SCRUB-tcpdump: A Multi-Level Packet Anonymizer Demonstrating Privacy/Analysis Tradeoffs

William Yurcik* , Clay Woolam†, Greg Hellings†, Latifur Khan† and Bhavani Thuraisingham †
*University of Illinois at Urbana-Champaign, USA
†University of Texas at Dallas, USA
{cpw021000,gsh062000,lkhan,bhavani.thuraisingham}@utdallas.edu

***Abstract*—To promote sharing of packet traces across security domains we introduce *SCRUB-tcpdump*, a tool that adds multi-field multi-option anonymization to *tcpdump* functionality. Experimental results show how *SCRUB-tcpdump* provides flexibility to balance the often conflicting requirements for privacy protection versus security analysis. Specifically, we demonstrate with empirical experimentation how different *SCRUB-tcpdump* anonymization options applied to the same data set can result in different levels of privacy protection and security analysis. Based on these results we propose that optimal network data sharing needs to have different levels of anonymization tailored to the participating organizations in order to tradeoff the risks of potential loss or disclosure of sensitive information.**

***Index Terms*—network data sharing, security data sharing, privacy protection, anonymization, data obfuscation, network monitoring, network intrusion detection, network packet traces**

I. Introduction

The focus of an individual Computer Emergency Response Team (CERT) is on preventing damage to their own organization. If an attacker is particularly active in most cases they are identified, tracked, and blocked – case closed. Only in rare cases under special circumstances may there be end-to-endinvestigation involving a law enforcement prosecution. With each CERT focusing on its own organization, this allows attackers to frequently shift their targets from one organization to another organization with impunity. With little sharing of information between CERTs, each organization is essentially on its own against all attackers and no cumulative learning occurs between organizations – new attacks/attackers against an organization necessitate "reinventing-the-wheel" of protection that other organizations may have already solved.

Organizational CERTs do not typically share information about attackers with peer organizations because (1) the costs of sharing (in terms of resources and effort) outweigh the perceived benefits and (2) there are valid concerns about information disclosure when sharing network data since sensitive information will be included. To address these problems, what is needed is: (1) a tool that easily prepares data so it can be shared and (2) a tool that protects sensitive information from being disclosed.

∗ based on work performed at University of Illinois at Urbana-Champaign, William Yurcik is now PET onsite at the Army Research Laboratory and can be contacted at byurcik@gmail.com.

Third IEEE International Workshop on the Value of Security through Collaboration (SECOVAL), Nice France, 2007.

To overcome this present inertia against sharing we have developed a new tool, *SCRUB-tcpdump*, that builds on the popular *tcpdump*[18] tool for easy data management of packet traces while simultaneously protecting sensitive information from being disclosed through the use of anonymization. With *SCRUB-tcpdump*, a user can anonymize fields considered sensitive to multiple desired levels by selecting options that can remove all the information (black marker), add noise, or permute the data. These multi-level anonymization techniques can be employed on different fields simultaneously – with varying effects on the statistical properties of the entire packet trace. Since different organizations have security policies with different privacy requirements, multi-level anonymization options for each field provides flexibility in selecting anonymization schemes to more closely match real sharing environments between parties. There is no one-size-fits-all anonymization scheme that will work, so it is critical to have multi-level anonymization options.

It is our intention that *SCRUB-tcpdump* become a valuable tool for sharing packet traces without revealing sensitive information, however, it is also part of a larger goal of creating a comprehensive network data sharing infrastructure for security analysis. Logs are the fundamental unit of shared information between CERT teams and there are many different types of logs [19]. *SCRUB-tcpdump* for packet trace builds directly upon two multi-level anonymizations tools previously developed by the first author: (1) *CANINE* [4] for NetFlows and (2) *SCRUB-PA* [14] for process accounting. Our judgment is that this set of multi-level anonymization tools (packet traces, NetFlows, and process accounting) cover the majority of the security data to be shared. *CANINE* and *SCRUB-PA* have been available for Internet download with several reported "success stories" from their use. [1] *To our knowledge, no other tools provide multi-level anonymization for privacy-protected network data sharing.*

The remainder of this paper is organized as follows: Section 2 surveys previous work utilizing anonymization for packet trace sharing. Section 3 presents *SCRUB-tcpdump* from system architecture to the details of field multi-level anonymization

[1]two example success stories: (1) FIRST'06 Conference reported *CANINE* was successfully used in Finland to protect user privacy during a law enforcement investigation, IEEE CCGrid'06 reported *SCRUB-PA* was successfully used in Singapore to study HPC cluster workloads while protecting user privacy.

options. Section 4 reports experimental results with *SCRUB-tcpdump* demonstrating packet traces can be anonymized at different levels of privacy protection – a potential key to facilitating sharing between organizations. We end with a summary, conclusions, and future work in Section 5.

II. Related Work

We provide a brief survey of packet anonymization tools currently in use, for a more comprehensive background on sharing facilitated by anonymization of network data see [15].

The *tcpdpriv* IP and TCP/UDP header anonymization tool uses pseudorandom functions that cause IP address mappings to depend on traffic patterns and differ across traces [8]. This provides only coarse-grained subnet preservation and mappings are not consistent between packet traces. [11] proposes persistent anonymization of IP addresses between packet traces by merging part of the original address with a value encrypted with a key provided by the user.

