

Remodeling and Simulation of Intrusion Detection Evaluation Dataset

J. Qian

Computer Science and Technology
Department,
Tsinghua University,
Beijing, P.R.China

C. Xu

Technology Department,
Organizing Committee for
Olympiad,
Beijing, P.R.China

M. L. Shi

Computer Science and Technology
Department,
Tsinghua University,
Beijing, P.R.China

Abstract - *Although the intrusion detection system (IDS) industry is rapidly maturing, the state of intrusion detection system evaluation is not. Although the off-line dataset evaluation proposed by MIT Lincoln Lab represents a significant undertaking, there remain several issues unsolved in design and modeling of the resulting dataset which may make the evaluation results biased. In this paper we present our efforts to improve on the traffic simulation. Unlike the existing model, our model takes advantage of user-level web mining, automatic user profiling and Enron email dataset etc, which is more reasonable for traffic modeling and simulation. The high fidelity of simulated traffic is shown in experiment. Moreover, different kinds of attacker personalities are profiled and more than 300 instances of 62 different automated attacks are launched against victim hosts and servers. All our efforts try to make the dataset more "real" and therefore be fairer for IDS evaluation.*

Keywords: Intrusion Detection, Evaluation, Dataset

1 Introduction

For more than a twenty-year's evolution, intrusion detection system (IDS) now has become an essential part of network security metrics. While the intrusion detection system industry is rapidly maturing, the state of intrusion detection system evaluation is not. Evaluations that focus on intrusion detection algorithm and system performance are essential for ongoing research because they can contribute to rapid research progress by revealing the weak points of current algorithms and systems. As far as it goes, there is no evaluation standard or methodology widely accepted to evaluate the performance of IDS. Most of the available IDS performance testings are based on odd packet sizes, flood packets, bizarre traffic mixes, TCP sessions that are mangled or incomplete and other things even won't happen in real world networks. Part of the mistake is due to a dearth of a comprehensive and complete evaluation methodology. Although the best way to evaluate an intrusion detection system is to test it in a live circumstance with real network traffic, the repeatability of experiment and concerned privacy of real network traffic deny it as a

general solution. While there is no substitute for live network traffic, there are ways of designing tests around real world environments, so synthetic traffic comes close to that of real world conditions.

In 1998 (again in 1999), MIT Lincoln Lab (MIT LL) conducted a comparative off-line evaluation of intrusion detection systems[1]. It is the most comprehensive evaluation of research intrusion detection systems that has been performed to date and provides a basis for pointing out the flaws and weakness of existing intrusion detection systems. Many researchers used the resulting dataset in their researches and experiments, such as [2-6]. Although the dataset evaluation represents a significant and monumental undertaking, some methods and models associated with its design and execution were questionable, such as HTTP traffic modeling, message classification and body construction, attacks injection, validation of simulated traffic and etc. Some researchers have noticed such problems and criticized the design and execution of the dataset, but there is no technical contribution for new efforts proposed *per se*[7]. Thus, the unsolved problems clearly remain. Besides the shortcomings of the dataset itself, the network has changed much during the last five years. New applications and attacks are booming. Such factors reshape the network streams and have effect on performance of IDS. Thus an improved and updated dataset is in requirement.

The rest of the paper describes our efforts to redesign and generate new evaluation dataset. In section 2, we give a brief summary of our test bed and reference network. Details concerning the background traffic simulation and attack scenarios design are discussed in section 3 and 4. We make our resulting dataset a brief comparison with that of MIT LL in section 5. Finally the conclusion is given in section 6.

2 Overview of Testbed

A reference network is required for dataset evaluation methodology. We choose a university laboratory LAN as our reference network. A three-month of continuous

sniffing data was collected on reference network and used to create statistics. Figure 1 shows our isolated test bed network used to generate background traffic and attacks. The left side by CISCO ROUTER represents the inside of the emulated reference network and the right side by CISCO ROUTER represents the outside Internet. Both automated and manual attacks were launched outside against the inside victim servers. The generated live background traffic is similar to reference network.

