The Role of Network Trace Anonymization under Attack

Martin Burkhart
ETH Zurich
burkhart@tik.ee.ethz.ch

Dominik Schatzmann
ETH Zurich
schatzmann@tik.ee.ethz.ch

Brian Trammell
Hitachi Europe
brian.trammell@hitachi-eu.com

Elisa Boschi
Hitachi Europe
elisa.boschi@hitachi-eu.com

Bernhard Plattner
ETH Zurich
plattner@tik.ee.ethz.ch

ABSTRACT

In recent years, academic literature has analyzed many attacks on network trace anonymization techniques. These attacks usually correlate external information with anonymized data and successfully de-anonymize objects with distinctive signatures. However, analyses of these attacks still underestimate the real risk of publishing anonymized data, as the most powerful attack against anonymization is traffic injection. We demonstrate that performing live traffic injection attacks against anonymization on a backbone network is not difficult, and that potential countermeasures against these attacks, such as traffic aggregation, randomization or field generalization, are not particularly effective. We then discuss tradeoffs of the attacker and defender in the so-called injection attack space. An asymmetry in the attack space significantly increases the chance of a successful de-anonymization through lengthening the injected traffic pattern. This leads us to re-examine the role of network data anonymization. We recommend a unified approach to data sharing, which uses anonymization as a part of a technical, legal, and social approach to data protection in the research and operations communities.

Categories and Subject Descriptors

C.2.3 [**Computer-Communication Networks**]: Network Operations

General Terms

Measurement, Experimentation, Security

Keywords

Anonymization, Injection Attacks, Privacy

1. INTRODUCTION

Sharing network traffic data among organizations enables many important functions such as network measurement research and operational cooperation in incident handling, but is often limited by privacy and security concerns. Anonymization techniques and tools [21, 25, 14] are often cited as a remedy to these concerns. However, anonymization itself poses a risk-utility tradeoff.

This risk-utility tradeoff, stated simply, is that anonymization necessarily removes information from a data set. Information left in a data set to make it more useful for a given analysis purpose can also be used to break the anonymization used to protect it. The risk of breaking a given anonymization technique is often evaluated by inspecting anonymized data and correlating it with external information about the network [11, 10, 22, 17]. These analyses show that objects with a distinctive signature can in general easily be de-anonymized. This calls into question the applicability of anonymization as a general-purpose approach to privacy protection.

However, these studies still underestimate the real risk of publishing anonymized data. The most powerful attack against anonymization is traffic injection. In an injection attack, an attacker causes packets with known characteristics to traverse the network under measurement in a known sequence. Such attacks are analogous to known-plaintext attacks against a cryptosystem, and ease the attacker's job in breaking anonymization. Therefore, if they can be shown to be easy to mount and difficult to detect and defend against, then risk evaluations merely based on inspection of anonymized data are shown to underestimate the disclosure risk of publishing anonymized traces.

While injection attacks are often discussed as a worst-case scenario [6], and widely acknowledged to be tricky to defend against [13, 26, 20], their success rate in real networks and possible countermeasures have not been widely studied. This is largely a consequence of the way such research is performed. Researchers generally work on anonymized traces that are months if not years old. Given this arrangement, there are limited opportunities to evaluate injection attacks against anonymized data.

This work empirically studies the feasibility of live injection attacks and potential countermeasures on a medium-sized backbone network. We show that permutation-based techniques for IP address anonymization can be easily reversed by injection attacks, either by the injection of complex patterns over short time periods, or simpler "stealth" patterns over longer time periods. We define the injection attack space to discuss tradeoffs of the attacker and defender and find an interesting asymmetry: it is significantly more difficult to defend against injection attacks that stretch in time than attacks using greater pattern complexity. Countermeasures designed to reduce pattern recognition rates, such as aggregation, binning and randomization of secondary fields, are not particularly effective, and may not preserve sufficient utility for many analysis tasks in any event.

This study leads us to re-examine the role of anonymization as part of a comprehensive approach to data protection for data sharing purposes. Recent work has already found that "anonymize, publish, and forget" is potentially insufficient on its own as an approach to data protection for network trace sharing [18]; the ease of successful injection attacks strengthens this finding.

