Extracting Attack Sessions from Real Traffic with Intrusion Prevention Systems

I-Wei Chen[1], Po-Ching Lin[1], Chi-Chung Luo[1], Tsung-Huan Cheng[1], Ying-Dar Lin[1], Yuan-Cheng Lai[2] and Frank C. Lin[3]
Department of Computer Science[1]
National Chiao Tung University, 300 Hsinchu, Taiwan
{iwchen, pclin, ccluo, thcheng, ydlin}@cis.nctu.edu.tw
Department of Information and Management[2]
National Taiwan University of Science and Technology, 106 Taipei, Taiwan
laiyc@cs.ntust.edu.tw
Cisco Systems Inc.[3]
Cisco, San Jose, USA
fclin@cisco.com

***Abstract*—False Positive (FP) and False Negative (FN) happen to every Intrusion Prevention System (IPS). No one could do better judgment than others all the time. This work proposes a system of Attack Session Extraction (ASE) to create a pool of traffic traces which cause possible FNs and FPs to IPSs. Developers of IPSs can use these traffic traces to improve the accuracy of their products. First, the ASE captures real traffic and replays captured traffic traces to multiple IPSs. From the logs of IPSs, we can find that some attack events are only "logged" or "not logged" at certain IPS. The former could be FPs, while the latter could be FNs to that IPS. The ASE then starts to extract this "suspicious" traffic from replayed traffic traces. The extracted traffic traces can be used for further analysis by IPS developers. Some of the traces may prove to be "guilty", while some are not, i.e. leading to FPs or FNs. To completely extract a "suspicious" session, the ASE uses an association mechanisms based on "anchor packets", "five-tuple and time", and "similarity" for the first packet, first connection, and whole session, respectively. It calculates the degree of similarity among packets to extract a "suspicious" session containing multiple connections. We define "variation" and "completeness and purity" as the performance indexes to evaluate ASE. The experiments demonstrate that 95% of extracted sessions have low variation, and the average completeness and purity is up to 80%.**

***Index Terms*—False Positive, False Negative, Intrusion Prevention, Intrusion Detection, Packet Trace, Session Extraction, Similarity**

I. Introduction

It is a challenge for Intrusion Prevention Systems (IPS) to defend a network since malicious traffic on the Internet keeps growing and changing. For example, rules of SNORT (www.snort.org/vrt/) are updated frequently. Many IPS vendors assign dedicated engineers to analyze malicious traffic and write appropriate signatures into their products' database. However, False Positives (FPs) and False Negatives (FNs) might still happen. An IPS may detect an attack in a traffic trace, while other IPSs do not, but the detection might be a FP. Sometimes an IPS may miss an attack in a session, so it has a FN. If we can provide traffic traces that cause the inaccuracy for the IPS developers, the quality of the signature database will be improved.

We design a system of Attack Session Extraction (ASE) to integrate efforts of signature analysis and development from different vendors. The ASE captures, replays, and extracts real traffic. First, it captures real traffic from the mirror port of an Ethernet switch. The captured traffic traces are usually in a PCAP format (www.tcpdump.org) and contain various applications such as FTP, HTTP, SMTP, P2P and IM. Second, it replays traffic traces to IPSs of different vendors. By comparing the log on each IPS, possible FPs and FNs to a certain IPS can be found. From the logs of IPSs, we can discover that some attack events are only "logged" or "not logged" at a certain IPS. The former could be FPs, while the latter could be FNs to that IPS. Third, the ASE extracts "suspicious" traffic that triggers the possible FPs and FNs, and provides it to developers for enhancing their IPS's accuracy.

Our main concern for traffic extraction is "completeness". Developers of IPSs need an intact traffic trace that causes a possible FP or FN for correct analysis. This "suspicious" traffic trace must contain not only packets that trigger the log of a possible FN or FP but also the entire session that the packets belong to. The extraction needs information from logs of IPSs, such as source IP address, source port number, destination IP address, destination port number, transport layer protocol and time. According to the information, the ASE can find "anchor" packets that trigger the log of a possible FN or FP and associate other packets with "anchor" packets if they belong to the same TCP or UDP connection (i.e. the same five-tuple). However, extracting a connection completely is still insufficient, since a session could consist of multiple connections and IPSs do not log all the connections in the session. For example, most IPSs log only the first connection of a DDOS attack session to avoid information explosion in the log system. Association of related connections is required for ASE.

