On Web Browsing Privacy in Anonymized NetFlows

S. E. Coull[*]   M. P. Collins[†]   C. V. Wright[*]   F. Monrose[*]   M. K. Reiter[†]

[*]Johns Hopkins University
{*coulls,cwright,fabian*}*@cs.jhu.edu*

[†]Carnegie Mellon University
mcollins@cert.org, reiter@cmu.edu

Abstract

Anonymization of network traces is widely viewed as a necessary condition for releasing such data for research purposes. For obvious privacy reasons, an important goal of trace anonymization is to suppress the recovery of web browsing activities. While several studies have examined the possibility of reconstructing web browsing activities from anonymized packet-level traces, we argue that these approaches fail to account for a number of challenges inherent in real-world network traffic, and more so, are unlikely to be successful on coarser Net-Flow logs. By contrast, we develop new approaches that identify target web pages within anonymized NetFlow data, and address many real-world challenges, such as browser caching and session parsing. We evaluate the effectiveness of our techniques in identifying front pages from the 50 most popular web sites on the Internet (as ranked by alexa.com), in both a closed-world experiment similar to that of earlier work and in tests with real network flow logs. Our results show that certain types of web pages with unique and complex structure remain identifiable despite the use of state-of-the-art anonymization techniques. The concerns raised herein pose a threat to web browsing privacy insofar as the attacker can approximate the web browsing conditions represented in the flow logs.

1   Introduction

Recently, significant emphasis has been placed on the creation of anonymization systems to maintain the privacy of network data while simultaneously allowing the data to be published to the research community at large [23, 24, 17, 9, 22]. In general, the goals of anonymization are (*i*) to hide structural information about the network on which the trace is collected, so that disclosing the anonymized trace does not reveal private information about the security posture of that network, and (*ii*) to prevent the assembly of behavioral profiles for users on that network, such as the web sites they browse.

Our goal in this paper is to evaluate the strength of current anonymization methodology in achieving goal (*ii*). Specifically, we focus on providing a realistic assessment of the feasibility of identifying individual web pages within anonymized NetFlow logs [4]. Our work distinguishes itself from prior work by operating on flow-level data rather than packet traces, and by carefully examining many of the practical concerns associated with implementing such identification within real network data. Previous work has focused on methods for web page identification within encrypted or anonymized packet trace data utilizing various packet-level features, such as size information, which cannot be readily scaled to flow-level data. Rather than assume the presence of packet-level information, our work instead focuses on the use of flow-level data from NetFlow logs to perform similar identification. Since NetFlow data contains a small subset of the features provided in packet traces, we are able to provide a general method for identifying web pages within both packet trace and NetFlow data. Also, use of NetFlow data is becoming more commonplace in network and security research [13, 21, 33, 5].

More importantly, our primary contribution is a rigorous experimental evaluation of the threat that web page identification poses to anonymized data. Though previous work has provided evidence that such identification is a threat, these evaluations do not take into account several significant issues (e.g., dynamic web pages, browser caching, web session parsing, HTTP pipelining) involved with the application of deanonymizing techniques in practice. To overcome these obstacles to practical identification of web pages, we apply machine learning techniques to accommodate variations in web page download behavior[1]. Furthermore, our techniques can parse and identify web pages even within multiple interleaved flows, such as those created by tabbed browsing, with no additional information. The crux of our identifi-

cation method lies in modeling the web servers which participate in the download of a web page, and using those models to find the corresponding servers within anonymized NetFlow data. Since the behavior of each server, in terms of the flows they serve, is so dynamic, we apply kernel density estimation techniques to build models that allow for appropriate variations in behavior.

Simply finding web servers is not enough to accurately identify web pages, however. Information such as the order in which the servers are contacted, and which servers are present can have significant impact on the identification of web pages. In fact, the ordering and presence of these servers may change based on various download scenarios, such as changes in browser cache or dynamic web page content. To capture these behaviors, we formalize the game of "20 Questions" as a binary Bayes belief network, wherein questions are asked to narrow the possible download scenarios that could explain the presence of a web page within the anonymized data. As such, our approach to web page identification begins with identifying likely servers and then employs the binary Bayes belief network to determine if those servers appropriately explain the presence of the targeted web page within the data.

Lastly, the evaluation of our techniques attempts to juxtapose the assumptions of closed world scenarios used in previous work to the realities of identifying web pages in live network data. The closed world evaluation of data collected through automated browsing scripts within a controlled environment was found to perform well — detecting approximately 50% of the targeted web pages with less than 0.2% false detections. In more realistic scenarios, however, true detection and false detection rates varied substantially based upon the type of web page being identified. Our evaluation of data taken through controlled experiments and live network captures shows that certain types of web pages are easily identifiable in real network data, while others maintain anonymity due to false detections or poor true detection rates. Additionally, we show the effects of locality (i.e., different networks for collecting training and testing data) on the detection of web pages by examining three distinct datasets taken from disparate network environments. In general, our results show that information leakage from anonymized flow logs poses a threat to web browsing privacy insofar as an attacker is able to approximate the basic browser settings and network conditions under which the pages were originally downloaded.

2   Background and Related Work

Network trace anonymization is an active area of research in the security community, as evidenced by the ongoing development of anonymization methods (e.g., [9, 23, 30]) and releases of network data that they enable (e.g., [26, 7]). Recently, several attacks have been developed that illustrate weaknesses in the privacy afforded by these anonymization techniques. In particular, both passive [6] and active attacks [2, 3] have shown that deanonymization of public servers and recovery of network topology information is possible in some cases. Until now, however, an in-depth examination of the extent to which the privacy of web browsing activities may also be at risk has been absent.

It would appear that existing approaches for inferring web browsing activities within encrypted tunnels [19, 32, 11, 1, 18, 8]) would be directly applicable to the case of anonymized network data—in both cases, payload and identifying information (e.g., IP addresses) for web sites are obfuscated or otherwise removed. These prior works, however, assume some method for unambiguously identifying the connections that constitute a web page retrieval. Unfortunately, as we show later, this assumption substantially underestimates the difficulty of the problem as it is often nontrivial to unambiguously delineate the flows that constitute a single page retrieval. The use of NetFlow data exacerbates this problem. Furthermore, as we show later, there are several challenges associated with the modern web environment that exacerbates the problem of web page identification under realistic scenarios.

To our knowledge, Koukis et al. [14] present the only study of web browsing behavior inference within anonymized packet traces, which anticipates some of the challenges outlined herein. In their work, however, the authors address the challenges of parsing web page downloads from packet traces by using packet inter-arrival times to delineate complete sessions. Though this delineation can be successful in certain instances, there are several cases where time-based delineation alone will not work (e.g, for interleaved browsing). In this paper, we address several challenges beyond those considered by Koukis et al. and provide a more in-depth evaluation that goes further than their exploratory work. Moreover, our work differs from all prior work on this problem (of which we are aware) in that it applies to flow traces, which offer far coarser information than packet traces.

