Extracting Ambiguous Sessions from Real Traffic with Intrusion Prevention Systems

I-Wei Chen[1], Po-Ching Lin[2], Tsung-Huan Cheng[3], Chi-Chung Luo[3], Ying-Dar Lin[3], Yuan-Cheng Lai[4] and Frank C. Lin[5]

(Corresponding author: I-Wei Chen)

Network Benchmarking Lab, National Chiao Tung University[1]
R611, MISRC Building, No.1001, DaXue Rd, Hsinchu City 300, Taiwan
(Email: iwchen@nbl.org.tw)
Department of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, Taiwan[2]
Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan[3]
Department of Information and Management, National Taiwan University of Science and Technology, Taipei, Taiwan[4]
San Jose State University, San Jose, CA, USA[5]

(Received June 9, 2011; revised and accepted Aug. 18, 2011)

Abstract

False Positives (FP) and False Negatives (FN) are common in every Intrusion Prevention System (IPS). None of the systems could judge better than others *all* the time. This work proposes a system of Ambiguous Session Extraction (ASE) to create a pool of *ambiguous* traffic traces. Traffic traces or sessions are called "ambiguous", meaning they cause potential FNs (abbreviated as P-FNs) and potential FPs (abbreviated as P-FPs) to IPSes. IPS developers can use these ambiguous traffic traces to improve the accuracy of their products. The key objective here is to design the ASE system to extract the traces as complete and pure as possible, which gives IPS developers resources for further analysis. First, the ASE captures real traffic and replays captured traffic traces to multiple IPSes. By comparing the logs of IPSes, we might find that some sessions are *logged* or *not logged* only at a certain IPS. The former is P-FPs, while the latter is P-FNs to that IPS. The ASE then starts to extract ambiguous traffic from replayed traffic traces. IPS developers can further analyse the extracted traffic traces and confirm that some are FNs or FPs. To completely and purely extract an ambiguous session, the ASE uses an association mechanism based on *anchor packets*, *five tuples and time*, and *similarity* for the first packet, first connection, and whole session, respectively. It calculates the degree of similarity among packets to extract an ambiguous session containing multiple connections. We define *variation* and *completeness/purity* as the indexes to evaluate the ASE. The experiments demonstrate that 95% of extracted sessions have low variation, and the average completeness/purity is around 80%. We also present two case studies, one is a P-FN and the other is a P-FP, found by the ASE and confirmed by the IPS developers to be an FN and an FP, respectively.

Keywords: False positive/negative, intrusion prevention, packet trace, session extraction, similarity

1 Introduction

It is a challenge for Intrusion Prevention Systems (IPS) to defend a network [1, 2], since malicious traffic on the Internet keeps growing and changing. Many IPS vendors assign a large number of dedicated engineers to analyse malicious traffic and write appropriate signatures into their products database, but False Positives (FPs) and False Negatives (FNs) still happen. If an IPS generates an attack log, but other IPSes do not when processing the same traffic, there is a high probability that the traffic causes an FP to that IPS. On the contrary, if an IPS does not generate an attack log, but other IPSes do when processing the same traffic, there is a high probability that the traffic causes an FN to that IPS. If we can provide the traffic trace that causes a potential FN (abbreviated as P-FN) or a potential FP (abbreviated as P-FP) to IPS developers for further analysis, the accuracy of the signature database will be improved. We call a traffic trace or a session within the trace *ambiguous*, if it causes a P-FN or a P-FP.

We design a system of ambiguous session extraction (ASE) to create a pool of ambiguous traffic traces. 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 the PCAP format (`www.tcpdump.org`) and contain packets from various applications such as FTP, Web, E-mail, P2P

and Instant Messenger. Second, it replays traffic traces to IPSes of different vendors and trigger them to generate logs. From the logs, we can discover that some attack logs are *generated* or *not generated* only at a certain IPS. The former are P-FPs, while the latter are P-FNs to that IPS. A voting process follows for the IPSes to determine whether the traffic is ambiguous or not. Third, the ASE system extracts ambiguous traffic and provides extracted traffic traces to developers for enhancing the accuracy of their IPSes.

