PCAPLib: A System of Extracting, Classifying, and Anonymizing Real Packet Traces

Ying-Dar Lin, *Fellow, IEEE*, Po-Ching Lin, Sheng-Hao Wang, I-Wei Chen, and Yuan-Cheng Lai

***Abstract*—This paper presents the `PCAPLib` system for providing *extracted*, *well-classified*, and *anonymized* packet traces from real network traffic with two mechanisms. First, *active trace collection* actively extracts and classifies packet traces into sessions by leveraging multiple detection devices. Second, *deep packet anonymization* protects the *privacy* in the packet payloads for hundreds of application protocols while preserving the *utility* of the traces. We evaluate 318 anonymized packet traces collected over a period of four months and show that the efficiency of anonymization is up to 96%. The usefulness of this system for assessing false positives/false negatives in intrusion detection has been also demonstrated.**

***Index Terms*—Packet anonymization, privacy, trace repository, utility.**

I. Introduction

REAL-WORLD Internet traffic is useful for studies ranging from traffic characterization and analysis, diagnosis of network events, to evaluation of network systems such as intrusion detection/prevention (IDP) systems. Network research communities, as well as developers of network appliances, usually rely on large, diverse, updated, and nonsynthetic packet traces for experimental study and evaluation [1]. For example, network analysts who collaborate on inspecting malicious incidents and network behavior can share traces among one another.

Capturing and sharing real traffic face two major challenges. First, the packet traces in a large repository involve diverse application protocols, and they should be *well classified* beforehand for users to easily find the desirable traces. Several organizations have released packet traces for public access, e.g., [2] and [3]. The traces are usually submitted and categorized subjectively by enthusiastic contributors. They are often outdated, inconsistently categorized, and even unusable, as the packet repository lacks a central control mechanism to maintain the quality of packet traces and consistently categorize them. Moreover, if the packet traces are derived from a real environment, e.g., from an Internet service provider [4], the volume will be usually huge, and it is essential to automatically extract and consistently classify the packet traces in a scalable way.

Second, the packet traces may contain private information such as host addresses, e-mail addresses, and even authentication keys. Such information should be anonymized before the traces are shared. Although packet anonymization helps to protect the *privacy* of packet traces, it hurts their *utility* at the same time. Many existing methods anonymize only the fields in the Transmission Control Protocol/protocol suite (TCP/IP) header [5]–[9] and strip away the *payloads*. However, the payloads are likely to contain key information for network analysis in terms of signature matching for intrusions [10], [11], traffic identification [12], or payload-based anomaly detection [13], [14]. Several research works have attempted to preserve the payloads and anonymize specified key information in them [15]–[19]. The anonymization methods are still limited to only a few common protocols (e.g., HTTP and FTP) and unable to satisfy the need for large packet traces, which may be from a large number of application protocols. Although it is possible to extend existing works to support the anonymization for so many application protocols, the support will require implementing that many protocol parsers as well, yet the implementation effort is nontrivial.

In this paper, we design the `PCAPLib` system to automatically *extract*, *classify*, and *anonymize* packet traces from a large amount of real network traffic. The packet traces will be useful for various purposes of network analysis. For extraction and classification, the *active trace collection* (ATC) mechanism can automatically classify captured packet traces into application data sets. If the traces in a data set have malicious content, they will be categorized into a separate class of that data set. Network traffic classification and malicious content identification leverage the knowledge of multiple application classification systems and security appliances such as IDP, antivirus, and antispam. For packet anonymization, the PCAPAnon mechanism for *semantics-preserving deep packet anonymization* can protect privacy while preserving several semantic features of application fields. PCAPAnon supports the configuration that allows users to specify the fields in hundreds of application protocols to be anonymized with consistent transformation while keeping the integrity and utility of the packet traces with best efforts.

Manuscript received July 18, 2013; revised October 22, 2013 and December 24, 2013; accepted January 3, 2014. Date of publication May 9, 2014; date of current version May 30, 2016. This work was supported in part by the National Science Council and the Industrial Technology Research Institute of Taiwan and in part by ZyXEL, Inc., D-Link Corporation. Cisco Systems, Inc., and Chung-Hwa Telecom.

Y.-D. Lin and I.-W. Chen are with the Department of Computer Science and the Network Benchmarking Laboratory, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: ydlin@cs.nctu.edu.tw; iwchen@nbl.org.tw).

P.-C. Lin is with the Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi 621, Taiwan (e-mail: pclin@cs.ccu.edu.tw).

S.-H. Wang was with the Department of Computer Science and the Network Benchmarking Laboratory, National Chiao Tung University, Hsinchu 300, Taiwan. He is now with the Institute for Information Industry, Taipei 106, Taiwan (e-mail: howz.cs97g@cs.nctu.edu.tw).

Y.-C. Lai is with the Department of Information Management, National Taiwan University of Science and Technology, Taipei 106, Taiwan (e-mail: laiyc@cs.ntust.edu.tw).

Digital Object Identifier 10.1109/JSYST.2014.2301464

The usefulness of this system for assessing false positives (FPs) and false negatives (FNs) on a set of IDP systems was demonstrated in [20]. The assessment is important to the IDP system developers to optimize the accuracy of detection by reducing both FPs and FNs because the FP/FN rate limits the performance of a network security system due to the base-rate fallacy phenomenon [21]. We also compare this work with the recent ISCX intrusion detection evaluation data set [22] in Section V-B. The packet traces in that data set are generated in an emulated testbed (thus claimed to be exempt from privacy concerns) based on a profile specifying the attacks and a statistical profile characterizing real packet traces.

The contributions of this paper are summarized in four.

1) The ATC mechanism can automatically classify and extract application sessions from bulk network traffic in a real environment such as BetaSite [23], and it is essential to continuously maintain the repository of packet traces in a scalable manner.
2) A ready-to-use implementation of the PCAPAnon mechanism can anonymize sensitive information in the packet traces for hundreds of application protocols as much as possible in current practices while preserving both the semantics and length of important application fields to keep the utility for later analysis.
3) The usefulness of this system for network analysis by IDP systems has been demonstrated. It is noted that the packet traces are not only useful for intrusion detection but also for various kinds of network traffic analysis.
4) The implementation of this system is publicly available as a SourceForge project [24]. The packet traces in the evaluation are also available in [25].

The rest of this paper is organized as follows. Section II presents the background and related work. Section III describes the design and ideas of our methodology. Section IV addresses the major system implementation issues. Section V evaluates the efficiency of the packet traces in this system and compares the traces with the ISCX data set [22]. Finally, Section VI concludes this work and reports the current status.

II. Background

This paper reviews the challenges of acquiring large, diverse, updated, and well-classified packet traces for network analysis in Section II-A, the existing methods of anonymizing packet traces in Section II-B, and the methods of FP/FN assessment with a repository of packet traces in Section II-C.

A. Challenges of Acquiring Packet Traces

There are two primary options of acquiring packet traces. First, emulated packet traces can be generated in the laboratory by custom scripts or traffic generators such as Harpoon [26]. A well-known example of this approach is the DARPA-sponsored evaluation data set for intrusion detection [27], [28], but the data set was criticized for being too old to reflect contemporary network traffic [29]. Moreover, the emulation is limited in practice due to the heterogeneity and dynamics of network traffic [30].

Some researchers capture packet traces from the backbone of their affiliations rather than generate emulated traffic. This approach can satisfy the requirement of working with network traffic of *realistic* user activities, but the researchers have to make a nontrivial effort to categorize the packet traces due to the huge volume of diverse network traffic. A *scalable* method to automatically extract and classify the packet traces is therefore required. Moreover, the network operators may be unwilling to allow the researchers to acquire the traces due to privacy concerns [31]. The work in [32] summarizes the issues and practices of using network data sets.

Overall, acquiring packet traces from public repositories is more convenient than making the effort to capture packets by individual researchers, if the issues raised in Section I can be resolved. Table I compares six publicly available repositories with `PCAPLib` in terms of the packet sources, update frequency, categorization, and anonymization of packet traces. A list of links to more repositories is available at www.netresec.com/?page=PcapFiles. The contributors can annotate the categories of the packet traces submitted from them, but the categorization may be inconsistent. If the packets are captured from the backbone (e.g., those in Cooperative Association for Internet Data Analysis), the packet payloads in the traces are truncated for privacy concerns, thereby seriously hurting the utility of the traces for network analysis. The ISCX data set is exempt from privacy concerns because its packet traces are emulated. Despite its innovative contribution for dealing with privacy, we find that the traces are not well classified for different application protocols and contain only few attacks to be analyzed statistically. In comparison, the `PCAPLib` system can automatically categorize packet traces in a scalable and consistent way and preserve the semantics of application payloads during anonymization to maintain the utility for analysis as much as possible.