3   Identifying Web Pages in Anonymized NetFlow Logs

The anonymized NetFlow data we consider consists of a time-ordered sequence of records, where each record summarizes the packets sent from the server to the client within a TCP connection. These unidirectional flow records contain the source (server) and destination (client) IP addresses, the source and destination port

numbers, timestamps that describe the start and end of each TCP connection, and the total size of the traffic sent from the source to the destination in the flow (in bytes). The NetFlow format also contains a number of other fields that are not utilized in this work. For our purposes, we assume that the anonymization of the NetFlow log creates consistent pseudonyms, such as those created by prefix-preserving anonymization schemes [9, 23], for both the source and destination IP addresses in these records. Furthermore, we assume that the NetFlow data faithfully records TCP traffic in its entirety.

The use of consistent pseudonym addresses allows us to separate the connections initiated from different hosts, thereby facilitating per host examination. Additionally, we assume that port numbers and sizing information are not obfuscated or otherwise altered to take on inconsistent values since such information is of substantial value for networking research (e.g., [10, 29, 12]). The unaltered port numbers within the flows allow us to filter the flow records such that only those flows originating from port 80 are examined[2].

Initially, we also assume that *web browsing sessions* (i.e., all flows that make up the complete download of a web page) can be adequately parsed from the NetFlow log. A similar assumption is made by Sun et al. [32] and Liberatore et al. [18]. Though previous work has assumed that web browsing session parsing algorithms are available, accurate web session parsing is, in fact, difficult even with packet traces and access to payload information [31, 15]. In §6, we return to the difficulty of parsing these sessions from real anonymized network data. By adopting the assumption (for now) that accurate web browsing session parsing can be done, it becomes possible to parse the complete NetFlow data into non-overlapping subsequences of flow records, where each subsequence represents a single, complete web browsing session for a client. Given the subsequent client web browsing sessions, our goal is to extract features that uniquely identify the presence of target web pages within the anonymized NetFlow data, and model their behavior in a manner that captures realistic browsing constraints.

3.1 Feature Selection

The most intuitive feature for discovering web pages in the anonymized NetFlow data is the sequence of flow sizes observed during a complete web browsing session. Each flow in the web browsing session is represented by an *index number* indicating its ordering in the session, and an associated flow size indicating the amount of data transferred during the flow. Naïvely, one would expect that the use of flow size, index pairs would suffice as a good distinguisher for web page identification. However, as Figure 1(a) shows, this is not the case. For instance,

notice that the front page of *msn.com* is fairly inconsistent in the number and size of flows, and there is a significant amount of overlap even among only these three examples. Since we are examining flows, the number of flows and their associated sizes are dependent on the manner in which the client requests objects, such as pictures or text. In many cases, the sequence in which the objects are downloaded may change due to dynamic web content, or the state of the client's browser cache may cause certain objects to be excluded. These changes to the client's download behavior cause *object drift* within the flows, where web page objects are downloaded in different flows or not downloaded at all. As a result, the number of flows and their respective sizes can vary widely, and are therefore a poor indicator of the identity of the web page in question.

An important observation regarding this inconsistency is that the size of any flow is regulated by the cumulative size of all the objects downloaded for the web page, less the size of all objects downloaded in prior flows. If a large flow early in the browsing session retrieves a significant number of objects, then the subsequent flows must necessarily become smaller, or there must be fewer flows overall. Conversely, a session of many small flows must necessarily require more flows overall. In fact, if we examine the cumulative perspective of web page downloads in Figure 1(b), we find that not only are these sites distinguishable, but that they take consistent paths toward their target cumulative size.

The existence of such paths and the inherent connection between flow size, index number, and cumulative size motivates the use of all three features in identifying web pages. These three features can be plotted in 3-dimensional space, as shown in Figure 2(a), and the path taken in this 3-dimensional space indicates the behavior exhibited by the download of objects for a complete web browsing session. Figure 2(b) shows an example of web browsing session paths for the front pages of both *yahoo.com* and *msn.com* overlaid on the set of points taken over many web browsing sessions of *msn.com*'s front page. Clearly, the path taken by *yahoo.com* is distinct from the set of points generated from web browsing sessions of *msn.com*, while the *msn.com* path remains similar to past web browsing sessions.

Server sessions The use of flow size, index number, and cumulative size information can be further refined by considering the sequence of flows created from each web server in the web browsing session, which we denote as a *server session*. Notice that when we separate the flows for *msn.com* by the server that produced them, each server occupies a very distinct area of the 3-dimensional space, as shown in Figure 3. This refinement offers two benefits in identifying web pages.
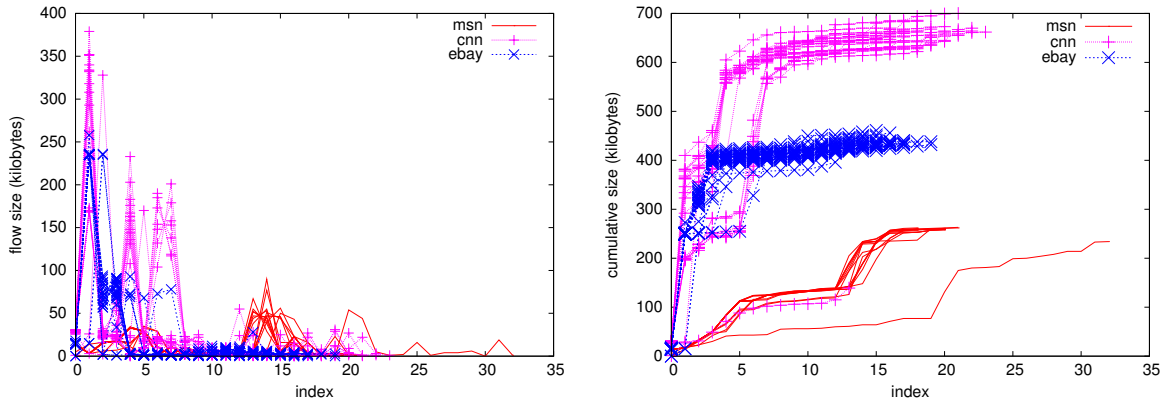
Figure 1: (a) Sequential and (b) cumulative views of page loads for *msn.com*, *cnn.com*, and *ebay.com* from a single client
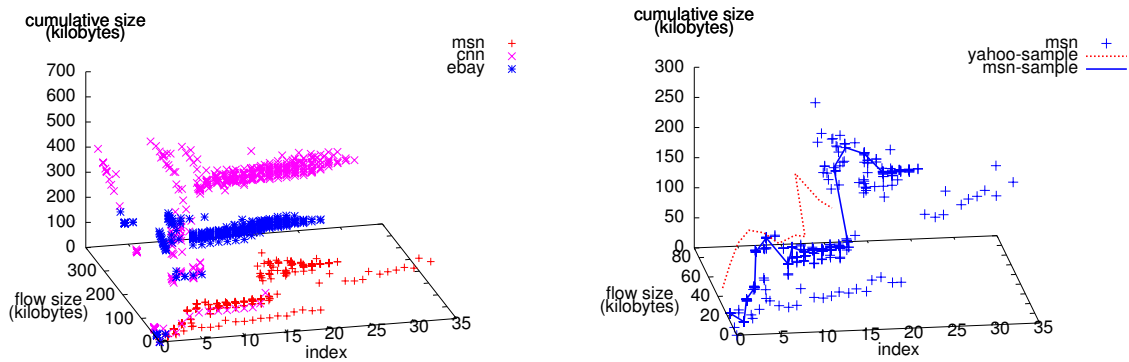

Figure 2: (a) 3-D view of page loads for *msn.com*, *cnn.com*, and *ebay.com* from a single client; (b) Regions for *msn.com* compared to sequences of *yahoo.com* and *msn.com* as downloaded by a single client

First, by abstracting the web browsing sessions to consist of individual server sessions, we can use the presence or absence of servers and their relative ordering to further differentiate web pages. The ordering of these web servers provides useful information about the structure of the web page since there is often a dependency between objects within the web page. For instance, the HTML of a web page must be downloaded before any other objects, and thus the first server contacted must be the primary web server. Second, by refining our flow information on a per server basis, we can create a fine grained model of the behavior of the web browsing session. If done correctly, the problem of identifying a web page within anonymized NetFlow data can be reduced to one of identifying the servers present within a given web browsing session based on the path created by the flows they serve, and the order in which the servers are contacted.

