Experimental Design for Attack Scenario Traces to Validate Intrusion Detection Alert Correlations

Jia Wang and Linda Zhao

April 2006

WSRC Paper 2006/4-1

Wharton-SMU Research Centre
Wharton-SMU Research Center

Experimental Design for Attack Scenario Traces to Validate Intrusion Detection Alert Correlations

Jia Wang and Linda Zhao

This project is partly funded by the Wharton-SMU Research Center, Singapore Management University
Experimental Design for Attack Scenario Traces to Validate Intrusion Detection Alert Correlations

Jia Wang and Linda Zhao

University of Pennsylvania

1 This research is sponsored in part by NSF and Wharton-SMU Research Center.
1 Introduction

The number of Internet security incidents is increasing continuously each year according to CERT [12] (Computer Emergency Response Team Coordination Center located at the Software
Increasing security incidents make intrusion detection a very important area. Intrusion detection technology emerged soon after the presence of network attacks, aiming at detecting inappropriate activities along the network traffic or malicious behavior on computer systems. However, it is difficult to evaluate the effectiveness of these systems. And realistic network attack traces are necessary in order to evaluate the effectiveness of ID systems, to validate the performance to alert aggregation method and to train those anomaly ID systems that aim at modeling traffic or host based behaviors.

Although previous work of MIT Lincoln lab provided us with a comprehensive research in the evaluation of ID system, there are several shortcomings. To overcome its shortcomings identified in following section is the major motivation of our experimental design. The objective of our design presented in this paper is to come up with realistic attack traces for relevant
researches to evaluate the effectiveness of the ID systems, the effectiveness of integration of
disparate alerts and to train traffic anomaly models.

The objectives of our experimental design are as follows:

1. To select a combinations of attacks with representative hacking skills that are frequently
   utilized by attackers;
2. To ensure that the attack scenarios designed in our work cover different complex attack
   structures;
3. To ensure that the attack scenarios include attack steps with reasonable inherent
   correlations;
4. To design realistic attack scenarios with the consideration of diversified hacking
   behaviors revealed by attackers with different personalities.
5. To control the distributions of different attack categories so that they conform to the real
   distributions on Internet, and
6. To generate enough attack instances.

In this paper, we present an approach that can fulfill the above objectives, and describe our
work to generate the datasets.

1.1 Analysis on existing attack traces for IDS evaluation

In the year of 1998, 1999 and 2000, researchers at Lincoln lab of MIT generated datasets in
the project of DARPA Intrusion Detection System Evaluations. The datasets contain synthetic
data of background traffic and attack traces, which were designed to simulate the real traffic to
and from a typical Air Force Base.

Since the project carried out at MIT is almost the most comprehensive effort that was made
in the area of dataset generation for IDS evaluation and available to the public, the datasets have
long been cited and used in related experiments to validate different models and methodologies to
improve ID system performance.

However, there are limitations on the MIT datasets as follows:

- **Validation of background traffic data.** In [40], John McHugh pointed out the lack of
  validation of the background traffic data that is supposed to be similar to the real traffic
  observed in the network of the Air Force Base. And there is no statistical measures
documented to describe the real traffic in the publications.

- **Insufficient of attack instances.** A variety of intrusions regarded to be representative were injected into the simulation traffic. However, the number of attack instances is relatively very small and do not meet the requirement of any training method. There are only 172 instances in the 1998 dataset and less than a hundred in the 1999 dataset. That is, it is almost impossible to design statistical tools to learn from the datasets or make use of the datasets to train certain model that involves statistics.

- **Insufficient of attack coverage.** Further more, the 1998 and 1999 datasets do not include attack behaviors that emerged afterwards. New vulnerabilities and new attacking skills are developed quickly during the past several years. Today, the most prevalent and intrusive attacking tools and scenarios might be completely different from the year of 1998.

- **No justification on distributions of attack categories.** In [40], McHugh also pointed out the distribution of major categories of the attack taxonomy used in the DARPA project (mainly User to Root, Remote to Local User, Denial of Service, and Probes) is not realistic when compared with other attack traffic analysis report. That is, the attacks invoked may not be realistically distributed in the background data. Besides, overall speaking, in average only 3 to 4 attacks were invoked per day during the experiment and this number is far below what is seen in a real network these days.

- **Lack of consideration in attack scenario design.** In reality, an attacker usually cannot grab the access of the target at one shot. An anatomy of attacks shows the generalized phases, which we discuss in details in next Section 2.1.1. An attacker usually collects information to prepare for intrusive actions through scanning or enumerations. And after gaining normal user access, he often tries to grab super-user privilege and then install backdoors, Trojan horses or crack other trusted links from there. Intuitively, actions in a sequence of an attack are correlated with each other in the sense that they might be invoked by the same person and originated from the same source or related sources. And some actions precede other ones. In short, to simulate the real traffic, scenario design is critical, especially for the training process modeling correlations among attacks or for the validation of algorithms trying to reconstruct attack scenarios.

