Sharing Computer Network Logs for Security and Privacy: A Motivation for New Methodologies of Anonymization

Adam Slagell William Yurcik

National Center for Supercomputing Applications (NCSA)
University of Illinois at Urbana-Champaign
{*slagell,byurcik*}*@ncsa.uiuc.edu*

***Abstract*— Logs are one of the most fundamental resources to any security professional. It is widely recognized by the government and industry that it is both beneficial and desirable to share logs for the purpose of security research. However, the sharing is not happening or not to the degree or magnitude that is desired. Organizations are reluctant to share logs because of the risk from exposing sensitive information to potential attackers. In this paper we survey current attempts at sharing logs and current log anonymization tools. We further define the problem and describe a roadmap to solve the issues that have to date inhibited large scale log sharing.**

I. Introduction

Log data is an essential resource to security teams at any organization large enough to hire full-time IT personnel. While IDSs can operate directly upon streaming network data, matching signatures and producing alerts, it is still necessary for human beings to examine logs to understand these alerts. Logs also form the core source of evidence for computer forensic investigations following security incidents. In addition to these very applied uses, logs are important to security researchers, for instance, those involved with research honeynets.

It is typical in the current security culture for each autonomous organization to use log data only to locally optimize network management and security protection. For example, when an organization notices an Internet attack (e.g., an external reconnaissance scan) a typical reaction is to block the offending IP addresses at the organization's perimeter but not to alert others—even administrators of the offending network—about this activity. Another example of this type of behavior is reactively scanning for a vulnerability on all the organization's systems after noticing vulnerabilities being exploited, but not alerting others of this activity. Although these examples are not universally true, as some security engineers have trusted channels for sharing security events, such sharing is the exception. There is a culture of pushing attackers away from oneself without any consideration of the poor overall security resulting from this lack of coordination between organizations. It is analogous to local policeman chasing criminals from one jurisdiction to another without crossing jurisdictional boundaries.

In these ways, administrators may miss the bigger picture and are unlikely to notice when they are just a piece of a larger target. Indeed, there are very few cross-sectional views of the Internet, and until recently there have been no mechanisms to enable such wider views. Furthermore, current examples of wide views, such as spam blacklists and worm signatures, are often focused on a specific characteristic even though signatures are gathered from events across the entire Internet.

While security professionals are having problems sharing logs, sharing data is in fact quite common among attackers. They trade zombies, publicly post information on vulnerable systems/networks and coordinate attacks. Recent events at several U.S. supercomputing centers [7] have demonstrated examples of coordinated attacks against organizations that do not have good mechanisms in place for data sharing and log correlation. These problems highlight why it is no longer satisfactory to focus solely on the local picture; there is a need to look globally across the Internet. And while the data needed exists, tapping into thousands of data sources effectively and sharing critical information—intelligently and to the data owners' satisfaction—is an open problem.

The importance of solving this problem has even caught the government's attention as the Department of Homeland Security has recognized the importance of sharing information and established Information Sharing and Analysis Centers (ISACs) to facilitate the storage and sharing of information about security threats [11]. The importance of log sharing has also gained industry recognition with investments in infrastructure dedicated solely for this purpose across multiple industry sectors [15]. The National Strategy to Secure Cyberspace (NSSC) explicitly lists sharing as one of its highest priorities—data sharing within the government, within industry sectors and between the government and industry. In fact, of the eight action items reached in the NSSC report, three of them are directly related to log data sharing: Item 2: "Provide for the development of tactical and strategic analysis of cyber attacks and vulnerability assessments"; Item 3: "Encourage the development of a private sector capability to share a synoptic view of the health of cyberspace"; and Item 8: "Improve and enhance public/private information sharing involving cyber-attacks, threats, and vulnerabilities".

However, the reluctance to share logs—which has resulted in fewer entities sharing—is understandable due to the sensitivity of the information contained within logs. Logs may be accidently mishandled by friendly peers, or fall directly/indirectly into the hands of malicious crackers. Security engineers have enough concerns without worrying about being a contributor to a security compromise of their own organization, or even worse, creating third party liability to a security compromise of another organization.

The first phase of any digital attack is reconnaissance, and

logs are like a treasure map for would-be attackers. Access to computer and network logs can provide intruders with special views of a network not visible from the outside, even with scanning tools. Information gleaned from these logs could indicate potential bottlenecks for DoS attacks or could even contain passwords—as often happens when a user accidentally types a password into the username field. Logs can even be used to identify machines infected by worms, which are most likely not well-maintained or monitored. Soft targets such as these can be used to get a foothold into a network.

While some would claim that the difficulties in sharing logs is social, we believe the problems are technical at the heart. To share logs and address these concerns about the sensitivity of the information within them, anonymization techniques must be employed. However, the field of log anonymization is still immature. There are very few tools, and the ones that exist are deficient in many ways. Further, current anonymization tools are one-size-fits-all, or better put one-size-tries-to-fit-all. Ideally, they would support multiple levels of anonymization that trade-off between security—of the anonymization scheme—and utility—the amount of useful information retained in anonymized logs. Thus far, no log anonymizers—besides a tool we recently developed for NetFlow anonymization [21]—support multiple levels of anonymization, even though there are typically multiple levels of trust between parties who might wish to share logs and those with whom they would share their logs.

We believe a new anonymization framework must be created that supports trade-offs between security and utility by providing multiple levels of anonymization (This trade-off is illustrated in Figure 1). To achieve this goal two important research problems must be solved. First, a metric must be created for the utility of a log that is based on the fields or data types within that log. Utility should be measured by the types of attacks that can be detected with a log or set of logs. Second, a metric for the security of an anonymization scheme must be developed. This metric should be based upon the types of attacks that can be used against an anonymization scheme—a log and the anonymization algorithms used on it. The difficulty of the attack and the amount of information that can be obtained by reversing the anonymization will both affect this metric.