Logical servers Intuitively, we could simply use the flows served by each distinct web server IP address (which we refer to as a *physical* server) to create the 3-dimensional space that describes the expected behavior of that physical server in the web browsing session. However, the widespread use of Content Delivery Networks (CDNs) means that there may be hundreds of distinct physical web servers that serve the same web objects and play interchangeable roles in the web browsing session. These farms of physical servers can actually be considered to be a single *logical* server in terms of their behavior in the web browsing session. Therefore, the 3-dimensional models we build are derived from the samples observed from all physical servers in the logical server group.

Of course, the creation of robust models for the detection of web pages requires that the data used to create the models reflect realistic behaviors of the logical servers and the order in which they are contacted. There are a number of considerations which may affect the ability

(a) *msn.com* Server 1   (b) *msn.com* Server 2   (c) *msn.com* Server 3
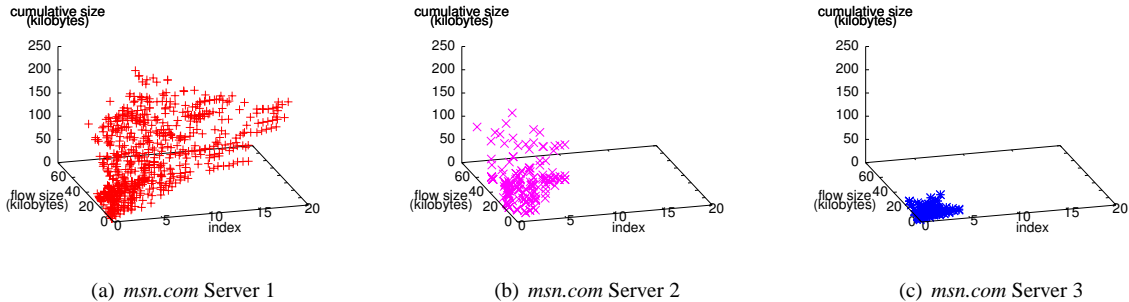
Figure 3: 3-D view of *msn.com* separated by server as observed by a single client

of data to accurately predict the behavior of web page downloads. These considerations are especially important when an attacker is unable to gain access to the same network where the data was collected, or when that data is several months old. Liberatore et al. have shown that the behavioral profiles of web pages, even highly dynamic web pages, remain relatively consistent even after several months [18], though the effects of web browser caching behavior and the location where the network data was captured have not yet been well understood.

4 An Automated Classifier for Web Pages in NetFlows

In this section, we address the problem of building automated classifiers for detecting the presence of target web pages within anonymized NetFlow data. Through the use of features discussed in §3, we create a classifier for each web page we wish to identify. The classifier for a target web page consists of the 3-dimensional spaces for each of its logical servers, which we formalize by using (*i*) kernel density estimates [28], and (*ii*) a series of constraints for those logical servers, formalized by a binary Bayes belief network [20]. The goal of the classifier is to attempt to create a mapping between the physical servers found in the anonymized web browsing session and the logical servers for the target web page, and then to use the mapping to evaluate constraints on logical servers for the web page in question. These constraints can include questions about the existence of logical servers within the web browsing session, and the order in which they are contacted by the client. If the mapping meets the constraints for the given web page, then we assume that the web page is present within the web browsing session; otherwise, we conclude it is not.

There are several steps, illustrated in Figure 4, that must be performed on the anonymized NetFlow logs in order to accurately identify web pages within them. Our first step is to take the original NetFlow log and parse

the flow records it contains into a set of web browsing sessions for each client in the log. Recall that our initial discussion assumes the existence of an efficient and accurate algorithm for parsing these web browsing sessions from anonymized NetFlow logs. These web browsing sessions, by definition, consist of one or more physical server sessions, which are trivially parsed by partitioning the flow records for each client, server pair into separate physical server sessions. The physical server sessions represent the path taken within the 3-dimensional space (i.e., flow size, cumulative size, and index triples) when downloading objects from the given physical server. At this point, we take the paths defined by each of the physical servers in our web browsing session, and see which of the logical servers in our classifier it is most similar to by using kernel density estimates [28]. Therefore, a given physical server is mapped to one or more logical servers based on its observed behavior. This mapping indicates which logical servers may be present within our web browsing session, and we can characterize the identity of a web page by examining the order in which the logical servers were contacted using a binary belief network. If we can satisfy the constraints for our classifier based upon the logical servers present within the web browsing session, then we hypothesize that an instance of the web page has been found. In §4.1 and §4.2, we discuss how the kernel density estimates and binary belief networks are created, respectively.

4.1 Kernel Density Estimation

In general, the kernel density estimate (KDE) [28] uses a vector of samples, $S = <s_1, s_2, ..., s_n>$ to derive an estimate for a density function describing the placement of points in some d-dimensional space. To construct a KDE for a set of samples, we place individual probability distributions, or *kernels*, centered at each sample point s_i. In the case of Gaussian kernels, for instance, there would be n Gaussian distributions with means of $s_1, s_2, ..., s_n$, respectively. To control the area covered by each distribution, we can vary the so-called *bandwidth* of the ker-
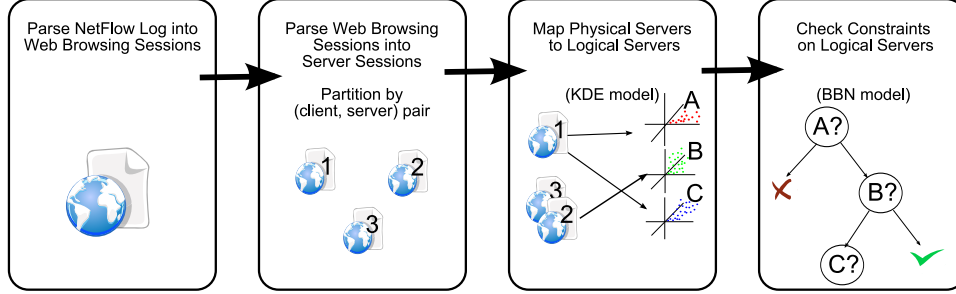
Figure 4: General overview of our identification process

nel. For a Gaussian kernel, its bandwidth is given by the variance (or covariance matrix) of the distribution. Intuitively, a higher bandwidth spreads the probability mass out more evenly over a larger space.

