# Analyze raw sockets using packet sniffers

# Lab Manual

**Objectives**

- Build a udp host discovery tool
- Contrast packet sniffing on windows and linux
- Decode IP and TCP layer packets
- Analyze TCP/IP traffic

## Table of Contents

1.

## Background Reading

Textbook:

Black Hat Python, chapter 3

Other resources:

https://www.sans.org/reading-room/whitepapers/malicious/basic-reverse-engineering-immunity-debugger-36982
https://sgros-students.blogspot.ca/2014/05/immunity-debugger-basics-part-1.html
http://tf.nist.gov/tf-cgi/servers.cgi
https://tools.ietf.org/html/rfc4330

## Notes common to all lab and home assignment problems

For every lab and home assignment, all work should go into your personal repository, subdirectory named mXX, where XX stands for the module number. For each problem, carefully name the program as described. The programs are extracted from your repository by a Python script, and errors in the program name will result in the instructor never seeing your program, and your mark for it will be ZERO!

Anything to record and report in this lab is to be written in plain text file (Word document is not a plain text file). The file MUST be named mXXpXX.txt. Mis-naming this file, or not having it in the proper location will result in mark ZERO for anything to be recorded or reported.

There are always many ways how to solve a programming problem, and usually one or two ways which are fast, compact and elegant.

Make sure to push your work to the server often, and have pushed the working version of the program by the deadline specified. The script extracting your programs from your repository will be run at any time after the deadline.

## Lab specific intro

In this lab, we will explore the use of raw sockets with Python. We will learn how to discover hosts on the network and how to send and receive raw packets. We will inspect the packet structure and decode the IP Layer. We will then decode the ICMP protocol packets and use the decoder to sniff packets on the network.

## Problem 1

Write script **`m12sniff1.py`**, which will run both on linux and windows, find the ip address of external interface, bind it, and read one raw packet. As a starting point, use the **`sniffer.py`** from the book. Run the sniffer, ping the machine and report the output of the sniffer. Describe the differences in implementation on Linux and Windows in your report.

## Problem 2

Discover other hosts on the network. Write script **`m12disco.py.`** Find your IP address (see problem 1), and send packets to all hosts within the network. Wait for ICMP reply packets until Ctrl-C is hit on the keyboard. Then dump all host IP addresses which were discovered. Use linux **`script`** to report results.

## Problem 3

Write python script **`m12decode.py`**, which will extend the m12disco.py script, and decode the IP and ICMP headers. Avoid using **`c_ubyte, c_ushort, c_ulong`** data types. Describe why as part of your report. Report different ICMP Type and ICMP Code values received, and explain them. The output should be similar to:

```
My IP: 192.168.1.250
Proto: ICMP 192.168.1.250 -> 192.168.1.250   ICMP -> Type: 3 Code: 3
Host up: 192.168.1.250
Proto: ICMP 192.168.1.250 -> 192.168.1.250   ICMP -> Type: 3 Code: 3
Host up: 192.168.1.250
Proto: ICMP 192.168.1.65 -> 192.168.1.250   ICMP -> Type: 3 Code: 3
Host up: 192.168.1.65
Proto: ICMP 192.168.1.76 -> 192.168.1.250   ICMP -> Type: 3 Code: 3
Host up: 192.168.1.76
```

## Problem 4

Write python scrip m12analyze.py, which will get the time to run as command line parameter,e.g.,

```
m12analyze.py 60
```

Analyze incoming packets for the specified time and report counts by source address and protocol, e.g.

```
Host=192.168.1.65   Proto=ICMP     Count=250
Host=192.168.1.65   Proto=TCP      Count=92
Host=192.168.1.65   Proto=UDP      Count=76
Host=192.168.1.76   Proto=ICMP     Count=250
Host=192.168.1.76   Proto=TCP      Count=46
Host=192.168.1.76   Proto=UDP      Count=28
```