Our main concern for traffic extraction is *completeness*. IPS developers generally need an intact session for precise analysis. The ambiguous traffic trace must contain not only packets that can trigger an FN or FP, but also the entire session that the packets belong to. The extraction takes some *clues* to ambiguous traffic from IPS logs, such as source IP address, source port number, destination IP address, destination port number, transport layer protocol and time [3]. According to the clues, the ASE can find *anchor packets* that trigger the FN or FP, i.e., the critical/essential packets that can make IPS generate logs. Without any one of the anchor packets, IPSes would not generate the log. The ASE then associates the other packets with the anchor packets if they belong to the same TCP or UDP connection[1]. However, completely extracting a connection is still insufficient, since a session could consist of multiple connections and IPSes do not log all the connections in that session. For example, most IPSes log only the first connection of a DDOS attack session to avoid information explosion in the log system. Associating related connections is required for the ASE.

The remainder of this paper is organized as follows: in Section 2, we discuss background knowledge and related work. Section 3 makes a detailed description of the design and implementation of the ASE. In Section 4, we evaluate the ASE with two indexes and discuss the results. Two case studies of FN and FP are illustrated in Section 5. We conclude with Section 6.

2 Background

2.1 Extraction of Ambiguous Sessions

In this work, we capture and replay real traffic to IPSes, and look for some *clues* in the IPS logs to help identify ambiguous sessions. The steps of capture and replay have been adopted for performance evaluation of IPSes [4, 5]. Open source tools such as Tcpdump (`www.tcpdump.org`) and Tcpreplay (`tcpreplay.sourceforge.net`) can be used to capture and replay real traffic, respectively. The former captures real traffic into files in the PCAP format as traffic traces, while the latter replays the traffic traces to the IPSes. However, the volume of traffic traces may be huge, and it is non-trivial to extract an ambiguous session involving multiple connections from the traces [6, 7, 8].

[1]We mean the same five tuples, even though UDP is connectionless.

Table 1: Three attack types

Attack type	Number of attackers	Number of connections per attacker	Examples
1-1	1	1	MySQL authentication bypass exploit
1-N	1	N	Blaster worm
N-1	N	1	DDoS

This work designs a method to extract an ambiguous session based on anchor packets, five tuples and time, and similarity.

We classify the attacks in an ambiguous session 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 [9] 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 [10] 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 [11, 12, 13, 14, 15] belongs to this type. This classification will be referred to in the following ASE design.

2.2 Related Work of the Voting Scheme

Voting schemes are prevalent in the area of malware detection. BotHunter [16] is a system for detecting malware infection through IDS-driven dialog correlation. It consists of a correlation engine driven by three malware-focused network packet sensors. One of the three sensors, SCADE, detects outbound traffic based on a voting scheme (AND, OR or MAJORITY) of three parallel anomaly detection models. Provos et. al. conducted an analysis of web-based malware [17]. They present the state of malware on the Web, and emphasize the importance of this rising threat. To classify the different types of malware, they use a majority voting scheme based on the characterization provided by popular anti-virus software. Employing multiple anti-virus engines allows them to determine whether some of the malware binaries are actually new, false positive or older exploits. VirusTotal [18], developed by Hispasec Sistemas, is a service that analyses suspicious files and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware detected by anti-virus engines. It uses several command-line versions of antivirus engines, updated regularly with official signatures published by their respective develop-

ers.

This paper uses multiple IPSes to vote for the ambiguous sessions in the first step of the ASE. The results and the IPS logs provide some *clues* to help identify the ambiguous session. After the voting scheme is the three-pass scanning/association to extract the ambiguous session as complete and pure as possible. Completeness and purity are required for IPS developers to precisely and quickly analysing the P-FN and the P-FP.

3 The System of Ambiguous Session Extraction (ASE)

Figure 1 illustrates the steps in the ASE system. After replaying the traffic traces to multiple IPSes and the voting scheme, the system only knows which packets trigger the logs, but it is still necessary to extract the entire session for helping the IPS developers identify an FP or FN precisely. The session extraction involves three-pass scanning through the traffic traces: (1) finding the anchor packets that trigger the P-FP or P-FN from the five-tuple information in the IPS logs, (2) associating the other packets with the anchor packets if they belong to the same TCP or UDP connection, and (3) associating the other connections with the connection which the anchor packets belong to, if they belong to the same session. The sessions associated with the anchor packets are therefore extracted in the scanning. They are replayed to the IPSes again to verify the correctness of extraction.

