On the Anonymization and Deanonymization of NetFlow Traffic

Michalis Foukarakis*, Demetres Antoniades*, Spiros Antonatos*, Evangelos P. Markatos*
*Institute of Computer Science (ICS),
Foundation for Research and Technology Hellas (FORTH),
P.O BOX 1385, Heraklion Crete,
GR7-1110, Greece
{mfukar,danton,antonat,markatos}@ics.forth.gr

***Abstract*—Netflow is an efficient and flexible mechanism to collect network data and share them to security applications that require distributed knowledge. As information sharing breeds the danger of revealing user and network information, anonymization of Netflow data has to be applied before they are shared. To accomodate anonymization needs we have developed *anontool*. *Anontool* allows per-field anonymization up to the NetFlow layer offering a wide range of primitives to choose from. However, although we have the tools to perform anonymization, work needs to be done on the policy part. Some policies may be proven weak, if a third party can deanonymize the data and reveal user information. We demonstrate 2 possible attacks on anonymized traces. The first one is called active fingerprinting, where a malicious user injects packets that she can identify in an anonymized trace and thus reveal the mappings of IP addresses. The second one is the disclosure of web pages that users access based on the flow sizes recorded in a trace. We also present solutions to these attacks along with two anonymization primitives which we have implemented in *anontool* in order to defend against such attacks, to the extent possible.**

I. Introduction

As computer networks evolve and grow in size, the need for distributed network management and monitoring becomes more important than ever. Network activity log sharing has gained significant popularity recently, not only among computer security engineers and administrators, but also among researchers, developers and educators. To accommodate this increasingly popular need for information sharing as well as the fundamental lack of trust between members of different communities, several tools have emerged which enable their users to anonymize potentially sensitive information within those logs.

A popular format used by network activity logs is the Cisco NetFlow [1] format. The NetFlow format is based on the concept of a flow, which Cisco has defined as a set of packets that have the following five properties in common: source and destination IP address, source and destination port numbers, as well as the IP protocol field value. The most recent evolution of the Netflow format is version 9, which is currently the basis of the IETF [2] standard for information export. Given this fact, NetFlow is likely to gain even more in popularity.

Many diverse tools and techniques have been implemented for anonymization purposes. Most of them, however, are customized for specific purposes or provide limited functionality. Our approach, *anontool*, is a general-purpose tool that can anonymize live or stored traffic. Using Anontool, a user can choose from a large variety of functions to use on each and every application field, to implement her anonymization policy of choice. *Anontool* supports a number of protocols, but in this work we will focus on the NetFlow protocol.

Despite what tools one might use, the final result log of the anonymization process is likely to be publicly available. There is always the potential for an adversary to be able to infer a large amount of information from the log, if the anonymization policy is not chosen carefully. We will describe a few scenarios where an adversary can manipulate the log anonymization process in order to deduce useful information about the original trace. Given those attack scenarios, the need for techniques to defend against them becomes clear. We therefore describe two anonymization primitives which can aid in protecting against such attacks. A proof-of-concept implementation of them has already been incorporated into the latest version of *Anontool*.

II. Anontool Description

Anontool is a command line tool which enables users to anonymize both live and stored traffic. Its functionality is based upon the Anonymization API [3], in short AAPI. AAPI allows users to write their own anonymization applications. They can define which anonymization function be applied on any field, having complete freedom in deciding their policy. It provides a large set of anonymization primitives, from setting fields to constant values and performing basic mapping functions to prefix-preserving anonymization and several hash functions and block ciphers, as well as support for regular expression matching and replacement. AAPI can operate on a wide variety of protocols, ranging from Ethernet to HTTP and FTP in the application layer. All protocol fields are being made available to the user application.

AAPI has been implemented as a user-level library in the C language; it provides function calls for creating packet ”streams”, filtering using BPF filters, and of course applying anonymization functions. One of its main design goals was to accomodate extensibility, and potential developers are able to write their own protocol decoders similar to the ones already available for HTTP or NetFlow protocols, such as SMTP, or

their own anonymization primitives. It is also straightforward to write code that supports new input sources with few code additions.

Since the NetFlow format for packet export continuously gains on popularity [4], [5], [6] we have decided to extend AAPI, and subsequently Anontool, with support of the Cisco NetFlow packet export format. We took advantage of AAPI's extensibility and implemented decoding and anonymization functions for both version 5 and the newly defined version 9 of the NetFlow format. We take full advantage of the template-based nature of the NetFlow v9 format, to accurately provide the user with complete control of every field made available from information export nodes, such as Cisco routers or network monitoring applications that support the NetFlow export format, even in the event NetFlow templates change during a monitoring period.

