國立交通大學

網路工程研究所

碩 士 論 文

萃取、分類及匿名封包流量與誤檔漏檔之個案研究

Extracting, Classifying and Anonymizing Packet Traces

with Case Studies on False Positive/Negative Assessment

研 究 生：王聲浩

指導教授：林盈達　教授

中 華 民 國　九 十 九　年 六 月

萃取、分類及匿名封包流量與誤檔漏檔之個案研究

Extracting, Classifying and Anonymizing Packet Traces with Case Studies on False Positive/Negative Assessment

研 究 生：王聲浩　　Student：Sheng-Hao Wang

指導教授：林盈達　　Advisor：Dr. Ying-Dar Lin

國 立 交 通 大 學

網 路 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Network Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2010

Hsinchu, Taiwan

中華民國九十九年六月

萃取、分類及匿名封包流量與誤檔漏檔之個案研究

學生：王聲浩　　指導教授：林盈達

國立交通大學網路工程研究所

摘要

真實網路流量是許多網路相關研究與發展的重要資產，一個具有完整分類的網路流量資料集，可提供研究人員依其需求快速的選擇所需流量類別，此舉有效的降低研究流程在流量採集上的成本與時間。然而公開網路流量所需面臨的風險是使用者個人隱私的洩漏，目前公開網路流量的組織大多依賴贊助者自行上傳且不保障網路流量內的隱私性資料安全。因此本研究旨在提供一個具萃取、分類及匿名的網路流量資料庫在此稱之為 PCAP Lib，PCAP Lib 包含三個目標，第一，利用具偵測流量功能的網路設備取其紀錄用於萃取與分類出十種不同類型且進一步區分出帶有惡意或良好的網路流量。第二，現今的匿名技術大多著重於 TCP/IP 標頭的欄位，即使有些能處理 payload，但解析大量的網路應用層協定仍是一項艱難的議題，在此我們採用深度匿名的方式，確保個人的隱私資料。此外，在網路流量測試中，參與測試的設備其偵測率並無法達到百分之分的準確，因此我們第三個目標是設計一套誤擋與漏擋的分析流程整合於 PCAP Lib 中。在五個月的網路流量中我們主動蒐集 323 筆不同性質的網路流量樣本，其中 33%為正常流量，67%為惡意流量。在匿名方法中，我們定義出 privacy/utility 及 efficiency 用於評量匿名的方法，本文所提出的匿名策略有效性達到 93%，優於其他方法的 27%和 33%。在誤擋與漏擋的分析上，觀察出 63%影響誤擋的主因在於 P2P 類型其動態埠的流量性質，常使偵測設備誤認為其他類型的應用協定，而 62%漏擋的主因在於設備中的特徵資料庫不具可用於比對的特徵值。

關鍵字：流量資料庫、流量分類、封包匿名、誤擋、漏擋

Extracting, Classifying and Anonymizing Packet Traces with Case Studies on False Positive/Negative Assessment

Student: Sheng-Hao Wang **Advisor: Dr. Ying-Dar Lin**

Department of Computer and Information Science

National Chiao Tung University

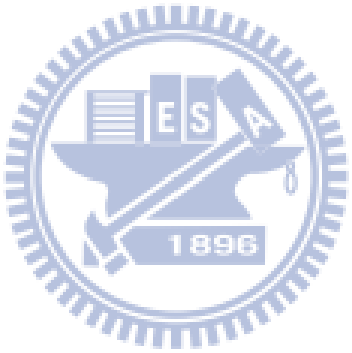
Abstract

Well-classified packet traces make researchers to pick up the class of traces what they want quickly. However, opening packet trace might expose the user's privacy information and let attackers use to exploit. This work aims to provide extracting, classifying, and anonymizing packet traces. We propose PCAP Lib framework which achieves three goals. First, actively extract healthiness and malicious traces from real-world traffic to classify into 10 types of application by multiple detection devices logs. Second, we present an anonymization method to protect personal privacy through deep packet anonymization (DPA). Besides, no one detection device can provide 100% accuracy under packet trace testing; we design an analysis procedure to investigate the cause of false positive (FP)/ false negative (FN) in devices and find out the frequent cases as the third goal. In the result, we collect 323 distinctive packet traces in five months. Among them, 33% are healthy and 67% are malicious. In anonymization, we define "privacy/utility" and "efficiency" to evaluate the different anonymization methods. DPA achieves the best efficiency of 93% than other 27% or 33%. In FP/FN case studies, 63% of FP causes are due to traffic similarity and 62% of FN causes are due to signature insufficiency.

Keywords: trace repository, traffic classification, packet anonymization, false positive, false negative

Acknowledgement

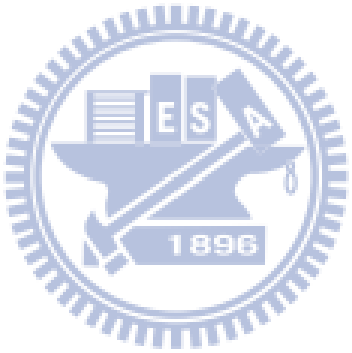
這篇論文是靠許多人的幫忙才能完成，因此要感謝的人也很多，首先謝謝我的指導教授 林盈達教授一直不厭其煩的指導我，給予我相當大的研究空間，其次我感謝林柏青學長適時的提點我，解答我對於研究上的疑惑與困境，不論是課業研究或是做人處事方面兩位都讓我成長許多，再來我要謝謝在高速網路實驗室的同學們，能跟你們一起奮鬥是件很愉快的事，最後我要謝謝我的家人不斷的給我支持與愛護讓我有動力完成研究。沒有你們也就沒有現在的我，謝謝!



Contents

List of Figures

List of Tables



Chapter 1 Introduction

Real-world Internet traffic is useful for intrusion detection systems/prevention (IDS/IDP) and network forensics analysis. For network and security research communities, they rely on large, diverse, active and non-synthetic traffic traces for experimental studies [1], such as investigating the causes of False Positives (FP) and False Negatives (FN) on IDS/IDP. As network analysts can collaborate in inspecting malicious and user network behaviors, they can share the extracted traces from their study. Lastly, educators can use the obtained exemplary data in their student projects and exercises. However, the availability of real-world traces is quite limited so far because they are likely to contain sensitive information such as host addresses, emails, and even authentication keys. Therefore, the packets in the traces should be well anonymized to hide private information before sharing among researchers.

In the past few years, researchers used to focus on anonymization in TCP/IP header fields [2-4]. Although some works can anonymize the payloads [5-8], most solutions are still limited to few protocols (e.g., HTTP, FTP, POP3) because parsing a large number of application protocols is a complex and tedious work. It is also non-trivial to clearly identify the semantics of so many protocols.

To date, most of organizations release the traces from user submission and classify the traces based on the users' perspective [10] [11]. The number of the participants is the largest impact factor to the quality of traffic traces. These traces are often poorly categorized, inactive, and even unusable, as they lack a central and unified control to maintain the quality. An automatic and unified approach to categorize the traces is important for the researchers to acquire the exact traces they want.

For security purpose, traffic traces can be applied into network security systems.

An important requirement of these systems is making them be effective; that is, it should detect a substantial percentage of malicious traffic into the supervised system, while still keeping the FP/FN rate at an acceptable level. The FP/FN rate is the limiting factor for the performance of a network security system. This is due to the base-rate fallacy phenomenon [12].

In this work, we design *PCAP Lib* aims to generate "valuable" packet traces for variety of application usage. The output traces will be extract, classify, anonymize and most importantly, provide false positive/negative assessment from original traces. In packet traces extraction and classification, we proposed an *Active Trace Collection* (ATC) that automatically classifies traffic traces into extracted applications datasets with healthiness and malicious classes. This mechanism based on multiple traffic classification systems such as IDS/IDP, Anti-Virus, Anti-Spam and application classifier (abbreviated as device under test DUT) that leverage their knowledge built into malicious signature databases. In packet trace anonymization, we proposed *Deep Packet Anonymization* (DPA) which provides privacy protection. This mechanism also supports configurable functionality that enables users to select hidden parts in the application layer with consistent transformation, nevertheless, keep the integrity and utility of traces. Finally, in *False Positive/Negative Assessment* (FPNA), it can automatically select potential FP/FN by comparing results produced from different devices and statistic results to most significant causes.

The rest of this paper is organized as follows. Chapter 2 presents the background and related works. Chapter 3 describes the design and solution ideas of our methodology. Chapter 4 addresses the main implementation issues of the PCAP Lib. Chapter 5 displays evaluation of our works. Finally, Chapter 6 concludes this work and discusses the future works.

Chapter 2 Background

2.1 Challenges in sharing traffic traces

Successful development and testing of network analysis devices require a wide available source of robust and accurate traffic traces. Researchers generally use two main approaches to collect traffic traces. One way to create test traces is to generate artificial traffic in lab, such as using Harpoon traffic generator [13] and DARPA-sponsored IDS evaluation datasets [14]. However, useful traffic traces must accurately represent *real* network, user, and system activities, so they must be continually *updated* to reflect new protocols, applications, or changes in user behaviors, it is usually difficult for the artificial traffic to satisfy these requirements. For instance, developers of security products will not be able to train the products with the behaviors of a new attack or create the signatures for that.

Rather than generate traffic on their own, some researchers prefer to capture traffic traces from the backbone traffic in their affiliations[19]. This approach can satisfy the requirement of realistically recreating the bandwidth or activity level of user behaviors. However, it is often difficult to well categorize real-world traffic into several kinds of application traces due to its complexity and volume. Privacy concerns also inhibit sharing traffic traces. Although it is possible to anonymize confidential information in the packets, the *anonymized* packet traces will affect the detection capability of IDS products as they might no longer contain the cues that would trigger an alert or might produce false alarms [20]. This issue will be further discussed in next section. Table 1 compares the packet sources in terms of completeness, update frequency, categorization and anonymization of trace repositories.

