Analyzing Privacy in Enterprise Packet Trace Anonymization[*]

Bruno Ribeiro[*], Weifeng Chen[+†], Gerome Miklau[*], and Don Towsley[*]

[*]Computer Science Department
University of Massachusetts
Amherst, MA, 01003
{ribeiro, miklau, towsley}@cs.umass.edu

[+]Department of Math and Computer Science
California University of Pennsylvania
California, PA, 15419
chen@cup.edu

Abstract

Accurate network measurement through trace collection is critical for advancing network design and for maintaining secure, reliable networks. Unfortunately, the release of network traces to analysts is highly constrained by privacy concerns. Several host anonymization schemes have been proposed to address this issue. Preservation of prefix relationships among anonymized addresses is an important aspect of trace utility, but also causes a number of vulnerabilities in trace anonymization. In this work we present an efficient host fingerprint attack targeting prefix-preserving anonymized traces. The attack is general (encompassing a range of fingerprinting host de-anonymization attacks proposed by others) and flexible (it can be adapted to emerging variants of prefix-preserving anonymization). Perhaps most importantly, we develop analysis tools that allow data publishers to quantify the worst-case vulnerability of their traces given assumptions about the kind of external information that is available to the adversary. Using this analysis we quantify the trade-off between privacy and utility of alternatives to full prefix-preserving anonymization.

1 Introduction

Accurate network measurement through trace collection is critical for advancing network design and for maintaining secure, reliable networks. While the technological means exist to collect and analyze traces, the release of these traces to analysts is highly constrained by privacy concerns. Network trace data can include information about individuals (their personal data, communication habits, evidence of location), about an enterprise (its structure, organization, business practices), as well as about the underlying system (network topology, active services, security practices, etc.).

Because trace data is so sensitive, its publication and analysis should be allowed only if the data is protected by an anonymizing transformation over its IP addresses. A desired characteristic of such transformation is to make the trace resistant against host re-identification attacks. This address anonymization policy is followed by a number of public trace repositories [1–7]. Packet content is virtually always removed since its inclusion for unencrypted communication would constitute a severe privacy violation. The next step is to obscure source and destination IP addresses. This can be done with any one-to-one mapping, however the utility of trace data to researchers often requires that prefix relationships be maintained. For example, peer-to-peer systems measurement [8], worm outbreak analysis [9], the study of router forwarding caches, and prefix-based clustering on traces all depend on address locality and require the preservation of prefix relationships. Prefix-preserving anonymization was first implemented as a feature of the Unix tool `tcpdpriv` [10]. A number of other techniques have since been developed for efficient prefix-preserving packet trace anonymization [11–13].

In this work we focus on enterprise IP traces; that is, traces collected near the perimeter of the Internet, at a gateway owned by an enterprise or institution. This is currently the most common type of trace available (because ISPs rarely release traces of backbone traffic). Numerous enterprise IP traces have been released in anonymized form [1–7].

Our work presents a systematic attack targeting prefix preserving anonymization which can be efficiently executed by an adversary in possession of a modest amount of public information about the network. The attack has a number of advantages over others recently proposed. In particular, we provide an analysis tool that allows data publishers to quantify the worst-case vulnerability of their trace given assumptions about the kind of external information available to the adversary. We demonstrate the effec-

[*]This research has been supported in part by the NSF under grant awards CT-ISG0534323, CNS-0627642, ANI-0325868, and by CAPES (Brazil). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

[†]This work was performed while Weifen Chen was with the Computer Science Department at the John Jay College of Criminal Justice, New York, NY, 10019

tiveness of the attack and analysis tools on real enterprise traces.

Background

The active addresses in an enterprise network with 2^k total IP addresses can be represented as leaves in a binary tree of height k, which we call an *address tree*. Figure 1(a) (left) shows an example of an address tree in a 4-bit address space. An anonymization function α is a bijective function taking an IP address x to its anonymized counterpart $\alpha(x)$ anywhere in the IPv4 address space. An anonymization function *preserves prefixes* when two addresses x and z agree on the first i bits if and only if $\alpha(x)$ and $\alpha(z)$ also agree on their first i bits. Figure 1(a)(right) shows one possible prefix-preserving anonymization of the original address tree.

Attack overview Our attack focuses on the central threat in trace publication: *host de-anonymization*, in which an adversary is able to associate the actual IP address, x, with the obscured IP address, $\alpha(x)$, present in a published trace. The attack is based on the observation that if the adversary can gather information about the existence of addresses in the real network, a set of structural constraints emerge that lead to serious disclosures. This attack is most effective when applied to the internal addresses of an enterprise trace, which are easily distinguishable from external addresses, and whose true identity are often the most sensitive to the enterprise.

For example, assume the adversary is able to discover all active addresses in the real network, as illustrated in Figure 1(a)(left). Using structural relationships alone, the adversary can de-anonymize some of the addresses shown in the anonymized tree. For each anonymized address, its possible de-anonymizations are shown in Figure 1(b). Two anonymized addresses (0010 and 1110) are uniquely de-anonymized based on the structure of the network alone. Intuitively, address 1110 is distinguishable because it is alone in its 2 bit subtree, but it is adjacent to a complete subtree of 4 nodes. For other addresses, unique de-anonymization is not possible and a set of possible candidates remain, as shown in Figure 1(b). For each anonymized address y in the $\{1000, 1001, 1010, 1011\}$, its de-anonymization $\alpha^{-1}(y)$ must be one of $\{0000, 0001, 0010, 0011\}$. These addresses remain hidden in a crowd.

Continuing the example, if the adversary has additional information about properties of real addresses, the possible de-anonymizations are further constrained. For example, the adversary may know that host 0011 has port 22 open and witnesses frequent traffic on this port. Traffic on port 22 can be observed in the trace for some hosts but not others. The host properties (known about real hosts, observed for trace hosts) can be represented as labels on the leaf nodes (in Figure 1(a) the labels are simply A and B). Assuming that only addresses with common labels should match[1] the possible de-anonymizations are further limited, as shown in the final column of Figure 1(b). Address 1001 and its sibling 1000, previously hidden in a crowd of size 4, are now uniquely de-anonymized.

The effectiveness of this attack depends, first, on the completeness and accuracy of an adversary's external information about hosts in the real network (whether they are active, properties of the traffic, etc.) and second, on properties of the network itself (the allocation of addresses, the uniformity of host properties). As suggested by this example, the adversary's inference process involves matching the leaves of the address trees in a way that respects constraints imposed by the tree structure as well as the host labels. In practice, an adversary's information will be incorrect for some hosts. As a result, the attack algorithm we describe is based on cost-based approximate tree matching [14, 15].

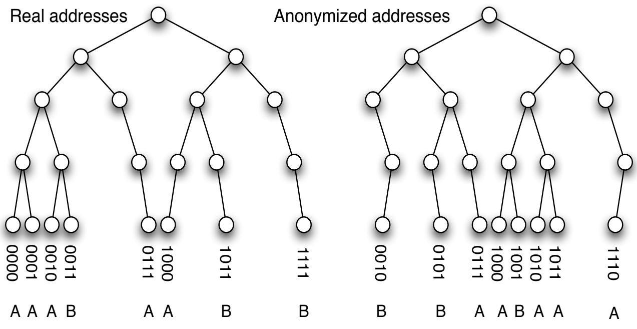
Related Work Vulnerabilities in IP trace anonymization are by now widely recognized. Slagell et al. [16] provide a thorough categorization including *injection attacks* (in which a recognizable pattern of traffic is introduced to the trace by the attacker), *fingerprinting attacks* (in which properties of real addresses are matched to addresses exhibiting those properties in the trace), and *structure recognition* (in which, for example, the de-anonymization of one address is used to narrow the possible de-anonymizations of other addresses when prefix structure is preserved).