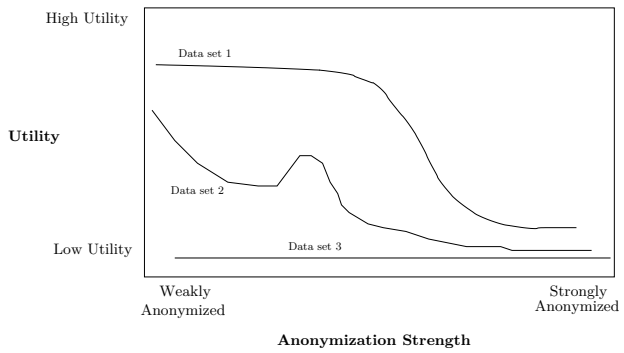

Fig. 1. Different datasets, i.e. sets of logs, will have different trade-offs between security and utility. Common to all data sets is that the utility is always highest with no anonymization, and over-anonymization—though it provides extremely high security because it is difficult to reverse the anonymization—can reduce the utility to unacceptable levels.

This new framework should be based upon the data types and fields within logs rather than the specific log types and versions. The prototype tools we are developing handle several log formats, but more importantly a framework that is extensible to handle almost any log is needed. Additionally, guidelines and standards must be developed to help organizations map trust levels to anonymization levels. In this way, organizations will be able to customize anonymization to fit their needs for the first time.

The rest of this paper is organized as follows. We further motivate the problem in Section II. In Section III, we discuss current attempts at large scale log sharing and current anonymization tools. Section IV describes current attacks against anonymization schemes, while Section V sketches a roadmap to achieving widespread sharing of logs. Lastly, we close with a summary and conclusions in Section VI.

II. Motivation

A. Why Logs Should be Shared?

There is both industry and academic interest in developing tools for computer forensics and log analysis [18], [16]. However, the development of such tools requires access to logs—the source of evidence in digital forensics and a fundamental resource to system administrators. While worms that are let go without further human intervention can be usefully modeled and simulated, human motives and complex multi-phase attacks cannot. Thus real, not simulated, data is needed to test and develop these tools. Furthermore, diverse data that cannot be gathered completely in-house is desired since a more diverse data set can be used to create more refined tools. This diversity comes in both the types of logs and in the environments from which they are drawn as different organizations have drastically different network traffic, usage patterns and threats to consider. To realistically gather such diverse data, logs must be shared.

In addition to those who develop tools that utilize logs as a data source, educators require diverse sets of logs. Professors and creators of educational materials need logs to effectively teach digital forensics in computer security courses. *Real* data is desired for insightful and meaningful student projects. Where simulated data—such as the type generated by the network simulator NS2—may suffice for projects in networking courses, it will not for security. In addition to college courses, security and system administrators receive training from institutions like SANS. Many SANS courses *require* students to bring sample logs to class. While this may be easy enough for an established security administrator to obtain, it is difficult for one who is getting training with the hope of moving to such a position to acquire sensitive logging information. Thus, those hoping to become system or security administrators and students in college courses depend upon others to share logs with them.

Security operators and researchers both need to share log information for additional reasons. Security administrators' views are usually focused locally, only on that which occurs on their own networks. However, we are seeing a rise in coordinated attacks across multiple organizations (e.g., recent attacks against several supercomputing facilities [7]). Addressing such attacks requires cooperation and data sharing between

the targeted organizations, something that we have struggled with firsthand at the NCSA. Without such sharing, the attackers have an advantage over the white hats. Black hats share information regularly (e.g., new vulnerabilities, exploits, information on vulnerable machines—even selling/trading zombies for DDoS attacks), and similarly, security teams must share information to effectively combat new threats.

In the future, log repositories could be setup to help victims detect broad, coordinated attacks. In fact, log repositories are already being setup for several reasons. They can be used to detect trends, determine the half-life of vulnerabilities, detect new attacks (such as the Honeynet Project has done [10]), and perform network measurement studies [19]. While this is progress, all of the current efforts at creating log repositories fall short of the desired solution from one or more of the following problems: (1) They do not have a wide view of the Internet but are quite localized, (2) the repositories are very specific, addressing one or only a few types of logs, (3) anonymization techniques used are weak—often nonexistent—and usually done inconsistently between submissions or (4) they collect many logs but do not share them with the research community.

B. Better Solutions to Share Logs are Needed

While it is understood and well accepted—by government [11], [15], academia and industry [24]—that log sharing is important, it currently happens on a very limited scale if at all. There is great reluctance because of the sensitive nature of information contained within logs. The first step of perpetrating any digital crime is reconnaissance, and logs are like a treasure map for would-be attackers since logs provide a special, internal view of the network. In the best case scenario, access to logs makes the reconnaissance easier and more stealthy. In the worst case, a cracker with private logs gains special information about the internal network and systems for which it would be impossible to scan from outside the network. Access to such privileged information can have devastating results for security administrators of organizations being attacked.

For example, logs can be used to determine weak points in the network, such as critical authentication servers (e.g., Kerberos or Radius) that are not replicated. Similarly, logs can indicate potential bottlenecks in a network, and such information can be used to create more effective DoS attacks. Of course, active services and their versions can be identified from logs, some of which may be internal and not normally visible to outsiders. Crackers will often make use of such information to find exploits for servers that are not fully patched. One with access to the logs may discover that there is little protection between a wireless and wired network within an organization. If that is the case and the attacker can drive within range of the network, they can gain access more easily through this alternative route. Logs can even help identify machines infected with worms or other malware. Such machines are not usually well-maintained and become soft targets that crackers can use as an entry point into a network. Sometimes sensitive information is more difficult to identify as many fields are sensitive only in special circumstances or when combined with other data. For example, while system logs do not generally record passwords, they may record incorrect passwords (which are often just a character off from the correct one). Quite commonly, invalid usernames are recorded. If this happens as a result of someone typing a password into the username field, a password will be recorded into a log—likely followed by an entry with the corresponding username.