TCPurify [2] is a packet sniffer/capture application that leverages user familiarity with *tcpdump* functionality and commands in the same way we do with *SCRUB-tcpdump*, however, our focus is different. While *TCPurify* is *tcpdump* with reduced functionality, *SCRUB-tcpdump* is *tcpdump* with enhanced functionality – with stronger multi-level anonymization options that have evolved (and been vetted) from anonymization on other data sources (NetFlows and process accounting).

SCRUB-tcpdump adapts anonymization algorithms previously developed in two tools, *CANINE* and *SCRUB-PA*. *CANINE* is an anonymizer (and format converter) for sharing NetFlow logs [4], [6]. NetFlows are concise descriptions of network connections including network endpoints, ports, time, bytes, number of packets, and TCP flags. NetFlows are one of the most common log types shared between organizational CERTs since they can characterize large traffic volumes and reveal many security events primarily through traffic analysis. *SCRUB-PA* is an anonymizer for sharing processing accounting logs [14], [7]. Unix/Linux systems collect information on individual/group usage and can record every process created by every user. This process accounting log data has proved especially useful in multi-user constrained environments (e.g., HPC clusters) to provide user audit trails for security analysis. From the usage of these two tools we have also made improvements. For instance, the *CryptoPAn* IP address anonymization flaw in *CANINE* has been corrected in *SCRUB-tcpdump*.

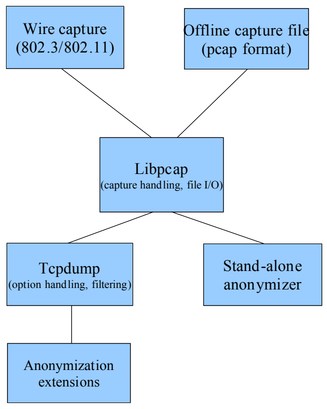
Pang and Paxson have two influential studies on packet trace anonymization. In [9] they implement a high-level programming environment for packet trace transformation using policy scripts to sanitize both network-protocol-level and payload data. They provide an example for FTP and verify the correctness of the transformation. [10] is a case study of using the *tcpmkpub* tool to make network data available. They provide the anonymization policy utilized and reveal many implementation details about performing anonymization on network data. There is discussion about anonymization of Ethernet addresses, IP addresses, ports, sequence numbers, and TCP timestamps for their particular environment but anonymization options for other environments are left open. In [5], Coull et al. show that sensitive information can be inferred when applying the anonymization policy in [10]. We feel this may be an unfair criticism since Pang and Paxson clearly state in [10] that they realize ports reveal services (for one example). However, Coull et al. do confirm the intuition that IP address anonymization alone is not enough to protect sensitive network information, multiple packet trace fields must be anonymized in coordination or sensitive information may be inferred.

Two tools provide generic frameworks for creating your own anonymization applications. *Anontool* provides an Anonymization API (AAPI) which allows users to write their own applications – tested examples include packet traces and NetFlows [1]. In [16] Slagell et al. present the *FLAIM* general framework for anonymization of log data. *FLAIM* users can create policies using XML to specify which fields are anonymized and in which way. Thus third party developers can add modules and policies to support any type of log data. For instance, the algorithms and policies of existing tools (e.g., *CANINE*, *SCRUB-PA*, *SCRUB-tcpdump*) can be configured in the general *FLAIM* framework with the creation of a parsing module (as needed) and an XML policy. The flexibility of both *Anontool* and *FLAIM* are also their major drawback since configuration programming (parsing, anonymization algorithms, and XML) may be necessary, however, as code is created and established for common applications they can be reused without programming.

Three papers outline the potential use of anonymized network data to enable new services. [20] describes how anonymized privacy-preserving network data will help overcome barriers to growth in the Managed Security Service Provider (MSSP) market by eliminating outsourcing risks to organizations. [17] describes the need to convey information about classified networks to researchers lacking clearances. There is a real need to understand how traffic on classified networks differs from traffic on unclassified networks in order to support network-centric warfare. The problems for sharing are the same for classified networks except with even more complexity.[2] They recommend a standard anonymization toolkit be developed to serve this purpose. [12] describes how a global analysis center may be enabled where logs (protected by anonymization) can be sent, threats discovered, and warnings disseminated – in other words a global MSSP. The challenges for establishing a global MSSP are formidable, especially establishing global centralized trust. *SCRUB-tcpdump* takes the exact opposite approach by providing a tool by which local trusted sharing can first be established between a small number of parties and then scaled as desired.

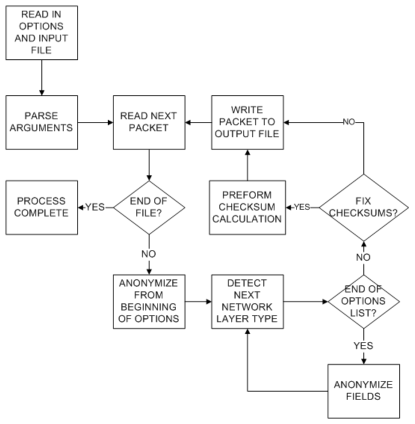
If data can be shared in an anonymized form that protects sensitive information and the same analysis results achieved then why would an organization share data any other way? The problem is that tradeoffs exist between the anonymization of data for privacy protection and the utility of anonymized

[2]One example of this additional complexity is the basic function of determining the national security classification of data on a classified network (SECRET/TOP SECRET etc.).



