Playing Devil's Advocate: Inferring Sensitive Information from Anonymized Network Traces

Scott E. Coull[*] Charles V. Wright[*] Fabian Monrose[*] Michael P. Collins[†] Michael K. Reiter[‡]

Abstract

Encouraging the release of network data is central to promoting sound network research practices, though the publication of this data can leak sensitive information about the publishing organization. To address this dilemma, several techniques have been suggested for anonymizing network data by obfuscating sensitive fields. In this paper, we present new techniques for inferring network topology and deanonymizing servers present in anonymized network data, using only the data itself and public information. Via analyses on three different network datasets, we quantify the effectiveness of our techniques, showing that they can uncover significant amounts of sensitive information. We also discuss prospects for preventing these deanonymization attacks.

1 Introduction

In order to provide a sound scientific foundation for some types of research on network systems, it is imperative that trace and log data be made publicly available for verification and comparison of results. Indeed, the lack of public datasets has been identified as a key weakness in current networking research [22], and work is underway to construct a large, shared repository of network trace data [23].

To protect the privacy of end users and the security of the networks themselves, it is clearly necessary to obscure certain identifying information (*e.g.*, IP addresses) before it is published. Several methods for such network trace *anonymization* have recently been proposed (*e.g.*, [19, 26, 10]). The authors of these schemes generally acknowledge both the difficulty in creating an anonymization scheme free from all forms of information leakage, and that more in-depth analysis of anonymization schemes is needed (*e.g.*, [19]). Lacking such analysis, however, the community continues to move forward with the release of network data anonymized using these and similar techniques (*e.g.*, [23]).

In this paper, we conduct such an analysis in order to inform the continuing debate over the release of anonymized network trace data. Specifically, we detail new techniques to infer sensitive information from such traces, and using these techniques we show that current anonymization techniques might not protect sensitive information as well as initially thought. While the existence of some classes of information leakage from individual header fields had been acknowledged [19, 26, 10], these attacks are often dismissed as easily mitigated by changes to the anonymization policy. Our study demonstrates that there are more substantial forms of information leakage that inherently compromise current anonymization methodology. Fundamentally, these leaks result from patterns that naturally occur as artifacts of useful anonymized network data. The ability to subvert current anonymization methodology with only public information sources provides attackers with a dangerous new tool for *passive* network reconnaissance.

To demonstrate the validity of these attacks, we explore the information leaked by the state-of-the-art anonymization system of Pang *et al.* [19] when used under the recommended policy settings. As a concrete example, we apply our techniques to anonymized traces from three distinct networks with different traffic mixes and numbers of hosts. Our results show that sensitive network topology and host behavioral information can be extracted from patterns observed in the anonymized IP addresses and port numbers. Using this topology and behavioral information, along with purely public information from search engines and DNS records, we are able to deanonymize between 28 and 100% of targeted servers. While we choose to focus on the Pang *et al.* approach, we also show that our findings are equally applicable to a number of other recent techniques in the literature, thereby calling into question the overall benefits provided by current anonymization strategies.

The remainder of this paper is organized as follows. In Section 2, we review various methods for anonymizing net-

[*]Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA; {coulls,cwright,fabian}@cs.jhu.edu

[†]CERT/Network Situational Awareness, Software Engineering Institute, Carnegie Mellon University; mcollins@cert.org

[‡]Electrical & Computer Engineering Department and Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA; reiter@cmu.edu

work data, as well as the pragmatic requirements on the resulting data. The primitives used in our inference techniques are introduced in Section 3, and their specific use in inferring topology information and selectively deanonymizing hosts is presented in Section 4. In Section 5, we apply our techniques to anonymized network data taken from a variety of networks, and show that our techniques are effective in deanonymizing hosts in a passive manner using only publicly available information. Finally, we suggest possible mitigation strategies in Section 6, and conclude with directions for future work in Section 7.

2 Background and Related Work

Anonymization systems, in general, are designed to accomplish three primary goals while providing usable network trace data. These goals, as described by Pang *et al.*, are aimed at preventing *i*) the true identities of specific hosts from being leaked such that an audit trail of user activity could be formed, *ii*) the true identities of internal hosts from being leaked such that a map of supported services can be constructed, and *iii*) the leakage of specific security practices within the publishing organization's network [19]. Several methods of data obfuscation are utilized in practice to achieve these goals [2, 20, 26]. *Destruction* of information in a field (*e.g.*, by outright removal from the dataset) is often used to anonymize the payload fields of packet traces. *Fixed transformation* provides a single pseudonym value for all values of the field. *Variable transformation* allows for different pseudonym values based on the context of the field. One example of variable transformation is to replace an IP address with different pseudonyms based upon the type of application layer protocol, such as HTTP or SMTP. *Typed transformation* allows for a single pseudonym value for each distinct value of the original field. Prefix-preserving address anonymization is a version of typed transformation.

There have been several attempts to provide anonymization techniques appropriate for network trace data. The TCPurify tool, for example, implements the destruction by randomizing IP addresses, and removing packet payloads [28]. By randomizing the addresses, TCPurify provides protection for the true identity of hosts, but also destroys useful network prefix information. TCPdPriv uses another approach known as prefix-preserving anonymization to provide typed transformation of the addresses, whereby the longest common prefixes within the trace data are mapped to the same pseudonym prefix [27]. The prefix table used by TCPdPriv is created on a per trace basis, and therefore it is likely that a particular host will map to different pseudonym addresses in different network traces.

Recently, Fan *et al.* developed a cryptographic approach, known as CryptoPAn, for the creation of prefix-preserving pseudonym addresses without the need for a prefix table [10]. The cryptographic approach to anonymization uses keyed hash functions to produce consistent anonymized prefix-preserving addresses, thereby allowing for the publication of several traces with consistent network information throughout. The approach of Fan *et al.* [10] has been widely used in several recent systems that provide policy-based anonymization of trace data. These policy-based systems allow for the application of any of the transformations described above based on the security policy provided by the user. One such example, suggested by Pang and Paxson, provides a method for defining policy-based anonymization scripts utilizing the Bro scripting language [20]. A different approach, called CANINE [25, 26], operates on NetFlow [6] data, but limits the abilities of the dataset publisher by allowing only for certain classes of anonymization on a particular subset of fields. In particular, the CANINE's prefix-preserving anonymization only allows the use of CryptoPAn, and provides no methods to implement other anonymization techniques.

Of course, while the aforementioned definitions suggest various ways in which network data may be sanitized, the use of anonymization techniques in practical network research forces the anonymization systems to adhere to several requirements with regard to the resulting data. Here, we attempt to summarize what appears to be a minimum set of requirements that show up repeatedly in networking research.