While a draconian vetting process for would-be log recipients and strict physical control measures over the location of log data could be used to address some of these concerns about sharing sensitive logs, anonymization techniques provide an eminently more flexible solution. (Note that privacy preserving data mining techniques could also address the problem when access to raw logs is not needed. This is the case when the data owner controls access through a specialized query interface that just responds with statistics or small sample results.) However, the field of log anonymization is still immature. There are very few tools, and the ones that exist are deficient in many ways. Just a small subset of log types have been considered. Further, current tools typically anonymize only one field and work with only one type of log. In fact, all of the tools we have seen focus solely upon IP addresses or usernames. These are not the only sensitive fields, and even when these are the only ones that directly identify users and machines, other fields can be used in context to reverse anonymization algorithms. Thus time-stamps, port numbers and even obscure TCP flags can become sensitive fields that must be addressed by anonymization engines.

We must move beyond the one-size-fits-all anonymization tools that have been developed to date and create more flexible solutions. Besides the NetFlow anonymizer we present in [21], we know of no other anonymization tools that support multiple levels of anonymization, even though there are typically multiple levels of trust between parties who might wish to share logs and those with whom they would share their logs. For example, here at the NCSA a partial hierarchy of trust for log sharing would put our security operations team at the top, full-time employees below that, NCSA student researchers and professors from other departments under them, students from other departments even lower, and those outside the University of Illinois at the lowest level of trust. This example is in fact a simplification since often trust levels cannot be linearly ordered. This fact makes creating metric for log utility and anonymization strength more challenging, and it means that it may not be possible to make anonymization levels linearly ordered by anonymization strength. Instead, we may see a partially ordered hierarchy of anonymization levels when standards for anonymization are created.

III. Current Efforts of Log Sharing and Anonymization

A. Current Internet Log and Data Collection Centers

While there are a growing number of data centers that collect and analyze logs, few are dedicated to sharing logs with the research community. Those that do share, do not anonymize sufficiently, and they tend to focus on only one particular type of log. The goal is to share as many types of logs from as many sources as possible. Furthermore, there needs to be adequate protection of these logs to suit the different needs of those who

share them. To accomplish this goal, there must be multiple, standardized levels of anonymization.

1) CAIDA: CAIDA [5] has one of the largest Internet log data catalogs available for public analysis. They have an index to many data sets stored off site and are developing a cataloging system for these data sets. They also seek to develop tools to analyze the data sets. CAIDA focuses on macro-level data to support network measurement research. As such, they do not have the level of detail or the many heterogeneous types of logs necessary for security research. The usefulness of these logs to security research is limited to a very high level. The network traces that CAIDA provides could be used to get a big picture of worm activity, but they do not contain the level of detail to capture new exploit binaries or hacker behaviors post-compromise.

Even though these logs are not primarily intended for security research, they still suffer from the problems of undefined anonymization standards. While it is important to anonymize logs so that adversaries cannot map out contributor networks, CAIDA itself does not anonymize logs that it does not generate. Different organizations anonymize to different levels—some not anonymizing at all—and in different manners. There is no consistency in the way anonymization is done because all of the contributors do it their own way. Thus we see that the development of anonymization standards applies to more than just security research.

2) DeepSight: Symantec's DeepSight [23] does utilize a cross-sectional view of the Internet and analyzes detailed IDS, firewall and virus scanner logs. However, it is not a log sharing system for research, but rather it is a data collection system with a commercial purpose. Anonymization is not a real issue because data is not shared, but rather all data is collected and sent to a trusted third party. The purpose of this collection is to notice trends and provide early warnings of attacks and new threats to customers. They allow certain thresholds and other variables to be set by customers to somewhat customize their alerts and reports, but mostly it is a way to alert their customers of new worms, viruses or software exploits being used in the wild. It does not provide data sets to share with the research community.

3) Internet Storm Center: The Internet Storm Center (ISC) [12] is like a grass-roots version of DeepSight that is run by SANS. They collect IDS logs from volunteers and analyze them to detect trends. Their purpose is to provide an early warning system of new worm activity on the Internet. They provide reports on the top ports being scanned with respect to time, and they use the trend information they find to determine the INFOCon threat level, much like Symantec defines the ThreatCon level with DeepSight data.

ISC does not share actual logs, but they produce high level statistics. For this reason, their port activity and trends data do not need to be sanitized. They sanitize information about scanner source IPs by looking very broadly at the number of scans per class C network. This kind of anonymization is also used in some of the CAIDA logs where they simply truncate IP addresses. The danger is pretty low in sharing this kind of information, but its usefulness is also minimal. It does nothing more than allow inferences such as "The US does the most scanning" or "Universities contribute to most of the P2P traffic". Many of these statistics can be predicted from the density of addresses assigned in the respective class C networks. ISC does share specific addresses in one place: it lists the top 10 scanners by IP address. Many organizations use this information to block misbehaving machines. The repercussion of doing this is again minimal. They do not provide specific details about those machines or the networks they are on. It simply serves to embarrass the ISPs that host the compromised machines. Any sort of anonymization here would defeat the purpose.

Overall, the type of data they provide is a homogeneous set of aggregated statistics. More information can be gathered from CAIDA logs because raw access is provided. Thus, one is not restricted to only the statistics they provide. The main difference of course is that the ISC is real-time and uses a more distributed sample. In conclusion, ISC works very well for monitoring general worm behavior, detecting trends that indicate new worms and analyzing the life cycle of an exploit. However, they are not gathering many types of logs, and they are not sharing them with the general community for research.

4) DShield.org: DShield.org [8] is a grass-roots log collection system, though it is now funded partially by SANS. They gather firewall logs and convert them to a standard format. Currently, they exclusively accept packet filter traces. These are used to create reports of types similar to the Internet Storm Center. They have reports on port activity trends, the top 10 most offensive scanners and the top 10 most probed ports. They produce the blacklist of offenders that the ISC uses and provide searches on activity by particular IP addresses.