- **The analysis unit of the dataset and the vagueness in false alarm rate:** Since most detection mechanisms work with packets and sessions, there exists very direct and
precise correspondence relationship between alerts and sessions. It is very natural that, in the experiment evaluation model of MIT Lincoln Lab, session is the unit of analysis. This model associated an attack with a single session. However, in the case of a complex attack or a multi-session attack, there is no such one to one correspondence between sessions and attacks. Most of the time, an attack is structurally complicated (details given in next chapter) and consists of multiple sessions. The definition of false alarm rate was never formally described in the documentations of MIT Lincoln Lab but the evaluation process and the datasets implied that false positive rate is the ratio of detected intrusive sessions and true attacks. Apparently sessions and attacks are not on the same comparable level. In our work, this is clarified through detailed analysis of sessions and attacks which is given in next chapter.

2 Background information

To generate sound and realistic attack traces, the following issues need to be investigated. This section summarizes our analysis and understandings on these aspects to serve as background information for our experimental design.

1. First we need to anatomize and generalize attack structure for the purposes of the following things:
   
a) To uncover the relationship of attacks and sessions under generalized attack structure;
   b) To help us understand the inherent correlations among attacking steps;
   c) To ensure that the scenario design will include all different kinds of structures.

2. The clarification of the notions of false positive rate and detection rate will justify the unit of analysis and the presentation that we choose for our datasets.

3. By looking into how hacker’s personality affects the hacking behavior, we would be able to ensure that our attack scenario contains interleaved attacks invoked by hackers with different personalities.

2.1 Attack Structure Analysis

Nowadays a large percentage of network attacks involve use of existing hacking tools, and the tools are getting more intelligent and structural in comparison with tools back in the late 90’s. They are also becoming more robust in the sense that terminating only one or several sessions of
an attack process will not stop the attack itself.

For example, 0x333samba is a buffer overflow hacking program making use of the vulnerability in Samba call_trans2open() procedure. The hacking process of 0x333samba is shown in Figure 2.1. It actually updates the four-byte shell code address continuously until the address matches what is on the target computer. Such an attack consists of multiple probing sessions and multiple hacking sessions to send the shell code. And the number of sessions in a 0x333samba attack varies as the environment (processes running on the target computer, in this case) changes. In our experiments (Section 4), 0x333samba was invoked 44 times altogether and the numbers of sessions in each attack varied from 1 to 137.

![Flowchart of 0x333samba hacking process](image)

*Figure 2.1 Hacking process of 0x333samba*

Although attacks have become more complex and structural, intrusion detection systems (both misuse and anomaly) still work in a flat way that only packets and sessions are checked and all correlation information of attack steps and structures are ignored. Such an alert simply implies that the particular packet or the particular session is probably intrusive. Obviously it is difficult to establish one to one correspondence between attacks and alerts, which complicates the definition of false positives and detection rates. In other words, current intrusion detection systems are not working at the same level of attacks but instead working at a lower level of sessions and packets. To clarify the confusion, in this section, we first analyze the process of an attack and then introduce the concepts of simple attacks, complex attacks and attack scenarios.

### 2.1.1 Attack Phases
Prior work [38] conducted on the generic process of network attacks defined different phases of an attack, illustrated by the following diagram:

![Figure 2.1.1 Attack Phases](image)

1. **Preparatory Phase.** A phase for information collection which includes:
   - **Footprinting:** The first thing attackers usually do is to come up with a complete security profile to describe security posture of the target organization by using a combination of tools and techniques, e.g., whois database query, DNS interrogation, network reconnaissance, etc. The most valuable information usually include address range, namespace, etc. that directly related to Internet, Intranet, remote access and extranet connection of the organization.
   - **Scanning:** Further target assessment will help attackers to identify servers alive in the address range of the organization, listening TCP or UDP services and types of operating systems running on the servers. Recent prevalent scanning tools include Nmap, X-Port, Twwwscan, ScanloOk, etc.
   - **Enumeration:** After non-intrusive information gatherings described above, attackers could then connect to the target and query for network resources and shares, valid users and groups, applications, etc. trying to dig out poorly protected assets. Enumeration activities differ from previous information gathering in the sense of involving active connections to the targeting systems.

2. **Phase of gaining access.** During this phase, informed attempts are carried out to access the target (usually as normal user) based on the information collected in preparatory phase. Typical techniques involved in this phase include password eavesdropping, file share, brute forcing, and the most popular buffer overflow. If normal user privilege is obtained in this phase, attackers usually will seek for higher privilege, otherwise, they sometimes turn to other techniques, e.g., denial of service, to disable the target or occupy all the bandwidth.

