Effects of network trace sampling methods on privacy and utility metrics

Phil Fazio, Keren Tan, and David Kotz
Dartmouth College

***Abstract*—Researchers choosing to share wireless-network traces with colleagues must first anonymize sensitive information, trading off the removal of information in the interest of identity protection and the preservation of useful data within the trace. While several metrics exist to quantify this privacy-utility tradeoff, they are often computationally expensive. Computing these metrics using a *sample* of the trace could potentially save precious time. In this paper, we examine several sampling methods to discover their effects on measurement of the privacy-utility tradeoff when anonymizing network traces. We tested the relative accuracy of several packet and flow-sampling methods on existing privacy and utility metrics. We concluded that, for our test trace, no single sampling method we examined allowed us to accurately measure the tradeoff, and that some sampling methods can produce grossly inaccurate estimates of those values. We call for further research to develop sampling methods that maintain relevant privacy and utility properties.**

I. Introduction

Wireless-network researchers depend on the availability of traffic traces collected from live production networks. It is difficult to collect such traces, requiring permission from network operators and installation of extensive infrastructure. Due to this scarcity of data, it becomes extremely important for the community of network researchers to share available traces among several research projects; this has resulted in the creation of data archive resources such as CRAWDAD [1].

Those that choose to share trace data with their colleagues encounter the additional burden to remove or otherwise anonymize sensitive information (e.g., identities of network users or details of the network topology). This "sanitization" naturally involves a tradeoff between the removal of information to fulfill privacy requirements and the preservation of information that may be useful in later analysis. To streamline the process of trace sanitization and to better analyze this tradeoff, we proposed the NetSANI (Network Trace Sanitization and ANonymization Infrastructure) framework and API [2]. Using either existing or user-defined metrics, the framework is designed to allow the researcher to analyze an anonymized trace to determine whether it meets pre-specified privacy and utility goals. In computing the metrics, NetSANI works with a *sample* of the collected trace with the goal of saving precious time and resources when developing an anonymization scheme.

Trace sampling is hardly a new concept; the benefits of various sampling techniques have been analyzed with respect to anomaly detection [3], [4], [5], [6], computation of traffic flow statistics [7], [8], network management [9], and sample space efficiency [10]. However, little or no attention has been paid to the effects of sampling when analyzing anonymized network traces.

978-1-4673-0298-2/12/$31.00

In this paper, we apply several well-known sampling methods to wireless-network traces and analyze their effect on some existing privacy and utility metrics. By comparing these results to the same analysis on the unsampled traces, we seek to discover which sampling methods produce the most accurate estimates of the tradeoff between privacy and utility and examine any trends in the experiments. We concluded that, for our test trace, no single sampling method we examined allowed us to accurately measure the tradeoff, and that some sampling methods produce inaccurate estimates of those values.

In the following section, we test a few existing sampling methods on an anonymized TCP/IP network trace to determine their effect on analysis of the privacy/utility tradeoff on that trace, and present the results of these experiments in Section III. Finally, we discuss related and future work in Section IV and conclude in Section V. Readers who wish to review our sampling methods and metrics, or explore details of our methods and results, may see Fazio's thesis [11].

II. Experiments

In this section, we present the results of our analysis of the effects of several sampling methods on privacy metrics and utility measurements on a wireless-network trace. This trace was collected over a nine-hour over-night period on the Dartmouth College campus wireless network in December 2003 and contains 1,651,553 IP packets with either TCP or UDP headers. It was collected using 18 wireless sniffers located in 14 buildings across campus.

A. Trace preparation

Prior to sampling the trace, we used the open-source `anontool` [12] program to produce a sanitized version of our trace. We configured `anontool` to use a prefix-preserving mapping for IP addresses, a random one-to-one mapping for port numbers, and hashing for the payload. With this configuration, the number of packets and flows remained the same in the raw and sanitized trace, and the distributions of features remained constant, which allows us to measure the changes incurred upon these distributions by the sampling methods.

B. Sampling configuration

For this analysis, we implemented deterministic and uniform random packet sampling, stratified packet sampling, determin-

istic and uniform random flow sampling, smart sampling, and selective sampling. Details are available in Fazio [11].

To perform packet-based sampling on our traces in `pcap` format, we used a custom Python wrapper for the C library `libpcap` [13] to count the number of TCP and UDP packets contained within the trace and collect field information as necessary (e.g., src values when performing a stratified sample on that field), performed the selected sampling method on the input trace, and constructed a new `pcap` trace containing only the packets selected for the sample.

For flow-based sampling, we used the tool `tcptrace` [14] on default settings. We first constructed a list of the distinct connections (flows) present within our packet trace, and determined the total number of TCP and UDP connections in the trace. With the detailed output from `tcptrace`, we were then able to collect the appropriate data needed about the trace (e.g., each flow's size in bytes, for smart sampling) – using this data, we wrote a Python program to perform the selected sampling method on the trace. Given the set of flows to be included in the sample, we again used `tcptrace` to filter the trace and determine the packets belonging to those flows; as before, we then constructed a new `pcap` trace containing the packets selected for the sample.

We chose parameters so that trends in privacy or utility measurements could be best distinguished, with a secondary goal that the sample trace sizes are roughly comparable between sampling methods. While deterministic and random sampling can decrease the sample size to 0, the opportunistic sampling methods' sample size converges towards a non-zero value, making direct comparison difficult between individual traces produced by differing sampling methods.