Anonymity is not dealt with seriously here, and they say "You should not submit any information you consider business critical or proprietary". They say that they "try" to hide destination IPs to mask who is being attacked, but raw data is searchable and may be made available to the public. Of course they do not indicate the submitter of the data. Decisions of what and how to release raw data is made on a per individual basis.

Several problems exist with the system as is. First, there is only one type of log. Second, there are no precautions to keep people from resubmitting logs and polluting the data set. If special clients are used instead of web submissions, accidental resubmissions are prevented. Third, anonymous submissions allow fake data to be submitted that could wrongly blacklist individuals. Fourth, even if their non-guaranteed anonymization of target hosts works, anyone can query information about specific hosts and networks. This allows attackers to find already compromised machines on a network rather easily. In conclusion, DShield.org provides data of limited types and minimal data protection mechanisms.

5) Packet Vault: The University of Michigan has worked on a secure, long-term archive of network packet data they call the *Packet Vault* [1], [2]. It is basically a special purpose network device and encrypted database system specifically designed for packet sniffers. It is designed so that selected traffic can be made available without exposing other traffic.

They sanitize logs by completely encrypting packet information. Since all fields are either encrypted or all fields in a record decrypted, it may not be appropriate to call it anonymization. They group items under the same encryption key if the packets are part of the same "conversation". Now, if instead they

used different keys for different fields, that would allow them to release different views of the same records to different organizations. That would be a crude sort of black marker (all-or-nothing) anonymization of selected fields. But their goals are different from ours. They are not trying to share logs while preserving privacy. They are making logs available to participants of the conversation, but not to anyone else. They give the appropriate keys to decrypt log records that describe a participant's actions but not those in which they were not a participant.

In conclusion, we see that while there are some centers dedicated to collecting log files, they all suffer from one or more of the following problems: (1) They do not have a wide view of the Internet but are quite localized, (2) the repositories are very specific, addressing one or only a few types of logs, (3) anonymization is weak or nonexistent and usually inconsistent, or (4) they collect many logs but do not share them with the research community.

B. Anonymizers

While there has been some research investigating log anonymization, most work has only addressed a small subset of all the available log sources (usually just network traces) and focuses exclusively on anonymizing IP addresses within a log. Though, even anonymizing IP addresses is not as simple as just removing or randomizing IPs. Such solutions are undesirable since they destroy a basic structure used in analyzing logs. Significant work has been accomplished on prefix-preserving anonymization of IP addresses. In prefix-preserving anonymization, IP addresses are mapped to pseudo-random anonymized IP addresses by a function we will call τ. Let $P_n()$ be the function that truncates an IP address to n bits. Then τ is a *prefix preserving* permutation of IP addresses if $\forall\ 1 \leq n \leq 32$, $P_n(x) = P_n(y)$ if and only if $P_n(\tau(x)) = P_n(\tau(y))$.

Several tools have been made that make use of this new form of IP address anonymization. Most are based on, TCPdpriv, a free program that performs prefix-preserving TCPdump trace anonymization using tables. Because of the use of tables, it is difficult to process logs in parallel with this tool. In [26], [27], Xu et al. have created a prefix-preserving IP pseudonymizer that overcomes this limitation by eliminating the need for centralized tables to be shared and edited by multiple entities. Instead, with their tool CryptoPAn, one only needs to distribute a short key between entities that wish to pseudonymize consistently with each other. In [20], we used CryptoPAn with our own key generator to perform prefix-preserving IP address pseudonymization on a particular format of NetFlow logs. We have completely reimplemented this in Java with a different key generator in vastly more advanced NetFlow anonymizer we call CANINE [21].

Pseudonyms are an important building block to many anonymization schemes. One of the earliest uses of pseudonyms can be found in [6] where public keys are used as pseudonyms. We now recognize what Chaum described in [6] as a "digital pseudonym" to be a specific type of pseudonym called an authorization certificate. As noted in [14], pseudonyms help define middle ground in the zero-sum tradeoff between security and privacy of audit logs. In [22], Sobirey et al. first suggested privacy-enhanced intrusion detection using pseudonyms and provided the motivation for the work of Biskup et al. in [3], [4].

While the work in [14], [3], [4] does deal with log data and anonymization, their goals are significantly different than ours. All three works deal specifically with pseudonymization in Intrusion Detection Systems (IDSs). The adversary in their model is the system administrator, and the one requiring protection is the user of the system. In our case, we instead assume that the system/network administrators have access to raw logs, and we are trying to protect the systems from those who would see the shared logs. To contrast how this makes a difference, consider that in their scenario the server addresses and services running are not even sensitive—just information that could identify clients of the system. Furthermore, we do not care about reversal of pseudonyms. We have no need for that capability, but since the system/network administrators do not have raw data in their case, the privacy officer must help the system security officer reverse pseudonyms if alerts indicate suspicious behavior. In [3], [4], they take this further and try to support automatic re-identification if a certain threshold of events is met. In that way their pseudonymizer must be intelligent, like an IDS, predicting when re-identification may be necessary and thus altering how it pseudonymizes data. They also differ from us in that they create transactional pseudonyms, so a pseudonym this week might map to a different entity the next week. We, however, require consistency with respect to time for logs to be useful. Lastly, all of the anonymizing solutions in these papers filter log entries and remove them if they are not relevant to the IDS; we endeavor to dispose of no entries because completeness is very important for logs released to the general research populace.

In [9], Flegel takes his previous work in privacy preserving intrusion detection [3], [4] and changes the motivation slightly. Here, he imagines a scenario of web servers volunteering to protect the privacy of visitors from themselves, and he believes IP addresses of visitors need pseudonymization. However, to a web server IP addresses already act as a pseudonym protecting the client's identity, since ISPs rarely volunteer IP address to person mappings. Though the motivation differs slightly, the system described is the same underlying threshold based pseudonymization system, and the focus of this paper is really about the implementation and performance of the system. As such, the results of [9] can be applied to [3], [4].

