On the Privacy Risks of Publishing Anonymized IP Network Traces

D. Koukis[1], S. Antonatos[1], and K.G. Anagnostakis[2]

[1] Distributed Computing Systems Group, FORTH-ICS, Greece
{koukis, antonat}@ics.forth.gr
[2] Infocomm Security Department, Institute for Infocomm Research, Singapore
kostas@i2r.a-star.edu.sg

Abstract. Networking researchers and engineers rely on network packet traces for understanding network behavior, developing models, and evaluating network performance. Although the bulk of published packet traces implement a form of address anonymization to hide sensitive information, it has been unclear if such anonymization techniques are sufficient to address the privacy concerns of users and organizations.

In this paper we attempt to quantify the risks of publishing anonymized packet traces. In particular, we examine whether statistical identification techniques can be used to uncover the identities of users and their surfing activities from anonymized packet traces. Our results show that such techniques can be used by any Web server that is itself present in the packet trace and has sufficient resources to map out and keep track of the content of popular Web sites to obtain information on the network-wide browsing behavior of its clients. Furthermore, we discuss how scan sequences identified in the trace can easily reveal the mapping from anonymized to real IP addresses.

1 Introduction

Packet-level traces of Internet traffic are widely used in experimental networking research, and have been proved valuable towards understanding network performance and improving network protocols (c.f. [18,13,11,12]). Since raw packet traces contain sensitive information such as emails, chat conversations, and Web browsing habits, organizations publishing packet traces employ techniques that remove sensitive information before making the traces available.

The most common process of "anonymizing" a packet trace involves removing packet payloads and replacing source and destination IP addresses with anonymous identifiers [14]. Several repositories of trace datasets have been made available that employ this technique and they have served the research community extremely well for many years[2,3]. Without payloads and host IP addresses, it is widely assumed that it is hard, if not impossible, to elicit any sensitive information from a packet trace. However, several researchers have recently expressed concerns, albeit without further investigation, that there *may* be ways of indirectly *inferring* sensitive information from sanitized traces[20,16]. This kind of

H. Leitold and E. Markatos (Eds.): CMS 2006, LNCS 4237, pp. 22–32, 2006.

information on individual users and organizations could be used in many ways that may be illegal or simply in a bad way (e.g., for marketing, surveillance, censorship, industrial espionage, etc.).

In this paper, we attempt to experimentally assess the risk of publishing "anonymized" packet traces. In particular, we examine whether it is possible to break the address anonymization scheme and identify specific host addresses in the network traffic packet trace. We focus on two potential statistical identification attacks: one using known web site footprints, and one using known patterns of port-scanning activity. The first attack has the potential of discovering the anonymized identifiers of known Web servers, which can be linked to (still anonymous) client IPs. The second attack is not restricted to a specific type of host or protocol, and is therefore "ideal" for recovering client IPs which cannot be recovered using the first technique alone.

2 Related Work

Although our work is not the first to raise questions about the risks of publishing anonymized packet traces, to the best of our knowledge it is the first report that tries to provide some answers. The issue was first raised by Ylonen[20], who discussed several ways of breaching the privacy of traces that have been anonymized using *tcpdpriv*[14]. This report also mentions matching known website footprints to the packet trace. The same attack is also imagined by Pang and Paxson in [16]. In both cases, the authors did not further examine the technical details of the attack and did not experimentally quantify its potential effectiveness.

Launching a "known-plaintext" attack on web-site footprints found in packet traces is similar in many ways to launching such an attack on footprints found in encrypted Web traffic, as seen by a Web proxy. For the case of the proxy server, Sun et al.[19] developed a technique that monitors a link between proxy and clients, gathers HTTP requests and responses, and compares them against a database of signatures. The authors show that most of the Web pages in their database can be identified and that false positives are rare. Similar results are presented by Hintz in [10]. Moreover, Pang et al. in [17] mention the problem of sequential scans found in traces that could lead to a potential disclosure of the anonymized IP addresses.

Although the basic idea of launching a known-plaintext attack on encrypted Web proxy traffic is similar to attacking a packet trace, there are two major differences that justify further investigation. First, attacking packet traces instead of proxy traffic seems a lot easier and thus more threatening, as anonymized packet traces are widely available through organizations such as NLANR[4] In contrast, web proxy logs are usually not shared outside an organization. Thus, the technique examined in this paper is an existent threat to the privacy of users.

