Short Paper: The NetSANI Framework for Analysis and Fine-tuning of Network Trace Sanitization

Phil Fazio, Keren Tan, Jihwang Yeo and David Kotz
ISTS, Dartmouth College, Hanover NH, USA
fazio, keren, jyeo, kotz@cs.dartmouth.edu

ABSTRACT

Anonymization is critical prior to sharing wireless-network traces within the research community, to protect both personal and organizational sensitive information from disclosure. One difficulty in anonymization, or more generally, sanitization, is that users lack information about the quality of a sanitization result, such as how much privacy risk a sanitized trace may expose, and how much research utility the sanitized trace may retain. We propose a framework, NetSANI, that allows users to analyze and control the privacy/utility tradeoff in network sanitization. NetSANI can accommodate most of the currently available privacy and utility metrics for network trace sanitization. This framework provides a set of APIs for analyzing the privacy/utility tradeoff by comparing the changes in privacy and utility levels of a trace for a sanitization operation. We demonstrate the framework with an quantitative evaluation on wireless-network traces.

Categories and Subject Descriptors: C.2.3 [Network Operations]: network monitoring, H.4 [Information Systems Applications]:decision support

General Terms: Design, Measurement, Security

Keywords: sanitization, network traces, APIs, privacy, utility, tradeoff

1. INTRODUCTION

Computer-network research advances more quickly when researchers are able to analyze traffic from live networks. Collecting trace data from production networks is often difficult, however, because it is difficult to obtain permission to install infrastructure and collect trace data. Thus, it is critical for the research community to share traces, leveraging this effort to benefit multiple research projects. Fortunately, a culture of sharing network traces exists [4, 9, 16], including CRAWDAD [8] for wireless networks.

Since traces collected from production networks capture the everyday business of network users, the privacy of these users is an increasing concern when sharing trace data. Researchers who wish to share traces must therefore properly "anonymize" the trace to remove personally identifiable information [12]. The researcher may also wish to "sanitize" the trace to conceal other sensitive information (e.g., network structure, critical server identities). We use the term *sanitization* to incorporate both of these goals. It takes great effort and care to correctly sanitize traces; from our own experience, and from other examples in the literature, researchers can make mistakes when sanitizing traces: they may not understand a tool's capabilities, be forced to write their own tools, or miss subtle ways in which information can leak from a trace. Many recent papers [18, 2, 6, 11] demonstrate methods to extract private information from traces thought to be suitably sanitized. Ma et al. [14] show, through analysis of wireless-network traces, that even a small amount of external information is enough for an adversary to infer a victim's true identity in a set of anonymized mobility traces from CRAWDAD. In a recent user survey [21], only 34% of survey participants with experience sanitizing traces used a third-party tool; the rest either used home-grown software or manually edited the traces. These solutions are inevitably likely to include errors affecting the privacy and/or utility of the resulting trace. Moreover, 84% of those with experience sanitizing traces stated that they did not use any quantitative metrics to measure sanitization strength. For a comprehensive survey of state-of-art network trace sanitization and de-sanitization research, we refer interested readers to our previous work [19].

Trace sanitization represents a tradeoff between removing information to protect privacy interests, and retaining information to allow meaningful analysis of the sanitized trace. Most users lack the means either to determine the privacy risks or the research utility of a sanitized trace. This information would allow the user to understand the quality of the sanitization result, controlling the tradeoff between privacy and utility to meet their desired privacy and utility goals.

In this paper, we address the following problems:

1. How can users evaluate sanitized traces in terms of privacy and utility, especially given the presence of a variety of existing metrics?
2. How can we provide users an easy framework to control and fine-tune the tradeoff between privacy and utility?

We thus propose a framework called NetSANI (Network Trace Sanitization and ANonymization Infrastructure), with which we can analyze and control this tradeoff between privacy and utility. To solve the first problem, we take a "framework" approach such that our evaluation tools can accommo-

WiSec'11, June14–17, 2011, Hamburg, Germany.

date most currently available privacy metrics [10, for example]. We demonstrate that this framework can measure the degree of utility for some typical uses of network traces (e.g., mobility analysis, network intrusion detection), and analyze the privacy and utility tradeoff by comparing changes in privacy and utility levels of a trace for a sanitization operation. To address the second problem, NetSANI provides tradeoff analysis results from different analyses of the same trace to allow the user to control and fine-tune this tradeoff. This allows the user to easily repeat the process with different sanitization parameters. Where possible, we seek to provide analysis results at several granularity levels, from encompassing the entire trace to analyzing individual fields or objects. To the best of our knowledge, NetSANI is the first framework to systematically support a broad range of tradeoff analyses for both wired and wireless network traces.