3. **Phase of escalating privilege.** After obtaining user level access, attackers usually begin to work towards super user privilege immediately to gain a complete control of the system.
Unfortunately to most of the operating system, it is difficult to prevent privilege escalation once the attackers have interactive logon connections. They will try to crack password or exploit known operating system vulnerabilities.

4. **Phase of pilfering.** From the compromised system, it is relatively easy for attackers to gather more information and identify avenues lead to the conquest of other trusted systems. They often evaluate trust links or search for clear text password file.

5. **Phase of covering tracks.** Covering up the intrusive traces is crucial to most attackers and their actions generally involves:
   - To disable the auditing
   - To clean the event log
   - To hide the attacking toolkits

6. **Phase of creating backdoors.** What are the attackers going to do with root privilege? Creating backdoors enables attackers to re-establish direct connections easily or to grab more valuable information for other intrusive actions. Backdoors could be laid in various parts of the system:
   - Rogue user accounts could be created;
   - Batch job scheduled to invoke certain programs periodically;
   - Startup files modified to ensure secret services be activated at boot time;
   - Application could be “Trojanized,” that is, be replaced by hideous Trojan horses, which purport to perform some useful functions but actually does completely different and malicious things behind the scenes.
   - Certain sniffing tools could be installed to monitor and capture telenet, ftp, pop or snmp passwords.
   - The binary code of operating system itself could be substituted by other software suite, which is given the name *rootkits*.

   Lots of things could happen in this phase, and all those actions are to ensure the compromised system behaves as designed by the attackers and they can re-gain privileges without much further effort.

   The following diagram shows a generalized phase sequence of an attack, and if after information enumeration, the attacker still fails to gain any access into the system, a patient and cautious hacker will choose to wait for new vulnerabilities to appear in the near future, or try to
hack other system or network devices, or to install sniffer to capture passwords of valid user accounts. Instead of spending more time and effort, an irritable and fearless attacker will turn to other tools to disable the target, for example, denial of service is a technique that is used often in this case.

2.1.2 Simple attacks, complex attacks and attack scenario

![Attack composition diagram]

We define simple attacks as those that carry out tasks in only one phase, e.g., Twwwscan probing is a Phase 1 simple attack. So does Nmap. Simple buffer overflow to gain user level access to a system is a Phase 2 simple attack.

Any attack covering tasks in more than one phase falls into the complex attack category. Complex attacks are composed of multiple related attacks (simple or complex) or attack steps against the same target computer, e.g., 0x333samba is a complex attack as it involves Phase 1, Phase 2 and Phase 4.

An attack scenario consists of a group of related attacks (simple or complex) against different targets in one network. Figure 2.1.2 gives an overall picture of an attack scenario and
its compositions. It is obvious that the structure of complex attacks is crucial in order to understand the correlations among simple attacks.

Each simple attack consists of a sequence of one to multiple sessions. And each session consists of one to multiple packets.

### 2.2 Complex attack structure

Hacking tools are getting more and more sophisticated and most of them fall into the complex attack category. After a thorough investigation of hundreds of such tools (maybe we should describe all the sources of the tools and our coverage?), we summarized four types of complex attack structures as shown in Figure 2.1.3. In the figures attacks can be both simple and complex.

Those patterns are not only revealed in hacking tools, but also in hacking activities carried out by real hackers. They differ in the way that hacking tools are more automatic thus with short time intervals between consecutive steps. Manual attacks can be stealthier and more pertinent to the target system with intervention of human intelligence.

![Figure 2.1.3 Complex attack structures](image)

1. **Twwwscan** is a good example of structure 1. Actually most of the scanning tools against IIS server share the same structure as structure 1. Attacking codes are sent out to the target one by one according to the list of known vulnerabilities in IIS. All probing steps will result in a comprehensive report of existing vulnerabilities of the target system.

2. **Worm MBblast** burst out in August 2003 was spread out to the networks all over the world in a very short period of time. It attacks the DCOM RPC vulnerability of windows platform...
and propagates the code automatically. The attacking process of MBblast can be described as follows which is identical to Structure 2 in Figure 2.1.3

3. Another famous piece of malicious code Nimda worm that burst out in Sept of 2001 is a good example of Structure 3. It scans the target network and selects one of the following mechanisms to spread worm code according to the vulnerabilities found in the target system in the network:

- From client to client via email
- From client to client via open network shares
- From web server to client via browsing of compromised web sites
- From client to web server via active scanning for and exploitation of various Microsoft IIS 4.0 / 5.0 directory traversal vulnerabilities
- From client to web server via scanning for the back doors left behind by the "Code Red II", and "sadmind/IIS" worms

The worm modifies web documents (e.g., .htm, .html, and .asp files) and certain executable files found on the systems it infects, and creates numerous copies of itself under various file names. Its propagation process can be very well described using Structure 3 of Figure 2.1.3 as follows:

(A diagram to be inserted…)