One of the most obvious and intuitive requirements is that the network and hardware addresses provided within the trace data be consistently anonymized, within and possibly across traces. Several areas of research (*e.g.*, traffic matrix estimation [5, 11, 21, 34], characterization of connection and packet arrival processes [15], and other work that rely on counting the number of distinct hosts in a trace over some period of time [17, 16]), require that metrics be evaluated on a per host, or per network basis. Without consistent anonymization, these metrics could be applied only to each individual trace, rather than to the dataset as a whole. As such, the resulting data would be of little value to researchers in search of large, realistic, datasets — indeed, the main impetus behind the current trend towards making more anonymized datasets available in the first place. For the remainder of this paper, we refer to this requirement on anonymized data as the *pseudonym consistency requirement*.

Similarly, while payload destruction has been standard practice in providing privacy, the transport, network, and link layer headers typically remain intact. Use of header information remains central to providing a flexible dataset for use in many areas of fundamental network research (*e.g.*, the effects of packet loss and reordering on TCP dynamics [14, 13, 1, 30, 32]). To remain appealing to the network re-

	Pang *et al.*	CANINE	TCPdPriv	TCPurify	
Transformation					
Destruction	✓	✓	✓	✓	
Fixed Transformation	✓	✓			
Variable Transformation	✓	✓	✓	✓	
Typed Transformation	✓	✓	✓		
Requirements met					*Examples*
Pseudonym Consistency Requirement	✓	✓	✓		[11] [21] [34] [15]
Header Requirement	✓		✓	✓	[13] [14] [1] [30]
Transport Protocol Requirement	✓	✓	✓	✓	[17] [16]
Port Number Assumption	✓	✓	✓	✓	[12] [24]

Table 1. Summary of anonymization techniques and the corresponding requirements they meet

search community, the anonymization approach must therefore respect this requirement for header information, which we herein refer to as the *header requirement*. In addition, records corresponding to transport layer traffic must be present within the dataset to facilitate studies on round trip times, reassembly, and fragmentation. This minimum amount of transport layer traffic is subsequently referred to as the *transport protocol requirement*.

Finally, the ability to uniquely map well-known application layer services to their constituent port numbers is often used in practice for protocol classification schemes (*e.g.*, [12, 24]). While the requirement that port numbers faithfully map to their related well-known services may seem onerous, we note that there are still many ways to infer the correct service being offered from anonymized or destroyed port numbers. Recent research, for example, has indicated that application layer services can be accurately identified through the use of timing and size information [31], as well as the unique behaviors of the application layer protocols [9, 18, 7]. The ability to map port numbers to services is subsequently called the *port number assumption*.

A cursory examination of anonymized network data repositories, such as CRAWDAD [8], reveals that these requirements are frequently relied upon in practice. Of all the trace anonymization techniques suggested to date [28, 27, 25, 19], we focus on what we believe to be the most comprehensive approach, namely that of Pang *et al.* [19]. We believe their approach best meets the pragmatic requirements outlined previously, and is arguably the most flexible in satisfying the privacy of publishing organizations. A comparison of the anonymization systems, their methods of anonymization, and their adherence to the pragmatic requirements is given in Table 1.

Additionally, our choice to examine the Pang *et al.* anonymization system is further motivated by a weakness in CryptoPAn [10]. Specifically, the prefix-preserving methodology of CryptoPAn anonymizes addresses such that any given bit of the anonymized address is dependent on all previous bits of the unanonymized address. This dependence causes a single deanonymization to affect all anonymized addresses that share a prefix with the true address. In fact, Brekne *et al.* recently demonstrated how active probing attacks can be used to systematically undermine the CryptoPAn anonymization scheme [4, 3].

Pang *et al.*, however, only use CryptoPAn to anonymize addresses external to the enterprise where the trace was collected. To anonymize internal addresses, Pang *et al.* break the dependency across bits by anonymizing the subnet and host portion of the addresses as independent blocks using a pseudo-random permutation. Although the separation of the anonymization into two independent permutations may still lend itself to attacks where the adversary learns the deanonymized host's true address, only that host's specific subnet is compromised in this case; that is, no information is discovered about other host mappings and subnet permutations from this single deanonymization. A more detailed review of the CryptoPAn weakness can be found in Appendix A.

3 Primitives

The deanonymization techniques we develop in this paper are based on a few simple primitives. These primitives, with the exception of Subnet Clustering, were first proposed by Xu *et al.* [33] to provide summary information about the traffic at backbone routers. We apply these techniques to determine the statistically significant values within the network trace data. In addition, we also apply our own Subnet Clustering technique to infer the subnets of networks from only the observed addresses. Discovery of the proper subnets and their sizes aid in the passive inference of topology from the provided trace data. However, unlike Xu *et al.* [33], we apply these techniques to anonymized network traces, and our deanonymization techniques require additional analysis on top of that provided by these tools.

For the remainder of the paper, we represent network data, in the form of packet traces or NetFlow logs, as a set of connections, C. Each connection $c \in C$ is described by a feature vector $\langle c_1, c_2, ..., c_k \rangle$. In our case, $k = 4$, and the features are: c_1 = source IP address, c_2 = destination

IP address, c_3 = source port number, and c_4 = destination port number. Several of our techniques employ analysis of the normalized entropy ($\overline{H}$) of one or more features of the data. The normalized entropy provides a measure of the uncertainty of outcomes for the given feature, relative to the maximum possible uncertainty of a random variable taking on the same values. Specifically, we denote the i^{th} feature across all connections in C as C_i, and treat it as a random variable. Then, if C_i takes on $n_i > 1$ distinct values in our data, we calculate its normalized entropy as

$$\overline{H}(C_i) = \frac{H(C_i)}{\log n_i}$$

where for each c, the probability $P_C(C_i = c_i)$ is the number of occurrences of value c_i in feature C_i, divided by $|C|$.

Thus, normalized entropy values near zero indicate a highly peaked distribution for that attribute, while values near one indicate a nearly uniform distribution of values. Given an anonymized dataset, we will use this measure to find a set of hosts within that data whose presence is most significant—the so-called 'heavy-hitters'. Intuitively, if a few IP addresses occur much more frequently than others in the data, then the normalized entropy of the addresses will be low. Because of this fact, we can employ an iterative algorithm, shown in `Algorithm 1`, to obtain the set of the most significant IP addresses in the data. This algorithm works by repeatedly removing very frequent (initially, of probability $> t_P$) values from the distribution and recomputing the normalized entropy until the distribution becomes sufficiently uniform, bringing the normalized entropy above a given threshold, t_H. Finding the 'heavy-hitters' this way allows us to focus our later deanonymization efforts on the most prevalent hosts in the network trace.

Algorithm 1 Find-Heavy-Hitters(Connections C, thresholds t_H and t_P)

```
HeavyHitters ← $\emptyset$
// Examine both source IP address ($C_1$) and destination IP address ($C_2$)
for $i = 1, 2$ do
    $k \leftarrow 0$
    $C' \leftarrow C$
    while $\overline{H}(C'_i) < t_H$ do
        for all $c \in C'$ do
            if $P_C(C_i = c_i) > 2^{-k} t_P$ then
                HeavyHitters ← HeavyHitters $\cup \{c_i\}$
                $C' \leftarrow C' \setminus \{c\}$
        $k \leftarrow k + 1$
return HeavyHitters
```

Dominant State Analysis Xu *et al.* also propose a novel approach for determining the most characteristic behaviors for a given host, known as Dominant State Analysis [33]. We apply this technique to the network traffic associated with the heavy-hitters found as described above to develop a *behavioral profile*, or set of recurring network activities that best characterize these hosts. These behavioral profiles can then be used as a means of fingerprinting various hosts based on their behaviors. The concept of profiling behaviors has been examined by others, including Karagiannis *et al.* and Aiello *et al.*, and has been found to be a practical mechanism for classifying hosts into logical groupings based on the characterization of their traffic [16, 17].

