國立交通大學

資訊科學與工程研究所

碩 士 論 文

從真實網路流量中萃取與重製攻擊流量

Attack Session Extraction and Replay from Real Traffic

研 究 生：羅榮鐘

指導教授：林盈達 教授

中 華 民 國 九 十 五 年 六 月

從真實網路流量中萃取與重製攻擊流量

Attack Session Extraction and Replay from Real Traffic

研 究 生：羅榮鐘　　Student：Chi-Chung Luo

指導教授：林盈達　　Advisor：Dr. Ying-Dar Lin

國 立 交 通 大 學

資 訊 科 學 與 工 程 研 究 所

碩 士 論 文

A Thesis

Submitted to Institute of Computer Science and Engineering

College of Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science

June 2006

Hsinchu, Taiwan, Republic of China

中華民國九十五年六月

從真實網路流量中萃取與重製攻擊流量

學生：羅榮鐘　　指導教授：林盈達

國立交通大學資訊科學與工程研究所

摘要

一個系統是否安全通常會使用弱點偵測的工具來進行測試，有一類弱點偵測的工具是使用外部的網路流量去詢問一個系統的某一服務是否開啟來找尋系統是否有漏洞。然而這樣的測試並不能精準的抓出系統的缺點，因其並非真知道系統漏洞可否破壞，因此我們想利用真實的網路攻擊來測試系統的弱點。事實上，真實的網路攻擊並不容易收集，因此本研究設計了一個攻擊流量的萃取系統。這個萃取攻擊流量的系統主要有三個重點，第一，本系統利用播放錄製的流量到入侵探測和防護系統來取得警示紀錄。第二，根據警示紀錄從真實流量中找出令入侵探測和防護系統發出警示的最重要封包，藉由前兩個重點，有相同網路特徵值的封包集合則稱為一個網路攻擊連線。然而，一個網路攻擊可能會有多個來源，或者一個來源卻有多條連線，因此，本研究經過分析觀察後設計了第三個重點。第三個重點是藉由內容相似

度比對來找出多個來源的攻擊。透過萃取攻擊流量系統所取得的 83%攻擊是不容易受外在影響而變化的，在低變化量攻擊中有 71%的攻擊是可被驗証為完整且無雜質的。透過此系統的協助，本研究除了可以萃取出完整無雜質的攻擊外，同時也透過這些被萃取的攻擊來比較與弱點偵測的工具流量的差異性。

關鍵字： 網路安全、弱點偵測、網路攻擊、流量萃取、內容相似度比對

Attack Session Extraction and Replay from Real Traffic

Student: Chi-Chung Luo Advisor: Dr. Ying-Dar Lin

Department of Computer and Information Science

National Chiao Tung University

Abstract

The tools of vulnerability assessment (VA) can be used to check the system security. One kind of the VA tools is using the network traffic to request the system service and waiting the response of the service. By the response of the service, the VA tool can find out the vulnerability of the system. However, this tool can not actually find out the vulnerability of the system because the tool can not check the vulnerability of the system is destruct or not. Therefore, we need to use the real attacks to test the system vulnerability. In fact, the real attacks are difficult to collect. Therefore, this work proposes an attack session extraction system. The attack session extraction system has the three key points. First, the attack session extraction system is replaying the recorded traffic to IDP products to get alarm logs. Second, the attack session extraction system found out the critical packet that the IDP products make alarm by the alarm logs. The first and second key points of the attack session extraction system can find out the packets that have the same network characteristic and merge to a set as a connection of network attacks. However, a network attack maybe have many attackers or single attacker but multi connections. Therefore, this work analyzed the attacks and designed the third key point. The third key point is using the packet payload similarity to find out the attacks that have the multi attackers. The 83% of the extracted attacks have low

variation. The 71% of the low variation attacks can be verified as completeness and purity. By the help of attack session extraction, this work can extract the complete attacks and also use the extracted attacks to compare the different between the VA tools and real attacks.

Index Terms: Network Security, Vulnerability Assessment, Network Attacks, Session Extraction, Payload Similarity

Acknowledgement

Many people have helped me with this thesis. I deeply appreciate my thesis advisor, Dr Ying Dar Lin, for his intensive advice and instruction. I would like to thank all the classmates in High Speed Networks Laboratory for their invaluable assistance and suggestions.

Finally, I thank my Father and Mother for their endless love and support.

Contents

Abstract (in Chinese) **i**

Abstract (in English) **ii**

Acknowledgements **v**

Contents **vi**

List of Figures **viii**

List of Tables **ix**

1. INTRODUCTION……...…….1

2. BACKGROUND……...……4

2.1 Vulnerability Assessment tools……..…….……...…4

2.2 Nessus: port-based security scanner and emulated attacks…….4

2.3 IDP products ……5

2.4 Traffic record and replay tools ……6

2.5 Attack identifiers and attack types ……6

3. SESSION EXTRACTION SYSTEM……..….8

3.1 OVERVIEW……...8

3.2 Extract attack sessions from recorded traffic……..…....8

3.3 Analysis of the difference between Nessus traffic and real attacks……..13

3.4 The example of the session extraction system……..….....15

4. EVALUATION AND DISCUSSION**…….....17**

4.1 The result of session extraction ……17

4.2 Variation, completeness and purity……..……...…20

4.3 Analysis of the difference between Nessus traffic and the real attacks……..25

5. CONCLUTION AND FUTURE WORK ……...27

REFERENCES…….....28

List of Figures

Figure 1. Coverage of Nessus attacks and real traffic attacks. ……..……..…2

Figure 2. The flowchart of the session extraction algorithm. 14

Figure 3. Replay traffic to the IDP products and mark attack packets in the alarm log table ……...……... ……...……16

Figure 4. The frequency of the 15 attacks . ……..19

Figure 5. The file sizes of the 15 attacks. …….……....19

Figure 6. The variation of extracted attacks . .……...21

Figure 7. The different sizes of attacks less than 3 ……..…….22

Figure 8. The different sizes of attacks equal to 0 ……..…….....22

Figure 9. The different sizes of attacks large than 3. ……..……..23

Figure 10. The real attacks v.s Nessus traffic. …….……....25

List of Tables

Table 1. Three types of attack definitions. …….. ……..……..7

Table 2. The definition of the components in session extraction algorithm …9

Table 3. The 15 attacks detected by the IDP product, Snort……18

Chapter 1

Introduction

Vulnerability Assessment (VA), an important technology to detect deficiencies of products and systems, can check whether a system is robust to malicious attacks. A VA tool should know attack definitions to examine the vulnerability of a system. Among VA tools, Nessus (*http://www.nessus.org*) is popular because of its rich database of attack definitions. Having the attack definitions, Nessus can point out possible security breaches by playing partial attacks to the systems under test. The partial attacks emulated by Nessus are helpful to detect the deficiencies of products and systems without harming them. In other words, Nessus can only indicate possible security breaches but does not know whether they are harmful. A complete episode of real attacks is mandatory to test whether a system can be harmed. However, the real attacks are very difficult to collect. Therefore, this work proposes a way of the attacks collection by extracting the recorded real attacks from the real traffic.