Xu et al. [12] evaluated the impact of structure recognition attacks from small sets of de-anonymized hosts by calculating the overall average number of discovered bits in the anonymization. Brekne et al. [17] describe both fingerprinting and injection attacks on prefix-preservation for powerful adversaries assumed capable of forging trace traffic and granted knowledge of the traffic distribution.

Despite these vulnerabilities, there are few techniques designed to resist them. One notable exception is the work of Pang et al. [18], in which scans are removed from traces to mitigate injection attacks, and prefixes are not preserved for addresses internal to the enterprise. The choice to forgo full prefix-preservation for internal addresses sacrifices the utility of the trace in order to resist attacks like the one presented in this work and others mentioned above. Instead of pure prefix preservation, the subnet and host portions of each address are anonymized separately, but subnet relationships are preserved. We refer to this as an example of *partial prefix preservation*.

Recently, Coull *et. al.* [19] discussed attacks on this presumably safer anonymization. They describe a fingerprinting attack which creates a behavioral profile for hosts using public information sources (e.g., DNS or web search engine queries), the perceived popularity of the target host, and its

[1] This assumption does not always hold. We consider consider this issue fully in Section 3.



(a)

Anonymized address y	Possible de-anonymizations, $\alpha^{-1}(y)$	
	structure alone	structure and labels
0010	{1111}	{1111}
0101	{1000,1011}	{1011}
0111	{1000,1011}	{1000}
1000	{0000,0001,0010,0011}	{0010}
1001	{0000,0001,0010,0011}	{0011}
1010	{0000,0001,0010,0011}	{0000,0001}
1011	{0000,0001,0010,0011}	{0000,0001}
1110	{0111}	{0111}

(b)

Figure 1. (a) A four-bit address space: (left) an address tree representing active hosts; (right) an anonymized address tree with prefixes preserved. (b) Anonymized addresses along with the set of possible de-anonymizations which can be inferred using structure alone (column 2) and using structure and attribute labels (column 3).

possible locations within the network topology. Some parts of the network topology hidden by the anonymization are recovered and some distinguished public servers are identified.

The literature also presents fingerprint attacks that complement host de-anonymization attacks and profile anonymized flows for traffic fingerprints. Koukis *et. al.* [20] identifies if an anonymized flow is related to a given set of known Web server contents and Coull *et. al.* [21] identifies if a given anonymized flow contains specific Web pages from a pre-defined set of Web sites. Note that these last two works are not host de-anonymization attacks and, therefore, are out of the scope of our work.

Concurrently with our work, Coull et al. [22] recently proposed a new de-anonymization technique. Trace and external fingerprint attributes used in [22] are represented by random variables and can capture a richer set of traffic characteristics than the fingerprints presented in Section 3. However, the de-anonymization algorithm in [22] is not provably optimal. This differs from our de-anonymization algorithm which is guaranteed to reach an optimal solution. It is a topic of future work to include the fingerprints described in [22] into our framework.

Contributions The existing fingerprint attacks on prefix-preserving anonymization for host de-anonymization suffer from two shortcomings which our techniques address. First, they attempt re-identification on each host independently, without reasoning collectively over the entire trace. Different from these attacks, which are more general and do not assume prefix preservation, our attack takes advantage of the constraints imposed by the prefix order to increase attack efficiency. Second, the effectiveness of the attacks is highly dependent on adversary knowledge which is nearly impossible for trace publishers to estimate. We make the following main contributions to address these issues:

- We provide a unified formal framework for host fingerprint assisted de-anonymization attacks. This framework considers prefix-preservation address mapping constraints for more efficient host de-anonymization. Our attack (described in Section 2) systematically combines fingerprinting and structure recognition attacks. It can easily accommodate any form of external information (by representing it as labels on address nodes) including information acquired through injection attacks. It can be applied to both full or partial prefix preservation, and the attack is efficient: it takes a few minutes on common hardware to execute the attack on a class B network.
- In Section 3 we perform a thorough experimental evaluation of the attack on a trace collected at a large university. We show that an adversary, using probes of 9 distinct TCP port numbers, can *uniquely* re-identify 17% of the active hosts in the trace, while about 50% of the hosts have candidate sets of size no greater than 4.
- Our work is the first to provide a tool that has the potential to help data publishers release more useful and secure traces. This tool comes from the formal analysis of our attack in Section 4. Given a set of (assumed) observable host properties as its sole input, this tool identifies all addresses vulnerable to a fingerprint attack. Note that, in spite of our ignorance on how the most skillful adversary attacks the trace, we are still capable of assessing the worst-case de-anonymization damage this adversary makes.
- In Section 5 we evaluate the impact of the partial prefix preservation recommended in [18]. Our techniques allow us to quantify the improvement in privacy gained

through this technique, but we also show that one aspect of it (randomization of subnets) sacrifices trace utility without increasing privacy.

The worst-case analysis of our generic fingerprinting attack has important practical consequences. It can form the basis for an efficient tool that data publishers can use to conservatively assess the risk of publishing traces for their particular network, to identify the most vulnerable hosts, or even to guide address allocation within their network.

2 De-anonymization attack

In this section we describe our attack on prefix-preserving anonymization and its connection with approximate tree matching algorithms. We assume in this section that the attack is applied to full prefix preservation and return to partial prefix preservation in Section 5. The attack consists of three major steps:

Step I The adversary derives traffic fingerprints for each host in the anonymized trace.

Step II The adversary collects information from external public sources to construct fingerprints for network hosts.

Step III The adversary uses the fingerprints obtained in steps I and II above to recover the anonymization function, α, in part or in full.

2.1 Attack description

Step I: *Deriving trace fingerprints.* There are many types of traffic information that can be recovered from an anonymized packet trace. For instance, an address y is "active" when the trace contains at least one packet with source address y. Adversaries can also gather information about which TCP services address y provides by examining the trace for packets with source address y from well known service port numbers. Information on host operating system types can also be obtained [23]. Other types of information include flow sizes, packet sizes, and packet interdeparture times. Appendix A includes more details on such attributes. We refer to the collection of traffic information about an anonymized address y as the *trace fingerprint* of y.

Step II: *Mining external fingerprints.* In this second step, the adversary gathers external information about the real network of hosts believed to be present in the trace. The host properties collected are similar to the those presented in Step I. The adversary may gather information from reverse DNS queries, URL analysis, Web-crawling, active probing, or other public sources. Network service port status, such as active/inactive Web and E-mail TCP ports, are among the easiest types of external information to gather from a network. Such information can be obtained by probing the network with TCP SYN packets sent to ports such as 25 (E-mail) or 80 (Web). The adversary infers that IP address x runs a network service at the Web server port when x replies with a TCP SYN ACK when probed on port 80.

In what follows we provide a formal definition of the above fingerprints. We associate a fingerprint, $F(y)$, with each anonymized address y. Here $F(y)$ is a tuple $F(y) = (Y_1(y), \ldots, Y_k(y))$ where $Y_i(y)$ takes a value from a finite set Γ_i, $i = 1, \ldots, k$. For instance, $\Gamma_i = \{\mathtt{A}, \mathtt{I}\}$ and $Y_i(y) = \mathtt{A}$ when y is "active" or $Y_i(y) = \mathtt{I}$ when y is "inactive". Let $\mathcal{A}$ be the set of all anonymized addresses and $F = \{(y, F(y)) : \forall y \in \mathcal{A}\}$ be the set of all fingerprints of a trace. Let x be an un-anonymized address of the network. External fingerprints $\Sigma(x) = (X_1(x), \ldots, X_k(x))$ are constructed in the same way as $F(y)$. Throughout this work we assume that X_i and Y_i refer to the same network characteristic such as "active address" or "TCP Web service".