B. Packet Anonymization

Packet anonymization is an important means to protect the privacy of packet traces. Ideally, the private fields in both the TCP/IP headers and the application level should be well sanitized, but packet anonymization in the application payloads still face the following two critical hurdles.

1) In addition to common application protocols, there are far more from various network applications, such as P2P online games. It is nontrivial to identify the private information in the packet traces due to the diverse semantics of so many application protocols. The work in [34] presents several heuristics to infer the sensitivity of protocol fields, if the fields can be correctly parsed.
2) Since most IDP systems rely on signatures for intrusion detection, anonymization may accidentally modify an attack signature and affect the analysis. For example, an HTTP request may contain a malicious shellcode exploit as follows:

```
GET/iframe3?C00jAE12BADcCRkAAAAAAAAAAAAAA

AAAAAAAAAAAAAAAAA4AAQACB3qzB.
```

A Snort rule [10] below contains the signature of all `A`'s to detect the shellcode, but an anonymization tool can make the detection fail by hiding the target of the request

TABLE I
COMPARISON OF EXISTING REPOSITORIES OF PACKET TRACES

repository	packet source	update frequency	category	anonymization	pros	cons
Packetlife.net [3]	user submission	medium	protocol	no	focusing on packets related to routing and switching	without consistent categorization
DARPA datasets [28]	emulation	suspended	attack-free attacks	no	popular evaluation traces	outdated traces
Wireshark SampleCaptures [33]	user submission	high	protocol	no	various traces	few malicious traces
CAIDA datasets [4]	OC192 Internet backbone	high	year	IP address	prefix-preserving anonymization	payload truncation
Pcapr [2]	user submission	high	protocol	no	on-line packet display	unclear categorization
ISCX datasets [22]	emulation	low	normal with attacks	unnecessary	avoiding privacy concerns by emulation	low diversity of attacks
PCAPLib	BetaSite [23]	high	application (malicious and benign)	entire packet	providing extracted, classified and anonymizied packet traces	probably less diverse traffic due to the sole source

TABLE II
COMPARISON OF ANONYMIZATION TOOLS

tool	feature	MAC address	IPv4 address	application layer
Tcpdpriv [5]	retain class designation	no	random prefix-preserving	truncation
Tcpmkpub [6]	anonymize TCP/IP header fields	random vendor-code-preserving	random prefix-preserving cryptographic	truncation
FLAIM [7]	support a broad set of anonymzation algorithms	random partial hiding	random prefix-preserving	no
Anonym [8]	anonymize time-stamps, port numbers, length fields besides IP and MAC addresses	block black marker truncation permutation, etc.	block black marker truncation permutation, etc.	no
TraceWrangler [9]	support the PCAPng format and anonymize fields in TCP/UDP, IP, ICMP, and DHCP	randomization replacement	random replacement	Only IP and MAC addresses in DHCP
Tcpanon [15]	anonymize fields in application layers	no	random prefix-preserving	hiding specified application fields
SCRUB-tcpdump [16]	substitute specified patterns in application layer	no	random prefix-preserving permutation	hiding specified patterns
Anontool [17]	provide anonymization API	block black marker	random prefix-preserving	hiding specified patterns
Bro [18]	anonymize application-level fields	no	prefix-preserving	hiding specified application fields
PktAnon [19]	Generic anonymization primitives and flexibility	random, hashing, etc.	random, hashing, etc.	truncation in length
PCAPAnon	deep packet anonymization length-semantics-preserving	block black marker	prefix-preserving length-prefix-preserving	field transformation pattern substitution

to protect the privacy of web browsing. In general, such anonymization may alter the analysis of an IDP system.

```
alert ip EXTERNAL_NET any− > HOME_NET any
(msg : “SHELLCODE x86 inc ecx NOOP”;
 content: “AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA”;
 classtype: shellcode − detect; sid : 1394; rev : 10; ).
```

Table II compares the available tools or libraries for packet anonymization [5]–[7], [15]–[18]. Most tools support *prefix-preserving* anonymization for IP addresses [35]. That is, if two IP addresses have a common prefix of k bits, the anonymized addresses will also have a common prefix of k bits. Only Tcpanon, Anontool, SCRUB-tcpdump, and Bro can specify the information in the application payloads to be removed; the others simply drop or truncate the payloads. PCAPAnon is also covered in the table. The tools that can specify the information in the payloads to be anonymized are described below:

Tcpanon [15], written in Python, can parse and anonymize specified fields in the HTTP, FTP, SMTP, POP, and IMAP protocols but discard the payloads of other unsupported protocols. It is possible to extend the number of supported protocols by writing new protocol parsers in Python, but the effort of extending the support to a large number of application protocols in real network traffic is nontrivial. The tool is still far from practical use in a real environment.

SCRUB-tcpdump [16] can search the payload for specified patterns in regular expressions. Anontool [17] also provides a set of application programming interfaces to support pattern searching. This approach will miss some sensitive information in the payloads if a precise pattern does not exist to describe the sensitive information, such as user identifier and password. In addition to the limitation of pattern searching, both tools lack the ability to parse application-level protocols and specify the application fields to be anonymized.

Bro [11] supports functions in its policy scripts to anonymize both online and offline packet traces [18]. Despite the flexibility of using scripts, the anonymization functions have two limitations. First, as Bro works with events, a protocol field can be anonymized only if the protocol has registered events that support trace transformation. That is, a user needs to write the parsers and policy scripts of individual protocols for anonymization. Second, Bro buffers the bytes of the anonymized application content and regenerates the packets from the buffer, e.g., after the payload size exceeds the maximum transmission unit. The characteristics of packet traces, such as number of packets and packet lengths, will be changed after anonymization and may affect network analysis that relies on such statistical characteristics.

PCAPAnon, on the contrary, provides hundreds of protocol parsers based on Wireshark dissectors [36], which can be applied to all packet fields up to the application level. The support of anonymizing so many application protocols has been ready to use, and protocol parsing allows precise specification of the right fields for anonymization. To preserve the statistical characteristics of packets as much as possible, PCAPAnon replaces the field values with those of the same semantics (e.g., replacing a URL with another URL) and length in the anonymization. The preservation will benefit the methods that rely on statistical characteristics for traffic analysis, e.g., [37], and it also prevents the protocol parsers from triggering an error during the parsing process since the semantics of application fields remain the same.

In addition to the above tools, we are also aware of the work in [38], which anonymizes private information in encrypted packet payloads. It is restricted to decode encrypted shellcode by emulation and then look for possible private information such as URLs. Because a generic method of decoding encrypted network traffic is unavailable, the best practice is probably hiding the entire encrypted payloads. Decoding and anonymizing encrypted payloads in general cases are therefore beyond the scope of this paper. There are also some studies about attacks intending to de-anonymize network traces, such as [39]. The countermeasures to such attacks are not part of this paper and will be left to the future work.

C. *Methods of FP/FN Assessment*

Packet traces in real traffic can serve as the test data set for evaluating the design of IDP systems, particularly in terms of the FP and FN rates. Chen et al. designed a system of *attack session extraction* (ASE) [40] to integrate the efforts of signature analysis and development from different vendors to find out FPs and FNs in IDP systems. ASE captures real packet traces in the PCAP files and then replays them to a set of IDP systems developed by various vendors. The time on the replay tool and the IDP systems are all synchronized before session extraction. By associating the replay logs for the packets with the alarm logs on the IDP systems by comparing the key information such as the timestamps and the five-tuple fields (i.e., the source/destination IP addresses, the source/destination ports. and the protocol), ASE can identify the anchor packets that trigger alarms on the IDP systems. For example, if the alarm log records that an attack occurs from 10:10:01 to 10:10:18, the ASE will look for the packets replayed during that period and having the same five tuples as the anchor packets. The system then extracts the associated connections (having the same five tuples as those of the anchor packets) and the associated sessions (having similar payloads) that contain the anchor packets. Thus, the attack sessions can be automatically extracted from real packet traces. The system also finds potential FPs and FNs of an IDP system with a *voting* mechanism. The work in [20] looked into the causes behind possible FPs and FNs using the packet traces from the PCAPLib system and demonstrated the usefulness of this system for assessing FPs and FNs.