2. PRELIMINARY

In this section, we introduce the trace model and briefly describe the two broad types of metric in network trace analysis: *privacy metrics* measure how well a sanitization method fulfills predefined privacy or secrecy requirements, and *utility metrics* measure how much a sanitized trace remains useful to a researcher performing trace analysis.

Network trace model. We assume that a network trace has a table structure analogous to a relational database (i.e., trace $\mathcal{T}$ consists of N rows of M fields each). For example, in a packet trace, a packet may be represented by a row whose fields represent fields located within that packet's header. A *network object* is an entity whose identity the trace publisher seeks to protect and/or seeks to gain utility (e.g., host, subnet, user handle). Network objects may be defined by more than one packet in the trace; likewise, a packet (or collection of packets) may belong to one or more network object. For example, suppose that a TCP/IP packet trace includes source and destination IP addresses among its fields, then each host (as a network object) is defined by one or more packets (as rows) that include the IP address of the host in either source or destination IP-address field.

Threat models and privacy. There are two well-known models for privacy in data sets: *network-based* and *micro-data* models, and our framework is designed to accommodate either in analyzing the privacy of trace data. In both models, the adversary's goal is to identify sanitized objects using available knowledge about unsanitized objects. However, the difference between the two lies in the forms of the available knowledge and how the adversary uses the available knowledge.

In this paper, due to space limitations, we focus on the network-based model. In a future expanded version of this paper, we will address the micro-data model.

In the network-based threat model, one assumes that given sanitized trace $\mathcal{T}'$ (corresponding to $\mathcal{T}$), the adversary has some external knowledge about some unsanitized objects of trace $\mathcal{T}$. Although secondary traces containing unsanitized objects are rarely available, knowledge may come in other forms, including statistical information (e.g., port-usage distribution) about unsanitized objects or other more generic information (e.g., network topology).

Several information-theoretic measures of privacy have been proposed for this threat model [5, 6, 17]. Kelly et al. describe most of the currently available privacy metrics [10]. One basic indicator of anonymity, agnostic to data type, uses Shannon's entropy $H(s)$:

$$H(s) = -\sum_{j=1}^{L} P(s = o_j{}') \log P(s = o_j{}'), \qquad (1)$$

where $P(s = o_j{}')$ is the probability that sanitized object s can be obtained by applying a sanitization operation to raw object o_j. Lower entropy values correspond to stronger similarity between s and o_j.

Utility metrics. When anonymizing a network trace, a researcher must balance the need to protect privacy with the desire to retain as much useful data in the anonymized trace as possible. Since anonymization techniques may potentially disturb the analysis of a trace [15], we seek a metric that quantifies how the research utility of a trace changes because of the anonymization. Developing universal utility measures to apply to network traces is difficult due to the inherent complexity and interdependent nature of a network trace [7]. Application-dependent utility metrics (that measure values useful in common cases) may be more applicable to many network traces [3, 13, 15], such as comparing the number of alarms generated by an intrusion-detection system (e.g., Snort) pre/post anonymization [15]. Compared to anonymization algorithms, there has been far less research on utility metrics. A framework like NetSANI allows researchers to try a variety of metrics, or to define their own.

3. METHODS

In this section, we describe the requirements for a network trace-sanitization framework, and introduce the ideas behind the NetSANI framework.

3.1 Challenges

Properly evaluating the privacy of an anonymized trace requires the framework to address several specific challenges.

First, the framework should provide a flexible interface with which to transform trace data $\mathcal{T}$ from one of several formats (e.g., pcap, NetFlow, WLAN user association log) into a consistent relational table structure.

Second, the framework should allow its users (data publishers) to define their assumptions about adversary resources or deanonymization techniques (e.g., access to a portion of the unsanitized trace, or the distribution of features in the unsanitized trace). The framework should allow the publisher a choice of several threat models (e.g., network-based or microdata-based models) and sanitization configurations (e.g., different sanitization operations with differing degree of privacy and utility).

