Project 3: Network Scan Detection

Due 11:59pm Sunday, May 6th. (~18 days)

Introduction

For this project you will be creating a network intrusion detection system (IDS). Your program should be capable of detecting weather or not a particular system has been scanned with a network scanning utility such as NMAP or Nessus. The essence of the project is to parse network logs, and look for signs of a network scan. You will need to generate your own logs using the **tcpdump** command in Linux. (100 points)

Setup

For this project you will need four VMs: Download your VMs from (**https://goo.gl/hXxMA6**)
If your personal laptop/desktop cannot handle this, I have a computer configured in my Office. You can use it for log collection. But I would recommend trying to setup your own environment for this if possible.

1. Attack VM (Debian with Metasploit framework). From this VM you will launch scans and exploits.
2. Monitoring VM (Debian with Open vSwitch). This VM acts as a network switch and monitoring point at the same time. Open vSwitch software installed in it will connect other VMs you have into one network. You will observe network traffic with tcpdump on this VM.
   NOTE: **OpenvSwitch is already installed. You don't have to do anything**.
3. Windows XP VM (victim 1)
4. Metasploitable VM (victim 2)

VMs should be connected to each according to layout featured in Figure 1.

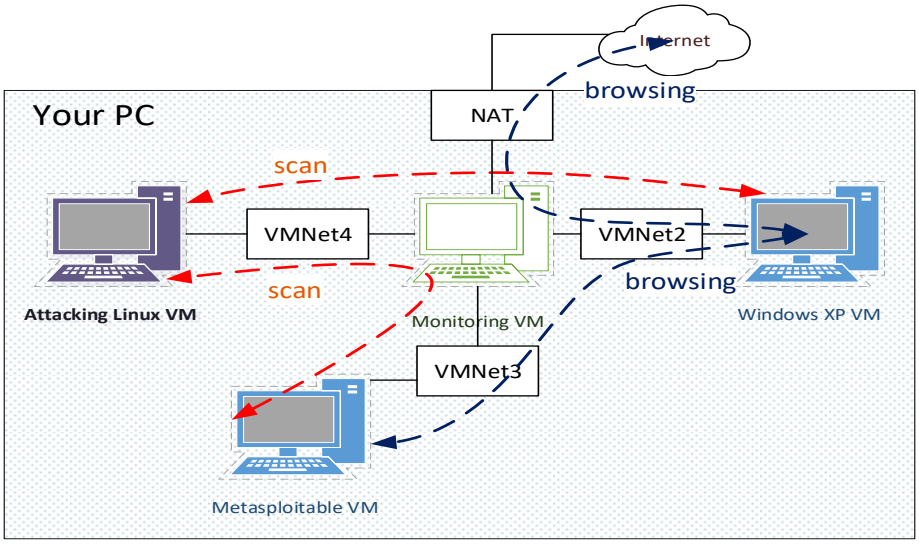


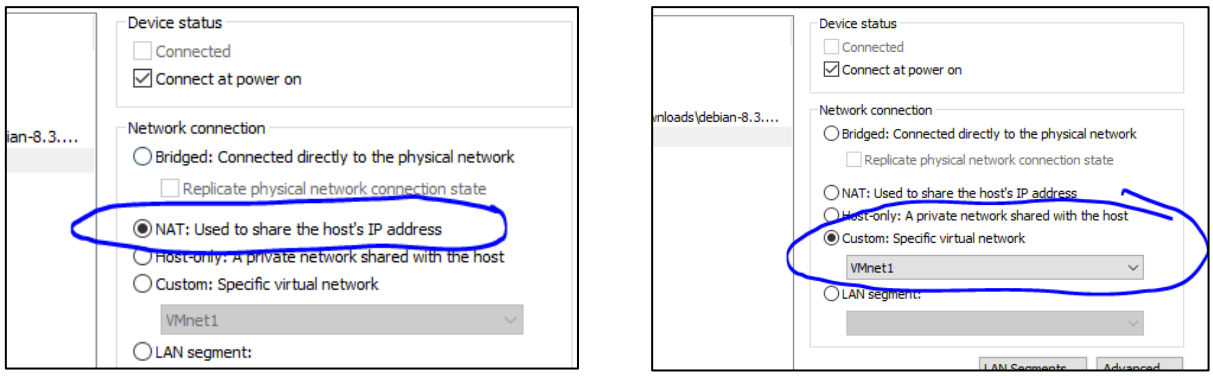
Figure 1. Virtual Machine Connection Diagram