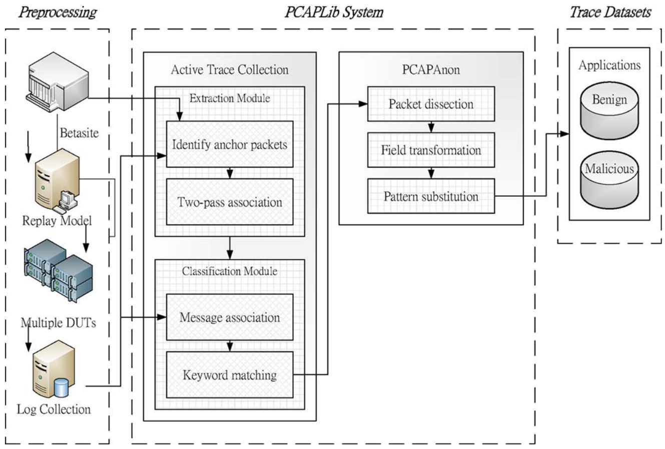


Fig. 1. PCAPLib block diagram.

III. PCAPLib Methodology

The PCAPLib system has two major components: 1) ATC, which is described in Section III-B; and 2) PCAPAnon, which is described in Section III-C.

A. *Overview of the Components in* PCAPLib

Capturing the entire traffic in an environment such as a campus can easily consume up the storage space, and searching for specific events of interest in a huge trace is time consuming; hence, leaving only the traffic associated with specific events and well classifying the packet traces are desired. As shown in Fig. 1, the ATC actively extracts both benign and malicious packet traces from real traffic in National Chiao Tung University (NCTU) BetaSite (see [23] or a brief introduction at speed.cis.nctu.edu.tw/~ydlin/Betasite.html), i.e., an operational network involving voluntary students at a large university campus. The network traffic is from the daily network usage by the students, and the volume in the BetaSite is roughly 100 GB/h. The traffic data are processed in an offline fashion. The packet traces from the BetaSite are stored in a repository of disk arrays first and are later retrieved by the ATC for further processing. The offline processing (i.e., extracting, classifying, and anonymizing) rate is roughly 60 GB/h in our experience. The rate is acceptable because we do not need to collect the packet traces from all of the captured traffic. Randomly picking

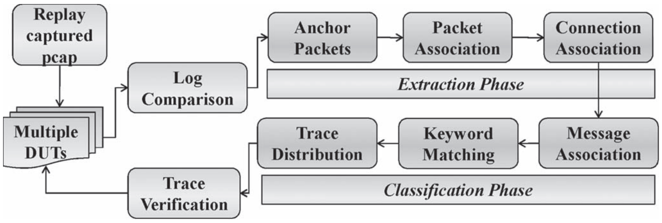


Fig. 2. ATC system flow.

the traffic captured over a period of time, for example, half a day, and then deriving the packet traces offline from the captured traffic are fine because the amount of captured traffic is huge enough.

Furthermore, packet anonymization protects privacy from leakage in the trace sharing. The PCAPAnon parses application protocols and anonymizes sensitive fields in the collected traces. Both ATC and PCAPAnon are essential components in the PCAPLib system to provide high-quality packet traces that meet the requirements. They are described in detail in the following sections.

B. ATC

The quality of many packet repositories relies on the active involvement and frequent update from users. In contrast, the ATC mechanism can *automatically* capture, extract, and classify large-scale packet traces from real traffic. Fig. 2 shows the ATC system flow. The collection involves two phases. In the first phase, a traffic replay tool (e.g., *tcpreplay*) replays captured raw traffic to multiple *devices under test* (DUTs) to leverage their domain knowledge. A DUT will trigger a log upon detecting specific behavior in the traffic or when the application protocols of interest appear (see Section V). The specific behavior may be web attacks (e.g., SQL injection), denial of service, spamming, and so on, depending on the detection results of the DUTs. The users can configure the DUTs to specify which application protocols are of interest. The logs contain primarily the source/destination IP addresses, the source/destination ports, the protocol, the name of behavior or protocol, and optionally the payloads with attacks in some DUTs. The exact log format may slightly vary with the DUTs. The packets that do not trigger any logs will be replayed again to avoid FNs. If they still do not trigger logs, they will be discarded without further processing. The ATC then locates the anchor packets that trigger the logs by comparing the timestamps of the packets and the five-tuple fields, and processes packet and connection associations to extract each specific session into the packet traces. The extraction is similar to that of ASE in [40].

In the second phase, we use supervised classification to categorize the extracted packet traces. The ATC associates the sessions according to the log messages and classifies the traces into different categories by matching the logs with representative keywords. The ATC also verifies the classification to ensure that the packet traces are useful. The verification replays the categorized sessions to the particular DUT(s) and see whether they can trigger the same logs again. If the logs are the same, the ATC stores the trace in that session into the database; otherwise, the trace is invalid. The ATC also judges whether the traces are benign or malicious with *majority voting* based on the detection results from a set of security devices (IDP, antivirus, etc.).

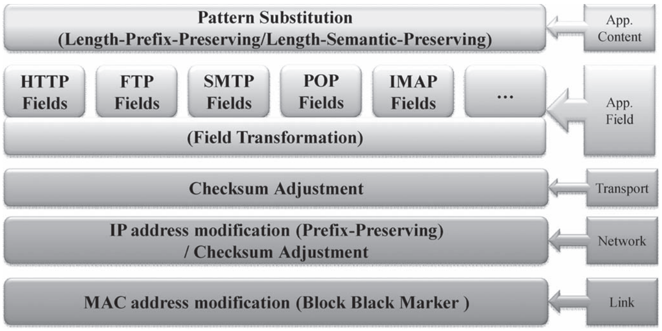

Pattern Substitution
(Length-Prefix-Preserving/Length-Semantic-Preserving)
App. Content
HTTP Fields
FTP Fields
SMTP Fields
POP Fields
IMAP Fields
...
(Field Transformation)
App. Field
Checksum Adjustment
Transport
IP address modification (Prefix-Preserving) / Checksum Adjustment
Network
MAC address modification (Block Black Marker)
Link

Fig. 3. PCAPAnon system.

C. PCAPAnon: Deep Packet Anonymization

The ATC provides well-classified packet traces, but we still face the problem of privacy leakage when releasing the traces. PCAPAnon supports a precise method for privacy protection of packet traces. The method involves two phases: 1) *trace parsing* to specify which information in the traces should be hidden; and 2) *identity substitution* to choose how the data elements are anonymized. Fig. 3 presents the PCAPAnon system for deep packet anonymization, which anonymizes each layer with different policies. The system provides a large set of protocol parsers and substitutes the identities with those of the same length and data type to maintain the semantics of application protocols. The two phases are detailed as follows.

1) Trace Parsing: It is exhausted to implement precise parsing for the semantics of possible application fields in real packet traces due to the large number of application protocols. We leverage Wireshark (www.wireshark.org), a network packet analyzer, for its hundreds of *protocol dissectors* (www.wireshark.org/docs/dfref) registered as plug-ins to parse the traces for the protocol fields. The Wireshark code is modified to support anonymizing the specified application fields from the command options or custom scripts, based on subjective judgment on sensitive fields or analysis from the heuristics in [34]. For example, the options “−Tfield −ehttp.host” mean anonymizing the content in the HTTP HOST field. The specification is particularly useful when an application field is private (e.g., a password) yet difficult to specify the content with a pattern.

To avoid missing sensitive application fields due to careless specification, PCAPAnon also supports *pattern substitution* with regular expression (RegExp) matching to seek and match-sensitive identities such as IP addresses, mail addresses, and URLs. The substitution is also useful if the identities are from an unknown application protocol or not precisely identified in protocol parsing.

2) Identity Substitution: PCAPAnon provides several anonymization functions, including *Block Black Marker* for MAC address, *Prefix-Preserving* and *Length-Prefix-Preserving* (LPP) for IP address, *Length-Semantics-Preserving* (LSP) for pattern substitution with RegExp matching, *Field*

Transformation for protocol fields, and *Checksum Adjustment* for all network and transport protocols,[1] thereby supporting adequate functionality for various identities. The functions are described as follows.

1) *Block Black Marker* sets the bits in a field to all zeros.
2) *Prefix-Preserving* function for IP addresses (particularly in the network layer) has been common in existing anonymization tools since the work in [35].
3) LPP preserves the length of an IP address represented in ASCII, in addition to the attempt of prefix-preserving anonymization. The details will be described in Section IV-B.
4) LSP searches the payload for an identity such as a mail address, URL, or domain name in the payload and then substitutes another mail address, URL, or domain name of the same length for that identity, thereby preserving the semantics of the identity.
5) *Field Transformation* fills a field identified in protocol parsing with a repeating pattern (e.g., "`XXXXX`"), which can be an integer or a string.
6) *Checksum Adjustment* keeps the checksum valid because some network appliances will drop the packets with invalid checksums.

This paper attempts to preserve the characteristics of anonymized packet traces as many as possible. For example, this work substitutes an application field of the same type and length for the original one in LPP and LSP. This approach has three main benefits.