Second, it seems difficult to directly apply the proxy technique to packet traces. We have identified several issues that need to be addressed in attacking a packet trace that were not explored in the proxy attack. For instance, caching

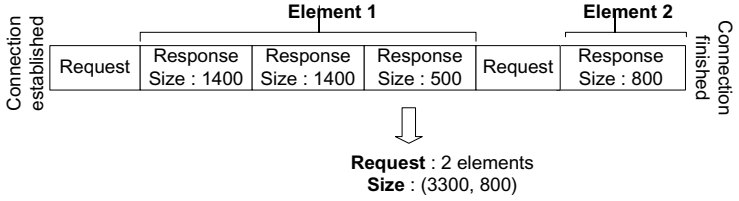

Fig. 1. Extracting elements from a packet trace. The responses between two successive requests belong to a single element (Element 1) and the element size is the sum of response packets. The connection termination also indicates the end of an element (Element 2).

schemes, the use of cookies, the type of browser used to make the request and various HTTP options and protocol details alter the underlying input and may influence the effectiveness of the method. Another problem that we faced in our analysis is the reconstruction of the HTTP data from the TCP/IP traces. This was not the case in Web proxy traffic where there is no need for HTTP-level reconstruction as the complete request sequence for a page is available. This issue affects the matching process and should be taken into account.

3 Identification of Anonymized Web Server Addresses

3.1 Extracting Web Signatures

Before describing the process for extracting HTTP requests from packet traces, it is important to understand exactly how the protocol under attack is implemented. For HTTP protocol version 1.0[7], a new connection should be created with the web server in order to retrieve a new element. The term element defines every item that contitutes a web page, which may include html pages, images, css and script files and in general everything that can be referenced in a web page. To reduce the delay perceived by the client during web browsing and additionally the traffic produced, HTTP version 1.1[9] introduced *persistent connections*. A persistent connection does not need to be terminated after retrieval of an element but can be reused to process further requests to the same server. Current browsers use HTTP/1.1 by default, although the user can specify to use HTTP 1.0. Typically, when a browser requests an HTML page, the retrieved page is parsed in order to find its embedded objects. Afterwards, an HTTP request is issued for each one of these objects. Responses sent by the server consist of the HTTP headers along the actual data that were requested.

The first step for signature extraction is to identify web traffic from traces. This is is relatively easy by taking packets to and from port 80. Packets with zero payload size are considered as acknowledgments. The second step is to analyze HTTP responses. In case of HTTP/1.0, a request is made, the response follows and then connection terminates. Thus, the size of the element is the

sum of payload sizes of packets with source port 80 (for this flow). In case of HTTP/1.1 (default for most browsers), we may have multiple requests on the same connection. All the packets from server to client that are found between two requests belong to the same response and this response belongs to the first request. The process is illustrated in Figure 1. In this study we do not consider "HTTP pipelining" – this would somewhat complicate our analysis, but is outside the scope of this work, since it is still an experimental feature, disabled by default in all browsers, and rarely used.

Having collected a set of elements, we need to assign them to the web page they belong. Elements are grouped on a per-client basis. It is necessary to distinguish successive Web page requests and assign the identified elements to the right page. A web page may contain elements from multiple web servers. Often, HTML files reside on the main web server, while images or embedded objects (like ads or banners) are loaded from another server. This behavior does not allow us to rely on IP address for assigning elements. Another problem arises when there are two requests for the same web server, e.g., when a user clicks a hyperlink for a page on the same domain as the original web page. Given that we have no HTTP-level information, we have to rely on a heuristically determined timeout value. After a user downloads a page, we assume that there is an "idle" period until the user clicks a link to another page. Figure 2 shows the process of assignment.

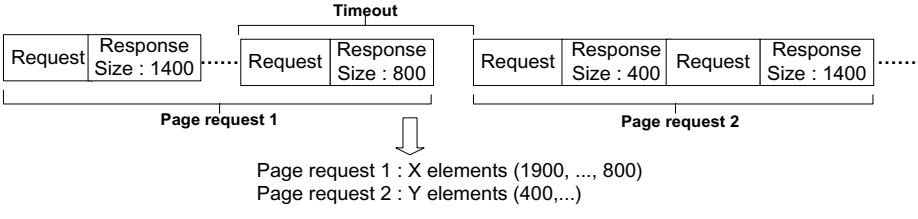

Fig. 2. Assigning elements into page requests. The timeout value is used to separate requests for two different pages.