Wire capture (802.3/802.11)
Offline capture file (pcap format)
Libpcap (capture handling, file I/O)
Tcpdump (option handling, filtering)
Stand-alone anonymizer
Anonymization extensions

Fig. 1. *SCRUB-tcpdump* System Architecture



READ IN OPTIONS AND INPUT FILE
PARSE ARGUMENTS
READ NEXT PACKET
WRITE PACKET TO OUTPUT FILE
NO
PROCESS COMPLETE
YES
END OF FILE?
PREFORM CHECKSUM CALCULATION
YES
FIX CHECKSUMS?
NO
NO
ANONYMIZE FROM BEGINNING OF OPTIONS
DETECT NEXT NETWORK LAYER TYPE
END OF OPTIONS LIST?
YES
ANONYMIZE FIELDS

Fig. 2. *SCRUB-tcpdump* Data Flow

data for analysis. These trade-offs are different for different domains (medical, transportation, computer networks) and for different types of analysis. For the specific case of computer network security, log data is used by a variety of automated analysis tools to identify security events and it is logical to assume that as log data is anonymized for privacy protection that less security events may be identified. However, this trade-off has never been empirically tested. The remainder of this paper presents a new tool we developed for multi-level anonymization of packet traces and experimental results using this new tool to examine this exact trade-off.

III. *SCRUB-tcpdump*

Figure 1 shows the system architecture of *SCRUB-tcpdump*. We use the libpcap engine to handle I/O, this enables input from either live capture or a capture log file. Our system reads, parses, and passes packets sequentially one packet at a time.[3] Libpcap (as a stand-alone library) can be combined into a stand-alone *SCRUB-tcpdump* anonymizer or extended directly with extensions to *tcpdump*. We are in the process of extending *tcpdump* with patches and libraries for its released code. Functionality and output results will be the same using either stand-alone *SCRUB-tcpdump* or *tcpdump* extended with anonymization options – we have implemented the same command set in both. For the purposes of this paper, we report results from the stand-alone *SCRUB-tcpdump* since its implementation was quicker than extending *tcpdump* which is still in progress. Since libpcap is the foundation upon which *tcpdump* rests, we find it well-suited for linking directly into *tcpdump* source, expanding command line options, and allowing users to utilize *tcpdump* directly for file handling as well as anonymizing during capture.

A data flow diagram of *SCRUB-tcpdump* is shown in Figure 2. The system is executed with command line arguments specifying the input file and list of anonymization routines to be performed. The anonymization routine list is mapped to a controller block, the specified input file is loaded, and desired pcap filters applied. In the main loop, the next sequential packet is brought into memory and selected fields anonymized. This continues until there are no more packets or the user chooses to stop. During the anonymization process, the pcap header is initially scanned and if any pcap-specific information is selected to be anonymized (e.g timestamps) then the necessary routine is invoked. Since packet fields are encapsulated and thus may be dependent, we use a layered approach that anonymizes information at each layer and then examines the next layer until all packet fields are exhausted.

One of the major contributions of this work is that while other tools have focused primarily on IP address anonymization or only binary options for field anonymization, *SCRUB-tcpdump* allows a user to select multiple packet fields and each selected field has multiple anonymization options that may be applied specific to that field. Table I provides an overview of the packet fields we examined at different layers and the anonymization options we employed. The anonymization options are derived from two tools previously developed by the first author: (1) *CANINE* [4] for anonymizing NetFlows and (2) *SCRUB-PA* [14] for anonymizing processing accounting. For more technical details about the employed anonymization options see [6], [7].

Three new anonymization options in *SCRUB-tcpdump* are not derived from *CANINE* or *SCRUB-PA*:

1) Prefix-Preserving Pseudonymization for IP Addresses addresses a *CryptoPAn* flaw revealed by Brekne et al. in [3] and also detailed in [5]. It was found that *CryptoPAn* anonymizes IP addresses such that any bit in the anonymized address is dependent on all previous bits of the unanonymized address. This dependence causes a deanonymization to affect all addresses that share a prefix with the unanonymized address. To address this flaw we used the technique used in [10], [5] which breaks the dependency across bits by separately anonymizing the subnet and host portion of an IP address as independent blocks using a pseudo-random permutation.
2) Selective Black Marker over Payload provides an intermediate option between leaving the payload intact versus

[3]Plans are to parallelize packet handling in future versions.

deleting the payload entirely. With this option, common data types of known private information can be selected for deletion (e. g. filenames, URLs, etc.) although information may still be inferred from context.

3) Selective Black Marker over Entire Packet is new to provide flexibility and future extensibility. Any data type anywhere in a packet that can be identified by a regular expression can be specifically deleted.

IV. Experimental Results

The objective for experimentation is to quantitatively investigate the effect of *SCRUB-tcpdump* multi-level anonymization options on the tradeoff between privacy protection and security analysis in packet traces to be shared. General intuition leads us to believe that anonymization is a zero sum tradeoff between privacy and security – the more network data is obscured for protection the less value it may be for analysis.