1) Preserving the number and lengths of the original packets allows anomaly detection and traffic classification based on statistical characteristics [37], [41] to still work after anonymization.
2) An IDP system can parse the payloads for protocol semantics as usual; otherwise, parsing errors (e.g., finding a malformed mail address) will occur.
3) Keeping the lengths of the original application fields facilitates the design of anonymization tools. Otherwise, fields such as the sequence/acknowledgement numbers in the TCP header, as well as those in an application protocol, e.g., the HTTP `Content − Length` field, should be adjusted accordingly, or traffic analysis that examines the sequence/acknowledge numbers (e.g., packet reassembly in an IDP system) will result in errors.

With the lengths preserved, the above values do not need to be recalculated after anonymization.

IV. Implementation Issues of PCAPLib

This section details the implementation of the `PCAPLib`, which is built on a 64-bit Linux system. Section IV-A will cover the extraction and classification phases in the ATC. Section IV-B will cover packet dissection, field transformation, and pattern substitution in the PCAPAnon.

TABLE III
Classification of Traces and Their Representative Keywords

category	keywords within the DUT logs
Web	HTTP
Email	POP3, SMTP, IMAP
FileTransfer	FTP, SMB, TFTP
RemoteAccess	Telnet, SSH, RDP, VNC
Encryption	SSL, FTPS, HTTPS
Chat	IRC, ICQ, Yahoo Messenger, MSN, AIM, Skype, Google talk
FileSharing	Bittorrent, eDonkey, Gnutella, Pando, SoulSeek, Winny, Xunlei
Streaming	PPLive, QuickTime, Octoshape, Orb, Slingbox
VoIP	SIP
Network	NetBIOS, DNS, SNMP, Socks, STUN

A. Implementation of ATC

The ATC involves two phases. In the *extraction phase*, the ATC extracts the sessions associated with the logs generated from the DUTs. In the *classification phase*, the ATC classifies the extracted packet traces into each application category with *keyword matching* against the logs.

1) Extraction Phase: This phase consists of three-pass scanning throughout the packet traces: 1) finding an *anchor packet* that generates a DUT log; 2) associating the other packets in the same TCP or User Datagram Protocol (UDP) connection[2] with the anchor packet; and 3) associating the other connections with this anchor connection into a session. The implementation involves an *alarm log table* (ALT) to record the logs from the DUTs and a *replay log table* (RLT) to record the time when *tcpreplay* sends each packet. The identification of the anchor packets correlates the five-tuple and time information in the ALT with those in the RLT. Since the five tuples may be unable to uniquely identify a packet, the log time in the ALT and the packet time in the RLT are correlated to set up a time frame for narrowing down the searching scope and correctly identifying the anchor packets. The packet association looks for all the other packets sharing the same five tuples with the anchor packet and groups them as the anchor connection. The connection association groups the connections sharing similar payloads into a session (see [40] for details of session extraction). The ATC then stores the packet traces in each individual session, as well as related information such as the logs, into the database for later classification.

2) Classification Phase: The repository is intended to provide packet traces in taxonomy for users to select the desirable category of traces. Table III lists ten categories for the classification, each associated with a few representative keywords. The classification matches the keywords in the DUT logs and finds out the associated packet traces. This phase then separates the packet traces into benign or malicious ones based on the detection results of multiple security devices. If a packet trace triggers a log on most of the security devices, it is considered malicious with high possibility. Therefore, the classification is 2-D: One is based on applications, and the other is based on the security events involved.

[1] Note that the frame check sequence in the data-link layer is not included in the adjustment since it is not part of the PCAP format.

[2] We mean having the same five tuples, although UDP is connectionless.

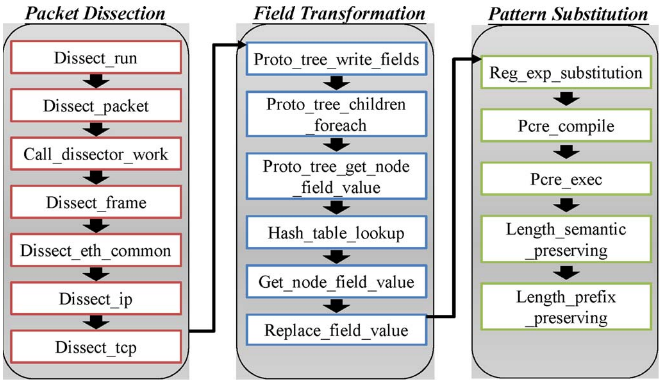
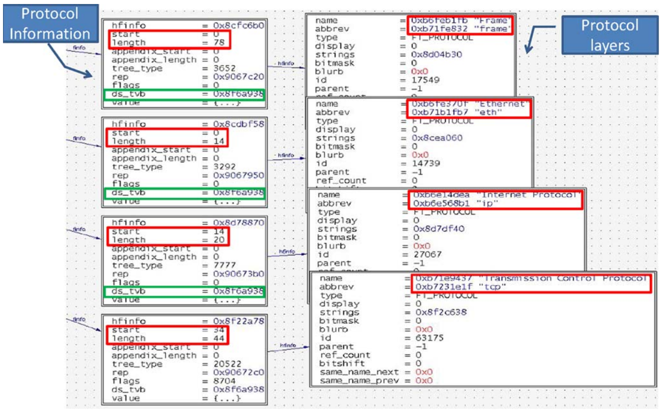


Fig. 4. PCAPAnon packet processing.



```
hfinfo            = 0x8cfc6b0
start             = 0
length            = 78
appendix_start    = 0
appendix_length   = 0
tree_type         = 3652
rep               = 0x9067c20
flags             = 0
ds_tvb            = 0x8f6a938
value             = {...}

hfinfo            = 0x8cdbf58
start             = 0
length            = 14
appendix_start    = 0
appendix_length   = 0
tree_type         = 3292
rep               = 0x9067950
flags             = 0
ds_tvb            = 0x8f6a938
value             = {...}

hfinfo            = 0x8d78870
start             = 14
length            = 20
appendix_start    = 0
appendix_length   = 0
tree_type         = 7777
rep               = 0x90673b0
flags             = 0
ds_tvb            = 0x8f6a938
value             = {...}

hfinfo            = 0x8f22a78
start             = 34
length            = 44
appendix_start    = 0
appendix_length   = 0
tree_type         = 20522
rep               = 0x90672c0
flags             = 8704
ds_tvb            = 0x8f6a938
value             = {...}
```

```
name          = 0xb6feb1fb "Frame"
abbrev        = 0xb71fe832 "frame"
type          = FT_PROTOCOL
display       = 0
strings       = 0x8d04b30
bitmask       = 0
blurb         = 0x0
id            = 17549
parent        = -1

name          = 0xb6fe370f "Ethernet"
abbrev        = 0xb71b1fb7 "eth"
type          = FT_PROTOCOL
display       = 0
strings       = 0x8cea060
bitmask       = 0
blurb         = 0x0
id            = 14739
parent        = -1
ref_count     = 0

name          = 0xb6e14dea "Internet Protocol"
abbrev        = 0xb6e568b1 "ip"
type          = FT_PROTOCOL
display       = 0
strings       = 0x8d7df40
bitmask       = 0
blurb         = 0x0
id            = 27067
parent        = -1

name            = 0xb71e9437 "Transmission Control Protocol"
abbrev          = 0xb7231e1f "tcp"
type            = FT_PROTOCOL
display         = 0
strings         = 0x8f2c638
bitmask         = 0
blurb           = 0x0
id              = 63175
parent          = -1
ref_count       = 0
bitshift        = 0
same_name_next  = 0x0
same_name_prev  = 0x0
```

Fig. 5. Protocol tree for dissection.

B. Implementation of the PCAPAnon System

PCAPAnon provides a configurable policy on three levels of anonymization: 1) TCP/IP header; 2) regular-expression matching; and 3) application-level protocol dissection. Most anonymization tools support the first; hence, we do not repeat here. Fig. 4 illustrates the three stages in packet processing, i.e., 1) packet dissection, 2) field transformation, and 3) pattern substitution, to realize the latter two levels. They are explained as follows.

1) Packet Dissection: Wireshark provides more than 800 protocol dissectors to handle packet parsing for various application protocols. Each dissector decodes its part of the protocol and then hands off decoding the encapsulated protocol to subsequent dissectors. Wireshark uses a data structure called *protocol tree* to keep the relationship between protocol layers and to handle each layer properly. Fig. 5 illustrates a simple protocol tree for parsing up to the TCP layer. Parsing starts with a frame dissector, which dissects the packet details (e.g., timestamps) of the capture file, passes the data to an Ethernet frame dissector that decodes the Ethernet header, and then passes the payload to the next dissector (e.g., IP), and so on. Each dissector decodes the information in the layer it is responsible for.