However, we emphasize that this is *not* to say that anonymization is useless, or should not be deployed. Instead, future work in network trace sharing should develop systems and processes that employ anonymization as part of a *comprehensive* approach to protection, alongside other technical means, as well as legal and social means.

2. REAL WORLD ATTACKER MODELS

We consider two general types of attacks against network data anonymization, inspection and injection.

Inspection attacks attempt to break anonymization using only information from the trace and information available from observation of the network and from other relevant sources (e.g., DNS and WHOIS databases) *after* the trace was collected. Note that inspection attacks are not completely passive; they may employ active network measurement techniques, e.g., scanning networks suspected to be the measured network in a given trace in order to locate services.

Inspection attacks are usually not *privileged*, that is, the attacker does not have any information not available to anyone else outside the operator of the measured network and data publisher. They are, however, possible against already-published data. Inspection attacks can be thought of as a superclass of all the attack classes enumerated in [16] except Data Injection.

Injection attacks exploit additional foreknowledge of traffic which will appear in a subsequently published trace, generally by generating traffic in the measured network with known characteristics. The attacker here is privileged, having information about the patterns injected into the trace that no other external observer does. This is analogous to a known-plaintext attack against a cryptosystem: by causing known traffic to be captured in the trace for subsequent anonymization and publication, the attacker has knowledge of some raw data within the trace. Note that while the operator of the measured network does have information about the injected patterns by virtue of having access to the raw data, they may not realize what portion of the traffic contains the injected patterns, or even indeed that patterns have been injected. Injection attacks do, of course, require knowledge about the network location and time period during which a trace will be collected and published *before the fact*, and the ability to cause network traffic to be generated on the measured network. Injection attacks are analogous to the Data Injection attack class in [16].

One-time publication of traces without prior announcement are more likely to be attacked by inspection alone, since successful injection would require either some measure of luck or a pervasive traffic injection infrastructure. But if traces are published on a regular basis from a given network, or publication is announced beforehand as in a coordinated collection activity such as [15], then potential injection attackers know the schedule of collection, and the publisher must assume that injection attacks are underway.

Note that long-term injection attacks do not pose significant storage requirements for the attacker. Deterministic or pseudorandom traffic sequences can be described completely by the generation algorithm and its parameters (e.g., a seed for a pseudorandom number generation). When stored with the start time of each injection activity, this enables the attacker to easily synchronize the trace and generation timeline. This information can be leveraged to reconstruct timing information removed from the traces, such as the date of the capture, or inter-flow timing as in timestamp enumeration.

3. RELATED WORK

In Table 1 we briefly summarize de-anonymization studies and note whether the analysis considers injection attacks or not, or have access to some non-public information before the attack. Most of the more recent work has focused on inspection attacks only. Of the studies that consider injection attacks, [20] does not empirically quantify involved risks and [13, 6, 5, 7] assume it is *somehow possible* to mount injection attacks but do not explore the success rate of actually recovering injected patterns in real traffic traces and potential countermeasures. In a slightly different context, [3] uses injection attacks to identify anonymous Internet sensors.

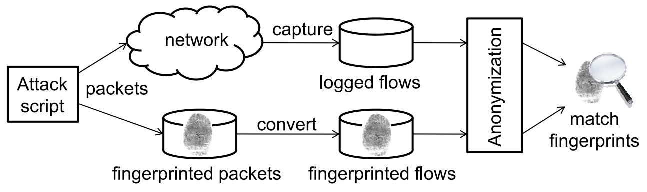

Figure 1: Setup of the injection attacks.

4. TRAFFIC INJECTION EXPERIMENTS

Traffic injection attacks are an instance of a covert channel problem and therefore inherently intractable. Almost any field in traffic data can be used to carry some bits of hidden information along with real data. An example of how timing fields of IP packets can be used as covert channels is given in [9].

The signal injected by an attacker can vary in terms of strength and duration. Signal strength refers to the number of bits used to encode the signal and how different the injected signal is from normal traffic. Of course, even a weak signal can be detected, given enough time. In this section, we empirically explore the success rate of these attacks by injecting artificial packet sequences into a network. We call the footprint these packets leave in flow traces *patterns*. The specific questions we try to answer are: Can sequences of packets injected in the attacker's home network, routed across the Internet to a target network, captured in the destination's backbone network, exported in NetFlow format together with all the backbone's traffic, and then after anonymization still be recognized by the attacker? If yes, what is the minimum pattern complexity (number of packets, data volume, specific ports) required? Could anonymization of secondary flow fields lower the pattern recognition rate or even mitigate these injection attacks? Are certain host classes (e.g., webservers) more easily identified by injection attacks than others?

