Tutorial for Wireshark projects

Seonghyeon Moon

Wireshark ?

- **Wireshark** is a popular network analysis tool to capture network packets. It saves analysis using pcap extension.

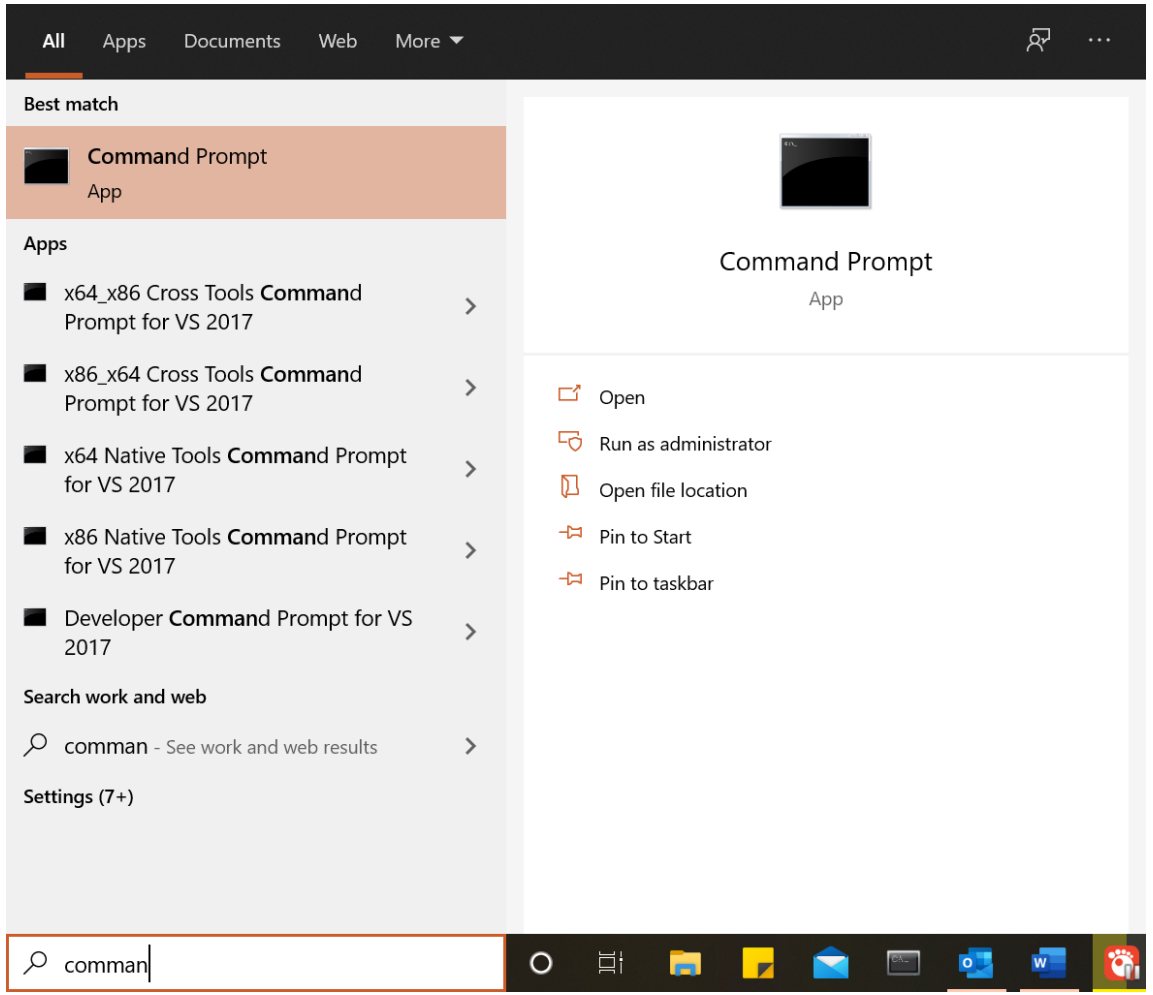
Pcap file has information about packets during certain time.

- Pcap file will be given for the projects.
- Need to analyze the pcap file to find packet’s destination and source IP and so on.