Anontool is a fairly simple C application that makes use of the AAPI library to support anonymization of packet traces. It does not implement any anonymization functions in itself; it is much more transparent and less error prone for all the anonymization functionality to reside inside the AAPI implementation. It provides users the choice of protocols and functions to apply in order to create their anonymization policy. *Anontool* also implements some functionality such as preprocessing a trace to extract information which may be consequently used in the actual anonymization process; we will explain in further detail in Section IV.

III. Attacks Against NetFlow Anonymization Policies

In this section, we describe in detail two attacks against conventional anonymization policies and outline related work in both packet traces and flows.

We assume that the anonymization process is applied onto NetFlow traces, which are, in the general case, generated from a router at the border of a monitored network, and export information for traffic entering and exiting this particular network. The network could be of any size or topology; from a small home network to larger networks belonging to research institutes, universities, and so on.

In our threat model, we assume an active adversary that is able to direct traffic to the monitored network at will, has knowledge of the address space it occupies and can potentially compromise hosts inside it. We assume a rational attacker, for whom it is less costly, or more useful even, to "probe" and profile the monitored network before mounting attacks against it. The adversary may also have several external hosts under her command. She is also able to gain access to the anonymized traces, which will most likely be publicly released.

The first attack, which we call *"Active Fingerprinting"*, aims to break the mapping algorithm when used on IP addresses. Mapping takes the set of IP addresses in a trace and performs a simple mapping function onto another totally different set. The second set may be the output of a deterministic function seeded by a random quantity, such as the *drand48()* family of functions, or a very simple sequential assignment of unique IP address numbers which results in a one-to-one mapping.

The second attack aims at using the information about flow sizes contained inside NetFlow traces in order to deduce information about either hosts inside the monitored network or hosts that may be outside it, such as their IP address, network usage profiles, etc. We name this attack *"Statistical Signature Inference"*.

A. Active Fingerprinting

The idea that active fingerprinting exploits is that the mapping between real and anonymized IP addresses is one-to-one. Consequently, if the mapping on one flow is discovered, the mapping on the whole trace is compromised. This attack has been described on packet traces in [7], [8] and we demonstrate its applicability on NetFlow records below.

Using this idea, an adversary can establish flows from a host under her control, which resides outside the victim network, to one or more victim hosts inside it. These flows will appear in the anonymized trace. The challenge for the adversary is to construct those flows in such a way that they will be easily distinguished in the final trace. This can be accomplished in a variety of ways; she can craft a flow with specific attributes which are known not to be anonymized (the list is as large as the potential fields listed in a NetFlow record, and may usually include TCP flags, IP ToS, and so forth), or in the unlikely scenario where flows are fully anonymized, she can use temporal patterns which are easy to detect. Even a specific packet size can be used as an identifier for the packets involved in a dictionary attack. If the traces from the monitored network span a wide enough time period, the latter attack is very feasible as the trace contains a large number of attack packets.

A trivial way an adversary could create these flows is to perform a SYN scan on the victim network's address space. In this case, even if there is a clear temporal pattern which is easily detectable in the anonymized trace, it can be defeated in an easy way. Setting only the SYN bit in TCP flags and setting the number of bytes to a specific quantity makes the adversary unable to distinguish live hosts from unused address space.

We discuss a more general measure to defend against this kind of attack in Section IV.

B. Statistical Signature Inference

The idea behind this kind of attack is that each web page has a unique and complex enough structure which allows them to be identifiable despite our best efforts to anonymize their presence in NetFlow logs and preserve useful information in them as well.

A naive first effort would be the following. Consider the web sites **interesting.com** and **newssite.com**, and that a web session with each of them is **n** and **m** bytes long, respectively. The adversary can use one of the hosts under her command to initiate a web session to these sites and view the NetFlow records for source and destination IP addresses, port numbers,

and the total size of traffic exchanged. Assuming web page sizes do not radically differ from one session to another, and that NetFlow data records TCP traffic in its entirety, it is possible to filter out the set of web browsing sessions from an anonymized trace and construct a frequency histogram with the number of bytes transferred in each flow. According to our assumptions above, it is possible to see a great deal of flows around the values of **n** and **m**. The adversary can employ the same tactic to find those flows, and then gather further information about hosts inside the monitored network, which can then be used to answer questions such as: *"What web sites does host A visit?"*, *"Which hosts do frequently visit www.google.com?"*, and make user profiles.