4.1 Setup

Figure 1 gives an overview of our experimental setup. Using a traffic generation and injection framework developed for this study [23], we injected fingerprinted packets into the SWITCH network, a medium-sized backbone operator with an assigned address space of approximately 2.3 million IP addresses. We then collected these packets with an existing collection infrastructure, which produces unsampled and non-anonymized NetFlow records from the entire SWITCH border. We sent packets to 30 target hosts during working hours, as up to 140 million flows per hour were collected. Both captured and injected flows were anonymized according to a variety of policies using software integrated into this collection infrastructure, which we developed for this study. Then, given this anonymized traffic, we attempted to de-anonymize the target IP addresses by recovering the fingerprinted flows from the set of all SWITCH flows. As we controlled the set of injected traffic and target addresses, we have ground truth to compare to; we emphasize that we did *not* break SWITCH-provided anonymization.

4.2 Pattern complexity

In this section we explore the pattern complexity necessary to identify fingerprints later on. We injected each of the patterns P_1 to P_5 listed in Table 2 to 30 target hosts, selected randomly from webservers, public student workstations and darkspace addresses. Each of the 150 pattern-host pairs was injected in a distinct 5-

Reference	Inj.	Priv.	Summary
Fan [13]	(✓)	(✓)	Evaluates the robustness of prefix-preserving permutation wrt. the de-anonymization of a number of IP addresses in general. Their analysis is oblivious wrt. how addresses are de-anonymized and the success rate of doing it. Discusses active attacks as a problem.
Ribeiro [22]	-	-	Fingerprints hosts in the live network by scanning 9 TCP ports. Leverages a tree editing algorithm to optimally attack prefix-preserving schemes.
Pang [20]	(✓)	-	Does not empirically quantify de-anonymization risks, but removes scans from traces to mitigate injection attacks. Prefixes are only partially preserved. Acknowledges that defending against general injection attacks, not just scans, is a tough problem.
Brekne [6]	✓	✓	Assumes a very powerful adversary that knows the traffic distribution of the attacked network and performs successful injection attacks to de-anonymize prefix-preserving schemes.
Brekne [5]	✓	-	Shows theoretically that injection attacks can be used to break any *static* pseudonymization scheme. Suggests dynamic (not 1-to-1) mapping of addresses.
Coull [11]	-	-	Fingerprints heavy-hitter hosts by building behavioral profiles and infers network topology from MAC addresses (if available).
Coull [10]	-	✓	Identifies objects at high risk of being de-anonymized with fingerprinting attacks. Assumes an attacker that has complete knowledge of object distributions in original traces.
Koukis [17]	-	-	De-anonymizes web browsing traffic by fingerprinting webservers with characteristic HTTP object sizes. Instead of injecting scans, they propose to detect existing scans in traces.
Burkhart [7]	✓	-	Assumes an attacker capable of injection attacks, also not studying feasibility of these attacks. Estimates the remaining risks when IP address truncation instead of permutation is applied.

Table 1: Summary of de-anonymization studies and their attacker models. *Inj*ection: paper considers injection attacks. *Priv*ileged: attacker has information not available to the public *before* the attack (e.g., the distribution of all objects in the network).

	Pkts	Src P.	Dst P.	Delay [ms]	Packet sizes
P_1	1	Fixed	80	-	160
P_2	5	R(65k)	R(65k)	200	256
P_3	10	Fixed	80	200	480 [+32]
P_4	10	R(65k)	R(65k)	200	832 [+32]
P_5	50	R(65k)	R(65k)	150+R(300)	1208 [+R(8)]

Table 2: Injected Patterns. Values in square brackets denote the field evolution between packets. R(x): random number between 1 and x. Delay: inter-packet delay.