Inaccuracy of external fingerprints. In a real world scenario, an adversary is unlikely to collect perfectly accurate external fingerprints. We say that an adversary is able to obtain *perfect external fingerprints* when for all x, $\Sigma(x) = F(\alpha(x))$; otherwise, fingerprints are *imperfect*. Firewalls and changes to the network are among the most common reasons for these mismatches. Firewalls can prevent adversaries from collecting reliable address attributes using active probing. Because we do not assume adversary probing is synchronized with trace collection, changes to the network can occur. Appendix B presents a compilation of fingerprint inconsistencies, each of them with a possible counter-measure that can be taken by an adversary. The compilation in Appendix B is far from complete but gives an idea of the challenges that an adversary faces.

Further, imperfect external fingerprints require that we perform an approximate matching between trace and external fingerprints. As a consequence, it is necessary to define a cost function which is applied to pairs of fingerprint attributes and acts as a measure of the adversary's certainty about the association between a trace and an external fingerprint.

Definition 2.1 (*Cost function*)**.** Let $c(\Sigma(x), F(y)) \geq 0$ be a cost function that is zero when $\Sigma(x) = F(y)$.

Step III: *Recovering the anonymization function.* The basic idea behind our attack is to find a set of de-anonymization functions that are "optimal approximations" to the correct de-anonymization function α^{-1}. In what follows we formally define this as the adversary's objective. Throughout the remainder of this paper we omit the dependency on F and Σ to simplify the notation.

Definition 2.2 (*Adversary's objective*)**.** Let $\mathbb{A}$ be the set of all valid de-anonymization functions. For de-

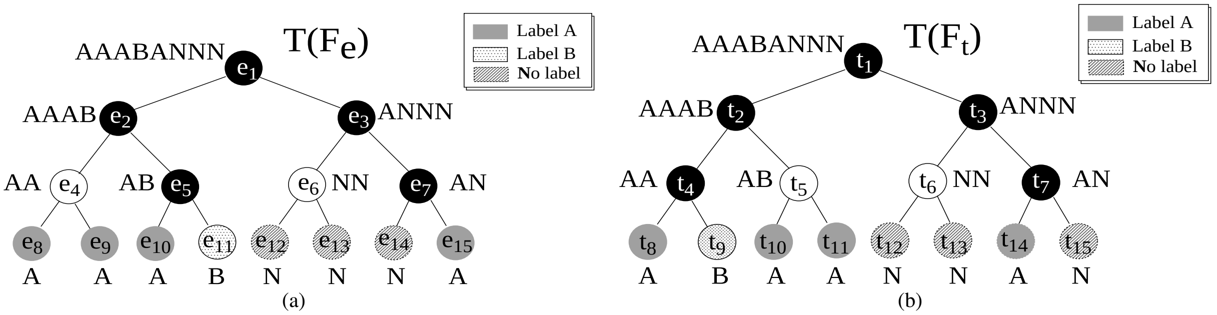


Figure 2. (a) External fingerprint tree $T(\Sigma)$ obtained from Figure 1(a)(left) example; (b) Trace fingerprint tree $T(F)$ obtained from Figure 1(a)(right) example. Symbols $\mathtt{A}$ and $\mathtt{B}$ represent the labels of Figure 1(a) and symbol $\mathtt{N}$ represents an address with no label.

anonymization function $\tau \in \mathbb{A}$,

$$\gamma(\tau, c) = \sum_{\forall y \in \mathcal{A}} c(\Sigma(\tau(y)), F(y)) \qquad (1)$$

is the cost of matching fingerprint $F(y)$ to its de-anonymized counterpart $\Sigma(\tau(y))$. The objective of the adversary is to find the set of all de-anonymization functions $\mathbb{B}(c) \subseteq \mathbb{A}$ that have the smallest cost γ for fingerprints Σ and F; or

$$\mathbb{B}(c) = \{\tau : \gamma(\tau, c) = \min_{\tau'} \gamma(\tau', c)\}. \qquad (2)$$

Note that the success of the adversary (in finding a "optimal" approximation to α^{-1}) depends on both the fingerprints obtained from Steps I and II (F and Σ respectively), and c (Definition 2.1). We can measure the adversary's success in attacking an anonymized address y using fingerprints Σ, F and function c using

$$\beta(y, c) = \{\tau(y) : \forall \tau \in \mathbb{B}(c)\}, \qquad (3)$$

which is the set of optimal matches of y (or the *match set* of y) with respect to cost function c. In what follows we present an algorithm that computes $\mathbb{B}(c)$ (equation (2)). Note that $\mathbb{B}(c) \subseteq \mathbb{A}$ and therefore $\mathbb{B}(c)$ contains only valid de-anonymization functions. Thus, Step III must enforce matchings to be consistent with the (full or partial) prefix-preserving order. For this we introduce the use of *fingerprint trees*.

Definition 2.3 (*Fingerprint tree*)**.** Let F be a set of fingerprints. A fingerprint tree $T(F)$ of a set of fingerprints F is an address-space tree in which each node is assigned a label. The label of the leaf corresponding to address y is a string of symbols $f(y) = Y_1(y) \dots Y_k(y)$. Inner vertex labels are defined by the following recursion: Let y and z be two sibling vertices in the fingerprint tree and a be their parent. Vertex a is labeled

$$f(a) = \begin{cases} f(y) \cdot f(z) & \text{if } f(y) <_{LEX} f(z), \\ f(z) \cdot f(y) & \text{otherwise,} \end{cases}$$

where "·" represents a string concatenation and $a <_{LEX} b$ if a is less than b in lexicographical order. We call inner vertex a "white" if $f(y) = f(z)$, otherwise it is called "black".

There are two types of fingerprint trees: *trace fingerprint trees*, $T(F)$, and *external fingerprint trees*, $T(\Sigma)$. We exemplify these two fingerprint trees by constructing them from the labels given in Figure 1(a). Figure 2(a) shows the subtree of $T(\Sigma)$ that encompasses addresses 0000 to 0111 of Figure 1(a)(left). Figure 2(b) shows their corresponding trace fingerprints. Inner vertices of these fingerprint trees are painted white and black according to the vertex denomination given in Definition 2.3.

2.2 Attack algorithm

Let $T(F)$ and $T(\Sigma)$ denote trace and external fingerprint trees created using fingerprints F and Σ respectively. $T(F)$, $T(\Sigma)$, and a cost function c are given as inputs to Step III, which outputs $\beta(y, c)$ for all anonymized addresses in the trace. The attack is quite straightforward. The above problem can be formulated as a trivial extension of a constrained tree edit distance problem for unordered trees, described as follows. Consider relabeling tree $T(\Sigma)$ such as to make the complete binary trees $T(F)$ and $T(\Sigma)$ isomorphic. Now consider that each leaf relabeling operation has a non-negative cost associated with it and relabeling of inner vertices have cost zero. Tree edit distance algorithms find all sets of relabeling operations over $T(\Sigma)$ with minimum total cost as to make $T(F)$ and $T(\Sigma)$ isomorphic. Using the above formulation, our problem is an instance of the tree alignment distance problem [15]. Our problem also has more specific constraints that can be used to optimize the minimum cost search. These other constraints are: $T(F)$ and $T(\Sigma)$ are complete binary trees and edit operations are restricted to relabeling operations (no insertion or removal of vertices). Using these restrictions we provide a fairly efficient algorithm for instances of the problem that the adversary is likely to encounter. Appendix C carries an example of how our algorithm works. The Java source code of our implementation of the above algorithm is also available for download.[2]

In the following section we apply our attack against real anonymized traces of a large university network. The following results were obtained for a network with 65536 addresses. In our experiments the algorithm typically finishes within a couple of minutes on a 1.83GHz Intel Centrino Duo processor.

A remark on the choice of cost function A cost function that works well in spite of the level of external fingerprint inaccuracy is likely to require knowledge of α beforehand, which is exactly what our attack is looking for. However, it is possible to learn better editing costs during the search for the edit distance matches [24]. The algorithm presented in [24] is exponential on the size of the network and thus unfit for our purposes. Further research is need to determine whether learning editing costs can be made practical for our setting.