Finally, the toolset must provide a "pluggable" interface such that publishers may easily define and apply different metrics to a sanitized trace, and to implement custom metrics as a plug-in to the framework without modifying the core evaluation tool.

NetSANI first requires that its user, the data publisher, describe the subset of fields to be processed and analyzed by the system. This includes basic information such as data type (e.g., IP and MAC addresses, GPS coordinates) and how to access the field when transforming the raw trace file into relational database form.

Since a network object may be defined by several fields, we must provide an abstraction allowing the publisher to define

an object as a function of the fields F. Examples of network objects include network users, mobile hosts, wireless access points, or network servers.

Finally, the publisher specifies the format of the trace input, and provides a method for converting raw trace data into table form compatible with the NetSANI database. A trace parser module should be easily implementable, able to use external parsing modules (e.g., libpcap), and still be flexible enough to allow the publisher to create a parser for custom complex formats. We also store additional information about the trace, such as its name, field descriptions, and network objects.

When choosing the network-based threat model, the publisher provides assumptions regarding the adversary's knowledge about unsanitized objects, with two levels of granularity: a unit of knowledge about network objects (e.g., the distribution of field values across a single object) and about the trace (e.g., the full collection of distributions across all objects). Since the adversary's type of knowledge may vary greatly in different uses of the framework, the definition of these types of knowledge is deliberately vague, allowing the user flexibility.

Given the adversary's knowledge, the framework calculates an "uncertainty degree" for the adversary to map between each sanitized object and its best-matched unsanitized object. It is assumed that the more uncertain the adversary is of the mapping, the stronger the privacy of the sanitized objects is maintained. In other words, the uncertainty degree obtained from a sanitization operation indicates the degree of privacy the sanitization operation can provide.

To calculate the uncertainty degree, we first calculate a "knowledge distance", which indicates a quantitative difference between the adversary's knowledge about each unsanitized object and the characteristic of the corresponding sanitized objects. We then determine the uncertainty degree of the mapping.

3.2 Utility evaluation

Universal utility measures are difficult to interpret due to their inherently complex and interdependent nature [20]. Supporting all application-specific utility measures is impossible, because the variety of uses for trace data lead to a broad range of different utility metrics. As a compromise, our framework allows users to apply their own utility measure to network traces, as well as the ability to use some common utility measures. Users explicitly provide the framework with a function to compute per-object utility values in both the original and sanitized traces, as well a function to compute distance between the utility values to calculate a utility measure of each sanitized object.

To put it formally, the users provide a set of values V, and a classifier function $f_v : O \cup O' \rightarrow V$, where O and O' are sets of objects in $\mathcal{T}$ and $\mathcal{T}'$, respectively. The users also define a utility distance function $Dist_u : U, U \rightarrow \mathcal{R}$. The utility difference between raw object o_i and sanitized object o'_i is thus defined as $\Delta U(o_i) = Dist_u(f_v(o_i), f_v(o'_i))$, where $Dist_u$ is a user-provided function and D_{max} is the (user-provided) maximum value of $Dist_u$. The *object utility* of sanitized object o'_i is defined in range [0..1] as:

$$Util(o'_i) = 1 - \frac{Dist(o_i, o'_i)}{D_{max}} \quad (2)$$

Given the object utilities, one can compute the average, standard deviation, median and other summary utility metrics over all the objects of a sanitized trace.

3.3 Tradeoff analysis

In this section, we describe how to use our framework to analyze the tradeoff between privacy and utility in a sanitized trace. The tradeoff is presented through comparisons of measured values of privacy and utility. Our framework can support the tradeoff analysis at three different levels: *objects*, *fields*, and *trace*.

In the object-level tradeoff analysis, we compare the privacy and utility values measured on each sanitized object s using the object privacy metric and the object utility metric, i.e., $Util(s)$. The purpose of the object-level tradeoff analysis is to identify the sanitized network objects that are most vulnerable to privacy risk, or those that become much less usable than other sanitized objects.

The data publisher should not only be able to understand the vulnerability or utility of a network object, but should also be able to use the analysis results to apply different sanitization techniques to better satisfy the privacy and utility requirements of the sanitized trace. Thus, we need to provide guidance about which fields contribute more to the adversary's privacy attack or to the loss of utility than other fields. NetSANI provides a field-level tradeoff, though only for fields used by the utility metric function.

