attacker to avoid parsing actual traffic. This assumption results in an unrealistically high true positive rate if an attacker is unable to actually isolate traffic in practice. Less obviously, the base rate of actual positives, which may be influenced both by the traffic itself and the parsing of the traffic, is fundamentally unknown. This problem is particularly relevant to transport layer proxies, such as Tor, where added noise, high latencies and non-web traffic make parsing more difficult. Without knowing the base rate we are unable to determine the acceptable range for the false positive rate and hence are unable to determine whether traffic analysis could effectively be deployed against these systems in practice.

4. ROBUST EVALUATION OBJECTIVES

Having analyzed evaluation methodologies, we now articulate several objectives for more rigorous evaluation of traffic analysis techniques. Given the wide variation of internet traffic and the nearly infinite range of potential features for use in traffic analysis, we do not believe that it is possible to consistently simulate or recreate traffic which is always "realistic." That said, we view these objectives as a non-exhaustive initial guideline and believe that they do result in increasingly realistic evaluations. We also recognize that meaningful evaluation is possible without satisfying all objectives, although each objective addresses important concerns and improves evaluation quality. We enumerate the objectives in bold with additional details below.

Software configurations should be in agreement with the threat model. Unless the work is considering attacks against a privacy technology which includes application level components, such as a customized browser configuration, the work should assume that all browser options are set to the default setting and any common plugins are installed (e.g. Flash).

The page scope should be in agreement with the threat model. A closed scope is reasonable only when the destination IP address of the traffic is known to the attacker, the attacker is able to associate the IP address with a website, and the attacker would reasonably be able to crawl all individual pages served from that website. If the destination IP is unknown, as is the case with transport layer proxies, the evaluation should be open scope and assume the attacker will encounter previously unseen traffic.

Pages analyzed and identification objectives must

be consistent with user behavior and priorities. The pages used for evaluation must include both website homepages and internal pages, ideally mixed in a way which is proportional to actual user browsing habits. This mixture is key not only because pages within a website share more content than those from different websites, yielding higher caching effects, but also because internal pages produce less traffic than website homepages. Separate from concerns about the mixture of pages used in the evaluation, the evaluation must also consider likely objectives of the user. For example, if the user is likely to want to conceal interaction with an entire website, rather than just individual pages within that website, then the analysis should consider the identification of website visits over a series of page loads rather than isolated page loads.

Software diversity and user specific content should be reflected in the evaluation. Differences in HTML and script engines between browsers may cause variations in the number of connections, ordering of requests, or even which objects are requested. Differences in the OS and plugins, screen resolution and language settings may cause additional differences. Separately, user contributed content and content targeted to the user by the website will also alter the traffic fingerprint.

Evaluations should include high-interaction traffic from Web 2.0 technologies. Websites including many UI elements that generate traffic for user interaction complicate the traffic fingerprint. For example, the traffic stream for a website (or webapp) that utilizes extensive client-side scripting to extend the functionality at a single page, rather than navigating to a different page, may contain more, smaller events than typical web traffic.

Evaluations should address difficulty in isolating individual page loads. Traffic segments extracted from real-world traffic may contain parts of multiple requests and/or not contain all packets from any single request. This is especially true for proxy systems such as Tor which have higher latency than direct connections. This difficulty in precisely identifying and isolating traffic caused by individual page loads may influence accuracy as extraneous traffic may decrease the true positive rate and the base rate will be affected by how many "requests" are parsed from the original traffic stream.

An evaluation using real-world HTTP traffic would satisfy many objectives, although techniques must use only packet meta-data and not features such as object length recovered from packet contents. Unfortunately, this approach has the problem of labeling page loads, as faced in Danezis' evaluation [6]. Alternatively, VMs with a range of software configurations could also generate diverse traffic. Making multiple visits to a website in order to build user history and potentially lead the site to customize content to the user could also increase traffic diversity. We leave the further development of these techniques as future work.

5. ATTACK TECHNIQUE

In this section, we present the design and mathematical specification of our attack. Section 5.1 presents an overview of the attack as well as the construction of traffic models from training data, Section 5.2 describes the details of how the attack is conducted given a traffic model, and Sections 5.3 and 5.4 describe extensions to the attack which offer performance improvements.

Throughout the section, we provide a running example with the aid of Figures 3, 4, 5, and 6. The running example uses hand-created feature sets and performance statistics and is not drawn from any website, real or simulated. This example is provided for demonstration purposes only to act as an aid in presentation of the attack technique.

5.1 Overview and Model Generation

In this section we present an overview of our traffic analysis technique. We begin by describing the features used in analysis and then present the motivation and rationale for selecting the underlying machine learning technique as well as how the technique is employed to generate traffic models. We have designed our attack to support the evaluation standards outlined in Section 4, including robust performance in the presence of caching and dynamically generated content.

The feature set we use in our analysis is the (outgoing, incoming) byte counts of traffic bursts seen on each individual TCP connection used while loading a page over HTTPS. For example, denoting outgoing packets as positive and incoming packets as negative, the packet sequence [+314, +660, -100, -1500, -1230, +70, -350] yields the features (974, 2830) and (70, 350). Once all features have been extracted from each TCP connection, the features are aggregated into groups based on the second level domain of the hostname associated with the destination IP of the TCP connection. Thus, if seven separate domains are contacted in the course of loading a website, the corresponding feature set will consist of seven sets of ordered pairs where each pair corresponds to an (outgoing, incoming) burst of traffic. All IPs for which the reverse DNS lookup fails to return a hostname are treated as a single "unknown" domain.