3 Experimental results

The attack presented in Section 2 has three parameters: F, the trace fingerprints, Σ, the external fingerprints, and c, the cost function. This section explores the effectiveness of our attack and the importance of these parameters for real prefix-preserved anonymized traces. The following experiments are performed over full prefix-preserved traces collected at an university Internet gateway. The observed network has 65536 addresses. The external fingerprints, Σ, are gathered through network probing from an external host. Note that our fingerprint attack is not an injection attack as we do not assume that the anonymized trace contains any information about our probes. Although network probing is used in the following example, our attack can also be made passive, i.e., external fingerprints can be taken from sources such a address allocation maps, DNS updates, or known public Web server addresses. Throughout this section we focus solely on the de-anonymization of addresses that are internal to the network.

Trace fingerprints Our traces were collected at an Internet gateway of a class B university network. The university has two other gateways connected to the Internet that were not monitored. We attack five 24 hour traces collected between June 18th to June 22nd, 2007 from 12:00AM to 11:59PM each. In this section we present the representative results of the June 18th trace, named here *Trace-0618*. This trace has 573,037,780 packets, 9097 active internal network addresses, and was anonymized using prefix-preserving anonymization before its release.

Following **Step I** of our attack, we derive the following fingerprint attributes for each address in the trace. The "Active" attribute is set "true" for an address y if the trace has at least one packet with source address y. Nine more attributes are derived from the existence of at least one TCP SYN ACK packet for address y on ports corresponding to common services: FTP, SSH, Telnet, E-mail, Time, DNS, HTTP, POP3, and SOCKS. A final attribute (TTL) is derived from the time-to-live IP field of traffic for address y. Popular host operating systems have distinct initial TTL values, and we distinguish between four main TTL initial values: Windows, Linux, MacOS, and "Other". Any inconclusive labels are assigned "undefined". Thus we consider 11 fingerprint attributes in total. Others are of course possible.

External fingerprints Following **Step II** of our attack, we actively probe all addresses in network from an external host (in Brazil). We do not assume the adversary knows when traces are being taken, so in general there will be a temporal mismatch between the time of trace collection and the time of adversary probing. In order to test the impact of time in the accuracy of the de-anonymization, we collect fingerprints on June 14th (*External-0614*), June 15th (*External-0615*), June 18th (*External-0618*), July 19th (*External-0719*), and August 27th (*External-0827*) of 2007. We were careful to remove the probes of *External-0618* from the trace *Trace-0618* in order to avoid introducing a bias into our measurements. External fingerprint attributes correspond directly to trace fingerprint attributes and represent the network characteristics "Active", FTP, SSH, Telnet, E-mail, Time, DNS, HTTP, POP3, SOCKS, and initial TTL value. The attribute "Active" represents the absence or presence of any response to the probes from a particular address.

Cost function Unless stated otherwise, we use a cost function where the cost of editing $Y_i(y)$ into $X_i(\tau(y))$ depends on the value of $Y_i(y)$. We define this type of cost function as an *asymmetric cost* function. Our cost function reflects the natural belief that, for example, it is more likely to find an open port 80 when probing $\alpha^{-1}(y)$ without recorded traffic on port 80 in the trace, than finding traffic on port 80 in the trace with port 80 closed during the probing. The cost of editing each attribute is 1. The cost of editing an "undefined" attribute is zero. Our experiments do not indicate significant sensitivity to symmetric or asymmetric costs. For lack of space we omit the experiments demonstrating stability under these variations of the cost function.

The above choice of cost function is very naive. We use this cost function to show that full prefix-preservation is vulnerable even to a naive cost function. The choice of cost function may be decisive to the success of the attack. Later in this section we set the cost of editing all but "Active"

[2] `http://www-net.cs.umass.edu/~ribeiro/deanonymization/`

and "SSH" attributes to zero and show that this new cost function significantly impacts our results. The following (realistic) scenario exemplifies the importance of the cost function. Consider a network of N subnets, of which only $M < N$ are captured in the trace data. Certain fingerprints, like the number of active addresses, may not help to differentiate the M trace recorded subnets from the $N - M$ non-trace recorded subnets if, in their external fingerprints, all these subnets have many active addresses. Thus, a good cost function is likely to have the cost of active fingerprints weighted less than the cost of more unique fingerprints, such as e-mail servers[3].

3.1 Analysis of results

To assess the overall quality of the de-anonymization we use two metrics. The first metric is the set $M(K)$, which measures which addresses y have match sets ($\beta(y, c)$) of size no greater than K. More formally let $\mathcal{A}_a$ be the set of anonymized addresses in the trace that are active then

$$M(K) = \{y : \forall y \in \mathcal{A}_a;\ \text{and } |\beta(y, c)| \leq K\}.$$

As Σ can have imperfect information, some of the addresses in M can have match sets that do not contain the correct de-anonymized address, i.e., $\alpha^{-1}(y) \notin \beta(y, c)$. Thus, we would also like to know which addresses in M have match sets that contain the correct de-anonymized address. For this we introduce a second metric

$$V(K) = \{y : \forall y \in M(K);\ \text{and } \alpha^{-1}(y) \in \beta(y, c)\}. \quad (4)$$

We refer to $V(K)$ as the set of K-vulnerable anonymized addresses with respect to parameters Σ, F, and c. Note that $|M(K) - V(K)|$ shows the number of "errors" our attack makes. Also note that M and V are cumulative, i.e., $V(K) \subseteq V(K + 1)$. Ideally we would like to have $V(K)$ large with $M(K) - V(K)$ small for any value of K. In what follows we omit the dependency of V and M on F, Σ, and c. The choices of F, Σ, and c are clear from the context.

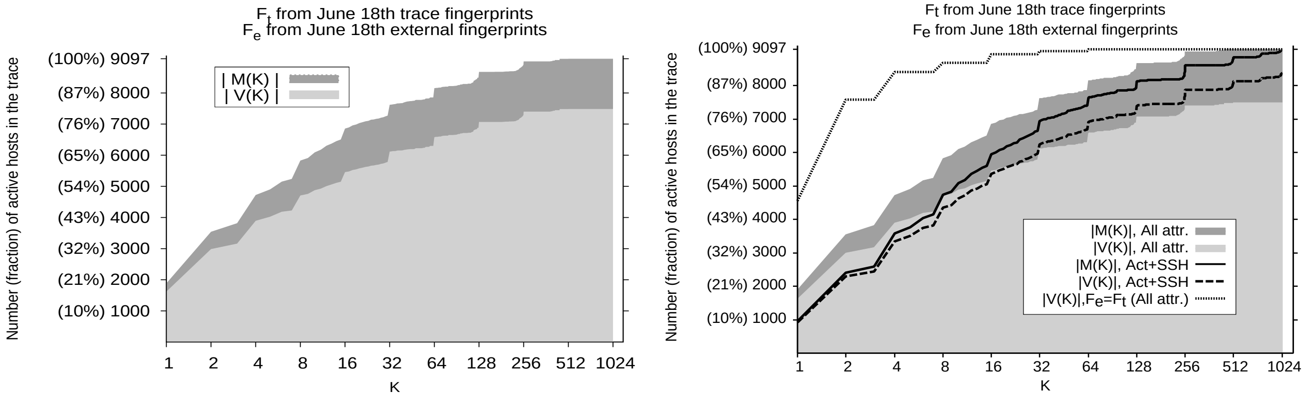
Following **Step III**, our first experiment uses external fingerprints obtained from *External-0618* and trace fingerprints obtained from *Trace-0618*. Figure 3(a) shows the values of $|M(K)|$ and $|V(K)|$ of this experiment. We observed that $V(1) = 1620$, i.e., 1620 active anonymized addresses, 17% of the number of active addresses in the trace, had their real addresses disclosed. The number of errors is $|M(1) - V(1)| = 264$ from which we conclude that our attack is fairly accurate. In this experiment, all active addresses had match sets of size at most 1024. A little more than 50% of the active addresses had matches of size no greater than 4. Among these matches, 85% of them are correct, i.e., $\alpha^{-1}(y) \in \beta(y, c)$.