After the assignment of elements to Web pages, a signature for that page can be created. As we have no payload from traces, the element size is the most promising piece of information that we can extract. Note that the size of each element is computed including the HTTP response headers, which causes variation of element size even for subsequent requests (due to variable length fields like cookies and date). Consequently, the signature for a web page is the set of element sizes that is consisted of. This definition for web page signature creation has several advantages but also limitations. As the number of elements forming a page increases, signatures become more accurate. However, it is possible for a web page to generate more than one signature for three main reasons. First, the web page may be dynamic or changing periodically. Dynamic pages may alter the actual content of the page (HTML file) and the embedded objects, for example, to present different ads each time. Second, due to HTTP headers

length variation, even for static pages we may get different signatures. Finally, caching may affect the creation of the signature and provide different results for the same page.

3.2 Web Server Fingerprinting Methodology

A fingerprint attack can be formed as follows. The first step is to obtain a signature for each target page that is to be identified. As mentioned before, a signature for a page may not be constant and thus, using information just from one signature may reduce the probability of a successful match. It is possible to obtain many signatures for the same page and extract all the elements into a unified set. The members of this set are most likely to be found in the trace as a subset of a complete request. Additionally, in order to compensate for minor HTTP header variations that may occur, a minor padding (ranging from 8 to 32 bytes) could be applied to the size of the elements. We should note here that signatures should have been created at approximately the same time as the trace packets because even static web pages may change over time. Moreover, if a target site is dynamic, multiple signatures should be created in order to match all possible appearances of the page.

After the signature database has been created, HTTP elements should be extracted from the packet trace and grouped into page requests. Since the adversary has no way to find out whether HTTP reconstruction was successful and up to which degree, this step can be done multiple times using different timeout values.

The next step is the matching process. Information extracted from the trace should be compared against the signatures contained in the database. The elements extracted from the trace are compared against the elements of each signature and a similarity score is computed. The similarity score is the percentage of common elements between the request on the trace and the signature. If the score is above a threshold then we have a potential match. As each web site has multiple signatures, if we have more than one matche for a site, our guess is more confident. If the request on the trace matches multiple web sites, then the site that gives the highest scores is selected as a possible match. The complete fingerprinting methodology is shown in Figure 3.

3.3 Experimental Results

A key parameter of the fingerprinting process is how signatures change over time. Initially, ten thousand different pages were collected from Google's web directory [1], which was choosed due to the large number of indexed pages. Pages were chosen randomly from various categories and only a few of them belong to popular web servers. For each page, its signature was generated multiple times in different time periods.

In our first experiment, we examined whether collected signatures remain constant over time. For each target site, we created its signature every half an hour. We target to calculate the percentage of static pages in the web, as an indication of whether the fingerprinting method can be applied succesfully.

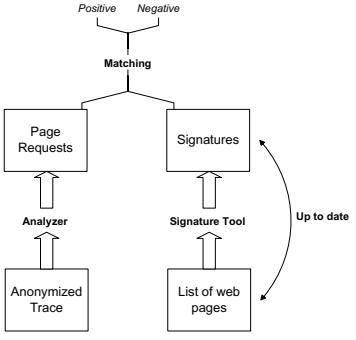

Fig. 3. Fingerprinting attack methodology

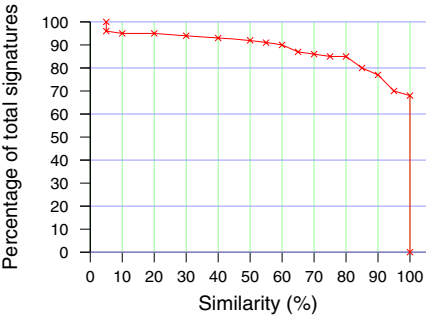

Fig. 4. Similarity among signatures

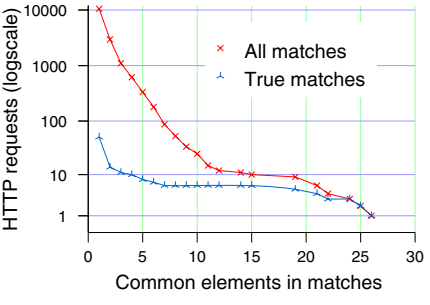

Fig. 5. Number of HTTP requests that share common objects with those found in the signature of target page