Unfortunately, determining the appropriate bandwidth for a given set of data points is an open problem. One approach that we have found to produce acceptable results is to use a "rule of thumb" developed by Silverman [28] and refined by Scott [27]. The bandwidth is calculated as $H_\Delta = N^{-1/(d+4)} \sigma_\Delta$, for each dimension $\Delta = 1...d$, where N is the number of kernels, d is the number of dimensions for each point, and σ_Δ is the sample standard deviation of the Δ dimension from the sample points in S. The primary failing of this heuristic is its inability to provide flexibility for multi-modal or irregular distributions. However, since this heuristic method provides adequate results for the problem at hand, we forego more complex solutions at this time.

Once the distributions and their associated bandwidths have been placed in the 3-dimensional space, we can calculate the probability of a given point, t_j, under the KDE model as:

$$P(t_j) = \frac{1}{n} \sum_{i=0}^{n-1} P_{s_i}(t_j) \qquad (1)$$

where $P_{s_i}(t_j)$ is the probability of point t_j under the kernel created from sample point s_i.

4.1.1 Application to Web Page Identification

To apply our anonymized NetFlow data to a KDE model, we take the set of paths defined by the triples of flow size, cumulative size, and index in each of the physical server sessions of our training data and use them as the sample points for our kernels. We choose the Gaussian distribution for our kernels because it allows us to easily evaluate probabilities over multiple dimensions. The bandwidth of these distributions is calculated as described previously, except that for the flow size and cumulative size dimensions, we take the average standard deviation across all index values as the bandwidth. Furthermore, we bound the bandwidth in each dimension such that it

is always ≥ 1 to allow for some minimum amount of variability.

To evaluate an anonymized physical server session on a particular KDE model, we simply evaluate each point in the path for that physical server session using Eqn. 1, and calculate the total probability of the given physical server session, t, as:

$$P(t) = \prod_{j=0}^{m-1} P(t_j) \qquad (2)$$

where t_j is the jth point in the physical server session t, and m is the number of flows in t. For classification, we consider any physical server session whose path has a non-zero probability (from Eqn. 2) under the given model to be a mapping between the logical server represented by the model and the physical server session being evaluated. Of course, it may be possible for physical server session t to follow a path that matches portions of several disjoint paths in the KDE model without exactly matching any paths in their entirety. Consequently, the path would achieve a non-zero probability despite the fact that it is not similar to any of the paths in the model. To prevent such situations from occurring, we apply linear interpolation to each pair of points representing consecutive flow indices on a path to create sample points at half index intervals.

The use of path probabilities alone, however, is insufficient in uniquely describing the behavior of the logical server. To see why, consider the case where we have a model for a logical server which typically contains ten or more total flows. It may be possible for a much smaller physical server session with one or two flows to achieve a non-zero probability despite the fact that there clearly was not an adequate amount of data transferred. To address this, we also create a KDE model for the final points in each sample path during training, denoted as *end points*. These end points indicate the requisite cumulative size and number of flows for a complete session with the given logical server. As before, we create distributions around each sample end point, and calculate the

probability of the physical server session's end point by applying Eqn. 1. Any anonymized physical server session which has a non-zero probability on both their path and their end points for a given logical server model is mapped to that logical server.

Automatically Building Logical Server KDE Models
To create our logical server models, we use two heuristics to group physical servers into logical server groups. First, if two physical servers in our training data use the same hostname and serve the exact same HTTP URL, we can assume they are the same logical server and their sample points can be merged into a single KDE model. Since we are in control of our training data, we can collect packet traces to find URL and hostname information before converting the data to NetFlow format to create the paths that will make up our KDE models. It is often the case, however, that different hostnames are used among physical servers in the same logical server group, and this may prevent some of the physical servers in our training data from being placed into the correct logical server groups.

To address this, we apply a second heuristic that merges these remaining physical servers by examining the behavior exhibited by their KDE models. If a randomly selected path and its end point from a given physical server's training data achieves non-zero probability on the KDE model of another physical server, then those two physical servers can be merged into a single logical server. The combination of these two heuristics allow us to reliably create KDE models that represent the logical servers found in the web browsing session. By applying the points found in an anonymized physical server session to each of the KDE models for a given web page, we can create candidate mappings from the anonymized physical server to the logical servers for the target web page.

4.2 Binary Bayes Belief Networks

As discussed earlier, we formalize the constraints on the logical servers using a binary Bayes belief network (BBN). In a typical Bayes network, nodes represent events and are inter-connected by directed edges which depict causal relations. The likelihood that a given event occurs is given by the node's probability, and is based on the conditional probability of its ancestors. In the binary Bayes belief network variant we apply here, we simply use a binary belief network where events have boolean values and the causal edges are derived from these values [20].

An example of a binary belief network is given in Figure 5, where the probability of event y is conditioned upon event $\neg x$. One way of thinking of this network is
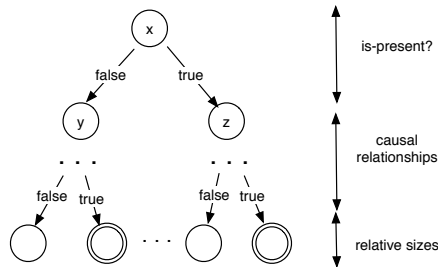

Figure 5: Example BBN

as a strategy for the game of "20 Questions" where the player attempts to identify an object or person by asking questions that can only be answered with 'Yes' or 'No' responses. Our binary belief network is simply a formalization of this concept (though we are not limited to ask 20 questions), where the answer to any question dictates the best strategy for asking future questions.

To create the belief network, we first decide upon a set of questions (or events) that we would like to evaluate within the data. In the context of web privacy, these events relate to the existence of logical servers, their causal relationships, and cumulative size. The belief network can be created *automatically* by first examining all possible existence and ordering events that occur within the training data. Next, from this set of events, we can simply select the event whose probability of being `True` in the training data is highest among all events. Having done so, the training data can then be partitioned into two groups: one group whose data has the value `True` for the selected event, and another whose value is `False` for that event. The selected event is then removed from the set of possible events and each partition of training data now selects another event from the remaining set whose probability on their respective data is highest.

This partitioning process is repeated recursively, allowing each branch to grow independently. A given branch halts its recursion when its conditional probability for an event is $< \epsilon$. The conditional probability threshold, ϵ, indicates the percentage of the training data that remains at a given leaf node, and therefore we stop our recursion before the tree becomes overfitted to our training data. Any leaf node that halts recursion with some amount of training data remaining is considered as an accepting node, and all other leaf nodes are labeled as rejecting nodes. Accepting nodes implement one additional check to ensure that the total size of all flows in the web browsing session is within $\pm 10\%$ of the total sizes observed during training.

5 A Closed-world Evaluation

To gauge the threat posed by the our web page identification techniques—and to place our results in context with prior work—we first provide an evaluation under a clean, closed world testing model. Prior work on this topic also focuses on the evaluation of identification techniques based on a controlled network environment, browsing a set of target web pages across an encrypted tunnel [32], through a proxy server [18], or within anonymized packet traces [14]. Each of these works, with the exception of [14], also assume that the web browsing session can be easily parsed from the stream of packets crossing the encrypted tunnel or proxy server. In what follows, we also adopt this assumption for this particular evaluation, though we will re-visit the inherent challenges with web browsing delineation in §6.