Note that although the percentage of hosts uniquely identified is a small portion of the total trace nodes, this represents a very significant vulnerability. Trace anonymization is intended to conceal the true identity of all nodes present in the trace. Instead, a significant portion of them are uniquely identified, and it is possible for the adversary to identify a small set of candidate matches for a much larger percentage of nodes. If an attacker wishes to re-identify a specific trace host, reducing the set of feasible candidates to 8 or even 16 may be more than sufficient. The attacker could then acquire more specific information to refine their fingerprints and also eliminate any false re-identifications that may be present. Overall, these results show that prefix-preserving anonymization is not safe against this type of attack. In the next two subsections we analyze the relative impact of different fingerprint attributes and the impact of collecting external fingerprints before or after trace collection.

3.2 Importance of distinct cost functions

In what follows we look at the sensitivity of our attack against distinct cost functions. The following analysis also quantifies the importance of a number of fingerprint attributes. We extract some traffic attributes from *Trace-0618* and *External-0618* and present the attack results using these attributes as fingerprints at Figure 3(b). Figure 3(b) shows five curves. The two solid areas show the curves of Figure 3(a). The curve "$|M(K)|$ Act+SSH" shows results of the same attack where all attribute editing costs are set to zero except for the Active and SSH attributes. Note that setting all attribute costs other than Active and SSH to zero tend to increase the size of the matches. However, from $|V(K)|$ we see that the fraction of matches that are correct is much higher in this scenario (95% of the matches of $M(1)$ are correct, against 85% obtained with all attributes). Further analysis of our data reveals that adding the TTL type attribute to the fingerprint (with editing cost of 1) significantly increases $M(K)$ but also increases the fraction of errors for small values of K. All other attributes cause minor impacts on M and V. From our analysis we conclude that high TTL attribute costs (relative to other costs) are likely to degrade the overall accuracy of our attack. Here we also compare our results to the case where external fingerprints are perfect, i.e., $\Sigma = F$. The curve "$|V(k)|$, $\Sigma = F$(All attr.)" shows the case where external fingerprints are perfect. We study this case in detail in Section 4. Note that in this case $M(K) = V(K)$. We can see that the noise introduced by imperfect external fingerprints significantly reduces the accuracy of our attack.

[3] We would like to thank our anonymous reviewers for this nice example



(a) (b)

Figure 3. (a) The x-axis of our graph shows K, the size of the match sets. The lighter area shows $|V(K)|$ and the darker area shows $|M(K)|$. $|M(K) - V(K)|$ is the number of de-anonymization errors; (b) This figure shows five curves. The two solid areas show the curves of Figure 3(a). The curve "$|V(k)|$, $\Sigma = F$(All attr.)" considers the case where external fingerprints are perfect. The remaining curves consider fingerprints with attributes "active" and "SSH".

3.3 Attack sensitivity to late or early probing

Because the adversary does not know the date and time of trace collection, their fingerprint gathering will not necessarily occur near that time. The passage of time can affect accuracy because of changes to hosts, services running on hosts, or to address allocation policies. We find in our experiments that probing for external fingerprints as long as a month after trace collection leads to reasonable results but that accuracy does noticeably decline after about two months.

In the following experiments we use external fingerprints (all fingerprints except the initial TTL attribute) from *External-0615*, *External-0618*, *External-0719*, and *External-0827*. We drop the "TTL" attribute and use only the most perfect external attributes in order to study the impact of the passage of time on our results. Figures 4(a) and 4(b) plot $|V(K)|$ against K for the external fingerprints obtained at these dates. We start comparing the matching results from *External-0618* (day 0) and *External-0615* (day -3). Probing 3 days before the trace was collected increases the number of matching errors (match sets without the correct de-anonymized address) but does not reduce the number of correct matches. However, considering only small match sets (e.g. $K = 1$), the absolute number of matching errors is still small. Thus the attack is still fairly accurate. Comparing the matching results from *External-0618* (day 0) and *External-0719* (day 29) we see that probing 29 days after the trace is collected significantly increases the number of matching errors. It also slightly reduces the number of correct matches. Once again the absolute number of matching errors is still fairly small for small values of K and $|M(K)|$ is still high. Again, the attack is still fairly efficient. Comparing *External-0618* and *External-0827* (Figure 4(b)) shows that matching errors are four or more times greater when external fingerprints are probed 71 days after the trace is collected. The absolute number of small match sizes also reduces. The fraction of matches that are incorrect with $K = 1$ is still small, a little less than 20%, and $|M(1)| = 917$ is still large. These results suggest that the adversary has a fairly large time window to collect useful external fingerprints.

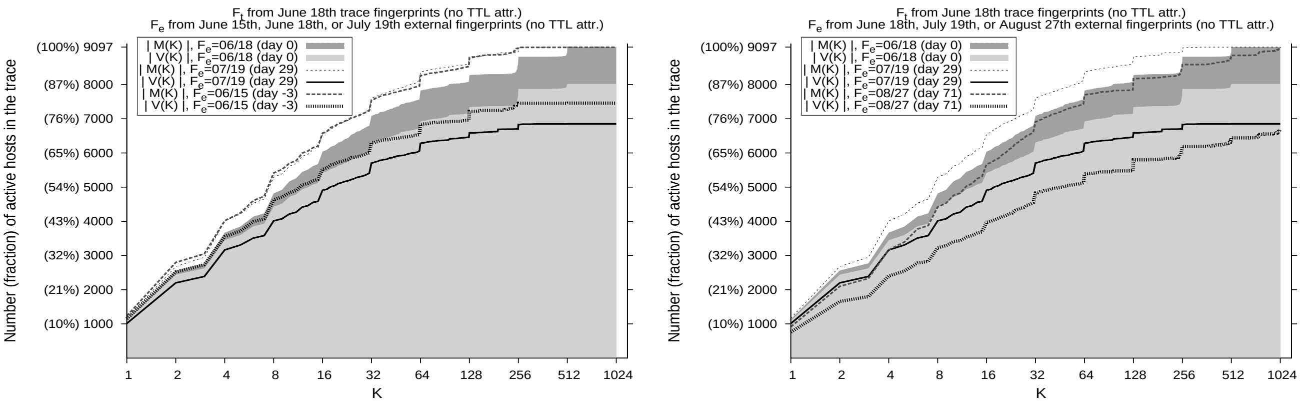
4 Worst case analysis of host de-anonymization

The experimental results presented in the previous section show that the effectiveness of the attack depends on cost function and the accuracy of the external fingerprints collected by the adversary. For hosts that remain hidden, it is not clear whether their anonymity is due to the trace transformation techniques, or due to the weakness on the part of the adversary. In this section we make the following assumptions about the data publisher:

- The data publisher has access to the set of trace fingerprint attributes used by the adversary.
- The data publisher does **not** have access to the cost function or external fingerprints used by the adversary.

Under these circumstances we present a tool that allows data publishers to efficiently calculate, prior to publication, the worst-case disclosure for a given trace and to pinpoint vulnerable addresses.

We apply this analysis to the experimental trace from the last section showing the unique de-anonymizations jump from 17% to 44% of trace nodes. In addition, we apply



(a) (b)

Figure 4. (a) These six curves show the values of $|M(K)|$ and $|V(K)|$ for all attributes except "TTL". The two solid areas represent external fingerprints collected at the day of the trace collection ("day 0"). External fingerprints of the remaining two pairs of curves were collected three days before the trace collection ("day -3") and 29 days after the trace collection ("day 29") respectively; (b) This figure complements Figure 4 (a) with external fingerprints collected 71 days after ("day 71") the trace collection.

the worst-case analysis to partial prefix preservation, with some surprising consequences.