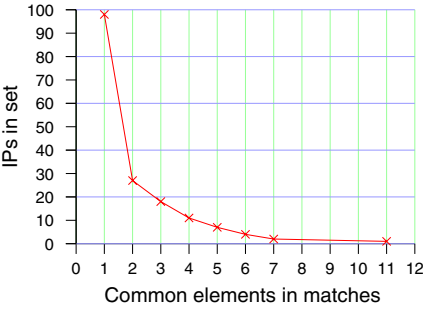

Fig. 6. Number of distinct IP addresses that exist in matches and share common elements with the target signature

Previous works [8] show that about 56% of the total pages appear to be static. In Figure 4, the similarity percentage for different signatures of the same page is shown. It can be observed that 67% of page signatures remain constant, while 90% of them have more than 60% similarity. It is clear that there still exist a large portion of static pages that give potential for our method to work.

In our second experiment, requests found in a trace collected at a local subnet of our institute were compared against the signature of the main page for a popular web server. The signature contained 27 elements and we had in our disposal both the anonymized and non-anonymized form of the trace to verify our results and identify true matches. In Figure 5, the number of requests that match the signature as a function of the number of common elements is presented. As we increase the number of common elements, the number of requests that match the signature decreases. Furthermore, near the maximum value of common elements (27) false positive ratio, e.g. potential matches against true matches, drops to zero.

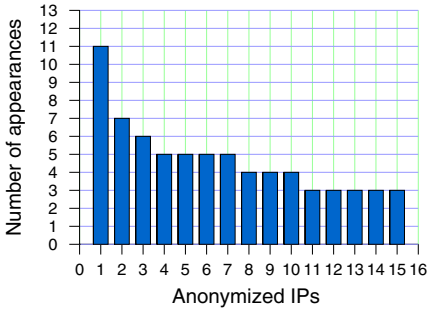

Fig. 7. Histogram of anonymized IP addresses found in requests with common elements to the target signature

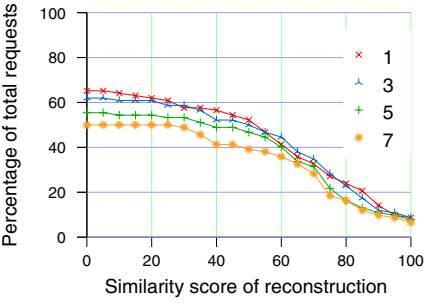

Fig. 8. Efficiency of the reconstruction process for assigning elements into sets

It is possible that the algorithm does not always succeed to come up with one result for a signature. Instead, two or more matches may contain the same number of common elements with multiple signatures. In this case the true match can be decided based on the number of times that each match has been found. This way, we try to find many matches to the target signature that each one has a few common elements. The first step of this process is to group requests into sets based on the number of common elements with the target signature and count distinct IP addresses found in these requests. Figure 6 shows the number of IP addresses in each of these sets. When the number of common elements reaches its maximum value the number of IP addresses remains almost constant so there is a precise guess.

In Figure 7, the histogram of the fifteen most popular IP addresses in the matches is shown. The IP address IP1 is by far the most frequently appearing address (considering that the number of web requests in the trace is limited to few). After evaluating this experiment with the non-anonymized trace, we have found that this IP was the right mapping for the target site.

In order to improve confidence of the guess, we could correlate the results of the previous methods. If both methods end up to the same result, that means identifying the IP address which has more common elements than all other requests and also is the most popular, our guess can be more confident.

As the techniques described previously depend on the information extracted from traces, we evaluated the HTTP reconstruction process. More precisely, we measured the correctness of assigning elements into Web pages. We analyzed the non-anonymized trace using HTTP level information and extracted the Web page along with their elements. Then the algorithm of assignment was applied to the anonymized trace for multiple timeout values. In Figure 8, we present the percentage of requests found in trace that have been reconstructed correctly, for timeout values 1, 3, 5 and 7 seconds. The similarity score of the reconstruction is computed as the percentage of elements of the request that have been identified using our technique compared to the number of real elements. As we can see