3.1 The Steps in Session Extraction

The details of each step in Figure 1 are elaborated as follows.

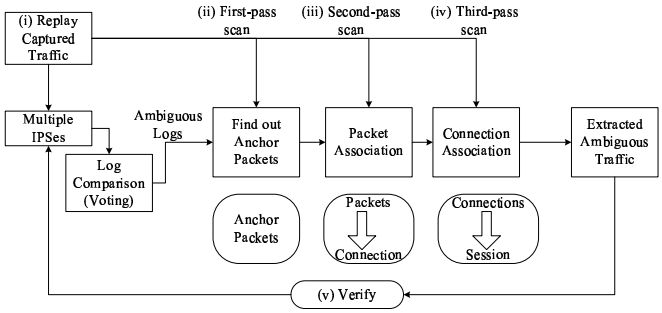

Figure 1: System architecture of ambiguous session extraction (ASE)

(i) Replaying traffic traces and voting In this step, the ASE system leverages the knowledge in signature databases on commercial IPSes and the open-source Snort IPS [19] to find ambiguous traffic traces. The system replays captured traces to multiple IPSes, and then compares the logs among them in the voting scheme to find out ambiguous logs, i.e., those only generated (P-FP) or not generated (P-FN) at a certain IPS. These logs will be correlated with the packets to help the session extraction.

(ii) Finding out anchor packets (the first pass) This step finds out anchor packets in the ambiguous traces, i.e., the critical/essential packets that can trigger IPS logs. For this purpose, we implement an Alarm Log Table (ALT) to record alarm logs from IPSes, and a Replay Log Table (RLT) to record the time of each packet sent from Tcpreplay. The two tables can keep the useful information for correlating the logs and packets. Matching the five tuples should be sufficient to identify the anchor packets, but unfortunately, some IPSes do not log the complete five tuples of packets, e.g., only the IP addresses of both ends. The ASE system therefore needs to match the time information in both tables. The time of a log entry in the ALT is correlated with that in the RLT to determine a time frame. The system then just needs to search for the anchor packets within the same time frame in the traffic trace.

(iii) Packet Association (the second pass) This step looks for all the other packets that have the same five tuples as the anchor packets. Those packets are then associated with the anchor packets to be in the same connection.

(iv) Connection Association (the third pass) Because a session may involve multiple connections, e.g., an ambiguous session with multiple attacking sources, it is insufficient to only depend on the five tuples and timestamp of packets. We design a session association algorithm based on the observation that such an ambiguous session often consists of only the TCP ACK or SYN segments, as well as a number of packets mostly having the same data payload. After finding the anchor packets of an ambiguous session, the algorithm checks each subsequent packet to see if its source IP address or destination IP address is identical to the victim's IP address of the anchor packet. If not, the packet will be considered not belonging to this attack; otherwise, the algorithm will continue to compare the payload of each packet that may belong to this attack for similarity. If the similarity is high for a packet, the algorithm will duplicate its copy in the DDoS attack buffer for later judgment.

The similarity is defined according to the longest common subsequence (LCS) [20] of two packet payloads. Given a sequence $X = (x_1, x_2, \ldots, x_m)$, a sequence $Z = (z_1, z_2, \ldots, z_m)$ is a subsequence of X if there is a strictly increasing sequence $(i_1, i_2, \ldots, i_k)$ of indices of X, such that $x_{i_j} = z_j$ for all $j = 1, 2, \ldots, k$. 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

Table 2: The notations used in the session association 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 tuples of packet P_i
A	The anchor packet of one attack
PDA(Probably DoS Attacks)	The set of packets that could be DoS attacks (N-1 type)
$PNDA$(Probably Not DoS Attacks)	The set of packets that could not be probably DoS attacks (1-1 or 1-N type)

longest one 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 between two packet payloads is defined by

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

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. Excluding 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 association algorithm watches the source IP address and the destination IP address at the same time. This step stores only the packets between the attacker and the victim, and drops the other packets.