4.1 Calculating worst-case de-anonymization

To formalize the worst case de-anonymization, we wish to find the largest set of all anonymized addresses that are K-vulnerable when attacked using trace fingerprints F. We refer to this set as $V^\star(K)$, and note that any set of K-vulnerable addresses obtainable by any de-anonymization technique using trace fingerprints F, (including the one presented by Coull et al. [19]) is a subset of $V^\star(K)$. We present a very efficient algorithm that computes $V^\star(K)$ given F.

Recall that $V(K, \Sigma, F, c)$ consists of all match sets with size less than K and which contain the correct de-anonymization. In what follows we use $V(K, \Sigma, F, c)$ as a metric and look for the worst-case attack for a given set of trace fingerprints F.

We introduce the following cost function

$$c^\star(\Sigma(x), F(y)) = \begin{cases} 0 & \text{if } \Sigma(x) = F(y), \\ \epsilon & \text{for } \epsilon > 0, \text{otherwise.} \end{cases} \quad (5)$$

We state the following theorem whose proof is found in [25].

Theorem 4.1. *$V(K, \Sigma, F, c) \subseteq V(K, F, F, c^\star)$ for any external fingerprints Σ and cost function c.*

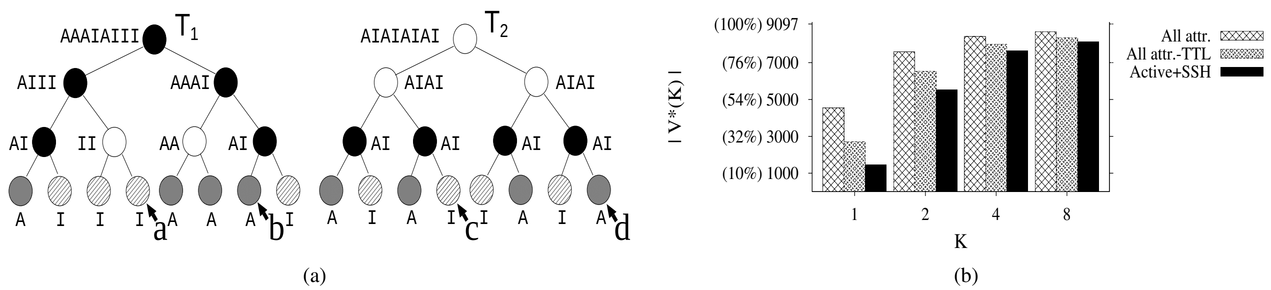
The calculation[4] of $V^\star(K) = V(K, F, F, c^\star)$, and thus the evaluation of worst-case de-anonymization, is even more efficient than the execution of the attack described in Section 2. Because fingerprints are assumed perfect, cost based tree-matching is not needed. In fact, the entire computation can be performed on the trace fingerprint tree so that it is possible to compute $V^\star(K)$ in linear time ($O(|\mathcal{A}|)$).

More specifically, [25] shows that the worst-case match sets for a host are determined by the number of "white" nodes in the fingerprint tree: the size of the match set for host y is $2^{W(y)}$ where $W(y)$ is the number of white nodes on the path from leaf y to the root of $T(F)$. As an example, Figure 5(a) shows trace fingerprint tree T_1 with two marked leaves a and b and trace fingerprint tree T_2 with two marked leaves c and d ($a, b, c,$ and d are the leaves pointed by an arrow). Note that here $T(F)$ is also the external fingerprint tree. Leaf a has one "white" vertex on its path to the root and thus $|\beta(\text{“}a\text{”}, c^\star)| = 2^1$. Leaf b has no "white" vertex on its path to the root and thus $|\beta(\text{“}b\text{”}, c^\star)| = 2^0$. Leaf c has two "white" vertices on its path to the root and thus $|\beta(\text{“}c\text{”}, c^\star)| = 2^2$.

4.2 Worst-case K-vulnerability under full prefix preservation

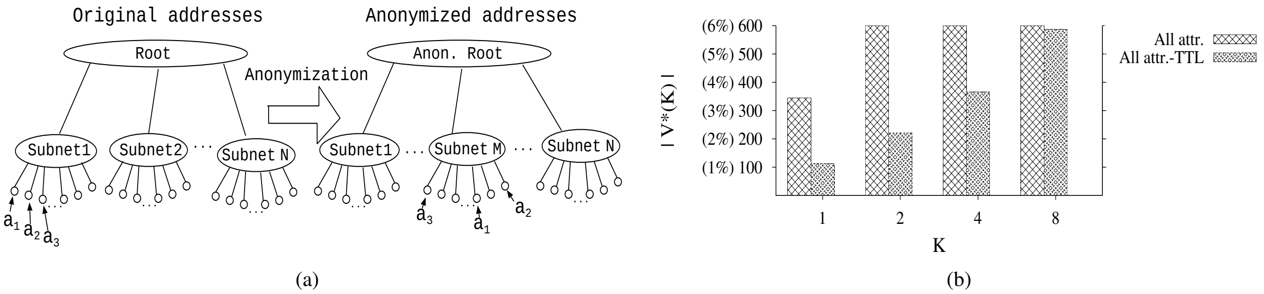
We can now apply the worst-case analysis to the trace studied experimentally in Section 3. Figure 5(b) shows the worst-case K-vulnerable hosts, $|V^\star(K)|$, for $K = 1, 2, 4, 8$ applied to *Trace-0618*. The bars labeled "All attr." represent all attributes seen in Section 3. This graph shows that a well informed adversary can do significantly more damage to the anonymization function: 44% of the addresses are 1-vulnerable (uniquely identifiable), compared with 17% using external fingerprints obtained by probing the network.

[4] A Java source code of an algorithm that computes $V^\star(K)$ is available for download at http://www-net.cs.umass.edu/~ribeiro/deanonymization/



(a) (b)

Figure 5. (a) Example of two fingerprint trees, T_1 and T_2, for two distinct prefix-preserved anonymized traces; (b) K-vulnerability of trace *Trace-0618* with respect to Section 3 fingerprint attributes.



(a) (b)

Figure 6. (a) Example of partial prefix preservation. The network administrator defines N subnets. Addresses sharing the same subnet in the original address space also share the same subnet after the anonymization; (b) K-vulnerability of trace *Trace-0618* using partial prefix preservation with respect to Section 3 fingerprint attributes.

Figure 5(b) also shows the worst case for other sets of attributes: "All attr. - TTL", which includes all attributes except TTL, and "Active + SSH", which includes only host activity and SSH traffic. It is clear from the graph that the TTL attribute only significantly increases the number of 1-vulnerable addresses. The "Active + SSH" bars show that other TCP port attributes besides "SSH" can significantly increase the precision of the de-anonymization. Figure 3(b) compares the "All attr." upper bound with the results obtained with external fingerprints from *External-0618*. We can see that the perfect fingerprints perform at least two times better than the probed external fingerprints.

5 Analysis of partial prefix preservation

Given the vulnerability of traces to fingerprinting attacks like ours and others noted in the literature, it is natural to consider sacrificing the utility of the trace to increase privacy. We refer to such techniques as *partial prefix preservation*, because some prefix relationships are not preserved, replaced instead by unconstrained randomization of addresses or parts of addresses.

Pang et al. proposed a version of partial prefix preservation for internal addresses in their enterprise trace anonymization scheme [18]. There the subnet and host portions of an address are treated independently. Each subnet is comprised of a group of addresses that share the same $32 - b$ most significant bits.[5] If two addresses share the first $32-b$ significant bits in the original trace then they also share the same first $32 - b$ significant bits in the anonymized trace. Within subnets sufixes are not preserved; instead addresses are randomized within the subnet. Figure 6(a) illustrates this method. In this figure we label three hosts a_1, a_2, and a_3. Note that a_1, a_2, and a_3 are just host labels, not the anonymized or unanonymized addresses of the hosts. Figure 6(a) shows both the original and the anonymized partial address-space trees. Note that a_1, a_2, and a_3 are consecutive addresses at the original address space. However, after the anonymization, the only address relation preserved is that a_1, a_2, and a_3 still belong to the same subnet. The subnet changed its address from "Subnet 1" to "Subnet M".