Trace-level analysis allows the data publisher to receive a broad overview of the security and usefulness of a sanitized network trace. When calculating an overall "score" for privacy and utility of an entire trace, the metric implemented within the framework may take into consideration prior calculations (e.g., a trace is as secure as its least secure object) and/or perform additional analysis of the trace structure at a level above individual fields or network objects. This type of analysis can help the user to easily identify and summarize the tradeoff between privacy and utility for a given sanitization configuration.

3.4 Tradeoff evaluation algorithm

Algorithm 1 shows our tradeoff evaluation algorithm, the NetSANI *Evaluation* module. It takes as inputs a raw trace, a sanitized trace, the adversary's knowledge file (which contains external knowledge about the raw trace), and a file for describing the mapping of objects between the raw trace and the sanitized trace. The evaluation algorithm compares the adversary's knowledge and each sanitized object so as to calculate object privacy for the object-level analysis.

The algorithm also calculates a degree of uncertainty for mapping a given sanitized object to an unsanitized object, and the per-object utility $Utils(i)$ according to Equation 2. Note that to calculate the object utility the *actual* mapping between sanitized objects and corresponding raw objects must be known to the tool.

3.5 Tradeoff control flow

Users can control the tradeoff between privacy and utility by selecting different sanitization configurations (e.g., changing a sanitization method on a trace field) within the framework until they are satisfied with the privacy and utility values ($Privs, Utils$) measured by the evaluation algorithm. In Figure 1, in which we show the overall flow during trace sanitization and analysis, the outer loop represents this manual iteration. In the inner loop of Figure 1, the tool adopts an

Algorithm 1: The NetSANI Evaluation algorithm

Input: A raw trace $RawTr$, the corresponding sanitized trace $SaniTr$, a file $MFile$ for mapping objects between $RawTr$ and $SaniTr$, and the adversary's knowledge file $KFile$.
Output: Per-object privacy values $Privs$, per-object utility values $Utils$

```
Parse RawTr, SaniTr and KFile;
Extract from RawTr network objects RObjs; N_r = 0;
foreach Network object RObj in RObjs do
    Calculate a utility value RawUtils[N_r] according to Equation (2);
    Extract a knowledge unit KUnit[N_r] from KFile;
    N_r = N_r+1;
Extract from SaniTr network objects SObjs;
Extract from MFile the mapping between SObjs and RObjs as Mappings;
i = 0;
foreach Network object SObj in SObjs do
    MinObjIdx = -1;
    MinDist = ∞;
    for j = 0 to N_r-1 do
        Calculate a knowledge distance between SObj and KUnit[j], into KGap[j];
        if KGap[j] < MinDist then
            MinDist = KGap[j];
            MinObjIdx = j;
    Calculate a degree of uncertainty for mapping between SObj and RObjs[MinObjIdx], into Privs[i];
    Calculate the object utility of SObj into SaniUtil;
    Calculate utility difference between SaniUtil and RawUtils[Mappings[i]] into Utils[i];
    i = i+1;
```

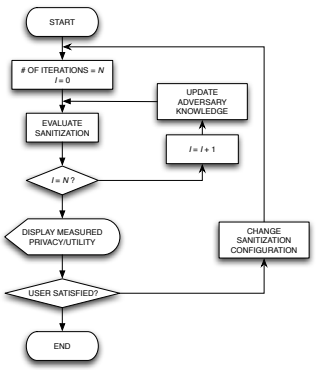

Figure 1: Tradeoff Control Flow

iterative approach to evaluating privacy (much as in Coull's work [6]). After assembling an initial adversary knowledge base, the framework evaluates that knowledge base against the chosen privacy metric, updating the knowledge base to reflect any new information derived when computing the metric. There may be a predetermined number of iterations ($N >= 1$ in Figure 1), or the engine may run until a certain goal is achieved (e.g., all sanitized objects are successfully mapped to an unsanitized equivalent). Utility is calculated on the first iteration only.

This approach not only simulates a common de-sanitization method by the adversary, but (if one were to log the progress of the algorithm) also shows the changes of privacy risks during repeated de-sanitization attempts.