In [17], Pang et al. developed a new packet anonymizer that anonymizes packet payloads as well as transactional information, though their methodology only works with application level protocols that their anonymizer understands (e.g., HTTP, FTP, Finger, Ident and SMTP). The process can also alter logs significantly, losing fragmentation information, the size and number of packets and information about retransmissions, skewing time stamps, sequence numbers and checksums. While their anonymizer is limited in its capabilities, it is fail-safe because it only leaves information in the packets that it can parse and understand. Further, they create a classification of anonymization techniques and a classification of attacks against anonymization schemes that we found useful. We use a similar classification which is based off of their work.

Waters et al. [25] address the tension between data access control and searchability of audit logs through a new method

they developed to search asymmetrically encrypted logs. In this way, the encrypted log can be made public for search, and the owner distributes private keys corresponding to keywords. Thus, instead of the data owner decrypting the log and running the search, he can simply give the query maker the ability to perform the query with a set of keywords he deems acceptable. However, this is less like anonymization, and more related to the earlier discussed "packet vault" system.

Most recently, Lincoln et al. in [13] proposed a log repository framework that enables community alert aggregation and correlation, while maintaining privacy for alert contributors. However, the anonymization scheme in the paper is partially based on hashing IP addresses. Such a scheme is always vulnerable to dictionary attacks. Moreover, their scheme mixes the hashes with HMACs and truncates the hashes/MACs to 32 bits, both actions which result in more hash collisions and inconsistent mappings. Finally, their suggested use of re-keying by the repository destroys the correlation between repositories and therefore limits the view to a single repository.

IV. Attacks Against Current Anonymization Schemes

It is difficult to create secure anonymization schemes, and few fully address even basic attacks. An anonymization scheme is said to be secure if one cannot link records or other information to specific entities (e.g., hosts or users). All anonymization schemes must strike a balance between security and utility—utility being a measure of how much useful information remains after anonymization. Schemes that can be deductively proven secure tend not to be useful because of severe information loss. Pseudonymization—where identifiers are replaced by pseudonyms—tends to have more utility, but there are often more attacks against such schemes. In what follows we examine five classes of attacks against anonymization/pseudonymization schemes. These are similar to what Pang et al. define in [17]. These attack types are the basic building blocks of more advanced attacks and are often combined together and used against multiple heterogeneous logs.

A. Fingerprinting

We closely match Pang et al. by defining fingerprinting as *the process of matching attributes of an anonymized object against attributes of a known object to discover a mapping between anonymized and unanonymized objects*. For example, consider an anonymized NetFlow log of an organization with only one web server with an IP address of 192.168.77.29. Suppose that we see an anonymized IP address of 10.19.21.3 that is responding to port 80 connections. Further, 95 per cent of all port 80 connections are going to this address. Then we can infer that 192.168.77.29 maps to 10.19.21.3. This is most likely correct unless someone is running an illegal web server with a tremendous amount of traffic. Fingerprinting is most useful in identifying servers because of the unique attributes they possess.

B. Structure Recognition

Structure recognition is *the act of recognizing structure between objects to use one mapping to discover multiple mappings between anonymized and unanonymized objects*. By themselves these attacks reveal no mappings, but when used with a known mapping they can reveal new mappings. One example would be a common attack against all prefix-preserving IP address pseudonymization schemes. In this case, knowing one IP address mapping reveals unanonymized bits of any addresses which shares a prefix with one of the known pseudonyms. Here, we are exploiting the structure of CIDR addressing and the structure-preserving properties of the anonymization technique. Another example is the recognition of a port scan in an anonymized trace. This could reveal a sequential ordering of anonymized addresses. Knowing just one mapping between anonymized and unanonymized addresses would reveal all mappings for the machines which were scanned.

Notice that our definition is more restrictive than Pang et al.'s. Their definition is broad enough to allow overlap between fingerprinting and structure recognition. For example, one could recognize the structure of traffic patterns to determine a gateway server. By their definition that could be fingerprinting or structure recognition. We, however, do not count that as structure recognition. Structure recognition itself does not reveal mappings. Rather, it discovers relationships that allow the discovery of one mapping to aid in the discovery of more mappings.

C. Known Mapping Attacks

Known mapping attacks *exploit the discovery of a mapping between unanonymized and anonymized data in one reference, to undo anonymization of that same data in multiple places*. This kind of attack can occur in two manners. For example, a username and password field may be anonymized in exactly the same manner. In that case the word "Emily" could be used as a username in one record and a password in another. Revealing the mapping in one field would reveal it in another. A similar attack could be carried out if IP mappings are consistent across multiple logs. In that case, revealing a mapping in one log reveals the same mapping in another log—not necessarily of the same type.

D. Data Injection

Data injection attacks *involve an adversary injecting information to be logged with the purpose of later recognizing that data in an anonymized form*. To illustrate this type of attack, imagine the following scenario. An organization has wanted to help the research community and has just heard about prefix-preserving anonymization. They now feel safe to share logs and decide to publicly post anonymized logs on a regular basis. An adversary, Anton, knows that anonymized logs are regularly posted from this organization and, like everyone else, has access to them. Anton knows that if he had an internal view of the network, he could learn a lot more about how machines interact and are configured. He can send limited probes to certain machines, but a lot of these probes are filtered and consequently his IP address quickly blocked. He knows the logs from the firewalls, routers and IDSs would help a lot, but they are protected by prefix-preserving anonymization. However, Anton quickly realizes that if he could recognize his own scans

in the anonymized logs, he could figure out the mappings of several IP addresses through a structure recognition attack. Because of the prefix-preserving property, every mapping he discovers yields many bits in the IP addresses of machines he did not even scan. So with just a few scans, he can glean information about almost all of the pseudonyms. The only problem left for Anton is how to make scans that he can later recognize. This is quite simple. His scans need only make use of special sequences that are not random, but could look random without tenacious investigation.