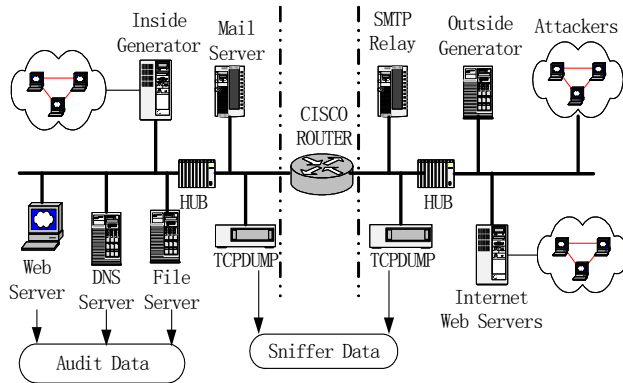


Fig.1. Test Bed of Background Traffic and Attacks Generation

3 Background Traffic Simulation

The dataset is a comprehensive simulation and synthesis of traffic in essence. The synthesized nature of the traffic allows widespread and relatively unrestricted access to the evaluation dataset. The contents of the dataset consist of a large amount of simulated network traffic (called background traffic) such as HTTP, SMTP, Telnet, POP, P2P and a small amount of elaborate attacks (called attack traffic). Background traffic is crucial for evaluation because it represents the network traffic characteristics, including details that affect false alarm rates.

3.1 HTTP Traffic Simulation

The problem with the model of HTTP traffic simulation of MIT LL is that the traffic is simply simulated in terms of broken sessions rather than individual personalities. Only a list of web sites visited by folks is gathered and shared by all users in simulation. The user for each session is randomly chosen, as are the web pages accessed and the number of pages accessed per session (randomly chosen between 1 and 15). While it is easy for design and execution, the user personalization is melting. Although the dataset claimed a wide feasibility for the evaluation of both anomaly based systems and signature based systems, the simplicity of web traffic design and simulation may affect the effectiveness of some anomaly based systems because they are studying the network traffic and user behaviors. Such systems are sensitive to network behaviors. MIT LL implicitly claimed that it used sniffer analyses to obtain a list of web sites visited by folks from

LAN. As a method we have tried, we find that the sniffer analyses faced two shortcomings to improve user personalization. One problem of the information capture is not simply a question of abundance, but also of granularity. For example, if a web page contains two or more frameworks, it is very difficult for sniffer analyses to tell whether the requests come from a single page or different pages when we intend to infer the user web behaviors. Moreover, the security HTTP connections are invisible to sniffer analyses.

Since windows interface are ubiquitous nowadays, user behaviors are very likely to be concurrent and interlaced, it also offer the challenge to simulate user behaviors. The method used by MIT LL is limited to provide user personalization and concurrent user behaviors. We are interested in how to profile personal browsing patterns, and more importantly, how to simulate a group of web users browsing the Internet simultaneously. To most computer users, the web browser is the gateway to Internet. The web browser can be a key element in studying the individual patterns. Thus we propose a method monitoring at web browser level that enables richer data collection and better personalization. Most of the users use Microsoft Internet Explorer (IE) as their web browser. The implementation of IE is based on the framework of COM, which makes it possible for us to design and implement a plug-in program to audit user's web behavior. IE implements its own event object model that defines a series of events. We hooked our own codes into event handler body which initially handled by operating system. This is how we audit the browsing behavior of web users. The plug-in program is provided as a dynamic library link (DLL) so that the data collection process is transparent to user and won't interfere with user activities. Every user's web browser log is preprocessed for data reduction and converted into a program readable format before being fed into the personalization model. We use probability transition diagram as part of our personalization model. Several definitions are necessarily declared before taking further steps.

Definition 1 (Request URL) The *URL* of the web page navigated by the users is notated as a four-tuple $URL=(SCHEME, NAME, PAGE, PARA)$. *SCHEME* is the name of the protocol, *NAME* is the DNS name of the machine where the web page is located, *PAGE* is the name of the file containing the web page, the *PARA* is parameter string which is required by dynamic web pages (i.e. the php or jsp web pages).

