

Analyzing Raw Sockets Using Packet Sniffers

Offensive and Defensive Tool Construction

Table of Contents

[Objectives 4](#_Toc477858191)

[Background Reading 4](#_Toc477858192)

[Important Information 4](#_Toc477858193)

[Introduction 5](#_Toc477858194)

[Problem 1 5](#_Toc477858195)

[Problem 2 5](#_Toc477858196)

[Problem 3 5](#_Toc477858197)

Offensive and Defensive Tool Construction

Analyzing Raw Sockets Using Packet Sniffers

Objectives

This lab focuses on the following objectives:

* Build a UDP host discovery tool.
* Contrast packet sniffing on Windows and Linux.
* Decode IP and TCP layer packets.
* Analyze TCP/IP traffic.

Background Reading

Read chapter 3 in the *Black Hat Python* textbook. The following links are also useful:

* <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>

# Important Information

* For *every* lab and home assignment, store all your work in your personal repository in a subdirectory named **mXX**, where XX is the module number. Carefully name the program as described in each problem.
* Your programs are extracted from your repository by a Python script. If there are any errors in the program name, then your instructor will never see your program, and you will receive a mark of zero.
* Push your work to the server often, and ensure that you push the final version of a program by the deadline specified, because the script extracting them can be run at any time after the deadline.

# Introduction

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

1. Write a program named **script m12sniff1.py** that runs on both Linux and Windows, finds the IP address of an external interface, binds it, and then reads one raw packet.
2. As a starting point, use the **sniffer.py** program from the textbook. Run the sniffer, ping the computer, and then report the output of the sniffer.
3. Describe the differences in implementation between Linux and Windows in your report.

# Problem 2

1. Write a script named **m12disco.py** that finds your IP address (see Problem 1), and then send packets to all hosts on the network. Wait for ICMP reply packets until **CTRL+C** is pressed on the keyboard. Then, dump all discovered host IP addresses.
2. Use a Linux script to report the results.

# Problem 3

1. Write a Python script named **m12decode.py** that extends the m12disco.py script from Problem 2 and decodes the IP and ICMP headers. Avoid using the **c\_ubyte**, **c\_ushort** or **c\_ulong** data types and describe why they should be avoided as part of your report.
2. Report on the different ICMP types and ICMP code values received, and explain them. Your output should be similar to the example below:

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