This step also distinguishes the attacks that have probably one attacker from those that are probably DDoS attacks. The algorithm keeps watching the subsequent packets, and returns the packet count in the DDoS attack buffer. The ambiguous session might be a DDoS attack if the count is larger than 200; otherwise the attack type is 1-1 or 1-N.

An anchor packet A triggers an log on an IPS. The problem, consequently, turns into looking for the packets having the payload highly similar to A's and the packets having the same source IP address or destination IP address as A's. Figure 2 lists the session association algorithm, and Table 2 summarizes the notations in the algorithm. The five tuples of the packet P_i is defined by

$$Tuple(P_i) = (S_{ip}(P_i), S_{port}(P_i), D_{ip}(P_i), D_{port}(P_i) TCP/UDP(P_i)). \quad (2)$$

Algorithm: The session association algorithm

```
// The set of packets that could be DoS
   attacks
PDA ⟵ φ;
// The set of packets that could probably
   not DoS attacks
PNDA ⟵ φ;
DDoS.packetcount ⟵ 0;
for P_i in the traffic traces do
    if (Tuple(P_i).S_ip = Tuple(A).D_ip or
    Tuple(P_i).D_ip = Tuple(A).D_ip) then
        if Similarity(P_i.Payload, A.Payload) ≥ 80%
        then
            PDA ⟵ PDA ∪ P_i;
            DDoS.packetcount ⟵
            DDoS.packetcount + 1;
        end
        if (Tuple(P_i).S_ip = Tuple(A).S_ip and
        Tuple(P_i).D_ip = Tuple(A).D_ip) or
        (Tuple(P_i).S_ip = Tuple(A).D_ip and
        Tuple(P_i).D_ip = Tuple(A).S_ip) then
            PNDA ⟵ PNDA ∪ P_i;
        end
    end
end
if DDoS.packetcount ≥ 200 then
    // N-1 type
    return PDA;
else
    // 1-1 or 1-N type
    return PNDA;
end
```

Figure 2: The pseudo code of the session extraction algorithm

(v) Extracting ambiguous sessions and verifying Finally, we replay the extracted ambiguous sessions to the IPSes to verify the correctness of the extraction. The extracted sessions should trigger exactly the same alarm logs as the whole traffic does. If an IPS product does not generate the same logs, the extraction is invalid.

3.2 An Example of Session Extraction

Figure 3 presents an example of replaying real traffic to an IPS. We replay the 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 of LSASS could be the 5-th, 6-th or 7-th packet, since the correlation between the ALT logs and the RLT logs determines the time

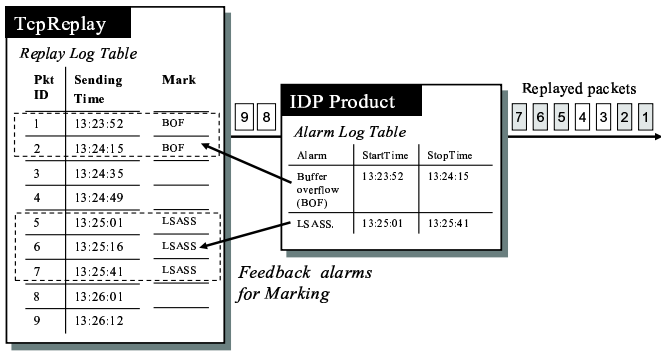

Figure 3: Replaying traffic to an IPS products and correlating the ALT logs with the RLT logs

frame starts from 13:25:01 to 13:25:41 (the first pass). We arbitrarily choose the 6-th. If the five tuples of the three packets are the same, they are in the same connection and will be extracted (the second pass); otherwise, the session association algorithm compares the IP addresses, checks the payload, and finally extracts related packets (the third pass). The extracted traffic trace is again replayed to the IPS to verify the correctness of extraction. If the LSASS alarm is not reproduced, we may choose an incorrect anchor packet, and would try the others to go through the ASE.