Past work [8] has demonstrated this kind of attack on packet traces. Recent work [9] has extended and demonstrated this attack on NetFlow logs as well. We argue that the fundamental property of web sessions that allows this kind of analysis to be exploited is the fact that web sessions to different hosts produce flows with similar characteristics, especially in flow size. In Section IV we are going to view our proposal of a primitive which deals with this issue.

IV. Countermeasures

The previous section described two attacks for revealing sensitive information from an anonymized NetFlow trace. In this section, we will describe our proposed ways to deal with the aforementioned attacks, and evaluate their consistency.

A. Bidirectional Mapping

We propose a way to deal with the issue that does not iteratively consider all the combination of fields an adversary may use to craft her flows. Instead, we aim to eliminate the one-to-one mapping property without losing all of the information the trace can provide. To this goal, we propose a bidirectional mapping to be used, that is different mapping for each traffic direction. Let **A** be the IP address of a live host inside the monitored network, and **B** the IP address of a host outside the monitored network. Conventional mapping functions would map a flow *(A, B)* to *(A", B")* and a flow *(B, A)* to *(B", A")*. Bidirectional mapping maps a different address to **A** according to the direction of the flow that involves it. In the case of our example, the flow *(A, B)* would be mapped to *(A", B")* yet the flow *(B, A)* would receive a different mapping, say *(C, D)*.

Using this anonymization scheme we prevent the attacker from identifying her own network flows inside the anonymized trace. Thus, it is made impossible to correlate her data with the trace information and reveal any sensitive data from it. Also, most of the statistic information derived from the trace remains the same. We can still gather information about the incoming and outgoing traffic of the organization and identify the producers and the consumers of the network. Correlation of incoming and outgoing traffic for a specific IP address can not be done, but we argue that this is a general trade-off of the anonymization process and is up to the organization to decide whether to sacrifice sensitivity for usability in the data it makes public.

The implementation of such a primitive is quite easy, and it is already included in the stable version of *Anontool*.

B. Random Value Shifting

In order to diminish the viability of a statistical identification approach, but still be able to calculate some basic representative statistics about a NetFlow log, we propose a randomized shifting of values, which we will describe below.

Given a NetFlow data field with a given value range, our intent is to "scramble" its values across the NetFlow log to the point that we make an adversary unable to distinguish between two web sessions with the same web site and two web sessions with web sites that have similar web pages in size, but not as much as to destroy all the useful information a log may provide. More specifically, we intend to preserve metrics such as arithmetic averages and standard deviation, as well as other descriptive statistics. On the other hand, we wish to obfuscate inferential statistics, so that an adversary would be unable to reach conclusions that extend beyond the immediate data alone.

For clarity, we are going to use the flow size NetFlow field as an example from here on.

Our method is to add to the value of the flow size field a random value. This value is chosen uniformly at random from a fixed range *[-d, d]*. One of the basic properties of our choice is it allows us to directly preserve the arithmetic average and standard deviation of the original distribution of flow size values. The parameter d may be chosen arbitrarily, but we will demonstrate the importance of an educated choice with an example. Consider, as an elementary example, three flows with sizes of *15*, *17* and *25* bytes, which repeatedly occur within a NetFlow log. Choosing d to be equal to two, this is what happens in the anonymized trace: The flows with the initial size of *15* now occur with flow sizes from thirteen to seventeen. Flows with the size of *17* bytes now occur with sizes from fifteen to nineteen. These two groups of flows are now "mixed up"; what happens is that the confidence intervals for the random variables which represent the flow sizes of each flow are now different, and they overlap. On the other hand, that is not the case for the flow with size *25* bytes. It now occurs on the anonymized trace with values from 23 to 27. An adversary is still able to distinguish this flow from the other two with relative ease. Now we can easily conclude that a proper choice for the parameter d will have to take the entirety of the NetFlow log into consideration. This is an interesting topic for future work, which we will not further explore in the rest of the paper due to lack of space.