In short, our initial evaluation is considered under controlled environments similar to past work, but with two notable differences. First, in the scenarios we examine, there is substantially less data available to us than at the packet trace level; recall that NetFlows aggregate all packets in a flow into a single record. Second, rather than assuming that the client's browser cache is turned off, we attempt to simulate the use of caching in browsers in our training and testing data. The simulation of browser caching behavior was implemented by enabling the default caching and cookie policies within Mozilla Firefox™, and browsing to the sites in our target set at random. Of course, this method of cache simulation is not entirely realistic, as the probability of a cache hit is directly proportional to the frequency with which the user browses that web site. However, in lieu of making any assumptions on the distribution of web browsing for a given user, we argue that for the comparison at hand, the uniform random web browsing behavior provides an adequate approximation.

Data Collection   The data for our closed world evaluation was collected with the use of an automated script that used Firefox to randomly visit selected web pages from a set of target pages (with Adobe Flash and Javascript enabled). The web browser was set to the default caching and cookie policies to ensure the most realistic behavior possible in such a closed world environment. Specifically, the script first initiated a new Firefox instance, and opened new tabs within the single Firefox instance for each new web page visited. While these web pages were not loaded in parallel, several web sites automatically refresh themselves at given intervals, thus adding noise to our data whenever they appeared among one of the tabs of the active Firefox instance. Once four web pages were opened in the current Firefox instance, the browser was closed gracefully to allow the cache to

flush to disk, and a new Firefox instance was loaded to continue the random browsing. For each visit to a web page, we captured the packets for that web browsing session and recorded it to a separate trace. The packet captures were then converted into NetFlow logs by creating single flow records for each TCP connection in the session. Notice that the use of an automated browsing script allowed us to cleanly delineate between browsing sessions, as well as to simulate cache behavior through random browsing.

Our target pages were the front pages of the top 50 most popular sites as ranked by *alexa.com*. Additionally, we also collected information about the front pages of sites ranked 51-100 on the *alexa.com* list for use in providing robust evaluation of the false detection rates of our technique. Though we have chosen to evaluate our techniques on the top 100 sites, there is nothing inherent in their structure that differentiates them from other web pages. In fact, the same techniques are equally applicable in targeting any web page of the attacker's choosing.

The web pages were retrieved by running the automated browsing script on a host within the Johns Hopkins University network for a total of four weeks, creating a total of 18,525 web browsing sessions across all 100 web pages in our list of web pages. From this data, we select the first 90 web browsing sessions of our target web pages (i.e., those within the top 50 of the *alexa.com* ranking) as the training data for the creation of the kernel density estimate (see §4.1) and binary Bayes belief network models (see §4.2) that make up the profiles for each target web page. The remaining sessions are used as test data and are anonymized by replacing IP addresses within the NetFlow data with prefix-preserving pseudonyms according to the techniques described by [23]. Notice that since we assume that the web browsing sessions are easily parsed, we can simply use each web browsing session in our test data directly to determine if that web browsing session can be identified as any of the 50 web pages in our target set using the techniques described in §3.

Results   The results for this evaluation are given in Table 1. The analysis shows that our web page identification method performs reasonably well in the closed world environment. Though the overall true detection rate is only 48%, its associated false detection rate is exceptionally low at only 0.18% across all web pages. For comparison, using random guessing to identify web pages would yield an overall true detection rate of only 2%. Moreover, keep in mind that under the goals of network data anonymization, *no* inference of browsing behavior should be possible.

For ease of exposition, we also partition the 50 target web pages into canonical categories based on the primary

Category	Examples	TD (%)	FD (%)
Other	*passport.net, statcounter.com*	95.33	0.16
Social Networking and Dating	*match.com, myspace.com*	64.59	0.11
Search Engines and Web Portals	*msn.com, google.com*	60.42	0.16
Reference	*imdb.com, wikipedia.org*	54.17	0.02
Media	*flickr.com, youtube.com*	52.82	0.42
Corporate	*microsoft.com, apple.com*	48.95	0.15
Shopping	*amazon.com, ebay.com*	39.22	0.00
News	*cnn.com, nytimes.com*	28.74	0.06
Job Search	*monster.com, careerbuilder.com*	26.73	0.00
Sports	*foxsports.net, mlb.com*	20.74	0.00
Overall		48.89	0.18

Table 1: True and false detection rates for canonical categories in closed world test

function of the web site. Notice that the performance of the canonical classes varies based on the dynamism of the contents in the web page. For instance, some of the more difficult categories in terms of true detection are those whose front page content changes frequently, e.g, *cnn.com*. Conversely, pages with simple, static content, like *passport.net* or *google.com*, can be identified reasonably well. Moreover, those web sites with simple layouts and little supporting infrastructure also tend to fare worst with respect to false detections, while complex, dynamic sites have few, if any, false detections. These initial results hint at the fact that the ability to reliably identify web pages is connected with the complexity and dynamism of the web page. In what follows, we examine whether these results hold under a more realistic examination based on real world browsing.

6   Considerations for the Real World

The closed world evaluation in the previous section made several assumptions about the attacker's ability to parse web sessions and simulate caching behavior. Moreover, since both the training and testing data were collected at the same location, the effects of locality on the effectiveness of the identification techniques were not accounted for. These assumptions lead to a disconnect between the results of our closed world testing and those that can be expected in realistic attacks on anonymized data. In order to perform a rigorous evaluation of the *real* threats posed by such identification techniques, we must address several issues, including web browsing session parsing and caching behavior.

Web Browsing Session Parsing   One of the biggest concerns with the closed world evaluation in §5 is that there is an implicit assumption that parsing web sessions from live network data is a simple and accurate task. There has been extensive work in attempting to parse

packet traces into web browsing sessions, yet much of this work requires access to plaintext payloads, and results show that this parsing is not completely accurate [16, 31, 15]. To our knowledge, there is no prior art on performing similar parsing on NetFlow data. Koukis et al. attempted to use a heuristic of packet inter-arrival times to delineate sessions in packet traces, but their techniques were only able to correctly identify 8% of the web browsing sessions—underscoring the difficulty of the problem [14].

Fortunately, our kernel density estimate (KDE) and binary Bayes belief network (BBN) models can be modified to overcome the challenges of web browsing session parsing without significant changes to our identification process. In our previous evaluation, we assumed that the KDE and BBN models were given a subsequence of the original NetFlow log that corresponded to a complete web browsing session for a single client. For our real world evaluation, however, we remove this assumption. Instead, to parse the NetFlow log, we assume that all flows of a given web browsing session are clustered in time, and partition the NetFlow log into subsequences such that the inter-arrival times of the flows in the partition is $\leq \delta = 10$ seconds. This assumption is similar to that of Koukis et al. [14] and provides a coarse approximation to the web browsing sessions, but the resultant partitions may contain multiple web browsing sessions, or interleaved sessions.