The experimental design is to compare measurements of a data set before and after executing different *SCRUB-tcpdump* anonymization options. The metric used as a proxy for security analysis is IDS alarms. IDS alarms are not a perfect proxy for security analysis since the relationship is nonlinear – the more IDS alarms, the more security analysis may have taken place if new information is revealed by the new IDS alarms. However, more IDS alarms may also decrease ability to perform security analysis if new information is not revealed by the new IDS alarms. Given this non-linear relationship between IDS alarms and security analysis as well as the practical reality of false positive IDS alarms (which we ignore for these experiments), IDS alarms still provide a quantitative metric for security analysis for this paper that we intend to tune by addressing its shortcomings in future work.

We used the open-source snort IDS [13]. The ruleset utilized is the official "Sourcefire VRT Certified Rules (registered user release)" for version 2.2[4] with every rule turned on. This is the set of rules developed by the Sourcefire company that is continually kept up-to-date to include alerts for the newest and most critical security problems.

The data we used is part of *tcpdump* packet traces from the LBNL/ICSI Enterprise Tracing Project. We report tests from one relatively large file within this data set.[5] This file contains no payload data and was collected on the date specified in the filename from an actual enterprise environment. Snort processed 3,285,253 packets with the following breakdown:

TCP = 2,313,433 packets

UDP = 505,485 fragmented UDP packets discarded

IPX = 2,441 packets

ICMP = 751 packets

OTHER = 689 packets

ARP = 146 packets

We performed 25 different experiments generating 16 different figures. Due to space limitations only a subset of figures are presented here, however, results not shown are described in words. Our first set of experiments measures the effect of multi-level anonymization options on each field separately. To our surprise, the port options (black marker, bilateral) had no effect on the number of IDS alarms – we expected some snort alarms would be singularly mapped to port numbers for common malware services. This will be analyzed in more detail in future work. Figure 3 shows the TCP Flags options (black marker, key randomize) had no effect on the number of IDS alarms, randomize had a small effect toward more IDS alarms, and grouping had a large effect toward more IDS alarms (200%). The timestamp options (black marker, timeshift) had no effect on the number of IDS alarms and the enumeration option slightly decreased the number of IDS alarms (increasing privacy protection by a few percent) – we expected more snort alarms to have signatures based on timing correlation. The IP transport protocol option (black marker) had a large effect (600%) toward more IDS alarms, this is explained by noting that both TCP and UDP rules will fire when not specified. Lastly, IP address options (truncate, black marker, prefix-preserving) had no effect on the number of IDS alarms and random permutation had only a small effect toward more IDS alarms (less than 1%) – we expected more snort rules to correlate IP addresses to trigger alarms.

Throughout our experiments we found IP address anonymization options had little or no effect on IDS alarms. We suggest this may be a function of snort being a signature-based network IDS and would expect IP address anonymization may have an effect on anomaly-based network IDSs. This is an empirical counter example for the zero sum tradeoff assumption – privacy can be protected (in the form of IP address anonymization) without impacting security analysis.

The second set of experiments measures the effect of multi-level anonymization options using combinations of fields together. Since each individual field has some variation in security analysis versus privacy protection for different options, when combining fields we consistently select the option that provides the most alarms for each field. Combining two fields (port+another field), has a small effect on the number of IDS alarms when ports are combined with TCP flags (5%) and a larger effect toward more IDS alarms when ports are combined with IP transport protocol field (600%). Figure 4 shows results from combining three fields (port+TCP flags+another field). There is a small effect toward more IDS alarms for ports+flags+timestamps (7%), a medium effect toward more IDS alarms for ports+flags+IP address (15%), and a large effect toward more IDS alarms for ports+flags+IP transport (600%). Combining four fields (ports+flags+timestamps+another field) has a small effect toward more IDS alarms for ports+flags+timestamps+IP address (10%) and a large effect toward more IDS alarms for ports+flags+timestamps+IP transport (600%). Combining five fields (ports+ flags+timestamps+IP transport+IP address) results in a large effect toward more IDS alarms (600%) compared to the unanonymized data set.

In being consistent selecting options for the most IDS

[4]downloaded June 1st 2007

[5]`ftp://bro-ids.org/enterprise-traces/hdr-traces05/lbl-internal.20041004-1305.port002.dump.anon`