C. Metric configuration

We explored three privacy and one utility metrics; details can be found in Fazio [11].

k-anonymity [15]. When calculating k-anonymity of the trace, we consider one feature to be "sensitive" and the others to be quasi-identifiers, leading to four cases:

Sensitive Field	Quasi-identifiers
src	$\{dst, srcprt, dstprt\}$
dst	$\{src, srcprt, dstprt\}$
$srcprt$	$\{src, dst, dstprt\}$
$dstprt$	$\{src, dst, srcprt\}$

L1-similarity [16]. The L1-similarity metric requires us to define the *network objects* that are to be protected with anonymity. Because we were using a packet trace the most natural network object is a $host$. A host is a unique IP address which is either a sender or receiver of data in the packet trace, represented in either the src or prt features. In other words, a host is either a sender or a receiver of data during the time period in which the trace was collected.

Formally, each host object A consists of records from a trace $\mathcal{T}$ with values on features F drawn from the set of fields above, such that $A = \{t \in \mathcal{T} \,|\, t_{src} = h \vee t_{dst} = h\}$, where h is the IP address for A. To measure the similarity between two objects, we first calculate the distributions of distinct values on each feature for the records contained within the host object.

We measured the L1-similarity between an unanonymized host A_r and an anonymized host A_s. For each feature $f \in F$, we defined A_{r_f} and A_{s_f} as the distribution of distinct values over f for each packet represented in A_r and A_s, respectively. We then compared the distribution of distinct values in A_{r_f} and A_{s_f}. Due to trace sanitization, these distinct values cannot be compared directly (e.g., IP address "67.23.134.45" anonymizes to "123.145.167.189") and we were thus forced to indirectly compare the distributions. We did this in our analysis by considering the most frequently occurring value in A_{s_f} to represent the most frequently occurring value in A_{r_f}, the second most frequently values in A_{s_f} and to A_{r_f} to be the same, and so on. The maximum value for $sim(A_{r_f}, A_{s_f})$ is 2, which indicates that the two objects have identical distributions on the feature f, and it is likely that $A_r = A_s$ [11].

Entropy anonymity degree (EAD) [17]. The EAD metric requires one to define a probability mass function Pr for a given analysis. We implemented a function used by Coull et al. [18] as a portion of their analysis to measure privacy over several iterations of adversary deanonymization, defined as follows:

$$Pr(A_{r_f} = A_{s_f}) = \frac{sim(A_{r_f}, A_{s_f})}{\sum_{a_r \in A_R, a_s \in A_S} sim(a_{r_f}, a_{s_f})}. \quad (1)$$

where A_R is the set of all hosts in the raw trace and A_S is the set of all hosts in the sanitized trace.

This definition of the probability mass function assigns the highest masses where similarity between two objects is relatively high compared to the total similarity measured between all pairs of objects. For example, let $sim(A_{r_f}, A_{s_f}) = 1.8$; if the average similarity between a raw object and sanitized object is 0.5, $Pr(A_{r_f} = A_{s_f})$ would be much higher than if the average similarity were 1.7.

We then calculated Shannon's entropy on the set of values in Pr on each feature to calculate the EAD at the field level. At the host level, the EAD is the sum of the field-level entropies; to calculate the normalized EAD, whose value is between 0 and 1, we divided the field-level EAD and host-level EAD by $\log_2 |A_S|$ and $|F| \log_2 |A_S|$; $|F|$ is the number of fields.

Snort [19]. To perform our experiments, we obtained the core Snort Engine and installed it on our test machine. Because the Snort intrusion-detection system depends on an updated set of rules to test traces against, we updated our rules to the set current as of 2011-04-27.

To collect alert logs, we then ran Snort with these rules on each input sampled trace, filtering for alerts on IP packets only but otherwise run on default settings. (Alerts generated by packets with protocols other than TCP or UDP were present in our log files; we discuss this effect when examining the results of our analyses below. After collection, we used a script to further parse and process these logs to obtain counts of each unique alert type for our sampled traces. We then calculate a utility metric by summing the number of alerts, weighted by relative severity.

D. Results

In this subsection, we present the results of our experiments on the sample traces described above.

a) L1-similarity and EAD: The potential resource benefits of sampling are clear upon examination of our chosen algorithm to calculate L1-similarity between raw and anonymized objects. Because we compare each raw object to each anonymized object, use of this metric requires $O(n^2)$ time and space; reducing the input size, even by a small amount, would therefore result in tangible performance gains.

Due to our implementation of the L1-similarity metric as the core for our EAD metric, we were able to directly examine whether similarity values were changed by the sampling process, and if so, whether entropy values based on that metric were also affected.

Calculating these metrics on our original unsampled trace served as a baseline by which we could compare the accuracy of the same calculations when run on sampled raw and sanitized traces; these baseline values are located in Table I. Note that in this and all other L1-similarity calculations in our experiments, the value listed represents the average of the L1-similarity values across all sanitized objects.

TABLE I
L1-SIMILARITY AND EAD FOR THE UNSAMPLED SANITIZED TRACE.