Moreover, Wireshark uses protocol signatures to identify an encapsulated protocol in the sibling nodes. Each protocol can register its specific signature for Wireshark to distinguish the children nodes from another. For example, registering the TCP port field "`tcp.port` = 21" is considered as a signature of FTP. Packets with this signature will be passed to the FTP dissector. The system is therefore highly flexible for expansion.

```
1) IPv4 address:
   [0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}
2) Mail address:
   [a-z0-9!#$%&′*+/=?^_`{|}˜-]+(?:\.[a-z0-9!#$%&′
   *+/=?^_`{|}˜-]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z
   0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?
3) URL:
   (?:(?:(?:http|ftp|gopher|telnet|news)://)(?:w
   {3}\.)?(?:[a-zA-Z0-9/;\?&=:\-_\$\+!\*′\(\|\\˜
   \[\]#%\.])+)
4) Domain name:
   ([a-zA-Z0-9]([a-zA-Z0-9\\-]{0,61}[a-zA-Z0-9])?
   \\.)+[a-zA-Z]{2,6}
```

Fig. 6. Patterns for private identities.

2) Field Transformation: This work leverages the packet dissection in Wireshark to correctly parse and locate the fields of hundreds of application protocols to be anonymized; hence, the number of fields that can be specified is much more than those of existing anonymization tools. A protocol field list is provided (www.wireshark.org/docs/dfref) for users to specify the identities to be anonymized, e.g., user identifier, password, or authentication key, which usually involve private information not easily specified as a pattern. The configuration can be also written in a script. In Fig. 4, the `Proto_tree_get_node_field_value` function returns the node field value of the protocol tree. The `Hash_table_lookup` function then locates the fields to be anonymized, and the subsequent functions replace the field values with a pattern of equal length (e.g., from `PASS 1234` to `PASS XXXX`).

3) Pattern Substitution: Anonymization also searches the packet payloads for private identities with the patterns in regular expressions (as listed in Fig. 6) to avoid missing sensitive application fields. If a pattern matches a specific identity in the payload, the LSP function substitutes another identity of the same length and semantics for that identity, but if the match is an IP address in ASCII, the LPP function is applied. Thus, the number and lengths of packets are preserved after anonymization. Preservation is important if the packet traces are to be analyzed with such statistical characteristics.

Fig. 7 presents the pseudocode for LPP and LSP. Suppose that the ASCII form of an IP address is $B_1 \cdot B_2 \cdot B_3 \cdot B_4$. The LPP follows four steps.

1) The number of digits in the four blocks is recorded in D_i for $i = 1, 2, 3, 4$.
2) DES_ECB encrypts each block B_i with the DES algorithm in the ECB mode [42]. The encryption guarantees that the blocks in an IP address are consistently anonymized if they appear multiple times in the packet traces.
3) MSB_PAD sets the most significant bit in each B_i with 1 to make every block a three-digit value.
4) The last step performs $B_i = B_i \%(10^{D_i})$ to restore each block to the original length in ASCII.

Note that the current implementation of LPP is prefix preserving only if the prefix length is a multiple of 8 bits, which is a common case. Prefix preserving for an arbitrary prefix length is feasible only if the IP address is represented in binary so far. Achieving full LPP for an arbitrary prefix length is left

```
if pattern is IP address then {Length-Prefix-Preserving}
    for $i \leftarrow 1$ to 4 do {Suppose IP address is represented as $B_1.B_2.B_3.B_4$ in ASCII.}
        Covert $B_i$ from ASCII to binary;
        $D_i \leftarrow Digit[B_i]$; {record number of digits}
        $B_i \leftarrow DES\_ECB[B_i]$; {encrypt block}
        $B_i \leftarrow MSB\_PAD[B_i]$; {extend digits};
        $B_i \leftarrow B_i\%(10^{D_i})$; {recover number of digits}
        Convert $B_i$ back to ASCII;
    end for
    Anonymize the IP address with new $B_i$;
else if pattern is mail address then {Length-Semantics-Preserving}
    $template\_pattern \leftarrow$ "z@c.b.a";
    {Assume email address is in the form $v@w.x.y$ without loss of generality, where $v, w, x, y$ are strings. The following replacements are based on $template\_pattern$.}
    Replace $v$ with $|v|$ z's;
    Replace $w$ with $|w|$ c's;
    Replace $x$ with $|x|$ b's;
    Replace $y$ with $|y|$ a's;
    {Example: john@nctu.edu.tw becomes zzzz@cccc.bbb.aa.}
else if pattern is URL then {Length-Semantics-Preserving}
    $template\_pattern \leftarrow$ "http://d.c.b.a";
    {Assume URL is in the form http://$v.w.x.y$ without loss of generality, where $v, w, x, y$ are strings. The following replacements are based on $template\_pattern$.}
    Replace $v$ with $|v|$ d's;
    Replace $w$ with $|w|$ c's;
    Replace $x$ with $|x|$ b's;
    Replace $y$ with $|y|$ a's;
    {Example: http://www.nctu.edu.tw becomes http://ddd.cccc.bbb.aa.}
else if pattern is domain name then {Length-Semantics-Preserving}
    replace with a domain name of the same length;
    {similar to the case of URL; omit the pseudo code for brevity}
end if
```

Fig. 7. Pseudocode of LPP and LSP.

to future work. In LSP, PCAPAnon derives a string from the template pattern of the same type and expands its length to that of the matched pattern (see the examples in Fig. 7). The matched pattern is then anonymized with the derived string.

V. Evaluation and Observation

The real traffic in the evaluation was captured from NCTU BetaSite during the period from October 1, 2009 to February 1, 2010. Eight DUTs are involved in the classification, namely, BroadWeb, Cisco, D-Link, Fortinet, McAfee, TrendMicro, TippingPoint, and ZyXEL security devices. It is noted that the library of traces in PCAPLib depends on the DUTs, but since the eight DUTs are representative ones on the market and we could always extend the set of DUTs, the dependence on a particular DUT is reduced.

Both the ATC and the PCAPAnon are evaluated in this paper, and the evaluation for assessing FPs/FNs is in [20]. First, the ATC classifies 318 packet traces[3] randomly sampled from the much larger repository in PCAPLib. The total size of the classified traces is 696.90 MB and should be sufficiently large for the evaluation. We could have used much larger traces, but using much larger ones would be too costly for manual inspection to find the ground truth in the evaluation. Table IV presents the numbers of the classified traces in a 10*5 classification matrix, each entry representing distinct application behavior. The number of connections in and the total size of each packet trace are also presented. Second, the privacy and utility of anonymization tools, as well as the efficiency of three different anonymization policies supported by PCAPAnon, are also evaluated. Table V summarizes the ten application categories and the application names.

A. Privacy, Utility, and Efficiency of Anonymization Policies

The privacy and utility for anonymization are defined in the evaluation. Privacy is evaluated with the percentage of sensitive fields in the packet traces that have been anonymized precisely, whereas utility is evaluated with the percentage of malicious packet traces after anonymization that can be still detected by the DUTs. We use the open-source IDP system, Snort 2.8.5, as the DUT for easily investigating and interpreting the results by comparing Snort's signatures with the packet traces.

The first step in the evaluation replays the traces collected by the ATC to Snort and collects the logs. Next, the evaluation anonymizes the packet traces and replays them again to calculate the four metrics defining privacy and utility.

1) TP_{field} denotes the set of sensitive fields that are really anonymized.
2) FN_{field} denotes the set of sensitive fields that are not anonymized.
3) TP_{trace} denotes the set of traces with malicious signatures that can be still detected by the DUT after anonymization.
4) FN_{trace} denotes the set of traces with malicious signatures that cannot be detected by the DUT after anonymization.

Privacy and utility are defined as follows:

$$\text{Privacy} = \frac{|TP_{\text{field}}|}{|TP_{\text{field}}| + |FN_{\text{field}}|} * 100\% \tag{1}$$

$$\text{Utility} = \frac{|TP_{\text{trace}}|}{|TP_{\text{trace}}| + |FN_{\text{trace}}|} * 100\%. \tag{2}$$

Figs. 8 and 9 compare the privacy and utility of PCAPAnon with two other anonymization tools, i.e., *anontool* and *tcpanon*. We choose the two tools for comparison because they represent typical examples of payload anonymization by pattern substitution and field transformation. Table VI lists the sensitive identities in the four protocols HTTP, FTP, POP3, and SMTP for a fair comparison because *tcpanon* supports these parsers. Fig. 8 presents the results. We have the following two observations.