	IP addr.	Ports	Time [s]	Packets	Bytes
A_1	Perm.	-	-	-	-
A_2	Perm.	-	-	O(5)	O(50)
A_3	Perm.	B(8)	O(30)	-	-
A_4	Perm.	B(2)	O(60)	-	-
A_5	Perm.	B(8)	O(30)	O(5)	O(50)
A_6	Perm.	B(2)	O(120)	O(10)	O(200)

Table 3: Anonymization Policies. B(x): bucketized in x buckets, O(x): Added a uniform random offset between $-x$ and $+x$, minimum value for packets and bytes being 1.

minute window. While some of the related work removes scans from traces [20] to mitigate injection attacks or avoids scans to not be detected [17], these patterns aim at being stealthier than scans and thus harder to detect.

Because the attacker relies on secondary flow fields like ports, timing or packet count/size to identify his patterns, we then blurred these patterns by applying anonymization policies A_1 to A_6 (see Table 3) to the captured and fingerprinted traces. From A_1 to A_6, anonymization increases in strength as data utility decreases.

For each pair of injected and captured flows we computed a difference score considering the flow fields used in the patterns. This compensates for potential imprecision as a result of retransmissions or measurement artifacts. For each injected flow we selected the captured flow with the smallest difference score. We considered a pattern correctly recovered, if the majority of flows belonging to the pattern were attributed to the correct target host.

The average number of recovered pattern-host pairs per anonymization policy is shown in Fig. 2. We recovered all patterns, even those where P_1 consists of a single packet, from policies A_1 and A_2. Bucketing of ports and timestamp randomization in A_3 and A_4 reduce the success of pattern recovery, especially for the simpler patterns P_1 and P_2. However, patterns P_3 – P_5 are still very powerful even when ports are sorted into two buckets and timestamps are distorted by up to one minute. P_3 and P_5 even resist A_5 that anonymizes every field. Policy A_6, which severely distorts the data, did prevent the recovery of all tested patterns. However, one could define more complex patterns to defeat even A_6.

The success of a pattern is greatly determined by its distinctness drom the background traffic. Consider P_1 and P_3, both resulting in a single flow. The unique combination of 10 packets with 6,240 Bytes makes P_3 detectable even with A_5, while P_1 is lost with A_4. However, if no a priori knowledge about network traffic statistics is available, it is best to inject patterns with random values that result in multiple flows.

We found no difference in identifiability among host classes. This is due to the fact that a pattern has to stand out in all flows, not just in flows involving the target. Consequently, all hosts are vulnerable to injection attacks, whereas with inspection attacks only hosts with unique traffic characteristics (e.g., large webservers) are identifiable.

4.3 Flow Aggregation

In addition to IP address permutation, we also experimented with anonymizing flows by address masking, or truncating the least significant x bits of the IP addresses. The smallest identifiable entities in the aggregated traces were therefore subnets of size /$(32 - x)$. Although this effectively prevents the de-anonymization of individual hosts, the injected patterns were equally recoverable, since truncation leaves secondary flow fields untouched. However, if flows

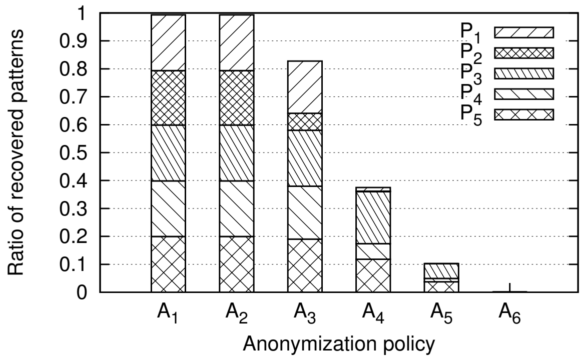

Figure 2: Average ratio of recovered patterns for each anonymization policy. There are 150 patterns in total, 30 for each P_i.

are instead *aggregated*, by truncating addresses and/or bucketizing other flow key fields, then collapsing each flow value field by performing the natural aggregation functions for that value field (e.g., summation for counters, set union for flags), the number of distinct flow signatures is significantly reduced. Borrowing from k-anonymity [27], all aggregated flows representing fewer than k flows are then deleted or aggregated in one huge flow to preserve overall traffic statistics. This significantly complicates the recognition of injected patterns, because the patterns are merged with at least k other flows.

We simulated the effects of flow aggregation with various anonymization policies. For instance, with bilateral port classification, deletion of seconds in timestamps, truncation of 16 (8) bits from external (internal) addresses, only 12.5% of the flows remained unique, (i.e., were not aggregated with at least one other flow). As expected, we could no longer recognize injected patterns in the aggregated flows.