4. Structure 4 in Figure 2.1.3 is mostly seen in adaptive hacking tools which adjust the code according to the environment to enhance the robustness of the hacking. In this case, terminating one session or even several sessions of an attack usually won’t stop it completely. Those hacking tools are coded to survive in heterogeneous and severe environments. As shown in Figure 2.1, attack 0x333Samba is a very good example of Structure 4. Its attacking process are given in Section 2.1

From the analysis given above, it is obviously that there are inherent correlations among attack steps, some are obvious and the others are not so obvious. The attack steps don’t come isolated with each other, and the result of one step could affect the coming ones. System Security managers always notice lots of probing before behaviors that are more intrusive. And the correlations among probes and subsequent attacks are obvious.
2.3 False positive rates and detection rates

Since detection mechanisms work with packets and sessions, there exists very direct and precise correspondence relationship between alerts and sessions. And from the analysis given in Section 2.1, it is very natural for one attack to have multiple sessions. Then what is the relationship between sessions and attacks? In prior work, the definition of false positive and detection are seldom given formally. For example, in the lab work carried out at MIT Lincoln Lab, their calculation implied that:

Detection rate $= \frac{(\text{number of correctly detected sessions})}{(\text{number of attacks invoked})}$

Unless we define detecting an attack is just same as detecting at least one session invoked by this attack regardless of how many sessions it actually invoked, the above equation does not make any sense. Besides, redundancies have to be eliminated beforehand to deal with the case that multiple sessions being alerted in one attack.

To clarify the relationship between sessions and attacks, we define “Hacking sessions” as the set of all sessions generated during the attack processes, while “Normal sessions” are those not generated by hacking activities. Note that sometimes hackers do use normal commands and if the actions do not belong to any attack phases, then those sessions are regarded as normal sessions as well. In our experimental design to generate attack traces given in Section 4, 2015 attacks are invoked and 19627 hacking sessions are generated. Meanwhile the background traffic generated 468093 normal sessions.

**Definition 1 False Alerted Session Rate:** Portion of false-alerted (misclassified) normal session.

$P(\text{Alert} | \text{Normal-session}) = \frac{(\text{number of false alerted sessions})}{(\text{number of normal sessions})}$

**Definition 2 False Positive Rate:** Portion of alerts that do not reflect true attack sessions.

$P(\text{False-alert} | \text{Alert}) = \frac{(\text{number of false alerts})}{(\text{number of alerts})}$

**Definition 3 True Positive Rate:** Portion of alerts that correspond to true attack sessions. We use $P(\text{True-alert} | \text{Alert})$ to denote it and:

$P(\text{True-alert} | \text{Alert}) = \frac{(\text{number of true alerts})}{(\text{number of alerts})} = 1 - P(\text{False-alert} | \text{Alert})$

**Definition 4 Session Detection Rate:** Portion of hacking sessions detected by IDS. In this article we sometimes use $P(\text{Alert} | \text{True-Attack-session})$ to denote detection rate. Note that session detection rate is defined based on sessions.
\[ P(\text{Alert} \mid \text{True-Attack-session}) = \frac{\text{number of true alerted sessions}}{\text{number of hacking sessions}} \]

**Definition 5** Simple Attack Detection Rate: Portion of simple attack detected by IDS. This detection rate is defined based on simple attack.

\[ P(\text{Alert} \mid \text{Simple-attack}) = \frac{\text{number of alerted simple attacks}}{\text{total number of simple attacks}} \]

Measurement of effectiveness is closely related to the purpose of detection.

If a system administrator utilizes ID systems for incident report only, then he cares most about simple attacks, thus Simple Attack Detection Rate will be chosen as the measurement to avoid multiple alerts for the same attack, sometimes redundancies can be reduced by applying alert aggregation algorithms. On the other hand, if detecting is for the purpose of automated prevention of such activities, then the system administrator might want to know all possible malicious sessions in one attack. Because terminating one session usually doesn’t stop the attack completely, we might want to stop as many sessions as possible. In this case, they will prefer Session Detection Rate as oppose to Simple Attack Detection Rate to measure the effectiveness of their detection boxes.

### 2.4 Hacking personalities and hacking behaviors

One attack consists of a series of actions. Usually an attack probes the network first to find its topology, active hosts and ports, services and vulnerabilities. Then they may choose the most vulnerable host in the network to start with. They usually carry out the hacking step by step but the way they hack differs more or less depending on their personality. So we are interested in the group of people and it is also important to know how the way they think influences the hacking process that they launch.