Figure 3 presents six hand-fabricated feature sets from three visits to two pages each. The feature sets from the same page have been aggregated to the same plot as they would be to form a set of training data. Notice that within each plot the data tends to form two clusters, where each feature set produces one data sample in each of the clusters. Intuitively, this occurs because each data point corresponds to a series of pipelined requests and corresponding responses, and these requests and responses are necessarily repeated on successive loadings of a page.

Notice that in Figure 3 features from Domain A are more tightly clustered and consistent than features from Domain B. This pattern is typical of websites which contain content from multiple domains. Consider a website `foobar.com` which displays dynamically generated ads from `google.com`. The core content for each page, served by `foobar.com`, is likely to be relatively consistent across successive loadings of a page, while the dynamically generated ads, served by `google.com`, will tend to vary. By treating features from different domains separately during the clustering process, we remove extraneous data which may impede the clustering process from recognizing sets of points caused by the same request and response sequence. We also recognize that the

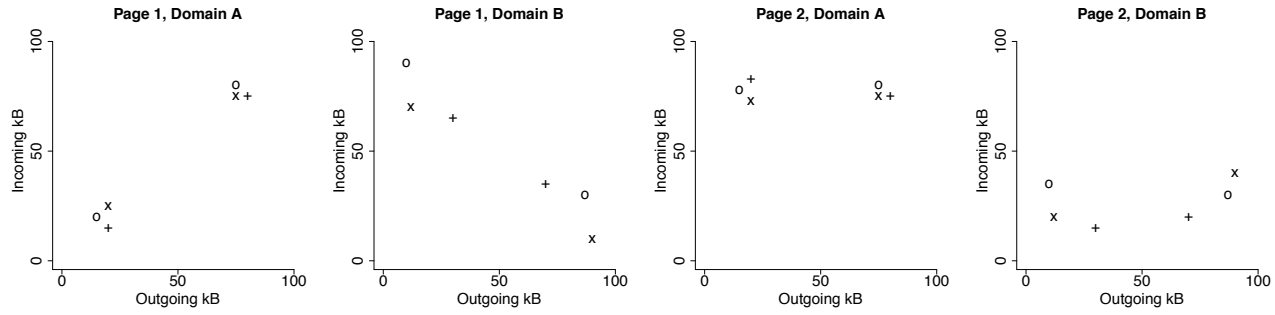


Figure 3: Training data for two pages, where each set of training data contains three feature sets, indicated separately by 'x', 'o' and '+'. Each feature set represents a single visit to a page and includes two (outgoing, incoming) traffic pairs to two domains each. Notice that traffic to Domain A tends to be more tightly clustered than traffic to Domain B, revealing greater information about traffic contents.

size (covariance) of each cluster will tend to vary as a function of the domain. As will be show in Section 5.3, treating data from different domains separately also allows for differences in the predictive accuracy of each domain to be considered in the attack process.

The (outgoing, incoming) size of traffic bursts offers two properties particularly well suited to traffic analysis. By aggregating the lengths of continuous packet sequences in a given direction into a single statistic, fragmentation effects are removed from the data. Fragmentation could occur either as a natural result of transmission through the network, or in the application layer as HTTP servers use chunked encoding to transfer responses to clients as content becomes available. Additionally, analyzing pairs of outgoing and incoming traffic bursts allows the data to contain a minimal amount of ordering information and go beyond techniques which focus purely on packet size distributions.

Having established the feature set for our analysis, we now turn to selection of an underlying machine learning technique. To identify similar feature sets while maintaining tolerance for minor variations, we selected an approach which uses Gaussian Mixture Models (GMMs) to cluster features rather than an approach which counts individual instances of specific feature values. By using a clustering approach, we are able to recognize that that the ordered pair (370,2200) is more similar to (375,2300) than it is to (632,50000).

To appreciate the benefit of tolerating minor traffic variations, consider a page which has an image banner. Although the banner may always be the same number of pixels on the page, this does not mean that the resource requested will always be the same size. Variations in the content of the image can impact the effectiveness of compression techniques used in image formats, resulting in changes in the total byte count. Similarly, as user specific content inserted into portions of a page can cause variations in traffic, the total byte counts are likely to remain similar yet distinct across many users.

To fit a GMM to training data, we elected to use a Bayesian approach to clustering rather than the maximum likelihood approach since the ability to introduce prior distributions over the parameters of the GMM fixes some difficulties associated with maximum likelihood. Recall that in the maximum likelihood approach, a cluster may be assigned to a single point, causing the cluster to collapse onto that point as the covariance converges to 0 during the M step of the Expectation-Maximization (EM) algorithm. The ability to set prior distributions over cluster parameters prevents this collapse from happening. Additionally, since there are many factors which can cause small variations in the size of resources requested on a page, the ability to specify that a cluster should not fit the training data as tightly as possible can be of great use.

Another advantage offered by the Bayesian framework concerns inferring the number of Gaussian components (clusters) in the GMM. As the number of GMM components increases in the maximum likelihood approach, the likelihood of training data given the model will continue to increase as well. This is because the model is able to fit the data increasingly closely. This creates a problem with over-fitting the data since there is no clear way to determine the correct number of clusters. In contrast, a Bayesian approach naturally introduces a trade-off between model complexity and how well the model fits the data, thereby helping to discourage over-fitting.

Although the Bayesian framework provides the advantage of allowing prior distributions over parameter values, this comes at the expense of adding the GMM parameters to the model as latent variables. This additional complexity prevents a straightforward application of EM as there is no known way to directly calculate the posterior distribution over all latent variables (including GMM parameters) during the E step of the EM algorithm.