4 Evaluation and Discussion

We use Tcpdump on Linux PCs to capture real traffic from a sub-network of more than a thousand users in our campus, and replay the traffic to four IPSes: Snort on FreeBSD, ISS Proventia, Fortinet FortiGate and TippingPoint UnityOne. The evaluations of the ASE system include *variation* and *completeness/purity*.

4.1 The Result of Session Extraction

Table 3 presents the logged events (due to replaying the traffic) on the four IPSes in the order of event counts. Out of total 187 attacks detected and logged, the table presents only the top 15 frequent attacks that have more than 500 event counts, meaning the attacks occur at least 500 times. The 15 attacks cover 87% of all event counts. Table 3(a) contains attacks with high severity, while Table 3(b) shows attacks with medium severity. The level of severity (i.e., high, medium and low) is defined by the IPSes and is used to notify network administrators how dangerous an attack could be.

4.2 Variation, Completeness/Purity

The extracted traces might not be complete or pure. The traces may be incomplete, since the ASE may miss some packets of low similarity. The traces may also be not pure. The ASE may extract extra packets, since the system drops only the packets whose IP addresses, either source or destination, do not match the destination IP address of the anchor packets. We design a method to assess the completeness and purity of the traces extracted by the ASE. The idea is extracting a target session multiple times from n different traffic traces, all containing the target session, and then comparing the sequence of packet sizes in the extracted traces with that in the target session. Let TS_i denotes the target session i and $COMP(TS_i)$ be the number of extracted traces having an identical sequence of packet size (in the order of packet time) to that of TS_i. The variation of TS_i is defined by

$$Variation(TS_i) = (1 - COMP(TS_i)/n) \times 100\%. \quad (3)$$

Besides the packet sizes, we also compare the packet content. The assessment of *Completeness/Purity* (abbreviated as *CP*) of an extracted trace is conducted by first calculating the similarity of every packet between that trace and the target session using Equation 1, and then averaging the similarity values of all the packets to derive the CP value of an extracted trace.

There are two scenarios in the experiment. In the first scenario, we prepare 100 attack traces, each is an attack session (also the target session) derived from the traffic generated by real attack programs. We also randomly capture 10 real traffic traces as the background traffic. Each attack trace is mixed with 10 background traces (i.e., mixed traces) separately, meaning that the ASE will later extract an attack session 10 times from the mixed traces. In the second scenario, we just capture real traffic traces and replay them to an IPS. The ASE extracts the attack sessions with the procedure in Figure 1. We then find and launch the attack programs corresponding to the extracted attack traces, and capture the traffic as the attack sessions (also the target sessions). The difference between the two scenarios is the way to mix traffic. The mixing in scenario 1 is artificial, while it is natural in scenario 2. No matter in which scenario, however, our aim is the same — to compare an extracted trace with the corresponding attack trace.

In Figures 4 and 5, the label of *Mixed traffic* represents scenario 1, the label of *Real traffic* represents scenario 2, and the label of *Real traffic-drop impossible traffic* means a variant of scenario 2 in which the packets that are not part of the attack session are dropped in advance. In scenario 1, Figure 4 shows that 52% of the attack traces have no variation, 95% have variation less than 20%, and 97% have variation less than 30%. In general, variation of n% means that, for an attack trace, only $10 \times n$% extracted traces are different from the others. We observe that the variation results from other connections between the attacker and the victim that do not belong to the attack session. Consequently, if we modify the IP address pair of the attack session such that the pair is always unique in the mixed traces, variations will all become 0%, since that means no other connections exist concurrently with the attack session between the attacker and the victim.