Extracting a complete episode of attacks from an overwhelmingly large amount of recorded traffic is non-trivial. For this goal, real traffic is recorded and then replayed to the intrusion detection and prevention (IDP) products to extract the complete episode of attacks. Such an approach to record traffic and send it to IDP products has been used for evaluation of the performance of the IDP [1], [2]. The IDP products indicate the signs of detected attacks on its logging system, but do not record the attack traffic. This work designs a method to extract attacks according to the logs of IDP products. This method records real traffic and then extracts attacks by associating packets via logs with connections and then with sessions. This session extraction system therefore can extract the desired complete episode of attacks.

Moreover, the association between the extracted real attacks through IDP and the

emulated attacks from Nessus deserves further study. By understanding the difference of the real attacks and Nessus traffic, the newly extracted attacks could be possibly inserted into Nessus as new plug-ins in the database to enhance the capability of Nessus.

This work defines two sets of attack definitions. One is the attacks extracted from the IDP products, and the other is the collected attacks in Nessus. Both attack definitions are to be studied before the association between them is understood. First, whether the attacks covered in Nessus can be detected by the IDP products are unknown. Therefore, we conduct Nessus to send attacks to the IDP products to watch the detection by the IDP products. Second, the recorded traffic can be replayed to the IDP products to see how many real attacks can be detected. We can then build up a set of real attacks that can be detected by the IDP products. Therefore, the coverage of both types of attacks could be derived. Because the Nessus attacks are only partial attacks, the proposed extraction method compares both Nessus attacks and complete real attacks detected by the IDP products to find out their difference. The new attacks that Nessus does not have can be inserted into it.

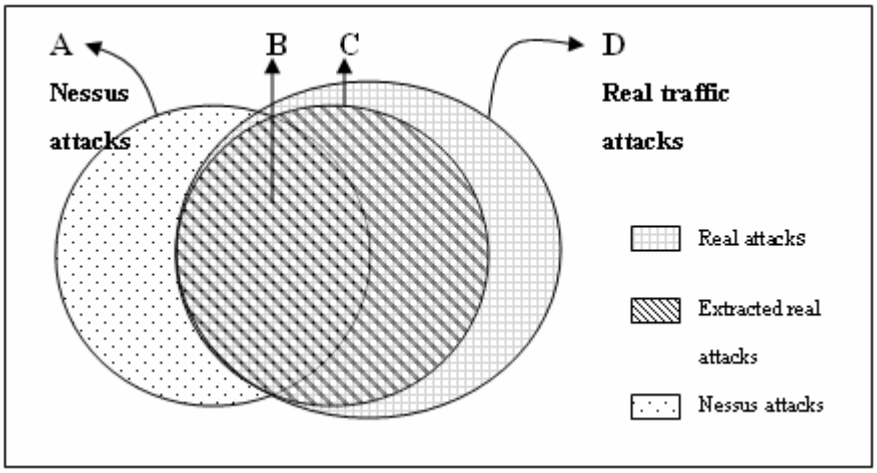

Figure 1. Coverage of Nessus attacks and real traffic attacks

In Fig. 1, *D* is the set of all types of attacks in real traffic. *C* is the set of types of attacks identified by existing IDP products. *A* is all the types of attacks in Nessus.

Therefore, *C*-*A* (or *C*-*B*) is the set of types of attacks newly found from real traffic, where *B* is the intersection of *A* and *C*, to be added, if extractable, into the Nessus database. Although some attacks might not be detected by existing IDP products, i.e. *D*-*C*, such a method at least guarantees that the Nessus database includes all attacks known by existing IDP products.

The rest of this paper is organized as follows. Section 2 presents the background and related works. Section 3 describes the session extraction system. Section 4 presents the experimental results. Finally, Section 5 summarizes the key results.

Chapter 2

Background

This section justifies the need to extract real attacks from the network traffic. Some tools and studies are introduced herein.

2.1 Vulnerability Assessment tools

VA tools can detect the system vulnerability in a number of ways. For example, VA tools can be installed on a host to detect the vulnerability by checking the system configuration and status. A well-known open-source security tool, Tiger (*http://www.nongnu.org/tiger*), is such a kind. Besides, VA tools can also detect the vulnerability externally. Nessus, a port-based security scanner, uses techniques such as port scanning and attack emulation to detect vulnerability. Nessus is useful when installing a VA tool on the system to be detected is not feasible. For example, most network security products do not allow installing third-party tools on their systems.

2.2 Nessus: port-based security scanner and emulated attacks

Nessus is a port-based security scanner, which means that Nessus scans the TCP/UDP ports of the target system before detecting the vulnerability. Nessus records the opened ports and sends emulated attacks through them to detect the vulnerability of the target system. The emulated attacks are not destructive, i.e., they will not harm the target system. Nessus collects already known attacks into its plug-in database without the harming part. Nessus also adds some detection content (to test the system vulnerability) into their traffic and then wait the response from the target system. In other words, Nessus uses three steps to detect the vulnerability. The first step is scanning the target ports and recording the opened ports. The second step is sending the emulated attacks. Finally, Nessus receives the vulnerability report. The emulated attacks in Nessus are different from real attacks because the Nessus traffic includes the

detection part and the report part but excludes the harm part.

Although Nessus can find out the vulnerability, it does not mean the target system could get hurt by real attacks. Some systems may patch the vulnerability or use devices like firewalls to protect the system security. The vulnerability may still exist under the protection of firewalls or patches. Nessus can only indicate possible security breaches but does not know whether a real attack will harm or not. For this reason, real attacks are needed to verify the system security accurately.

2.3 IDP products

Extracting a complete episode of attacks is non-trivial. Two problems exist with the attack extraction. The first one is the attacks definition. This work does not intend to define an attack because it is a well-known skill and it also takes adequate human works. In this work, we replay the real traffic to existing IDP products individually and identify attacks by the log of these IDP products. The second problem is how to extract multi-connection attacks. An attack session may involve multiple connections. It is challenging to extract an attack session from a huge traffic trace. Besides, we also need to define an attack session and extract it. Therefore, this work designs a method to extract attacks through the logs of IDP products. For this design, real traffic is recorded by Tcpdump and then replayed by Tcpreplay to the IDP to extract the complete episode of attacks.

However, each IDP product has its own signature set, false positive rate, log style and alarm level. The different signature sets effect the number of the detected attacks. The different false positive rates effect the correction of the detected attacks. The different log styles and alarm levels affect the types of the detected attacks because the same attack maybe has different names. This work needs to extract attacks as possible as we can, therefore, more than one IDP is needed to get the complete attack and avoided the incorrect detect. Beside, we design a method to associate the different log

styles and alarm levels of each IDP products. The section 2.5 will describe the detail content.

Therefore, the set of C in Figure 1 (*C* is the set of types of attacks identified by existing IDP products) can be consider as $C = \bigcup_{i=1}^{n} C_i$, where $\{C_i \mid i = 1, \ldots\ldots, N\}$, assuming we have N IDP products. The one goal of this work is extracting the union of the detected attacks by IDP products.

2.4 Traffic record and replay tools

Tcpdump*(http://www.tcpdump.org)*and Tcpreplay *(http://tcpreplay.sourceforge.net)* are used in this work. Both Tcpdump and Tcpreplay are famous open-source tools. Tcpdump can record network traffic ordered by timestamp over important routes or backbone networks. The recorded traffic is stored in a file of PACP format. Tcpreplay can play the recorded traffic packet by packet according to their timestamps. Tcpreplay can also send packets at specified speed to the target system. They both can help this work to extract attacks by recording and replaying packets to the IDP products.

