Marian Gawron (marian.gawron@hpi.de)
Martin Ussath (martin.ussath@hpi.de)

Exercise #2
Internet Security – Weaknesses and Targets
Prof. Dr. Ch. Meinel
October 28th, 2015 - Winter 2015/16

Topic: Reconnaissance – Port Scanning
Maximum points: 100%

This task covers network programming and reconnaissance! The goal is to write a simple TCP port scanning tool. You have two weeks to solve this task.

For implementation, we suggest to use one of the following packages:

Python libpcap (pylibpcap) - *http://pylibpcap.sourceforge.net/*
Java jNetPcap (jnetpcap) - *http://jnetpcap.com/*

We know, there are sophisticated packages for at least Python that allow convenient network packet creation, but the use of those (e.g., scapy) is not intended in this assignment.

Task Develop a simple TCP port scanner in one of the following languages: Python (recommended) or Java. We strongly recommend to solve this exercise with Python, because the Java library can cause some strange errors.

Choose two of the three mentioned scan techniques and implement these in your scanning application.

- Support TCP-Connect scans using simple sockets and standard network programming
- Support TCP-SYN scans using packet creation of SYN packets and capturing of response packets
- Support TCP-FIN scans using packet creation of FIN packets and capturing of response packets

Additional requirements:

- Take the target, the scanning method, and the ports as command line input and output open port numbers, one number per line.
- Make sure you are scanning the full range of possible ports!

Example
python scanner.py IP SCAN_METHOD PORTS
java -jar scanner.jar IP SCAN_METHOD PORTS

- IP is the IP address of the target
- SCAN_METHOD is the one of the specified scanning techniques (SYN, FIN or Connect)
- PORTS is a comma seperated list of ports(i.e. 1,80,8080,22)
  or a range(i.e. 20-100)
  or a combination of it(i.e. 21,22,100-200,8080)

BONUS If you are interested in this topic, we suggest you to go further and try to implement an operating system detection. This could be done with a look at TCP window sizes, sequence numbers, acknowledge numbers, etc. These characteristics might give hints about the operating system which allow an educated guess finally.

Hints To guide you, we prepared a short list of necessary steps:

1. Investigate the capabilities of the packet capturing/creation libraries!
2. Why do you need special libraries for TCP-SYN/ TCP-FIN scans?
3. Investigate the difference between TCP-SYN, TCP-FIN, and TCP-Connect scans! Design different algorithms to perform such scans!

Hand-In Requirements You will lose points if you don't follow the Hand-In requirements!

- Make sure the source code is of high quality, is *documented well* and *compiles* without warnings!
- Make sure your code is *UNIQUE*. We do not allow programming in groups or copying code from other sources. Everybody should design the scanner individually.
- Place a note about your development environment (operating system + version, compiler/interpreter + version) in the leading comments of your main source file.
- Provide a build file for ant if you use Java.
- Commit the source as a ZIP archive into the submission system
- Name the archive `firstname_lastname_portscan.zip`

Hand-in: November 11th, 2015, 12:00 p.m. (at noon) online at `fb10moodle.hpi.uni-potsdam.de`