Figure 5 shows the maximum, minimum and average CP values for different variations. For attack traces with

Table 3: The attacks detected by the IPSes

(a) attacks of 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	——

(b) attacks of 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

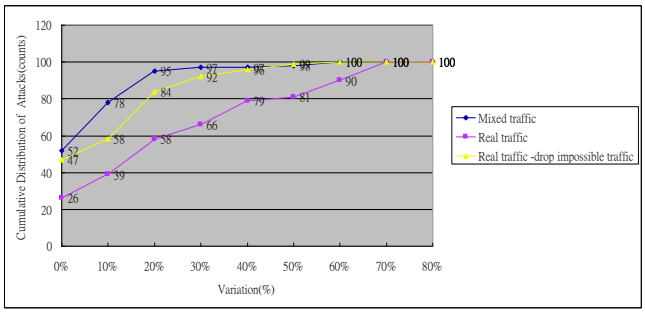

Figure 4: The variations of the extracted attacks

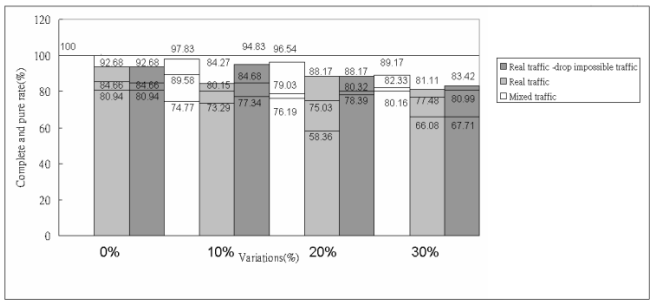

Figure 5: The CP values for different variations

variation of 10%, the maximum CP value is 97.83%, the minimum is 74.77%, and the average is 89.58%. The CP is affected by the thresholds of similarity and packet count defined in the session extraction algorithm. In our experiment, the best CP is achieved when we set the threshold of similarity to 80% and the threshold of packet counts to 200. Figure 6 shows the effect of different thresholds on the CP values. At variation of 30%, however, the CP values of different thresholds are the same because there is no attack of the N-1 type like the DDoS in that situation. As illustrated in Figure 2, the two thresholds of similarity and packet count determine only the packets in PDA. If there is no attack of the N-1 type in the traces (i.e., an empty PDA), they do not affect the results of session extraction, and thus neither the CP values.

In scenario 2, we evaluate the 15 attacks on Table 3 based on the experiment in Section 4.1. It means each of 15 attacks is extracted over 500 times and every extraction is from different mixed traces. 26% of the attack traces have no variation and only 58% have variation less than 30%. For the attack traces with variation of 10%, the maximum CP value is 84.27%, the minimum is 73.29%, and the average is 80.15%. If we filter out packets that do not belong to the attack session (the result shown in the line labeled *Real traffic-drop impossible traffic*), the percentage of attack traces with no variation will increase to 47%, and that of the attack traces with variation of 10% will increase to 72%. The CP also increases. Since the improvement is due to filtering out packets not in the attack session, we need a *white list* in the ASE to describe what kinds of packets should be filtered out in the beginning without being classified into PDA or PNDA.

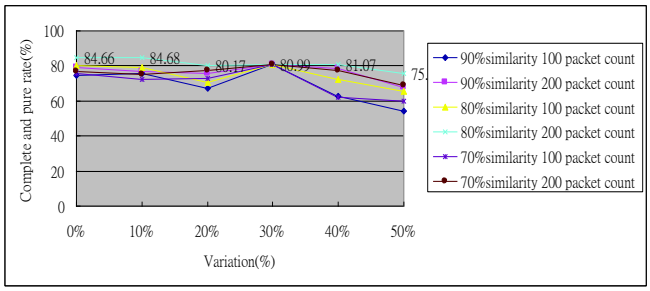

Figure 6: The effect of different thresholds of similarity and packet count on the CP in the session extraction algorithm

5 Case Studies on False Negatives and False Positives

The outputs of the ASE are ambiguous sessions that cause P-FNs and P-FPs to IPSes. IPS developers can analyse the ambiguous sessions to improve the accuracy of their products. Below are two examples, one is a P-FN and the other is a P-FP, found by ASE and confirmed by IPS developers to be an FN and an FP, respectively.

5.1 False Negatives (FN)

The traffic traces in the first example does not trigger a log on IPS-1, but on all the other IPSes, meaning the event is a P-FN to IPS-1. The triggered log is something like "SQL SA brute force login attempt TDS". We use the ASE to extract the session as complete and pure as possible, and then provide the session to the developers of IPS-1 for further analysis. They find the signature in IPS-1 is too specific to detect the event. The signature is composed based on the assumption that some patterns, such as "|0000|"[2] and "s|00|a|00|", will appear frequently. To solve the problem, the developers should assume less frequent appearance of those patterns in its signature.