Definition 2 (Navigation Request) A navigation request is notated as a four-tuple $R=(URL, sT, dT, wID)$. *sT* is the start time of the navigation request, *dT* is the duration of the navigation request, *wID* is the handle of the window (a long integer) where the navigation behavior happens.

Definition 3 (Class) if there exist a set of *URLs* which have the same *NAME* property, then we define them as a Class

$C=\{url_1,url_2,\dots,url_n\}$, satisfying that $\forall url_i \in C, 1 \leq i \leq n, NAME(url_i)=name$. $NAME(url)$ means the operation of getting the name property from request URL . $name$ is also defined as the class name of C . The number of the elements that belong to a class C is defined as the degree of the class, which is notated as $|C|$. As can be seen, every request URL falls into a unique class.

Definition 4 (Subsession) A subsession $U=(r_1,r_2,\dots,r_n)$ is a set of navigation requests launched in time sequence. $\forall i, j, 1 \leq i < j \leq n, sT(r_i) < sT(r_j), wID(r_i)=wID(r_j)$, and all the navigation requests of U belong to the same class. The start time of subsession is identical with that of the first navigation request, $sT(U)=sT(r_1)$. The length of a subsession is defined as the number of its members.

Definition 5 (Trigger Relationship) If the first navigation request of subsession u_2 is triggered by the last navigation request of subsession u_1 , then we define a trigger relationship t between u_1 and u_2 , t is a pair of class names. If $u_1 \in C_{name1}, u_2 \in C_{name2}$, then $t=(C_{name1}, C_{name2})$.

Definition 6 (Session) Session $S=(\sum U, T)$, where $\sum U=\{u_1,u_2,\dots,u_n\}, T=\{t_1,t_2,\dots,t_{n-1}\}, n \geq 1, \forall i, 1 \leq i \leq n, \exists j, 1 \leq j \leq n, j \neq i$. Where subsession u_i triggers subsession u_j or is triggered by subsession u_j . The session length equals to the number of members of $\sum U$.

The context mining is to label the trigger relationship between subsessions. Two kinds of trigger relationships are labeled. One is between the subsessions launched in a same window. The other is between the subsessions in different windows by clicking a link and opening a new IE window. An example is given in figure 2, where shows four IE windows launched by a user.

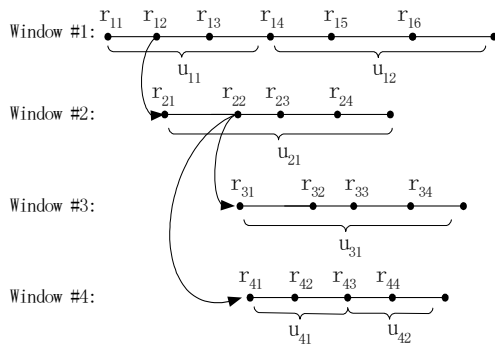


Fig.2. An example of trigger relationship

Windows are listed in sequence of start time. Window #1 contains six navigation requests that belong to two different subsessions, u_{11} and u_{12} . Window #2 is triggered by r_{12} , containing a single subsession u_{21} . Window #3 contains a single subsession u_{31} triggered by r_{22} . Window #4 is triggered by r_{22} , containing two subsessions, u_{41} and u_{42} . Suppose $u_{11}, u_{12}, u_{21}, u_{31}, u_{41}$ and u_{42} belong to class C_1, C_2, C_3, C_4, C_5 and C_6 respectively, then we can deduce the trigger relationships between the subsessions. The

corresponding probability transition diagram is shown as figure 3.

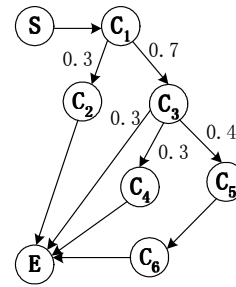


Fig.3. Session probability transition diagram

The S state and E state indicate the start and end of a user session. Each edge of the diagram is labeled with the probability that the corresponding trigger relationship occurs in the sessions (some of the edges are not labeled explicitly if the probability is 1.0). Each path from the start to end corresponds to a session. Each node the path passes by is a class that consists of a set of subsessions with statistical information. The session probability transition diagram of a real user's web browser log is much more complicated than the given example.