Table 1: NetSANI API – Evaluation Functions

Function	*Description*
MyUtility	define a utility value
MyUtilityComp	compare two utility values
MyObjectUtility	define an object utility value ($Util$)
MyKDist	define a distance between a network object and knowledge units ($Dist_F$)
MyUncertainty	define a degree of uncertainty to map a sanitized object to a raw object
MyMDMetric	calculate an overall microdata metric

Table 2: NetSANI API – Classes

Class	*Description*
NetSANIFramework	master class abstracting away internal data storage and configuration
*Field**	define a basic data type
DataDescriptor	describe a set of columns or fields
*NetworkObject**	abstract a network object as defined in Section 2
SensitiveField	designate an field as "sensitive" as described in Section 3.1
Trace	abstract a trace
*TraceParser**	transform raw trace data into relational table model
TraceRow	abstract a row in a trace file
*KnowledgeUnit**	abstract an additional unit of information not found in trace itself
*AdversaryKnowledge**	abstract the adversary's knowledge

4. THE NETSANI API

In this section, we briefly describe the NetSANI API, primarily by example. When designing the API, we placed particular emphasis on maintaining balance of the *generality*, *elegance*, and *efficiency*.

We allow users to provide their own code as *user-defined functions* (shown in Table 1) or as *user-defined subclasses* of the classes shown in Table 2. Each of the starred classes in Table 2 is the Python equivalent of an abstract base class. We encourage the publisher (or programmer) to extend, augment, and reuse as much functionality as possible.

We have space here for only one example, as follows.

4.1 SNMP traces and mobility research

In this example, we used a log of wireless network association data collected in Winter 2010 and containing the MAC address of the user, the identification of the access point (AP), and timestamps indicating the start and end of that users' session at that AP. The access-point identification is stored in "building.floor.ID" format. We wish to evaluate sanitization techniques in terms of the tradeoff between privacy and utility for specific research goals outlined below, focusing on a network-based privacy metric.

To sanitize the traces, we used a custom anonymization script that supports the following operations: prefix-preserving transform (p), zero-truncation (z), and one-to-one mapping (m). We sanitized four fields: MAC address "user", access point name "ap", and timestamps "start" and "end".

We evaluated the results for three configurations:

- *pm-pm-z*: a sanitized trace with the vendor prefix of the MAC address preserved and three least significant bytes of randomly mapped (pm), the building and floor prefix of AP identifier preserved and ID values randomly mapped (pm), and the timestamp truncated (zero) to the minute (z).

- *pz-pz-z*: a sanitized trace with the vendor prefix of the MAC address preserved and three least significant

bytes of zeroed (pz), the building prefix of AP identifier preserved and floor and ID values are zeroed (pz), and the timestamp truncated (zero) to the hour (z).

- *pm-pz-z*: a sanitized trace with the vendor prefix of the MAC address preserved and three least significant bytes of randomly mapped (pm), the building and floor prefix of AP identifier preserved and ID values zeroed (pz), and the timestamp truncated (zero) to the minute (z).

4.1.1 User-defined functions and classes

We implemented the *DataDescriptor* as a set of four subclasses of *Field*: one of type *MACAddr*, one of type *APName*, and two of type *Integer*. These correspond to the fields "user", "ap", "start", and "end" respectively. In this trace, "ap" consists of a three-tuple of 8-bit integers corresponding to building, floor, and ID, respectively; thus, for convenience, *APName* is a simple subclass of *IPv4Address*.

We needed two *NetworkObject* classes for this example: *DeviceAddress* and *AccessPoint*. A *DeviceAddress* object is defined as a unique MAC address represented in the trace, and an *AccessPoint* is a unique access point identifier represented in the trace.

To parse the raw and sanitized traces (which are in comma-separated text files) into *Trace* objects, we used an existing module built into the NetSANI framework: *TraceCSV*, which subclasses *TraceParser* to serve as a wrapper around the Python standard library CSV parsing module, `csv`.

The "adversary's knowledge" represents what the adversary knows prior to attempting deanonymization. Many existing network-based metrics are incompatible with this trace, despite its simplicity. Because this trace does not consist of sender-receiver pairs, metrics such as *combinatorial anonymity degree* (CAD) are unavailable [10]. We therefore used a non-iterative L1-similarity metric ($N = 1$) as described in Kelly et. al [10]. The whole of the adversary's knowledge is represented in the *KellyL1* subclass of *AdversaryKnowledge*, described by Equation 3, where X is an anonymized object, Y is an unanonymized object, and $z \in X \cup Y$.

$$sim(X, Y) = 2 - \sum_{z} |P(X = z) - P(Y = z)| \qquad (3)$$

Rather than iterate over the sanitized trace several times, we instead chose to use a threshold value of sim_{min} to represent the minimum acceptable value of the metric before an adversary may be able to gain unintended information. With 2 as the maximum value that $sim(X, Y)$ may take, we let $sim_{min} = 1.8$, similar to prior work by Coull [6].

