Using a VMware Network Infrastructure to Collect Traffic Traces for Intrusion Detection Evaluation

Frédéric Massicotte
Mathieu Couture
Annie De Montigny-Leboeuf
Communications Research Centre
3701 Carling Avenue
P.O. Box 11490, Stn. H
Ottawa, ON K2H 8S2
Canada
http://www.crc.ca

E-mail: {frederic.massicotte,mathieu.couture,annie.demontigny}@crc.ca

1 Introduction

Since the DARPA Intrusion Detection Evaluation Data Set [2] was made available in 1998, and then updated in 1999 and 2000, it seems that no other significant freely available data sets have been provided to allow benchmarking of Intrusion Detection Systems (IDS). Even if those traffic traces are still used by the security research community, they have not been updated since. The absence of additional data is mainly due to the cumbersomeness of the task.

This lack of data was mentioned in a NIST Interagency Report published in 2003 [3], which raised the fact that more data sets are needed to test and evaluate Intrusion Detection Systems. In the conclusion of this report, some recommendations for IDS Testing Research are made. Among those recommendations, the authors insist that data sets should contain realistic data and be shared freely between multiple organizations. They also state that *there is a great need to provide the security community with a large set of attack traces. Such information could be easily added to and would greatly augment existing vulnerability databases. The resulting vulnerability/attack trace databases would aid IDS testing researchers and would provide valuable data for IDS developers.*

To address those issues and facilitate certain aspects of this task, we developed a strategy to rapidly generate and collect a large number of attack traffic traces for intrusion detection system testing and evaluation. To develop such a large scale data set, a controlled network infrastructure had to be developed. This infrastructure had to allow:

1. Recording of all network traffic,
2. Network traffic noise control,
3. Control of attack propagation,
4. Usage of real and heterogeneous system configurations,
5. Fast recovery to initial conditions.

To meet those requirements we developed a controlled virtual network using VMware Workstation 5.0 [6]. VMware provides a virtual broadcasting network environment allowing traffic capture within a single traffic trace of all communications generated by the attack. These traffic traces can then be used for the study of attack behaviors (req. 1). This network also allows us to control the network traffic to create clean traces that only contain network traffic relevant to the attack scenarios (req. 2). Since the test environment is virtual, the attack propagation is confined to this environment, thus preventing infection of the physical machines running the virtual network (req. 3). VMware facilitates the creation of template virtual machines having different software configurations (operating system, services, etc). So far, we have successfully installed over 200 operating system versions. This allows us to create a database of virtual machine templates that can be used to rapidly deploy

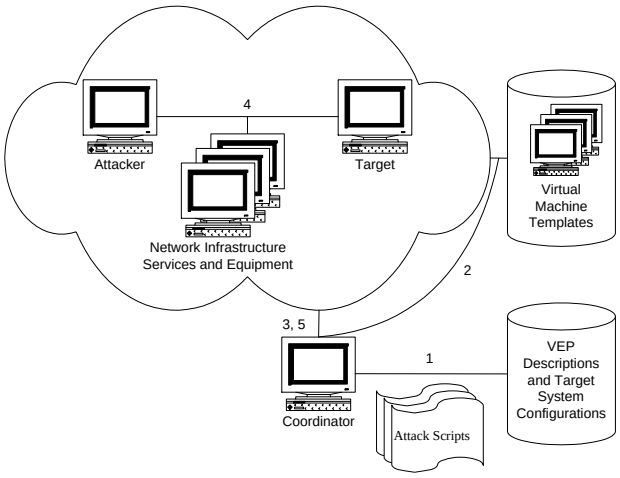

Figure 1. Virtual Attack Network

custom network configurations within a single physical machine (req. 4). Also, the VMware snapshot functionality allows us to backtrack the test network to the state at which it was before each attack attempt. All attack scenarios can then be performed under the same initial conditions (req. 5).

2 Virtual Network Infrastructure Overview

This virtual network contains attack systems, target systems and network infrastructure services and equipment. The attack systems are used to launch attacks against the target systems by using vulnerability exploitation programs. The attack systems are also used to capture the packets that are generated by the execution of the vulnerability exploitation programs. The network infrastructure services and equipment are there to ensure the network communications needed while the attack is in progress.

3 Collecting Process

The virtual network we used to collect our traffic traces is shown in Figure 1. Each step of our collection process is indicated on the link between the involved actors.

1. **Script Generation.** The process of choosing which vulnerability exploitation program should be run against a given target system and how it will be configured is automated. For this data set, we decided to run every vulnerability exploitation program against every target system offering a service on the same port as the program targeted service. To automate this, we developed a database containing the complete system configuration for each target template, as well as the ports targeted by the vulnerability exploitation programs (VEP) we downloaded.

2. **Virtual network setup.** A different virtual network is built for every script. Each contains the target virtual machine template, the attacking virtual machine, and some other machines offering the network services needed for the attack to be successful such as a DNS or a remote mail server.

3. **Current attack script setup.** Once the virtual attack network is ready, the coordinator provides the attacking virtual machine with the proper attack configuration. To communicate with the attacking virtual machine, the coordinator (which is the only physical machine) uses a hard drive shared via VMware. This shared drive is the only way the coordinator can communicate with the virtual network. This enables us to isolate the virtual network (from a networking point of view) while keeping some communication capabilities.