Notice that we can simply use each of the physical servers within these partitions as input to the KDE models for a target web page to determine which logical servers may be present in the partition. Thus, we apply the flows from every physical server in our partitioned NetFlow data to the KDE models for our target page to create the logical server mappings. If a physical server in our partition does not map to a logical server, we ignore that physical server's flows and remove it from the partition. Thus, by removing these unmapped physical

servers, we identify a candidate web browsing session for our target site. Since the BBN operates directly on the mappings created by the KDE models, we traverse the BBN and determine if the web page is present based on the physical servers that were properly mapped. This technique for finding web browsing sessions is particularly robust since it can find multiple web pages within a single partition, even if these web pages have been interleaved by tabbed browsing.

Browser Cache Behavior   Another serious concern in our closed world evaluation is the variability of web browsing session behavior due to the client's browser cache. In our closed world evaluation, we created our models from data collected by an automated script that randomly browsed the front pages from among the top 100 sites according to *alexa.com*. The use of uniform random browsing with the default cache policy, however, does not accurately reflect the objects that would be cached by real clients. In reality, the client's browser cache would tend to hold more objects from the most frequently visited web pages, making the cache states highly specific to the client. Clearly, using our simulated caching data alone is not enough to create models that are able to detect both frequently and infrequently visited sites. To alleviate this shortcoming, we create a second set of training data by setting the browser's cache limit to 1.5GB. With such a large browser cache, objects should not be evicted from the cache even when we perform our random browsing, thereby allowing us to gain information about web browsing behaviors for our target sites when they are viewed frequently. The training data that we use to create our models now consists of 90 web browsing sessions of simulated cache data, and 64 browsing sessions of unlimited cache data for each target site. The procedure for building our models remains the same, except we now use the flow records from both cache scenarios.

Results   To provide a more realistic evaluation of the threat our identification techniques pose to anonymized NetFlow data, we re-examine its performance on three distinct datasets. First, we use the testing data from our closed world evaluation to measure the effect that the introduction of unlimited cache data and web session parsing have on the performance of our technique. Second, we capture web browsing sessions from different network providers in Maryland, and in Pennsylvania. By comparing the performance of our technique on these three datasets, we can glean insight into the effects of locality on the success of attacks on anonymized NetFlow data.

The effects that the changes to our models have on the performance of our technique are shown in Table

2. Clearly, the false detection rate increases substantially, but the true detection rate also increases. As in the closed world scenario, we find that the web pages with constantly changing content are more difficult to detect than static web pages, and that those sites with complex structure (i.e., many logical servers, and many flows) achieve a significantly lower false detection rate than those sites with simple structure. The substantial change in performance can be explained by the relaxation of the BBN constraints to allow for web browser session parsing. This relaxation allows any web browsing session where a subset of physical servers meets the remaining constraints to be identified, thereby causing the increase in both true detection and false detection rate. A more detailed analysis of the implications of these effects is provided in §7.

It is often the case that published network data is taken at locations where an attacker would not have access to the network to collect training data for her models, and so we investigate the effect that the change in locality has on the performance of our technique. The results in Table 2 show that there is, indeed, a drop in performance due to changes in locality, though trends in true detection and false detection rates still hold. In our evaluation, we noticed that the Johns Hopkins data used to train our web page models included a web caching server that caused significant changes in the download behavior of certain web pages. These changes in behavior in turn explain the significant difference in performance among data collected at different localities. It would appear that these results are somewhat disconcerting for a would-be attacker, since she would have to generate training data at a network that was different from where the anonymized data was captured. However, she could make her training more robust by generating data on a number of networks, perhaps utilizing infrastructure such as PlanetLab [25], though the effects of doing so on the performance of the technique are unknown. By including web page download behavior from a number of networks, she can ensure that the KDE and BBN models for each target web page are robust enough to handle a variety of network infrastructures.

7   A Realistic Threat Assessment

Finally, we provide a threat assessment by applying our technique to live data collected from a public wireless network at Johns Hopkins University Security Institute over the course of 7 days. From this data, we examine the expected real world accuracy of our techniques and discuss the features that make some web pages prime targets for identification.

| | | | Maryland | | Pennsylvania | |
Category	TD (%)	FD (%)	TD (Δ)	FD (Δ)	TD (Δ)	FD (Δ)
Other	100.00	10.71	-40.00	+1.98	-6.25	+10.74
Search/Web Portals	94.29	13.92	-35.47	+3.92	-26.60	-0.45
Social Network/Dating	75.75	9.02	-15.75	+4.93	+13.61	+9.95
Media	75.71	30.61	-3.71	+2.00	-30.97	+5.75
Corporate	75.00	11.64	-37.50	+2.55	-43.09	+3.06
Job Search	72.50	1.81	-47.50	+1.10	-44.91	+7.94
Shopping	71.00	4.89	-17.67	+3.55	-40.05	+1.37
News	70.71	2.50	-41.54	+0.15	-35.00	-0.14
Reference	67.00	14.64	-25.82	+9.59	+1.52	+19.13
Sports	66.67	26.57	-9.53	-6.23	+4.16	-11.87
Overall	75.58	13.28	-26.86	+1.47	-24.39	+4.59

Table 2: True and false detection rates for canonical categories in JHU data, and comparison to remote datasets

Category	Web Page	TD (%)	FD (%)
Reference	*imdb.com*	100.00	13.1
News	*nytimes.com*	0.00	0.06
	digg.com	61.76	0.35
	washingtonpost.com	9.09	0.01
	cnn.com	44.00	8.30
	weather.com	0.00	2.27
Search/Web Portals	*google.com*	28.57	22.31
	msn.com	0.00	6.18
	yahoo.com	60.98	0.89
Social Network/Dating	*facebook.com*	0.00	0.07
	myspace.com	25.00	65.55
Shopping	*ebay.com*	0.00	0.10
	amazon.com	0.00	0.78
Corporate	*usps.com*	0.00	4.66
Overall		33.65	9.09

Table 3: True and false detection rates for web pages in live network data

Results The results of our experiment on live data, shown in Table 3, provides some interesting insight into the practicality of identifying web pages in real anonymized traffic. In our results from local testing data collected via automated browsing, we observe that certain categories made up mostly of simple, static web pages (e.g., search engines) provide excellent true detection rates, while web pages whose content changes often (e.g., news web pages) perform significantly worse. Furthermore, categories of sites with complex structure (i.e., many logical servers) generally have exceptionally low false detection rates, while categories of simple sites with fewer logical servers produce extremely high false detection rates. Upon closer examination, not all web pages within a given category perform similarly despite having similar content and function. For instance, *cnn.com* and *nytimes.com* have wildly different performance in our live network test despite the fact that both pages have

rapidly changing news content.

To better understand this difference in our classifier's performance for different web pages, we examine three features of the page loads in our automated browsing sessions for each site: the number of flows per web browsing session, number of physical servers per web browsing session, and the number of bytes per flow. Our goals lie in understanding (*i*) why two sites within the same category have wildly different performance, and (*ii*) why simple web pages introduce so many more false detections over more complex web pages. To quantify the complexity of the web page and the amount of variation exhibited in these features, we compute the mean and standard deviation for each feature across all observations of the web page in our training data, including both simulated cache and unlimited cache scenarios. The mean values for each feature provide an idea of the complexity of the web page. For instance, a small average

number of physical servers would indicate that the web page does not make extensive use of content delivery networks. The standard deviations tell us how consistent the structure of the page is. These statistics offer a concise measure of the complexity of each web page, enabling us to objectively compare sites in order to determine why some are more identifiable than others.