It is not easy to define the term “hacker.” On the web the most complete materials about hackers is *The New Hacker's Dictionary* [39], which covers almost everything about them from hacker slang, habits, folklore, writing styles, hacker humor, to hacker personalities. According to it, a hacker can be defined as:

- A person who enjoys exploring the details of programmable systems and how to stretch their capabilities, as opposed to most users, who prefer to learn only the minimum necessary.
- One who programs enthusiasm (even obsessively) or who enjoys programming rather than just theorizing about programming.
A person capable of appreciating hack value
A person who is good at programming quickly
An expert at a particular program, or one who frequently does work using it or on it
One who enjoys the intellectual challenge of creatively overcoming or circumventing known limitations
A malicious meddler who tries to discover sensitive information by poking around

And that is not all. There are definitions vary from very positive (intellectual, creative, constructive, etc.) to very negative (malicious, criminal, etc.). While there are good and bad in any group of people, the hacker group is clearly a unique one. They grew up in modern IT environment, which shaped their perspectives of the world, and meanwhile how this small group people think have disproportional influences over the trend of modern technology development.

Hacker community is also a very organized group of people. They communicate through Internet applications and services, such as email, IRC-chat, Usenet news, etc, and they gather regularly at specialized conferences to exchanges knowledge and experiences. Among those conferences, the HOPE and Dec Con are probably the most famous.

The most interesting and relevant part is their personalities revealed during the process of hacking, which we believe contains traces of the patterns of their hacking behavior including the hacking steps they take, hacking tools they usually choose or how long they usually wait and much more. Those characteristics are directly related to what we are simulating.

Unfortunately these aspects of hacking are not well documented at all. We have to turn to the empirical way to classify them and figure out the percentages of each group. Several system security managers with intensive experiences with hacker preventions (most of them work for Internet Service Providers) are invited and a good discussion on hacker personalities enable us to roughly categorize hacker group into four classes. This classification is very empirical but we found this a very interesting and important topic that deserves more effort.

The four classes are:

1. **Irritable**: Those attackers usually try to hack obvious vulnerabilities first, if without any success, they will then choose to DOS or DDOS the target.

2. **Patient**: If they could not find any obvious way, they will do things like turn to other more vulnerable machine close to the target, and install sniffers there to listen to the target. They can hack the target continuously for a long time, or even wait for a new security hole to emerge.
3. **Fearless**: Those are usually not very experienced hackers. They adopt all scanning or hacking tools they can find and try them over the network without any hesitation. Most of their behavior can be caught and recorded by intrusion detection systems.

4. **Scrupulous**: They usually don’t even scan networks or servers. They are experienced enough to collect information manually, although slowly but very safe. They also usually hack in stealthy way, hide themselves from some “springboard” and could destroy all evidence once they have gotten into the system.

Empirically, the percentages of the four groups are 45%, 15%, 30% and 10% respectively.
3 Our Approach

It is always a challenging issue to simulate something about which we don’t have a comprehensive understanding. As it comes to attack traces, it is going to be very complicated to simulate novel and unknown attacks because it is hard to model them, neither can we identify their hacking mechanisms and make them to recur in a particular environment. In this section we propose a method to generate synthesized traffic data with two components, background traffic and attack traffic. The background traffic should be clean and normal traffic generated by normal users and applications in the network environment and the attack traffic contains all hacking activities in the time period of simulation. All hacking activities are planned and scheduled in attack scenario design. However, due to the above reasons, unknown attacks are not considered in this approach.

A comprehensive and rigorous approach of generating realistic attack traces to simulate certain network environment should include tasks shown in the following figure:
Figure 3.1 the tasks in the process of attack trace generation

1. The selection or definition of appropriate attack taxonomy to be used in the experimental design;

2. Investigation of the network that would be simulated and real traffic data collection for the following purposes:
   a) To Model background traffic
   b) To investigate both time and frequency distributions of different attack categories (defined by the taxonomy in use) in the real traffic

3. Experimental network setup. An experimental environment design should at least include a typical net structure similar to the real network to simulate, target machines with the platforms used in the real environment to simulate, and traffic recorder mechanism that’s basically capable to capture all passing by packets under the network traffic load to be generated in the experiment. And a tool to replay the traffic later should be ready for the convenience of other researchers.

4. The design and implementation of the scheme to generate background traffic that conforms to the patterns established through modeling real data;

5. The test of background generator and the validation of data generated in the test;

6. Attack selection. Establish certain criteria to select a combination of representative attacks.

7. Attack scenario design. Scenario design is important to ensure that the attacking steps keep the inherent correlations as pointed out in the previous analysis in attack phases and attack structure. At least the following issues should be considered in this stage:
   a) Time and frequency distributions of each attack category;
   b) Inherent correlations among attacking steps;

8. Attack Insertion design
   a) For network based behavior: Some of the attacking tools can be invoked automatically but others need human intervention, e.g., the tools with GUI interface.
   b) For host-based behavior: As shown in the discussion in previous sections, there
are lots of tasks that an attacker usually does after grabbing initial access to the target system. Most of the activities on hosts like this require human intervention and cannot be carried out automatically unless there is a way to model or describe the onsite hacking behaviors.