1) *anontool* provides only pattern substitution in the payloads, leading to serious privacy leak.
2) *tcpanon* uses customized parsing to hide more identities than *anontool*. However, it does not design the FTP protocol parser correctly and misses some sensitive identities such as PORT and STOR. Even if we ignore its parsing error, PCAPAnon still performs slightly better than *tcpanon* for the other protocols in terms of privacy.

Like the F_1 score, which is a single measure of a test's accuracy,[4] we define a similar measure for the *efficiency* of

[3] The evaluated packet traces are publicly accessible at [25].

[4] F_1 score is the harmonic mean of *precision* and *recall*.

TABLE IV
TRACE CLASSIFICATION MATRIX (NUMBER OF PACKET TRACES/NUMBER OF CONNECTIONS/TOTAL SIZE)

category	benign	attack	virus	spam	total
Web	74/77/195.03MB	49/57/84.12MB	0	2/2/3.64K	125/136/279.15MB
Email	12/12/20.54MB	6/7/87.68KB	0	3/3/544.22KB	21/22/21.17MB
File transfer	36/46/49.02MB	15/17/246.00MB	0	0	51/63/295.02MB
Remote Access	10/10/4.73MB	5/10/18.79MB	1/1/6.92KB	0	16/21/23.53MB
Encryption	6/6/37.11MB	6/6/278.38KB	1/1/3.08KB	0	13/13/37.39MB
Chat	16/19/4.48MB	5/4/49.20KB	0	0	21/23/4.53MB
File Sharing	16/24/2.23MB	0	0	0	16/24/2.23MB
Streaming	6/6/1.05MB	0	0	0	6/6/1.05MB
VoIP	2/2/4.37KB	2/2/5.93KB	0	0	4/4/10.30KB
Network	32/70/172.64KB	13/13/32.65MB	0	0	45/83/32.82MB

TABLE V
CATEGORIES OF APPLICATIONS

category	applications (number of packet traces/number of connections/total size)
Web	HTTP(125/136/279.15MB)
Email	POP3(5/6/45.56KB), SMTP(11/11/21.09MB), IMAP(5/5/29.65KB)
File transfer	FTP(28/37/48.96MB), SMB(22/24/246.06MB), TFTP(1/2/1.06KB)
Remote Access	Telnet(6/11/109.61KB), SSH(4/4/18.45MB), RDP(4/4/693.00K), VNC(2/2/4.27MB)
Encryption	SSL(11/11/37.18MB), FTPS(1/1/204.42KB), HTTPS(1/1/14.63KB)
Chat	IRC(7/7/34.54KB), ICQ(4/6/40.67KB), Yahoo Messenger(4/4/4.35MB), MSN(3/3/94.18KB), AIM(1/1/10.83KB), Skype(1/1/2.27KB), Google talk(1/1/3.20KB)
File Sharing	BitTorrent(2/2/1.78KB), eDonkey(1/1/2.31KB), Gnutella(1/1/1.13KB), Pando(1/1/6.34KB), SoulSeek(1/6/1.68KB), Winny(1/1/2.17MB), Xunlei(1/1/20.12KB), Azureus(1/1/0.47KB), BearShare(1/1/1.30KB), Groove(1/1/2.32KB), LimeWire(1/1/0.11KB), Pruna(1/1/2.44KB), Sharelike(1/3/8.41KB), SoftEther(1/1/9.17KB), TeamViewer(1/1/3.18KB)
Streaming	PPLive(2/2/87.65KB), QuickTime(1/1/917.23KB), Octoshape(1/1/1.52KB), Orb(1/1/11.55KB), Slingbox(1/1/36.79KB)
VoIP	SIP(4/4/10.30KB)
Network	NetBIOS(21/21/32.72MB), DNS(19/55/100.71KB), SNMP(3/5/1.75KB), Socks(1/1/0.18KB), STUN(1/1/0.42KB)

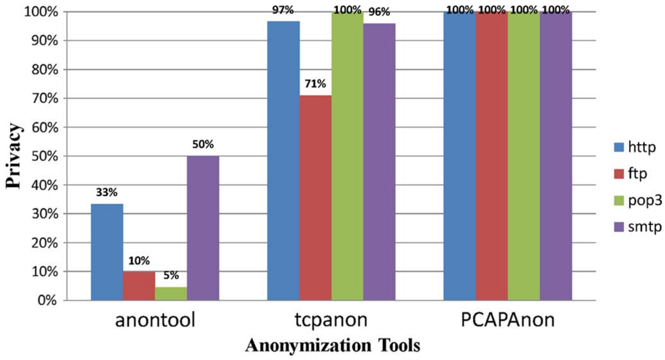


Fig. 8. Privacy of the anonymization tools.

anonymization to consider both privacy and utility simultaneously. The measure takes the *harmonic mean* of privacy and utility in the following equation:

$$\text{Efficiency} = \frac{2}{\frac{1}{\text{Privacy}} + \frac{1}{\text{Utility}}} * 100\%. \quad (3)$$

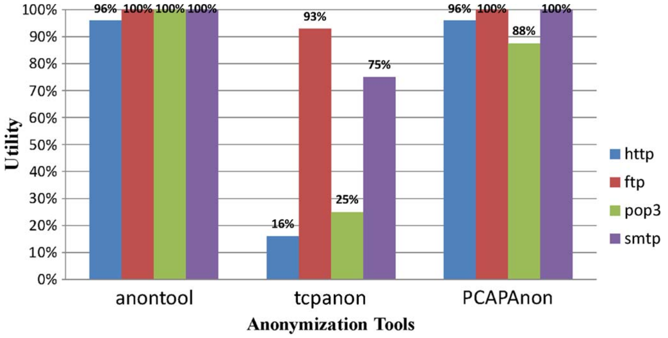


Fig. 9. Utility of the anonymization tools.

TABLE VI
SENSITIVE IDENTITIES FOR THE PROTOCOLS IN THE PRIVACY EVALUATION

HTTP	FTP	POP	SMTP
Proxy_authentication	USER	USER	HELO
Proxy_authorization	PASS	PASS	MAIL FROM
WWW_authenticate	PORT	Reply.+OK	RCPT TO:
Content	RETR	Mail address	DATA
Authorization	STOR	IP address Reply	Reply.220
Set_cookie	Reply.150	URL address	Reply.250
Referer	Reply.227		Mail address
HOST	Reply.230		IP address
Cookie	Reply.331		URL address
Mail address	Mail address		
IP address	IP address		
URL address	URL address		

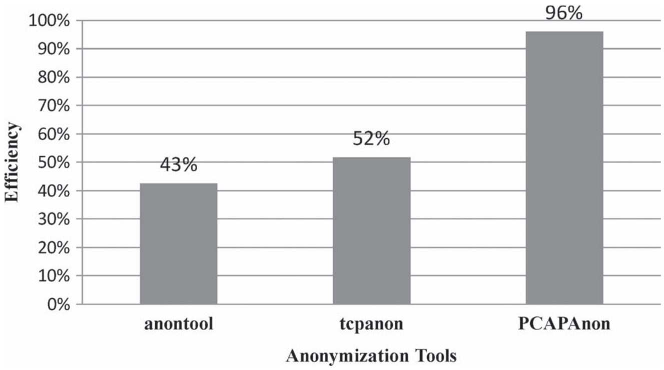


Fig. 10. Efficiency of the anonymization tools.

Fig. 10 presents the efficiency of the anonymization tools. The efficiency of *anontool* and *tcpanon* is 43% and 52%, respectively, whereas that of PCAPAnon is up to 96%, meaning PCAPAnon can maintain both privacy and utility well.

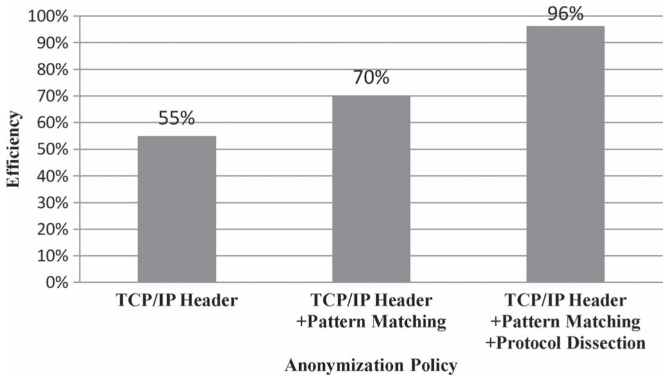


Fig. 11. Efficiency of the three-level anonymization policies.

Fig. 11 presents the efficiency of PCAPAnon's three-level anonymization policies. Here, are three observations.