One interface on Monitoring VM should be connected to the NAT option (see Figure 2a). For the rest of the connection choose new VMNet# for each pair of connections (see Figure 2a, "Custom: Specific Virtual Network" option in VMware Workstation).

Make sure all VMs can ping each other and 8.8.8.8 when Monitoring VM is running. VMs should not be able to ping each other when Monitoring VM is not running. Make sure your virtual networking is working properly as soon as possible.

NOTE: Do not start your virtual machines without making the Network connections as shown. If the connections are not done properly, then there are chances of starting a local broadcast storm and you might slow down your computer.

For Mac users, there is Help online on how to create custom network adapters. If you don't get it, TA can help you.



(a) (b)

Figure 2. Selecting VM virtual network connection.

Capturing trace files

1. On the Monitoring Linux machine start network traffic capture and write it to a file with the command

tcpdump –i any –Q in > tcpdump_no_scan_1.log

you can find more command options by typing *man tcpdump*.

NOTE: If you decide to use other tcpdump options, please document it in the readme file.

2. On a attack machine (scanner), launch various network scans. This can be general host enumerations scans or more targeted scans against another Linux host or windows VM.

3. After several monitoring and scanning sessions you should have a number of network capture files

tcpdump0.log, tcpdump_no_scan1.log, tcpdump_junk.log …

3. Your captures should include at least 1) nmap –sS; 2) nmap –F; 3) nmap –sV in separate files.

Program

Your goal is to write a program in Python able to detect if there is some scanning activity on your network by sequentially reading all capture log files in current directory and analyzing them.

Minimal output should print file names and the instances of the scans if there is any. Try to keep output rate low. One line per scan instance.

Something like:

tcpdump0.log -->

scanned from 192.168.100.15 at 01:49:07

tcpdump1.log -->

scanned from 192.168.100.17 at 01:50:23

scanned from 192.168.100.16 at 02:40:03

…

Keep in mind that there might be huge amount of normal Internet browsing traffic with the scanning traffic mixed in. This can be simulated by opening Windows XP and browsing the Internet with ancient Internet Explorer (google and some other web sites don't work with IE. Start browsing session by visiting bing.com). You can also browse into metasploitable.

Program specifications:

1. Source code with comments has to be named scanproject.py (NOT myproject.py, prog1.py or anything else)

2. The program should not ask TA any questions. It should read all log files in the current directory and produce the report file within the same directory. *Note: Don't hardcode the names of the files into your program. I may run it against 10 log files or 100 log files. Try to find all log files in current directory and process them all.*

3. Your program should run with no errors on python 2.7. (if it doesn't run you lose most of the grade)

4. It should not take more than 1 minute to produce the result for log files of at least 10MB in size.

5. Test your code against different scenarios like network scans, single machine scan, with or without internet traffic scan, etc. Your code will be tested against different test cases. The more test cases you cover the better the performance will be for your code.

Note: There are grades for documentation. Please make sure you submit a proper README file with the project. Your documentation should include detailed analysis performed by you.

Submission

1. Submit your code in the Folder Scanning_Detection

a. Source code named scanproject.py (NOT myproject.py, prog1.py or anything else)

b. All your reasonably sized capture log files in the same directory as your code.

c. Short explanation which file contains what type of scan in readme.txt

2. Push your commit to GitHub Repository before deadline. Any submission after deadline will not be downloaded.

3. Make sure everything you did is well documented.