In place of EM, we employ *variational inference* in order to fit the model to the data [2]. Variational inference requires that the distribution over latent variables be specified as a product of several factors, such that each factor can be optimized separately while the others are held constant. This

restriction makes successive iterations feasible, although at the cost of accuracy since the posterior distribution has been artificially constrained and can no longer be an arbitrary function. Similar to EM, variational inference terminates when the marginal improvement incurred by each iteration to the likelihood of the observed data diminishes to less than a preset *convergence threshold*. Consequently, lower values of the convergence threshold produce models which fit the training data more closely, although at the cost of additional computation.

Although each invocation of the clustering algorithm is guaranteed to converge and terminate, the algorithm must be invoked multiple times. Recall that this is necessary in part to determine the number of components in the GMM, as invocations with too many or too few clusters will tend to produce lower likelihood of the training data. Multiple invocations are also necessary since the clustering technique involves an initial randomized assignment of points to clusters and often converges to local rather than global optima. Hence, successive invocations may produce improved results depending on the initial randomized assignment. After invoking the clustering algorithm with a range of number of clusters, the single invocation which produced the highest likelihood of the training data is selected and returned as the model.

Towards offsetting the cost of multiple invocations, we developed a heuristic which separates invocations of the clustering algorithm into three separate rounds. The heuristic limits computational cost in successive rounds by varying K, the number of components used in each invocation. The first round considers values of K ranging from 1 to a maximum determined by the following equation:

$$K_{max} = \begin{cases} N, & \text{if } N \leq 10 \\ 10 + \frac{N-10}{2}, & \text{if } 10 < N \leq 30 \\ 20 + \frac{N-30}{3}, & \text{if } 30 < N \leq 42 \\ 24, & \text{if } N > 42 \end{cases} \quad (1)$$

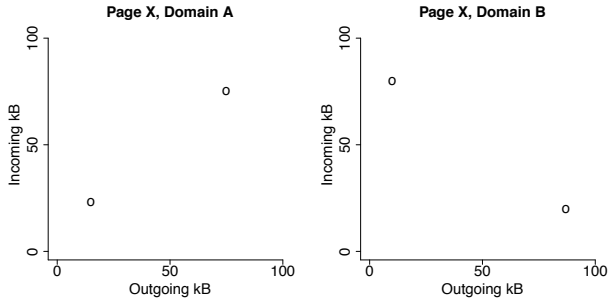
where N represents the maximum number of data points contributed by any single feature set to the set of points being clustered.

Once the first round of invocations has been completed, the range of K values for the second round is determined using a window of size 9 centered on the value of K which produced the highest data likelihood during the first round. For the third and final round, the invocation which produced the highest data likelihood from the first two rounds is identified, and a search is performed around a window of size 5 centered on the best performing value of K. In Section 6 we present an experimental justification of our heuristic design.

5.2 Basic Attack

Having described the type of feature used in analysis and the machine learning technique used to produce a model for each page, we now present a precise specification of the traffic model as well as the approach used to classify traffic given a set of models. We begin by extending our running example to illustrate the classification of traffic by an attacker.

To obtain an intuitive understanding of classification consider Figures 3 and 4. Recall that Figure 3 presents several (fabricated) feature sets, similar to what an attacker would have available during training. Figure 4 presents a single feature set as would be available to an attacker during an attack. The attacker's task is to identify whether the page which caused the features observed in Figure 4 is Page 1 or Page 2, as seen in Figure 3. Visual inspection reveals that the patterns observed in the feature set are most similar to those of Page 1, indicating that the unknown Page X depicted in Figure 4 is most likely Page 1.



(a) (b)

Figure 4: Fabricated feature set corresponding to a single visit to an unknown page.

In effect, our visual inspection has created a model for each page and each domain observed in Figure 3 and concluded that the samples observed in Figure 4 are more likely to have been generated by the models corresponding to Page 1 than the models corresponding to Page 2. This type of comparison is formalized in our attack, as we use the series of GMMs produced for each page to quantify the similarity of a traffic sample to training data for each page.

The exact likelihood of a feature set given a model for a specific page is defined below. Let several variables be defined as follows:

- Let X denote the entire set of features from a trace, with X^d denoting all features observed at domain d and X_i^d denoting ordered pair i seen at domain d.
- Let U denote the set of all pages for which there is a traffic model.
- Let $\Theta = \{\Theta^u \mid u \in U\}$ denote the attacker's entire model over all pages, where Θ^u denotes the model specifically associated with Page u.
- Let $Dom(X)$ denote the set of domains seen in a feature set, and $Dom(\Theta^u)$ denote the set of domains seen in the training data for Page u.
- Let $\Theta^u = \{(w_d, \Theta_d^u) \mid d \in Dom(\Theta^u)\}$, where w_d denotes the portion of training data for Page u seen at Domain d and Θ_d^u denotes the GMM corresponding to Domain d at Page u.
- Let $\Theta_d^u = (K, \Sigma, \mu, \pi)$, where K denotes the number of components in the GMM, and Σ_k, μ_k, π_k denote the

covariance, mean and weight of component k in the GMM.

Thus, the likelihood of feature set X given a model Θ^u for a specific page is defined as follows

$$P(X|\Theta^u) = \prod_{d}^{Dom(X)} P'(X^d|\Theta^u) \quad (2)$$

$$P'(X^d|\Theta^u) = \begin{cases} P''(X^d|\Theta^u_d) & \text{if } d \in Dom(\Theta^u) \\ P'''(X^d|\Theta^u) & \text{if } d \notin Dom(\Theta^u) \end{cases} \quad (3)$$

$$P''(X|\Theta^u_d) = \prod_{x}^{X} \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\Sigma_k, \mu_k) \quad (4)$$

$$P'''(X|\Theta^u) = \sum_{d}^{Dom(\Theta^u)} w_d P''(X|\Theta^u_d) \quad (5)$$

Notice that Equation 4 simply denotes the likelihood of a set of points X given a GMM Θ^u_d. Notice also that this approach is particularly amenable to caching because the absence of a particular feature will not directly cause a decrease in the likelihood of a feature set occurring.