To find these behaviors, for each 'heavy-hitter' address x, we begin with a simple behavioral profile: $src\ address = x$. Recall that c_1 is the source IP address of connection c. Then, denoting the set of connections with $c_1 = x$ as C^x, we reorder the remaining features i such that $\overline{H}(C^x_i) \leq \overline{H}(C^x_{i+1})$. Then, for each feature in the set of connections, $C^x_{i=2..4}$, we look for values of c_i whose conditional probability with the current profile exceeds our threshold t, and append that value to the profile vector for further consideration. The algorithm extends these profiles in an iterative fashion until no value meeting our threshold can be found, or all features have been examined. The output of this Dominant State Analysis (`Algorithm 2`) for each 'heavy-hitter' IP address is a set of feature vectors describing its behavioral profiles. After performing the Dominant State Analysis on the source IP addresses, we repeat the same process using the destination IP addresses.

Subnet Clustering The subnets associated with the IP addresses found in the network data are not always made available, and thus we must consider alternate methods of determining this information. The determination of the subnets found in the anonymized data is an important step in the creation of network topology maps and the deanonymization of hosts, as it allows us to distinguish the distinct areas of the network where the data was collected. Subnets can be inferred from the set of IP addresses present within the network data using an application of the *k-means* clustering algorithm. Note that this process does not deanonymize the addresses in the trace, but it does take advantage of the prefix-preserving anonymization to extract information about the underlying network topology.

A naïve solution for discovering subnets within the data is to simply look for groupings of contiguous addresses separated by large gaps. Of course, an arbitrary cutoff would have to be determined to specify the required gap between the groupings. This gap size would necessarily relate to a predetermined subnet size, which may not be uniform across all subnets. It is therefore likely that such a scheme would improperly group addresses if a variety of subnet sizes were present. Our approach, on the other hand, is able to choose the proper sizes for a variety of subnets without the use of a priori information on the subnets sizes being grouped. To achieve that, we automatically determine the best subnets based on the observed data and an initial ap-

Algorithm 2 Dominant State Analysis for Heavy-Hitter x

```
FinishedProfiles $\leftarrow \emptyset$
CurrentProfiles $\leftarrow \{\langle x \rangle\}$
$i \leftarrow 2$
// examine each feature, in order of increasing normalized entropy
while $i \leq 4$ do
    NewProfiles $\leftarrow \emptyset$
    for all profile $\in$ CurrentProfiles do
        ExtendedProfiles $\leftarrow \emptyset$
        for all $c \in C^x$ do
            if $P_{C^x}(C_i^x = c_i | \text{profile}) > t$ then
                ExtendedProfiles $\leftarrow$ ExtendedProfiles $\cup$ $\{\text{profile} || \langle c_i \rangle\}$ // where || indicates vector concatenation
        if ExtendedProfiles $= \emptyset$ then
            FinishedProfiles $\leftarrow$ FinishedProfiles $\cup$ $\{\text{profile}\}$
        else
            NewProfiles $\leftarrow$ NewProfiles $\cup$ ExtendedProfiles
    CurrentProfiles $\leftarrow$ NewProfiles
    $i \leftarrow i + 1$
FinishedProfiles $\leftarrow$ FinishedProfiles $\cup$ CurrentProfiles
return FinishedProfiles
```

proximation of the density of addresses within the subnet through the use of unsupervised learning techniques, in our case, the *k-means* algorithm.

Specifically, we treat IP addresses as 4-dimensional vectors, where each element of the vector corresponds to one octet of the IP address as written in dotted decimal notation. To determine cluster membership, we use a modified Euclidean distance with bitwise exclusive-OR instead of subtraction as our distance metric. The dimensions corresponding to the octets are exponentially weighted such that the left-most octet carries the most weight. This weighting ensures that the hierarchical nature of the subnetting is preserved in the clustering algorithm [1].

We note that *k-means* clustering requires that the number of clusters be specified a priori; however, since we have no way of inferring the number of expected subnets, nor their density within the 4-dimensional space, we first evenly divide each of the octet dimensions into m blocks. In doing so, we make an approximate guess about the length of the subnet prefix. The initial centroids are placed at the boundaries of these partitions. Therefore, we create $k = m^d$ initial centroids where m is the number of partitions and d is the number of dimensions to which the partitioning was applied. Notice that as we increase m, we allow for greater density of subnets by decreasing the spacing among initial centroids [2]. Also, by including or excluding dimensions from the initial partitioning scheme, we can control how many octets participate in the creation of initial clusters.

After this initial step, we iteratively recompute the centroid and the corresponding cluster membership. This process continues until the membership of the clusters reaches a steady state. This iterative refinement of the clusters makes it possible to accommodate for variable length subnets. In particular, at the beginning of the refinement process, the clusters start in an unoptimized state where some addresses that reside within the same subnet appear in different clusters due to the inaccuracy of our initial approximation. However, as the refinement continues, new cluster centroids are created that better represent the addresses within each cluster, and the process eventually converges toward a set of stable centroids for the observed subnets in the data. This ensures that the addresses within a given cluster all reside in the same subnet without requiring exact initial centroid placement.

Upon completion, each nonempty cluster represents a subnet in the underlying network, and for each cluster we calculate its subnet address as the *longest common prefix* shared by all IP addresses in the given cluster. As with any application of the *k-means* algorithm, the results must be empirically evaluated to ensure proper selection of initial clusters to approximate the anonymized address space. In our evaluation, the Subnet Clustering technique achieves $> 96\%$ accuracy across all datasets used within this paper without significant tuning of the initial cluster centroids.

We reiterate that the primitives do not actually deanonymize anything by themselves; the application of the Dominant State Analysis algorithm can reveal typical behaviors for each anonymized IP address in the trace, and the Subnet Clustering technique can derive a list of anonymized subnet addresses, but more work is required to infer a map of the network and to deanonymize hosts.

[1]For example, using this weighted metric we compute the distance between the IP addresses 192.168.1.2 and 10.0.0.10 as $\sqrt{((192 \oplus 10) * 2^3)^2 + ((168 \oplus 0) * 2^2)^2 + ((1 \oplus 0) * 2^1)^2 + ((2 \oplus 10) * 2^0)^2}$.

[2]In practice, the parameter m can be tuned based on the observed behavior of the Subnet Clustering output to ensure it properly accommodates the density of addresses present in the given data.

