

Implementing Basic Networking Utilities

Offensive and Defensive Tool Construction

Table of Contents

[Objectives 4](#_Toc477850579)

[Background Reading 4](#_Toc477850580)

[Important Information 4](#_Toc477850581)

[Introduction 5](#_Toc477850582)

[Problem 1 5](#_Toc477850583)

[Problem 2 5](#_Toc477850584)

[Problem 3 5](#_Toc477850585)

[Problem 4 5](#_Toc477850586)

Problem 5 ………………………………..........................................................................................5

Offensive and Defensive Tool Construction

Implementing Basic Networking Utilities

Objectives

This lab focuses on the following objectives:

* Create simple clients and servers.
* Write Python code to replace netcat.
* Build a TCP proxy.
* Create an SSH server.
* Apply SSH tunnelling.

Background Reading

Read chapter 2 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 basics of networking with Python. You will learn about network sockets, create clients and servers, and write code for other networking utilities in Python. The core Python module for network programming is the socket module, and you will explore how to create a socket, connect to a remote host, listen on a socket on a specific port, and send and receive data packets.

# Problem 1

1. Write a simple TCP web client in Python named **m11tcpcli.py** that connects to the website <http://itss.biomea.com> and gets page **itsc203/cgi-bin/echo.cgi**.
2. Report on the information you receive.

# Problem 2

1. Write a simple TCP server named **m11tcpsrv.py** using information from the “TCP Server” chapter in the *Black Hat Python* textbook.
2. Run the server, point it to your browser and record what the server receives.

# Problem 3

1. Write a daytime server in Python named **m11timed.py** and have it listen on port 2013.
2. Implement UDP protocol.
3. Have the server reply to any packet received with a packet containing the UTC current date and time in the following format: YYYY-MM-DD HH:MM:SS <crlf>
4. Use a Linux command script to save your interactive terminal session.
5. Connect to your server using netcat in a Linux environment.
6. Report the relevant part of your console log.

# Problem 4

1. Write a simple UDP client in Python named **m11udpc.py** that gets two parameters: the name or IP address of the server and the port to send one UDP packet to.
2. Run the client and request the time from the server you implemented in Problem 2.
3. Report the time, corrected for your local time zone.
4. Report the console log.

# Problem 5

1. Improve the daytime server from Problem 2 and name it **m11synctimed.py**.
2. Have this server get accurate time information by sending a UDP packet to **time.nist.go** on port 37, using the SNTP protocol described in RFC4330.
3. Connect to the server using the UDP client you wrote in Problem 3.
4. Report the console log.