TABLE I
SCRUB-tcpdump SELECTABLE PACKET FIELDS AND CORRESPONDING ANONYMIZATION OPTIONS

PACKET FIELDS	DESCRIPTION and ANONYMIZATION OPTIONS
Ethernet "MAC" Address (source and destination)	MAC address encapsulated from data link layer designed to be unique for each Ethernet card. Options include left alone or set to 0.
Network Type	Flag to indicate network protocol included in payload. Options include left alone or set to 0.
PCAP Timestamp	Options include: left alone, black marker, random shift, and enumeration.
IP address (source and destination)	For this study we use only 4-byte IPv4 addresses, future versions will incorporate 16-byte IPv6 addresses. Options include: left alone, truncation, black marker, random permutation, keyed random permutation, and prefix-preserving pseudonymization.
IP Transport Protocol	Protocol used for this packet (e.g. ICMP=1, TCP=6, UDP=17). Options include left alone or set to 0.
IP Options	A field used for testing, debugging, and security. Options include left alone or set to 0.
IP Checksum	Recalculate IP checksum for consistency if other fields are altered. If disabled, original checksum is used. If enabled, checksum is recalculated.
Type of Service (TOS)	TOS has parameters for delay, throughput, reliability, and cost with one check bit. Redefined as Differentiated Services Code Point (DSCP).
Ports (UDP and TCP, Source and Destination)	Designates program endpoints between logical connections. Options include: left alone, black marker, and bilateral classification (above or below port 1024).
TCP Flags	Flags for TCP connection dynamics (FIN, SYN, RST, PSH, ACK, URG). Options include: left alone, black marker, randomization, keyed randomization, and grouping.
TCP Checksum	Recalculate transport layer checksum for consistency if other fields are altered. If disabled, original checksum is used. If enabled, pseudo TCP checksum is recalculated and placed into TCP checksum field.
Payload	Application data from higher layers. Options include: left alone, entire payload deleted (black marker), and common data types selectively black markered (IP addresses, host names, URLs, filenames, Email addresses, common people names, DNS queries, DHCP queries, ARP requests, HTTP/HTTPS headers, SMTP headers, RTP/RTCP information, FTP fields, IM information, etc.).
All Fields	*Selective Black Marker over Entire Packet* – specific information to be removed anywhere in the packet (all fields) can be selectively black markered with a properly constructed regular expression.

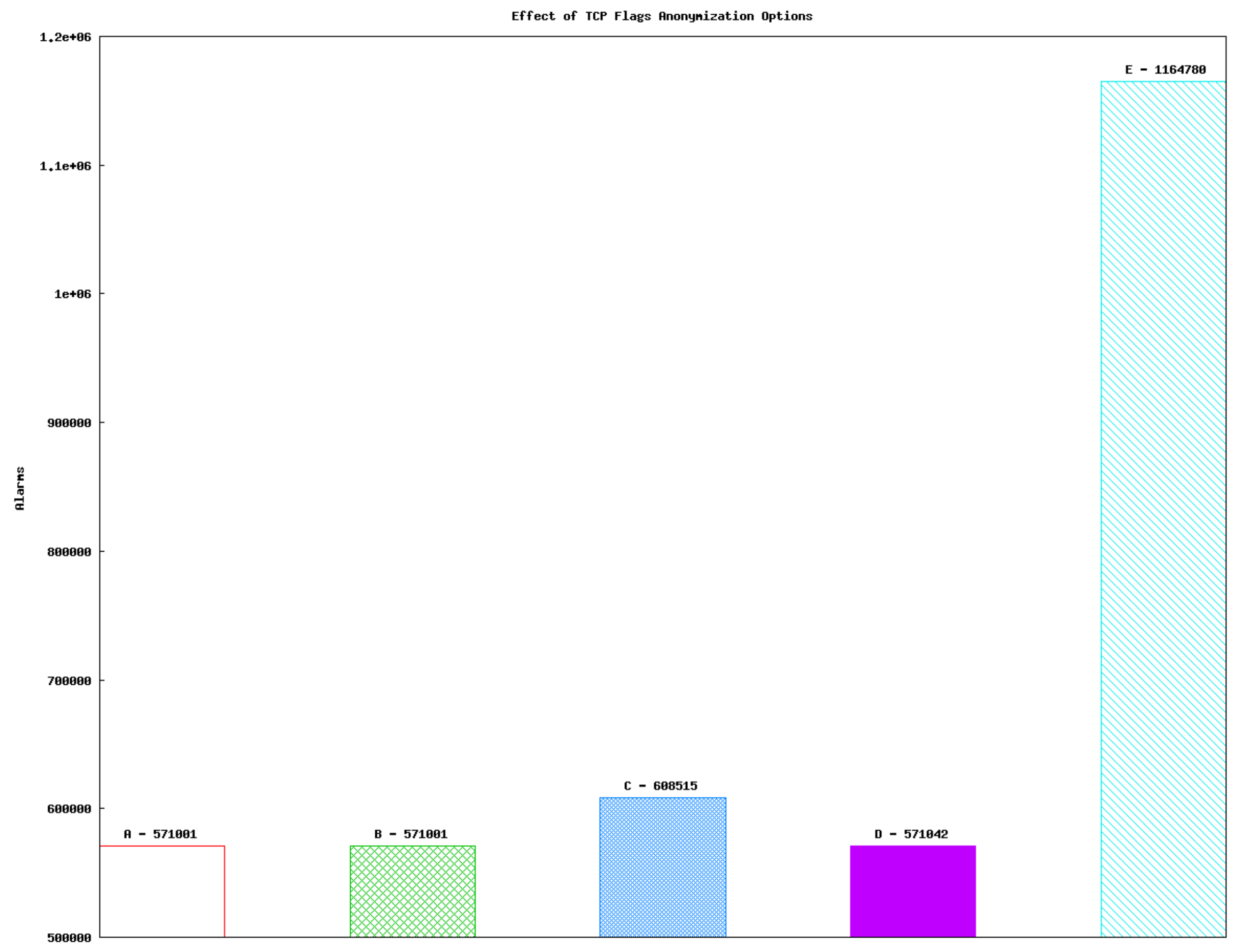


Fig. 3. Effect of TCP Flags Multi-Level Anonymization Options: [linear vertical axis is number of IDS alarms] (A) no anonymization; (B) black marker; (C) randomize; (D) keyed randomize; (E) grouping.