Returning to the difference in true detection rates for *cnn.com* and *nytimes.com*, the front pages of both sites have constantly changing news items, a significant number of advertisements, and extensive content delivery networks. One would expect that since the web pages change so rapidly, that our accuracy in classifying them would be comparable. Instead, we find that we can detect nearly half of the occurrences of *cnn.com* in live network data, while we never successfully detect *nytimes.com*. In Table 4, we see that both web pages have similar means and standard deviations for both flow size and number of physical servers. This similarity is likely due to the nature of the content these two sites provide. However, the number of flows per web browsing session for *nytimes.com* is nearly double that of *cnn.com*. Moreover, *nytimes.com* exhibits high variability in the number of flows it generates, while *cnn.com* seems to use a fairly stable number of flows in all web browsing sessions. This variability makes it difficult to construct high-quality kernel density estimates for the logical servers that support *nytimes.com*, so our detector necessarily performs poorly on it.

Another interesting result from our live network evaluation is that some web page models appear to match well with almost all other pages, and therefore cause an excessive amount of false detections. For instance, Table 3 shows that *yahoo.com* has an exceptionally low false detection rate among all web pages in our live traffic, while *google.com* has one of the highest false detection rates. Both web pages, however, provide adequate true detection rates. In Table 5, we see that *google.com* and *yahoo.com* have very distinct behaviors for each of the features.

The metrics for our three features show that *google.com* transfers very little data, that there is almost always only one physical server in the web browsing session, and that there are normally only one or two flows. On the other hand, *yahoo.com* serves significantly more data, has a substantial number of physical servers, and causes the browser to open several flows per web browsing session. The web browsing sessions for *google.com* and *yahoo.com* both exhibit very little variability, though *yahoo.com* has more variability in its flow sizes due to its dynamic nature. Since both web pages have relatively low variability for all three features, they are both fairly easy for our techniques to detect, which corroborates our earlier claim that *cnn.com* is easy to detect because

of the relative stability of its features. However, since *google.com* is so simplistic, with only a single physical server and very few flows on average, its BBN and KDE models have very few constraints that must be met before the detector flags a match. Hence, many physical servers in a given NetFlow log could easily satisfy these requirements, and this causes the detector to produce an excessive number of false detection. By contrast, the models for *yahoo.com* have enough different logical servers and enough flows per session that it is difficult for any other site to fit the full description that is captured in the BBN and KDE models.

Discussion With regard to realistic threats to anonymized network data, these results show that there are certain web pages whose behavior is so unpredictable that they may be very difficult to detect in practice. Also, an attacker has little chance of accurately identifying small, simple web pages with our techniques. Complex web pages containing large content delivery networks, on the other hand, may allow an attacker to identify these pages within anonymized flow traces with low false detection rates. Finally, we have found that an attacker must consider the effects of locality on the training data used to create the target web page models, such as the presence of private caching servers or proxies. These locality effects adversely influence the true detection rates, but they might be overcome through diversification of the training data from several distinct locations. It is unclear how this diversification would affect the performance of our techniques.

When evaluating the threat that our web page identification attack poses to privacy, it is prudent to consider the information an attacker can reliably gain, possible practical countermeasures that might hamper such attacks, and the overarching goals of network data anonymization. With the techniques presented in this paper, an attacker would be able to create profiles for specific web pages of interest, and determine whether or not at least one user has visited that page, as long as those target web pages were of sufficient complexity. While the attacker will not be able to pinpoint which specific user browsed to the page in question with the technique presented in this paper, such information leakage may still be worrisome to some data publishers (e.g., web browsing to several risqué web pages).

There are, however, practical concerns that may affect the attacker's success aside from those described in this paper, such as the use of ad blocking software and web accelerators that dramatically alter the profiles of web pages. These web browsing tools could be used to make the attacker's job of building robust profiles more difficult, as the attacker would not only have to adjust for locality effects, but also for the effects of the particu-

Feature	cnn.com		nytimes.com	
	Mean	Std. Dev.	Mean	Std. Dev.
Number of Flows	18.44	4.21	30.69	10.62
Number of Physical Servers	12.79	2.27	15.32	4.14
Flow Size (KB)	568.20	286.95	692.87	298.73

Table 4: Comparison of mean and std. deviation for features of *cnn.com* and *nytimes.com*

Feature	google.com		yahoo.com	
	Mean	Std. Dev.	Mean	Std. Dev.
Number of Flows	1.73	0.56	9.02	3.02
Number of Physical Servers	1.03	0.17	5.25	1.79
Flow Size (KB)	13.64	10.37	219.51	187.26

Table 5: Comparison of mean and std. deviation for features of *google.com* and *yahoo.com*

lar ad blocking software or web accelerators. Moreover, while our evaluation has provided evidence that certain classes of web pages are identifiable despite the use of anonymization techniques, it is unclear how well the true detection and false detection rates scale with a larger target web page set. Therefore, our techniques appear to be of practical concern insofar as the attacker can approximate the behavior of the browsers and network environment used to download the web page.

8 Conclusion

In this paper, we perform an in-depth analysis of the threats that publishing anonymized NetFlow traces poses to the privacy of web browsing behaviors. Moreover, we believe our analysis is the first that addresses a number of challenges to uncovering browsing behavior present in real network traffic. These challenges include the effects of network locality on the adversary's ability to build profiles of client browsing behavior; difficulties in unambiguously parsing traffic to identify the flows that constitute a web page retrieval; and the effects of browser caching, content distribution networks, dynamic web pages, and HTTP pipelining. In order to accommodate for these issues, we adapt machine learning techniques to our problem in novel ways.

With regard to realistic threats to anonymized NetFlow data, our results show that there are certain web pages whose behavior is so variable that they may be very difficult to detect in practice. Also, our techniques offer an attacker little hope of accurately identifying small, simple web pages with a low false detection rate. However, complex web pages appear to pose a threat to privacy. Finally, our results show that an attacker must consider the effects of locality on the training data used to create the target web page models.

Our results and analysis seem to contradict the widely held belief that small, static web pages are the easiest target for identification. This contradiction can be explained by the distinct differences between closed world testing and the realities of identifying web pages in the wild, such as browser caching behavior and web browsing session parsing. On the whole, we believe our study shows that a non-trivial amount of information about web browsing behaviors is leaked in anonymized network data. Indeed, our analysis has demonstrated that anonymization offers less privacy to web browsing traffic than once thought, and suggests that a class of web pages can be detected in a flow trace by a determined attacker.

Acknowledgments

Notes

[1]Though machine learning techniques are certainly not the only method for handling variability in web pages, their application in this context seems to be intuitive.

[2]Note that even if this assumption did not hold there are still techniques that can be used to infer the presence of HTTP traffic (e.g, based on traffic-mix characteristics).