5.2 False Positives (FP)

The traces in the second example trigger a log only on IPS-2, but not on the others, meaning the event is a P-FP to IPS-2. The log is "DDOS mstream". We again use the ASE to extract and provide the ambiguous session to the developers of IPS-2 for further analysis. The session is actually from BitTorrent (a P2P application for file sharing) communication between two peers, not a DDOS. The developers find the signature is too general, including the destination port 12754 as well as the content ">". Coincidentally, the ambiguous session matches the two patterns and causes the false positive. The developers should figure out a more specific pattern, rather than a pattern of a single character like ">".

[2] $|b_1 \dots b_n|$ denotes a sequence of bytes in hexadecimal.

6 Conclusion

This work proposes an ASE system to extract ambiguous sessions from real traffic as complete and pure as possible. The ambiguous sessions cause P-FPs or P-FNs to an IPS and can be used for further analysis by IPS developers to improve the accuracy of an IPS. The system associates related packets with an attack by scanning traffic traces three times — to identify the anchor packets, the packets in the same connection and the connections in the same session. For connection association in a session, similarity between two packets is defined to extract an ambiguous session of the N-1 type. We define *variation* and *completeness/purity* to evaluate the ASE. 95% of the attack traces have low variation. Also, the average completeness/purity is around 80%. This method could be extended to other detection systems such as anti-virus and P2P management.

References

[1] T. Bhaskar, N. Kamath B and S. D. Moitra, "A hybrid model for network security systems: Integrating intrusion detection system with survivability," *International Journal of Network Security*, vol.7, no.2, pp. 249–260, Sept. 2008.

[2] J. Zeng and D. Guo, "Agent-based intrusion detection for network-based application," *International Journal of Network Security*, vol.8, no.3, pp. 201–210, May 2009.

[3] P. Kabiri and A. A. Ghorbani, "A rule-based temporal alert correlation system," *International Journal of Network Security*, vol.5, no.1, pp. 66–72, July 2007.

[4] H. G. Kayacik 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`.

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

[6] V. Corey, C. Peterman, S. Shearin, M. S. Greenberg and J. V. Bokkelen, "Network forensics analysis," *IEEE Internet Computing*, vol. 6, no. 6, pp. 60–66, Nov–Dec 2002.

[7] L. Wang, A. Ghorbani and Y. Li, "Automatic multi-step attack pattern discovering," *International Journal of Network Security*, vol.10, no.2, pp. 142–152, Mar. 2010.

[8] A. Almulhem and I. Traore, "Detecting connection-chains: A data mining approach," *International Journal of Network Security*, vol.10, no.1, pp. 62–74, Jan. 2010.

[9] W. D. Yu, D. Aravind and P. Supthaweesuk, "Software vulnerability analysis for Web services software