II. Background

2.1 Traffic capturing and replaying to an IPS

In this work, we replay real traffic to IPSs and identify attacks by the logs in the IPSs. Such an approach of capturing and replaying has been used for performance evaluation of IPSs [1] [2]. Extracting an attack session [3] involving multiple connections from a huge number of traffic traces is non-trivial. This work designs a method to extract an attack session based on the similarity of packets. Tcpdump (www.tcpdump.org) captures real traffic in a PCAP file, and Tcpreplay (tcpreplay.sourceforge.net) replays the traffic trace packet by packet to IPSs at the specified speed or in the order of the timestamps that indicate the capturing time of the packets.

IPSs differ in many aspects such as signature set, accuracy and logging system. The signature set affects the number of detected attacks. The accuracy affects the correctness of deciding whether an event is an attack or not. The logging system affects the name of an attack. First two properties are reasons that we want to do the integration of efforts from different IPS vendors. However, the last property is what we need to resolve to correctly compare logs of IPSs.

2.2 Attack identifiers and attack types

Although IPSs may name an attack differently, most of them have a system of common identifier for attacks and that is CVE number. CVE (Common Vulnerabilities and Exposures, cve.mitre.org) is a list of security vulnerabilities and exposures that provides common names for publicly known vulnerabilities for easily sharing data across separate vulnerability capabilities (tools, repositories, and services) with this "common enumeration".

We divide attacks into three types according to the number of attackers (i.e. source IP address) and the number of connections per attacker, as presented in Table 1. An attack of the first type (i.e. 1-1) involves one attacker and a single connection per attacker. For example, the MySQL Authentication Bypass Exploit [4] allows a user to login a MySQL database without authentication. An attack of the second type (i.e. 1-N) involves one attacker and more than one connection per attacker. For example, the Blaster worm [5] establishes three connections for each victim. An attack of the third type (i.e. N-1) involves multiple attackers and a single connection per attacker. A Distributed Denial of Service (DDoS) attack belongs [6] [7] to this type. This classification will be used in the following ASE algorithm.

Table 1: The three attack types

Attack type	Number of attackers	Number of connections per attacker	Example
1-1	1	1	MySQL Authentication Bypass Exploit
1-N	1	N	Blaster worm
N-1	N	1	DDoS

III. The System of Attack Session Extraction (ASE)

This ASE algorithm consists of three-pass scanning of the traffic trace: finding the "anchor" packets that trigger possible FPs and FNs by log comparison, associating other packets with the "anchor" packets if they belong to the same TCP or UDP connection, and associating other connections with the "anchor" connection if they belong to the same session. The ASE algorithm is described as follows:

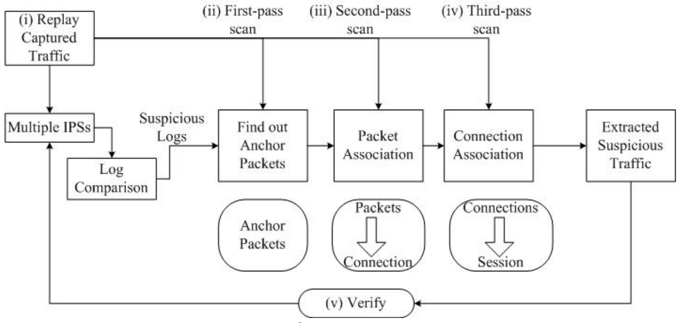

Figure 1: The system of Attack Session Extraction (ASE)

(i) *Replay traffic traces to Multiple IPSs*

The ASE replays traffic traces to multiple IPSs and then compare logs among them to find out suspicious logs, including logs which only logged or not logged at certain IPS. Two tables are implemented. One is Alarm Log Table (ALT), generated by IPSs and used to record logs from IPSs. The other is Replay Log Table (RLT), generated by the machine running Tcpreplay and used to record the time when Tcpreplay sends each packet. In this step, the ASE is to integrate existing signature databases in commercial IPSs and the open-source IPS Snort [8] (*www.snort.org*).

(ii) *First-pass: finding out anchor packets.*

This step finds the anchor packets, the critical packets that make IPS generate logs, in the traffic traces according to the information of ALT and RLT. Five-tuple in the ALT is enough to identify anchor packets. Unfortunately some IPSs 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.

(iii) *Second-pass (Packet Association): finding out attack packets within the same connection as anchor packets.*

This step discovers the attack connection where the anchor packets belong to. If a packet shares common five-tuple with anchor packets, it belongs to the same connection.