However, in addition to the fact that flow signature reduction requires severe anonymization policies applicable only to certain specific analysis purposes, an attacker can adapt to aggregation if he is capable of arbitrarily choosing source IP addresses (e.g., via a large scale botnet). In particular, to evade aggregation of injected patterns, the attacker simply needs to find a unique source subnet from which to inject traffic. If no other traffic originates from this subnet, he can easily inject patterns resulting in more than k flows. These patterns would then be recognizable in their aggregated form.

4.4 Pattern duration

Instead of developing more complex patterns, an attacker could also use very simple patterns but inject them over longer time periods. Here we assume an attacker that is able to inject *only one bit of information per time window*, due either to strong anonymization or a desire for stealth. In particular, our worst-case attacker decides in each time window to either send traffic to the target host (1) or not (0). If IP addresses are permuted 1-to-1, this allows the construction of a sequence of 0s and 1s for each src/dst IP address pair. For instance, 1011... describes a pair that appears in the first, third and fourth window but not in the second. Given enough time windows, there will eventually be only one pair matching the injected on/off pattern, namely the one with the attacker as source and the target as destination, allowing de-anonymization of the target host. This method is essentially independent of the anonymization technique. In fact, secondary flow fields could all be overwritten with random values.

Fig. 3 shows an analysis of the probability that a random on/off pattern of a certain length creates a unique sequence in a network

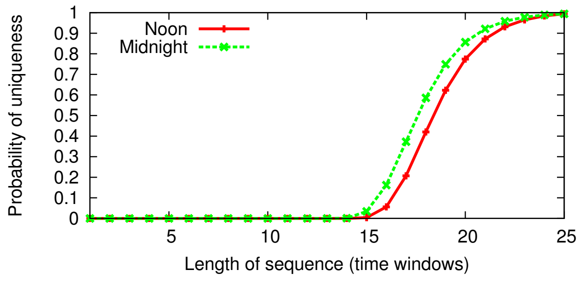

Figure 3: Probability of a random traffic presence sequence of a given length to be unique in the trace.

trace collected at noon and at midnight of a CET (UTC+1) workday. At noon, 80% of the 20-bit sequences and 99% of the 25-bit sequences uniquely identify the fingerprinted src-dst pair. Even though the traffic volume is much higher at noon than at midnight, and therefore the number of possible candidates at noon (4.1 M per 5 min) is significantly higher than at night (2.5 M per 5 min), the average number of windows required to be unique is only increased by one. This is due to the fact that already an increase of one time window doubles the number of existing on/off patterns and therefore reduces the chance of a collision by a factor of 2. This exponential increase of the search space by a linear increase of the time allows scaling the attack to networks of arbitrary size. Depending on the anonymization of timestamps, this allows quick identification of targets. For instance, if A_5 is applied, a window size of 1 min. is sufficient to separate patterns and allows identification within 25 min. Of course, several of these sequences can be injected in parallel.

5. INJECTION ATTACK SPACE

In [24] the authors show that defending against injection attacks (with the goal of identifying alert submitters in collaborative intrusion detection) inevitably involves the suppression of rare or low-density attacks. This is basically what we did with A_3 to A_6. We suppressed small signals by generalizing and randomizing the traces, hence success rate dropped for simple (i.e., low-volume) patterns. Packet sampling has a similar effect. After packet sampling, only events of a certain size are detectable, be it injected patterns or genuine traffic properties. Sampling does not effectively counter injection attacks, but it increases the required volume/time for attacks to work.

Technical defenses against injection attacks are limited by the decision problem of whether a given packet is legitimate or belongs to injected patterns. It gets even worse: A packet could be *both legitimate and injected*. For instance, an attacker could inject patterns using ordinary HTTP GET requests to de-anonymize a web server.

To discuss attacker and defender tradeoffs, we outline a conceptual injection attack space in Fig. 4. The x-axis denotes the duration of the attacks and the y-axes the pattern complexity, i.e., how much is the injected pattern different from ordinary traffic. Towards the top right corner of the plot, the complexity and duration of injected traffic constantly increases until a significant part of the collected trace consists of injected traffic. The plot shows three areas:

Area 1: The injected pattern is not recovered by the attacker, either because the pattern does not stand out from normal traffic or because the deployed anonymization policy blurred the pattern. This is, for instance, the case for pattern P_2 with policy A_4 in Fig. 2.