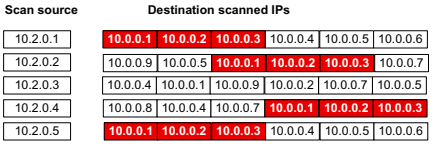

Fig. 9. Passive scan-based identification tries to identify the MCS in the SYN packets (marked as red)

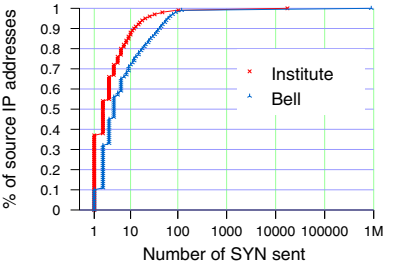

Fig. 10. CDF of SYN packets sent by hosts in BELL trace and in a trace captured at our institute

about 8% of the requests can be correctly reconstructed and this is common for all timeout values. Although this is a rather low percentage, the existence of these requests is an evidence that matching can be partially successful. Also, requests that remain constant regardless the timeout value are more probable to have been correctly reconstructed and matching should be based on these.

4 Passive Scan-Based Identification

It is trivial for an attacker to discover the real IP addresses from an anonymized trace if he can inject a scan to a specific IP address range. Given that the attacker knows the range that is scanned, and assuming he is able to identify his own packets in the trace, obtaining the mapping of any anonymized IP address to the real address is straightforward. However, such attempts may also be easy to detect, hereby exposing the attacker. In some cases, the attacker may not be willing to take such a risk. In this Section we present a more stealthy attack that does not require the attacker to inject any traffic in the trace.

Instead of relying on injected scans, our algorithm relies on passive identification and analysis of existing scans performed by others, using tools such as nmap[5]. Scans found in packet traces have different forms: some are linear scans targeting specific subnets, others are random scans within the same subnet, others are completely randomly generated IP addresses throughout the Internet. The number of elements in a scanning sequence may also vary, although it is not uncommon for such tools to map whole subnets.

To illustrate the basic operation of our algorithm we first describe a simple scenario. Assume a target /24 subnet a.b.c.X/24 for which the anonymized IPs need to be mapped to real IPs, and a trace that contains only full scans across the subnet. The scans are either linear or random. If there are at least two scan sequences that are identical, we know that with very high probability these scans are sequential. The sequential scans then directly provide us with the mapping of the /24 subnet, as the sequence identified in the anonymized trace can be mapped to addresses a.b.c.1 through a.b.c.254.

This observation can be generalized to any set of scan sequences. We assume that some of the sequences are in random order and some of the sequences are linear. Our goal is to identify maximum common subsequences between different scans. The longer the common subsequence, the more likely it is that it represents a linear scan. Since short sequences are more likely to be coincidental, we cannot simply consider every pairwise match as legitimate. Instead, we iteratively construct the complete map of a subnet from pairwise matches in order of confidence, and look for the maximum consistent set of pairwise matches that cover the whole subnet.

To evaluate our approach, we used a network trace collected from two /24 subnets, locally at our institution. The duration of the trace was 4 days and it contained header-only information. We had at our disposal both the anonymized and non-anonymized version. The first step is to recognize the source IP addresses that perform scanning. A simple heuristic is to select only these hosts that send a large number of SYN packets. We measured the cumulative distribution function of SYN packets sent by hosts in both our trace and a network trace from Bell Labs which had the same duration with our trace. The results are summarized in Figure 10. It can be observed that only 1% of the hosts generate more than 80 SYN packets in the whole duration of the trace, thus leaving us only a small percentage of hosts to be investigated. For our next experiment, we used 80 as the threshold for selecting hosts that are considered as scanners.

After having selected the set of source IP addresses to check, we try to find the longest subsequence of destination hosts that they sent SYN packets to. We found on the trace two IP addresses that shared a subsequence of 512 destination hosts. After looking at the non-anonymized traces we verified that these two IP addresses were scanning linearly a local subnet, sending two SYN packets per host, apparently using the nmap tool. Although this is a specific example with a large common subsequence, it demonstrates the effectiveness of our technique as it is indicative of how our approach would work on real traffic. In lack of a reasonably large trace with both anonymized and real addresses, we were unable to evaluate in more detal the identification of linear scans through smaller common subsequences where the probability of false matches is higher. However, we believe that long linear scans are very likely to occur even within shorter timescales, thus enabling the straightforward application of our algorithm with high confidence.