9. Attack traffic generator design and implementation to generate attack traffic in which numbers of different attack instances conform to the overall attack scenario designed in previous stage.

10. Test and validation of attack traffic generated.

11. Synthesis of background traffic and attack traffic into realistic traffic datasets. There are basically two different methods to combine two sources of traffics, offline or online. An offline method is to generate the traffics respectively (but should be within the same period of time and in the same experimental network setup) and then merge the two datasets offline according to the timestamps of the packets. An online method involves setting up the experimental environment and activating both background traffic generator and attack traffic generator together with manual attack insertion processes at the same time to inject packets into the links.

Additionally, to provide datasets for researches in alert analysis, e.g., alert aggregations and alert correlations, alert data needs to be generated also, which involves the following steps:

12. Selection and installation of ID systems in the experimental environment to generate the alerts. A comprehensive experiment for alert generation should include both network based and host based systems with different detection mechanisms, i.e., signature based or anomaly based.

4 Experimental design to generate attack traces

In this section, we describe our experimental design to generate realistic attack traces, which serves as an illustration to the generalized approach presented in Section 3. A detailed description about our experimental network setup is given in this section and we will focus on the generation of attack traffic (Step 6 – Step 9 in Section 3), which mainly include the criteria to select a representative combination of attacks, the details of attack scenario design preserving the inherent correlations among hacking steps, and the attack scheduling to insert attack traffic automatically and manually.

In step 11 of the approach, two components are synthesized through inserting attack traffic designed in our experiment into the background traffic in MIT simulations. Although we did not
investigate the validity of MIT background traffic, our experimental design illustrates the feasibility of our approach and makes the attack traces more realistic through improvements on the following aspects:

1. To select a combination of attacks with representative hacking skills that are frequently utilized by attackers;
2. To ensure that the attack scenarios designed in our work cover different complex attack structures;
3. To ensure that the attack scenarios include attack steps with reasonable internal correlations;
4. To design realistic attack scenarios with the consideration of diversified hacking behaviors revealed by attackers with different personalities;
5. To control the distributions of different attack categories so that they conform to the real distributions on Internet, and
6. To generate enough attack instances.

By interlacing attack sessions into the background traffic, we create a new simulation and record it in a re-playable way.

4.1 Attack Selection

To reflect real world scenarios as much as possible is one of the objectives of our experiment. We apply the following two criteria to select a combination of attacks:

- It is popular in real network. Its popularity is indicated by the data provided by Security Focus (top 14 attack categories in January to June of 2002);
- The attacking technique(s) employed in the attack is typical.
- It was invented after 1999, since new hacking techniques were developed quickly as time goes by. Most attacks will be prevalent for just certain amount of time.

Typical and popular attack techniques emerged after year of 1999 mainly include the following:

1. To sniff networks to gain passwords and other sensitive information;
2. To attack vulnerabilities of scripts like asp, php and perl, etc.;
3. To utilize shared network services of Windows platform;
4. To embed malicious code into web pages or mail attachments;
5. Session hijacking including fake IP, DNS or ARP, etc.;
6. Worms propagating through attacking network or system vulnerabilities;
7. Distributed Denial of Services;
8. Other attacks aiming at application level bug or vulnerabilities;

To illustrate what is going on in real Internet, we cite numbers from Security Focus [34], which is a web site created to facilitate security discussions and to promote security awareness. According to its published statistical result, security events happened during the first half of 2002 can be categorized as follows and the top 14 attack categories include:

<table>
<thead>
<tr>
<th>No</th>
<th>Category</th>
<th># Events</th>
<th>Specific Attack Selected in Our Experiment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Probe</td>
<td>287,199,691</td>
<td>X-Port, X-Scan, Nmap, CIS, SSS, SDS, Twwwscan, Scanlo0k</td>
</tr>
<tr>
<td>3</td>
<td>DOS</td>
<td>21,599,725</td>
<td>UDPFlood, UPNP dos</td>
</tr>
<tr>
<td>4</td>
<td>FTP Attack</td>
<td>5,677,588</td>
<td>Wuftpd2600</td>
</tr>
<tr>
<td>5</td>
<td>Remote Service Attack</td>
<td>2,894,429</td>
<td>BSDTelenetd, SQL2kUDP, SqlExec, SendMail2</td>
</tr>
<tr>
<td>6</td>
<td>Manipulation/Spoofing</td>
<td>1,890,230</td>
<td>FakePing, DNSfake</td>
</tr>
<tr>
<td>7</td>
<td>Windows Attack</td>
<td>943,365</td>
<td>CrackPasswd, GetAccount, SMBCrack, VMICracker</td>
</tr>
<tr>
<td>8</td>
<td>Malicious Code</td>
<td>643,663</td>
<td>T-Cmd, NtRootKit, BingHe, MIME, AckCmd</td>
</tr>
<tr>
<td>9</td>
<td>SMB/NetBIOS Attack</td>
<td>473,754</td>
<td>SMBdie, IpHacker</td>
</tr>
<tr>
<td>10</td>
<td>SMTP</td>
<td>350,172</td>
<td>IIS-smtp</td>
</tr>
<tr>
<td>11</td>
<td>Infrastructure Attack</td>
<td>193,650</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>DNS Attack</td>
<td>6,819</td>
<td>DNSfake</td>
</tr>
<tr>
<td>13</td>
<td>RPC Attack</td>
<td>4,186</td>
<td>RPC Nuke, Re</td>
</tr>
<tr>
<td>14</td>
<td>NFS Attack</td>
<td>2,855</td>
<td>Pqwak, Shed</td>
</tr>
</tbody>
</table>