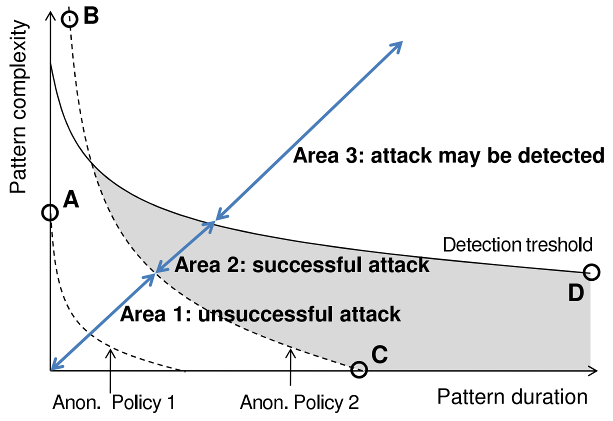

Figure 4: Injection attack space.

Area 2: The injected pattern is successfully recovered by the attacker, for instance P_4 with policy A_3. In addition, the network operator is unaware of this attack. This denotes a successful injection attack.

Area 3: The injected pattern is recovered by the attacker, but also the operator becomes aware of the attack. The operator might run intrusion detection systems capable of detecting abnormal traffic (e.g., packet sequences not consistent with application-layer protocol state). Also, if the injected patterns pass a certain volume, the attack becomes visible in network wide statistics. For instance, [17] refrain from active scanning in order not to be detected. Instead, they use existing scans in the trace.

Of course, the goal of the attacker is to operate in area 2 where he recovers the injected patterns but is not discovered himself. As this is a conceptual diagram, an empirical determination of the exact dimensions of this area would require an exhaustive study of varying pattern complexities and anonymization techniques. This is a suggested area for future work. However, we can identify two approaches to minimize the size of this area by restricting the number space from which patterns can be generated. Bogon filters [12] limit the size of the source IP address space, and have other security benefits as well. Aggressive ingress filtering based upon known services limits the size of the destination IP address space as well as the destination port space, but may interfere with applications using port negotiation or other NAT traversal techniques. Measurement of the impact they have on injection attacks is an area for future work.

5.1 Attack Space Asymmetry

Although the specific shape of the injection attack space is hard to determine and depends on the target network, some vertices can be fixed as shown in Fig. 4, leading to an asymmetry between pattern complexity and duration.

First, with weak or no anonymization, it is possible to easily recover injected patterns, even for very short durations (point **A**). For instance, all patterns where recovered for A_2. Secondly, the stronger anonymization is, the harder it is to recover a pattern (see Sec. 4.2). Thus, *anonymization can be used to raise the bar for attackers*. Anonymization can even be made strong enough to prevent pattern recognition by complexity alone. This corresponds to point **B**. Because the line of policy 2 never intersects the y-axis, it is not possible for the attacker to get into area 2 for small pattern durations. In our experiments, this corresponds to A_6, which makes the detection of all injected patterns impossible. Of course, rigorous anonymization policies have the downside of also reducing data utility significantly. Finally, it is always possible to de-anonymize hosts given enough time. This is even achieved by patterns with minimum complexity of 1 bit per time window, as shown in Section 4.4. This corresponds to point **C**.

Points **B** and **C** together suggest that *it is significantly harder to defend against longer-duration attacks*. The attacker only needs constant storage to stealthily inject a sequence of random patterns over time. At the same time, the defender is looking for a needle in a (continuously growing) haystack. Therefore, the attack detection threshold line never touches the x-axis (point **D**). The conclusion is that *attackers can always reach area 2*, specifically by stretching patterns over time.

6. THE ROLE OF ANONYMIZATION

The difficulty of defending against injection attacks leads us to call into question the role of anonymization as a complete solution to the problem of data protection. First, we note that there is no general solution to the risk-utility tradeoff posed by anonymization; it must be considered within the context of the analysis to be done on the data, which information needs to be protected, and which information is required to be present for a given analysis task. A singular, purely technical solution to the problem is not forthcoming; technical aspects of a solution may draw from several techniques.