Worst-case analysis of partial prefix preservation While this technique seems sensible, Pang et al. do not provide an analysis justifying this choice of partial prefix preservation. Attacks against this technique have since been demonstrated [19]. We now apply our analysis tools to evaluate the impact of partial prefix preservation on our sample trace.

The calculation of $V^\star$ under partial prefix preservation requires a modest variation of the technique described in

[5] In [18] the value of b is not necessarily constant among subnets; for the sake of simplicity we consider a constant value of b for all subnets.

the last section (details can be found in [25]). The results of the analysis appear in Figure 6(b) which plots $|V^{\star}(K)|$ against $K = 1, 2, 4, 8$ for a subnet size of $b = 8$ bits. Once the change in the y-axis has been noted, Figure 6(b) and Figure 5(b) (full prefix preservation) can be compared. We see that partial prefix preservation is much safer than full prefix preservation, reducing the unique de-anonymizations from about 4000 (44%) to 345 hosts (just under 4%). Partial prefix preservation also achieves much smaller values of $|V^{\star}(K)|$ for $K = 2, 4, 8$.

Another interesting result comes from removing some attributes from the fingerprints. Figure 6(b) also plots the value of $|V^{\star}(K)|$ for attributes "All attr.-TTL". Note that unlike full prefix preservation, the TTL attribute also significantly increases the number of 2 and 4-vulnerable addresses. Fingerprint attributes "Active+SSH" are not shown in Figure 6(b) as $|V^{\star}(1)| = |V^{\star}(2)| = 0$ and $|V^{\star}(4)| = |V^{\star}(8)| = 7$. It is interesting to note that a fingerprint ("Active+SSH") which is highly efficient for full prefix preservation is highly inefficient for partial prefix preservation. There is a simple explanation for the bad performance of fingerprint "Active+SSH" in partial prefix preservation: in our trace we found that most addresses with SSH traffic were clustered within a small number of subnets.

We also found that the size of the subnet, defined by b, can be varied with somewhat predictable results. When $b = 4$ bits, $|V^{\star}(1)| = 1039$, or 11% of hosts. When b is increased to 11 bits, few prefixes are preserved and $|V^{\star}(1)| = 106$, or just 1%. Larger b improves host anonymity at the cost of host prefix information.

Trading-off privacy and utility

Overall, for our trace, we find that partial prefix preservation as described by Pang et al. has a significant positive impact on trace anonymity. At the same time, 345 uniquely disclosed hosts may still be a concern to some publishers. In addition, these unique de-anonymizations confirm and help to explain the attack result found by Coull et al [19] in which a small number of distinctive servers are re-identified despite partial prefix preservation.

Interestingly, we found that it was possible to identify randomized subnets with very high likelihood. The concept of K-vulnerability can be applied not only to anonymized addresses but also to anonymized subnets. Consider the example in Figure 6(a). An adversary may be able to uniquely map anonymized "Subnet M" into its unanonymized counterpart "Subnet 1". Note that this does not mean that an addresses inside anonymized "Subnet M" is also uniquely matched to an address inside unanonymized "Subnet 1". In the following experiment we use $b = 8$ bits and fingerprints "All attr.-TTL" obtained from *Trace-0618*. We choose fingerprint attributes "All attr.-TTL" instead of "All attr." following Pang et al. [18] which removes TTL values from the trace. This experiment shows that 96% of all subnets, that have more than one active host, are 1-vulnerable and the remaining 4% are 2-vulnerable. When $b = 4$ bits we find 17% of the subnets to be 2-vulnerable (of which 13% are 1-vulnerable). And when $b = 11$ bits all subnets are 1-vulnerable. These results suggest that as subnets get larger their corresponding fingerprint are more likely to be unique.

In the light of these results we propose a change to the subnet anonymization scheme of Pang et al. [18]. In [18] all prefix information from the subnet level up to the root is removed. We propose that for large enough values of b (such that most subnets 1-identifiable), one can increase trace utility (keeping the same upper bound anonymity level) by applying full prefix preservation from the subnet level up to the root.

Note finally that our upper bound also allows data publishers to reduce trace utility (e.g. removal of TCP port information for a given address) when it is absolutely necessary to ensure anonymity against a set of fingerprint attributes. This can be seen as an improvement over a previous recommendation [19] which advocates for the removal of TCP port numbers from all records in the trace. Our results also show that an attacker is able to achieve significantly more damage when address fingerprints contain TTL attributes. This result corroborates the notion in [18] that TTL attributes should not be published in the trace.

6 Conclusion and Future work

We have analyzed a novel attack on prefix-preserving trace anonymization that encompasses other proposed attacks, and can be adapted to partial prefix preservation and to alternative information sources. We demonstrated the effectiveness of this attack on a real trace, and measured experimentally the impact of the accuracy of information on the adversary's success. Past works describing attacks on trace anonymization have left trace publishers without methods to evaluate the risks of publication for their traces and with few mitigation techniques. Our analysis techniques allow trace publishers to compute an upper bound for the risk of host de-anonymization in the context of adversaries assumed capable of collecting a given class of external information. In the future we hope to use these techniques to formally evaluate partial prefix preservation alternatives which can maximize utility relative to a desired level of trace privacy. We would also like to consider the application of this kind of attack to the external hosts in the trace. The challenges are that for external addresses structural information is less informative, and that probing must be limited to some known set of popular destinations since exhaustive probing of internet addresses would be infeasible.

7 Acknowledgments

We would like to thank our anonymous reviewers for their insightful comments and our shepherd Niels Provos for helping us with the final version of this paper.

References

[1] "Gateway link measurements at umass amherst," "http://www-net.cs.umass.edu/dag/".

[2] "Enterprise tracing project," "http://www.icir.org/enterprise-tracing/".

[3] "The passive measurement and analysis project," "http://pma.nlanr.net/".

[4] "The skitter project," "http://www.caida.org/tools/measurement/skitter/".

[5] "The internet traffic archive," "http://ita.ee.lbl.gov/", Apr. 2000.

[6] "Network tools and traffic traces at university of napoli federico ii," "http://www.grid.unina.it/Traffic/".

[7] T. McGregor, H. Braun, and J. Brown, "The NLANR network analysis infrastructure," *IEEE Communications Magazine*, vol. 38, no. 5, pp. 122–128, May 2000.

[8] Y. Liu, X. Liu, L. Xiao, L. M. Ni, and X. Zhang, "Location-aware topology matching in P2P systems," *Proceding of the IEEE INFOCOM 2004*, vol. 4, pp. 2220–2230, 2004.

[9] M. Rajab, F. Monrose, and A. Terzis, "Fast and Evasive Attacks: Highlighting the challenges ahead," in *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID)*, Hamburg, Germany, Sept. 2006.

[10] "Tcpdpriv," "http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html".

[11] R. Pang and V. Paxson, "A High-Level Programming Environment for Packet Trace Anonymization and Transformation," in *Proceedings of the ACM SIGCOMM Conference*, August 2003.

[12] J. Xu, J. Fan, M. Ammar, and S. Moon, "Prefix-preserving IP address anonymization: Measurement-based security evaluation and a new cryptography-based scheme," in *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP'02)*, Paris, France, November 2002.

[13] R. Ramaswamy and T. Wolf, "High-speed prefix-preserving IP address anonymization for passive measurement systems," *IEEE/ACM Transactions on Networking*, vol. 15, no. 1, January 2007.

[14] K. Tai, "The tree-to-tree correction problem," *Journal of the Association for Computing Machinery (JACM)*, vol. 26, no. 3, pp. 422–433, 1979.

[15] P. Bille, "A survey on tree edit distance and related problems," *Theor. Comput. Sci.*, vol. 337, no. 1-3, pp. 217–239, 2005.

[16] 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*, September 2005.

[17] 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 (PET 05)*, May 2005.