*Figure 5.2 Top 14 Attack Categories and Attacks Selected to Extend MIT 1999 Database*

After a thorough investigation into the attack databases, we found that, in the top 14 categories, there are 46 attacks that were prevalent of the time being and were implemented with typical attacking techniques.

All the attack scripts are collected and ready to be triggered by the program “attack scheduler,” which invoke the scripts with appropriate parameters at appropriate times.

### 4.2 Attack Scenario Design

In real Internet, it is usually the case that various types of hackers from different locations are trying to get into one target. So we need to design the entire scenario to reflect the real world.
From the previous analysis on hacking personalities, we know empirically that the percentages of the four groups (irritable, patient, fearless and scrupulous) are 45%, 15%, 30% and 10% respectively.

By assuming there are around 40 hackers working on the protected network simultaneously, if we take the empirical distribution, we have 18 irritable, 6 patent, 12 fearless and 4 scrupulous. We then prepare a hacking timetable (or hacking profile) for each of them to describe all of their actions during the 5 days. One patient hacking profile looks like the following:

<table>
<thead>
<tr>
<th>Hacker No.</th>
<th>Date</th>
<th>Time</th>
<th>Action</th>
<th>Source IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>July 17th</td>
<td>08:01</td>
<td>Ping 172.16.12.5</td>
<td>192.168.1.8</td>
</tr>
<tr>
<td>23</td>
<td>July 17th</td>
<td>08:02</td>
<td>Telnet 172.16.12.5 80</td>
<td>192.168.1.8</td>
</tr>
<tr>
<td>23</td>
<td>July 17th</td>
<td>11:00</td>
<td>Re-h 172.16.12.5</td>
<td>192.168.1.8</td>
</tr>
<tr>
<td>23</td>
<td>July 17th</td>
<td>18:00</td>
<td>Net View \172.16.12.5</td>
<td>192.168.1.8</td>
</tr>
<tr>
<td>23</td>
<td>July 18th</td>
<td>10:00</td>
<td>Nbtstat -A 172.16.12.5</td>
<td>192.168.1.8</td>
</tr>
<tr>
<td>23</td>
<td>July 18th</td>
<td>12:00</td>
<td>Unicode 172.16.12.5</td>
<td>192.168.1.8</td>
</tr>
<tr>
<td>23</td>
<td>July 19th</td>
<td>11:00</td>
<td>IisAttackTools 172.16.12.5 0 -p</td>
<td>192.168.1.8</td>
</tr>
<tr>
<td>23</td>
<td>July 19th</td>
<td>11:01</td>
<td>Telnet 172.16.12.5 7788</td>
<td>192.168.1.8</td>
</tr>
<tr>
<td>23</td>
<td>July 21st</td>
<td>11:00</td>
<td>WebDavX3 172.16.12.5</td>
<td>192.168.1.8</td>
</tr>
<tr>
<td>23</td>
<td>July 21st</td>
<td>11:01</td>
<td>Telnet 172.16.12.5 7788</td>
<td>192.168.1.8</td>
</tr>
</tbody>
</table>

![Figure 5.2.2: Examples of a hacking profile](image)

Thus the big scenario consists of 40 prescheduled hackers.

### 4.3 Attacking Scheduling

Both automated insertion and manual insertion will be used during the simulation.

Forty small detailed profiles were derived from the scenario design. They are combined into one big table. We then sort the table by time and make it the input of the attack scheduler. The attack scheduler will switch IP address to the “Source IP” and trigger the “Action” at “Time.” This is [an] automatic insertion.

To add some randomness into the simulation, we also scheduled some hacking events to carry out manually. One machine on the outside subnet is devoted to a real person to do the hacking in real time fashion. A program residing on the machine records all their behavior for later analysis.
4.4 Experimental Environment

Our experimental network consists of inside subnet and outside subnet, which are connected together by a CISCO 1720 router. The physical topology of the experiment environment is shown in figure 5.3.2.2.