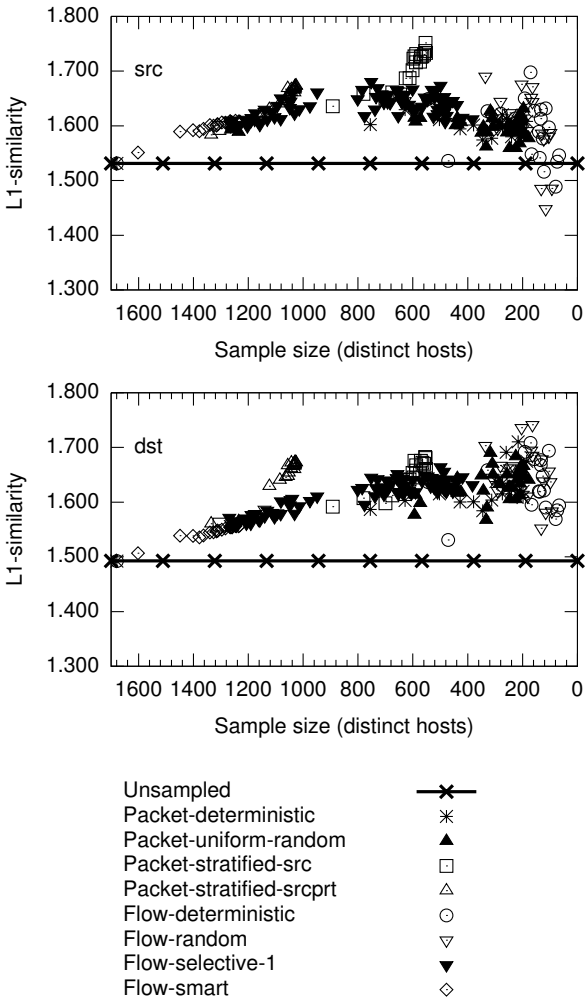
Unique hosts	1679			
L1-similarity	src	dst	$srcprt$	$dstprt$
	1.5314	1.4925	0.9112	1.4503
	$host$: 1.3463			
EAD	src	dst	$srcprt$	$dstprt$
	0.99530	0.99513	0.97028	0.99486
	$host$: 0.98889			

These results indicate that, over the 1,679 unique hosts located in the trace file, the average similarity of hosts across the src and dst objects are roughly the same and that the distributions of these similarities are also comparable; the entropy values for both src and dst were approximately 0.995 in the unsampled trace. Lessened similarity and EAD in the $srcprt$ feature indicates an increased probability that a host may be uniquely identified using external information about $srcprt$ distributions. Clients who send TCP requests over a wide range of ports could be responsible for this result (as opposed to external servers, which typically respond over well-known ports). We may, therefore, consider the $srcprt$ field to be the "least private" field when measuring anonymity of hosts in the unsampled trace.

Figure 1 presents the results of our calculations of the L1-similarity of two features *src,dst*. We explored other features, too, and larger plots are available in Fazio's thesis [11].

We chose the number of distinct hosts as our x-axis in these plots because they reveal a clear trend in similarity values for all features as the sampled trace size decreases; L1-similarity measurements become artificially high even at low sampling rates and then begin to decrease again as the sample trace size approaches 0. None of the sampling methods tested here alters or mitigates this trend towards an overestimation of L1-similarity values. Were there an end-user who attempted trace sampling in conjunction with an L1-similarity metric, that user could be led to believe, in error, that the anonymized trace is more secure than it actually is.

Fig. 1. Relationship between number of distinct hosts in a sampled trace and the corresponding L1-similarity of each feature.



Relationships in similarity between features are also distorted (or lost entirely) in every sampling method tested. For example, the similarity of field src is slightly greater than the similarity of field dst in our non-sampled trace, but as the sample size decreases, this situation quickly reverses itself. Distortions such as this make developing an effective sanitization strategy more difficult, as they can alter the perception of which fields are most insecure.

The results in Figure 1 are likely a result of the bias of the sampling methods to select packets from larger "elephant" flows and more prominent hosts in the trace – even selective sampling fails to correct this bias; the resulting decreased diversity in feature distributions would cause the similarity values to increase. As sample size approaches zero, similarity values are much more varied and scattered – especially when

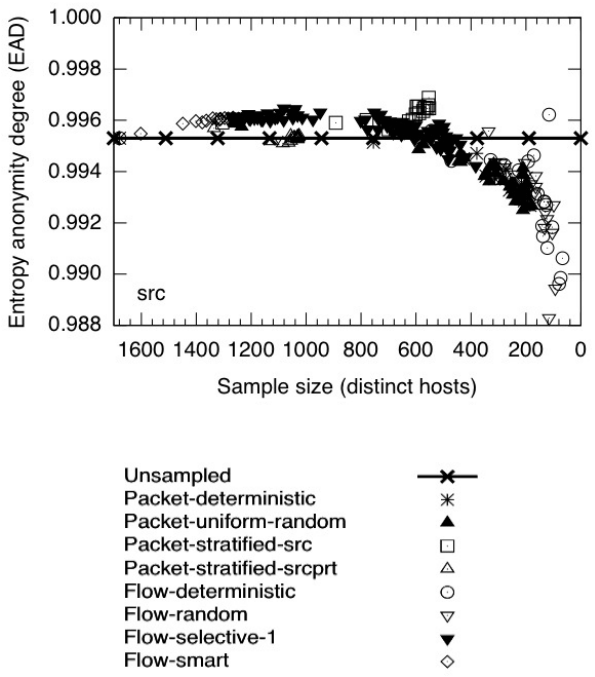


Fig. 2. Relationship between number of distinct hosts in a sampled trace and the corresponding EAD of each feature.

using flow-sampling methods. This variation is due to the fact that as the sampled trace shrinks, the removal of additional packets or flows could just as easily increase the similarity (as the sample becomes relatively more homogeneous) or decrease it (if the sample becomes less saturated with dominant "elephant"-sized flows).