Besides the session probability transition diagram, a user profile also includes a user's daily connection distribution, daily connection cumulative density and session length distribution. We analyzed the connections in five-minute intervals over a three-month continuous user log to discover how user varies during a day. After user personalization, a corresponding user profile is created and stored as a knowledge base for simulation. As a knowledge base, the user profiles can't be used for traffic simulation directly. A simulation configuration is essential for simulation. It includes the global information that is essential to simulation, such as the total connections during a day, IP address of every simulated user, the amplification factor (AF) and the algorithm of generating virtual user profiles (if AF is set greater than 1). Each virtual user profile is synthesized by existing user profiles. The virtual user profile is very useful because we don't really need to gather the equivalent number of user profiles if we are going to simulate a large group of web users. We can achieve the number of user browsing plans we required by adjusting the value of AF . For example, we can achieve 20 different browsing plans by setting the value of AF as 2 if we only have a knowledge base of 10 user profiles available. As can be seen, the bigger AF is, the more virtual user profiles are synthesized and the worse simulation fidelity is. It is a trade-off between fidelity and scalability. The algorithm of generating virtual user profile is also very crucial to the fidelity of simulation. Here, we adopt a full copy algorithm, where AF is set as 1.

Each virtual user profile is converted into a browsing plan for simulation and submitted to the scheduler. The scheduler program looks through all of the submitted browsing plans in order to check the possible collisions and then dispatch a timetable for each host simulator. The host simulator is a software automaton that is responsible for simulating web user activities according to the timetable assigned by scheduler program. It allows an actual host to appear as if there are a lot of web users surfing the Internet. All the pages are cached on the outside web server that simulates the whole Internet. If a page requested by automata doesn't exist, a "404 not found" error information is returned, as if the pages are removed.

3.2 SMTP Traffic Simulation

SMTP traffic simulates emails being sent to and from users on the LAN and on the Internet. Figure 4 shows the SMTP connections of four-month statistics on reference network.

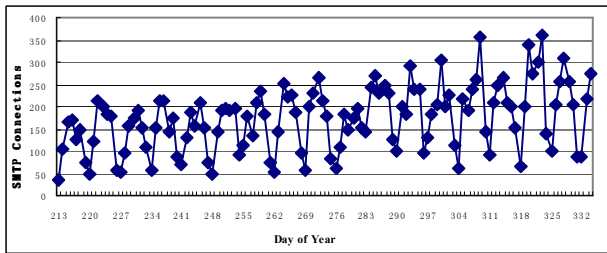


Fig.4. SMTP connections of four-month statistics

The difficulties of SMTP traffic simulation mainly focus on message classification, contents generation and thread maintenance. MIT LL manually classifies the mail messages into seven different types. Each type accounts for a specified percentage of total messages. Unlike MIT LL, we use a clustering method to automatically classify the mail messages.

Definition 7 (Mail Message) A mail message is defined as a nine-tuple, notated as $msg=(FROMADDR, RCPT, TOADDR, TITLE, TIMESTAMP, SIZE, STAT, ERRMSG, CONTENT)$. $FROMADDR$ and $TOADDR$ denote the sender and receiver respectively. $RCPT$ is the number of receivers of this message. $TITLE$, $SIZE$ and $CONTENT$ denotes the title, size and content of the message respectively. $TIMESTAMP$ is the sent time of the message. $STAT$ denotes the status of the message, successfully sent or failed. The error information is stored in $ERRMSG$.

Definition 8 (Message Class and Degree of Message Class) A message class C contains the messages that have identical $(FROMADDR, TOADDR)$ pairs. The quantity of messages contained by a class is defined as the degree of the class, notated as $|C|$.

Definition 9 (Supporting Rate and Frequency) For each class C , defines the supporting rate $S_c=|D_c|$, where D_c is a set of integers. Every member of D_c is between $[0, 365]$, which denotes the day of year of message $TIMESTAMP$. The frequency of class C is defined as $f_c=|C|/S_c$.

Definition 10 (Thread) A thread is a set of email messages discussing a particular topic.