1. Connect to ilab machine

- Open command prompt (Windows 10)

- Or open terminal in Linux



1. Connect to ilab machine

Open a terminal window and type: `ssh Netid@ilab.cs.rutgers.edu`

```
sm2062@ilab3: ~
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-112-generic x86_64)

Last login: Mon Sep 14 21:49:36 2020 from 172.25.152.158

Ubuntu 18.04.4 LTS4.15.0-112-generic
-----------------------------------------------------------------------------
Machine Name:   ilab3.cs          IP No:                        128.6.4.103
Mon Sep 14 21:59:53 EDT 2020      Uptime:                     12 days 11:17
-----------------------------------------------------------------------------
Processes:      4138              Local/SSH/X2Go/Xrdp:  1/28/25/0 (54)
Connections:    157               System Load:          41
Free Memory:    608G of 1.0T      Free Swap:            1.0T of 1.0T
-----------------------------------------------------------------------------
CPU Info:       Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz - 80 cores
System CPU:     2.48%             User CPU:             19.76%
CPU Idle:       77.61%            IO Wait:              0.03%
-----------------------------------------------------------------------------
Login as:       sm2062            No. of Sessions:      1
Avail.UserDisk:                   Avail.Freespace:      3607.88 GB
CUDA Driver:    11.0              CUDA Cores:           14336
-----------------------------------------------------------------------------

A T T E N T I O N:

- If you run CPU/Memory intensive or Long jobs, see:
https://resources.cs.rutgers.edu/docs/limitation-enforced-on-cs-linux-machines/

-----------------------------------------------------------------------------

sm2062@ilab3:~$
```

You should connect to Ilab1, ilab2 or ilab3

You can connect this way

ssh Netid@ilab3.cs.Rutgers.edu

2. Install pcap4j

1. Create folder and type Git clone https://github.com/kaitoy/pcap4j.git in the folder

2. Add the JDK path to Maven toolchains: Create toolchains.xml in ~/.m2/.

Go to the path (~/.m2/) (Type cd ~/.m2/)

Make toolchains.xml like below

```
<?xml version="1.0" encoding="UTF-8"?>
<toolchains xmlns="http://maven.apache.org/TOOLCHAINS/1.1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-i
  xsi:schemaLocation="http://maven.apache.org/TOOLCHAINS/1.1.0 http://maven.apache.org/xsd/toolchains-1.1.
  <toolchain>
    <type>jdk</type>
    <provides>
      <version>11</version>
    </provides>
    <configuration>
      <jdkHome>/path/to/jdk-11</jdkHome>
    </configuration>
  </toolchain>
</toolchains>
```

/usr (where ilab machine installed java)

toolchains.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<toolchains
xmlns="http://maven.apache.org/TOOLCHAINS/1.1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/TOOLCHAINS/1.
1.0 http://maven.apache.org/xsd/toolchains-1.1.0.xsd">
        <toolchain>
          <type>jdk</type>
          <provides>
                <version>11</version>
          </provides>
          <configuration>
               <jdkHome>/usr</jdkHome>
          </configuration>
         </toolchain>
</toolchains>
```

2. Install pcap4j

Go to the project root directory, and execute ./mvnw install
(The root directory is where you downloaded pcap4j)

```
sm2062@ilab3:~/tutorial/pcap4j$ ./mvnw install
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Build Order:
[INFO]
[INFO] Pcap4J                                                             [pom]
[INFO] Pcap4J Core                                                        [jar]
[INFO] Pcap4J Packet Test                                                 [jar]
[INFO] Pcap4J Static Packet Factory                                       [jar]
[INFO] Pcap4J Properties-Based Packet Factory                             [jar]
[INFO] Pcap4J Sample                                                      [jar]
[INFO]
[INFO] -------------------------< org.pcap4j:pcap4j >--------------------------
[INFO] Building Pcap4J 1.8.3-SNAPSHOT                                   [1/6]
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] --- maven-toolchains-plugin:1.1:toolchain (default) @ pcap4j ---
[INFO] Required toolchain: jdk [ version='[9,12)' ]
[INFO] Found matching toolchain for type jdk: JDK[/usr]
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:compile (base-compile) @ pcap4j ---
[INFO] Toolchain in maven-compiler-plugin: JDK[/usr]
[INFO] No sources to compile
[INFO]
[INFO] --- maven-install-plugin:2.4:install (default-install) @ pcap4j ---
[INFO] Installing /ilab/users/sm2062/tutorial/pcap4j/pom.xml to /ilab/users/sm2062/.m2/repository/org/pca
[INFO]
[INFO] -----------------------< org.pcap4j:pcap4j-core >-----------------------
[INFO] Building Pcap4J Core 1.8.3-SNAPSHOT                              [2/6]
[INFO] --------------------------------[ jar ]---------------------------------
```

3. Create a project that uses pcap4j library

- 1. Create folder and type below

```
mvn archetype:generate -DgroupId=com.github.username -DartifactId=pcap -Dversion=1.1.0 -
DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

- 2. Add the pcap4j dependency in pom.xml file.