Figure 2 shows the calculation of entropy anonymity degree using the same input traces and our similarity metric from Figure 1 at its core. (We explore more fields in [11]; for most fields the behavior of anonymity calculations is similar.) As the sample size (measured in distinct network hosts) shrinks, entropy calculations remain at or slightly above the accurate value. When the sample size decreases beneath approximately 800 distinct hosts, or about half the unsampled trace size, entropy begins to decrease across the board, with this decrease accelerating as the sample size approaches zero.

Entropy decreases when fewer network hosts are present in the sample because the unanonymized and anonymized objects are more easily placed in a one-to-one mapping than may have occurred in a larger sample set; each sanitized object is more likely to have a unique distribution of features that can be matched with an equivalent unsanitized object, whereas in a larger trace, there may be several unanonymized objects with a similar distribution of features.

b) k-anonymity: The effects of sampling on k-anonymity are less clear, however, due to the fact that k for the sanitized trace is equal to 1 on all fields. This is to be expected, as our sanitization configuration did not involve the truncation of any data. By not truncating data, we do not make any alterations to the equivalence classes present in the unsanitized trace, and the presence of just one quasi-identifier with one instance in the dataset is enough to make $k = 1$ for the entire field.

The sole use of k-anonymity as a privacy metric is difficult, however, due to the inability to identify sensitive attributes with certainty. In our experiments, we attempted to judge each field as sensitive, using the other three tested fields as quasi-identifiers, as described in Section II-C, with mixed results. While an increase in k-anonymity in one of our samples, such that $k > 1$, would indicate an inaccurate value that could give a researcher a false sense of security, we did not see this result in any of the sampled traces on any field.

c) Snort alerts: Our investigation of the effects of sampling on intrusion and anomaly detection was similarly difficult because of the tendency of sampling to reduce substantially the number of alerts that Snort detects, which adversely affects the ability to accurately predict the utility of the whole trace. A summary of the 2068 alerts triggered by the unsampled trace is contained in Table II.

TABLE II
SNORT ALERTS FROM THE UNSAMPLED TRACE.

Alert type	Frequency
ICMP unreachable host	1
ICMP unreachable port	206
ICMP ping	285
TCP reset	1012
TCP window close	564
total	2068
utility	568.4
utility (without ICMP included)	371.6

Fully half of our sample traces (129 of 257) failed to trigger any Snort alerts, while those that did trigger alerts only triggered a small fraction of the utility total on its own; the results are plotted in Figure 3. While the overall utility of the sanitized trace was measured by Snort to be 568.4, this figure includes ICMP packets that were filtered out in all of the sampling processes – the sans-ICMP utility measure of the unsampled anonymized trace was thus 371.6.

Only when using smart sampling, whose sample sizes all contained greater than 1200 unique hosts, were utility measurements somewhat reliable and followed a basic trend, decreasing sharply in a somewhat-linear fashion until reaching the smallest smart sample ($z = 9503$, 1231 unique hosts, utility = 0.46). When sample size shrinks further, alert generation becomes hit-or-miss regardless of sampling method – because different random samples can behave differently at the same parameter settings – with no sample registering a utility of greater than 11.4 (which occurred with deterministic flow sampling, $n = 211$). Because of the unreliability (or total lack) of alert counts, it is difficult to conclude that any particular sampling method outperforms another. With the possible exception of smart sampling, none of the sampling methods tested would have allowed an accurate measurement of utility; all would have severely underestimated the utility of the trace, as defined by our metric.

E. Limitations

The tests and measurements that we performed were conducted on a single, relatively small (200MB) trace, which pre-

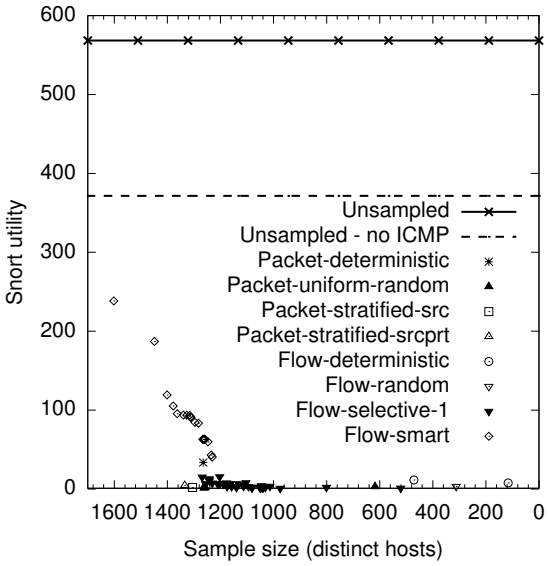


Fig. 3. Relationship between sample size (in distinct hosts) and the utility measured by Snort alerts.

cludes us from making broad-reaching conclusions about the applicability of trace sampling to measure the privacy-utility tradeoff in most circumstances or use cases. Nonetheless, the results show that sampling often has a substantial impact on the privacy and utility metrics we studied, and that conclusion is likely to occur for larger traces. Because only TCP and UDP headers were examined in our experiments, there may be other types of records or protocols that are more amenable to sampling. Finally, our experiments themselves may not provide a comprehensive summary of sampling methods to measure privacy and utility, as we have focused on a small number of distinct metrics and did not test any adaptive sampling methods, which may or may not outperform the conventional methods tested here.