2.5 Attack identifiers and attack types

The association between the extracted real attacks and the Nessus plug-in(s) is one of our research goals. Newly found attacks can be possibly added into the Nessus database. The CVE (*http://cve.mitre.or/*) attack identifier here is used to make association between the extracted real attacks and the Nessus plug-in(s). CVE (Common Vulnerabilities and Exposures) is a list of security vulnerabilities and exposures that provides common names for publicly known problems. Its goal is to make it easier to share data across separate vulnerability capabilities (tools, repositories, and services) with this "common enumeration". The organization of CVE studies new attacks and defines the attack identifiers and descriptions. The Nessus traffic has the CVE attack identifiers and the IDP products often have, too. The CVE attack identifier

can be used to check whether an attack belongs to the Nessus plug-in database or not.

This work collects 83 attacks as the samples for the extraction system. These attacks can be divided into three types according to the number of attackers and the number of connections per attack, as presented in Table 1. We assume only one target is in each attack. An attack of the first type involves one attacker and a single connection. An example is the MySQL Authentication Bypass Exploit. This attack can login in a MySQL database without the password. An attack of the second type involves one attacker and more than one connection. An example is the Blaster worm, which establishes three connections when it tries to attack a target. An attack of the third type involves multiple attackers and a single connection from each attacker. A DDoS attack belongs to this type. This observation is helpful to build an extraction system [3], [4].

Table 1. Three types of attack definitions

Number of attackers	Number of connections per attack	Example
1	1	MySQL Authentication Bypass Exploit
1	N	Blaster worm
N	1	DDoS

Chapter 3

The Session Extraction System

The proposed session extraction system includes logs and traffic analysis. This system has two parts. The first part is an algorithm to extract sessions from the attack traffic. The second part is the analysis of the difference between Nessus traffic and the real attacks.

3.1 Overview

There are two goals in this work. First, because the partial attacks emulated by Nessus can only indicate possible security breaches but does not know whether a real attack will harm or not, this work proposes to extract the complete episode of attacks to make sure the system vulnerability. For this goal, the real traffic is recorded and then replayed to the IDP products to extract the complete episode of attacks. Trivially, the logs of the IDP products can help this work to find out the connection of the detected attacks. However, the attack may have the multiple connections. All related connections of the attack can not be all extracted by the logs of the IDP products because the IDP products only alarm and log the most important connection. Therefore, this work proposes an algorithm to extract multi connections from the attack traffic. We named the multi connections extraction algorithm as the session extraction algorithm.

The new attacks that Nessus does not have can be inserted into Nessus is the second goal of this work. Also, this work can add or replace the attacks that Nessus plug-in database already has, depending on whether a complete episode is needed. Therefore, the association between the extracted real attacks from IDP and the emulated attacks from Nessus is needed to understand. This work proposes a method to analyze the difference between Nessus traffic and the real attacks extracted.

3.2 Extract attack sessions from recorded traffic

One goal of this work is to extract a complete episode of attacks from a large amount of traffic. The session extraction algorithm is a three-pass algorithm designed for this goal by associating packets, connections and sessions to extract attack sessions. Before the description of the session extraction algorithm, Table 2 shows the definition of the components in session extraction algorithm. The algorithm consists of five steps as follows. Step (i), (ii), (iii) and (v) are trivial works while the step (iv) is the essence of this work.

Table 2. The definition of the components in session extraction algorithm

Names	Descriptions
S_{ip}	Source IP address
S_{port}	Source Port number
D_{ip}	Distance IP address
D_{port}	Distance Port number
Tcp/*Udp*	The TCP packet or UDP flag
Payload	The content of the packet
P	A TCP or UDP packet in the IP network.
$Tuple(P_i)$	The five-tuple of a packet
A	The anchor packet of the attack
PDA(Possible DoS Attacks)	The data structure that store the packets could be the DoS attacks
PNDA(Possible Not DoS attacks)	The data structure that store the packets could be not the DoS attacks

(i) *Replay real traffic to IDP products by Tcpreplay*.

This algorithm uses the domain knowledge of IDP products, including the well-known Open Source tool, Snort [5]. A IDP product illustrate what attacks have happened with its logs.

(ii) *Find out anchor packets by the first-pass scan.*

This step finds out anchor packets, the critical packets that IDP products alarm when receiving them. There are two tables used herein. One is the alarm log table , which records the alarms of attacks from the replay of attack traffic. The other is the replay log table, which records the time when Tcpreplay replays each packet.(The timestamps from the replay log table are used to mark the attack types by looking for the relation from the alarm log table. The replay log table is then compared with the alarm log table to identify the attack packets.)

Time synchronization could be a problem between the replay system and the IDP products. Even if the time has been synchronized, IDP products may not log the times accurately. Therefore, the five-tuple information is used herein. Many IDP products also log the five-tuple information of an attack (some may record fewer than five tuples). The five-tuple information and the timestamp from the alarm log table and the replay log table can locate the anchor packets in the real traffic.

(iii) *Find out the association among attack packets within the same connection by the second-pass scan.*

This step discovers the anchor connection by looking for the relation of the recorded packets with the anchor packets. If the packets have common five tuples with the anchor packet, they belong to the same connection.

(iv) *Find out the association among attack connections within the same session by the third-pass scan.*

The attack connections can be associated with their session. The association may be difficult since the relation among the connections is obscure. Because the attacks have more than one connection, only five tuples and timestamp are insufficient to find out the other connections. The obscurest relation among the connections is the attack of multiple attackers and a single connection from each attacker because the five tuples of

the packets from these attackers are different. A common attack of this type is the DDoS or DoS attack. These two types of attacks overwhelm a server to deny its capability of providing services. From our observation, such an attack often has only the TCP ACK or SYN message, as well as a number of packets with the same data payload. The session extraction algorithm is designed based on the above observation. The algorithm parses the recorded traffic packet by packet and extracts an attack session by analyzing the attack types.

After anchor packets of an attack have been found, 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 classified to other type of attacks. If the packet belongs to this attack, the algorithm will compare each packet's payload for similarity. The algorithm duplicates a copy in the possible DDoS attack buffer and increases the packet count by one if the similarity is high. The similarity is defined according to the *longest common subsequence* (LCS) of two packet payloads [6]. Formally, given a sequence $X = (x_1, x_2, ..., x_m)$, another sequence $Z = (i_1, i_2, ..., i_k)$ is a *subsequence* of *X* if there exists a strictly increasing sequence $(i_1, i_2, ..., i_k)$ of indices of *X*. given two sequences *X* and *Y*, we say that a sequence *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 subsequence.* Consider the payloads of two packets as two sequences of bytes, S_1 and S_2. The LCS of S_1 and S_2, LCS (S_1, S_2), is the longest sequence of bytes that are subsequences of S_1 and S_2. The similarity is defined by the equation

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

The similarity threshold is 80% in the proposed algorithm because the packets we collected in the DDoS or DoS attacks are often the minimum Ethernet packets of 64

bytes. Excluding 14-byte MAC header, 20-bytes IP header, 20-bytes TCP header and 4-byte checksum, the payload is only 6 bytes long. From our observation, 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%, so the similarity threshold is set to 80%.