Table 1. Comparison of trace repositories

Repository	Source	Update	Category	Anony-mization	Pros	Cons
OpenPacket .org	User submission	Low	Normal Suspicious Malicious	N/A	Categorized traces	PCAP amount is low
Packetlife.net	User submission	High	Property Protocol	N/A	Focus on routing and switching	Un-unitary categorization
DARPA datasets	Simulation	Suspend	Attack free Attacks	N/A	Most popular evaluation traces	Update-less traces
Wireshark/ SampleCaptures	User submission	High	Protocol	N/A	Various traces	Few malicious traces
CAIDA Traces Dataset	OC192 Internet backbone	High	Year	IP address	Prefix-preserving anonymization	Payload truncation
Pcapr	User submission	High	Protocol	N/A	On-line packet display	Unclear categorization
PCAP Lib	Beta-site	High	Application (Malicious/ Healthy)	Entire packet	Various functionalities with unitary repository	No on-line packet display

2.2 Packet trace anonymization

As previously mentioned, *packet trace anonymization* has become an important means to protect the privacy of packet traces for network research. Over the past decade, the anonymization of packet trace has been shifted from the TCP/IP header fields to the application level, but the repositories of sharing packet traces are still few. We consider that several hurdles still for open sharing of traffic traces:

(1) Too many application-level protocols: Besides common protocols (e.g., HTTP, FTP, SMTP, POP3, etc.), there are many others for applications in the real-world, such as P2P, online gaming, and instant messengers. It is difficult to identify and anonymize private information in the packet traces based on the semantics of so many application layer protocols.

(2) Since most IDS/IPS relies on signatures for intrusion detection, the anonymization may accidentally modify an attack signature and affect the detection results. For example, an HTTP GET method may include a malicious content following is a shellcode exploit example :

```
GET
/iframe3?C00jAE12BADcCRkAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA4AAQACB3qzB
```

However, to protect the privacy of web browsing, anonymization tools often hide the URL in the GET method, but this URL may contain an important signature for IDS/IPS detection. For example, a Snort [15] IDS rule below contains a string of all A's in the signature to detect a shellcode. The detection will fail if the string of A's in the URL is anonymized.

```
alert ip $EXTERNAL_NET any -> $HOME_NET any (msg:"SHELLCODE x86 inc
ecx NOOP"; content:"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA";
classtype:shellcode-detect; sid:1394; rev:10;)
```

Anonymization Policies

Several tools or libraries for packet anonymization are available for network research [2-8]. Table 2 compares these tools. Among the tools, only tcpanon, anontool, scrub-tcpdump and Bro can anonymize the packet payloads and remove sensitive information. The others just drop the entire payloads. The tools that can anonymize the payloads are described below:

Table 2. Comparison of anonymization tools

Tools	Feature	MAC addresses	IPv4 address	Application layer
Tcpdpriv	Retain class designation	N/A	Random, Prefix-preserving	Truncation
Tcpmk-pub	Various TCP/IP header fields	Prefix-preserving	Random, Cryptographic, Prefix-preserving	Truncation
FLAIM	Contain a broad set of algorithms	Random, Partial hiding	Random, Prefix-preserving	N/A
Tcpanon	Fields in application layers	N/A	Random, prefix-preserving	Partial field hiding
SCRUB-tcpdump	Substitution specific string in application layer	N/A	Random, subnet/host permutation	Pattern hiding
Anontool	Provide AAPI	Block black marker	Random, prefix-preserving	Specific string hiding
Bro+anon	Various application-level fields	N/A	N/A	Specific fields hiding
PCAPAnon	Deep Packet Anonymization Length-Semantic-Preserving	Block Black Marker	Length-Prefix-Preserving	Deep packet anonymization

Tcpanon [5], written in Python, can parse and anonymize certain fields in the HTTP, FTP, SMTP, POP and IMAP application protocols. However, the payloads of other protocols not supported by tcpanon will be discarded. Writing a new protocol parser in Python is necessary to extend the number of supported protocols. This approach is complicate and error-prone.

Scrub-tcpdump [6] and [21] can search the payload for specified patterns in regular expressions, but these approachs may be imprecise for payload anonymization such as user ID and password. Anontool [7] provides a set of APIs that can also support pattern searching, but the ability of parsing application-level protocols is necessary.

Bro IDS [16] implements an anonymization process as a plug-in [8]. The process can anonymize both on-line and off-line traffic traces. Despite its flexibility, the process still has several limitations. First, as Bro works with events, the fields in the packet header can be altered only for those protocols which have registered events that support trace transformation. That is, a user needs to write the suitable policy scripts, one for each protocol (HTTP, FTP, POP, etc.) for the anonymization. Our framework *PCAPAnon*, on the contrary, provides a larger set of protocol parsers based on wireshark dissectors [17] that can be applied to all packet fields up to the application level. The parsing can help to identify the right fields for anonymization. Second, after application-level anonymization, Bro will store the content into a buffer, and re-create the packets, so the number of packet, packet lengths and other characteristics of traffic traces will be completely different from the original traces. In our framework, we replace the field values with those values of the same semantic and length in the anonymization to maintain the characteristics as much as possible.

2.3 Method of FP/FN assessment

Many researchers and developers consider ways to acquire modern traffic traces

as the test datasets for evaluating their own IDS/IPS designs. Clearly, how well the traffic traces can match real-world ones will have a significant impact on the false-alarm rate. It is demonstrated that even a small rate (1 in 10,000) of false positives could generate an unacceptable ratio of false alarms to real detections [12]. Chen et al. design a system of Attack Session Extraction (ASE) [18] to integrate efforts of signature analysis and development from different vendors to figure out false alarms. The ASE captures, replays, and extracts real-world traffic. The system replays traffic traces to IDS/IPs of different vendors, and finds potential FPs and FNs (denoted by P-FPs and P-FNs) to a certain IDS/IDPby comparing the logs of IDS/IPSs because some attack logs are “logged” or “not logged” only at a certain IDS/IDP. The former is P-FPs, while the latter is P-FNs to the IDS/IDP.

Chapter 3 PCAP Lib Methodology

This chapter details the mechanisms of PCAP Lib which includes three major parts. The first part is Active Trace Collection to provide valuable packet trace from real-world traffic. The second part is Deep Packet Anonymization to precisely and deeply substitute the sensitive information, while still keeping the trace suitable for research. The third part is False Positive/Negative Assessment to find out the frequent causes of FP/FN in detection in security devices from selective packet traces.

3.1 Overview of three mechanisms

The goal of trace sharing is to reserve real-world traffic behaviors in packet traces, which can be replicated and picked up easily among researchers for network forensics. However, recording entire real-world traffic could easily consume up the storage space and searching for specific events in the huge traces is time-consuming. Thus, recording only traffic associated specific events would be better. Furthermore, packet

anonymization protects privacy from leakage in trace sharing. This work proposes *Active Trace Collection* (ATC) and *Deep Packet Anonymization* (DPA) to provide high-quality packet traces that meet the above requirements.

As shown in Figure 1, the ATC actively extracts both healthful and malicious traces from real-world traffic captured in NCTU BetaSite, which is to construct a test network in the campus and record the student network interaction into PCAP files (http://speed.cis.nctu.edu.tw/~ydlin/Betasite.html). The DPA then parses application-level protocol identities and anonymizes sensitive fields in the collected traces. We investigate the causes of FP/FN in the DUTs to the collected traces by *FP/FN Assessment* (FPNA). The details of the three mechanisms are explained in the following subsections.

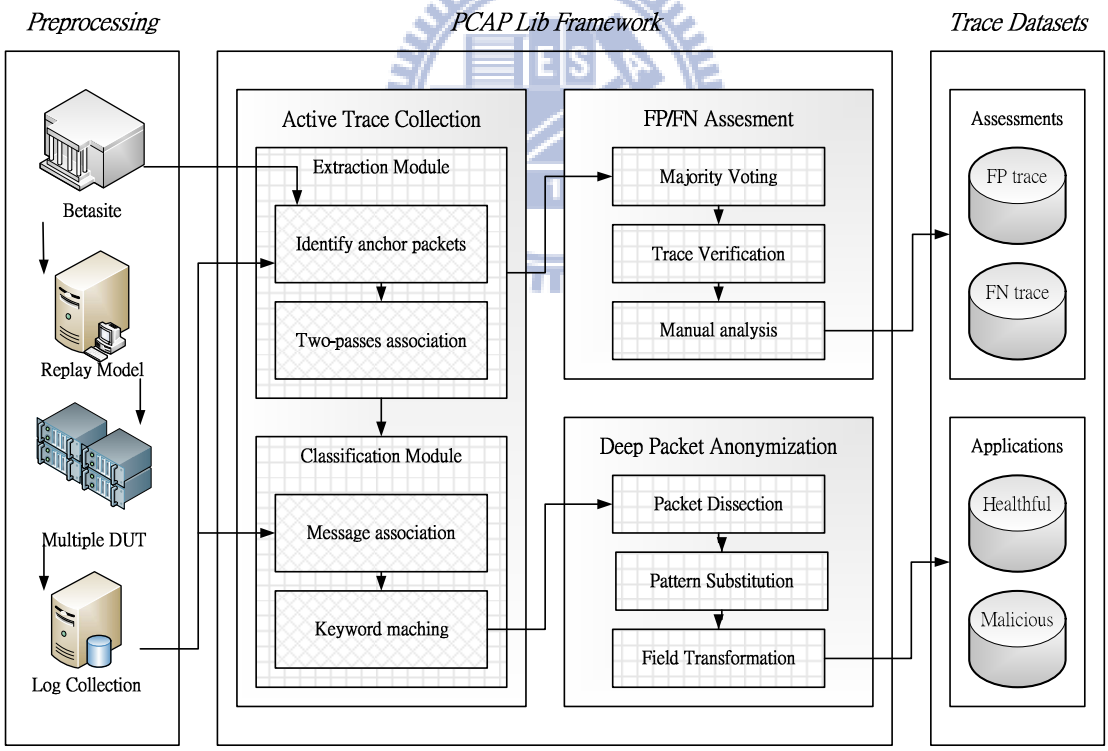


Figure 1: PCAP Lib Block Diagram.