A quick solution might be to use something as simple as the Fibonacci sequence. A smarter solution would be to use a sequence determined by a PRNG seeded with a password. For instance, what if Anton scanned a particular machine with the target or source ports being Fibonacci numbers? That should be very recognizable in a log if one knows to look for it. He could do the same thing with timings between probes. Of course, the timing intervals must be large enough that jitter in the network does not cause mismatches on a pattern. In fact, such an attack could be made with a number of fields, even in obscure TCP options.

E. Cryptographic Attacks

Many anonymization techniques depend on cryptographic functions. Such systems can be vulnerable to attacks on the cryptographic primitives, such as known or chosen plaintext attacks. These attacks could reveal secret keys or other information. Such attacks do not affect just single entries, but all entries of a particular field in a log. However, cryptography is less likely the weak link in a scheme. It is usually easier to perpetrate the other types of attacks. Also, this kind of attack typically requires mappings to first be discovered through other attacks. For example, a data injection attack may be needed in conjunction with a chosen plaintext attack against a particular cryptographic algorithm. It is interesting to note that there are many similarities between these data injection attacks and covert channels.

V. Roadmap to Widespread Log Sharing

Ultimately, a new anonymization framework—independent of specific log types—that can anonymize logs to multiple levels is needed. This would allow organizations to customize anonymization techniques to their needs, and they would be able to map different trust relationships to different levels of anonymization. The ability to handle multiple log types and to perform trade-offs between security and utility are vital features missing from current tools. Secondary to this, anonymization techniques should be standardized for most log types through IETF drafts and RFCs. Along these lines it would also be useful to document usage studies to help organizations map levels of trust shared with others to the newly standardized anonymization levels. However, many steps must be taken to meet these goals.

A. Ground Work

We believe a classification of logs must first be developed that is based upon the kinds of attacks that can be detected with different logs. This classification should be based on the fields in the logs and not the log versions/types themselves. This classification system will allow one to measure the maximum utility of a log after a particular subset of fields has been anonymized. Obviously, as more fields are anonymized, the balance between security and utility will lean more towards security, and with this classification, a metric—of how much utility remains post-anonymization—can be created.

Similarly, a metric of the security of an anonymization scheme—the types of data being anonymized and the anonymization algorithms applied to the data set—must be created. This will require an understanding of the types of attacks against anonymization schemes—work we have already begun as seen in the previous section of this paper. This metric should depend upon the amount of information revealed by the attacks as well as the difficulties in perpetrating them—this being dependent upon the adversary in the model.

Simply creating these two metrics is novel work, and their completion will allow one to determine, quantitatively, where a particular anonymization scheme sits on the sliding scale between security and utility. This will enable the creation of more useful levels of anonymization and assist the process of standardization. It is also important to note that there are two approaches to creating these metrics: an information theoretic approach and an empirical approach. An information theoretic approach creates deductively proven measures and yields hard guarantees about disclosure of information within the logs. However, such a metric would only tell us we are guaranteed some minimum or maximum levels of utility and security, and it does not tell us how secure a scheme is in practice. An approach, like ours, that takes currently known attacks and correlates them with anonymization methods, gives a more precise and pragmatic measure of the security of an anonymization scheme. It is likely that some anonymization schemes are secure in practice, but they cannot be proven without some reasonable assumptions. In fact, this is true of the most commonly used cryptographic algorithms today. While no crippling attacks against these algorithms are known, many have not been proven secure. So while this is often a more useful way to measure security, it makes the metric dynamic. Something that is secure one day could be insecure the next month as a new attack may be developed and published quickly. Thus standards based off of these metrics would also need to be dynamic.

B. Architecture

We envision a client-server architecture for this framework. A client, who can be a system administrator or a regular user, requests data from a server that we call the *Coordinator*. The coordinator must retrieve the data for the client, determine the required anonymization level, anonymize the data and return it to the user. To answer a request, the coordinator will have to communicate with two other entities: the *Profile Manager* and the *Data Store* (Figure 2 illustrates the relationships between these components).

The profile manager stores profiles that determine the anonymization level for a client. These profiles will be created by security administrators and/or data owners. A profile will specify an anonymization algorithm for each field of a data

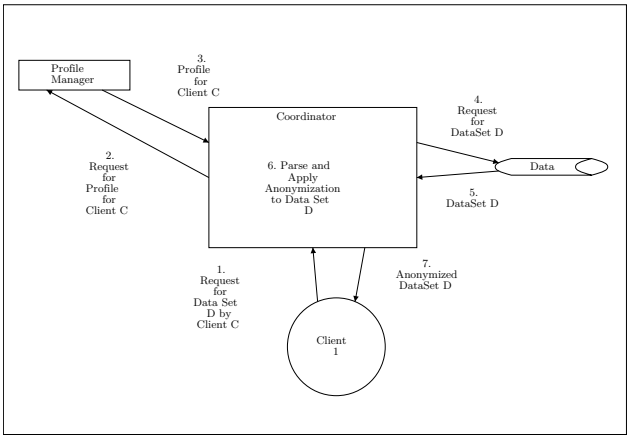

Fig. 2. This figure shows the overall system architecture and how information is passed between components upon requests for specific logs.

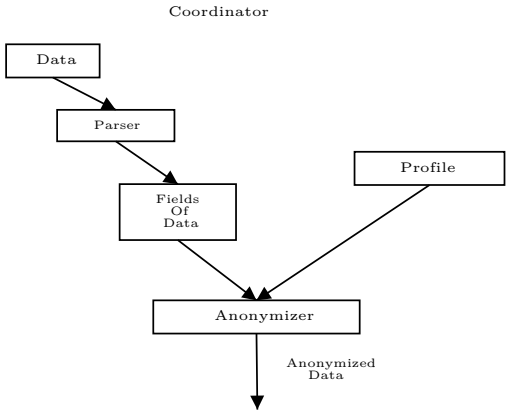

Fig. 3. This is a detailed figure presenting how information is passed through the different subcomponents of the *Coordinator*.

set. For instance, suppose a student from University A requests data from University B. University B's security administrator can create a general profile for students from other universities. This profile will specify what fields of the requested data source must be anonymized and in what manner.