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

The algorithm continues to watch the next packet until the end. The algorithm returns the packet count in the possible DDoS attack buffer. The attack might be a DDoS attack if the count is larger than 200, and might be a 1-1 attack otherwise. Figure 2 shows the flowchart of the algorithm.

The algorithm can be written as some formulas and pseudo code as follows. We defined the packet P is the set of five-tuple and payload. The $Tuple(P_i)$ is the five-tuple of the packet i, $i \geq 1$. The anchor packet A is the set of the five-tuple and payload that the IDP products make alarm when they receive it.

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

$$Tuple(P_i) = (S_{ip}(P_i), S_{port}(P_i), D_{ip}(P_i), D_{port}(P_i), Tcp/Udp(P_i)), \quad (3)$$

Therefore, the session extraction problem turns into a problem to find out the set of packets that have the high similarity of payload with anchor packet A or the same source IP address and distance IP address with anchor packet A. Assume the x is the sequence number of anchor packet in the all packets. The session extraction algorithm can be described as follow.

The pseudo code of the session extraction algorithm

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

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

Finally, we replay the extracted attack sessions to IDP products to verify the correctness of the extraction. The extracted session must cause the same alarms as the whole traffic was replayed to the same IDP product. If an IDP product cannot find the attack, the extraction is invalid.

3.3 Analysis of the difference between Nessus traffic and real attacks

The second part is to compare the extracted attack sessions with the attacks from the Nessus plug-in database. The comparison depends on replaying both real traffic and

Nessus to IDP products in the following five steps.

(i) Use Tcpdump to record traffic at the backbone network.

(ii)Replay the real traffic to each IDP product to obtain a set of attacks, $\{C_i \mid i = 1, \ldots\ldots, N\}$*, assuming we have N IDP products.*

(iii) Replay the Nessus Plug-in traffic to each IDP product to generate another set of attacks, $\{A_i \mid i = 1, \ldots\ldots, N\}$.

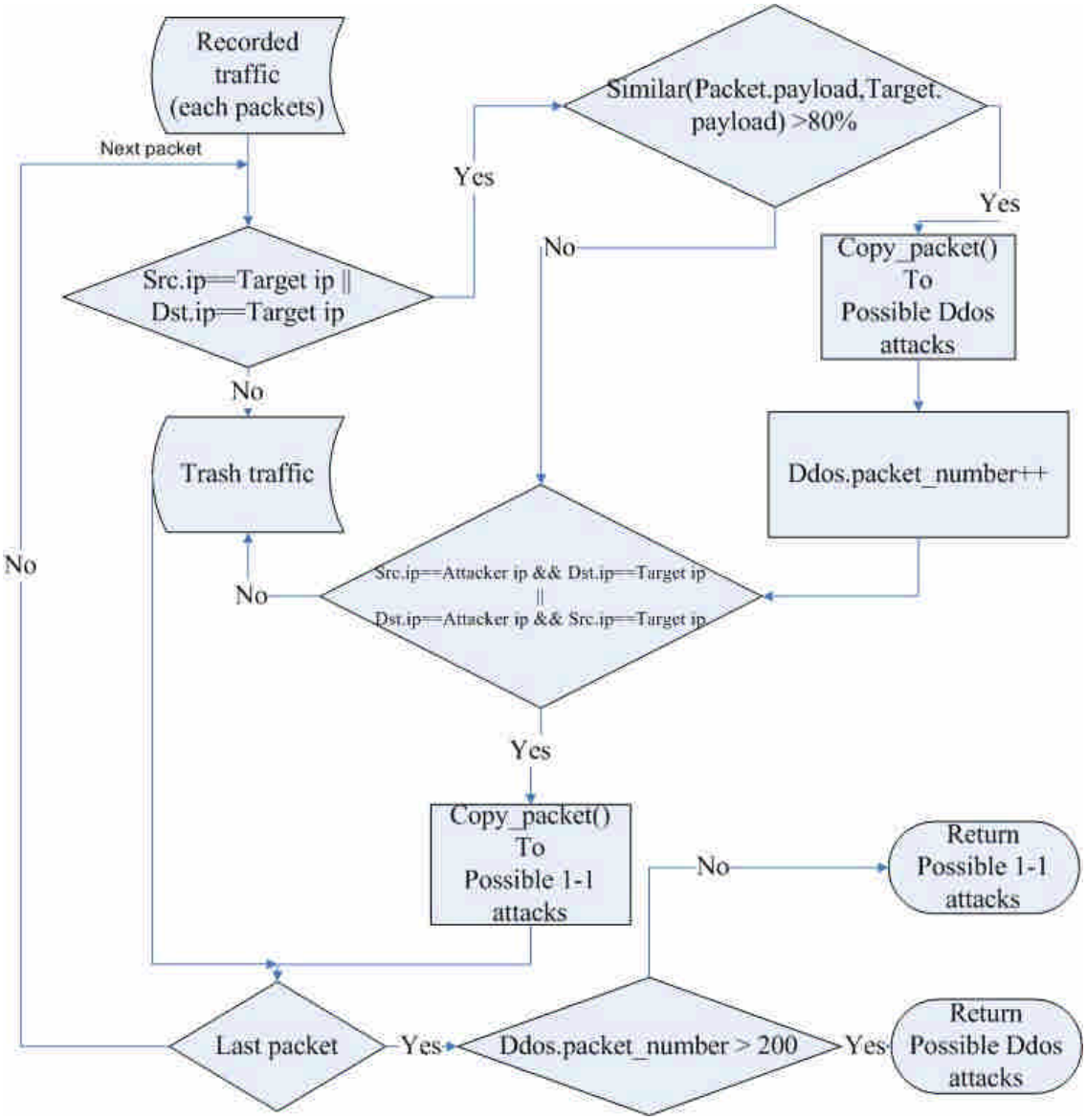

Figure 2. The flowchart of the session extraction algorithm

Some IDP products mark the detected Nessus traffic as "Nessus attack", but the others do not. The mark is confusing because an attack may be marked different in

different IDP products. We solve this problem with the CVE identifier of Nessus traffic. Each Nessus attack has its own CVE identifier, and the alarm log also marks the attacks with the identifier. An attack can be identified from the CVE identifier if the mark is confusing.

(iv) For every attack record in C-A, where $C = \bigcup_{i=1}^{n} C_i$ *and* $A = \bigcup_{i=1}^{n} A_i$ *, i.e. attacks which are not in the Nessus Plug-in database, exercise the extraction algorithm to extract its attack session.*

(v) Those new attack sessions can be added into the Nessus plug-in database.

This work can also add or replace the attacks that already exist in Nessus plug-in database. Nessus is a convenient tool for updating its database. Because a complete episode of real attacks is necessary to detect the real vulnerability of a system, both the complete episode and the partial attack of an attack can be used arbitrarily for the system vulnerability assessment, depending on whether a complete episode is needed.