3.2 ATC: Active Trace Collection

The quality of trace repositories to represent real activities relies on the active involvement and frequent update of user submission. We therefore design the ATC

mechanism to extract and categorize large-scale packet traces from real-world traffic, so that users can easily choose what they want. Figure 2 shows the ATC system flow. The procedure of the ATC to provide valuable traffic traces involves two phases: First, it uses traffic replay tool (e.g., tcpreplay) to replay captured raw traffic to multiple DUTs to leverage their domain knowledge. If a DUT detects a specific behavior in the traffic, it will trigger a log. According to the DUT logs, we can find out the anchor packets by 5-tuple, and process 2-level association to extract each specific session into packet trace [18].

Second, we use supervised classification to category the extracted packet traces. From the generated logs, we can also separate various traces into different classifications by keywords. Here we define 10 attributes, each of them includes a variety of keywords to match the logs and decide to which attributes packet traces belong. We also verify the classification to ensure the packet traces are useful. The verification replays the categorized packet trace to the particular DUT(s) which triggered a log from the categorized session in the raw PCAP files. If the session can trigger the same log, we insert it into database; otherwise the trace is invalid. At the same time, we judge whether the traces are unharmful or malicious from the detection result of a set security devices (IDS/IDP, Anti-Virus, Anti-Spam).

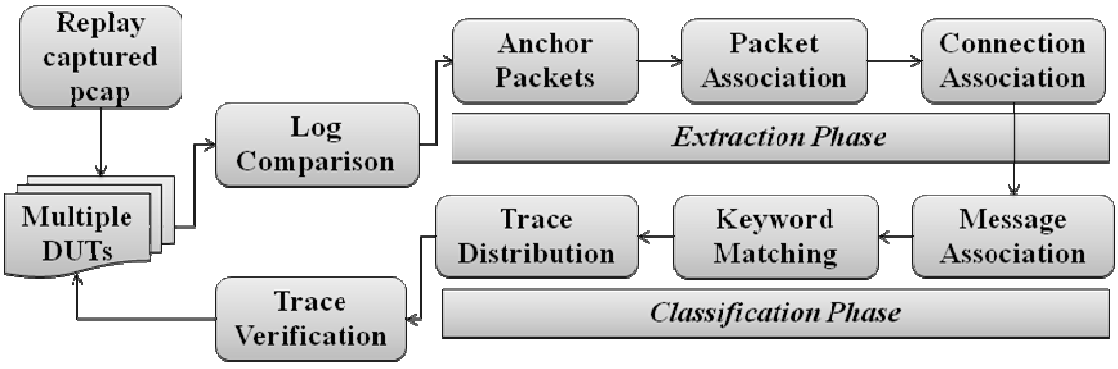


Figure 2: ATC system flow.

3.3 PCAPAnon: Deep Packet Anonymizer

ATC provides the well-classified traces, but when we release these traces for open research organization, we have to face the privacy leakage problem. PCAPAnon supports a precise method for privacy protection of packet traces. The method depends on two elements: (1) *trace parsing* to decide what information in the trace we need to hide and (2) *identity substitution* to choose how we anonymize data elements. Figure 3 presents the PCAPAnon framework of deep packet anonymizer, which anonymizes each layer with various policies. The framework 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 the application protocols. We detail the two elements, *trace parsing* and *identity substitution*, in the following.

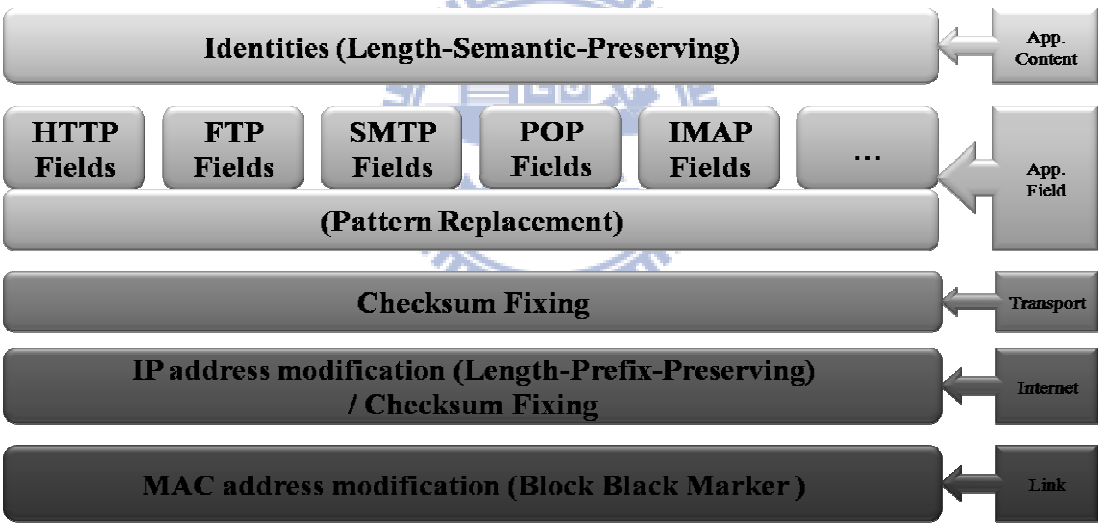


Figure 3: PCAPAnon Framework.

Trace Parsing

A major challenge in protocol parsing is that it is non-trivial for the parser to precisely recognize the semantics of application protocols due to the large number of protocols. Writing a large number of application protocol parsers is complex and tedious. Thereby, we leverage Wireshark (www.wireshark.org), a network packet analyzer, for its plenty of protocol dissectors (www.wireshark.org/docs/dfref)

registered as plugins to parse the traces. Each dissector decodes its part of the protocol, and then hands off the decoding to subsequent dissectors for an encapsulated protocol. User can make a custom script to decide which protocol identities should be transferred by (www.wireshark.org/docs/dfref). For example, "-Tfield –e http.host" changes the http host content.

Besides protocol dissection, we also support regular expression matching (RegExp) to seek and match the sensitive identities, such as IP addresses, Mail addresses, and URLs, for two reasons. First, RegExp is fast and has been used in deep packet inspection. Second, RegExp can match some identities that protocol dissection cannot define in protocol. However, RegExp might miss matches of important privacy information, if it doesn't define the patterns well.

Identity Substitution

PCAPAnon provides a variety of anonymization functions, including *Block Black Marker* for MAC address, *Length-Prefix-Preserving* for IP address (based on cryptographic and mapping table), *Semantic-Length-Preserving* for RegExp matching, *Pattern Replacement* for protocol fields, and *Checksum adjustment* for all protocols, thus providing adequate functionality for each different identity. (1) *Block Black Marker* sets field encryption. (2) *Length-Prefix-Preserving* (LPP) ensures that if two original IP addresses share the same prefix and length, both will still share the same length and prefix, i.e., the subnet and length of the IP addresses remain unchanged. LPP contains three steps: First, DES-ECB records the original digits of four blocks (D_i) in IP address format and encrypts each block value ($Block_i$). Second, MSB-Pad make most significant bit OR 1 in each block to extend every block into 3-digits value. Last step makes $Block_i = Block_i \% (10 \times D_i)$ to reduce it to the original length. (3) *Length-Semantic-Preserving* (LSP) matches an identity like a mail address or URL in the payload, and then substitutes it for

another mail address or URL of the same length. Therefore, the semantics of the identity is reserved. (4) *Pattern Replacement* fills the field with a pattern repeatedly. The pattern can be an integer or string. (5) *Checksum Adjustment* aims to keep checksum valid. If the checksum is invalid, for instance, some IDS/IDP will drop the packets with invalid checksums. The checksum should be re-calculated carefully. The pseudo code for the (2) and (3) procedure are described in Figure 4.

```
IF Pattern is IP address                          //Length-Prefix-Preserving
    Then
    Transfer ascii.pattern to bin.pattern
    For i ← 1 to 4                                //Divide address into four blocks
            D_i ← Digit [Block_i]                 //Record block digits
            Block_i ← DES-ECB [Block_i]           //Encrypt block
            Block_i ← MSB-Pad [Block_i]           //Extend Digits
            Block_i ← Block_i mod (10 x D_i)      //Recovery digits
    Transfer bin.pattern to ascii.pattern
    Replace pattern                               //Copy data buffer to packet
ELSE IF Pattern is Mail address                   //Length_Sematic_Preserving
    Then
    n ← length[pattern]
    m ←length[replace_vec]                        //Basic_replace_vec = "x@nctu.tw"
    IF n > m
        h ← n - m
        Generate data_buf by h
        Merge data_buf and replace_vec
        Replace pattern                           //Copy data buffer to packet
    ELSE
        Replace pattern
ELSE Pattern is URL                               //Length_Sematic_Preserving
    n ← length[pattern]
    m ←length[replace_vec]                        //Basic_replace_vec =
                                                  "http://www.nctu.tw"
    IF n > m
        h ← n - m
        Generate data_buf by h
        Merge head_vector, data_buf, and tail_vector
                                                  //Head_vector="http://www.",
                                                  //Tail_vector=".nctu.tw"
        Replace pattern                           //Copy data buffer to packet
    ELSE
        Replace pattern
```

Figure 4: Pseudo code of LPP and LSP.