In the event that $Dom(X) \not\subset Dom(\Theta^u)$, there will be traffic in X to domains for which there is no GMM in Θ^u. The likelihood of traffic to domains not in Θ^u is calculated by taking a weighted linear combination of the GMMs for each domain in Θ^u, where weights are determined by the fraction of training data contributed by the domain. This technique is seen in Equation 5. We assume this approach since one possible cause of $Dom(X)$ containing domains not seen in Θ^u is that content may be served via dynamic redirection, leading to domains observed during testing which were not observed during training.

Having calculated $P(X|\Theta^u)$ for each page $u \in U$, we predict that feature set X was generated by Page $\hat{u} \in U$ as follows

$$\hat{u} = \arg\max_{u} \; P(X|\Theta^u) \quad (6)$$

Unfortunately, the nature of GMMs is such that for $u \neq \hat{u}$, $P(X|\Theta^u)$ is often extremely small relative to $P(X|\Theta^{\hat{u}})$, in effect guaranteeing that feature set X would not be generated by any of the models Θ^u. This is an undesirable consequence of our modeling technique since in effect the model portrays higher confidence that $\hat{u}$ is the page being visited by the victim than is appropriate. In the next section we will explore ways to correct this bias, as well as ways to incorporate information about the different predictive accuracy of each domain.

5.3 Weighting Extension

In this section, we explore a technique for extending the attack described in Section 5.2 to weight the impact which each domain d has on traffic identification. Although we begin our discussion from the perspective of weighting domains, our technique will also be useful in assigning a likelihood to each page u rather than simply identifying the single page $\hat{u}$ which is most likely to have produced a feature set X.

Returning to our running example, recall that in Figure 3 traffic to Domain A is more consistent and tightly correlated than traffic to Domain B. This is a common consequence of Domain A serving core page content and Domain B serving dynamically generated content such as advertising. Frequently, the consequence of this difference in repeatability is that Domain A will more accurately predict traffic than Domain B. We would like for our attack to automatically recognize this difference and weight the impact of Domains A and B appropriately.

Figure 5 extends our example and demonstrates our technique for weighting the impact of each domain. To elucidate differences in accuracy, we consider eight separate pages, rather than two, served by Domains A and B. We first measure the accuracy of each domain by rank ordering the likelihoods $P'(X^d|\Theta^u)$ over all $u \in U$, and then checking to see at which position in the ordering the page u which actually produced sample X occurred. This is performed repeatedly over a set of samples X. Intuitively, this is akin to asking each domain to make predictions about which page generated X, and recording how many predictions are necessary in order to obtain the correct answer. As shown in Figure 5, we then fit a curve to the measured accuracy of predictions issued by each domain. Curve fitting helps to smooths over variations in observed accuracy, ensures that likelihood of each prediction will consistently decrease, and guarantees that the likelihood of each prediction will be strictly positive. The unnormalized likelihood of a page u is now calculated by rank ordering a sample according to each domain and multiplying the values on the fit curve which page u receives at each domain.

This approach can be formalized as follows. Assuming N samples are used in evaluating the accuracy of each domain, let N^d_k denote the total number of times that model Θ^u corresponding to the correct page u was assigned likelihood of rank k. We define an accuracy function for domain d as follows

$$Accuracy_d(k) = \frac{N^d_k}{N} \quad (7)$$

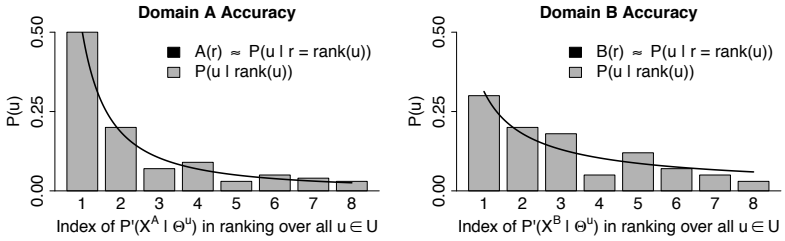
In general, we would expect the *Accuracy* function to be strictly decreasing as the attack should tend towards identifying correct URLs. In order to smooth out variations in the accuracy resulting from the data, we fit a curve of the form

$$Fit_d(k) = \frac{\alpha}{k^\beta} \approx Accuracy_d(k) \quad (8)$$

where d denotes the domain corresponding to the curve and α and β denote the parameters of the curve itself. Notice that the value $Fit_d(k)$ predicts the likelihood that the page predicted at rank k using data from domain d will be correct. To combine data from different domains to produce a single prediction, we define additional functions *Scores* and *Rank* as follows

$$Scores(X, d) = sorted(\{P'(X^d|\Theta^u) \mid u \in U\}) \quad (9)$$

$$Rank(X, d, u) = Scores(X, d).indexof(P'(X^d|\Theta^u)) \quad (10)$$

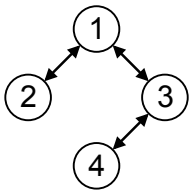


(a) (b)

Page	Domain A	Domain B	Potential
Page 1	$A(3) = .10$	$B(2) = .18$	.0180
Page 2	$A(8) = .03$	$B(5) = .09$	.0027
Page 3	$A(1) = .50$	$B(3) = .13$	.0650
Page 4	$A(4) = .07$	$B(6) = .08$	.0056
Page 5	$A(7) = .03$	$B(8) = .06$	.0018
Page 6	$A(2) = .19$	$B(1) = .31$	.0589
Page 7	$A(5) = .05$	$B(7) = .07$	.0035
Page 8	$A(6) = .04$	$B(4) = .10$	.0040

(c)

Figure 5: Demonstration of domain weighting approach. Figures 5a and 5b depict the measured accuracy of Domains A and B respectively, where each domain d predicts the page u which generated a feature set X by ordering $P'(X^d|\Theta^u)$ over all $u \in U$. Having determined accuracy of each domain, Figure 5c demonstrates how predictions are combined for a feature set containing multiple domains. For a hypothetical feature set, Domains A and B have rank ordered page likelihoods as [3, 6, 1, 4, 7, 8, 5, 2] and [6, 1, 3, 8, 2, 4, 7, 5], indicating that Domains A and B predict that pages 3 and 6 are respectively most likely to have produced the feature set. Notice that Page 3 ultimately retains the highest ranking as Domain A is more accurate than Domain B. Note that Potential is the unscaled likelihood.