3.4 The example of the session extraction system

Fig. 3 presents the alarm log table that records the alarms of attacks during the replay of attack traffic. We replay traffic from 13:23:52 to 13:26:12 to an IDP product, and the IDP generates two alarms (buffer overflow and LSASS) into the log table. Second, the anchor packet can be found by logs of the IDP product. As the Figure 3 shows, the anchor packet can be the 5th, 6th or 7th packet and we choose the 6th. If the five-tuple of the packets are the same, the chosen packet has no difference. Third, the five-tuple of anchor packet can be known. Fourth, each packet will be checked the payload similarity and the IP address comparison by session extraction algorithm. If the 5th and 7th packets have different five-tuple, the session extraction algorithm will find them. After the session extraction algorithm, the result is a recorded attack file. In this example, the recorded attack file is the 5th, 6th and 7th packets. Finally, the recorded

attack file can be replay to IDP product. The extracted session must cause the same alarms as the whole traffic was replayed to the same IDP product. If an IDP product cannot find the extracted attack, the extraction is invalid.

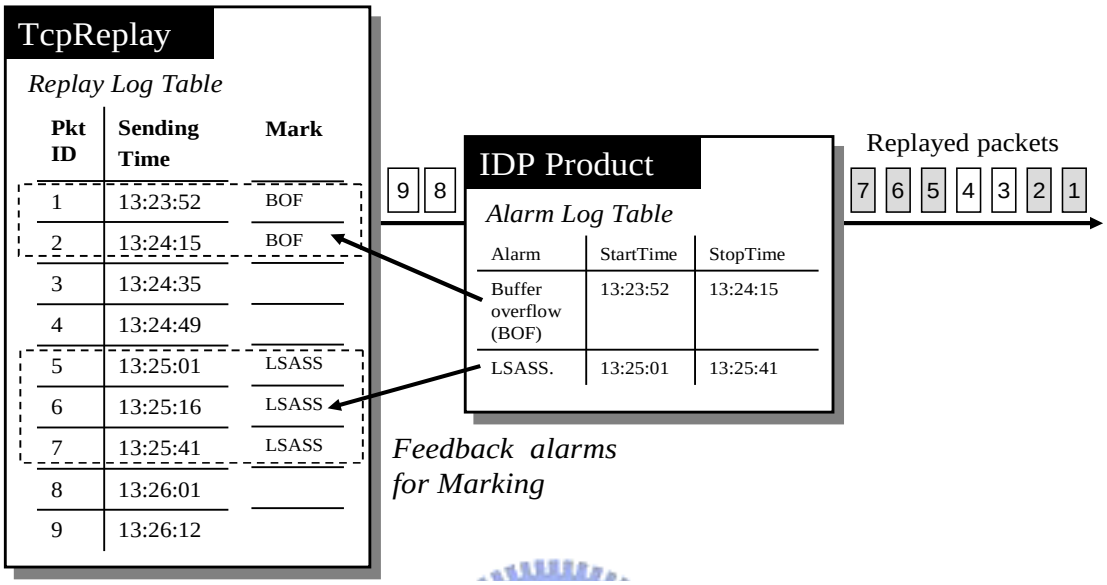

Figure 3. Replay traffic to the IDP products and mark attack packets in the alarm log table

Chapter 4

Evaluation and Discussion

We used Tcpdump on a Linux PC to collect and record all traffic from the other 50 PCs in our lab during the period from 10/23/2005 to 10/29/2005. The 50 PCs belong to the same subnet and have the same gateway to the outside. Therefore, we recorded the traffic by mirroring all traffic through that gateway.

Three evaluations are designed for this work. First, the number of real attacks that could be extracted by the session extraction system is evaluated. The evaluation also looks for the mapping of the real attacks to the Nessus plug-in traffic. We use Tcpreplay to replay the recorded traffic to the IDP product, Snort in this evaluation. The CVE identifiers and Nessus identifiers are also used for the mapping.

Second, the completeness and purity of the extracted attacks are evaluated. This evaluation also answers the variability of the session extraction system. For this evaluation, we replay the recorded traffic and also play the Nessus traffic at the same time.

Finally, the differences between Nessus traffic and the real attacks are evaluated. For this evaluation, we chose a Nessus attack and a real attack with the same CVE identifier, and observe their differences with Ethereal.

4.1 The result of session extraction

Table 3 presents the 15 attacks detected by the IDP product (Snort) from the recorded traffic. They are ordered by the detection time. These attacks all have CVE identifiers when they are detected by Snort.

4.1.1 The occurring percentage of the extracted result

Fig. 4 shows the occurring percentage of each attack in the recorded traffic. We can see the famous worm of Worm.Win32.Slammer has high percentage. The Slammer

worm would be the highest occurring rate of the world during the 2005 to 2006. As Figure 4 shows, the top 6 attacks had 74% of total frequency and the sum of the fewest 3 attacks only 3% of total. The top 1 attack has 16% of total. This result told us the common attacks are the most part of the total attacks.

Table 3. The 15 attacks detected by the IDP product, Snort

Types	Name	Nessus id	CVE id
DoS	Apache.CGI.Byterange.Request.DoS	no	CVE-2005-2728
Worm	Worm.Win32.Slammer	11214	CVE-2002-0649
Windows (1-1)	IE.File.Download.Ext.Spoof	no	CVE-2002-0875
Dos(N-1)	Smtp service logging NULL session	11308	CVE-2002-0054
SMTP(1-1)	smtp_decoder: unknown_cmd	10885	CVE-2002-0055
RPC(1-1)	RPC portmapper	10223	CVE-1999-0632
SMTP(1-1)	Smtp unauthorized user	10703	CVE-2001-0504
General(1-1)	IIS server HTR ISAPI filter mapped	10932	CAN-2002-0071
Dos(N-1)	DoS: FrontPage server Extensions	11311	CAN-2002-0692
Backdoors(1-1)	Sasser Virus Detection	12219	CAN-2003-0533
General (1-1)	Netbios-ssn	10394	CVE-2000-0222
General(1-1)	ICMP timestamp request	10114	CAN-1999-0524
General(1-1)	Remote OS guess	11268	CAN-1999-0454
FTP(1-1)	ftp: Overflow.CWD	no	CAN-2002-0126
DNS(1-1)	FTP Serv-U 2.5e Dos	10488	CVE-200-0837