III. Discussion

For the trace used in performing the above analyses, it is apparent that none of the sampling methods tested would have yielded accurate information about the privacy and utility of the sanitized trace that could aid a researcher in the task of releasing a network trace to the community. This result does not rule out the ability of trace sampling to allow an accurate estimation of the privacy-utility tradeoff, as the trace used for these experiments was relatively small and the sampling methods used here may differ in behavior on larger or more diverse traces.

But these results suggest that either additional research into new sampling algorithms is needed, or that it may be inappropriate to pursue a one-size-fits-all approach to sampling traces when measuring the privacy-utility tradeoff without a more intimate knowledge of the metrics being used to measure that tradeoff.

Metrics relying on entropy, such as entropy anonymity degree, seemed more able to absorb the changes in successively-smaller sampled traces and produce somewhat accurate measurements despite the noted changes in the L1-similarity metric values that EAD was itself based on. Attempts to distill EAD values for individual features down to a single per-host privacy value could be misleading, however, as the behavior of one field ($srcprt$) largely defines the EAD for the entire $host$ object in our experiments.

Because flow-based sampling did not definitively outperform packet-based sampling in any of our experiments, and given its increased resource requirements, it would be difficult to recommend its use over packet-based sampling based on these results. Were we to recommend a sampling strategy for the trace tested in our experiments, a potential sampling strategy to yield minor performance gains without significant loss of accuracy could include:

1) limited deterministic (sample rate ≈ 2) packet sampling to perform L1-similarity and EAD tests at a feature level, as they are the most performance-intense metrics and least susceptible to disturbance by sampling,
2) measuring k-anonymity without sampling, as the results (while somewhat unhelpful in this case) are relatively easy to calculate and equivalence class sizes degrade quickly with even limited sampling, and
3) measuring utility using Snort without sampling, as any sampling can seriously affect the number of alerts triggered (and thus the inferred utility).

IV. Related and future work

Related work on packet sampling and its effects on network traffic characterization was conducted by Claffy et al. [7] and Hernandez et al. [9], the latter introducing adaptive methods to equal or outperform non-adaptive packet sampling while reducing hardware or storage requirements to collect the relevant traces.

Brauckhoff et al. [4] specifically examined the impact of packet sampling on flow statistics and the ability to detect the Blaster worm by examining changes in entropy. Mai et al. [5], [6] demonstrated the ability to detect a range of anomalies using a number of packet and flow-sampling methods, and Androulidakis et al. [3] used flow-sampling methods to specifically target anomalies dependent on changes in entropy across fields. Tune and Veitch [20] examined the relative benefits of using sampling and sketching techniques to measure flow size distributions.

Kelly et al. [16] gathered a list of existing privacy metrics, including metrics based on information entropy described by Diaz et al. [17]. Coull et al. [18], [21] examined the sensitive information that can be inferred from traces, methods to measure privacy using a combination of entropy and L1-similarity of feature distributions, and described an iterative strategy to simulate an adversary's attempt to deanonymize data using externally available information.

Lakkaraju and Slagell [22] examined the use of Snort to measure utility of a network trace. Research from Pang et al. [23] and Slagell and Yurcik [24] discuss the inherent tradeoff between privacy and utility when sanitizing network traces. Finally, Fazio et al. [2] outlined a framework to streamline

the process of sanitizing traces for researchers looking to best address this tradeoff.

To the best of our knowledge, however, there is no existing work that measures the effect of packet sampling (or flow sampling) on the simultaneous measurement of privacy and utility of a network trace, the so-called privacy-utility tradeoff. Additionally, this work treats trace sampling as one of a series of steps to best utilize the resources of a researcher seeking to anonymize and release a network trace with the specific goal of allowing colleagues to conduct useful research on the anonymized traces.

Future directions in work related to sampling and its effect on measuring the privacy-utility tradeoff in network traces include research towards more generally effective sampling methods, and their specific effects on privacy and utility metrics, and the ability to accurately "correct" measurements using sampled traces to their unsampled equivalents. More rigorous mathematical casting of the privacy-utility tradeoff could lead to increased usage of formal methods to determine a representative subset of the trace, leading to metric calculations substantially similar to the same metrics calculated over the original trace. Finally, future research could also examine multi-stage sampling methods, or processes that combine one or more distinct sampling methods based on context, and their effects on privacy and utility compared to using a broadly-applicable sampling method.

V. Summary

In this paper, we examine several simple non-adaptive sampling methods to discover their effects on measurement of the privacy-utility tradeoff when sanitizing network traces prior to their sharing or publication. The results will be applied to the recently-introduced NetSANI framework for trace sanitization that seeks to ease the burden on researchers to adequately sanitize their network traces (while preserving useful data) and share them with their colleagues. While packet and flow sampling have been used and analyzed in the past for such applications as anomaly detection and traffic measurement, little or no research has been done to sampling's direct effect on measuring both privacy and utility at the same time.