In our work, we wish the characteristics of anonymized packet trace can be close to the original one as possible. For instance, we substitute the application-level protocol fields for those of the same type and length. This approach has two main benefits. First, the number of the original packets, the time and the length are reserved. The reservation allows statistical approaches to anomaly detection and flow classification to have a greater chance to still work after anonymization [22]. An IDS/IPS can also parse the protocol semantics as usual.. Second, keeping the length of the original packet makes the design of anonymization tools relatively simple. Fields such as TCP header sequence number, acknowledgment number, options in the field and so on need not be transformed, so are the information fields in an application protocol, such as the HTTP header Content-Length field. The mechanism does not handle the complexity in recalculating the above values after anonymization.

3.4 FPNA: False Positive/Negative Assessment

As previous work, The ATC uses the domain knowledge of IDS/IDP, Anti-Virus, Anti-Spam and application classifier to collect packet traces from real-world traffic. The detection of DUTs might be wrong due to FPs/FNs. To find out these FP and FN cases, we process three steps as follows. First, by majority voting of DUTs, we can find out the potential FPs and potential FNs. The concept is that if the ratio of detection result in DUTs for a specific trace is 1：N-1, than the minor one, which means the unique DUT that detects this trace, is more likely the FP result and the FN case vice versa. Second, after finding out potential FP/FN, we replay the extracted packet traces to DUTs again. This step is to verify whether the cases are reproducible to the original DUTs. Last, to ensure the result is correct, we process manual analysis to investigate the FP/FN causes and count the occurrences of frequent cases.

Chapter 4 Implementation issues of PCAP Lib

This chapter focuses on the implementation issues of the PCAP Lib, which is built on 64-bit Linux kernel 2.6.31. Section 4.1 will cover the extraction and classification phase in ATC. Section 4.2 will cover packet dissection, pattern substitution, and field transformation in PCAPAnon.

4.1 Active Trace Collection

ATC leverages the domain knowledge of security DUTs for intrusion detection, anti-virus, anti-spam, P2P/IM management, and so on, to interpret the specific packet trace. These functions are typical and comprehensive in ordinary security devices.

Extraction phase

The first phase consists of three-pass scanning of the traffic trace: (1) finding out the anchor packets, (2) associating other packets with the anchor packets into the same TCP or UDP connection, and (3) associating connections with the anchor connection into a session. First, there are two tables: Alarm Log Table (ALT), generated by DUTs and used to record logs from DUTs. The other is Replay Log Table (RLT), generated by the machine running Tcpreplay and used to record the time when Tcpreplay sends each packet. Five-tuple in the ALT should be sufficient to identify anchor packets. Unfortunately, some DUTs won't have "five"-tuple, they simply log part of five-tuple, e.g. source IP address and destination IP address. The ASE, therefore, also needs the time information in the ALT and RLT to set up a time frame. This time frame is used to narrow down the searching scope, and thus identify anchor packets correctly. Second, packet association discovers the attack connection where the anchor packets belong to. The packets of same connection share common five-tuple. Last, connection association uses longest common

subsequence (LCS) to calculate the similarity of two packets. After identifying similar packets, we watch the source and destination IP addresses at the same time. This step keeps only the packets which consists the specific traffic behavior within session. The other packets are dropped to reduce the size of traffic trace. After extraction, the information of ALT will be inserted into database and associated with extracted traffic trace which triggers the log for later classification usage. The details of extraction phase can refer to ASE [18].

Classification phase

We wish to provide researchers and developers with packet traces in taxonomy, so that they can select which class to apply in the research. Table 4 defines 10 attributes as our supervised classification training sets that generated by heuristic method. Each class includes a variety of keywords. We apply the keywords to match the DUT logs by regular expressions, and find out the matching packet traces.

Table 3. Classification for Traces and their representative keywords

Attribute	Keywords within DUT Log
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

Next, we separate the packet trace into benign or malicious ones from the detection result of security devices. If most security devices trigger a log from a packet trace, the trace may be malicious with high possibility. Thus, the classification is two-dimensional: one is based on applications, and the other is based on security.

4.2 PCAPAnon framework

This work intends to keep the privacy and utility of traces with deep packet

anonymization. To keep the semantics of packet traces, the anonymzation steps involve trace parsing and identity substitution, respectively. In our work, we propose a configurable policy with three levels of anonymization: (1) TCP/IP header (2) Regular Expression matching and (3) Application-level protocol dissection. Most anonymization tools support the first , so we do not repeat here. We realize the latter two policies with the three stages in the packet processing illustrated in Figure 5, including (1) Packet dissection (2) Field transformation and (3) Pattern substitution. We explain them in the following subsections.

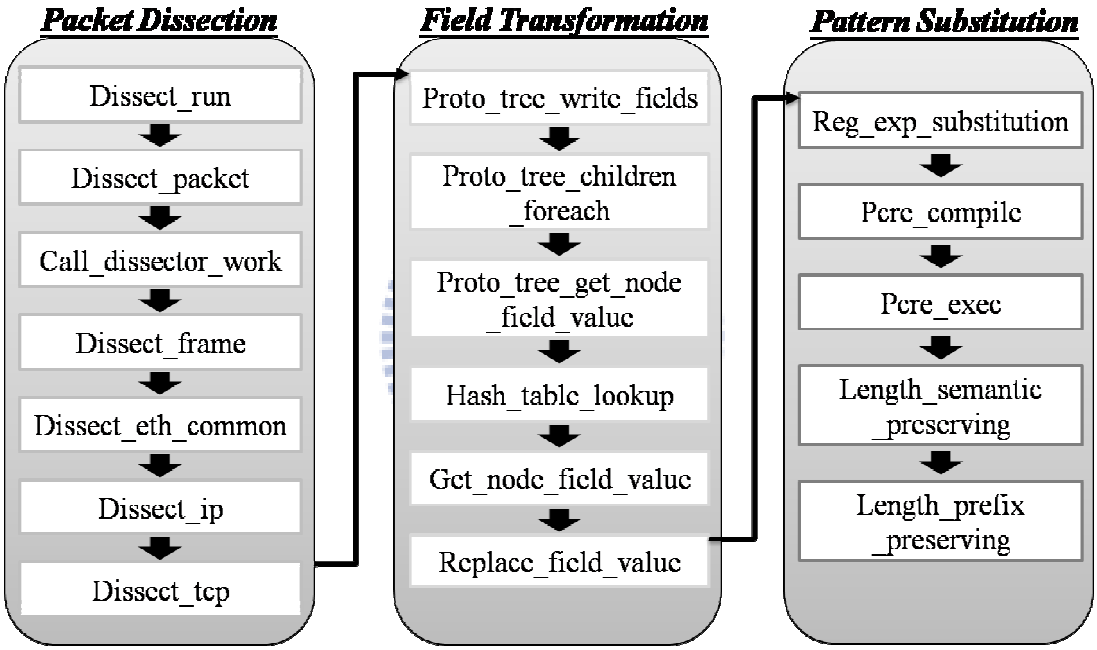


Figure 5: PCAPAnon packet processing.

Packet Dissection

Wireshark provides more than 800 protocol dissectors to handle trace parsing for various application protocols. Each dissector decodes its part of the protocol, and then hands off decoding of the encapsulated protocol to subsequent dissectors. To keep the relationship between protocol layers and handle each layer properly, wireshark uses a data structure of *Protocol Tree*. Figure 6 presents a simple protocol tree for parsing up to the TCP layer. The parsing all 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. The dissector decodes the information in the layer that it is responsible for.

Moreover, wireshark use protocol signature to identify an encapsulated protocol in the sibling node (a child of the same parent node). Each protocol can register their own specific signatures, [data_handle=find_dissector("foo"); foo_handle=create_dissector_handle(dissect_foo,proto_foo);dissector_add("udp.port",3001,foo_handle)];(http://anonsvn.wireshark.org/wireshark/trunk/doc/README.developer). Wireshark can distinguish the children nodes from each other with these signatures. For example, registering TCP port field “tcp.port = 21” can be considered as a signature for FTP. Packets with this signature will be passed to the FTP dissector. The signature can be defined. As a result of enhancing the protocol tree with signature identification, the system is highly flexible for expansion.