(iv) *Third-pass (Connection Association): finding out attack connections within the same session as anchor connections.*

Because a session may involve multiple connections, it is insufficient to depend on only five-tuple and timestamp, e.g., the attack session with multiple attackers in a DDoS attack. The session extraction algorithm is based on this observation that such an attack often consists of only the TCP ACK or SYN message, as well as a number of packets mostly having the same data payload. After finding the anchor packets of an attack, the algorithm checks each following packet to see if its source IP address or destination IP address is identical to the target IP address of the anchor packet. If not, the packet will be considered to belong to other attacks; otherwise, the algorithm will continue to compare the payload of each packet that may belong to this attack for similarity. The algorithm duplicates a copy of this packet in a buffer and increases the packet count by one if the similarity is high.

The similarity is defined according to the *longest common subsequence* (LCS) [9] of two packet payloads. Given a sequence $X = (x_1, x_2, ..., x_m)$, a sequence $Z = (i_1, i_2, ..., i_k)$ is a *subsequence* of *X* if there is a strictly increasing sequence $(i_1, i_2, ..., i_k)$ of indices of *X*. Given two sequences *X* and *Y*, *Z* is a *common subsequence* of *X* and *Y* if *Z* is a *subsequence* of both *X* and *Y*. The *longest common subsequence* is the longest subsequence of the all *common subsequences*. Consider the payloads of two packets as two sequences of bytes, S_1 and S_2. LCS (S_1, S_2) denotes the longest sequence of bytes that are subsequences of S_1 and S_2. The similarity of two packets is defined by

$$\text{Similarity}(S_1, S_2) = \frac{2 \times |\text{LCS}(S_1, S_2)|}{|S_1| + |S_2|} * 100\% \cdot \quad (1)$$

The similarity threshold is 80% in the proposed algorithm because the packets collected from the DDoS attacks are often minimum Ethernet packets of 64 bytes. Excluded the 14-byte MAC header, 20-byte IP header, 20-byte TCP header and 4-byte checksum, the remaining payload is only 6 bytes long. We observe the packet payloads of the DDoS or DoS attacks we collected are often the same, and the difference is only one byte if the payloads are different. The similarity in this case is 83.33%. This work, therefore, sets the similarity threshold to 80%.

After identifying similar packets, the session extraction algorithm watches the source IP address and the destination IP address at the same time. This step keeps only the packets between the attacker and the target. The other packets are dropped. This step intends to distinguish the attacks that have possibly one attacker from those that are possibly DDoS attacks.

Table 2. The notations used in the session extraction algorithm.

Notations	Descriptions
S_{ip}	Source IP address
S_{port}	Source Port number
D_{ip}	Destination IP address
D_{port}	Destination port number
Tcp/Udp	The TCP/UDP flag
Payload	The content of the packet

P_i	A TCP or UDP packet in the IP network.
$Tuple(P_i)$	The five-tuple of packet P_i
A	The anchor packet of one attack
PDA (Possible DoS Attacks)	The set of packets that could be DoS attacks
PNDA (Possible Not DoS attacks)	The set of packets that could be not DoS attacks

The algorithm keeps watching the subsequent packets. The algorithm returns the packet count in the DDoS attack buffer. The attack might be a DDoS attack if the count is larger than 200, adjusted by the evaluation in Section 4, or an attack of type 1-1 otherwise. Table 2 lists the notations in the session extraction algorithm. The five-tuple of the packet P_i is defined by

$$Tuple(P_i) = (S_{ip}(P_i), S_{port}(P_i), D_{ip}(P_i), D_{port}(P_i), Tcp/Udp(P_i)) \quad (2)$$

An anchor packet *A* is a packet that triggers an alarm. The problem, consequently, turns into looking for the packets having highly similar payload to *A* or those having the same source IP address or destination IP address as *A*. The session extraction algorithm is described as follows.

```
PDA = $\phi$; // a set of packets, possible the DoS attack
PNDA = $\phi$; //a set of packets, possible  not the DoS attack
DDos.packet_count = 0;
For all $i$   {//
    if ( $Tuple(P_i).S_{ip} = Tuple(A).D_{ip} \,||\, Tuple(P_i).D_{ip} = Tuple(A).D_{ip}$ ){
        if (Similarity($P_i.Payload, A.Payload$) $\geq$ 80%){
        PDA = PDA $\cup P_i$;
        DDos.packet_count ++;
        }// End of if
        if ( ($Tuple(P_i).S_{ip} = Tuple(A).S_{ip}$ && $Tuple(P_i).D_{ip} \neq Tuple(A).D_{ip}$)) ||
            $Tuple(P_i).S_{ip} = Tuple(A).D_{ip}$ && $Tuple(P_i).D_{ip} \neq Tuple(A).S_{ip}$)){
        PNDA = PNDA $\cup P_i$;
        }// End of if
    } // End of if
}//end of for
if ( DDos.packet_count $\geq$ 200){
    return PDA;
}else{
    return PDNA;
}//end of if
```