Figure 5 shows the clustering results. The classes can be mainly classified into four types. Each type is briefly described in table 1.

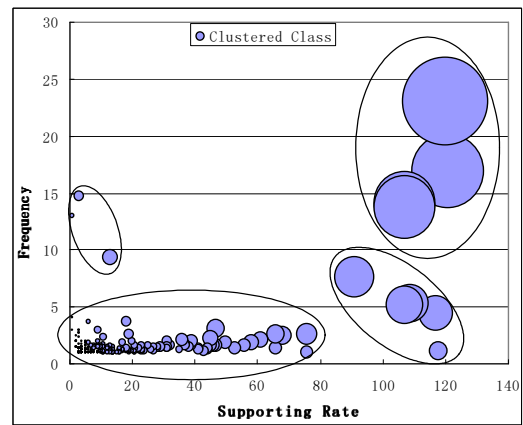


Fig.5. Distribution of Clustered Classes

Table 1. Class types and brief description

Type	S_c	f_c	Quantity	Description
I	<80	<9	Large	Most are normal messages
II	<80	>9	Small	Spams, viruses or worms
III	>80	<9	Large	List or newsgroup
IV	>80	>9	Large	List or newsgroup

Message contents generation is implicitly related with thread maintenance. MIT LL generates email messages in two ways. Some email messages are actual messages downloaded from a variety of public domain mail list servers. The others are created by using statistical bi-grams frequencies to preserve word and two-word sequence statistics from a sampling of roughly 10,000 actual email messages. To avoid concerned privacy, all the email messages are filtered using a 40,000 words dictionary to remove names and other private information. MIT LL mentions little about thread classification and maintenance. Although all the recipients have a probability to respond to the sender, the response message is randomly chosen, which breaks the thread information.

The model adopted by MIT LL is questionable because it has several shortcomings. Using filtered actual messages as SMTP traffic is the same as using filtered real network traffic as background traffic in essence. Since filtered real network traffic is denied by MIT LL as a

solution to background traffic simulation, it is not reasonable to using filtered actual messages as SMTP traffic. In addition, neither a 40,000 words dictionary is big enough for privacy filtering nor it is practical for administrators to check through all the created messages to avoid concerned privacy. And more importantly, after filtering by such a huge words dictionary, most of the filtered emails are predictably unreadable or hardly understandable for human. It is even worse for the messages created by using statistical bi-grams frequencies.

The difference between thread information and the other kinds of email data is that thread data is not provided explicitly. It must be deduced from other data fields. There has not been much work in how to detect thread information automatically. Thus, we use a publicly released email dataset for message contents generation and thread maintenance. The Enron email dataset[8] is a large set of email messages with threads detected and classified. There are totally 200,399 messages in the Enron corpus with 101,786 threads. 71,695 of these threads were trivial threads, consisting of only one message. Other 30,091 remaining consists of 123,501 messages. The average thread consist 4 messages, which is similar to that of reference network. Messages with empty subjects are not considered to be a thread. SMTP sessions are generated from both the inside Mail Server and outside SMTP Relay Server during the time of simulation.

3.3 Other Types of Traffic Simulation

We give a short description of other types of traffic simulation because of page limitation. They include the network applications and services such as telnet, ssh, ftp, P2P and etc.

FTP sessions simulate inside users using FTP clients to transfer files to and from file server inside and outside. According to our analysis on sniffing data, FTP traffic is a big but not important traffic because most of the FTP traffic is data traffic. Hence we make a trade-off for FTP traffic simulation. We simulate the FTP sessions of commands but omit the file transferring. Otherwise the dataset size may become too big to be portable and publicly released.

Telnet traffic simulates interactive sessions of users on remote servers. Users are simulated based on individual statistical profiles. Thousands of history commands are collected for user personalization. Telnet traffic simulation needs to log every user's commands during the simulation so as to roll back to the previous state after simulation if it is necessary. Some commands don't change the state of the server (e.g., ls, cd), some commands do. For example, if a programmer user edits programs and compiles them, and finally run the programs. Some files are created by commands (vi and cc) after simulation.