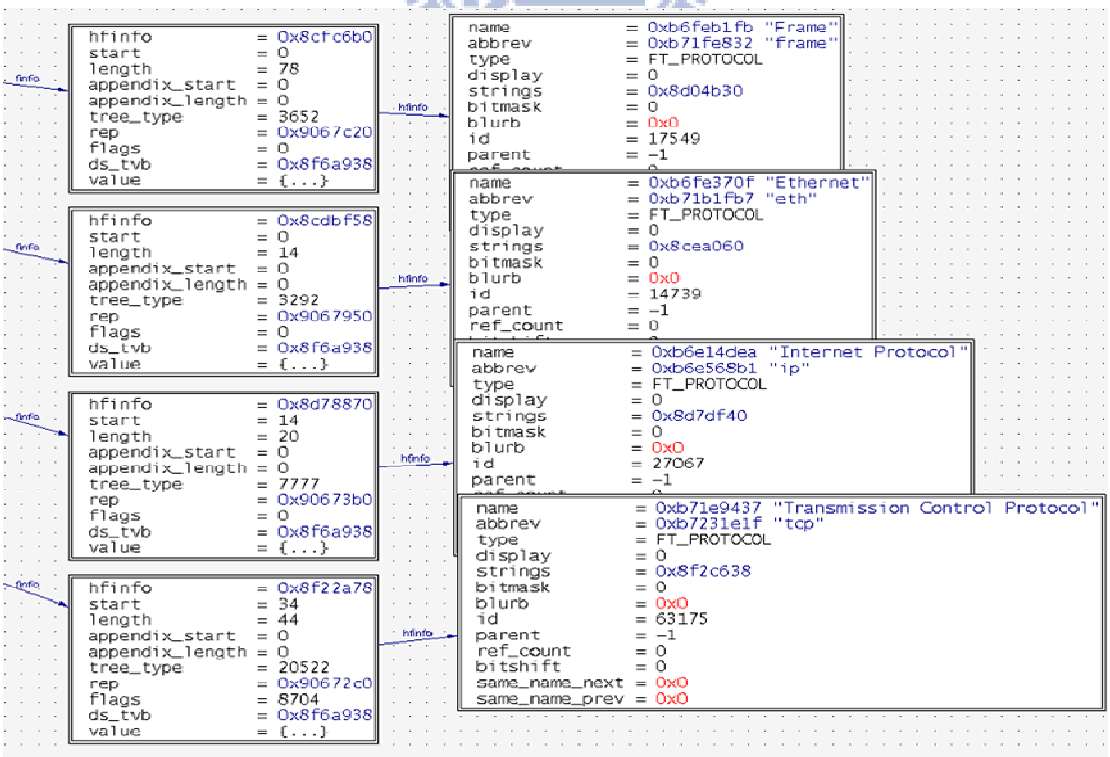


Figure 6: Protocol Tree of Dissection

Field transformation

Packet dissection can unmistakably parse the protocol fields for anonymization of application-level protocol fields. We provide a protocol field list (www.wireshark.org/docs/dfref) for the developer to describe the fields needed for anonymization. The identities can configure in a script, for example, that certain fields such as user ID, password, authentication key, contain private information for anonymization. In the Figure 5, we can get the node field value of protocol tree by the *Proto_tree_get_node_field_value* function. Through Hash_table_lookup, we locate the field values which need to be anonymized and replace the values with consistent length of patterns (e.g.. pass 1234 to pass XXXX).

Pattern substitution

After packet dissection, we can separate TCP/IP header and payload. The anonymization searches for private identities in the packet payload with patterns in regular expressions (listed in Table 4), and hides the identities that are found. If the pattern matches a specific identity in the payload, then we substitute it for another of the same length and semantics. If the payload length is changed due to anoymization, then the sequence/acknowledge numbers of the packets should be adjusted as well to meet the semantics of TCP. If not, traffic analysis that examines the sequence/acknowledge numbers (e.g., packet reassembly in IDS) will result in an error. On the other hand, keeping the semantics of protocol fields is also important, or it will affect the DUT parsing, like malformed mail address in packet.

Table 4. Identity Patterns

Identity	Regular Expression Pattern
IPv4 address	[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}
Mail address	[a-z0-9!#$%&'*+/=?^_`{\|}~-]+(?:\.[a-z0-9!#$%&'*+/=?^_`{|}~-]+)*@(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.)+[a-z0-9](?:[a-z0-9-]*[a-z0-9])?
URL	(?:(?:(?:http|ftp|gopher|telnet|news)://)(?:w{3}\.)?(?:[a-zA-Z0-9/;\?&=:\-_\$\+!*'\(|\\~\[\]#%\.])+)

Domain name	([a-zA-Z0-9]([a-zA-Z0-9\\-]{0,61}[a-zA-Z0-9])?\\.)+[a-zA-Z]{2,6}

We propose two anonymization functions for *Length-Prefix-Preserving* and *Length-Semantics-Preserving*in identity substitution. If two IP addresses *a* and *b* share a common k-bit prefix, and their address lengths are ℓ and m in the ASCII representation, then LPP(a) and LPP(b) also share a k-bit prefix and their lengths ℓ and m are reserved as well. Therefore, the characteristics of packets, such as lengths and numbers, can be reserved after anonymization. The reservation is important if the packet traces are to be analyzed statistically.

Three steps are in LPP: (1) DES-ECB record the original digits and scrambles the four blocks of the original IP address individually with the DES algorithm. (2) MSB-Pad extends each block to three digits in decimal by padding the most significant bit with 1 for length preserving. (3) Because we recorded the block digits of original IP address in DES-ECB, we mod each block according to their digit records to restitute original IP address ASCII length.

Chapter 5 Evaluation and Observation

In this chapter, we evaluate the three main modules of the PCAP Lib framework. First, ATC provides 323 packet traces with different traffic behaviors that be classified into 10 x 5 classification matrix. Second, the utility and privacy of anonymization tools are evaluated. We also evaluate the efficiency of three different anonymization policies supported by PCAPAnon. Finally, we assess the most frequent FP/FN cases, and find out main causes of these FP/FP cases.

5.1 Completeness of various class traces

Our real-world traffic is captured from the NCTU BetaSite (http://speed.cis.nctu.edu.tw/~ydlin/Betasite.html) from October 1, 2009 to February

1, 2010. Table 6 presents number of the classified traces generated by ATC which includes 8 DUTs (BroadWeb, Cisco, D-Link, Fortinet, McAfee, TrendMicro, TippingPoint, and ZyXEL). During the period of five months, the web traffic is the most popular application traces. It includes 40% of web-type traces are malicious, meaning attackers frequently exploit Web applications. File transfer is the second popular application, and 30% traces are malicious. Table 7 summarizes the 10 application classes, each containing various application name that represents the number of application traces that we collected in the five-month by ATC.

Table 5. Trace Classification Matrix

	Web	**Email**	**File Transfer**	**Remote Access**	**Encryption**	**Chat**	**File Sharing**	**Streaming**	**VoIP**	**Network**
Healthy General	53	8	36	8	6	15	21	6	2	32
Healthy Special	21	4	0	2	0	1	0	0	0	0
Attack	49	6	15	5	6	5	0	0	2	13
Virus	0	0	0	1	1	0	0	0	0	0
Spam	2	3	0	0	0	0	0	0	0	0
Total	125	21	51	16	13	21	21	6	4	45

Table 6: Number of Application Packet Traces (number)

Attribute	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**
Web	HTTP (125)						
Email	POP3 (5)	SMTP (11)	IMAP (5)				
File Transfer	FTP (28)	SMB (22)	TFTP (1)				
Remote Access	Telnet (6)	SSH (4)	RDP (4)	VNC (2)			
Encryption	SSL (11)	FTPs (1)	HTTPs (1)				
Chat	IRC (7)	ICQ (4)	Yahoo Messenger (4)	MSN (1)	AIM (1)	Skype (1)	Google talk (1)
File Sharing	Bittorrent (2)	eDonkey (1)	Gnutella (1)	Pando (1)	SoulSeek (1)	Winny (1)	Xunlei (1)
Streaming	PPLive (2)	QuickTime (1)	Octoshape (1)	Orb (1)	Slingbox (1)		
VoIP	SIP (4)						
Network	NetBIOS (21)	DNS (19)	SNMP (3)	Socks (1)	STUN (1)		

5.2 Privacy, Utility, and Efficiency of Anonymization policies

In this evaluation, we define the utility and privacy for the anonymization. The

privacy is evaluated with the percentage of sensitive fields in the packet traces that have been anonymized precisely. Utility is evaluated with the percentage of malicious packet traces after anonymization that can be still detected by DUTs. We use Snort 2.8.5, which is an open source signature-base IDS, as the DUT to verify trace that we can investigate the effect of anonymization by comparing snort's signatures and packet traces. The first step is to replay the raw ATC traces to Snort and collect the logs. Next, we anonymize these traces and replay them again to calculate the metrics. We choose true positive (TP) and false negative (FN) as our metrics to define utility and privacy as follows: TPfield is True Positive of field, FNfield is False Negative of field and TPtrace is True Positive of trace, FNtrace is False Negative of trace, $\text{Privacy} = \frac{\#\text{ of TP}_{\text{field}}}{\#\text{ of TP}_{\text{field}} + \#\text{ of FN}_{\text{field}}}$, $\text{Utility} = \frac{\#\text{ of TP}_{\text{trace}}}{\#\text{ of TP}_{\text{trace}} + \#\text{ of FN}_{\text{trace}}}$. Since the HTTP malicious traces dominate our ATC collected traces, we use these traces in the evaluation. Figure 7 presents the impact on the utility from of the anonymization of HTTP header fields. The evaluation observes that most malicious signatures are embedded in host/cookie/request.uri fields. If these fields are anonymized, the traces will not be triggered by DUTs. Figure 8 illustrates an example of false negatives that occurs in anonymizing http URI field. The original packet trace tried to access the password configuration file against GET URI /etc/passwd. If the URI argument is anonymized for privacy protection, the / signature in the packet trace will be lost --- the trace will not trigger the alert of "WEB-MISC /etc/passwd" anymore.