[18] R. Pang, M. Allman, V. Paxson, and J. Lee, "The devil and packet trace anonymization," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 29–38, 2006.

[19] S. Coull, C. Wright, F. Monrose, M. Collins, and M. Reiter, "Playing devil's advocate: Inferring sensitive information from anonymized network traces," in *Proceedings of the Network and Distributed System Security Symposium*, 2007.

[20] D. Koukis, S. Antonatos, and K. Anagnostakis, "On The Privacy Risks of Publishing Anonymized IP Network Traces," in *Proceedings of the 10th IFIP Open Conference on Communications and Multimedia Security*, October 2006.

[21] S. Coull, M. Collins, C. Wright, F. Monrose, and M. Reiter, "On Web browsing privacy in anonymized NetFlows," in *Proceedings of the 16th USENIX Security Symposium*, August 2007, p. 339352.

[22] S. E. Coull, C. V. Wright, A. D. Keromytisz, F. Monrose, and M. K. Reiter, "Taming the devil: Techniques for evaluating anonymized network data," in *Proceedings of the 15th Network and Distributed Systems Security Symposium*, to appear, 2008.

[23] M. Z. et al., ""http://lcamtuf.coredump.cx/p0f.shtml"."

[24] M. Neuhaus and H. Bunke, "Automatic learning of cost functions for graph edit distance." *Information Sciences*, vol. 177, no. 1, pp. 239–247, 2007.

[25] B. Ribeiro, W. Chen, G. Miklau, and D. Towsley, "On the loss of privacy in packet trace anonymization," Department of Computer Science, University of Massachusetts at Amherst, Tech. Rep. TR–07, Jun. 2007.

[26] S. M. Bellovin, "A technique for counting NATed hosts," in *IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002, pp. 267–272.

APPENDIX

A Obtaining external information address fingerprints

In what follows we compile a small list of possible external fingerprint attributes.

- *TCP port attribute (Web, FTP, ...).* (Active probing) To test whether a address could have TCP activity on a given port one can simply try to use TCP to connect to that port.
- *Traffic volume.* Traffic volume can be divided into levels. For instance, all main network servers (main web servers, main e-mail servers and so on) may be inferred as the busiest ones for that type of service. The number of levels depends on the adversary's knowledge of network servers traffic loads.
- *Flow sizes.* Many file transfers over the Internet have very specific flow size signatures.
- *Packet timing.* Internet activities (such as browsing the Web) can have easily identifiable packet interdeparture signatures.
- *Network changes.* Networks normally evolve with time. Such evolution can be available to the adversary. Abnormal traffic patterns or scheduled changes to the network that are known to the adversary creates the possibility of a new fingerprint attribute. Here even a harmless 4pm weekly department-wide coffee-break can result in a fingerprint attribute.

B Sources of fingerprint attribute mismatches and their counter-measures

The following list is a short but representative compilation of the troubles that adversaries face:

- *Change in address fingerprints.* It is possible that some hosts/services are active (inactive) during the time of trace collection and inactive (active) when the adversary obtains address fingerprints. Under such circumstances it is likely that some address fingerprints appearing in the trace are distinct from their corresponding external information fingerprints. A good representative of this class are IP addresses allocated using the DHCP protocol (dynamic allocation).
- *Host or host service is active but had no traffic recorded.* Some hosts or host services may be active at the time of the trace collection but have no traffic recorded in the trace. This creates a fingerprint mismatch as the adversary sees a potential source of traffic that is not present in the trace.
- *Firewalls, IDS.* Firewalls and Intrusion Detection Systems (IDS) may limit the adversary's ability to determine address fingerprints or even masquerade them.
- *NAT boxes.* Many distinct hosts can share an unique IP address which may induce contradictory fingerprints attributes at the same IP address.

Attenuating fingerprint attribute mismatches. Each item of the aforementioned list can be attenuated if the adversary takes the following precautions:

- *Dynamic IP address allocation.* An adversary that has access to a trace that spans over a long time frame may use it to obtain which portions of the address space may have temporal fingerprint changes. Information on which portions of the address space are more "dynamic" can actually help the adversary if the adversary has this type of external information (e.g., which portions of the network use DHCP). Another way to use such fingerprint is to mark the main e-mail server of a network as "stationary" and the rest of the e-mail servers as potentially "dynamic".
- *Host or host service is active but had no traffic recorded.* A service or a host that sees has almost no recorded traffic in the trace is likely to .
- *Firewalls, IDS.* This is the hardest mismatch to cope with. Here the adversary cannot trust some fingerprint attributes and must resort to other types of external information, such as personal web-traffic preferences. For instance, Google.com flow sizes vary according to user language settings.
- *NAT boxes.* Might be detected in the trace [26]. If the adversary knows the true address of some NAT boxes, it can be used in the adversary's advantage as a new "true, false or undefined" fingerprint attribute: "is it a NAT box?".

C Sample attack

Let $T(F_t)$ and $T(F_e)$ denote the trace and external fingerprint trees depicted in Figures 2(b) and 2(a), respectively. In this example we use the edit cost function $c^\star$ (equation (5)). Let's follow the disclosure of address t_{15} of Figure 2(b). The algorithm works on the fingerprint trees in a deep-first-search-similar order, starting from the root of $T(F_e)$. It first decides which vertex of $T(F_t)$, t_2 or t_3, is the best match for vertex e_2 of $T(F_e)$. Here, inner vertex fingerprints can help us decide which match is the most likely one. The fingerprint of e_2 is `AAAI` and the fingerprints of t_2 and t_3 are `AAII` and `AAAI` respectively. The algorithm decides that t_3 is the most likely match for e_2. The match (e_2, t_3) forces the match (e_3, t_2) due to the prefix preservation of the anonymization. The above illustrates our main optimization over the brute force search. Since $f(e_2) = f(t_3)$, $f(e_3) = f(t_2)$, $f(e_2) \neq f(t_2)$, and $f(e_3) \neq f(t_3)$, the minimum cost of the match (e_2, t_3) is zero rather than the minimum cost of (e_2, t_2) which is 2ϵ (for simplicity, we use (e, t) to denote that vertex e is de-anonymized or *matched* to vertex t). If later in the process we believe for some reason that this decision was incorrect, we backtrack and follow the opposite match. But if the cost of matching the leaves of the (e_2, t_3) match is still zero (as predicted), then there is no need to verify the match (e_2, t_3). The next step is to decide if the best match for e_4 is t_6 or t_7. Again we use fingerprints to decide the most likely match: (e_4,t_6). Next we have to decide if e_8 matches t_{12} or t_{13}. In this case we could go either way. The reason we cannot decide is that both t_{12} and t_{13} have the same labels. In this case we say that e_8 is in the match set of t_{12} and t_{13}, or more formally: $e_8 \in \beta(t_{12}, F_e, F_t, c^\star)$ and $e_8 \in \beta(t_{13}, F_e, F_t, c^\star)$. In the next step we find that t_{15} is a good match to e_{10} (thus e_{11} is a good match to t_{14}). If we continue the algorithm we find that the edit distance is zero and that $\beta(t_{15}, F_e, F_t, c^\star) = \{e_{10}\}$. After finishing the match (e_2, t_3), we follow the match branch (e_3, t_2). As $f(e_4) = f(t_5)$ and $f(e_4) \neq f(t_4)$, we first follow the match (e_4, t_5), from which we conclude that $\beta(t_{10}, F_e, F_t, c^\star) = \beta(t_{11}, F_e, F_t, c^\star) = \{e_{14}, e_{15}\}$. Likewise, we also find $\beta(t_8, F_e, F_t, c^\star) = \beta(t_9, F_e, F_t, c^\star) = \{e_{12}, e_{13}\}$. Note that we find all matches that have added costs (eq. 1) zero. The edit distance is therefore zero. However, when external fingerprints are imperfect, the edit distance may not be zero. More details of this algorithm can be found in [25].