Figure 2: The pseudo code of the session extraction algorithm

(v) *Replaying the extracted attack session to IPSs to verify whether the same logs are generated. If it is true, the extraction is valid.*

Finally, we replay the extracted attack sessions to IPSs to verify the correctness of the extraction. The extracted session should exactly cause the same alarms as the whole traffic is replayed. If an IPS product does not have the same alarm, the extraction is invalid.

An example of the session extraction system

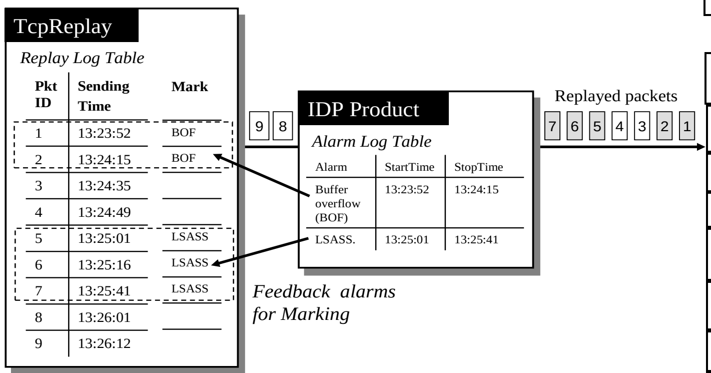

Figure 3: Replay traffic to the IPS products and mark attack packets in the ALT.

Fig. 3 presents an example of the ALT that records the attack alarms during the replay of attacking traffic. We replay traffic from 13:23:52 to 13:26:12 and the IPS generates two alarms (buffer overflow and LSASS) into the ALT. The anchor packet can be found in the logs of the IPS. Figure 3 shows the anchor packet of LSASS can be the 5-th, 6-th or 7-th packet according to the time frame 13:25:01 ~ 13:25:41 (First-pass). We arbitrarily choose the 6-th. If the five-tuple of these three packets is the same, which means they are in the same connection, they will be extracted (Second-pass). If the 5-th and 7-th packets have different five-tuple, the session extraction algorithm compares the IP address, checks the payload, and finally extracts packets (Third-pass). The extracted traffic trace is replayed to IPSs. It should cause the same alarms as the entire traffic was replayed.

IV. Evaluation and Discussion

We use Tcpdump on several Linux PCs to capture real traffic from a sub-network of more than a thousand users in our campus. The evaluations of the ASE algorithm include the number of attacks that could be extracted, as well as the "variation" and "completeness and purity".

4.1 The result of session extraction

Table 3 presents attack events detected by four IPSs (Snort 2.4.5 on FreeBSD 6.1, ISS Proventia G-1200, Fortinet FortiGate 400A and TippingPoint UnityOne-1200) from the captured traffic in the order of the event counts. Out of total 187 extracted attacks, this table presents only 15 frequent attacks that have more than 500 event counts. These 15 attacks cover 87% of all event counts. Table 3a contains attacks with high severity while Table 3b shows attacks with medium severity. The level of severity (i.e. high, medium, low) is defined by IPSs and is used to notify network administrators how dangerous the attack could be.

Table 3a.The attacks detected by the IPSs (high severity)

Attack Name	Event Counts	CVE ID
SSH_ChallengeResponse_Bo	13489	CVE-2002-0640
SNMP_InvalidTag_OID	11660	CA-2002-03
SNMP_Bad_Variable_Type	6409	CA-2002-03
SSH_Brute_Force	4347	------
DNS_Address_Length	3292	CVE-2001-1329
HTML_Hostname_Overflow	2255	CVE-2005-0554
HTTP_Cisco_Catalyst_Exec	1226	CVE-2000-0945
FTP_List_dotdot	810	------

Table 3b.The attacks detected by the IPSs (medium severity)