References

[1] BISSIAS, G., LIBERATORE, M., JENSEN, D., AND LEVINE, B. N. Privacy Vulnerabilities in Encrypted HTTP Streams. In *Proceedings of the 5th International Workshop on Privacy Enhancing Technologies* (May 2005), pp. 1–11.

[2] BREKNE, T., AND ÅRNES, A. Circumventing IP-Address Pseudonymization. In *Proceedings of the 3rd IASTED International Conference on Communications and Computer Networks* (October 2005).

[3] BREKNE, T., ÅRNES, A., AND ØSLEBØ, A. Anonymization of IP Traffic Monitoring Data – Attacks on Two Prefix-preserving Anonymization Schemes and Some Proposed Remedies. In *Proceedings of the Workshop on Privacy Enhancing Technologies* (May 2005), pp. 179–196.

[4] Cisco IOS NetFlow. `http://www.cisco.com/go/netflow`.

[5] COLLINS, M. P., AND REITER, M. K. Finding Peer-to-Peer File-Sharing Using Coarse Network Behaviors. In *Proceedings of the 11th European Symposium on Research in Computer Security* (September 2006), pp. 1–17.

[6] COULL, S., WRIGHT, C., MONROSE, F., COLLINS, M., AND REITER, M. Playing Devil's Advocate: Inferring Sensitive Information from Anonymized Network Traces. In *Proceedings of the 14th Annual Network and Distributed System Security Symposium* (February 2007). Available at: `http://www.cs.jhu.edu/~fabian/NDSS07.pdf`.

[7] CRAWDAD: A Community Resource for Archiving Wireless Data at Dartmouth. `http://crawdad.cs.dartmouth.edu`.

[8] DANEZIS, G. Traffic Analysis of the HTTP Protocol over TLS. Unpublished manuscript available at `http://homes.esat.kuleuven.be/~gdanezis/TLSanon.pdf` as of February 1, 2007.

[9] FAN, J., XU, J., AMMAR, M., AND MOON, S. Prefix-preserving IP Address Anonymization: Measurement-based Security Evaluation and a New Cryptography-based Scheme. *Computer Networks 46*, 2 (October 2004), 263–272.

[10] GUPTA, P., AND MCKEOWN, N. Packet Classification Using Hierarchical Intelligent Cuttings. In *Proceedings of Hot Interconnects VII* (1999), pp. 147–160.

[11] HINTZ, A. Fingerprinting Websites Using Traffic Analysis. In *Proceedings of the 2nd International Workshop on Privacy Enhancing Technologies* (April 2003), pp. 171–178.

[12] KARAGIANNIS, T., PAPAGIANNAKI, K., AND FALOUTSOS, M. BLINC: Multilevel Traffic Classification in the Dark. In *Proceedings of ACM SIGCOMM* (August 2005), pp. 229–240.

[13] KIM, M., KANG, H., AND HONG, J. Towards Peer-to-Peer Traffic Analysis Using Flows. In *Self-Managing Distributed Systems, 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management* (October 2003), pp. 55–67.

[14] KOUKIS, D., ANTONATOS, S., AND ANAGNOSTAKIS, K. On the Privacy Risks of Publishing Anonymized IP Network Traces. In *Proceedings of Communications and Multimedia Security* (October 2006), pp. 22–32.

[15] KREIBICH, C., AND CROWCROFT, J. Efficient Sequence Alignment of Network Traffic. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement* (October 2006), pp. 307–312.

[16] LEI, J. Z., AND GHORBANI, A. A. The Reconstruction of the Interleaved Sessions from a Server Log. In *Proceedings of the 17th Canadian Conference on AI* (May 2004), pp. 133–145.

[17] LI, Y., SLAGELL, A., LUO, K., AND YURCIK, W. CANINE: A Combined Conversion and Anonymization Tool for Processing NetFlows for Security. In *Proceedings of Tenth International Conference on Telecommunication Systems* (2005).

[18] LIBERATORE, M., AND LEVINE, B. Inferring the Source of Encrypted HTTP Connections. In *Proceedings of the ACM conference on Computer and Communications Security* (October 2006), pp. 255–263.

[19] MARTIN, D., AND SCHULMAN, A. Deanonymizing users of the safeweb anonymizing service. In *Proceedings of the 11th USENIX Security Symposium* (August 2002), pp. 123–137.

[20] NEAPOLITAN, R. *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*. John Wiley & Sons, Inc., 1989.

[21] NICKLESS, W., NAVARRO, J., AND WINKLER, L. Combining CISCO NetFlow Exports with Relational Database Technology for Usage Statistics, Intrusion Detection, and Network Forensics. In *Proceedings of the 14th Large Systems Administration Conference (LISA)* (December 2000), pp. 285–290.

[22] ØVERLIER, L., BREKNE, T., AND ÅRNES, A. Non-Expanding Transaction Specific Pseudonymization for IP Traffic Monitoring. In *Proceedings of the 4th International Conference on Cryptology and Network Security* (December 2005), pp. 261–273.

[23] PANG, R., ALLMAN, M., PAXSON, V., AND LEE, J. The Devil and Packet Trace Anonymization. *ACM Computer Communication Review 36*, 1 (January 2006), 29–38.

[24] PANG, R., AND PAXSON, V. A High-Level Environment for Packet Trace Anonymization and Transformation. In *Proceedings of the ACM Special Interest Group in Communications (SIGCOM) Conference* (August 2003), pp. 339–351.

[25] PETERSON, L., ANDERSON, T., CULLER, D., AND ROSCOE, T. A Blueprint for Introducing Disruptive Technology into the Internet. *SIGCOMM Computer Communication Reviews 33*, 1 (January 2003), 5964.

[26] PREDICT: Protected Repository for the Defense of Infrastructure Against Cyber Threats. `http://www.predict.org`.

[27] SCOTT, D. *Multivariate Density Estimation: Theory, Practice, and Visualization*. John Wiley & Sons, New York, 1992.

[28] SILVERMAN, B. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, 1986.

[29] SINGH, S., BABOESCU, F., VARGHESE, G., AND WANG, J. Packet Classification Using Multidimensional Cutting. In *Proceedings of ACM SIGCOMM* (August 2003), pp. 213–224.

[30] SLAGELL, A., LI, Y., AND LUO, K. Sharing Network Logs for Computer Forensics: A New Tool for the Anonymization of NetFlow Records. In *Proceedings of Computer Network Forensics Research Workshop* (September 2005).

[31] SPILIOPOULOU, M., MOBASHER, B., BERENDT, B., AND NAKAGAWA, M. A Framework for the Evaluation of Session Reconstruction Heuristics in Web-Usage Analysis. *INFORMS Journal on Computing 15*, 2 (April 2003), 171–190.

[32] SUN, Q., SIMON, D. R., WANG, Y., RUSSELL, W., PADMANABHAN, V. N., AND QIU, L. Statistical Identification of Encrypted Web Browsing Traffic. In *Proceedings of the IEEE Symposium on Security and Privacy* (May 2002), pp. 19–31.

[33] XIE, Y., SEKAR, V., MALTZ, D., REITER, M. K., AND ZHANG, H. Worm Origin Identification Using Random Moonwalks. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy* (May 2005), pp. 242–256.