P2P traffic is a highlighted new member of dataset. The P2P applications running on reference network include file exchanging applications, instant messengers and real time video broadcasting application. The file exchanging applications and real time video broadcasting application are relatively easy to simulate because they are less interactive. Instant messengers are hardly to simulate because of too much concerned privacy. Thus we give up the simulation of instant messengers in experiment.

Since most of the users on reference network are working on windows workstations, windows GUI applications are ubiquitous. We developed *ScriptMaster* scripts specially used for GUI interactive simulation. It takes advantage of the message-driven nature of Windows and allows user automata behave as if a real user was using the applications.

4 Attack Scenarios

All attack initiations are stored in a database and scheduled by attack scheduler automata. All the exploit scripts are collected from the Internet and maintained in a vulnerability database internally. The vulnerability database contains sorted information concerning vulnerabilities and hundreds of exploit scripts we used in our experiment. Each exploit is tested with result being recorded before it was used in experiment. Replaying attacks restricts the kind of attacks that we launched in test bed. Some attacks obviously depend on the environment. For example, the buffer overflow attacks. Such factors reduce the generality of attacks targeted to a specific kind of application, host, operating system, service or architecture.

We add a lot of new attacks that were not described and implemented in MIT's dataset. Table 2 shows the attack categories and the numbers of instances.

Table 2. Attack categories and numbers of instances

Categories	Numbers of instances
Probes	156
WWW Attacks	68
DOS/DDOS	3
FTP Attacks	2
Remote Services Attacks	19
Manipulation/Spoofing	6
Windows Attacks	18
Malicious Code Attacks	5
SMB/NetBIOS Attacks	6
SMTP Attacks	3
Infrastructure Attacks	5
DNS Attacks	7
RPC	12
Network File System Attacks	-

In addition, we generate emails containing malicious attachment (trojans, viruses, worms and malicious code) as attacks because email now has become a usual channel for

attackers to compromise computers. For example, some emails are generated with malicious attachments that allow remote attackers access any file on the compromised system. Totally more than 300 instances of 62 different automated attacks were embedded in generated dataset.

5 Experiment

MIT LL claimed that the generated dataset is similar to that observed on reference network, but the statistics used to describe the real traffic and the measures used to establish similarity are not given, except for the claim that word pair statistics of email messages matching those sampled.

We bring our synthetic traffic in comparison with MIT's dataset (Monday, Week 1) and observed traffic respectively. We analyze the distribution of total HTTP connections according to the number of packets of each HTTP connection.

Figure 6 shows the result of MIT's dataset. We present ours in figure 7. A significant difference between figure 6 and 7 is that the distribution curve in figure 6 is absent when packet number is less than 8. This indicates that none of HTTP connections contains less than 8 packets. It is quite odd and not true of the characteristic of real network traffic. For example, if a requested page doesn't exist (often caused by typos), then the connection will be closed. Figure 8 shows the result of observed traffic on our reference network. We only give the result of our reference network because MIT didn't publish their observed network traffic, neither can we repeat it.

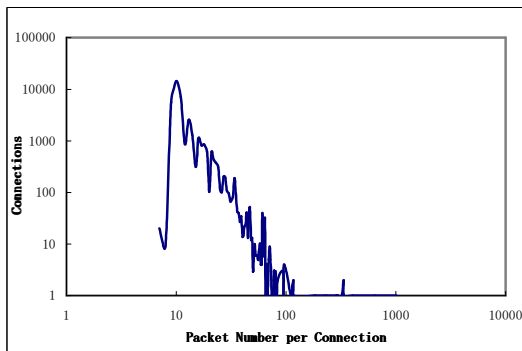


Fig.6. Statistics of HTTP connections of MIT's dataset

The x-axis distribution of curve in figure 6 is quite narrow, and the packet number mainly falls between 10 and 100 because the number of pages accessed per session is randomly chosen between 1 and 15 without user personalization. By contrast, the x-axis distribution of curve in figure 7 is much broader because it is based on user personalization. The curve in figure 7 is very similar with

that in figure 8, which indicates that the simulated traffic keeps high fidelity to real network traffic.