(a)

	Sample 1	Sample 2	Sample 3
Actual	Page 1	Page 3	Page 4
Page 1	.0100	.0010	.0001
Page 2	.1000	.0100	.0010
Page 3	.0010	.1000	.0100
Page 4	.0001	.0001	.1000
Prediction	Page 2	Page 3	Page 4

(b)

Path	Independent Likelihood	HMM Likelihood
2 - 3 - 4	.0010	.0000
2 - 3 - 3	.0001	.0000
2 - 2 - 4	.0001	.0000
1 - 3 - 4	.0001	.0001

(c)

Figure 6: Performance improvement from Hidden Markov Model. Figure 6a depicts the link structure of a website with four pages. Figure 6b depicts the likelihood assigned to features extracted from three samples taken from the website, the actual page which generated the samples, and the prediction which results from considering the pages in isolation. Figure 6c presents the likelihoods of the correct browsing path of the three samples as well as all other paths which are equally or more likely when considering the samples both independently and using a HMM. Notice that the HMM correctly identifies all samples.

where the function *sorted* transforms an unordered set into a vector sorted in descending order and $indexof$ returns the index of an element in a vector.

This allows us to produce predictions which leverage the fact that some domains are more accurate than others. Together, Fit and $Rank$ allow us to predict the page u associated with a feature set X as follows

$$\Psi(u|X, Fit, Rank) = \prod_{d}^{Dom(X)} Fit_d(Rank(X, d, u)) \quad (11)$$

Notice that we specify a potential function Ψ rather than a probability distribution P. This approach does not specify a valid probability distribution over u since Ψ is unnormalized. Under the domain weighted model, the predicted page $\hat{u}$ is specified as follows

$$\hat{u} = \arg\max_{u} \ \Psi(u|X, Fit, Rank) \quad (12)$$

In order to use this approach to obtain a likelihood for each page u rather than simply the most likely page $\hat{u}$, a curve is fit to the accuracy of predictions made by all domains in the same way that a curve is fit to the accuracy of predictions made by a single domain. This extension is useful in the coming section, which will extend the domain weighting approach using a technique similar to a Hidden Markov Model in order to leverage the link structure of a page.

5.4 Sequential Data Extension

In this section, we discuss a final extension to our technique which utilizes the link structure of the page. Up to this point, we have considered all feature sets to be independent. In practice, we know that this assumption is false as the link structure of the website will restrict the sequence of pages which ultimately generate traffic.

We remove this assumption using an approach similar to a Hidden Markov Model (HMM). Recall that a HMM for a sequence of length N is defined by a set of latent variables $Z = \{z_n \mid 1 \leq n \leq N\}$, a set of observed variables $X = \{x_n \mid 1 \leq n \leq N\}$, transition matrix A such that $A_{i,j} = P(Z_{n+1} = j|Z_n = i)$, an initial distribution π such that $\pi_j = P(Z_0 = j)$ and an emission function $E(x_n, z_n) = P(x_n|z_n)$.

Figure 6 concludes our running example by presenting a case in which the use of a HMM increases the accuracy of traffic identification by ruling out identifications which are not possible given the link structure of the website.

Applied to our context, the HMM is configured as follows:

-2	-1	0	1	2
36	34	413	53	40

Table 2: Tally of difference values between the number of components in the best fit model and center of the search window in round 3.

-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
3	5	20	23	56	70	130	68	60	62	55	16	8

Table 3: Tally of difference values between the number of components in the best fit model and center of the search window in round 2.

- The latent variables z_n correspond to pages u modeled by the attacker
- The observed variables x_n correspond to observed feature vectors X
- The initial distribution π assigns an equal likelihood to all pages
- Letting $links(i)$ denote the set of pages linked to from page i, the transition matrix A is defined as $A_{i,j} = 1$ if $j \in links(i)$ and 0 otherwise
- The emission function $E(x_n, z_n) = \Psi(z_n|x_n, Fit, Rank)$

Note that neither the emission function E nor the transition matrix A represent valid probability distributions, as would be required by a directed graphical model. Instead, we treat the HMM as an undirected graphical model parameterized by potential functions. This alteration has no effect in practice as the max-sum algorithm is equally applicable to both directed and undirected graphical models.

To further increase accuracy, the emission function E can be replaced with the fit likelihood function described in the conclusion of Section 5.3. We refer to this as the Scaled HMM in our evaluation as the likelihood of each page u has been scaled according to the accuracy of the model.

6. EVALUATION

In this section we present our evaluation results. The evaluation was conducted in two phases, first on a smaller set of pages and subsequently on a larger set of pages. Since there is considerable computational cost associated with our attack, evaluation on a smaller set of pages allowed us to hone aspects of the attack such as convergence threshold and number of invocations of the training algorithm before operating on larger datasets. Accuracy on the larger set of pages ranged from 64% - 99% when examining portions of several common websites containing between 176 and 366 unique pages. For comparison, we were able to achieve 95% accuracy when identifying traffic corresponding to 402 website homepages while ignoring all information about packet destination.