After sanitizing a small sample trace collected from the Dartmouth College wireless network, we tested the relative accuracy of a variety of previously-implemented packet and flow-sampling methods when measuring privacy with micro-data, similarity, and entropy-based metrics, and on utility by use of the Snort intrusion-detection system. The results of this analysis led us to conclude that, for the test trace, no single sampling method we examined was able to accurately measure privacy and utility, and that some sampling methods can produce grossly inaccurate estimates of those values. We also found it unlikely that a single "universal" sampling method could be used to perform this analysis accurately on a trace of any size. We were unable to draw conclusions on the use of packet versus flow sampling in these instances, nor were we able to gauge the accuracy of experiments on larger traces.

References

[1] (2010) Community Resource for Archiving Wireless Data At Dartmouth (CRAWDAD). At http://www.crawdad.org/

[2] P. Fazio, K. Tan, J. Yeo, and D. Kotz, "Short paper: The NetSANI framework for analysis and fine-tuning of network trace sanitization," in *Proceedings of the ACM Conference on Wireless Network Security (WiSec)*. ACM Press, Jun. 2011.

[3] G. Androulidakis, V. Chatzigiannakis, and S. Papavassiliou, "Network anomaly detection and classification via opportunistic sampling," *The Magazine of Global Internetworking*, vol. 23, January 2009.

[4] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina, "Impact of packet sampling on anomaly detection metrics," in *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*. ACM, 2006.

[5] J. Mai, C.-N. Chuah, A. Sridharan, T. Ye, and H. Zang, "Is sampled data sufficient for anomaly detection?" in *Proceedings of the Internet Measurement Conference (IMC)*, 2006.

[6] J. Mai, A. Sridharan, C.-N. Chuah, H. Zang, and T. Ye, "Impact of packet sampling on portscan detection," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, December 2006.

[7] K. C. Claffy, G. C. Polyzos, and H.-W. Braun, "Application of sampling methodologies to network traffic characterization," in *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*. ACM, 1993.

[8] N. Duffield, C. Lund, and M. Thorup, "Properties and prediction of flow statistics from sampled packet streams," in *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*. ACM, 2002.

[9] E. A. Hernandez, M. C. Chidester, and A. D. George, "Adaptive sampling for network management," *Journal of Network and Systems Management*, vol. 9, December 2001.

[10] J. Zhang, X. Niu, and J. Wu, "A space-efficient fair packet sampling algorithm," in *Proceedings of the 11th Asia-Pacific Symposium on Network Operations and Management: Challenges for Next Generation Network Operations and Service Management (APNOMS)*. Springer-Verlag, 2008.

[11] P. A. Fazio, "Effects of network trace sampling methods on privacy and utility metrics," Dartmouth College, Computer Science, Hanover, NH, Tech. Rep. TR2011-697, Jun. 2011. At http://www.cs.dartmouth.edu/reports/TR2011-697.pdf

[12] D. Koukis, S. Antonatos, D. Antoniades, E. P. Markatos, P. Trimintzios, and M. Fukarakis, "CRAWDAD tool tools/sanitize/generic/anontool (v. 2006-09-26)," Downloaded from http://crawdad.org/tools/sanitize/generic/AnonTool, Sep. 2006.

[13] (2011) tcpdump/libpcap public repository. At http://www.tcpdump.org/

[14] (2011) tcptrace - Official Homepage. At http://www.tcptrace.org/

[15] V. Ciriani, S. Capitani di Vimercati, S. Foresti, and P. Samarati, "k-anonymity," in *Secure Data Management in Decentralized Systems*, ser. Advances in Information Security, T. Yu and S. Jajodia, Eds. Springer US, 2007, vol. 33.

[16] D. J. Kelly, R. A. Raines, M. R. Grimaila, R. O. Baldwin, and B. E. Mullins, "A survey of state-of-the-art in anonymity metrics," in *Proceedings of the ACM Workshop on Network Data Anonymization (NDA)*. ACM Press, 2008.

[17] C. Diaz, J. Claessens, S. Seys, and B. Preneel, "Information theory and anonymity," in *Werkgemeenschap voor Informatie en Communicatietheorie*, B. Macq and J.-J. Quisquater, Eds., 2002.

[18] S. Coull, C. Wright, F. Monrose, A. Keromytis, and M. Reiter, "Taming the Devil: Techniques for evaluating anonymized network data," in *Proceedings of the Annual Symposium on Network and Distributed System Security (NDSS)*, Feb. 2008. At http://www.cs.unc.edu/~reiter/papers/2008/NDSS.pdf

[19] M. Roesch, "Snort: A free, open source network intrusion detection and prevention system," 1998, version 2.9.0.5. At http://www.snort.org/

[20] P. Tune and D. Veitch, "Sampling vs sketching: An information theoretic comparison," in *Proceedings of IEEE INFOCOM*, Apr. 2011.