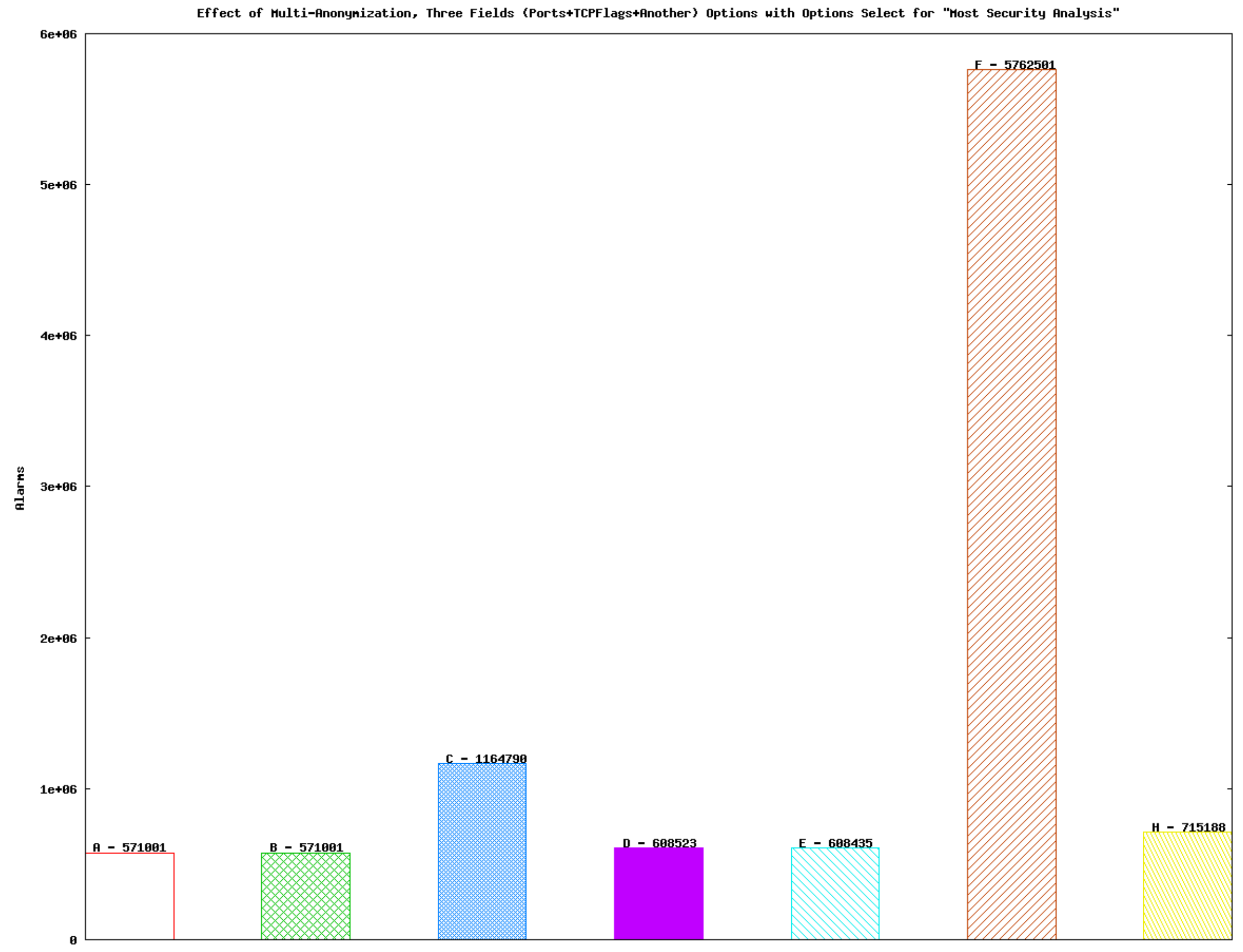


Fig. 4. Effect of Anonymizing Three Fields Simultaneously with a Bias Toward Selecting Options for "Most Security Alarms" Options: [linear vertical axis is number of IDS alarms] (A) no anonymization; (B) ports only; (C) TCP flags only; (D) ports and TCP flags; (E) ports, TCP flags, and timestamps; (F) ports, TCP flags, and IP transport; (G) ports, TCP flags, and IP address.

alarms we find that as more fields are simultaneously anonymized that more IDS alarms fire. This is a counter-intuitive result in that as data is obscured we would expect a trend toward less IDS alerts and thus less security analysis. Preliminary explanations for this effect are: (1) IP transport field anonymization options create so many IDS alarms they swamp other effects, (2) as more fields are obscured then IDS rules are no longer suppressed and thus fire, and (3) selecting options favoring more IDS alarms causes more IDS alarms to fire (a cascade effect). We will continue to investigate this to determine the exact cause.

The third/final set of experiments measures the effect of multi-level anonymization options using combination of fields together when consistently selecting the option that provides the most privacy protection for each field – in most cases this will be the black marker option. Combining two fields (ports+another field), has a little/no effect on the number of IDS alarms except for a large effect when ports is combined with IP transport protocol field (600% more IDS alarms). We find a consistent result when combining three, four, and five fields – no effect on IDS alarms except for when the IP transport field black marker option is included in which case there is a large effect. Again, IP transport field anonymization options appear to be dominant when biased toward selecting privacy protection options - an effect we would not have predicted.