Analyses may be designed to take advantage of the fact that certain analytically useful techniques (e.g., flow aggregation) have anonymizing effects. Anonymization techniques may be applied in concert with these analysis design techniques to improve data protection. Query-based systems [18] may prove beneficial for network research, in which questions and answers are exchanged while data repositories are protected and immobile, but require further development in order to provide comprehensive ways to safely query a protected database, as well as an analysis of new attack models to which such systems may be vulnerable.

Anonymization must also be considered within its legal context. When anonymization is done for privacy reasons (which is the case, e.g., in data sharing), the data to be protected is specified by data protection law. Laws change depending on the jurisdiction, as does the definition of what "personal data" is, and how it must be protected. Discussions held at a panel on Legal Requirements and Issues in Network Traffic Data Protection [4] among U.S., European, and Japanese lawyers found that the regulations are heterogeneous across jurisdictions, and more work is needed to find appropriate interjurisdictional solutions for data sharing.

Specifically, European law [1] defines personal data as *any data that identifies a person either directly or indirectly* (i.e., through the use of additional information in possession of third parties). To this category belong, e.g., IP addresses and user profiles. The law restricts the processing allowed on this data and mandates anonymization for subsequent storage or before further processing (e.g., research). Note that anonymized data as defined by the law may identify persons *neither directly nor indirectly*. Therefore, current anonymization techniques applied alone do not necessarily provide "anonymization" in the legal sense.

While U.S. law is less restrictive than European with respect to data protection, there is the same disconnect between anonymization in practice and anonymization in the legal sense [8]. The real impediment in this case is in the publication of collected data, not the protection of identifying information it contains. The Stored Communications Act (18 U.S.C. §2702(a)(3)) is the primary legal obstacle to data sharing; [8] advocates primarily architectural changes in data collection and export to address it.

Legal, social, and technical means must be used together to achieve the aims of better data sharing for research and operations. Design of network data analyses must take into consideration the potential privacy impact and legal implications of the data used at each stage of the analysis. We reiterate many of the points made in [2]: threat models are situation dependent, and a clear Acceptable Use Policy is essential, as is deeper interaction between data providers and researchers. In any case, legal solutions are required to enforce those policies, apart from guaranteeing compliance to relevant laws. Here, anonymization protects against accidental disclosure, assists in compliance, and provides evidence of good faith.

7. CONCLUSION

In this work, we have summarized the landscape of attacks against data anonymization, an important tool in enabling data sharing for research and cooperative network operations purposes. We have found that more research has been done on *inspection* attacks, in which only information from the trace and information available from observation of the network and from other relevant sources are used in breaking anonymization, than on *injection* attacks, which use privileged knowledge of traffic patterns within a trace as well.

From this realization, we performed experiments demonstrating the ease of recovery of injected traffic patterns and the difficulty to defend against injection attacks. These experiments led us to define an injection attack space which has an important asymmetry: it is relatively easy for an attacker to increase the chance of a successful pattern recovery by simply lengthening the pattern, but no easier for a defender to detect a longer pseudorandom injected pattern.

The implications of this are similar to much other recent work in anonymization. The inherent difficulty to defend against these attacks implies that the role of anonymization needs to be re-examined, as anonymization alone cannot protect against determined attempts to break it. Anonymization cannot be considered without respect to the analytic purposes to which anonymized data will be put, especially if the analysis has an anonymizing effect itself. Legal issues in data sharing [1, 8, 4] must be addressed, and the entire legal and social context in which data is analyzed [19] must be considered. The application of anonymization techniques alone does not solve these issues.

Anonymization, then, should be applied as part of a legal, social, and technical approach to protection in network trace sharing. Anonymization itself remains useful as evidence of good faith, for protection of locally stored data against accidental disclosure of identifying information, and in raising the bar for potential attacks against such an integrated approach.

Acknowledgments

We are grateful to SWITCH for providing the traffic traces used in this study.

8. REFERENCES

[1] Directive 95/46/EC of the European Parliament and of the Council. OJ L 281, 23.11.1995, p. 31, October 1995.

[2] M. Allman and V. Paxson. Issues and etiquette concerning use of shared measurement data. In *ACM SIGCOMM conference on Internet measurement (IMC)*, 2007.