You can find the pom.xml file in the created folder

Command for fixing indentation in Vim : gg=G

Add dependencies in the pom.xml file

```
<dependency>
    <groupId>org.pcap4j</groupId>
    <artifactId>pcap4j-core</artifactId>
    <version>1.7.3</version>
    <type>jar</type>
  </dependency>

  <dependency>
          <groupId>org.pcap4j</groupId>
             <artifactId>pcap4j-packetfactory-static</artifactId>
                <version>1.6.3</version>
  </dependency>


  <dependency>
         <groupId>org.slf4j</groupId>
           <artifactId>slf4j-simple</artifactId>
             <version>1.7.21</version>
 </dependency>
```

Add a section for **build** and add the following plugin configurations to your existing **pom.xml**

```
<build>
  <plugins>
    <!-- Specify to the compiler we want Java 1.8 -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.7.0</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>

    <!-- Tell the JAR plugin which class is the main class -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-jar-plugin</artifactId>
      <version>3.0.2</version>
      <configuration>
        <archive>
          <manifest>
            <mainClass>com.github.username.App</mainClass>
          </manifest>
        </archive>
      </configuration>
    </plugin>

    <!-- Embed dependencies inside the final JAR -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-shade-plugin</artifactId>
      <version>3.1.0</version>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
            <goal>shade</goal>
          </goals>
        </execution>
      </executions>
      <configuration>
        <finalName>uber-${project.artifactId}-${project.version}</finalName>
      </configuration>
    </plugin>
  </plugins>
</build>
```

4. Run simple example

- Compile first using mvn command
  - Type in the folder where pom.xml exist : mvn package

- Run java file in the created folder (target)
  - Type : java -jar uber-pcap-1.1.0.jar

```
sm2062@ilab3:~/tutorial/pcap/target$ java -jar uber-pcap-1.1.0.jar
Hello World!
sm2062@ilab3:~/tutorial/pcap/target$
sm2062@ilab3:~/tutorial/pcap/target$
```

5. Coding

- Inside the src/main/java directory, drill down the directories until you get to the App.java file

- We can start editing in that file. (App.java)

Simple Example (Read Pcap file)

```
package com.github.username;
import java.io.IOException;
import java.net.Inet4Address;

import com.sun.jna.Platform;

import org.pcap4j.core.NotOpenException;
import org.pcap4j.core.PacketListener;
import org.pcap4j.core.PcapDumper;
import org.pcap4j.core.PcapHandle;
import org.pcap4j.core.PcapNativeException;
import org.pcap4j.core.PcapNetworkInterface;
import org.pcap4j.core.PcapStat;
import org.pcap4j.core.BpfProgram.BpfCompileMode;
import org.pcap4j.core.PcapNetworkInterface.PromiscuousMode;
import org.pcap4j.core.Pcaps;
import org.pcap4j.packet.Packet;
import org.pcap4j.util.NifSelector;
import org.pcap4j.packet.IpV4Packet;
```

```
public class App {

  public static void main(String[] args) throws PcapNativeException, NotOpenException {
    System.out.println("Let's start analysis ");

    final PcapHandle handle;

    handle = Pcaps.openOffline("small.pcap" );

    PacketListener listener = new PacketListener() {
        public void gotPacket(Packet packet) {
            System.out.println(handle.getTimestamp());
            System.out.println(handle);
            System.out.println("packet info");
            System.out.println(packet);

            IpV4Packet ipV4Packet = packet.get(IpV4Packet.class);
            Inet4Address srcAddr = ipV4Packet.getHeader().getSrcAddr();
            System.out.println(srcAddr);

        }
    };

    try {
          int maxPackets = 5;
            handle.loop(maxPackets, listener);
          } catch (InterruptedException e) {
            e.printStackTrace();
      }


    // Cleanup when complete
    handle.close();
  }
}
```