1) Hiding MAC/IP addresses in the L2/L3/L4 headers is insufficient to protect sensitive identities in the payloads.
2) Pattern matching allows to replace email addresses, IP addresses, and URLs within the payloads, but if an identity is undefined in the patterns, it will be exposed without protection.
3) Protocol dissection can precisely parse application payloads and anonymize the field values. Combining the three policies can gain good efficiency up to 96%.

We also consider the comparison with binary protocols. The comparison is harder than that with the preceding four human-readable ASCII protocols because neither *tcpanon* nor *anontool* has the parsers of the binary protocols, and it is difficult to specify sensitive patterns for binary protocols in *anontool*, except for a few cases such as the domain names appearing in the DNS queries. Even an IP address in a DNS response cannot be specified as a pattern because it is represented in a 4-byte binary value. If the payloads are truncated by default, the utility will be seriously affected since the DUT relies on deep packet inspection for malicious signatures in the packet payloads. Thus, rather than compare PCAPAnon with the other two tools, we study the privacy and utility of the packet traces anonymized by PCAPAnon.

We select four binary protocols, i.e., SSH, Telnet, DNS, and SMB/NBT, in this paper. For privacy, it is feasible to study the syntax of each protocol and specify the right fields to be anonymized. The anonymized fields include `ssh.protocol` in SSH, `telnet.data` in Telnet, five fields in DNS such as `dns.qry.name` and `dns.resp.name`, and 15 fields in SMB/NBT such as `smb.security_blob` and `smb.sm.password`. We also perform pattern substitution to ensure that no private information will be inadvertently leaked. After manual inspection of the packet traces in the extracted sessions, we see that no sensitive information is leaked. The utility values for SSH, Telnet, DNS, and SMB/NBT are 50%, 100%, 100%, and 72.5%, respectively.

This paper also selects malicious HTTP traces to study which fields will affect the utility most, as many of them are in the collection. Fig. 12 presents the impact of the anonymized HTTP header fields on the utility. It is observed that most malicious signatures are embedded in the `host`, `cookie`, and `request URI` fields. If these fields are anonymized, the traces will be unlikely to trigger logs. Fig. 13 illustrates an example of FN that occurs after anonymizing the `request URI` field. The original packet payload includes a request for `/etc/passwd` in the URI. If the field is anonymized, the "`/etc/passwd`" signature in the payload will be lost, i.e., the alert of "`WEB – MISC/etc/passwd`" will not be triggered anymore.

B. Comparison With the ISCX Data Sets

We also compare `PCAPLib` with the recent ISCX data sets [22], which consist of packet traces appearing from June 11, 2010 to June 17, 2010, and are available upon requests. That work claims to be free from privacy issues because the data sets are emulated from a controlled testbed environment based on the profiles describing the attack scenario and the statistical distribution from real networks.

Despite the innovation of that work, we find that it still has a few weaknesses when compared with this paper. First, the packet traces are not well classified in terms of application protocols. A user is unable to easily get the packet traces of a desired application protocol on demand and may have to seek it from the huge packet traces (up to several gigabytes for each trace). `PCAPLib` can automatically classify and extract application sessions for users to search and download. This feature is essential to a repository of packet traces.

Second, the attacks present in the ISCX data sets should be manually specified with an exploit language. In other words, it takes human efforts to write the scripts to describe attack scenarios. Therefore, the provider of packet traces has to study the attacks in real network traffic before being able to precisely describe them. In contrast, this paper can automatically extract packet traces with attacks and is more scalable.

Third, we find the data sets sacrifice the diversity of activities in real network traffic. For example, in the packet traces of web traffic in June 12, the data set contains mostly the browsing content on an online bookstore, probably sampled from a real user's browsing and examined beforehand to ensure that no privacy is inside. The SMTP traffic on that day, which contains repeated deliveries with the mail content of the sentences from the Exodus, is also not diverse in content. The activities in a real environment are obviously much richer and more diverse than those in the emulated data sets. To be numerically specific, there are 318 packet traces out of 47 application protocols (see Table V) from `PCAPLib` in the evaluation alone, but the ISCX data sets consist of only six traces from few protocols (HTTP, IRC, SSH, etc.). Moreover, the ISCX data sets lack a mechanism to automatically extract packet traces from real traffic, and without active maintenance, the data sets may be easily outdated. It is a fundamental limitation of emulated traffic. We argue that *extraction from real network traffic* is valuable for network analysis to study Internet activities.

It is noted that we are unable to numerically compare the `PCAPLib` with the ISCX data sets in terms of privacy and utility defined in Section V-A since the original packet traces from which the emulated traffic was derived are unavailable. We speculate that the privacy of the ISCX data sets should be excellent, if the packet payloads in them are closely examined

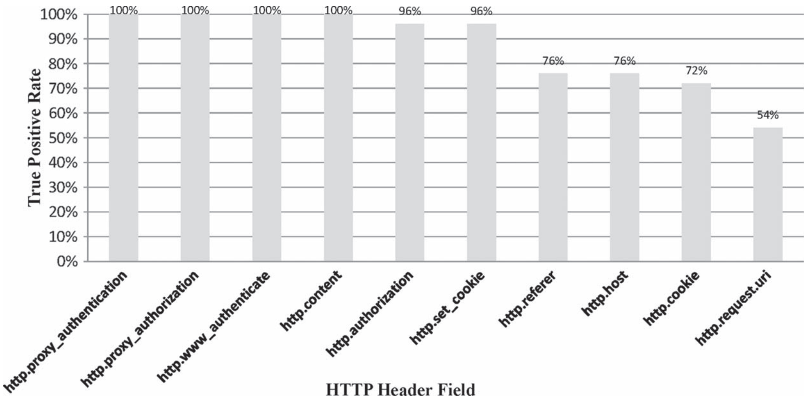


Fig. 12. Anonymization impact of HTTP header fields.

```
[Before Anonymization]
GET //lists/admin/index.php?_SERVER[ConfigFile]=
../../../../../../../../../../../../../../../../
../../../../../../../etc/passwd HTTP/1.1

[After Anonymization]
GET XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX HTTP/1.1

[Snort rule]
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS
$HTTP_PORTS (msg:"WEB-MISC /etc/passwd"; flow:
to_server,established; content:"/etc/passwd";
nocase; metadata:service http; classtype:attempted-
recon; sid:1122; rev:6;)
```

Fig. 13. FN in an anonymized HTTP trace.

and sanitized beforehand. Nonetheless, the utility for network analysis will be limited because the diversity of network activities in them has been greatly reduced.

VI. Conclusion and Future Work

This paper has presented the PCAPLib system to automatically extract and classify application sessions from bulk traffic in a scalable manner. The packet traces can be flexibly anonymized for protecting privacy and used for network analysis of various purposes not just for network security study. The system has been proven effective by running it in an operational network, the BetaSite, and its source code is available as a SourceForge project at [24].

The PCAPLib system has been operating for a long time to continuously extract the application sessions from the BetaSite. The packet traces have been provided to cooperating organizations and are planned to be made public completely in the future. We are also working on another work toward improving packet anonymization [34]. That work will be effectively integrated with the PCAPLib and ease locating sensitive application fields to be anonymized.

Acknowledgment

The authors would like to thank the staff at Network Benchmarking Lab (NBL) for their assistance in the experimental works.

References

[1] P. Porras and V. Shmatikov, “Large-scale collection and sanitization of network security data: Risks and challenges,” in *Proc. Workshop NSPW*, 2006, pp. 57–64.

[2] PCAPR collaborative network forensics. [Online]. Available: http://www.pcapr.net/forensics

[3] Packetlife repository. [Online]. Available: http://www.packetlife.net/captures

[4] CAIDA traces dataset. [Online]. Available: http://www.caida.org/home

[5] G. Minshall, TCPDPRIV: Program for eliminating confidential information from traces. [Online]. Available: http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html

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

[7] K. Lakkaraju and A. Slagell, “Evaluating the utility of anonymized network traces for intrusion detection,” in *Proc. 4th Intl. Conf. SecureComm*, 2008, p. 17.

[8] T. Farah and L. Trajkovi, “Anonym: A tool for anonymization of the Internet traffic,” in *Proc. IEEE Int. CYBCONF*, Jun. 2013, pp. 261–266.

[9] TraceWrangler. [Online]. Available: http://www.tracewrangler.com

[10] M. Roesch, “Snort: Lightweight intrusion detection for networks,” in *Proc. 13th USENIX Conf. System Admin. LISA*, Nov. 1999, pp. 229–238.

[11] V. Paxson, “Bro: A system for detecting network intruders in real-time,” *Comput. Netw.*, vol. 31, no. 23/24, pp. 2435–2463, Dec. 1999.