systems," *Proc. 11th IEEE Symp. on Computers and Communications (ISCC)*, pp. 740–748, June 2006.
[10] M. Bailey, E. Cooke, F. Jahanian, D. Watson and J. Nazario, "The blaster worm: Then and now," *IEEE Security and Privacy*, vol. 3, no. 4, pp. 26–31, July 2005.
[11] C. L. Schuba, I. V. Krsul, M. G. Kuhn, E. H. Spafford, A. Sundaram and D. Zamboni, "Analysis of a denial of service attack on TCP," *IEEE Symp. on Security and Privacy*, May 1997.
[12] V. Paxson, "An analysis of using reflectors for distributed denial-of-service attacks," *ACM SIGCOMM Computer Communication Review*, vol.31, no.3, pp. 38-47, July 2001.
[13] J. Udhayan and T. Hamsapriya, "Statistical segregation method to minimize the false detections during DDoS attack," *International Journal of Network Security*, vol.13, no.3, pp. 152–160, Nov. 2011.
[14] M. C. Lee, Y. J. He and Z. Chen, "Towards improving an algebraic marking scheme for tracing DDoS attacks," *International Journal of Network Security*, vol.9, no.3, pp. 204–213, Nov. 2009.
[15] Y. Chen, S. Das, P. Dhar, A. E. Saddik and A. Nayak, "Detecting and preventing IP-spoofed distributed DoS attacks," International Journal of Network Security, vol.7, no.1, pp. 69–80, July 2008.
[16] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee, "Bothunter: Detecting malware infection through IDS-driven dialog correlation," *Proc. of the 16th USENIX Security Symposium*, Aug. 2006.
[17] N. Provos D. McNamee, P. Mavrommatis, K. Wang and N. Modadugu, "The ghost in the browser analysis of Web-based malware," *Proc. of the First Workshop on Hot Topics in Understanding Botnets*, Apr. 2007.
[18] Virustotal, `http://www.virustotal.com`.
[19] M. Roesch, "Network security: Snort - lightweight intrusion detection for networks", *Proc. of the 13th USENIX conference on System Administration*, Nov. 1999.
[20] T. H. Cormen, C. E. Leiserson and R. L. Rivest, Introduction to Algorithms, The MIT Press, pp. 314–320, 1990.

I-Wei Chen received his B.S. degree and M.S. degree in Computer and Information Science from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 2001 and 2003, respectively. He joined Network Benchmarking Lab (NBL) at NCTU since 2003. His research interests include network security, network benchmarking technologies, and real-world scenario testing methodologies. He is currently the executive director of NBL (2007).

Po-Ching Lin received the B.S. degree in computer and information education from National Taiwan Normal University, Taipei, Taiwan, in 1995, and the M.S. and Ph.D. degrees in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 2001 and 2008, respectively. He joined the faculty of the Department of Computer and Information Science, National Chung Cheng University (CCU), Chiayi, Taiwan, in August 2009. He is currently an Assistant Professor. His research interests include network security, network traffic analysis, and performance evaluation of network systems. He is also a member of the IEEE.

Ying-Dar Lin is Professor of Computer Science at National Chiao Tung University (NCTU) in Taiwan. He received his Ph.D. in Computer Science from UCLA in 1993. He served as the CEO of Telecom Technology Center during 2010–2011 and a visiting scholar at Cisco Systems in San Jose during 2007–2008. Since 2002, he has been the founder and director of Network Benchmarking Lab (NBL, www.nbl.org.tw), which reviews network products with real traffic. He also cofounded L7 Networks Inc. in 2002, which was later acquired by D-Link Corp. He recently, in May 2011, founded Embedded Benchmarking Lab (www.ebl.org.tw) to extend into the review of handheld devices. His research interests include design, analysis, implementation, and benchmarking of network protocols and algorithms, quality of services, network security, deep packet inspection, P2P networking, and embedded hardware/software co-design. His work on "multi-hop cellular" has been cited over 500 times. He is currently on the editorial boards of IEEE Transactions on Computers, IEEE Network, IEEE Communications Magazine Network Testing Series, IEEE Communications Surveys and Tutorials, IEEE Communications Letters, Computer Communications, and Computer Networks. He recently published a textbook "Computer Networks: An Open Source Approach" (www.mhhe.com/lin), with Ren-Hung Hwang and Fred Baker (McGraw-Hill, 2011). It is the first text that interleaves open source implementation examples with protocol design descriptions to bridge the gap between design and implementation.

Yuan-Cheng Lai received the Ph.D. degree in computer science from National Chiao Tung University, Hsinchu, Taiwan, in 1997. In August 2001, he joined the faculty of the Department of Information Management at National Taiwan University of Science and Technology, Taipei, Taiwan, where he has been a professor since February 2008. His research interests include wireless networks, network performance evaluation, network security, and content networking.

Frank C. Lin retired from his engineering career in 2008 after 30+ years of industry experiences with Unisys, Octel/Lucent, and Cisco Systems. He is now teaching at San Jose State University in California on networking, security, algorithms, object programming and architecture. Frank received his BSEE, MSEE, Ph.D. in CS from National Taiwan University, Utah State University, and University of Utah, respectively.