We implemented *MyUtilityComp* to reflect the use of trace data by a researcher involved in analysis of wireless-network mobility behavior. We used an existing paper from Balazinska [1] to simulate various uses of SNMP trace data for use in wireless network mobility and usage research, and assigned utility scores for the ability of an object to contribute to a given experiment present in the paper. Utility scores increase with granularity (e.g., knowledge of user distribution across all access points is weighted more heavily than user distribution across buildings). For a listing of utility value assignments for this experiment, see Table 3, and for details of each experiment, see Balazinska [1].

Table 3: Value Assignment to Utility Classes

Class	*Value*
users present per day per building	0.1
users present per hour per building	0.2
number of access points per floor	0.1
number of access points per building	0.2
users associated with building at given time	0.1
users associated with floor at given time	0.2
users associated with AP at given time	0.3
idle time per building	0.1
idle time per AP	0.2
buildings visited per user	0.1
access points visited per user	0.2
building prevalence per user	0.1
access point prevalence per user	0.2
building persistence per user	0.1
access point persistence per user	0.2
v_{max}	2.4

Table 4: Object Privacy and Utility (Normalized)

	Privacy		Utility
Config	*DeviceAddress*	*AccessPoint*	v
pz-pz-z	0.993	0.995	0.000
pm-pz-z	1.0	1.0	0.458
pm-pm-z	0.986	0.984	1.000

4.1.2 Tradeoff analysis/control result

Table 4 displays the analysis of the NetSANI framework and compares the sanitization configurations in terms of overall privacy and utility. The overall utility and privacy results are normalized: the fraction of objects that meet the minimum privacy threshold sim_{min}, and the utility defined by methods outlined in Section 3.2.

We see a considerable difference in utility based on the configuration of trace sanitization. We can infer that the zero utility values of the *pz-pz-z* configuration is caused at least in part by stripping away unique user identity in the "user" field, since each of the factors in the utility metric (Table 3) depend on the ability to uniquely identify users.

Configuration *pm-pz-z* instead applies a one-to-one mapping to the least significant bytes of the user's MAC address. We see a slight gain in the anonymity of both network objects; all objects' privacy values exceeded our minimum similarity threshold. From the utility results, we can confirm that the ability to maintain the identities in the "user" field is the reason for the considerable increase in utility.

Finally, in configuration *pm-pm-z*, rather than zeroing the ID number within the access point name, we applied a one-to-one mapping to that portion of the field. We may infer from the framework results that allowing the researcher access data accurate down to the access point level, we obtain our optimum utility values, albeit with a sacrifice in privacy. It is to be expected, since this configuration reveals the most information of any, that it also has the lowest privacy for both network objects analyzed.

5. DISCUSSION

The NetSANI framework can accommodate most of the currently available privacy metrics; Table 5 shows how to implement other currently available metrics in Kelly et al. [10] using the NetSANI APIs. Our framework can use network-based metrics by implementing the user-defined functions *MyKUnit*, *MyKDist* and *MyUncertainty*. Most of the network-based metrics can define *either* the representation and distance of the adversary's knowledge (e.g., Anonymity Set Size or Individual Anonymity Degree) *or* the uncertainty of an

Table 5: Implementation of metrics using NetSANI

Metrics	*MyKUnit*	*MyKDist*	*MyUncertainty*
Anonymity Set Size	implemented as binary (known or unknown)	binary distance	not needed
Individual Anonymity Degree	implemented as any types	difference between probabilities	conditional probabilities compared
Entropy Anonymity Degree	implemented as any types	any distance metric	entropies compared

object mapping (e.g., Entropy). Therefore, users can implement metrics on the adversary's knowledge using the *MyKUnit* and *MyKDist* functions, while implementing uncertainty metrics using the *MyUncertainty* function.