Attack Name	Event Counts	CVE ID
HTTP_Connect_Proxy_Bypass_SMTP	29316	-----
YahooMSG_UserID_Overflow	13916	CVE-2003-1135
SNMP_Community	9983	CA-2002-03
HTML_NullChar_Evasion	4001	-----
TCP_Data_Changed	1212	CA-1995-01
Synflood	519	CVE-1999-0116
ICMP_Redirect	500	CVE-1999-0265

4.2 Variation, Completeness and Purity

The extracted attacks might not be complete or pure. ASE might miss the packets of low similarity (i.e., not complete). On the other hand, it might extract extra packets since the system drops only the packets that must not be part of an attack (i.e., not pure). We design a method to assess how complete and pure the ASE is. The concept is to extract an attack session from different traffic traces, totally n traces, that contain the attack, and compare the extracted sessions. We compare the packet size of extracted sessions packet by packet. COMP(Attack_i) denotes the number of extracted sessions which have identical sequence of packet size for Attack *i*. The variation of Attack *i* is defined by

$$Variation(Attack_i) = (1 - COMP(Attack_i)/n) * 100\%$$
.(3)

If there exists more than one set of extracted sessions which have identical sequence of packet size for Attack *i*, the biggest COMP(Attack_i) will be used to calculate Variation (Attack_i).

We prepare 100 traffic traces (i.e. attack traces), and each has an attack session. We also prepare 10 other traffic traces (i.e. background traces) as background traffic. Each attack traffic trace is mixed with 10 background traffic traces (i.e. mixed traces), separately, meaning that the experiment will extract an attack session 10 times from different background traffic.

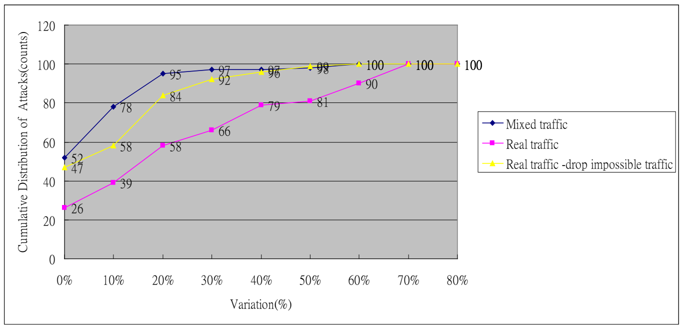

Figure 4. The variation of extracted attacks

Figure 4 shows in the line of "Mixed traffic" that 52% of the attack traces have no variation, 95% have less than 20% variation, and 97% less than 30% variation. For example, the variation of 30% means that only three extracted attack traces are different from other seven extracted attack traces. We observe that the variation results from other normal connections between the attacker and the victim, such as normal HTTP and FTP which are not attacks. If we replace the IP address pair of the attack connections such that it is always the unique IP address pair in the mixed traces, variations will all become 0%.

We abbreviate the "Completeness and Purity" as the "CP". It is calculated by first calculating the similarity of each packet in the extracted attack traces and the corresponding packet in original trace using Equation 1. Then, we average similarities calculated above to get the "CP" of two traffic traces.

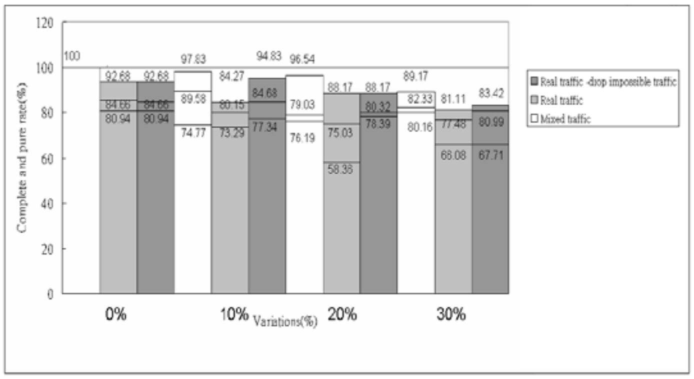

Figure 5: The CP for different variations

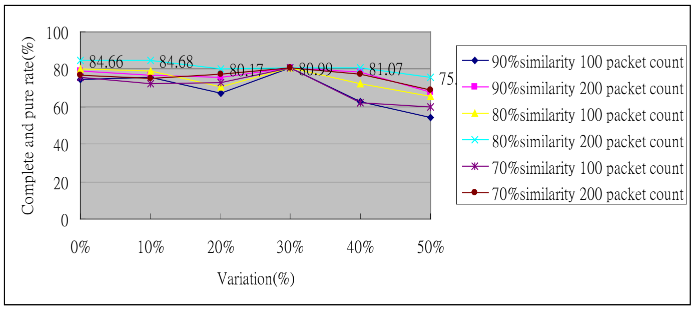

Figure 6: The effect on the CP for two different thresholds (i.e. similarity and packet count) in Figure 2.