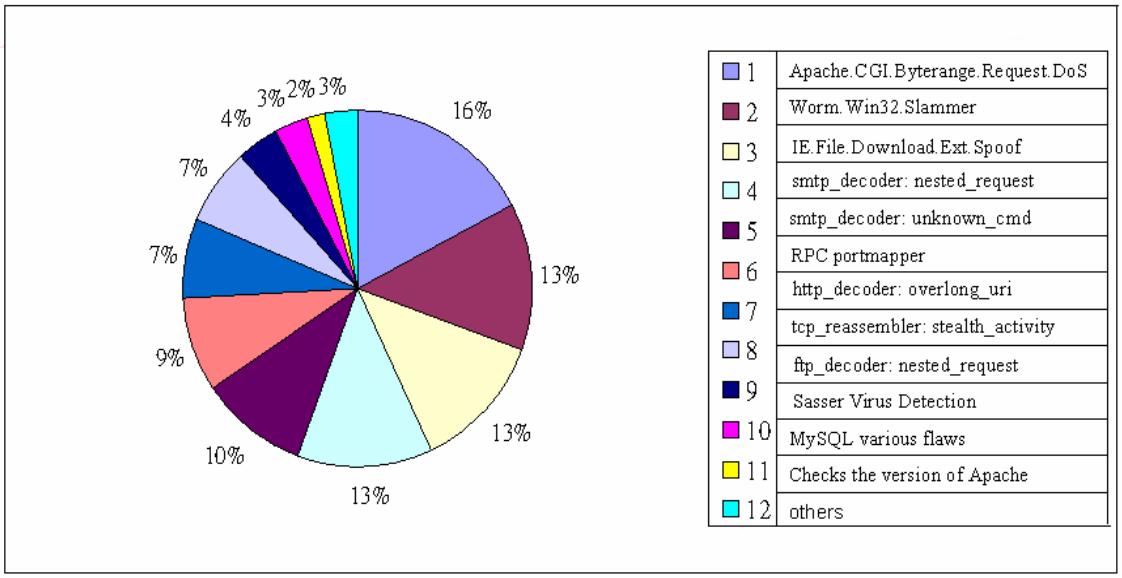

Figure 4 the frequency of the 15 attacks

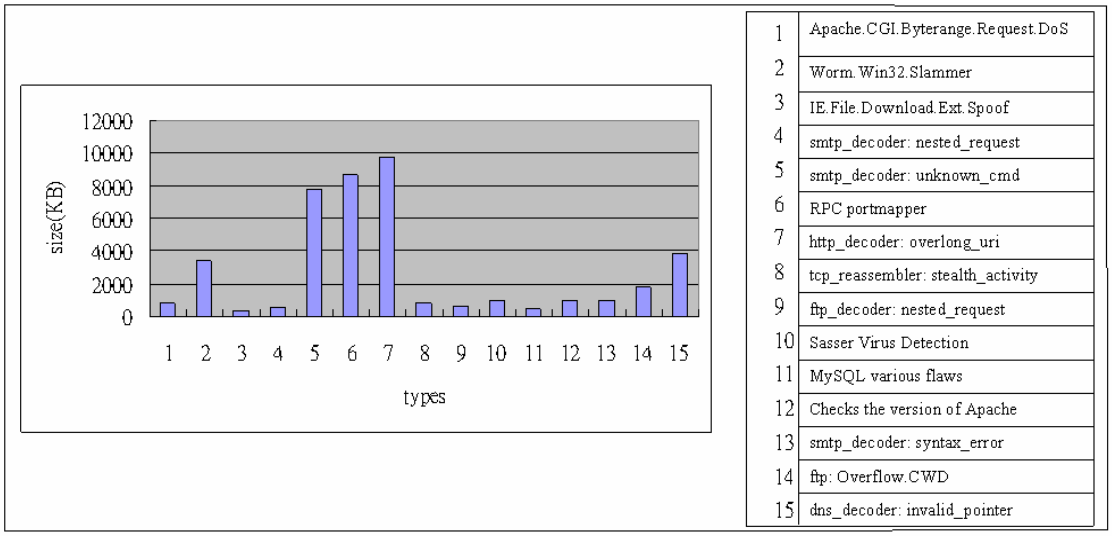

Figure 5 the file sizes of the 15 attacks

4.1.2 The file size of the extracted result

The definition of size is the file size of the extracted attack including all packet headers and payloads. Figure 5 shows the file size of extracted attacks by session extraction system. The x-axis is the 15 extracted attacks and order by the occurring frequency. The largest size of those attacks is 9731 Kbytes. The smallest size of those attacks is 491 Kbytes. However, the session extraction system just only excludes the "must not" packets of the attack from real traffic and reserves the possible packets of

the attack at the same session form the real traffic. Therefore, the extracted attacks of the same type maybe have different sizes at different dates. For example, a normal HTTP request maybe extracted with HTTP attacks when the attacks and the request happen at the same time between attacker and target. For this reason, we choose the attack size equal to the size that the most times in our experiment. If the attacks are the DDoS or DoS attacks we choose the attacks size by smallest size the IDP product can detect. Next section will describe the variability of the session extraction system.

4.2 Variation, completeness and purity

4.2.1 The variation of the session extraction system

The definition of variation in this work is the complement of the probability of the extracted attack's mode value. The mode value is the most frequent value. Therefore, the variation is defined by the equation

$$\text{Variation}(\text{Attack}_i) = (1 - P(\text{mode}(\text{Attack}_i))) * 100\%. \quad (4)$$

In our experiment, the different extracted attack sizes for each attack when they could classify as the same attacks come from the result of the comparing the attack size with the size that the most frequent size. The low variation of the session extraction system must be proved if we want to use the results of the session extraction system. In this experiment, we replay 100 attacks and the common real traffic at the same time. We mixed the 100 attacks with 10 different real traffics to observe the variation. Therefore, there are total 10 results (the extracted attacks) of the each attack and the total 1000 results by session extraction system.

Figure 7, 8 and 9 show the 3 case of the result that we extracted the attacks from the real traffic. The x-axis is 10 extracted attacks of each attack. Figure 7 shows the case one that is the different sizes of attacks less than 3. In this experiment, the 37% of the 100 attacks were in case one. Figure 8 shows the case two that is different sizes of attacks equal to 0. In this experiment, the 46% of the 100 attacks were in case two.

Figure 9 shows the case three that is the different sizes of attacks more than 3. In this experiment, the 17% of the 100 attacks were in case three. Figure 6 shows the accumulated number of the attacks of each variation by increasing. The 83% of the extracted attacks is less than 30% variation. The 30% variation could be easy to choose the attack size equal to the size that the most times in our experiment. But, there are also 17% of the extracted attacks could be hard to choose the result of the experiment because they had high variation.

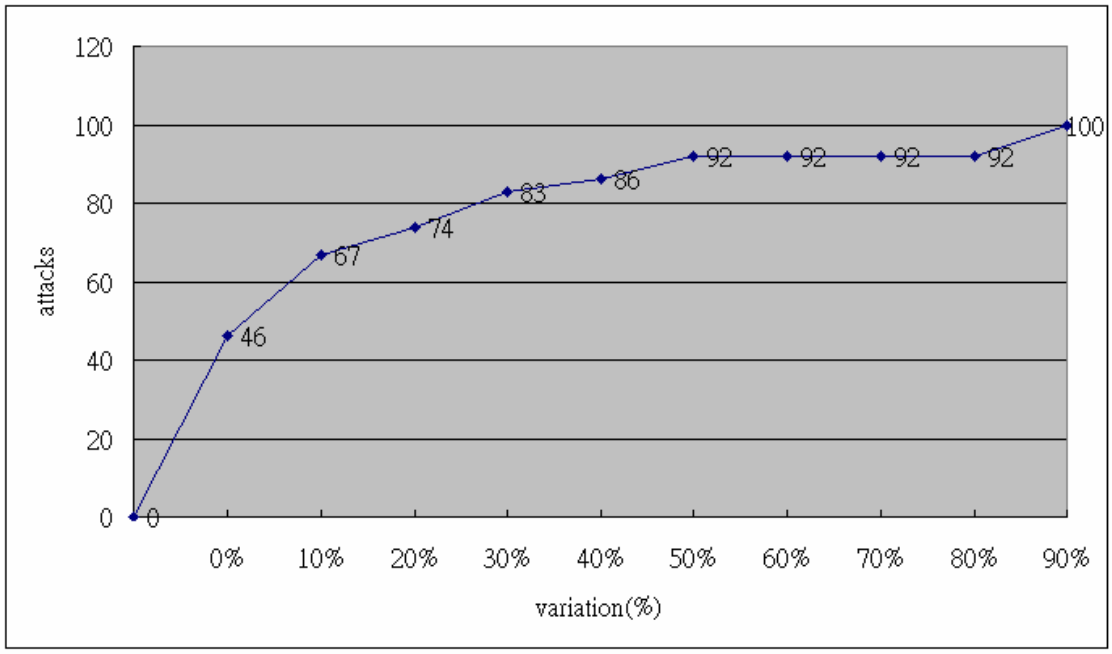

Figure 6. The variation of extracted attacks

4.2.2 The completeness and purity of the session extraction system

By adding a new step in the experiment of variation, we can use to observer the completeness and purity. The new step is only playing and recording the 100 attacks without other traffic. therefore, we can get original attacks of the 100 attacks. In this experiment, we compared the 10 results of each attack from the experiment of variation with the original attacks.

