Project 3: Network Scan Detection

Due 11:59pm Friday, December 16th. (14 days)

This project is take-home final exam.

You may get TA feedback and resubmit if I get you preliminary version by 5PM December 11th. It may take up to 2 days to give you the feedback.

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)

The project contains various additional tasks that you can implement for additional points which may help you to improve the overall class grade. The additional features should work in order to qualify for additional points. (120 points if all features attempted)

Setup

For this project you will need four VMs:

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.
3. Windows XP VM (victim 1)
4. Metasploitable VM (victim 2)

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

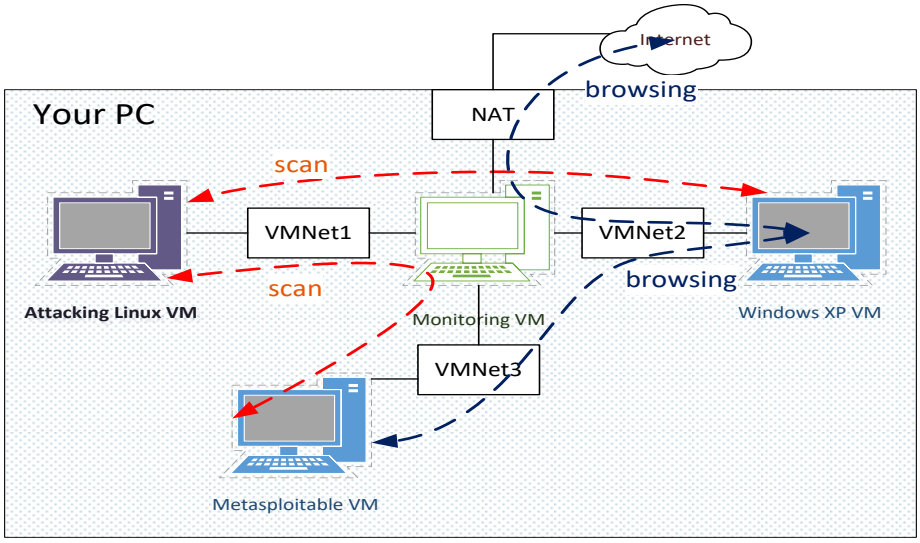


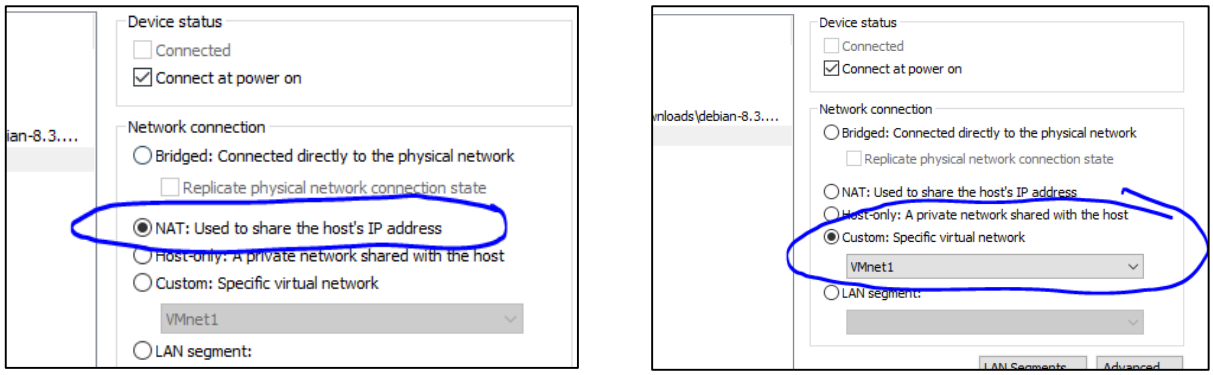
Figure 1. Virtual Machine Connection Diagram

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

tcpdump_junk.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.

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

BONUS POINTS IF

1. (+20) You can detect scanning in real time. Use pipelining with the program key (--online) to distinguish from offline mode. You should not implement it in the separate program. It should be the same *scanproject.py* accepting the feature key (--online). With this feature key present in command line your program shouldn't read log files, instead it should accept input from stdin. Below are three examples of reading from stdin.

a) Pipeline directly from tcpdump into your program:

```
tcpdump –Q in –i any | python scanproject.py --online
```

b) Save into log file and pipeline it at the same time:

```
tcpdump –Q in –i any –w tcpdump_qwe.log

tail –f tcpdump_qwe.log | python scanproject.py --online
```

This set of commands saves the output of tcpdump in the file and at the same time feeds it into your pogram in realtime.

c) Or from previously saved file:

```
cat tcpdump_qwe.log | python scanproject.py --online
```

2. (+20) You can differentiate between different types of scans (minimum 3). E.g.

tcpdump0.log -->

nmap –sV from 192.168.100.15 at 01:49:07

tcpdump1.log -->

nmap –sV from 192.168.100.15 at 01:49:07

nmap –sS from 192.168.100.16 at 02:40:03

3. (+20 for good technical research, +20 for each additional exploit reliably detected. No more than +80 total) You can identify some exploits against XP or metasploitable e.g.

tcpdump0.log -->

killbill exploit from 192.168.100.15 to 192.168.100.17 at 01:49:07

For this task you will need to do a research on how to detect the exploit. Let's say you plan to detect ftp smiley face exploit. How would you recognize the exploit in the network traffic? How would you separate it from other legitimate uses of ftp server? What if there are smiley faces in browser http traffic? Please, document the research you did in a document file and submit it with other files. As you can see the research part gives you partial credit. To get full points for exploit detection you will need to demonstrate that your program can reliably detect an exploit without triggering on legitimate use of the service.

Advanced features of the tcpdump you may consider using:

This command will capture IP packets in full

tcpdump -nnXs 0 –i any –Q in

-nn prevents name lookups

-X prints packets in Hex

-s 0 by default tcpdump captures beginning of the packet only. –s 0 will force it to capture full packets. -s 80 will force it capture first 80 bytes of each packet

Document all of your advanced features in readme.txt. Provide all the logs with scans capable of triggering your feature.

Submission

1. Compress into ZIP file the following:

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

b. All of your reasonably sized (fit into one e-mail in zipped format) capture log files.

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

2. e-mail to eece480f@gmail.com with the subject line "Your name Project 3"

Note: Since bonus tasks are more analysis type, make sure you document everything about the analysis you did and how you detected certain scan. Grades are assigned for documentation.