4 Information Leakage

In what follows, we show that not only can a map of the network be recovered from anonymized network data, but we are also able to deanonymize hosts in the trace through the use of behavioral profiling. Such successful deanonymization provides for a new method of passive network reconnaissance and undermines the stated goals of existing approaches to anonymization. We note that the information required to mount these attacks follows the requirements on the anonymized dataset laid out in Section 2.

4.1 Recovering network topology

Network topology can be inferred by first determining the locations within the network where each of the traces were captured (i.e, the *observation points*), then identifying the routers at each location, and finally examining the traffic to infer the connectivity between them.

To associate traces with observation points, we first identify the network subnets which are present on the local area network where the trace was recorded. We then regard two traces as coming from the same observation point if they have one or more subnets in common. If the anonymized trace data contains Address Resolution Protocol (ARP) traffic, we can find the local subnets by performing subnet clustering (see Section 3) on the IP addresses in the ARP requests and replies. When given only IP traffic, subnets surrounding the trace's observation point can be inferred by performing subnet clustering on all IP addresses found in the trace and taking the *n* most frequently occurring subnets in the results as the observation point's subnets. The parameter *n* can be chosen to represent a best estimate of the number of subnets for the observation point.[3]

Next, routing devices can be identified by the hardware address used in the link layer headers. For each subnet at each observation point, we examine TCP and UDP traffic sent from local hosts to out-of-subnet hosts and record the destination hardware address found in the link layer header as the gateway for that subnet. Similarly, routers can be found by recording the hardware addresses that appear to have multiple network addresses associated with them over some time period δ. To avoid also flagging hosts that use the Dynamic Host Configuration Protocol (DHCP) as routers, this time period should be shorter than a typical DHCP lease. In practice, this has not been an issue in any of the traces we have examined, and we typically set δ to be on the order of several seconds.

[3]For example, with an initial guess of $n = 5$ for the Johns Hopkins University trace in the example below, we find the top two subnets share a common prefix and both occur twice as often as the third most frequent subnet. The similarity of the prefixes and the substantial difference in number of occurrences suggest that we should instead set $n = 2$.

Inferring the routes taken by observed traffic can be useful in understanding the interconnection of observation points at the network layer. To characterize the routes handled by the discovered routing devices, the subnet clustering technique is applied to the source and destination IP addresses of TCP and UDP traffic that transit the devices. The unique pairs of source and destination subnets, as well as the hardware addresses of the other routers to which traffic is sent, can be used to better understand the routing topology of the network in question.

Lastly, the network layer topology can be reconstructed by examining the routers and gateways for each observation point, and the hardware addresses with which they communicate. Formally, the network can be represented as a graph G consisting of nodes V and edges $E \subseteq V \times V$. Then if the hardware address for router s is associated with some other discovered routing device t, we add an edge $e = (s, t)$ to E in the graph. Once the topology map is created via this matching process, the routes can be superimposed using the hardware addresses to characterize the actual routes taken by network layer traffic. This inference can also occur for link layer topology if the network traces provide ARP traffic. In this case, one may logically assume that only the IP addresses observed in the ARP packets are members of the local link layer network. Therefore, any traffic that is present at an observation point not destined to or sent from a local host must be transiting that observation point. Specifically, the path taken from hosts in a given observation point to their gateway router can provide an approximation of the topology, including the presence of switches and bridging devices.

Note that to perform this topology inference, the provided data must at least meet the header and transport protocol requirements discussed in Section 2. The header requirement is necessary to retrieve the anonymized hardware addresses of the routing devices within the topology, and therefore datasets that do not meet this requirement, such as NetFlow data, are not vulnerable to the topology inference attack. The transport protocol requirement provides the minimum network traffic from which the inference of network topology information is made.

4.2 Inferring Host Behavior

While the inference of network topology information is certainly disconcerting, the anonymization of host addresses should make it difficult to accurately map this topology to real-world addresses. If, however, the behaviors of the anonymized hosts can be uniquely mapped to the behaviors of their real-world counterparts, the attacker can begin deanonymizing portions of the topology.

To infer the host behavior, the unique, recurring traffic to and from a particular host is characterized as a *behavioral*

Host	Src IP Address	Src Port	Dst IP Address	Dst Port	$\overline{H}$(Src Port)	$\overline{H}$(Dst IP)	$\overline{H}$(Dst Port)
web server	128.220.231.207	80	*	*	0.0	0.93	1.0
SSH server	128.220.231.147	22	17.138.176.51	*	0.0	0.0	1.0
web client	128.220.231.17	*	*	80	1.0	0.85	0.17

Table 2. Example behavioral profiles observed in the departmental network

profile. To create these profiles, we take as input a list of k-dimensional feature vectors $c = \langle c_1, \ldots, c_k \rangle$ describing a set of connections C. The set of significant source and destination addresses, or 'heavy-hitters', is obtained from the connections in C via `Algorithm 1`. Next, Dominant State Analysis is applied to the connections $c \in C$ containing each of these significant addresses to determine the set of dominant behaviors that comprise the given host's behavioral profile. Upon termination, the set of behavioral profiles for each significant address is returned (see `Algorithm 2`).

Like the topology inference attack, our inference of host behavior requires that the transport protocol requirement be met, as well as the pseudonym consistency requirement and port number assumption. In particular, the pseudonym consistency requirement allows us to accurately build profiles for the hosts in the data, which would otherwise be difficult if their identities in the dataset changed. The port number assumption simply allows us to easily determine what services a given host offers when building our profiles. Unlike the topology attack, these requirements can be met regardless of the form that the network data takes. Therefore, while the use of NetFlow data hampers the topology inference attack due to its lack of header information, it still meets all requirements for our behavioral inference attacks and remains vulnerable to deanonymization via behavioral profiling.

4.3 An Example

For concreteness, we illustrate the application of the aforementioned techniques on an anonymized trace from the Johns Hopkins University (JHU) network. In general, deanonymization begins by first identifying an interesting service, and finding an appropriate host offering that service to deanonymize. The goal is to create a behavioral profile query based on public information regarding services being run on the target host, the perceived popularity of the host, and its possible locations within the network topology. Developing this profile often requires the use of public information sources, such as DNS or web search engine queries, to specify unique information about the host. Beyond that, it is simply a matter of matching hosts within the trace dataset that best match the derived criterion.

The trace in question contains TCP traffic from a single observation point between a departmental network and the main campus network. It contains packets from over 26,000 TCP connections involving 38 internal hosts and more than 1,500 external hosts. The topology inference technique indeed finds a single observation point with subnets of 128.220.231.0/24 and 128.220.116.0/26. These subnets are the two anonymized subnets present within the trace, which is correct given the location of the observation point just outside of the departmental network. The inference technique also finds one router and one gateway that are directly connected to one another, with the departmental network behind the router. The routes discovered are trivial considering the topology of the observation point.

Anonymized address	Dest. address $\overline{H}$	Real hostname
128.220.231.207	0.93	simnet1
128.220.231.50	0.93	skdnssec
128.220.231.168	0.87	spar
128.220.231.121	0.85	-

Table 3. Deanonymization of significant Web servers in local network trace

We can also deanonymize many of the web servers within the departmental network. To find the web servers in this database, we must first create an estimate of the behavioral profile for which we are looking. Table 2 shows example behavioral profiles for three 'heavy-hitters', found via `Algorithm 1`, in the department's network.