Figure 5 is a summary of how selection of multi-level anonymization options effects the tradeoffs between security analysis and privacy protection. While selection bias toward anonymization options with more IDS alarms or more privacy protection is not meaningful for the case of two fields or five fields, we do find that selection bias makes a significant difference for the case of one field, three fields, and four fields as shown by the relative number of IDS alarms in the paired histograms as shown in Figure 5. This is empirical evidence that it does make a significant difference how fields are anonymized and that multi-level anonymization options can provide different security analysis and privacy protection for the same data set.

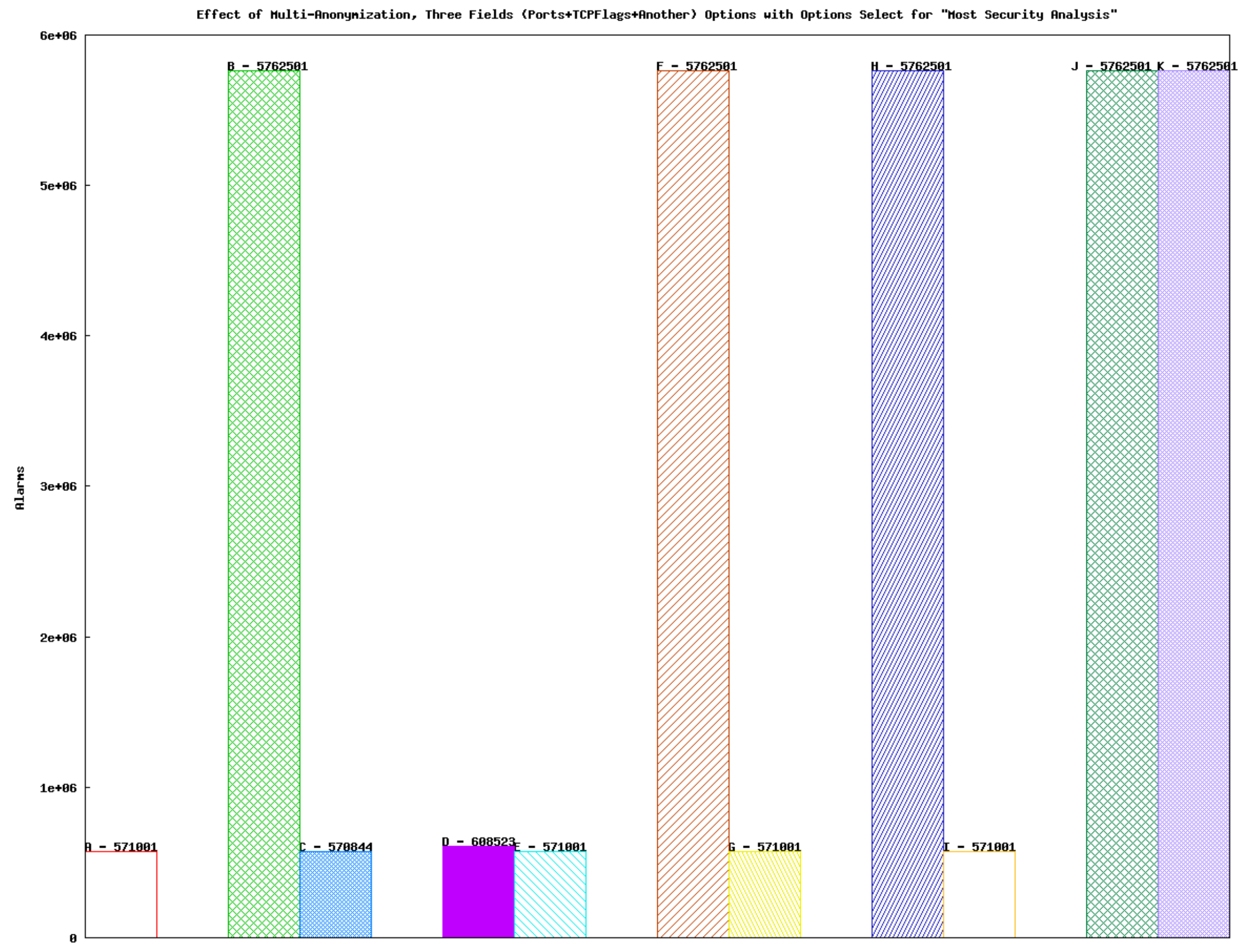


Fig. 5. Paired Comparison of Multi-level Anonymization Options – "Most IDS Alarms" Options versus "Most Privacy Protection" Options: [linear vertical axis is number of IDS alarms] (A) no anonymization; (B/C) 1-field Most Alarms/Most Privacy; (D/E) 2-field Most Alarms/Most Privacy; (F/G) 3-field Most Security/Most Privacy; (H/I) 4-field Most Alarms/Most Privacy; (J/K) 5-field Most Alarms/Most Privacy.