5 Concluding Remarks

In this paper we have examined whether an adversary can break the privacy of anonymized network packet traces. Such a threat is significant, as there are organizations that publish packet traces from their networks for research purposes.

We have examined two attacks: one that identifies web servers in the trace through known web site fingerprints, and one that attempts to recover the original IP addresses in the trace from well-structured port-scanning activity. Associating the inferred information with other sources, such as web server's log files,

could lead to significant privacy problems. Our results show that these attacks are reasonably effective, despite being inexact and error-prone.

Although our results are not particularly surprising, they provide solid experimental evidence confirming the concerns previously expressed by other researchers. One interesting observation is that the attack may be more complex and error-prone than previously thought. However, it is not unlikely that careful engineering can lead to higher identification accuracy compared to our results.

Since real-world datasets are essential for research, our results also reinforce the need for alternatives to publishing sanitized packet traces. One possible direction is the use of systems that have access to the raw traces but only allow access to them through a query interface[6,15]. Since the system controls the queries, it may be possible to control their privacy implications, or at least retain an audit trail in case a privacy violation is discovered at a later point.

Acknowledgments

This work was supported in part by the IST project LOBSTER funded by the Europen Union under Contract No. 004336 and the GSRT project Cyberscope funded by the Greek Secretariat for Research and Technology under the Contract No. PENED 03ED440. We would also like to thank Kostas Anagnostakis for his insightful comments. Spiros Antonatos is also with the University of Crete.

References

1. Google's directory. http://directory.google.com.
2. The internet traffic archive. http://www.acm.org/sigs/sigcomm/ITA.
3. NLANR network traffic packet header traces. http://pma.nlanr.net/Traces/.
4. Nlanr passive measurement and analysis. http://pma.nlanr.net/PMA/.
5. Remote OS detection via TCP/IP Stack FingerPrinting. http://www.insecure.org/nmap/nmap-fingerprinting-article.html, June 2002.
6. K. G. Anagnostakis, S. Ioannidis, S. Miltchev, J. Ioannidis, Michael B. Greenwald, and J. M. Smith. Efficient packet monitoring for network management. In *Proceedings of the 8th IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pages 423–436, April 2002.
7. T. Berners-Lee, R. Fielding, and H. Frystyk. RFC 1945: Hypertext Transfer Protocol — HTTP/1.0, May 1996.
8. B. E. Brewington and G. Cybenko. How dynamic is the Web? *Computer Networks (Amsterdam, Netherlands: 1999)*, 33(1–6):257–276, 2000.
9. R. Fielding, J. Gettys, J. Mogul, H. Nielsen, and T. Berners-Lee. Hypertext transfer protocol - HTTP/1.1. *RFC 2616*, June 1999.
10. A. Hintz. Fingerprinting websites using traffic analysis. In *Privacy Enhancing Technologies (PET 2002)*, pages 171–178. Springer-Verlag, LNCS 2482, 2002.
11. H. Jiang and C. Dovrolis. Passive estimation of tcp round-trip times. *Computer Communications Review*, July 2002.
12. S. Jin and A. Bestavros. Sources and characteristics of web temporal locality. In *MASCOTS*, pages 28–35, 2000.

13. M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM Computer Communication Review*, 27(3), July 1997.
14. G. Minshall. Tcpdpriv: Program for eliminating confidential information from traces, 2005. http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html.
15. J. Mogul. Trace anonymization misses the point. Presentation on WWW 2002 Panel on Web Measurements.
16. R. Pang and V. Paxson. A High-Level Programming Environment for Packet Trace Anonymization and Transformation. In *Proceedings of the ACM SIGCOMM Conference*, August 2003.
17. Ruoming Pang, Mark Allman, Vern Paxson, and Jason Lee. The devil and packet trace anonymization, January 2006.
18. V. Paxson and S. Floyd. Wide-area traffic: the failure of Poisson modeling. In *Proceedings of ACM SIGCOMM*, pages 257–268. August 1994.
19. Q. Sun, D. R. Simon, Y. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu. Statistical identification of encrypted web browsing traffic. In *Proceedings of IEEE Symposium on Security and Privacy*, Oakland,CA, May 2002.
20. T. Ylonen. Thoughts on how to mount an attack on tcpdprivs "-a50" option. http://ita.ee.lbl.gov/html/contrib/attack50/attack50.html.