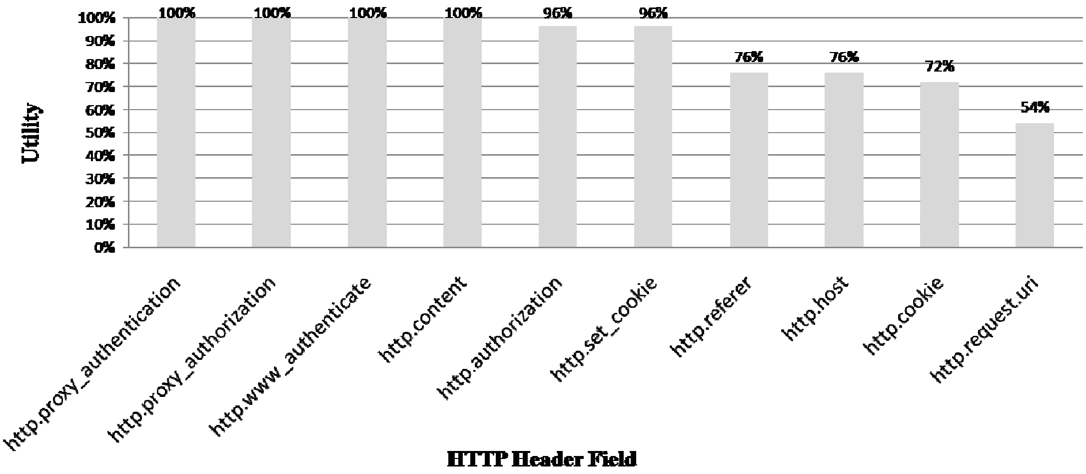


Figure 7: The anonymization impact of HTTP header fields

[Before Anonymization] GET //lists/admin/index.php?_SERVER[ConfigFile]=../etc/passwd HTTP/1.1
[After Anonymization] GET XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX 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;)

Figure 8. False negative in anonymized HTTP trace

Figure 9 and 10 compare the privacy and utility of PCAPAnon with two other anonymization tools, anontool and tcpanon which can anonymize packet payload. In the privacy evaluation, we defined sensitive identities for four protocols,http, ftp, pop and smtp, as Table 8 lists. Figure 9 shows the comparison result, we can observe that (1) anontool provides only pattern substitution and leads to privacy leak seriously, (2) tcpanon uses customized parsing to hide more identities than anontool, but its ftp protocol parser not be defined well. It misses some sensitive identities such as PORT and STOR, (3) PCAPAnon provides DPA to hide identities accurately due to its rich set of protocol parsers.

Figure 10 shows the utility result, we can observe that

(1) Although anontool keeps excellent utitility after anonymization, it maintains the privacy poorly withitd rough pattern substitution;

(2) tcpanon's customized parsing overwrites significant content, so the utility becomes poor.

(3)Because PCAPAnon provides protocol dissection which can parse protocol fields accurately, it avoids overwriting the packet payload and modified the specific content which occurred in tcpanon, and results in less alternation of signatures.

Table 7. Identities of Privacy Measurement

HTTP sensitive fields	FTP sensitive fields	POP sensitive fields	SMTP sensitive fields
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.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		

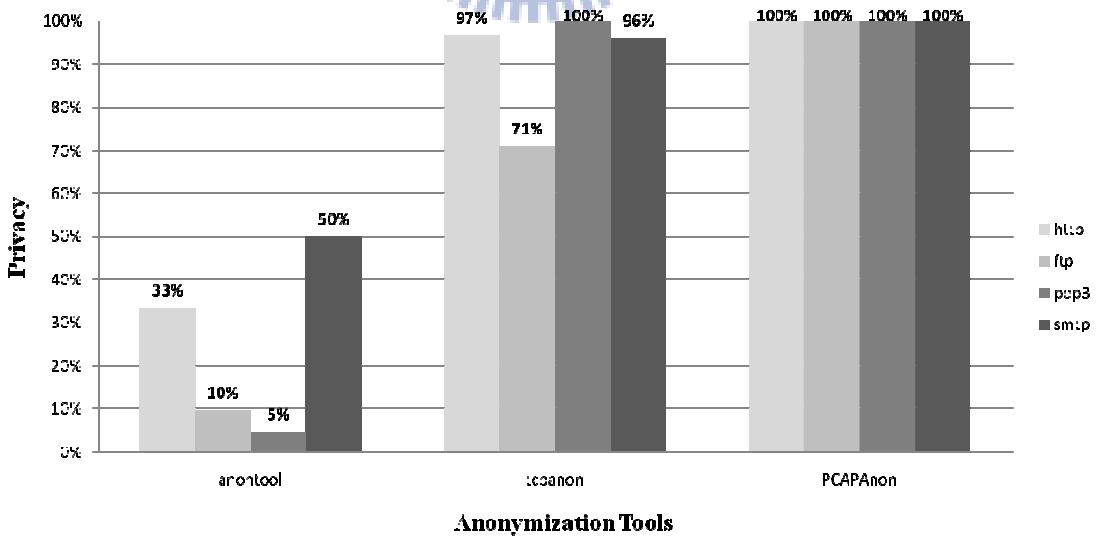


Figure 9: Privacy of Anonymization Tools

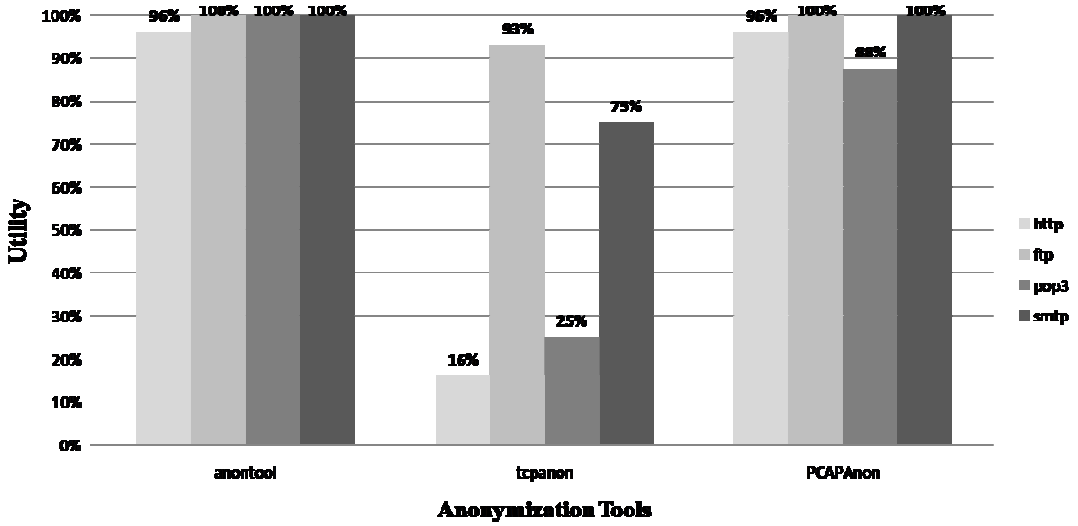


Figure 10: Utility of Anonymization Tools

Besides privacy and utility, we define a metric for the efficiency of anonymization as follows to consolidate the former two metrics as one. The single metricis useful for comparing anonymization tools and policies when both privacy and utility are of concern.

$$\text{Efficiency} = \text{Privacy} \times \text{Utility}$$
$$= \frac{\#\text{ of TP}_{\text{field}}}{\#\text{ of TP}_{\text{field}} + \#\text{ of FN}_{\text{field}}} \times \frac{\#\text{ of TP}_{\text{trace}}}{\#\text{ of TP}_{\text{trace}} + \#\text{ of FN}_{\text{trace}}}$$

Figure 11 presents the efficiency of the anonymization tools in the evaluation. The efficiencies of anontool and tcpanon are 27% and 33%, while that of PCAPAnon's DPA support is as high as 93%.

Figure 12 shows the efficiency of PCAPAnon three-level anonymization policies. (1) Hiding MAC/IP addresses and Checksum in the L2/L3/L4 headers provides little protection of sensitive identities, (2) Pattern Matching allows to replace Mail/IP/URL within the payload, but if identities were not defined in patterns, these identities will be missed from protection. (3) Protocol Dissection can precisely parse for protocol semantics (e.g.. Host, Cookie, Authentication Key, Password and User ID) to anonymize not only the patterns but also field values. In our experiment, the three combo policies gain good efficiency up to 93%.