Because all the web servers in the trace have behavioral profiles with similar TCP ports and destination IP addresses, we must use the normalized entropy of the destination IP addresses to distinguish between the web servers based on their popularity. More popular servers communicate with a wider range of clients, and thus have a higher normalized entropy ($\overline{H}$) for their destination addresses. To find the names of the most popular web servers in the departmental network and to estimate their relative popularity, we use Alexa.com's popularity-based search engine.[4] By counting the number of references made to each hostname over the first $l(= 5)$ pages of results, we arrive at the following ordered ranking of web servers: $\langle$*simnet1: 30, skdnssec: 11, spar:9*$\rangle$.

The rankings provide strong indication that there are ap-

[4]See `http://www.alexa.com`

Trace	Type	Connections	Internal Hosts	Dates	Duration
JHU	packet trace	2.63e+04	38	Apr 2005	24 hr
CERT-A	NetFlow	1.04e+05	20	Aug 2006	4 hr
CERT-B	NetFlow	1.40e+05	24	Aug 2006	4 hr
LBNL	packet trace	8.71e+05	5976	Oct 2004 - Jan 2005	60 hr

Table 4. Summary information for anonymized datasets

proximately three significant web servers within the departmental network. Due to their popularity, it is appropriate to assume that the normalized entropy of the destination addresses they service will be nearly one, indicating a large variety of hosts. By querying the behavioral profiles derived from the local trace data for a web server with a destination address normalized entropy greater than $\gamma = 0.85$, we obtain the list of anonymized addresses shown in Table 3.

By pairing the ranking of web server addresses in the departmental subnet found via behavioral profile to the three hostnames found via the Alexa.com search, we obtain the correct mapping between the real hostnames and the anonymized addresses, which are verified by our original, unanonymized trace. This mapping is given in the last column of Table 3. Overall, we are able to deanonymize three of the seven significant web servers in the trace (43%), and two of the three mail servers (66%). Moreover, these mappings have correctly deanonymized one of the discovered anonymized subnets to 128.220.247.0/24, thereby reducing the uncertainty of the remaining anonymized hosts.

5 Evaluation

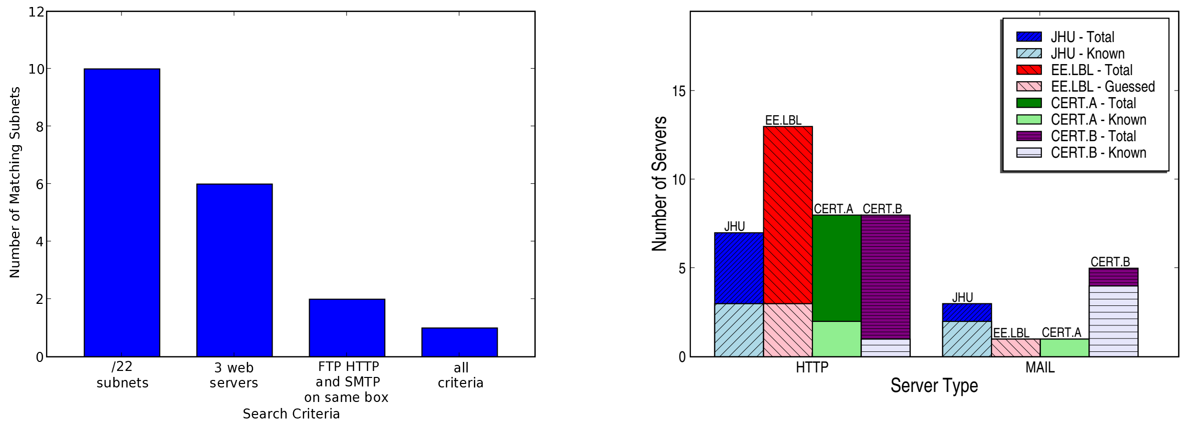
Our inference techniques are evaluated on network traces for three distinct networks, including the Johns Hopkins network described in Section 4. Table 4 provides a breakdown of each dataset. The anonymized dataset provided by Pang *et al.* [19] is the most diverse, and contains packet traces that were collected at Lawrence Berkeley National Laboratory (LBNL) over the course of several months. The LBNL traces are arranged into 131 individual trace files anonymized by the system described in [20, 19]. Several network and transport layer protocols are present within the anonymized trace data, which include only the anonymized link, network, and transport layer headers. Payload data is deleted, and fields are anonymized per the policy defined in [19]. Pang *et al.* also remove the packets generated by routers and certain security devices to prevent their discovery. In addition to the anonymized trace data, Pang *et al.* provide meta-data to ensure sound measurement practices. This meta-data includes the identification of anonymized subnets, anonymized gateway addresses for those subnets, and corrupt packets. This meta-data is used in our evaluation only to provide a form of ground truth for our analysis.

One solution to preventing the inference of topology information is to simply publish NetFlow logs, a data format consisting of tuples of transport and network layer information gathered from packet headers on a per connection basis, rather than complete packet traces. In fact, release of anonymized NetFlow data is gaining popularity and therefore it seems only prudent that we validate our techniques against both NetFlow and packet trace data. To show the viability of our techniques on NetFlow data, we also evaluate a dataset obtained from CERT containing NetFlow logs of two distinct /24 networks taken over four hours on a single day. To anonymize this NetFlow data, we apply prefix-preserving anonymization of addresses (as specified by Pang *et al.* in [19]), remove connections of less than four packets in size to compensate for scanning behavior, and explicitly remove references to routers and network security devices. Since this is NetFlow data, topology inference is not possible, but we are able to apply our deanonymization techniques just as in the LBNL dataset.

To streamline our analysis of these datasets, we parse packet trace or NetFlow logs and automatically dump all information about the connections (C), the 'heavy-hitter' hosts, topology information, and their behavioral profiles into a relational database for fast and easy querying. The database can then be queried for the features of the behavioral profile, the normalized entropy values, specific connection information to deanonymize hosts, etc. Hence, no active probing of the network under scrutiny is necessary.

5.1 Network Topology

To infer the network topology of LBNL, we employ the techniques described in Section 4. Our techniques find 29 distinct observation points with a total of 31 associated enterprise subnets. The subnets found through the Subnet Clustering technique agreed with those provided in the meta-data, with the exception of one subnet whose size was overestimated, thereby providing 96% accuracy. This overestimate appears to have occurred due to an improper estimate of the longest common prefix computed from a small number of distinct addresses within the ARP trace data. Interestingly, the anonymization system used to provide privacy for the LBNL data makes extra effort to mask



(a) Uniqueness of the *ee.lbl.gov* subnet

(b) Deanonymization results for 3 networks

Figure 1. Deanonymization

the presence of routing devices by eliminating their traffic. Our inference techniques, however, are able to infer their positioning due to the necessary presence of hardware addresses in link layer headers. When combined with additional information—like the placement of hosts and allowed routes—maps of the network topology provide attackers with information on the security posture of the publishing organization.