[21] S. E. Coull, C. V. Wright, F. Monrose, M. P. Collins, and M. K. Reiter, "Playing Devil's advocate: Inferring sensitive information from anonymized network traces," in *Proceedings of the Annual Symposium on Network and Distributed System Security (NDSS)*. IEEE Press, Feb. 2007. At http://www.isoc.org/isoc/conferences/ndss/07/papers/playing_devils_advocate.pdf

[22] K. Lakkaraju and A. Slagell, "Evaluating the utility of anonymized network traces for intrusion detection," in *Proceedings of the International Conference on Security and Privacy in Communication Networks (SecureComm)*. ACM, 2008.
[23] R. Pang, M. Allman, V. Paxson, and J. Lee, "The devil and packet trace anonymization," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, 2006.
[24] A. Slagell and W. Yurcik, "Sharing computer network logs for security and privacy: a motivation for new methodologies of anonymization," in *Proceedings of the International Conference on Security and Privacy for Emerging Areas in Communication Networks*, 2005.
[25] A. Serjantov and G. Danezis, "Towards an information theoretic metric for anonymity," in *Proceedings of the International Symposium on Privacy Enhancing Technologies (PET)*, ser. Lecture Notes in Computer Science, vol. 2482. Springer-Verlag, 2002.
[26] M. Burkhart, D. Brauckhoff, M. May, and E. Boschi, "The risk-utility tradeoff for IP address truncation," in *Proceedings of the ACM Workshop on Network Data Anonymization (NDA)*. ACM, October 2008.
[27] A. Lakhina, M. Crovella, and C. Diot, "Mining anomalies using traffic feature distributions," in *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*. ACM, 2005.

APPENDIX

There are two types of metrics used in the analysis of network traces: *privacy metrics*, which measure the degree to which a sanitization method fulfills its predetermined requirements, and *utility metrics*, which measure the usefulness of data preserved after the sanitization of a trace.

A. Privacy metrics

A *privacy metric* (also known as an *anonymity metric*), is defined by Kelly et al. [16] as a quantification of how well an anonymization strategy hides the identity of sensitive information or users against a particular attack.

Individual privacy metrics may be sensitive to the underlying types or structure of the dataset to be analyzed or may be generic enough to apply to a wider range of data formats. In our work, we classify anonymity metrics into two broad models: those that are *microdata-based* and those that are *network-based*. Microdata-based metrics assume the dataset is organized like a relational database in which certain fields may be designated as "sensitive" *a priori*; network-based metrics use statistical and probabilistic information about the dataset to simulate an adversary's knowledge. The NetSANI framework [2] is designed to accommodate both models, and our experiments here use examples of both types of metrics.

a) k-anonymity: We begin with a well-known and simple microdata-based metric, *k-anonymity* [15]. When preparing to release a dataset that contains fields known to contain "sensitive" information, we must recognize that the adversary may use some non-sensitive fields in conjunction with externally available information to identify sensitive data; this set of attributes are called *quasi-identifiers* [15]. An adversary may have external information that maps quasi-identifiers to actual identities, and thus may be able (with the released dataset) to map identities to sensitive values. The microdata-based metric of *k-anonymity* states that for each combination of values of quasi-identifiers, that combination can be matched to at least k identities.

The dataset presented in Table III is an example of an anonymized dataset that achieves 2-anonymity with the set of sensitive attributes $S = query$. This means that for each individual record in the dataset, there exist at least 2 instances of a single quasi-identifier $Q = ip, date, time$ that could be associated with that record's sensitive value. In our sample dataset, this means that the sensitive query of "skin rash" could have originated from at least 2 records with the quasi-identifier (96.234.68.2*, 2008-10-**, 23**).

TABLE III
SAMPLE DATASET FOR A SEARCH-ENGINE LOG ANONYMIZED FOR 2-ANONYMITY IS ACHIEVED [16].

IP address	Date	Time	Query
96.234.69.*	2008-10-2*	234*	AIDS medicine
96.234.69.*	2008-10-2*	234*	AIDS medicine
222.154.155.***	2008-10-**	23**	m-invariant
222.154.155.***	2008-10-**	23**	l-diversity
96.234.68.2*	2008-10-**	23**	cook book
96.234.68.2*	2008-10-**	23**	skin rash
96.234.68.2*	2008-10-**	23**	filling station
96.234.68.2*	2008-10-**	23**	winter coats
129.170.111.1**	2008-10-2*	235*	tan salon
129.170.111.1**	2008-10-2*	235*	mcdonalds jobs

TABLE IV
k-ANONYMITY PRIVACY METRIC [16].

Privacy level	Metric level = z
Preserved	$k \geq z$
Degraded	$k < z$
Eliminated	$k = 1$

k-anonymity is quite limited in its scope, and does not attempt to measure the variety of sensitive values associated with each quasi-identifier; for instance, all records within the quasi-identifier (96.234.69.**, 2008-10-2*, 234*) in the sample dataset can be associated with the sensitive query "AIDS medicine". Additional microdata-type metrics such as l-diversity and t-closeness attempt to address this and other shortcomings [16].