We gathered traffic traces using a system similar to that described in Section 3 for gathering data about the effects of caching. We analyzed data using the clustering algorithm described in Section 5 implemented in `R`, along with the `multicore` package to enhance performance. Unless otherwise stated, evaluation was performed on `c1.xlarge` instances on Amazon EC2. Each instance offers 8 virtual cores for a total of 20 EC2 compute units and 7GB of memory. Note that one EC2 compute unit is approximately equivalent to the performance of a 2007 Xeon processor.

Training Time	Number of Iterations	Convergence Threshold	Model Accuracy (%)
1:23:19	1-4-4	50.0	59.1
2:45:52	1-4-4	5.0	61.1
4:50:11	1-4-4	0.5	61.2
2:53:05	2-8-8	50.0	58.5
5:12:06	2-8-8	5.0	62.5
8:43:58	2-8-8	0.5	63.3

Table 4: Impact of changes to number of iterations and convergence threshold on model accuracy and training time. Note that X-Y-Z denotes X, Y and Z iterations used in rounds 1, 2 and 3 respectively.

6.1 Small Scale Results

We conducted our small scale evaluation on a common financial website. The set of pages used included 72 distinct pages spread over nine separate sections within the bank's website, with some pages generated in response to form submissions or validation checks As with the gathering of caching data, the order of pages within each section remained constant and the order of sections themselves was randomized. For our small scale evaluation, we collected 48 sample traces over the course of 36 hours, with the cache sizes set to 0MB, 1MB, 8MB, and 64MB for 12 samples each. Recall that both the memory and disk caches are set to the specified cache size, yielding a total possible cache size twice as large, although the effective cache size is likely somewhere in the middle as Firefox may use the two types of cache differently. The cache was cleared and the browser restarted after each sample was gathered in order to restore a consistent state at the beginning of each trace.

Our small scale evaluation was designed to allow us to experiment with a wide range of factors, such as the browser cache settings, clustering convergence threshold, number of invocations of the clustering algorithm and the amount of training data used. By experimenting with these factors on a smaller scale, we are able to explore a wider range of factors and settings in order to decide which aspects to consider in our full scale evaluation. Since these factors relate only to the clustering technique and browser configuration, we do not include the HMM extension to the attack technique. Furthermore, we further simplify the evaluation by treating all data as being from a single domain.

We begin with an experimental justification of our clustering iteration heuristic. Tables 2 and 3 present the difference between the number of clusters at the center of the search window and the number of clusters which ultimately resulted in a model with the highest lower bound on data likelihood. Each table includes the results of 576 distinct model fittings, where each single fitting includes all invoca-

Training	Num. of Training Traces				Avg. Accuracy per Page (%)				Avg. Accuracy per Trace (%)			
Time (Avg.)	0MB	1MB	8MB	64MB	0MB	1MB	8MB	64MB	0MB	1MB	8MB	64MB
5:02:04	4	0	0	0	55.4	57.5	53.3	54.0	59.7	61.2	58.2	58.7
3:07:02	0	4	0	0	25.2	76.6	71.5	71.7	31.1	77.5	73.3	74.0
1:52:15	0	0	4	0	21.4	65.0	72.2	73.5	23.4	61.5	74.7	76.2
1:31:16	0	0	0	4	20.5	62.8	71.2	75.3	21.9	58.6	73.7	78.4
2:42:11	1	1	1	1	38.5	63.6	59.4	60.6	43.3	65.8	62.8	64.2

Table 5: Accuracy for varied cache sizes. Each row presents the average performance of three models. Samples for models in the first four rows were allocated on a rotating basis so that each model includes samples spanning the entire collection period. Samples for the final row were selected at random. Models were evaluated using all 44 remaining traces not used in training. Accuracy per Page denotes the average of the accuracy rates for each page. Accuracy per Trace denotes the average accuracy over all requests in a trace. Note that due to link structure some pages occur more than others in each trace.

tions of the clustering algorithm over each of the 3 rounds to produce a single best-fit model. Although it is provably impossible for the experimentally optimal value of K (the number of Gaussian components) to be outside of the window of size 5 used during round 3, or more than 2 outside of the window used on round 2, the concentration towards the center of each range provides support for our window size choices. Note that there is a slight bias towards larger numbers of clusters. This is due to websites which produced a large amount of traffic, resulting in a true optimum which was greater than the maximum number of clusters considered.

Although our heuristic does guide the value of K used for clustering invocations during each round of our search, the heuristic does not determine the correct number of iterations to perform in each round or the convergence threshold which should be used to terminate each invocation. Small convergence thresholds and larger numbers of iterations should always produce better results, although at the cost of greater computation.

In order to assess the effects of adjusting these parameters, we trained and evaluated models using three different convergence thresholds and two different sets of iteration counts for each round. The training data included one sample selected at random from each cache size and remained constant for all six combinations of convergence threshold and number of iterations. The resulting model was evaluated using all remaining samples. Table 4 shows the impact of varying numbers of iterations and convergence threshold on model accuracy as well as training time.

Having assessed the effects of convergence threshold and number of iterations on accuracy, we now examine the performance of our traffic analysis attack under a range of cache conditions. Table 5 presents the performance of models trained from samples with homogenous cache size as well as models trained from one sample of each cache size. All models were trained using 1 iteration in the first round and 4 iterations in the second and third rounds of training, with a convergence threshold of 5. Results are presented separately for each cache size in order to distinguish performance against various cache states.