5.2 Deanonymization

To demonstrate the effectiveness of behavioral profiles in the selective deanonymization of anonymized hosts, we provide a deanonymization of the HTTP server for the Bro IDS project [29] and other servers within its subnet, including *ee.lbl.gov*. We pick these targets as candidates for deanonymization since public information about these servers is readily available—and hence, can be easily used to unmask the anonymization.

We begin by querying DNS records for the addresses of the Bro web server (*www.bro-ids.org*) and related hosts found in public information sources. The results of these queries show that *www.bro-ids.org* resides in the same subnet as the *ee.lbl.gov* domain, which includes SMTP, HTTP, and FTP services on *ee.lbl.gov* and HTTP services on *ita.ee.lbl.gov*. Knowledge of these services was gleaned only from Google and Alexa searches, as well as DNS records for the *ee.lbl.gov* and *bro-ids.org* domains.

By inferring the subnet size from the addresses provided by DNS records, the set of possible subnets within the trace data is reduced to ten /22 subnets out of the twenty nine total subnets (see Figure 1(a)). Furthermore, if we consider that the target subnet contains at least three ‘heavy-hitter’ web servers, we find six subnets in the dataset that match this criteria. Finally, by noting that the *ee.lbl.gov* server provides SMTP, HTTP, and FTP services, we find only two subnets that contain such a server. By combining each of the characteristics of our target subnet, only one of the subnets present in the dataset matches all criterion. In essence, the combination of required subnet size, unique services offered by *ee.lbl.gov*, and the specific mix of significant HTTP servers fingerprints the subnet within the anonymized trace.

Since the *ee.lbl.gov* server provides such a unique mix of services, it is fairly easy to detect among the significant hosts present in the subnet. The two remaining significant HTTP servers exhibit similar behaviors, except with respect to the normalized entropy for the destination address dimension. Thus, one host serves a wider variety of clients than the other. Due to the nature of the *www.bro-ids.org* and *ita.ee.lbl.gov* web servers and their respective presence within web search engines, it seems reasonable to assume that *www.bro-ids.org* serves a wider variety of hosts while *ita.ee.lbl.gov* has a more narrow appeal.

Another HTTP server within this subnet handles only a few distinct clients, but serves a large number of connections overall. These connections carry relatively few packets, thus indicating the presence of a web page or application that performs several small HTTP connections to retrieve data. Upon inspection of the Google results for the *ee.lbl.gov* domain, we suspect this machine might be *froggy.lbl.gov*. The Froggy site features a CGI web application which generates short bursts of HTTP connections as a function of the CGI usage. Table 5 provides some sample deanonymization results for various publicly available services, including *www.bro-ids.org*. The deanonymizations are supposition, as there is no ground truth avail-

Target Host	Candidate Mappings	Search Criteria	Auxiliary Information
www.bro-ids.org *ita.ee.lbl.gov*	131.243.95.232 131.243.93.0	CIDR Block = 131.243.0.0/16 Source Port=HTTP $(\bar{H}) = (< .05, > .5, > .5)$ Same subnet as ee.lbl.gov	Knowledge of HTTP, DNS A records Public subnet info.
ee.lbl.gov	131.243.94.172	CIDR Block = 131.243.0.0/16 Source Ports=HTTP, SMTP, FTP $(\bar{H}) = (> .1, > .5, > .5)$ co-located w/ ita.ee.lbl.gov,www.bro-ids.org	Knowledge of SMTP, DNS A records Public subnet info.
froggy.lbl.gov	131.243.95.168	CIDR Block = 131.243.0.0/16 Source Port=HTTP; same subnet as ee.lbl.gov	Knowledge of HTTP protocol
lbl.gov	128.3.164.249	CIDR Block = 128.3.0.0/16 Source Ports=SMTP, finger co-located w/ postalnp.,imap4.,smtp.lbl.gov	DNS MX records, Mail info, public web/subnet info.
smtp.lbl.gov *postalnp.lbl.gov*	128.3.164.248 128.3.164.57	CIDR Block = 128.3.0.0/16 Source Port(s) = SMTP co-located w/ lbl.gov, imap4.lbl.gov	DNS MX records, Mail, public subnet info.
imap4.lbl.gov	128.3.164.15 or 128.3.164.194	CIDR Block = 128.3.0.0/16 Source Port(s) = IMAP, POP3 Same subnet as lbl.gov	Public web info.

Table 5. Sample deanonymizations of select public services

Dataset	Observation Point	Dominant Source Ports
JHU	*JHU*	22 (SSH), HTTP (80)
CERT-A	*CERT-A*	20 (FTP), 21(FTP), 25 (SMTP), 53 (DNS), 80 (HTTP), 443 (HTTPS)
CERT-B	*CERT-B*	25 (SMTP), 53 (DNS), 80 (HTTP)
LBNL	*Routing Core*	161 (SNMP)
	ee.lbl.gov	22 (SSH), 80 (HTTP), 123 (NTP), 137 (WINS), 161(SNMP), 427 (SRVLOC), 515 (LPR), 524,793,795-800 (Unknown)
	Client Network	22 (SSH), 25 (SMTP), 67 (BOOTP), 68 (DHCP), 80 (HTTP), 137 (WINS), 138 (NetBIOS), 139 (NetBIOS), 161 (SNMP), 427 (SRVLOC), 445 (Microsoft-DS), 497 (Unknown), 548 (Apple Filesharing), 575:696:878 (Unknown)

Table 6. Dominant source ports for selected observations points

able, but the supporting evidence provides confidence in our deanonymization results.

For the JHU and CERT datasets, we are able to provide a similar method of deanonymization by mapping behavioral profiles of true hosts to their anonymized counterparts; more importantly, we can validate these deanonymizations via ground truth. Figure 1(b) shows our overall deanonymization performance across the four datasets. For each trace, the results show the number of deanonymized servers for each protocol ("known" or, in the case of *ee.lbl.gov*, "guessed") compared to the total number of significant servers for that protocol.

It was initially believed that information leakage attacks of the kind presented herein were too difficult for an adversary to achieve in practice [19]. However, our results indicate that behavioral profiling is a plausible method for deanonymizing a variety of network traces. Our success rate ranged from 66% to 100% for significant SMTP servers and 28.6% to 50% for the significant HTTP servers in the subnets we examined. While the results for HTTP servers may not seem impressive, consider that *no* level of deanonymization is expected from the anonymized datasets. Also, consider that any server providing unique public services, such as FTP or DNS, would obtain similar deanonymization results to the SMTP servers. While these results are troubling, we argue that a determined adversary, with the luxury of time, would be able to deanonymize even more servers.

Once this deanonymization occurs, the published data can be used to characterize the services offered by that host. Moreover, if the prefix-preserving anonymization were based on CryptoPAn, rather than the Pang *et al.* model, these deanonymizations would not only affect the deanonymized host, but would also reduce the uncertainty about all other hosts in the dataset and deanonymize the neighboring anonymized address (see Appendix A).