V. Conclusion

To facilitate sharing network data, we developed *SCRUB-tcpdump* which adds anonymization features to the popular *tcpdump* tool so packet traces can be shared for analysis without sensitive information being divulged. *SCRUB-tcpdump* provides the unique capabilities to: (1) allow user selection to anonymize multiple fields within a packet trace, and (2) allow user selection of multi-level anonymization options within each selected field. To our knowledge, none of the previous research applying anonymization techniques to network data has implemented and demonstrated multi-level anonymization options. Multi-level anonymization options are critically important because the reality is that organizations have different security policies requiring different levels of privacy protection and thus different requirements for anonymization. While one possible approach is to protect privacy of data from all organizations at the highest possible level, there is a tradeoff between privacy protection and security analysis such that the strongest privacy protection anonymization options may significantly degrade (or even eliminate) any insights from desired security analysis.

The primary novel contribution of this work is the results from the use of *SCRUB-tcpdump* on network data demonstrating the tradeoffs between privacy protection and security analysis. First we show that this tradeoff is not always a zero sum tradeoff. For example, different IP address anonymization options for privacy protection have little impact on security analysis (in the context we tested) so it is indeed possible to simultaneously satisfy both privacy protection and security analysis requirements. Second we show that selection of different multi-level anonymization options on the same fields can drastically change the output information available. For example, we found the number of IDS alarms generated from anonymized network data was about an order of magnitude different depending if the user selected anonymization options biased toward maximizing privacy protection or IDS alarms. Thus we demonstrate that multi-level anonymization options are needed to tailor network data sharing between organizations in order to manage the risk of valuable network data either being needlessly lost (privacy protection too stringent) or needlessly disclosed (privacy protection not stringent enough).

Future work will focus on *SCRUB-tcpdump* performance measurements, reporting feedback from the use of *SCRUB-tcpdump* in production environments, and quantitatively measuring the strength of *SCRUB-tcpdump* against adversary deanonymization attacks.

VI. Acknowledgments

The authors would like to thank anonymous SECOVAL'07 peer reviewers for their insights which we have incorporated to improve this paper.

References

[1] *Anontool*. `http://dcs.ics.forth.gr/Activities/Projects/anontool.html`

[2] E. Blanton, TCPurify - "a sanitary sniffer", `http://irg.cs.ohiou.edu/ eblanton/tcpurify/`

[3] T. Brekne, A. Ames, and A. Oslebo, Anonymization of IP Traffic Monitoring Data – Attacks on Two Prefix-Preserving Anonymization Schemes and Some Proposed Remedies, *Workshop on Privacy Enhancing Technologies (PET)*, 2005.

[4] *CANINE*: Converter and ANonymizer for Investigating NetFlow Events, `http://security.ncsa.uiuc.edu/distribution/CanineDownLoad.html`

[5] S. E. Coull, C. V. Wright, F. Monrose, M. P. Collins, and M. K. Reiter, Inferring Sensitive Information from Anonymized Network Traces, *Network and Distributed Systems Security (NDSS)*, 2007.

[6] Y. Li, A. J. Slagell, K. Luo, and W. Yurcik, *CANINE*: A Combined Converter and Anonymizer Tool for Processing NetFlows for Security, *13th Intl. Conf. on Telecom. Systems*, 2005.

[7] K. Luo, Y. Li, C. Ermopoulos, W. Yurcik, and A. J. Slagell, *SCRUB-PA*: A Multi-Level Multi-Dimensional Anonymization Tool for Process Accounting, *ACM Computing Research Repository (CoRR) Technical Report cs.CR/0601079*, January 2006.

[8] G. Minshall, *tcpdpriv*. `http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html`

[9] R. Pang and V. Paxson, A High-Level Programming Environment for Packet Trace Anonymization and Transformation, *ACM SIGCOMM*, 2003.

[10] R. Pang, M. Allman, V. Paxson, and J. Lee, The Devil and Packet Trace Anonymization, *ACM Computer Communications Review*, January 2006.

[11] M. Peuhkuri, A Method to Compress and Anonymize Packet Traces, *ACM SIGCOMM Internet Measurement Workshop*, 2001.

[12] P. A. Porras, Privacy-Enabled Global Threat Monitoring, *IEEE Security and Privacy*, November 2006.

[13] M. Roesch, Snort: Lightweight Intrusion Detection for Networks, `http://www.snort.org/`

[14] *SCRUB-PA*: A Tool for Multi-Level Anonymization of Process Accounting Logs. `http://security.ncsa.uiuc.edu/distribution/Scrub-PADownLoad.html`

[15] A. J. Slagell and W. Yurcik, Sharing Computer Network Logs for Security and Privacy: A Motivation for New Methodologies of Anonymization, *1st IEEE SECOVAL Workshop*, 2005.

[16] A. J. Slagell, K. Lakkaraju, and K. Luo, *FLAIM*: A Multi-level Anonymization Framework for Computer and Network Logs, *Usenix LISA*, 2006.

[17] R. Stapleton-Gray and S. Gorton, Rendering the Elephant: Characterizing Sensitive Networks for an Uncleared Audience, *7th IEEE "West Point" Workshop on Information Assurance*, 2006.

[18] Tcpdump Libpcap Official Site, `http://www.tcpdump.org/`

[19] X. Yin, K. Lakkaraju, Y. Li, and W. Yurcik, Selecting Log Data Sources to Correlate Attack Traces For Computer Network Security: Preliminary Results, *11th Intl. Conf. on Telecom. Systems*, 2003.

[20] J. Zhang, N. Borisov, and W. Yurcik, Outsourcing Security Analysis with Anonymized Logs, *2nd IEEE SECOVAL Workshop*, 2006.