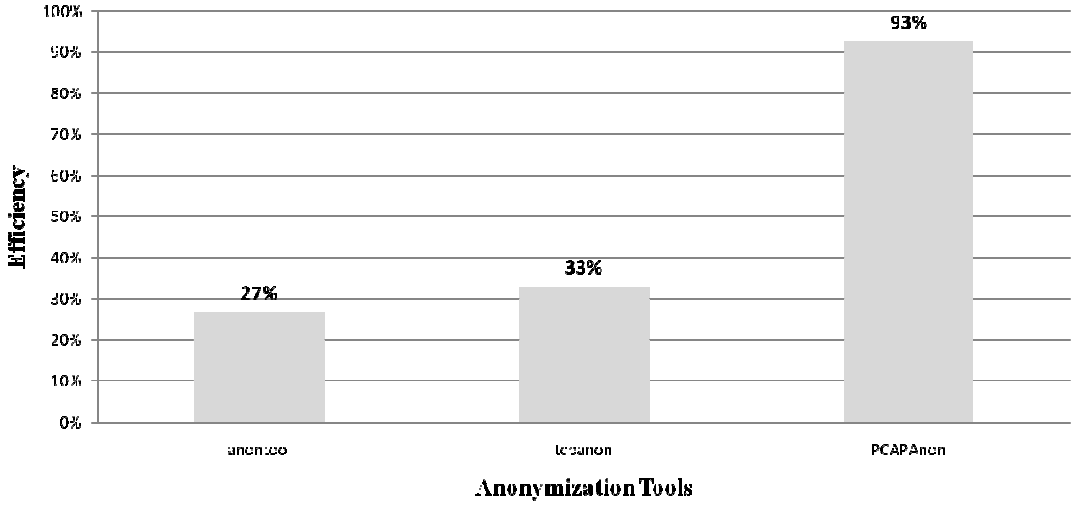


Figure 11: Efficiency of anonymization tools

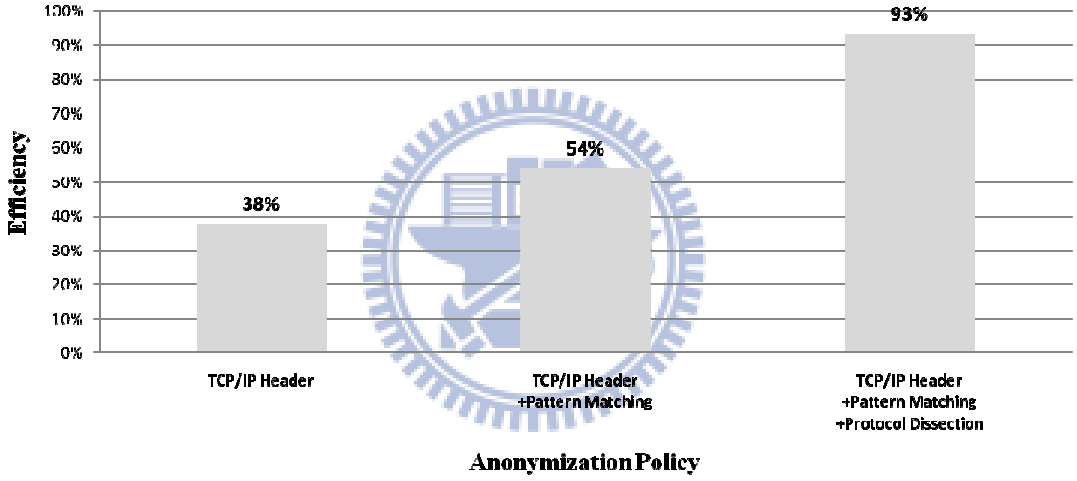


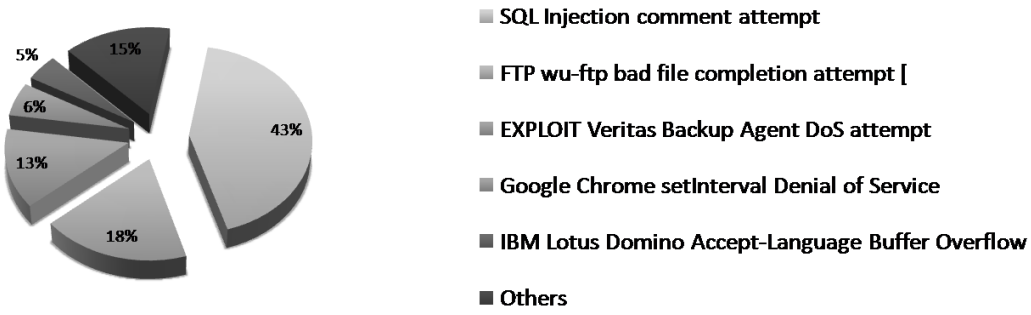
Figure 12: Efficiency of three-level anonymization policies

5.3 Statistics of FP/FN cases

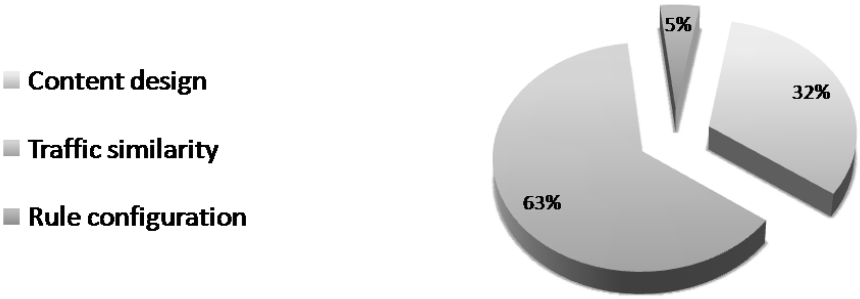
FPNA provides a convenient way to find out the FPs and FNs of DUTs. In this section, we study the causes of FP/FN cases using the same ATC source traffic captured from the NCTU BetaSite. In our investigation, the main causes of FP/FN can be subsumed into three types: Type 1 is attributed to signature design.The signatures are too general or rough, so that they can easily match the packet content. Type 2 is attributed to traffic similarity, where normal traffic may behavior weirdly or mistake to other network protocol. Type 3 is Rule Configuration, meaning some configuration

arguments may not be instituted well, such as a threshold too high to detect a specific malicious behavior of.

Figure 13(a) shows the most frequent FP cases (1) SQL Injection comment attempt results from BitTorrent traffic similarity because the client binds port 80 (2) FTP wu-ftp bad file completion attempt [results from "[" character often appear in ftp transfer data (3) EXPLOIT Veritas Backup Agent DoS attempt results from BitTorrent traffic similarity because the client bind port 10000 (4) Google Chrome setInterval Denial of Service results from SetInterval('swltxtColor()', 500) may be used in many web pages and its' User-Agent is Mozilla/4.0 not Chrome. (5) IBM Lotus Domino Accept-Language Buffer Overflow results from Accept-Language field does not exist buffer overflow code just because field length over 100. Figure 13(b) present the proportion of three types that Traffic Similarity accounts for 63% of the FP cases because P2P dynamic port make the DUTs mistake the application protocols (e.g. (1) and (3)).



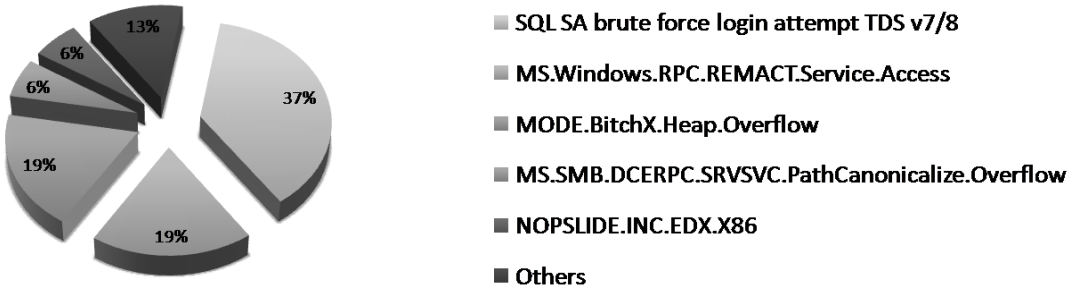
(a) Top five frequent cases



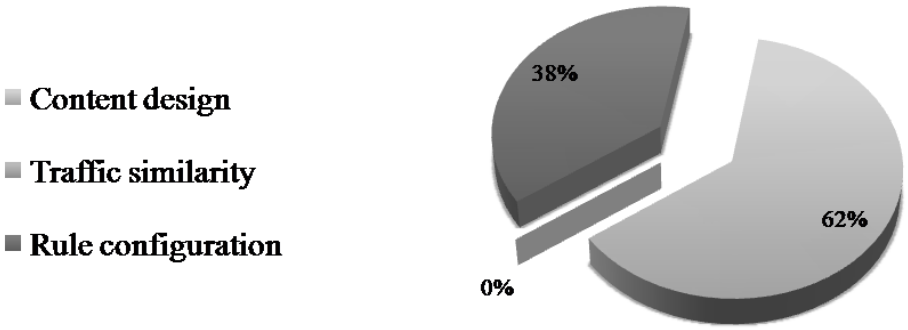
(b) Proportion of three causes

Figure 13: Statistic of False Positives

Figure 14(a) shows the most frequent FN cases. (1) SQL SA brute force login attempt TDS v7/8 due to threshold be set to 5 times in 2 seconds, but in our investigation, it happens in 3 times in 2 seconds. The other four cases result from the DUTs does not have such signatures. From Figure 14(b), we can know the insufficiency signatures account for 62% of FN cases.



(a)Top five frequent cases



(b) Proportion of three causes

Figure 14: Statistic of False Negatives

Chapter 6 Conclusions and Future Works

This work proposes a PCAP Lib framework to provide well-classified packet traces with anonymization and FP/FN case studies from these traces. ATC collects 323 distinctive packet trace in five months. 33% of the packet traces are healthy and 67% are malicious. The distribution of collected traces shows that web applications, which occupy 40%, are a frequent way that attacker used to exploit.

In anonymization, we define “privacy/utility” and “efficiency” to evaluate the different anonymization methods. PCAPAnon uses DPA to achieve the best efficiency 93%. Moreover, PCAPAnon’s efficiency of pattern matching 51% higher than anontool due to it supports global search.

In FP/FN case studies, FPNA gives the statistic of cases from ATC collected traces. Herein, we focus on security devices, but the method could be extended to other DUTs. In false positive, we observe that *traffic similarity* 63% dominates the high percentage because P2P dynamic port makes the DUTs mistaking the application protocol. In false negative, *signature insufficiency*, which is the main cause, occupies 62% high proportion. To researcher and developer, PCAP Lib provides completeness and flexibility to satisfy their various purposes.

Although PCAP Lib has many functions, it still exists an issue needs to be solved. As Section 5.2 shows false negative in anonymized trace, if malicious signatures are embedded in privacy fields, we choose to protect privacy first. Because these signatures are modified, packet trace will not be triggered by IDS/IDP. According to the feedbacks of IDS/IDP then reserving the signature contents is a way to avoid this situation happen. Another issue is due to our anonymization policy script base on manual decide which protocol field should be transferred. But if the packet traces contain various protocols, it will hard to configure. Hence, a good way is to use traffic statistic tool (e.g. trace-summary) identify the protocols in traces and provide a collaborative mechanism for user can modify the same policy script.