In addition to deanonymizing select hosts, we are able to characterize the traffic present within the various observation points. Table 6 characterizes the protocols within select observation points in the three datasets and provides deeper insight into the presence of important servers and the general usage of the observation points.

6 Mitigation

Undoubtedly, there are mitigation strategies that could be implemented by publishing organizations to foil the inference attacks demonstrated in this paper. A simple strategy for preventing the creation of network topology maps is simply to publish only anonymized NetFlow logs, or to remove link layer headers from packet traces, thereby removing the hardware addresses of routing devices. Additionally, the exclusion of ARP traffic makes it more difficult to discover link layer topology information.

By remapping port numbers, publishing organizations hinder the ability of an attacker to directly infer services offered within the private network and makes the creation of behavioral profiles more difficult. Note, however, that methods do exist to determine the true service based on the application layer protocol behavior, e.g., [31, 9, 18, 16, 7]. Also, because behavioral profiling is less effective when similar profiles are shared among a large number of hosts (note, for example, the results for HTTP versus SMTP servers), it may be possible to improve the privacy of the dataset by removing the hosts whose behavior is most unique. Finally, hiding the true identity of the publishing organization makes it more difficult for an attacker to gather the necessary information to mount the inference attacks presented. However, this begs the question as to how difficult it would be for an attacker to still infer the publishing organization.

It must be noted, however, that these mitigation strategies destroy the research value of the data to varying degrees. For instance, header information and port numbers are important to several areas of network research. The removal of unique hosts may dramatically affect traffic characteristics within the dataset. Furthermore, hiding the identity of publishing organizations can make verification of results obtained from the trace data difficult.

Of course, there are other, non-technical, means of providing privacy protection for data publishers. One method—employed by data repositories such as Predict and CRAWDAD [23, 8]—is to impose legal requirements that the data will be used appropriately, and that adequate levels of protection will be put in place for accessing the data. Another possibility is to require that data remain only on secured servers to which researchers have remote access. Violations of these requirements, particularly the former, can be difficult to detect, however. An even more cumbersome approach is to require that analysis be performed by trusted individuals from the publishing organization itself, thereby preventing any direct access to the data by third parties. While inefficient, this method allows for better privacy than is offered by anonymization methodologies currently in use. That said, it may also significantly impede many types of research.

7 Conclusion and Future Work

In this paper we provide new algorithms for inferring sensitive information from network traces that have been anonymized using state-of-the-art techniques. Our work shows that network topology information can be inferred as an artifact of usable network packet traces, and that behaviors of hosts are an important piece of identifying information that can be leveraged to subvert an anonymization system. At the least, our evaluation shows that selective recovery of sensitive information from anonymized network data is not as difficult as once thought [10, 19]. Moreover, obfuscating behavioral and network topology information is not a trivial task. Hence, although there are substantial benefits from releasing anonymized data, publishers of such data need to be more cognizant of the thin line between the utility of the data and the privacy it provides.

Our results naturally raise questions as to what network trace data, if any, should be released for research. This, however, is a policy decision that we believe each network owner must make for himself. The results reported in this paper should not be construed as an indication of our opinion on the propriety of releasing network data in any particular form. Rather, our goal has simply been to inform the continuing debate over the release of network traffic for research purposes. It is our belief that the creation of any overarching policy should be formulated through the combined perspective of the network security research community as a whole.

On the technical front, our study underscores the need for a better understanding of the conditions under which a particular anonymization method (e.g., prefix-preserving anonymization for IP addresses) may provide an adequate privacy solution for a particular set of network traces. For instance, while this paper has shown that servers are at risk of deanonymization through behavioral profiling, it remains unclear to what extent the privacy of clients is threatened. Indeed, if the lack of client privacy turns out to be a serious issue, then a prudent course of action is to re-examine which types of transformation are appropriate solutions to the problem. As part of future work, we intend to explore a formal framework for examining this question, and in particular, for expressing the privacy properties of anonymization techniques in general.

Acknowledgments

The authors would like to thank Michael Bailey and Patrick McDaniel for their suggestions for improving an earlier draft of this work. We also thank the anonymous reviewers for their insightful comments. This work is supported in part by NSF Grant CNS-0546350.

References

[1] F. Baccelli and K. B. Kim. TCP Throughput Analysis Under Transmission Error and Congestion Losses. In *Proceedings of IEEE INFOCOM*, pages 2833–2843, March 2004.

[2] M. Bishop, B. Bhumiratana, R. Crawford, and K. Levitt. How to Sanitize Data. In *Proceedings of the* 13th *IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 217–222, June 2004.

[3] T. Brekne and A. Årnes. Circumventing IP-Address Pseudonymization. In *Proceedings of the* 3rd *IASTED International Conference on Communications and Computer Networks*, October 2005.

[4] 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 Workshop on Privacy Enhancing Technologies*, pages 179–196, May 2005.

[5] H. Chang, S. Jamin, Z. M. Mao, and W. Willinger. An Empirical Approach to Modeling Inter-AS Traffic Matrices. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, October 2005.

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

[7] M. P. Collins and M. K. Reiter. Finding Peer-to-Peer File-Sharing Using Coarse Network Behaviors. In *Proceedings of the* 11th *European Symposium on Research in Computer Security*, pages 1–17, September 2006.

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

[9] J. Early, C. Brodley, and C. Rosenberg. Behavioral authentication of server flows. In *Proceedings of the* 19th *Annual Computer Security Applications Conference*, pages 46–55, December 2003.

[10] J. Fan, J. Xu, M. H. Ammar, and S. B. Moon. Prefix-preserving IP Address Anonymization: Measurement-based Security Evaluation and a New Cryptography-based Scheme. *Computer Networks*, 46(2):253–272, 2004.

[11] A. Gunnar, M. Johansson, and T. Telkamp. Traffic Matrix Estimation on a Large IP Backbone - A Comparison on Real Data. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, pages 149–160, October 2004.

[12] P. Gupta and N. McKeown. Packet Classification Using Hierarchical Intelligent Cuttings. In *Proceedings of Hot Interconnects VII*, pages 147–160, 1999.

[13] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Measurement and Classification of Out-of-Sequence Packets in a Tier-1 IP Backbone. In *Proceedings of IEEE INFOCOM*, pages 1199–1209, April 2003.

[14] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley. Inferring TCP Connection Characteristics Through Passive Measurements. In *Proceedings of IEEE INFOCOM*, pages 1582–1592, March 2004.

[15] H. Jiang and C. Dovrolis. Source-Level IP Packet Bursts: Causes and Effects. In *Proceedings of ACM SIGCOMM Internet Measurement Conference*, October 2003.

[16] T. Karagiannis, K. Papagiannaki, and M. Faloutsos. BLINC: Multilevel Traffic Classification in the Dark. In *Proceedings of ACM SIGCOMM*, pages 229–240, August 2005.

