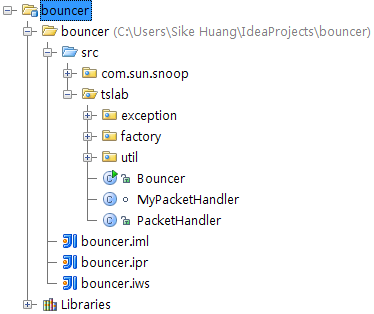
|  |
| --- |
| KTH - Royal Institute of Technology |
| Bouncer |
| IK2213 Network Services and Internet-based Applications |

|  |
| --- |
| Sike Huang and Shanbo Li  2008-5-27 |

## 1. Project Structure



The project is composed of several packages:

**com.sun.snoop** is used for packet validation

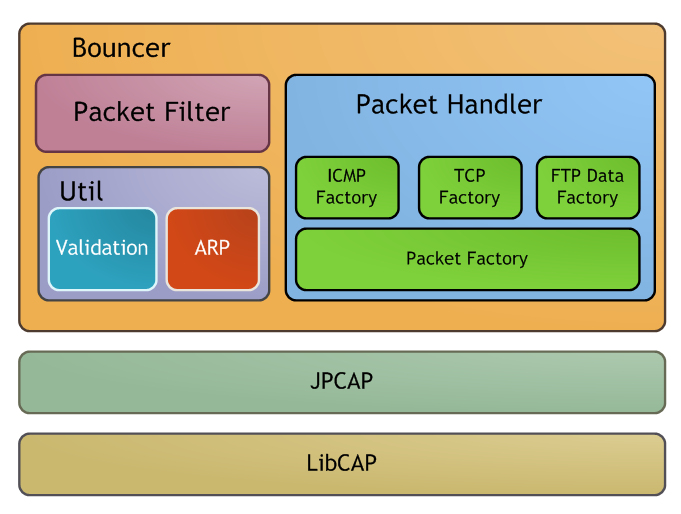
**tslab.exception** contains exceptions that might be thrown during packet creation

**tslab.factory** contains various classes to generate ICMP, TCP and FTP packets

**tslab.util** has common utilization code to support factory

The entry point is **tslab.Bouncer**, which parses the command line arguments, then listens to the incoming packets by **tslab.(My)PacketHandler**, and creates corresponding outing packets using certain suitable factory.

## 2. Architecture



As shown in the figure above. Bouncer uses JpCap to capture packet. And it has a Packet Filter, a set of tools (Util) and Packet Handler. The Packet Handler handles the incoming packet, produces new packet according to it and sends the new packet out. The kernel of Packet Handler which is also the core of Bouncer is a set of Packet Factory.

## 3. Packet Factory



The whole packet producing system is based on Abstract Factory Design Pattern. That is the ICMPFactory, TCPFactory and FTPDataPacketFactory which extend PacketFactory. In the PacketFactory abstract class. We define three abstract methods createPacket(), toServer(), toClient() which are implemented separately by the three concrete factories. toServer() and toClient() method can be invoked directly by external class. While createPacket() is a high level method which invokes toServer() and toClient(). In createPacket() method, it will analyze the incoming packet first and call toServer() or toClient() according to the type of incoming packet. The product of the Factory is outgoing packet which can be sent directly from upper API.

Following table gives a brief introduction of factories.

|  |  |
| --- | --- |
| PacketFactory | The base Factory.  Define abstract method  Use initial() method to configure server properties |
| ICMPFactory | Forward ICMP packet, use for ping |
| TCPFactory | Forword TCP packet, use for TCP an FTP command |
| FTPDataPacketFactory | Forword FTP Data packet. |

Abstract Factory endorses maximum flexibility of switching different factories. Each and every packet is sent to certain factory according to its type, for example, a HTTP packet (TCPPacket) will be passed to TCPPacketFactory and the factory generates outgoing packet in accordance. Therefore, the process of creating and sending packet is factory-independent.

See class PacketHandler for detail.

## 4. Bounce rule

We define four paths as below:

Client Bouncer Server

| ------------packet1----------🡪 | -------packet2--------🡪 |

| | |

| 🡨---------packet4----------- | 🡨-----packet3---------- |

Set routing table:

% route add **192.168.100.100** gw 10.8.0.50

### 4.1 IP level packet modification rule:

|  |  |  |  |
| --- | --- | --- | --- |
| changed field | packet1 | -> | packet2 |
| src.mac | client.mac | -> | bouncer.mac |
| dst.mac | bouncer.mac | -> | server.mac |
| src.ip | client.ip | -> | packet1.dst.ip |
| dst.ip | 192.168.100.100 | -> | server.ip |

|  |  |  |  |
| --- | --- | --- | --- |
| changed field | packet3 | -> | packet4 |
| src.mac | server.mac | -> | bouncer.mac |
| dst.mac | bouncer.mac | -> | client.mac |
| src.ip | server.ip | -> | packet3.dst.ip |
| dst.ip | 192.168.100.100 | -> | client.ip |

### 4.2 TCP level packet modification rule:

|  |  |  |  |
| --- | --- | --- | --- |
| changed field | packet1 | -> | packet2 |
| src.port | client.port | -> | bouncer.outToServer.port |
| dst.port | packet1.dst.port | -> | server.port |

|  |  |  |  |
| --- | --- | --- | --- |
| changed field | packet3 | -> | packet4 |
| src.port | server.port | -> | packet1.dst.port |
| dst.port | packet4.dst.port | -> | client.port |

### 4.3 Special rule for FTP port command:

An adjust value is used to make sequence number and ACK correct when bounce ftp port command. The adjust value is cumulative.

Formula:

Interval = outPacket.data.length – inPacket.data.length;

AdjustValue += Interval;

## 5. Command Line Argument

Our bouncer takes command line argument specific as following:

*java tslab.Bouncer [interface] listen\_ip:listen\_port server\_ip:server\_port*

**interface** is optional, it is the network device used to accept and forward packets, user can either specify it, such as eth0. Or the program will give a list of devices to be chosen, such as:

List of interfaces

0: \Device\NPF\_{8C34DCC7-8F0C-475E-8F62-F159F050B026} [ip=/0.0.0.0]

1: \Device\NPF\_{E09BD06A-3EBA-4364-9F94-0383CADD6DE1} [ip=null]

2: \Device\NPF\_{B920C176-DC06-4740-886B-1051777BB8DE} [ip=/192.168.1.104]

3: \Device\NPF\_{24234719-6C60-4BB4-A604-9600239CDFE5} [ip=/10.8.0.62]

Select one:

**listen\_ip** and **server\_ip** are mandatory.

**listen\_port** and **server\_port** are optional, and they behave in such a way:

|  |  |  |
| --- | --- | --- |
| listern\_port | server\_port | Effect |
| Given | Given | Listen packets from given port, and forward to given port on server (TCP, FTP) |
| Not given | Not given | Listen packets from all ports, and forward to same port on server (ICMP, TCP, FTP) |
| Given | Not given | Listen packets from given port, and forward to same port on server (TCP, FTP) |
| Not given | Given | Listen packets from all ports, and forward to given port on server (TCP, FTP) |

**Please pay attention to these two points:**

1. **ICMP** only works when there is neither listen\_port nor server\_port.
2. And in case of **FTP**, the data channel port on server is assumed to be the given server\_port minus one, and on bouncer port 20 is always opened and used as data transmission channel towards client.