References

[1] P. Porras, and V. Shmatikov, "Large-scale collection and sanitization of network security data: risks and challenges," Proc. of the 2006 workshop on New security paradigms, Germany, pp. 57-64, Sep. 2007.

[2] G. Minshall, "TCPdpriv: Program for Eliminating Confidential Information from Traces," Ipsilon Networks, Inc. http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html.

[3] R. Pang, M. Allman, V. Paxson, J. Lee, "The Devil and Packet Trace Anonymization," ACM SIGCOMM Comput. Commun. Rev., vol. 36, pp. 29-38, Jan. 2006.

[4] K. Lakkaraju and A. Slagell, "Evaluating the Utility of Anonymized Network Traces for Intrusion Detection," Proc. of the 4th international conference on security and privacy in communication netowrks, Sep. 2008.

[5] Tcpanon, available at http://www.ing.unibs.it/ntw/tools/tcpanon/

[6] W. Yurcik, C. Woolam, G. Hellings, L. Khan, B. Thuraisingham, "SCRUB-tcpdump: A Multi-Level Packet Anonymizer Demonstrating Privacy/Analysis Tradeoffs," 3rd IEEE Intl. Workshop on the Value of Security through Collab. (SECOVAL), pp. 49-56, 2007.

[7] D. Koukis, S. Antonatos, D. Antoniades, E. P. Markatos, P. Trimintzios "A Generic Anonymization Framework for Network Traffic," Communications, 2006. ICC '06. IEEE International Conference on, pp. 2302-2309, Nov. 2006.

[8] R. Pang, and V. Paxson, "A High-Level Programming Environment for Packet Trace Anonymization and Transformation," Proc. of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications, Karlsruhe, Germany, pp. 339-351, Aug. 2003.

[9] G. C. Tjhai, M. Papadaki, S. M. Furnell, N. L. Clarke, "Investigating the Problem of IDS False Alarms: An Experimental Study Using Snort," IFIP, Proc. of The IFIP Tc 11 23rd International Information Security Conference, pp. 253-267, Jul. 2008.

[10] Packetlife, available at http://www.packetlife.net/captures/.

[11] Pcapr, available at http://www.pcapr.net/.

[12] S. Axelsson, "The Base-Rate Fallacy and the Difficulty of Intrusion Detection," ACM Transactions on Information and System Security (TISSEC), vol. 3, pp. 186-205, Aug. 2000.

[13] Harpoon traffic generator, available at
http://pages.cs.wisc.edu/~jsommers/harpoon

[14] DARPA Intrusion Detection Evaluation Data Sets, available at
http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html

[15] M. Roesch, "Snort: Lightweight intrusion detection for networks," USENIX

LISA Conference, 1999.
[16] V. Paxson, "Bro: A system for detecting network intruders in real-time," Comput. Networks, vol. 31, no. 23, pp. 2435-2463, 1999.
[17] U. Lamping, "Wireshark Developer's Guide," Wireshark Foundation, 2008. available at http://www.wireshark.org/docs/wsdg_html_chunked/
[18] I. W. Chen, P. C. Lin, C. C. Luo, T. H. Cheng, Y. D. Lin, Y. C. Lai, F. C. Lin, "Extracting Attack Sessions from Real Traffic with Intrusion Prevention Systems," ICC, IEEE, 2009.
[19]R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning For Network Intrusion Detection," Proc. IEEE Symposium on Security and Privacy, May 2010
[20]J. Heidemann and C. Papadopoulos. "Uses and Challenges for Network Datasets," In Proc. IEEE Cybersecurity Applications and Technologies Conference for Homeland Security (CATCH), Washington, DC, USA. Mar. 2009
[21]M. Foukarakis, D. Antoniades and M. Polychronakis,"Deep packet anonymization," Proc. ACM EUROSEC, Mar. 2009
[22]Ying-Dar Lin, Chun-Nan Lu, Yuan-Cheng Lai, Wei-Hao Peng, and Po-Ching Lin, "Application Classification Using Packet Size Distribution and Port Association," Journal of Network and Computer Applications, Vol. 32, Issue 5, pp. 1023-1030, Sep. 2009.

Appendix. POP3 Payload Deep Anonymization

This table shows the anonymization result of PCAPAnon, The left column shows the original payload data which is a plain mail content. The right column shows mail address in column (2, 3, 5, 13, 15, 25, 29-34) are substituted by Length-Semantic-Preserving and IP address in column (7 and 20) are substituted by Length-Prefix-Preserving.

1 +OK 797880 octets	1 +OK 797880 octets
2 Return-Path: <bluemoon609.ac95@g2.nctu.edu.tw>	2 Return-Path: <nblnblnblnblnblnblnb@nbl.org.tw>
3 X-Original-To: nightmare0918.ac95@nctu.edu.tw	3 X-Original-To: nblnblnblnblnblnbln@nbl.org.tw
4 Delivered-To:	4 Delivered-To:
5 nightmare0918.ac95@d2-spool-lb-0.nctu.edu.tw	5 nblnblnblnblnblnblnblnblnblnbl@nbl.org.tw
6 Received: from mail-iw0-f194.google.com	6 Received: from mail-iw0-f194.google.com
7 (mail-iw0-f194.google.com [209.85.223.194])	7 (mail-iw0-f194.google.com [138.52.206.189])
8 by d2-spool-lb-0.nctu.edu.tw (Postfix) with ESMTP id	8 by d2-spool-lb-0nctuedu.tw (Postfix) with ESMTP id
9 B010969A8D7;	9 B010969A8D7;
10 Tue, 27 Oct 2009 09:38:09 +0800 (CST)	10 Tue, 27 Oct 2009 09:38:09 +0800 (CST)

11 Authentication-Results: d2-spool-lb-0.nctu.edu.tw;	11 Authentication-Results: d2-spool-lb-0.nctu.edu.tw;
12 sender-id=none	12 sender-id=none
13 header.from=bluemoon609.ac95@g2.nctu.edu.tw;	13 nblnblnblnblnblnblnblnblnblnb@nbl.org.tw;
14 spf=none	14 spf=none
15 smtp.mfrom=bluemoon609.ac95@g2.nctu.edu.tw	15 nblnblnblnblnblnblnblnblnbln@nbl.org.tw
16 Received: by iwn32 with SMTP id 32so6481732iwn.23	16 Received: by iwn32 with SMTP id 32so6481732iwn.23
17 for <multiple recipients>; Mon, 26 Oct 2009	17 for <multiple recipients>; Mon, 26 Oct 2009
18 18:38:08 -0700 (PDT)	18 18:38:08 -0700 (PDT)
19 MIME-Version: 1.0	19 MIME-Version: 1.0
20 Received: by 10.231.1.22 with SMTP id	20 Received: by 37.188.7.86 with SMTP id
21 22mr1672949ibd.56.1256607488296; Mon, 26	21 22mr1672949ibd561256607488296; Mon, 26
22 Oct 2009 18:38:08 -0700 (PDT)	22 Oct 2009 18:38:08 -0700 (PDT)
23 Date: Tue, 27 Oct 2009 09:38:08 +0800	23 Date: Tue, 27 Oct 2009 09:38:08 +0800
24 Message-ID:	24 Message-ID:
25<8197f2480910261838h50b01e49x933c16c77428dba1@	25<nblnblnblnblnblnblnblnblnblnblnblnblnblnbl@n
26 mail.gmail.com>	26 bl.org.tw>
27 Subject: =?Big5?B?xbLD0azsvsekwLLVs/inaarsqqk=?=	27Subject: =?Big5?B?xbLD0azsvsekwLLVs/inaarsqqk=?=
28 From: =?Big5?B?pP2pycV0?=	28 From: =?Big5?B?pP2pycV0?=
29 <bluemoon609.ac95@g2.nctu.edu.tw>	29 <nblnblnblnblnblnblnb@nbl.org.tw>
30 To: codyp.iac93g@nctu.edu.tw, vm3m4bj6@hotmail.com,	30 To: nblnblnblnbln@nbl.org.tw, nblnblnbl@nbl.org.tw,
31 seyron.ac95@g2.nctu.edu.tw,	31 nblnblnblnblnbl@nbl.org.tw,
32 nightmare0918.ac95@nctu.edu.tw,	32 nblnblnblnblnblnbln@nbl.org.tw,
33 joy0910.ac95@nctu.edu.tw,	33 nblnblnblnbln@nbl.org.tw,
34 agnesbird99@hotmail.com, nevertears@gmail.com	34 nblnblnblnbl@nbl.org.tw, nblnblnbl@nbl.org.tw
35 Content-Type: multipart/mixed;	35 Content-Type: multipart/mixed;
36 boundary=00151773eaa0f64c850476e0badb	36 boundary=00151773eaa0f64c850476e0badb
37	37
38 --00151773eaa0f64c850476e0badb	38 --00151773eaa0f64c850476e0badb
39 Content-Type: multipart/alternative;	39 Content-Type: multipart/alternative;
40 boundary=00151773eaa0f64c740476e0bad9	40 boundary=00151773eaa0f64c740476e0bad9
41	41
42 --00151773eaa0f64c740476e0bad9	42 --00151773eaa0f64c740476e0bad9
43 Content-Type: text/plain; charset=Big5	43 Content-Type: text/plain; charset=Big5
44 Co	44 Co