[12] B. Park, Y. J. Won, M. S. Kim, and J. W. Hong, “Towards automated application signature generation for traffic identification,” in *Proc. IEEE/IFIP NOMS*, Apr. 2008, pp. 160–167.

[13] K. Wang, G. Cretu, and S. Stolfo, “Anomalous payload-based worm detection and signature generation,” in *Proc. 8th Int. Symp. RAID*, 2005, pp. 227–246.

[14] W. Lu, M. Tavallaee, G. Rammidi, and A. A. Ghorbani, “BotCop: An online botnet traffic classifier,” in *Proc. 7th Annu. Commun. Netw. Services Res. Conf.*, May 2009, pp. 70–77.

[15] Tcpanon. [Online]. Available: http://www.ing.unibs.it/ntw/tools/tcpanon

[16] W. Yurcik, C. Woolam, L. K. G. Hellings, and B. Thuraisingham, “Scrub-tcpdump: A multi-level packet anonymizer demonstrating privacy/analysis tradeoffs,” in *Proc. 3rd IEEE Intl. Workshop SECOVAL*, 2007, pp. 49–56.

[17] D. Koukis, S. Antonatos, D. Antoniades, E. P. Markatos, and P. Trimintzios, “A generic anonymization framework for network traffic,” in *Proc. IEEE ICC*, Nov. 2006, pp. 2302–2309.

[18] R. Pang and V. Paxson, “A high-level programming environment for packet trace anonymization and transformation,” in *Proc. Conf. Appl., Technol., Architectures Protocols Comput. Commun.*, Aug. 2003, pp. 339–351.

[19] Pktanon. [Online]. Available: http://www.tm.uka.de/software/pktanon

[20] C. Y. Ho, Y. C. Lai, I. W. Chen, F. Y. Wang, and W. H. Tai, “Statistical analysis of false positives and false negatives from real traffic with intrusion detection/prevention systems,” *IEEE Commun. Mag.*, vol. 50, no. 3, pp. 146–154, Mar. 2012.

[21] S. Axelsson, “The base-rate fallacy and the difficulty of intrusion detection,” *ACM TISSEC*, vol. 3, no. 3, pp. 186–205, Aug. 2000.

[22] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, “Towards developing a systematic approach to generate benchmark datasets for intrusion detection,” *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, May 2012.

[23] Y. D. Lin, I. W. Chen, P. C. Lin, C. S. Chen, and C. Hsu, "On campus beta site: Architecture designs, operational experience, and top product defects," *IEEE Commun. Mag.*, vol. 48, no. 12, pp. 83–91, Dec. 2010.

[24] The PCAPLib project in the SourceForge. [Online]. Available: http://sourceforge.net/projects/pcaplib

[25] The PCAPLib framework. [Online]. Available: http://scholars.nbl.org.tw/pcaplib

[26] Harpoon traffic generator. [Online]. Available: http://pages.cs.wisc.edu/jsommers/harpoon

[27] R. Lippmann, R. K. Cunningham, D. J. Fried, I. Graf, K. R. Kendall, S. E. Webster, and M. A. Zissman, "Results of the 1998 DARPA offline intrusion detection evaluation," in *Proc. Int. Symp. RAID*, 1999, pp. 262–271.

[28] DARPA intrusion detection evaluation data sets. [Online]. Available: http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html

[29] V. Paxson, "Considerations and pitfalls for conducting intrusion detection research," in *Proc. 4rth GI Int. Conf. DIMVA*, 2007, pp. 1–37.

[30] S. Floyd and V. Paxson, "Difficulties in simulating the Internet," *IEEE/ACM Trans. Netw.*, vol. 9, no. 4, pp. 392–403, Aug. 2001.

[31] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symp. Secur. Privacy*, May 2010, pp. 305–316.

[32] J. Heidemann and C. Papadopoulos, "Uses and challenges for network datasets," in *Proc. IEEE CATCH*, Mar. 2009, pp. 73–82.

[33] Wireshark SampleCaptures. [Online]. Available: http://wiki.wireshark.org/SampleCaptures

[34] P. C. Lin and Y. W. Lin, "Towards packet anonymization by automatically inferring sensitive application fields," in *Proc. ICACT*, Feb. 2012, pp. 87–92.

[35] R. Ramaswamy and T. Wolf, "High-speed prefix-preserving IP address anonymization for passive measurement systems," *IEEE/ACM Trans. Netw.*, vol. 15, no. 1, pp. 26–39, Feb. 2007.

[36] U. Lamping, "Wireshark developer's guide," Wireshark Foundation, Tech. Rep., 2008. [Online]. Available: http://www.wireshark.org/docs/wsdg_html_chunked

[37] J. Erman, M. Arlitt, and A. Mahanti, "Traffic classification using clustering algorithms," in *Proc. ACM SIGCOMM*, 2006, pp. 281–286.

[38] M. Foukarakis, D. Antoniades, and M. Polychronakis, "Deep packet anonymization," in *Proc. 2nd EUROSEC*, Mar. 2009, pp. 16–21.

[39] M. Burkhart, D. Schatzmann, B. Trammell, E. Boschi, and B. Plattner, "The role of network trace anonymization under attack," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 5–11, Jan. 2010.

[40] I. W. Chen, P. C. Lin, C. C. Luo, T. H. Cheng, Y. D. Lin, Y. C. Lai, and F. C. Lin, "Extracting attack sessions from real traffic with intrusion prevention systems," in *Proc. IEEE ICC*, Jun. 2009, pp. 1–5.

[41] Y. D. Lin, C. N. Lu, Y. C. Lai, W. H. Peng, and P. C. Lin, "Application classification using packet size distribution and port association," *J. Netw. Comput. Appl.*, vol. 32, no. 5, pp. 1023–1030, Sep. 2009.

[42] Block cipher modes of operation. [Online]. Available: http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

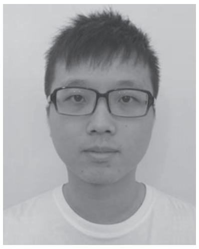
Ying-Dar Lin (F'13) received the Ph.D. degree in computer science from the University of California, Los Angeles, CA, USA, in 1993.

He is currently a Professor of computer science with National Chiao Tung University, Hsinchu, Taiwan. He served as the Chief Executive Officer of Telecom Technology Center, Taipei, Taiwan, in 2010–2011 and a Visiting Scholar with Cisco Systems, San Jose, CA, in 2007–2008. In 2002, he founded the Network Benchmarking Laboratory (www.nbl.org.tw), National Chiao Tung University, which reviews network products with real traffic where has been also the Director. He also cofounded L7 Networks, Inc., in 2002, which was later acquired by D-Link Corporation. He published a textbook "Computer Networks: An Open Source Approach" (www.mhhe.com/lin), with R.-H. Hwang and F. Baker (McGraw-Hill, 2011). His research interests include the design, analysis, implementation, and benchmarking of network protocols and algorithms; quality of services; network security; deep packet inspection; and embedded hardware/software codesign. His work on multihop cellular was the first along this line, and it has been cited for more than 600 times and standardized into IEEE 802.11s, IEEE 802.15.5, IEEE 802.16j, and 3GPP LTE-Advanced.

Prof. Lin became an IEEE Fellow (class of 2013) for his contributions to multihop cellular communications and deep packet inspection. He is an IEEE Distinguished Lecturer for 2014/2015. He has also served in the editorial boards of several journals and program committees of many conferences.

Po-Ching Lin received the B.S. degree in computer and information education from National Taiwan Normal University, Taipei, Taiwan, in 1995 and the M.S. and Ph.D. degrees in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2001 and 2008, respectively.

In August 2009, he joined the faculty of the Department of Computer and Information Science, National Chung Cheng University, Chiayi, Taiwan, where he is currently an Assistant Professor. His research interests include network security, network traffic analysis, and performance evaluation of network systems.



Sheng-Hao Wang received the Master's degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2010.

He is currently a Research and Development Software Engineer with the Institute for Information Industry, Taipei, Taiwan. His research interests include system behavior analysis, network security, and cloud security.

I-Wei Chen received the B.S. and M.S. degrees in computer and information science from National Chiao Tung University (NCTU), Hsinchu, Taiwan.

In 2003, he joined the Network Benchmarking Laboratory (NBL), NCTU, where he is currently the Executive Director. At NBL, he is engaged in the development of testing technologies for network and communication devices. He is particularly interested in technologies using real-world network traffic to test products.

Yuan-Cheng Lai received the Ph.D. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 1997.

In 2001, he joined the faculty of the Department of Information Management, National Taiwan University of Science and Technology, Taipei, Taiwan, where he has been a Professor since 2008. His research interests include wireless networks, network performance evaluation, network security, and social networks.