Notice that each model performs the best against samples from the same cache size, and progressively worse against samples with progressively smaller cache sizes. This trend reveals insight into why our clustering based approach performs well in the presence of caching. Notice that the likelihood of a sample given a GMM based only on the likelihood of each individual point in the sample occurring. This presents a contrast with techniques that set a fixed number of times that each value of a feature is expected to occur. By allowing the likelihood of the sample to be determined only by the points which actually *occur* in the sample and assessing no direct penalty for points which are *absent* from the sample, the model is able to easily accommodate the absence of specific features in traffic as a result of cache hits.

Correspondingly, the likelihood of a sample is significantly decreased when a data point occurs which is sufficiently distant from the mean of each component of the GMM. This property accounts for the worsening performance in trials where the cache size for test data is smaller than the cache size of the samples used to train the model. Since items are likely to be cached during training but not cached during evaluation, the evaluation data inevitably contains data points which are far from any cluster mean that was fit to the training data. The data point appears unlikely for the page, and the likelihood is correspondingly decreased.

Additionally, notice that the traffic analysis technique actually performs worse with the cache disabled than with the cache enabled. This is counterintuitive since disabling the cache has the effect of increasing both traffic volume and regularity, both of which would be expected to increase attack accuracy. Our approach fails to capture this advantage due partly to disadvantages of clustering and partly to computational limitations. Since disabling the cache results in a greater number of data points in each sample, it is harder to develop models which fit the data closely. Models developed for cache size 0MB have on average 15.1 clusters, where as models developed for 1MB, 8MB and 64MB caches have 11.9, 11.5, and 10.3 clusters respectively. This suggests that computational limitations may restrict the number of clusters more often for models with cache size of 0MB. In addition to potential limitations in the number of clusters, higher convergence thresholds may cause clusters which do not fit data as tightly. This can result in models which assign moderate likelihood to a relatively broad range of samples, often resulting in false positives.

Having assessed the effects of number of iterations, conver-

Training Time	Number of Samples	Cache Disabled	Cache Enabled
2:44:57	3	23.3	69.9
6:57:55	6	27.0	80.1
7:56:07	9	30.3	81.5
12:05:56	12	29.0	87.4

Table 6: Attack accuracy for varying amounts of training data. Each row presents the average performance of two models. Samples from the 1MB, 8MB and 64MB cache sizes contributed evenly to to both the training and evaluation data in all rows. Note that the 9 and 12 sample models were trained using `m2.4xlarge` instances on EC2.

	k=1	k=2	k=3	k=4	k=5	k=6
model 0	87.2	92.2	93.8	94.7	95.2	95.7
model 1	87.6	92.6	94.7	95.6	95.9	96.1

Table 7: k-Accuracy allows the traffic analyzer to specify a set of k potential pages for each sample. We present both of the 12 sample models generated from four samples of each cache size.

gence threshold and cache state on accuracy, we now attempt to determine the impact of amount of training data as well as the maximum accuracy achievable with this attack. Table 6 shows the accuracy for models trained using samples from each cache size. All models were trained using 2 iterations in the first round and 8 iterations in the second and third rounds each, with a convergence threshold of 5. Since our traffic analysis technique focuses on HTTPS, a setting in which the cache is unlikely to be disabled, we present the evaluation results with the cache enabled and disabled separately. Given the large amount of third party content on the web such as Javascript libraries and advertising, the user's cache is unlikely to be entirely empty even the first time that the user visits a site. Notice that increases in the amount of training data provide the greatest accuracy improvements with the least additional computation relative to both decreases in the convergence threshold and increases in the number of clustering algorithm invocations.

Table 7 presents the k-Accuracy of each of the models trained from 12 samples shown in Table 6. Notice that when the traffic analyzer is allowed a second guess at the contents of the encrypted traffic, accuracy increases from 87% to above 92%. Since pages judged to be similar by the traffic analyzer tend to contain similar content, these "close" results can still be of use to an attacker. In our evaluation, the bank website advertised multiple credit cards, often on web pages which had very similar layouts. Although the exact page was occasionally mistaken, the attacker is still correct in believing that the user is viewing information related to credit cards.

6.2 Full Scale Results

We now present the results of a full scale evaluation using a larger set of pages. Having observed the effects of variations in the number of iterations, we perform the full scale evaluation using 1-4-4 for the number of iterations (indicating 1 iteration in the first round, and 4 iterations in the second and third rounds of model fitting). Additionally, we use a convergence threshold of 5.0 and 8 traces as training data. Since whether the cache was enabled or disabled appeared to have larger effect than the specific cache size, we leave the cache enabled to the default size during evaluation.

Our final evaluation uses three common websites, with one having a medical focus, one having a legal focus, and one having a financial focus. All websites included dynamically generated content, such as either user-specific forms or advertising. Since the HMM extension to our attack considers the link structure of the website, each site we selected was chosen to have a different link structure. We describe these structures and the websites in more detail below.

In preparing the bot for the medical website, we selected the pages describing 26 different diseases on the site for inclusion in our analysis, which each disease including on average 14 pages dedicated to the disease. We assume that the user visits a single disease within the site, and visits all pages associated with that disease twice during the browsing session. Furthermore, we assume that the link structure of the site allows the user to move arbitrarily between all pages associated with a disease, even though the actual link structure does not permit this behavior. This allows the medical website to represent a link graph which is composed of a series of cliques which are fully separated from each other. This property is particularly challenging to our attack since a HMM will only be useful for discerning which clique the traffic is within but can not help to distinguish pages within the same clique, given the uniform transition probability. This discrepancy is reflected in the fact that the attack identified the correct disease 99.5% of the time while identifying the correct page only 70% of the time.