6. SUMMARY

In this work, we address how to analyze and control the privacy and utility tradeoff, in general, for network sanitization efforts. For this, we propose NetSANI, a network trace sanitization and anonymization framework, which consists of built-in classes and extensible user-defined functions for the analysis and control of the tradeoff in sanitization evaluation. The NetSANI framework can work on both wireless and wired network traces, and it can accommodate most of the currently available privacy and utility metrics, either collectively or separately, by providing the metrics as user-defined functions. Ultimately, NetSANI should make it easier for wireless-network researchers to share traces with the broader research community.

Acknowledgements

This paper results from a research program in the Institute for Security, Technology, and Society (ISTS), supported by DHS award 2006-CS-001-000001 and by NSF award CNS-0831409. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies of DHS or NSF.

7. REFERENCES

[1] M. Balazinska and P. Castro. Characterizing mobility and network usage in a corporate wireless local-area network. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 303–316, 2003.

[2] T. Brekne, A. Årnes, and A. Øslebø. Anonymization of IP traffic monitoring data: Attacks on two prefix-preserving anonymization schemes and some proposed remedies. In *Proceedings of the International Symposium on Privacy Enhancing Technologies (PET)*, volume 3856 of *Lecture Notes in Computer Science*, pages 179–196, 2005.

[3] J. Brickell and V. Shmatikov. The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 70–78, 2008.

[4] Cooperative Association for Internet Data Analysis (CAIDA). www.caida.org, 2008.

[5] S. Clauß. A framework for quantification of linkability within a privacy-enhancing identity management system. In *Proceedings Emerging Trends in Information and Communication Security*, volume 3995 of *Lecture Notes in Computer Science*, pages 191–205, 2006.

[6] 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)*, February 2008.

[7] S. E. Coull, F. Monrose, M. K. Reiter, and M. D. Bailey. The Challenges of Effectively Anonymizing Network Data. In *Proceedings of the Cybersecurity Applications & Technology Conference For Homeland Security (CATCH)*, pages 230–236, March 2009.

[8] Community Resource for Archiving Wireless Data At Dartmouth (CRAWDAD). www.crawdad.org, 2010.

[9] Internet measurement data catalog (DatCat). www.datcat.org, 2010.

[10] 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)*, pages 31–40, 2008.

[11] D. Koukis, S. Antonatos, and K. G. Anagnostakis. On the privacy risks of publishing anonymized IP network traces. In *Proceedings of the International Conference on Communications and Multimedia Security (CMS)*, volume 4237 of *Lecture Notes in Computer Science*, pages 22–32, 2006.

[12] D. Koukis, S. Antonatos, D. Antoniades, E. P. Markatos, and P. Trimintzios. A generic anonymization framework for network traffic. In *Proceedings of the IEEE International Conference on Communications (ICC)*, volume 5, June 2006.

[13] 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)*, pages 1–8, 2008.

[14] C. Y. Ma, D. K. Yau, N. K. Yip, and N. S. Rao. Privacy vulnerability of published anonymous mobility traces. In *Proc. of the International Conference on Mobile Computing and Networking (MobiCom)*, pages 185–196, 2010.

[15] R. Pang, M. Allman, V. Paxson, and J. Lee. The devil and packet trace anonymization. *ACM SIGCOMM Computer Communication Review*, 36(1):29–38, 2006.

[16] Protected Repository for the Defense of Infrastructure against Cyber Threats (PREDICT). www.predict.org, 2010.

[17] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *Proceedings of the International Symposium on Privacy Enhancing Technologies (PET)*, volume 2482 of *Lecture Notes in Computer Science*, pages 41–53, 2002.

[18] K. Tan, G. Yan, J. Yeo, and D. Kotz. Privacy analysis of user association logs in a large-scale wireless LAN. In *Proceedings of the 30th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM) mini-conference*, April 2011.

[19] K. Tan, J. Yeo, M. E. Locasto, and D. Kotz. Catch, clean, and release: A survey of obstacles and opportunities for network trace sanitization. In *Privacy-Aware Knowledge Discovery: Novel Applications and New Techniques*. Chapman and Hall/CRC Press, December 2010.

[20] M. Woo, J. P. Reiter, A. Oganian, and A. F. Karr. Global measures of data utility in microdata masked for disclosure limitation. *Journal of Privacy and Confidentiality*, 1:111–124, 2009.

[21] J. Yeo, K. Tan, and D. Kotz. User survey regarding the needs of network researchers in trace-anonymization tools. Technical Report TR2009-658, Dartmouth College, 2009.