To verify our assumptions about the descriptive statistics of a NetFlow log being preserved after the application of random value shifting, we implemented it in *Anontool* and proceeded to process a NetFlow packet trace with it. Our choice for the parameter d was the minimum flow size observed, divided by 2. As we previously mentioned, this is most likely not a good choice for real world applications, but it is good

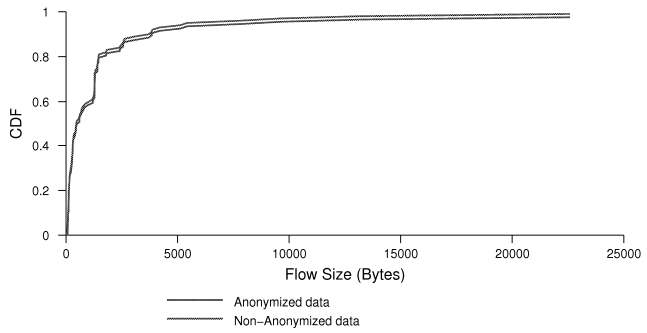

Fig. 1. Cumulative distribution function of flow sizes

enough for the experimental evaluation we describe. We then calculated the arithmetic average and the standard deviation for both the original and the anonymized trace, which we present here. The NetFlow trace spanned a time period of three minutes and a bit more than 150.000 bytes transferred. Table I presents the values calculated for both traces. We can see that the average and standard deviation do not largely differ. This supports our initial hypothesis, that we can preserve some amount of general information about NetFlows in the trace even after performing random value shifting. Figure 1 presents the cumulative distribution function of the distribution of flow sizes in the original, and the anonymized traces, for comparison and reference. As we can see the distribution remains, almost, identical after the anonymization process. This enforces our initial argument that the information that can be derived by the anonymization process are not affected by the value scrambling.

Trace	**Average Flow Size (bytes)**	**Std. Deviation**
Original	1843.69	7336
Anonymized	1845.02	7335.52

TABLE I
SOME BASIC DESCRIPTIVE STATISTICS REGARDING A NETFLOW TRACE BEFORE, AND AFTER ANONYMIZATION.

Currently, *Anontool* performs some basic trace preprocessing when random value shifting is going to be used. It processes all packets in a trace in order to extract the information it needs to calculate d. This information is dependent on the field that random value shifting is being applied on. After the value of d is calculated and chosen according to the user's method of choice, the actual anonymization process takes place. It is possible to estimate d during the anonymization process, however, as knowledge of the whole trace is impossible to have until the whole trace has been processed, we believe the estimated value will not yield as good a result as when a trace can be preprocessed and the value of d calculated on it.

V. CONCLUSIONS AND FUTURE WORK

We described two scenarios where an attacker can, by manipulating the anonymization process, deduce useful information about non-anonymized data in NetFlow traces. Those attacks have already been carried out in packet traces, and we have shown their applicability on NetFlow logs as well. In order to protect against these types of attacks, we have introduced two anonymization primitives and discussed their use and parameterization in order to make educated choices about anonymization policies. We also provided data which suggest their use still preserves useful data about NetFlow logs, without exposing inferential statistics to potential adversaries.

VI. AVAILABILITY

Anontool can be downloaded from http://dcs.ics.forth.gr/Activities/Projects/anontool.html. The application has been installed and tested on RedHat and Debian distributions of the Linux operating system.

REFERENCES

[1] Cisco Systems, Inc, "Netflow Specification." [Online]. Available: http://www.cisco.com/warp/public/732/Tech/nmp/netflow/index.shtml
[2] "Ip flow information export (ipfix)." [Online]. Available: http://www.ietf.org/html.charters/ipfix-charter.html
[3] D. Koukis, S. Antonatos, D. Antoniades, P. Trimintzios, and E. Markatos, "A generic anonymization framework for network traffic," in *Proceedings of the IEEE International Conference on Communications (ICC 2006)*, Jun. 2006.
[4] M. P. Collins and M. K. Reiter, "Finding peer-to-peer file-sharing using coarse network behaviors," in *ESORICS*, 2006, pp. 1–17.
[5] H.-J. Kang, M.-S. Kim, and J. W.-K. Hong, "A method on multimedia service traffic monitoring and analysis," in *DSOM*, 2003, pp. 93–105.
[6] B. Nickless, J.-P. Navarro, and L. Winkler, "Combining cisco netflow exports with relational database technology for usage statistics, intrusion detection, and network forensics." pp. 285–290.
[7] R. Pang, M. Allman, V. Paxson, and J. Lee, "The devil and packet trace anonymization," *ACM Computer Communication Review*, vol. 36, no. 1, pp. 29–38, Jan. 2006.
[8] D. Koukis, S. Antonatos, and K. G. Anagnostakis, "On the privacy risks of publishing anonymized ip network traces," in *Communications and Multimedia Security*, 2006, pp. 22–32.
[9] S. Coull, M. Collins, C. Wright, F. Monrose, and M. Reiter, "On web browsing privacy in anonymized netflows," in *16th USENIX Security Symposium*, 2007, pp. 339–352.