The definition of completeness and purity are as following. If the size of the extracted attacks is less than the original attack size, we will say the extracted attack is

not completeness. If the size of the extracted attacks more than the original attacks size, we will say the extracted attack is not purity. If the size of the extracted attack equal to the original attack size, we will compare the extracted attack with the original attack by bits comparison. The bits comparison compares to the bits in the extracted attack file and in the original attack file individually. If the extracted attacks are different with the original attack, we will say the extracted attacks are not completeness and purity.

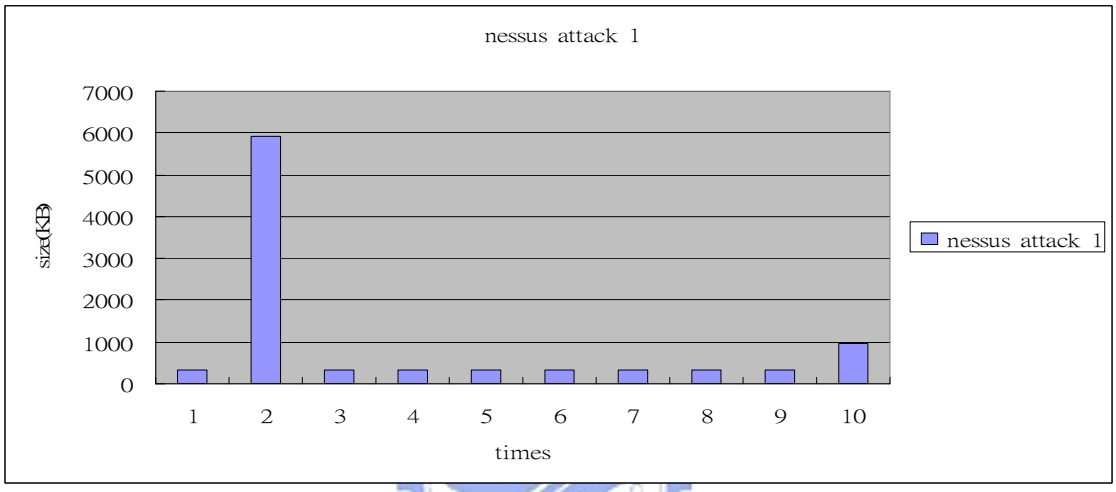

Figure 7. The different sizes of attacks less than 3

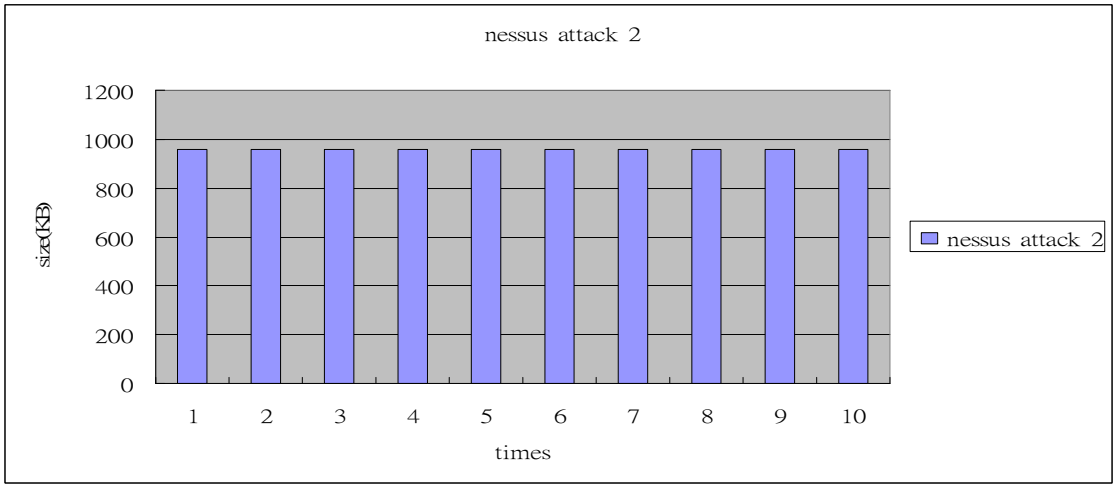

Figure 8. The different sizes of attacks equal to 0

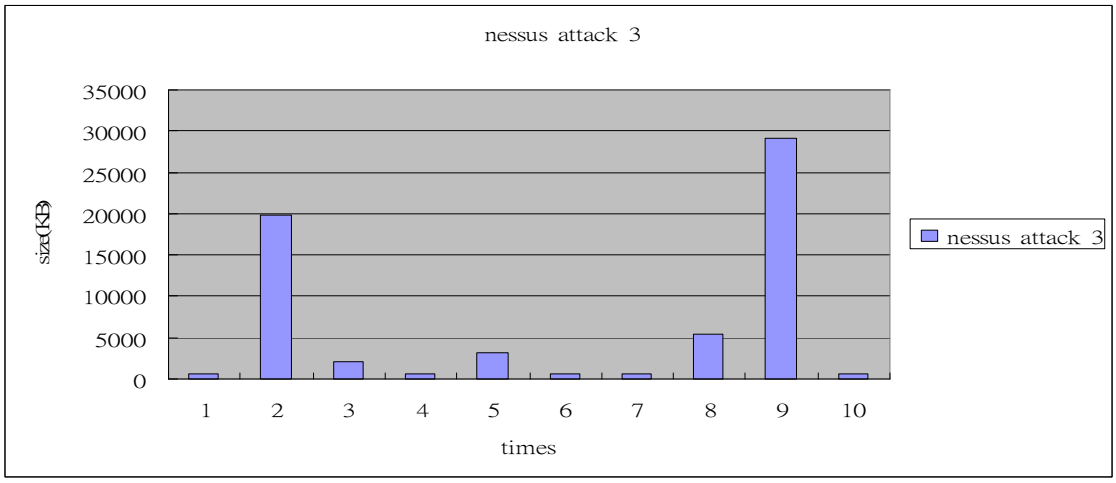

Figure 9. The different sizes of attacks large than 3

For this definition, we consider an attack as completeness and purity by sizes comparison and bits comparison. However, because the different size between the extracted attacks and the original attack can not comparing the bits, we can not definition the completeness if the size of the extracted attack is large than the original attack size (they are not purity already). For this reason, we assumed they are "Undefined". We also assumed some attacks can not define the purity for the same reason.