[3] J. Bethencourt, J. Franklin, and M. Vernon. Mapping internet sensors with probe response attacks. In *USENIX Security Symposium*, 2005.

[4] E. Boschi. Legal requirements and issues in network traffic data protection. In *ACM Workshop on Network Data Anonymization (NDA)*, 2008.

[5] T. Brekne and A. Årnes. Circumventing IP-address pseudonymization. In *IASTED International Conference on Communications and Computer Networks*, 2005.

[6] T. Brekne, A. Årnes, and A. Øslebø. Anonymization of IP traffic data: Attacks on two prefix-preserving anonymization schemes and some proposed remedies. In *Workshop on Privacy Enhancing Technologies*, 2005.

[7] M. Burkhart, D. Brauckhoff, M. May, and E. Boschi. The Risk-Utility Tradeoff for IP Address Truncation. In *ACM Workshop on Network Data Anonymization (NDA)*, 2008.

[8] A. Burstein. An Uneasy Relationship: Cyber Security Information Sharing, Communications Privacy, and the Boundaries of the Firm. In *Workshop on the Economics of Information Security (WEIS)*, 2007.

[9] S. Cabuk, C. E. Brodley, and C. Shields. IP covert timing channels: design and detection. In *ACM conference on Computer and communications security (CCS)*, 2004.

[10] S. Coull, C. Wright, A. Keromytis, F. Monrose, and M. Reiter. Taming the devil: Techniques for evaluating anonymized network data. In *Network and Distributed System Security Symposium (NDSS)*, 2008.

[11] S. Coull, C. Wright, F. Monrose, M. Collins, and M.K.Reiter. Playing devil's advocate: Inferring sensitive information from anonymized network traces. In *Network and Distributed System Security Symposium (NDSS)*, 2007.

[12] D. Dietrich. Bogons and bogon filtering. In *33rd meeting of the North American Network Operator's Group (NANOG 33)*, Feb. 2005.

[13] J. Fan, J. Xu, M. H. Ammar, and S. B. Moon. Prefix-preserving IP address anonymization. *Comput. Networks*, 46(2):253–272, 2004.

[14] M. Foukarakis, D. Antoniades, S. Antonatos, and E. Markatos. Flexible and High-Performance Anonymization of NetFlow Records using Anontool. In *SECURECOMM Conference*, 2007.

[15] kc claffy. A Day in the Life of the Internet: Proposed community-wide experiment. *ACM SIGCOMM Computer Communications Review*, 36(5):39–40, Oct. 2006.

[16] J. King, K. Lakkaraju, and A. Slagell. A taxonomy and adversarial model for attacks against network log anonymization. In *ACM symposium on Applied Computing (SAC)*, 2009.

[17] D. Koukis, S. Antonatos, and K. G. Anagnostakis. On the privacy risks of publishing anonymized IP network traces. In *Communications and Multimedia Security*, 2006.

[18] J. Mirkovic. Privacy-safe network trace sharing via secure queries. In *ACM Workshop on Network Data Anonymization (NDA)*, 2008.

[19] P. Ohm. The rise and fall of invasive ISP surveillance. *University of Illinois Law Review*, 2009(5).

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

[21] R. Pang and V. Paxson. A high-level programming environment for packet trace anonymization and transformation. In *ACM SIGCOMM*, 2003.

[22] B. Ribeiro, W. Chen, G. Miklau, and D. Towsley. Analyzing privacy in enterprise packet trace anonymization. In *Network and Distributed System Security Symposium (NDSS)*, 2008.

[23] D. Sauter. Invasion of Privacy Using Fingerprinting Attacks. Master Thesis MA-2008-22, ETH Zurich, 2009.

[24] V. Shmatikov and M.-H. Wang. Security against probe-response attacks in collaborative intrusion detection. In *Workshop on Large scale attack defense (LSAD)*, 2007.

[25] A. Slagell, K. Lakkaraju, and K. Luo. FLAIM: A Multi-level Anonymization Framework for Computer and Network Logs. In *USENIX Large Installation System Administration Conference (LISA)*, 2006.

[26] A. Slagell and W. Yurcik. Sharing computer network logs for security and privacy: A motivation for new methodologies of anonymization. In *Workshop on the Value of Security through Collaboration (SECOVAL)*, 2005.

[27] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.