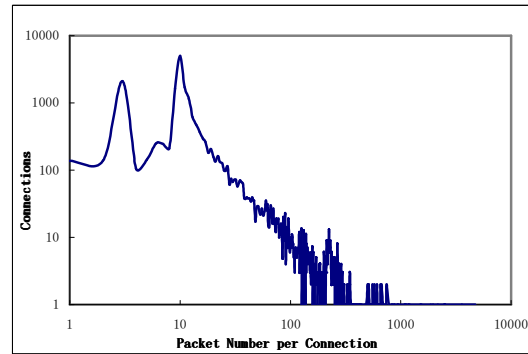


Fig.7. Statistics of HTTP connections of our dataset

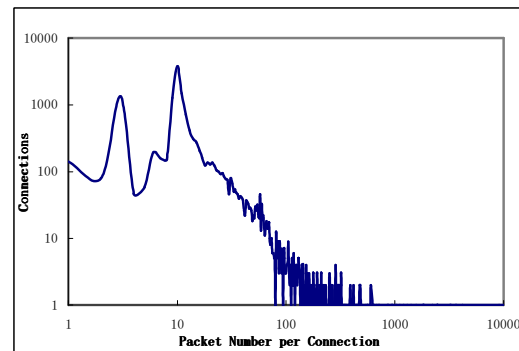


Fig.8. Statistics of HTTP connections of reference network

6 Conclusions

The off-line dataset evaluation methodology proposed by MIT LL is a practical solution in terms of evaluating the performance of IDS, which contributes to research progress by revealing the weak points of current algorithms and systems. Like other open project and research, continuous improvement makes it a key element to keep the dataset valuable for researchers. Network bandwidth is increasing quickly with booming applications and attacks. Such factors require the dataset to be updated in time.

User behavior simulation is very important in IDS evaluation. Simulating how user behaves is an immensely challenging undertaking because of the complexity and intricacy of human behaviors, but it does not mean to weaken the efforts by claiming too many difficulties for them. MIT LL models the synthetic traffic from session level rather than from user level. We think it more reasonable to model the synthetic traffic from user level because it is the user behaviors that have effect on traffic sessions rather than the converse. Therefore, we adopt a model taking advantages of user-level behavior mining and user pattern profiling. It gives our model the ability to

feasibly construct synthetic traffic with high scalability and fidelity, which make the dataset more “real” and therefore is fairer for IDS evaluation.

7 References

[1] R. P. Lippmann, D. J. Fried and I. Graf *et al.*, “Evaluating Intrusion Detection Systems: The 1998 DARPA Off-Line Intrusion Detection Evaluation,” In Proc. of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX), Los Alamitos, CA, pp.12-26, 2000.

[2] P. Ning and Y. Cui, “An intrusion alert correlator based on prerequisites of intrusions,” Technical Report TR-2002-01, Department of Computer Science, North Carolina State University, January 2002.

[3] V. M. Matthew and K. C. Philip, “Learning nonstationary models of normal network traffic for detecting novel attacks,” In Proc. of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining, Edmonton, Alberta, Canada, pp. 23-26, July 2002.

[4] R. Sekar, A. Gupta and J. Frullo *et al.*, “Specification Based Anomaly Detection: A New Approach for Detecting Network Intrusions,” In Proc. of ACM CCS’02, Washington DC, USA, pp. 18-22, Nov. 2002.

[5] W. Lee and S. Stolfo, “A Framework for Constructing Features and Models for Intrusion Detection Systems,” *ACM Transactions on Information and System Security*, Vol 3, No. 4, pp. 227-261, Nov. 2000.

[6] W. Ke and J. S. Salvatore, “Anomalous Payload-based Network Intrusion Detection,” In Proc. of 7th RAID2004, pp. 203-222, Sep. 2004.

[7] J. McHugh, “Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory,” *ACM Transactions on Information and System Security*, Vol 3, No. 4, pp.262-294, Nov. 2000.

[8] K. Bryan and Y. Yiming, “The Enron Corpus: A New Dataset for Email Classification Research,” European Conference on Machine Learning, Pisa, Italy, pp. 217-226, Jan. 2004.