By our definition, the total 370 attacks in the case one had 71% completeness (1% not completeness by different size, 4% not completeness by comparing and 24% can not define) and 71% purity (24% not purity by different size, 4% not purity by comparing and 1% can not define). The total 460 attacks in the case two had 100% completeness and 100% purity. The total 170 attacks in the case three had 12% completeness (23% not completeness by different size, 29% not completeness by comparing and 36% can not define) and 12% purity (36% not purity by different size, 29% not purity by comparing and 23% can not define).. Table 3 shows the comparing of those three cases.

The completeness and purity of the case three was very interesting. By our

observation, the 170 attacks (the 17 different types mix with the 10 real traffic) of the case three had 15 DDoS or DoS attacks. The size of DDoS and Dos attacks are not fixed. The 500 packets, 300 packets and 1000 packets of a DDoS or DoS attack all can say they are the attack. Therefore, the 11th attack that we recorded by playing a DDoS or DoS attack can not be a standard because the smaller or larger one also can be considered as attack. For this reason, the completeness and purity are not good in case three and the variability also because the same reason.

Table 4. The completeness and purity of the extracted attacks

	(Variation) Number of different sizes	Completeness (Purity)	Not Completeness (Purity) by different sizes	Not Completeness (Purity) by comparing	Undefined
Completeness (Total 370)	0<Variation ≦30%	262(71%)	4 (1%)	15 (4%)	89 (24%)
Purity (Total 370)	0<Variation ≦30%	262 (71%)	89 (24%)	15 (4%)	4 (1%)
Completeness (Total 460)	Equal to 0	460 (100%)	0 (0%)	0 (0%)	0 (0%)
Purity (Total 460)	Equal to 0	460 (100%)	0 (0%)	0 (0%)	0 (0%)
Completeness (Total 170)	Variation >40%	20 (12%)	39 (23%)	50 (29%)	61 (36%)
Purity (Total 170)	Variation >40%	20 (12%)	61 (36%)	50 (29%)	39 (23%)

4.3 Analysis of the difference between Nessus traffic and the real attacks

In this experiment, we also play the Nessus Plug-in traffic that can mapping to those 15 attacks. Figure 10 shows the sizes of the nessus traffic and the real attacks in the same attack, we only choose the largest 4 attacks to show their relation. We can see the Nessus traffic have the fewer size. The Nessus Plug-in traffic often only has the emulated attacks (sometimes they only have request of the service) and waiting the response to know the existence of the vulnerability. Therefore, the size of the Nessus Plug-in traffic is usually smaller than the real attacks.

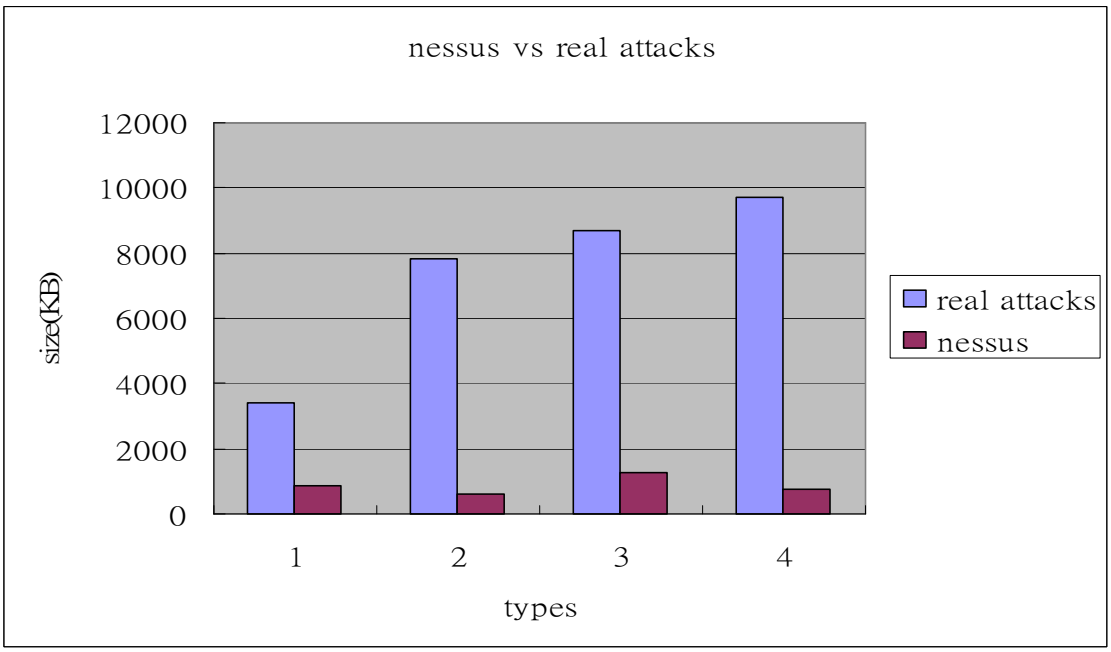

Figure 10 The real attacks v.s Nessus traffic

Chapter 5

Conclusions and Future Work

This work proposes a session extraction algorithm to extract a complete episode of attacks from a large amount of real traffic. The extraction uses the three-pass scanning to find out the anchor packet of an attack. It excludes the packets that do not belong to an attack from the real traffic and reserves the possible packets of the attack in the same session. Similarity between two packets is also defined to extract the attacks of DDoS and DoS accurately.

In the evaluation of the session extraction system, 15 attacks can be extracted from the real traffic in our lab. Six of the 15 attacks can be mapped to Nessus plug-ins. By the session extraction system, we can extract the Nessus plug-in traffic that had mapping with the extracted attacks. In Section 4.1 and Section 4.3, the result of the experiment told us the size of Nessus plug-in traffic often less than real attack, and the Nessus Plug-in traffic often only request the service and waiting the response. In this work, we also prove the variation of our session extraction system is low. The 87% of our extracted attacks had low variation. The lower variation of the extracted attacks also had high completeness and purity. The variation of the extracted attacks less than 30% had 87% completeness and purity. The variation of the extracted attacks equal to 0 had 100% completeness and purity. However, the higher variation of the extracted attacks had low completeness and purity, but it is because the DDoS and DoS attacks had no fixed size. This work also proved the Nessus Plug-in traffic is the smaller than the real attacks. The Nessus Plug-in traffic requests the system service and waits the response of the service. The real attack not only requests the services but also destroy the system.

The future work is automatic inserting the new attacks into Nessus Plug-in. Also, the Nessus Plug-in can add a new function that test a system by normal Nessus

detecting and real attacks detecting.

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] Christoph L. Schuba, Ivan V. Krsul, Markus G. Kuhn, Eugene H. spafford, Aurobindo Sundaram, Diego Zamboni, "*Analysis of a Denial of Service Attack on TCP," sp, p. 0208, 1997 IEEE Symposium on Security and Privacy,* 1997.

[4] Vern Paxson, “An analysis of using reflectors for distributed denial-of-service attacks” *ACM SIGCOMM Computer Communication Review,* 2001

[5] Martin Roesch, “Network Security: Snort - Lightweight Intrusion Detection for Networks”, Proceedings of the 13th USENIX conference on System administration, November. 1999.

[6] T. H. Coemen, C. E. Leiserson, R. L. Rivest, ”Introduction to Algorithms”, pages 314-320, 1990.

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

[8] 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.