There are altogether nine servers in this experiment network. Two servers run attack scheduler program to simulate 40 attackers automatically. One server is provided for manual attack insertion. VMware is installed on some of the remaining servers to increase the number of target machines that we can simulate in the experiment.

The following table gives all the configuration and software installations of each server.

<table>
<thead>
<tr>
<th>Server No.</th>
<th>IP Address</th>
<th>Server Function</th>
<th>OS Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>172.16.12.10</td>
<td>Snort For Win(ACID) timer server</td>
<td>Win2K SP3</td>
</tr>
<tr>
<td>#2</td>
<td>172.16.12.20</td>
<td>Snare for Windows</td>
<td>Win2K Server</td>
</tr>
<tr>
<td>#3</td>
<td>172.16.12.66</td>
<td>Netpoke to replay the traffic</td>
<td>Redhat7.3</td>
</tr>
<tr>
<td>#4</td>
<td>172.16.12.40</td>
<td>Sniffer 4.7</td>
<td>Win2K SP3</td>
</tr>
<tr>
<td>#5</td>
<td>172.16.12.60</td>
<td>BRO</td>
<td>Redhat7.3</td>
</tr>
<tr>
<td>#6</td>
<td>172.16.12.80</td>
<td>Snare Core for Linux</td>
<td>Redhat7.3</td>
</tr>
<tr>
<td>#7</td>
<td>172.16.12.100</td>
<td>Tcpdump outside</td>
<td>Redhat7.3</td>
</tr>
<tr>
<td>#8</td>
<td>192.168.1.5–24</td>
<td>Attack Scheduler #1</td>
<td>Win2K SP3</td>
</tr>
<tr>
<td>#9</td>
<td>192.168.1.50–44</td>
<td>Attack Scheduler #2</td>
<td>Win2K SP3</td>
</tr>
<tr>
<td>#10</td>
<td>192.168.1.x</td>
<td>Manual attackers</td>
<td>Redhat7.3</td>
</tr>
<tr>
<td>#11</td>
<td>192.168.1.100</td>
<td>Tcpdump inside</td>
<td>Redhat7.3</td>
</tr>
</tbody>
</table>

After environment setup and some preliminary test, we started the experiment while replaying the second week traffic of [the] MIT 1999 datasets. Due to time limitation, the experiment was run only for 5 days, 22 hours a day. (Appendix I gives a flow chart for the experiment process each day.)

During the experiment, all sessions were recorded by TCPDUMP, altogether more than
half million sessions and the size of the file is around 1.5G. And the session file extracted from tcpdump output file has 1.2G content.

Alerts data were generated by SNORT, BRO, STAT and LERAD individually in their specific format. We wrote scripts to convert the outputs to readable formats. Their sizes add up to more than 200 M.

**Figure 5.3.2.2 Experimental environment**

### 4.5 Future Experiments

We regard this experiment as the preliminary; it validates our idea by creating a simulation with enough attack instances on the base of previous research done by MIT group. It is a good step towards better work. And the experiences that we gained through this experiment are valuable for future work. In future experiments, we will improve from the following aspects:

1. We need to locate other host based intrusion detection tools more appropriate to our work, especially anomaly-based tools. In this experiment, host based anomaly alert data is missing.

2. Because a big percentage of the hacking activities was automatically invoked by scheduling program, once the user or root privileges of the targeted hosts are compromised, the program immediately quits to return with a successful status. However, a real hacker usually performs a lot after getting into the system and this part supposedly should be monitored by host based IDS. Our auto program just did not perform enough to be possibly noticed by IDS.
3. Both DOS and DDOS attacks could generated two many packets for our program to record. Our experiment has to be redone for two days because of it. We had to run DOS and DDOS in a constrained manner afterwards. We have no such attack instances in the 1998 and 1999 MIT data. However, MIT 2000 dataset is more or less devoted to DDOS attacks, and each of the attacks runs only for a very short period of time, say, several minutes. A special isolated environment could be better in this case. Plus, the scenario of DDOS attacks deserved to be specially designed.

4. During this experiment, there are times that when an attack slows down target host dramatically or crashes the target host, it follows that the host based IDS stopped to alert normally. We lose some of those data that could be valuable input to our analysis.

5. After developing the decision rules from current dataset, we need another dataset to try them on. This can be accomplished by updating the attack scenario in our design and regenerate another sets of alerts.

The above are the aspects that need to be taken care of in the future experiments.

5 Evaluations of the datasets

6 Conclusions and Future work

7 References


Lafayette, IN: Purdue University, Computer Sciences Department, 

http://www.ce.chalmers.se/staff/jonsson/publ97-.html, 

[19] Online 1999 MIT Lincoln Lab Intrusion Detection Attacks Database: 
http://www.ll.mit.edu/IST/ideval.docs/1999/attackDB.html


[24] Stuart McClure, Joel Scambray and George Kurtz, Hacking exposed, third edition, 
Osborne/McGraw Hill