For the legal website, we created a bot which was capable of creating a series of legal forms. The bot was given a dictionary of common names, addresses, etc. in order to complete short text fields. The bot also filled long text fields by randomly selecting a message length between 50 and 300 words using a uniform distribution and filling the message with words randomly selected from a dictionary of the 1000 most common English words. Lastly, the bot set checkboxes and radio buttons at random. Since the legal website was composed of a series of forms, this resulted in a link graph structure which more closely resembled a series of arcs, where each arc began at the homepage, ended at the payment page, and corresponding to filling out a separate legal form. Although some arcs did contain branches created by conditional completion of portions of the form, the overall structure allowed much less variation in browsing than the medical website.

The financial website was designed to more closely model average website structure, with a connected link graph allowing the user to navigate between any two pages in the graph. By constructing a link graph with smaller cliques, the benefit of the HMM is amplified since the total number of pages which could come between any known pair of pages is smaller. Note that the financial website presented considerable difficulty in analysis since common use of redirections, multiple URLs aliasing the same webpage, and dynamically differing content (as seen in A/B testing) created consid-

Website	Domain Weighting	Number of Pages	k=1	k=5	k=10
legal	weighted	176	68.0	90.8	94.3
legal	unweighted	176	72.6	85.9	89.5
medical	weighted	366	57.1	84.8	90.3
medical	unweighted	366	55.9	72.8	76.7
financial	weighted	291	46.9	65.7	74.1
financial	unweighted	291	46.3	59.9	65.1

Table 8: k-Accuracy allows the traffic analyzer to specify a set of k potential pages for each sample. This table presents the k-Accuracy of the model predictions without support from a HMM.

Website	Domain Weighting	HMM (unscaled)	HMM (scaled)
legal	weighted	97.7	98.5
legal	unweighted	85.3	95.6
medical	weighted	70.0	70.0
medical	unweighted	65.3	65.3
financial	weighted	65.3	64.4
financial	unweighted	43.7	53.5

Table 9: Final results on each webpage using both weighted and unweighted models.

erable difficulty in labeling traffic samples. This behavior presented a significant obstacle to analysis by both causing the same page to have multiple labels and potentially for the same label to be associated with multiple pages. Both of these effects could be eliminated with additional work on the part of the attacker and would only improve results.

Table 8 presents the k-Accuracy seen at each site in the full scale evaluation. Notice that although the benefit from weighting domains is minimal for k=1, the benefit increases with the value of k. This is particularly helpful when used in combination with a HMM, as the HMM will use the context of surrounding requests to narrow the scope of likely pages.

Table 9 presents the accuracy of the full attack technique, including the use of a HMM. Notice that the performance gain using the HMM is much greater for the legal website, as the link structure of this website increases the significance of the sequential nature of the data.

As a means of comparing our approach with others stated in the literature, we also describe the accuracy of our attack when evaluated using website homepages. We selected the top 500 domains as measured by Alexa and removed duplicates (such as the Google homepage in many languages) to arrive at 402 unique homepages. In order to remove the benefit of being able to observe the destination of traffic, we performed our analysis treating all traffic as belonging to a single domain. We also considered all traces in isolation, making no use of a HMM in the analysis. Even with these stipulations, we were still able to achieve 95% accuracy, demonstrating both that our technique performs strongly when compared with existing approaches and that traffic analysis of pages within a single website is considerably harder than traffic analysis of different homepages.

7. CONCLUSION

In this thesis, we review evaluations in related work relating to traffic analysis, discuss and propose improved evaluation methodologies, and present and evaluate a traffic analysis attack designed to perform well under realistic settings when comparing pages within the same website. By leveraging differences in the destination of traffic as well as the sequential nature of browsing caused by link structure, we are able to achieve accuracy ranging from 64% - 99% over subsets of common websites. By comparison, we are able to achieve 95% accuracy when evaluating our approach using website homepages, further affirming the potential for our technique and highlighting the difference in difficulty when comparing website homepages and pages within a single website.

8. REFERENCES

[1] Torbutton faq. `https://www.torproject.org/torbutton/torbutton-options.html.en`, Accessed May 2012.

[2] Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[3] Bissias, G., Liberatore, M., Jensen, D., and Levine, B. N. Privacy Vulnerabilities in Encrypted HTTP Streams. In *Proc. Privacy Enhancing Technologies Workshop* (May 2005).

[4] Cheng, H., and Avnur, R. Traffic analysis of ssl encrypted web browsing. `http://www.cs.berkeley.edu/~daw/teaching/cs261-f98/projects/final-reports/ronathan-heyning.ps`, 1998.

[5] Coull, S. E., Collins, M. P., Wright, C. V., Monrose, F., and Reiter, M. K. On web browsing privacy in anonymized netflows. In *Proc. USENIX Security* (2007).

[6] Danezis, G. Traffic analysis of the http protocol over tls. `http://research.microsoft.com/en-us/um/people/gdane/papers/TLSanon.pdf`.

[7] Dyer, K. P., Coull, S. E., Ristenpart, T., and Shrimpton, T. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *Proc IEEE S&P* (2012).

[8] Herrmann, D., Wendolsky, R., and Federrath, H. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naT̈ve-bayes classifier. In *Proc. of the ACM workshop on Cloud computing security* (2009).

[9] Hintz, A. Fingerprinting websites using traffic analysis. In *Proc. Privacy Enhancing Technologies Conference* (2003).

[10] Liberatore, M., and Levine, B. N. Inferring the source of encrypted http connections. In *Proc. ACM CCS* (2006).

[11] Panchenko, A., Niessen, L., Zinnen, A., and Engel, T. Website fingerprinting in onion routing based anonymization networks. In *Proc. ACM Workshop on Privacy in the Electronic Society* (2011).

[12] Sun, Q., Simon, D. R., Wang, Y.-M., Russell, W., Padmanabhan, V. N., and Qiu, L. Statistical identification of encrypted web browsing traffic. In *Proc. IEEE S&P* (2002).