The profiles could be created ahead of time by the security administrator or data owners and should reflect the level of sharing the organization wishes to allow. A particularly private organization may wish to anonymize every possible field in one way or another for all possible clients. A more liberal organization might have a continuum of profiles with some clients (e.g., other security administrators) being granted data that has only been lightly anonymized, whereas the general public might receive heavily anonymized data.

In order to apply the profile to a data set, the coordinator must first determine what fields are in the data and then actually apply the anonymization algorithm to the appropriate fields. The *Parser*—a component of the coordinator—will take a data set and break it down into its component fields (This subcomponent can be seen in figure 3 which gives a detailed view of the coordinator). This can be a difficult task. Some logs, such as syslogs, cannot be completely parsed. There is no algorithm to completely parse and analyze syslogs since there are almost no parameters set on how a program must send alerts to the syslog daemon. As an example of how difficult it is to parse syslogs, consider that it is possible to find binary dumps in a syslog—this commonly indicates the occurrence of a buffer overflow. The conservative solution is to parse what is understood, and to throw away the rest. It is dangerous to leave information that cannot be processed because it will not be anonymized. Instead, we believe it better to be fail-safe. While an implementation cannot parse every conceivable syslog entry, by recognizing the structure of entries from, say, the ten most commonly logged applications in an environment, most alerts sent to a syslog daemon can be processed.

The *Anonymization Engine*—another subcomponent of the coordinator—will contain of a set of algorithms that anonymizes different types of data. For any particular data type (e.g., IP addresses, port numbers, timestamps, etc.) there may be multiple ways to anonymize the field. There are several ways to anonymize IP addresses alone. These algorithms would be reusable for different logs as the set of unique fields is much smaller than the set of possible log types. For example, port numbers and IP addresses are common to most network logs, and almost every log type has timestamp fields. Thus, as the framework matures, we expect there to be many times fewer anonymization algorithms than supported log types. To start, many of these algorithms will need to be developed as most previous research has focused only on IP address anonymization. However, anonymization techniques common to IP addresses, such as truncation, are likely to be applicable to other data types as well.

Implementing a coordinator, anonymization engine, parser and profile manager would require significant work. Work should begin by compiling a list of fields common to most logs and identifying the sensitive fields. Fields that by themselves are not sensitive, but in combination are vulnerable to exploitation, must be considered. With this list of sensitive fields, work creating different anonymization algorithms can begin. With these algorithms in place, results from user studies about sharing scenarios and an established utility metric, one can create different, useful anonymization levels—sets of fields paired with specific anonymization algorithms. This must be followed with a thorough analysis of attacks—such as those in the previous chapter—against the different anonymization schemes. This should be done as a security metric for anonymization schemes is developed.

C. Defining Standards and Beyond

Obviously different organizations trust each other for different tasks and at different levels. We have made it a point to emphasize this fact and that there is a balance between security (the difficulty in breaking an anonymization scheme) and utility (the usefulness of the information to other parties). For example, in the previous chapter we saw that prefix-preserving anonymization of IP addresses, a special type of anonymization unique to that data type, does not protect NetFlows from an active adversary. Anyone who can inject traffic to be logged, can begin to determine actual IP addresses. One solution to this problem would be to simply truncate or delete the IP address fields, but now the ability to correlate actions of a single

machine is lost. No longer can one attacker be distinguished from another while remaining anonymous. And even still, port numbers could give away information, particularly if used with time-stamps. To counter this attack one could add noise to the time-stamps and delete port numbers, but then there would be very little useful information left in the logs.

Organizations must determine how much they trust organizations that may see their logs and think of them as adversaries to some extent. By this we mean they must determine what kind of attacks are likely from parties who will see their logs. This, combined with the need-to-know level of the receiving party, will determine how to anonymize the logs. One may say this is easy because you simply never give more information out than is necessary, but our response to this is that one often does not realize how much information they are giving out.

Ideally, we would like to see the development of anonymization levels that are completely independent of the specific log type. Instead, they will depend upon the fields in the logs—of which there are significantly fewer types than of logs themselves. These levels could be published through an IETF RFC or a similar outlet. This will be particularly useful in creating more usable log repositories. Submitters will have a succinct and accurate way to describe how they anonymized their logs. Parties who are interested in viewing them need not waste their time with logs anonymized too much to be of use to them. They could simply search for all logs anonymized at a certain level or below.

While the main goal is to develop a framework, and creating standards is a secondary goal that we believe vital to the first, there is one more important task to enable widespread sharing of logs. People must understand how to apply the standards. Having different levels of anonymization is not enough. People must be able to map their specific situations and needs to the appropriate anonymization level. This is in effect a need to map trust levels to anonymization standards. For example, as a university we may trust graduate student researchers more than those from lab in a foreign country for the simple fact that we can take action against students who exploit the privileged information. Additionally, those with campus IP addresses already have a better view of our internal network. Thus a lower level of anonymization may be needed for sharing logs with students. Consequently, examples and scenarios to help implementors use the new anonymization technology are important. We believe usage studies will be imperative to develop such documentation, and their creation is an important tertiary goal to lead to increased log sharing.

VI. Summary

In this paper, we have defined the problems of sharing logs for security research and surveyed current attempts at widespread log sharing. We have argued that anonymization is necessary to solve these problems and analyzed current solutions to anonymize logs. Log sharing is still not happening, or at least not to the degree that the research community would like. We believe this reason to be that the field of log anonymization is immature, and current solutions and tools are insufficient to tackle the problem.

In response, we have started developing our own tools and presented a roadmap to overcome the problems plaguing log anonymization. We believe that there must be multiple levels of anonymization that trade-off between security of the anonymization scheme and utility of the anonymized log data. We have proposed a framework for a new class of anonymization tools that supports multiple levels of anonymization that map to the multiple levels of trust between those that would share their logs and would-be receivers of the logs.