Figure 5 shows the maximum, minimum and average CP for different variations. For attack traces with variation of 10%, the maximum CP is 97.83%, the minimum CP is 74.77%, and the average CP is 89.58%. The CP is affected by the thresholds of similarity and packet counts, which are defined in Section 3. In our experiment, the similarity of 80% and 200 packet counts have the best CP. Figure 6 shows the effect on the CP for different thresholds of similarity and packet count. The CP of 30% variation are the same because there is no attack likes the type of DDoS in this case. If we evaluated all the 187 attacks with more than 50,000 event counts from real traffic (The result is the line labeled "Real traffic" in Figure 5), which means each of 187 attacks is extracted over 50,000 times and every extraction is from different traffic traces, the variation will increase and the CP will decrease. 26% of the attacks have no variation and only 58% have a variation less than 30%. For the attacks of 10% variation, the maximum CP is 84.27%, the minimum CP is 73.29%, and the average CP is 80.15%. From our observation, if we drop packets that cannot be identified as an attack (The result is the line labeled "Real traffic –drop impossible traffic" in Figure 5), the attack traces of 0% variation will increase to 47% and the attack traces of 10% variation will increase to 72%. The CP will be also increased. In other words, we need a "white list" in ASE to describe what kinds of packets should be ignored without being classified as PDA or PNDA.

V. Conclusions

This work proposes a system to completely extract suspicious sessions from traffic traces. These suspicious sessions may cause FPs or FNs to an IPS and the extracted traffic traces can be used for analysis by signature developers to improve the accuracy of the IPS. The extraction process scans a traffic trace three times. Similarity between two packets is defined to extract a DDOS attack completely. We define "variation" and "completeness and purity" to evaluate the accuracy of ASE. 95% of the extracted attacks have low variation. Also, the average CP is up to 80%. This method could be extended to other detection system such as Anti-Virus, P2P/IM management, and Network Forensics.

References

[1] H. G. Kayacık and A. N. Zincir-Heywood, "Using Intrusion Detection Systems with a Firewall: Evaluation on DARPA 99 Dataset", Project in Dalhousie University, [Online]. Available:http://projects.cs.dal.ca/projectx/files/NIMS06-2003.pdf.

[2] DARPA 99 Intrusion Detection Data Set Attack Documentation. [Online]. Available: http://www.ll.mit.edu/IST/ideval/docs/1999/attackDB.html.

[3] V. Corey, C. Peterman, S. Shearin, M. S. Greenberg, J. V. Bokkelen, "Network Forensics Analysis," IEEE Internet Computing, vol.06, no.6, pp. 60-66, 2002.

[4] W. D. Yu, D. Aravind, P. Supthaweesuk, "Software Vulnerability Analysis for Web Services Software Systems," iscc, pp. 740-748, 11th IEEE Symposium on Computers and Communications (ISCC'06), 2006.

[5] M. Bailey, E. Cooke, F. Jahanian, D. Watson, Jose Nazario, "The Blaster Worm: Then and Now," IEEE Security and Privacy, vol. 03, no. 4, pp. 26-31, 2005.

[6] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram, D. Zamboni, "Analysis of a Denial of Service Attack on TCP," sp, p. 0208, 1997 IEEE Symposium on Security and Privacy, 1997.

[7] V. Paxson, "An analysis of using reflectors for distributed denial-of-service attacks" ACM SIGCOMM Computer Communication Review, 2001.

[8] M. Roesch, "Network Security: Snort - Lightweight Intrusion Detection for Networks", Proceedings of the 13th USENIX conference on System administration, November. 1999.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, "Introduction to Algorithms", p.p. 314-320, 1990.

[10] T. Ye, D. Veitch, G. Iannaccone and S. Bhattacharyya, "Divide and Conquer: PC-Based Packet Trace Replay at OC-48 Speeds", IEEE TRIDENTCOM, 2005.

[11] W. C. Feng, A. Goel, A. Bezzaz, W. C. Feng, and J. Walpole. "TCPivo: A high-performance packet replay engine". ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research (MoMeTools), Aug. 2003.R. W. Lucky, "Automatic equalization for digital communication," *Bell Syst. Tech. J.*, vol. 44, no. 4, pp. 547–588, Apr. 1965.