We can define k-anonymity as a simple ternary privacy metric, as shown in Table IV; it measures privacy at three levels: preserved, degraded, or eliminated. By specifying a threshold value z, we measure whether the network trace is k-anonymous such that $k \geq z$. If so, then privacy is considered to be preserved; else privacy is considered degraded, or in the case of $k = 1$, eliminated [16].

b) L1-similarity: Prior to introducing the next metric, we introduce the concept of a *network object* [18], [2]. A network object is an entity whose identity a trace publisher seeks to protect and/or retain utility, such as a host, subnet, or web page. It is important to note that an object may be defined by more than one record in a trace (multiple packets may be from the same host); the converse holds, as a record may belong to one or more network objects (for example, a TCP packet refers to both the src host and the dst host).

L1-similarity [16] estimates the anonymity of an object by computing a distance between the distribution of values of an anonymized object X and the distribution of values of an

TABLE V
L1-SIMILARITY ANONYMITY METRIC [16].

Privacy level	**Metric level =** $sim(X,Y)$
Preserved	$sim(X,Y) = 2$
Degraded	$0 < sim(X,Y) < 2$
Eliminated	$sim(X,Y) \approx 0$

unanonymized object Y, defined as

$$sim(X,Y) = 2 - \sum_{z \in X \cup Y} |P(X = z) - P(y = z)|. \quad (2)$$

The maximum value of $sim(X,Y)$ is 2, which represents an identical distribution of features between the two objects. This notion is somewhat counterintuitive, representing the maximum preservation of anonymity because an attacker fails to gain additional knowledge from the anonymized dataset. Conversely, if the distributions of the two objects are totally disjoint, the similarity is 0, and the attacker gains "complete or substantial knowledge of identities and relationships" [16]. This metric is summarized in Table V.

c) Entropy anonymity degree (EAD): Consider a situation of an adversary attempting to discern the author of each of several messages sent across a network. The adversary knows the set of all the possible authors, but at the onset, it appears equally likely that any author may have sent a given message. However, upon learning additional information, such as how prolific each author is, the adversary is able to guess with more certainty which author may have written a given message. The metric of *entropy anonymity degree* uses entropy to measure how much information the adversary has gained, and thus, the degree of anonymity that the author of a message retains after the attack [17].

Mathematically, let I be the set of distinct values that are represented in a probability distribution X; in this case, each $i \in I$ represents an author a_i in the set of all possible authors A. Let p_i represent the probability that a_i is responsible for a message. Therefore, $p_i = Pr(X = i)$, where Pr is a probability mass function.

The entropy of this probability distribution is defined as follows, where N is the size of the sample space [18], [17]:

$$H(X) = -\sum_{i=1}^{N} p_i \log_2(p_i) \quad (3)$$

Entropy anonymity degree (noted D_r) normalizes the result of $H(X)$ above, dividing it by the maximum entropy $H_{max} = \log_2 N$ of the system:

$$D_r = \frac{H(X)}{H_{max}}, \quad (4)$$

where $0 \leq D_r \leq 1$ [25].

Logically, the maximum degree of anonymity is achieved when the attacker finds it equally likely that any author is responsible for sending a given message; likewise, anonymity has been eliminated when the attacker is certain or near-certain of the author of that message. This may be represented in an anonymity metric as follows (summarized in Table VI):

TABLE VI
EAD PRIVACY METRIC [16].

Privacy level	**Metric level =** D_r
Preserved	$D_r = 1$
Degraded	$0 \leq D_r < 1$
Eliminated	$D_r \approx 0$

when $D_r = 1$, all values across the attribute are equally likely, and privacy is fully preserved. As D_r decreases, privacy becomes increasingly degraded until, when $D_r = 0$, privacy is considered fully eliminated.

Note that the value of EAD, D_r, is specific to the probability mass function Pr of the sample space X, which is dependent on the type of information the attacker possesses.

B. Utility metrics

Quantified measurement of utility is difficult, because those looking to use trace data (researchers) often have specific use cases. Therefore, unlike the privacy metrics, developing an overarching utility metric is a much more challenging endeavor [26]. Almost all metrics to date center on the concept of anomaly detection as a measure of utility, because often the search for anomalous traffic (e.g., DoS attack, portscan) requires a wide range of useful data from the trace. For example, several important anomalies may be mined from examining the entropy of traffic features [3], [27]:

- **DDoS attack**: a *distributed denial of service* attack attempts to target a service to make its resources unavailable to others; it may come from many sources.
  Fields affected: large decrease in H for dst and $dstprt$.
- **Portscan**: in a *portscan* attack, a single sender sends packets to a host over a wide range of ports, with the intent to identify services available at the host.
  Fields affected: large decrease in H for src, dst, and $srcprt$, slight increase in H for $dstprt$, and slight decrease in H for F_x, the flow size.
- **Worm propagation**: a program that replicates itself in an attempt to exploit and infect other machines.
  Fields affected: large decrease in H for src and $dstprt$, slight increase for dst and $srcprt$, and slight decrease in H for F_x.

While it is possible to implement separate metrics measuring detection of the attacks above, the open-source intrusion-detection tool Snort [19] contains the tools necessary to determine the type and extent of a wide range of anomalies during either live packet capture or post-capture analysis [22, for example]. Because Snort contains rules designed to discover instances of all three attacks described above, we assign a weight $0 \leq w \leq 1$ to each alert based on the number and severity of alerts when running a TCP/UDP network trace through the intrusion-detection tool; alerts with higher severity or lower frequency are assigned higher utility value.

Because a trace theoretically has no measurable utility limit, we compare the values of the utility metric of the raw trace against the same analysis run on the sanitized trace.