Furthermore, we have argued that the creation of new metrics for the utility and the security of an anonymization scheme are necessary to create multiple, useful levels of anonymization. These metrics will also be a first step in defining standards that will allow the community to succinctly describe a level or type of anonymization used on a log. Lastly, we believe usage studies are necessary to develop resources to help organizations map levels of trust shared with would-be receivers to these newly standardized anonymization levels.

Before solving a problem, it must be understood. In this paper we have identified major issues facing log anonymization. There is much work to be done in reaching the goal of widespread sharing of log data; standards must be developed, metrics created and attacks on anonymization schemes considered. To date people have only focused locally on anonymizing a few selected fields in a limited number of logs; we look to the larger picture.

VII. Acknowledgments

First, we would like to thank our colleagues in the SIFT research group at the NCSA <http:/www.ncassr.org/projects/sift/> who indirectly contributed to this work; especially Ratna Bearavolu, Kiran Lakkaraju, Yifan Li, Katherine Luo and Xiaoxin Yin. We would also like to thank Jim Basney, Jun Wang and the SECOVAL workshop reviewers for their comments and input. This work is funded in part by a grant from the Office of Naval Research (ONR) under the umbrella of the National Center for Advanced Secure Systems Research (NCASSR).

References

[1] C.J. Antonelli and P. Honeyman, "Wiretapping the Internet", SPIE Symposium on Enabling Technologies for Law Enforcement and Security, 2000.

[2] C.J. Antonelli, M. Undy, and P. Honeyman, "The Packet Vault: Secure Storage of Network Data", USENIX Workshop on Design Issues in Anonymity and Unobservability, 2000.

[3] J. Biskup and U. Flegel, "On Pseudonymization of Audit Data for Intrusion Detection", USENIX Workshop on Design Issues in Anonymity and Unobservability, 2000.

[4] J. Biskup and U. Flegel, "Transaction-Based Pseudonyms in Audit Data for Privacy Respecting Intrusion Detection", Recent Advances in Intrusion Detection (RAID), 2000.

[5] "CAIDA – Cooperative Association for Internet Data Analysis", Jun. 2005; http://www.caida.org/projects/trends/data/

[6] D. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms", Communications of the ACM, Vol. 24 No. 2, 1981, pp. 84-88.

[7] A. Desjanun, "Hacker Teams Breach Powerful Research Networks", Apr. 2004; http://www.usatoday.com/tech/news/computersecurity/2004-04-14-synchronized-hacking_x.htm

[8] "DShield.org – Distributed Intrusion Detection System", Jun. 2005; http://www.dshield.org

[9] U. Flegel, "Pseudonymizing UNIX Log Files", Infrastructure Security Conference (InfraSec), 2002.

[10] The Honeynet Project, *Know Your Enemy: Learning about Security Threats (Second Edition)*, Addison-Wesley, 2004.

[11] *Information Sharing & Analysis Centers*, Department of Homeland Security, Jun. 2005; http://www.dhs.gov/dhspublic/display?theme=73&content=1375

[12] ISC – Internet Storm Center, Jun. 2005; http://isc.incidents.org

[13] P. Lincoln, P. Porras and V. Shmatikov, "Privacy-Preserving Sharing and Correlation of Security Alerts", USENIX Security Symposium, 2004.

[14] E. Lundin and E. Jonsson, "Privacy vs Intrusion Detection Analysis", Recent advances in Intrusion Detection (RAID), 1999.

[15] *The National Strategy to Secure Cyberspace*, The White House, Jun. 2005; http://www.securecyberspace.gov

[16] New Technologies Inc. – Computer Incident Response Suite, Jun. 2005; http://www.secure-data.com/suite1.html

[17] R. Pang and V. Paxson, "A High-Level Programming Environment for Packet Trace Anonymization and Transformation", ACM SIGSOMM, 2003.

[18] SIFT – Security Incident Fusion Tools Project, Jun. 2005; http://www.ncassr.org/projects/sift/

[19] S. Shakkottai, N. Brownlee and K. Claffy, "A Study of Burstiness in TCP Flows", Passive and Active Measurement (PAM), April 2005.

[20] A. Slagell, J. Wang, W. Yurcik, "Network Log Anonymization: Application of Crypto-PAn to Cisco NetFlows", NSF/SFRL Workshop on Secure Knowledge Management (SKM), 2004.

[21] A. Slagell, Y. Li and K. Luo, "Sharing Network Logs for Computer Forensics: A New tool for the Anonymization of NetFlow Records", Computer Network Forensics Research Workshop, held in conjunction with IEEE SecureComm, September 2005.

[22] M. Sobirey, S. Fischer-Hubner and K. Rannenburg, "Pseudonymous Audit for Privacy Enhanced Intrusion Detection", IFIP TC11 13^{th} International Conference on Information Security (SEC), 1997.

[23] Symantec DeepSight, Jun. 2005; http://enterprisesecurity.symantec.com/products/products.cfm?ProductID=158&EID=0

[24] I. Thomson, "IT Industry's 12-point Cyber-security Plan", Dec. 2004; http://www.vnunet.com/news/1160087

[25] B. Waters, D. Balfanz, G. Durfee and D.K. Smetters, "Building an Encrypted and Searchable Audit Log", Internet Society Network Distributed Systems Symposium (NDSS), 2004.

[26] J. Xu, J. Fan, M. H. Ammar and S. B. Moon, "On the Design and Performance of Prefix-Preserving IP Traffic Trace Anonymization", ACM SIGCOMM Internet Measurement Workshop, 2001.

[27] J. Xu, J. Fan, M. H. Ammar and S. B. Moon, "Prefix-Preserving IP Address Anonymization: Measurement-based Security Evaluation and a New Cryptography-based Scheme", IEEE International Conference on Network Protocols (ICNP), 2002.