[17] P. McDaniel, S. Sen, O. Spatscheck, J. V. der Merwe, W. Aiello, and C. Kalmanek. Enterprise Security: A Community of Interest Based Approach. In *Proceedings of Network and Distributed Systems Security*, February 2006.

[18] A. W. Moore and D. Zuev. Internet Traffic Classification Using Bayesian Analysis Techniques. In *Proceedings of ACM SIGMETRICS*, pages 50–60, June 2005.

[19] R. Pang, M. Allman, V. Paxson, and J. Lee. The Devil and Packet Trace Anonymization. *ACM Computer Communication Review*, 36(1):29–38, January 2006.

[20] R. Pang and V. Paxson. A High-Level Environment for Packet Trace Anonymization and Transformation. In *Proceedings of SIGCOMM*, pages 339–351, August 2003.

[21] K. Papagiannaki, N. Taft, and A. Lakhina. A Distributed Approach to Measure IP Traffic Matrices. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, pages 161–174, October 2004.

[22] V. Paxson. Strategies for Sound Internet Measurements. In *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, pages 263–271, October 2004.

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

[24] S. Singh, F. Baboescu, G. Varghese, and J. Wang. Packet Classification Using Multidimensional Cutting. In *Proceedings of ACM SIGCOMM*, pages 213–224, August 2003.

[25] A. Slagell, J. Wang, and W. Yurcik. Network Log Anonymization: Application of Crypto-PAn to Cisco NetFlows. In *Proceedings of NSF/AFRL Workshop on Security Knowledge Management*, September 2004.

[26] A. Slagell and W. Yurcik. Sharing Computer Network Logs for Security and Privacy: A Motivation for New Methodologies of Anonymization. In *Proceedings of SECOVAL: The Workshop on the Value of Security through Collaboration*, pages 80–89, September 2005.

[27] TCPdPriv. `http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html`.

[28] TCPurify. `http://irg.cs.ohiou.edu/~eblanton/tcpurify/`.

[29] V.Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks*, 31(23-24):2435–2463, December 1999.

[30] R. Wang, G. Pau, K. Yamada, M. Y. Sanadidi, and M. Gerla. TCP Startup Performance in Large Bandwidth Delay Networks. In *Proceedings of IEEE INFOCOM*, pages 796–805, March 2004.

[31] C. V. Wright, F. Monrose, and G. M. Masson. On Inferring Application Protocol Behaviors in Encrypted Network Traffic. *Journal of Machine Learning Research*, Special Topic on Machine Learning for Computer Security, To appear, 2006.

[32] Y. Xiang, J.-C. Liu, K. G. Shin, and W. Zhao. On the Modeling and Optimization of Discontinuous Network Congestion Control Systems. In *Proceedings of IEEE INFOCOM*, pages 2812–2820, March 2004.

[33] K. Xu, Z. Zhang, and S. Bhattacharyya. Profiling Internet Backbone Traffic: Behavior Models and Applications. In *Proceedings of ACM SIGCOMM*, pages 169–180, August 2005.

[34] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An Information-Theoretic Approach to Traffic Matrix Estimation. In *Proceedings of ACM SIGCOMM*, pages 301–312, August 2003.

A Crypto-PAn

To illustrate the flaw in Crypto-PAn, consider the case where a single anonymized address $a' = a'_1a'_2 \ldots a'_{n-1}a'_n$ has been deanonymized to reveal its true address $a = a_1a_2 \ldots a_{n-1}a_n$. Then, because of the prefix-preserving nature of Crypto-PAn's transformation, the attacker also learns the pseudonym for the address $a^* = a_1a_2 \ldots a_{n-1}\bar{a_n}$. Because a and a^* share a prefix of length $n-1$, the attacker knows that their pseudonyms, a' and $a^{*\prime}$, must also share an $n-1$ bit prefix. Therefore the only valid pseudonym for a^* must be $a^{*\prime} = a'_1a'_2 \ldots a'_{n-1}\bar{a'_n}$. In general, for any remaining anonymized host with pseudonym address b', the attacker can remove $m+1$ bits of uncertainty about the host's true address b by deanonymizing a host which shares an m-bit prefix with b.

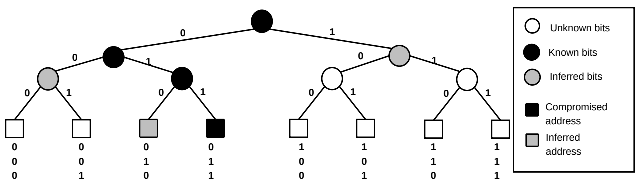


Figure 2. Binary tree with nodes indicating bits of the anonymized address. Root indicates left-most bit, and shaded nodes indicate compromised bits where the mapping to the unanonymized address is known.

For concreteness, consider a simple example with $n = 3$, given by the binary tree in Figure 2. At the onset, the anonymization maintains 7 bits of uncertainty with regard to the permutation used to anonymize the addresses. In other words, there are 128 possible permutations that are allowed by the prefix-preserving anonymization, and each of those permutations is equally likely. Suppose that we are then able to deanonymize the address `011`, as shown in Figure 2. In this case, the bits represented by the shaded nodes are now compromised and therefore lose their uncertainty. The remaining, unknown bits provide 4 bits of uncertainty for this anonymization, or 16 valid permutations for the remaining addresses. In this example, a single deanonymization provides an 87% reduction in the anonymized address space. This reduction can lead to further deanonymization as the set of possible permutations remaining allows the attacker to better refine their search for public information and thereby deanonymize more hosts.

As can be seen by this simple example, within conserved anonymization spaces a single deanonymization can be devastating to the CryptoPAn anonymization system. For instance, consider the application of CryptoPAn to the Johns Hopkins University trace given in Section 4. This trace contains a subnet with an address space of 2^8 and an associated anonymized address space with 255 bits of uncertainty. To deanonymize this entire space requires a significant number of individual deanonymizations, 128 to be exact. In practice, however, organizations frequently allocate addresses in contiguous fashion. This concentrates the density of hosts within a conserved subtree of the entire anonymization space, in essence reducing the entire tree with 255 bits of uncertainty to a substantially smaller subtree. These realistic scenarios underscore the danger of using the prefix-preserving anonymization methodology advocated by Fan *et al.* in CryptoPAn, and shows that the methodology is certainly much more dangerous than the analysis in [10] suggests. Recent work by Brekne *et al.* takes advantage of this very weakness by using active probing attacks to deanonymize a small subset of hosts within anonymized datasets, which in turn leads to the complete deanonymization of all hosts within the dataset [4, 3].

We note that the approach of Pang *et al.* does not fall prey to this attack, as they instead use a customized prefix-preserving scheme that decouples the host and subnet portions of the IP address. Specifically, Pang *et al.* use CryptoPAn only to anonymize external addresses, but performs a pseudo-random permutation on the subnet and host portions of internal addresses. This decoupling still allows for the compromised host's subnet portion of the address to be deanonymized with the use of our attacks, but the other hosts' bits retain their uncertainty as do all other subnets within the dataset.