4. **Attack execution.** While the traffic being generated is recorded, the attack machine performs the attack.

5. **Tear down.** This step includes saving the attack traces (vulnerability exploitation program outputs and the recorded traffic) on the same shared drive used in step 3. Then, the coordinator stores these attack traces into the data set and reverts the attacker and target virtual machines to their initial state to avoid the vulnerability exploitation program side effects on the next attack executions.

Steps 2 to 5 have to be repeated for every remaining attack script.

4 Labeling and Documentation

For our intrusion detection data set, each traffic trace is labeled by four characteristics: the target system configuration, the vulnerability exploitation program configuration,

```
System Configuration
  IP: 10.92.39.14
  Name:  VMWin2000ServerTarget
  Operating System:  Windows 2000 Server
  Ports:
    21/tcp Microsoft IIS FTP Server 5.0
    25/tcp Microsoft IIS SMTP Service 5.0
    80/tcp Microsoft IIS Web Server 5.0
Vulnerability Exploitation Program Configuration
  name:  jill.c
  reference:  Bugtraq,2674
  command:  jill 10.92.39.14 80 10.92.39.3 30
Vulnerable:  yes
Success:  yes
```

Figure 2. Traffic trace label example

Operating System Family	Operating System Versions	Scenario instances	Vulnerable/ Not Vulnerable	Success/ failure/ unclassified
FreeBSD	7	270	73/197	4/27/239
Linux	6	436	79/357	10/77/349
Windows	13	1637	948/689	166/729/740

Table 1. Intrusion Detection Data Set Summary

whether or not the target system has the vulnerability exploited by that program and the success of the attack. Figure 2 shows an example of a label associated to a particular traffic trace of our data set.

The target system configuration description includes the list of the installed software (e.g. the operating system, the different daemons and their versions), as well as its IP configuration. The vulnerability exploitation program configuration defines the options used to launch the attack. The vulnerability of the target system is decided on the basis of its configuration, the vulnerability exploitation program being used in this attack and the vulnerability information available in the Bugtraq database [4]. The success of the attack is the hardest information to acquire. Since, as seen in the previous section, the execution of the vulnerability exploitation programs is automated, a postmortem analysis to determine whether they have been successful or not has to be performed. Traffic traces are labeled according to three categories: one for those that succeed in exploiting the vulnerability, one for those that fail, and one for those for which we were not able to determine whether they were successful. This classification is automatically made using human specified criteria by looking at the programs output (hacker point of view) and at the effect on the targeted system (victim point of view).

5 Intrusion Detection Data Set Summary

The intrusion detection data set contains 2343 traffic traces each of which contains a different attack scenario. These scenarios were conducted in less than 24 hours against 26 different operating system configurations using 92 vulnerability exploitation programs. The vulnerability exploitation programs are mainly taken from the ones available on the SecurityFocus web site [4]. They were launched against vulnerable as well as non-vulnerable systems. Table 1 provides a summary of the intrusion detection data set generated using our virtual network.

Unfortunately, it often happens that the vulnerability exploitation program outputs are not sufficient to decide whether the attack succeeded or not. In this case, we believe that improvement of these outputs will reduce the number of unclassified attack scenarios. Efforts are currently being made in this direction.

6 Conclusion and Future Work

This experience has shown that a coordinated virtual network infrastructure can increase the efficiency and flexibility of labeled traffic traces generation for purposes such as intrusion detection evaluation and research in traffic analysis.

Another aspect we want to explore with this virtual network is the study of worm viruses behavior and propagation as well as distributed denial of service on large networks. However, these studies require large network environments and our virtual network is limited in size by the resources available on the physical machines. To address this issue, we plan to develop a distributed version that will allow several physical machines to share the same virtual network.

Even restricted to a single physical machine, our approach has proven to be useful in other situations requiring automatic labeling and collection of large amounts of traffic traces involving a wide scope of system configurations. In particular, a previous version of our virtual network has also been used to collect data for passive operating system fingerprinting [1]. The current version has also been used to collect traffic traces for automatic passive network discovery. We also plan to use it to study operating system packet reassembly behavior (Fragroute [5]) as well as to evaluate the accuracy of packet reassembly algorithms in IDS.

In the near future, if the security community expresses an interest for those attack traffic traces, we plan to make them available together with their documentation on the Internet. The security community could then take advantage of a recent common reference to evaluate intrusion detection systems.

References

[1] A. DeMontigny-Leboeuf. A multi-packet signature approach to passive operating system detection. CRC/DRDC joint Technical Report CRC-TN-2005-001 / DRDC-Ottawa-TM-2005-018, December 2004.

[2] Lincoln Laboratory Massachusetts Institute of Technology. Darpa intrusion detection evaluation. `http://www.ll.mit.edu/IST/ideval/data/data_index.html`.

[3] P. Mell, V. Hu, R. Lipmann, J. Haines, and M. Zissman. An overview of issues in testing intrusion detection systems. Technical Report NIST IR 7007, National Institute of Standard and Technology.

[4] SecurityFocus. SecurityFocus Homepage. `http://www.securityfocus.org/`, September 2005.

[5] D. Song. Fragroute 1.2. `http://www.monkey.org/~dugsong/fragroute/`.

[6] VMWare Inc. Vmware. `http://www.vmware.com`, September 2005.