BRUTEFORCE PORT SCANNER

IN PYTHON

By

ACHO ARNOLD E3WIN (FE12A003)

NKWETEYIM DAISY AATSE (FE12A145)

FONJI FONKENG ROCARD (FE12A069)

EKWOGE ESEH BLAISE (FE12A050)

NNOKO DEGRANDO ANYOPEH (FE12A147)

FOKO MABOU PINEL VALDES (FE12A063)

# Title

## Bruteforce port scanner In Python

# Requirements

## Hardware Requirements

| Part | Specification |
| --- | --- |
| RAM | 8GB |
| Processor | 3.5GHz |

## Software Requirements

| Software Type | Specifications |
| --- | --- |
| Operating System | Windows 8 Professional |
| Python Version | 2.7.9 |

# Flow Chart

# C:\Users\User\Downloads\Flow Chart.png

# Source Code

*#!/usr/bin/python*\_\_author\_\_ = **‘Group 1’***# 2016, FET, UB  
# Learning python***import** socket,sys,time,datetime,argparse,os  
  
flag = 0 *# we’ll use this flag later*os.system(**'cls'**) *# Clear the console window*line = **"+"** \* 80 *# Just a fancy line consisting '+'*

desc = line+**'''\nA Simple port scanner that works!! (c) digitz.org  
 Example usage: python port\_scanner.py example.com 1 1000  
 The above example will scan the host \'example.com\' from port 1 to 1000  
 To scan most common ports, use: python port\_scanner.py example.com\n'''**+line+**"\n"** *# Just a description about the script and how to use it*

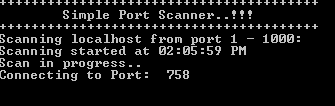
parser = argparse.ArgumentParser(description = desc, formatter\_class=argparse.RawTextHelpFormatter)  
parser.add\_argument(**'host'**, metavar=**'H'**, help=**'Host name you want to scan'**)  
parser.add\_argument(**'startport'**, metavar=**'P1'**, nargs=**'?'**, help=**'Start scanning from this port'**)  
parser.add\_argument(**'endport'**, metavar=**'P2'**, nargs=**'?'**,help=**'Scan until this port'**)  
args = parser.parse\_args()  
  
host = args.host *# The host name to scan for open ports*ip = socket.gethostbyname(host) *# Converts the host name into IP address  
  
# Here, we're checking if both starting port and ending port is defined  
# If it is not defined, we will do a scan over most popular TCP ports.***if** (args.startport) **and** args.endport :  
 *# If this condition is true, the script will scan over this port range* start\_port = int(args.startport)  
 end\_port = int(args.endport)  
**else**:flag = 1  
open\_ports = [] *# This list is used to hold the open ports  
  
# This dictionary contains the most popular ports used  
# You can add ports here.  
# The key is the port number and the values is the service used by that port*common\_ports = {**'21'**: **'FTP'**, **'22'**: **'SSH'**, **'23'**: **'TELNET'**, **'25'**: **SMTP'**, **'53'**: **'DNS'**, **'69'**: **'TFTP'**, **'80'**: **'HTTP'**, **'109'**: **'POP2'**, **'110'**: **'POP3'**, **'123'**: **'NTP'**, **'137'**: **'NETBIOS-NS'**, **'138'**: **'NETBIOS-DGM'**, **'139'**: **'NETBIOS-SSN'**, **'143'**: **'IMAP'**, **'156'**: **'SQL-SERVER'**, **'389'**: **'LDAP'**, **'443'**: **'HTTPS'**, **'546'**: **'DHCP-CLIENT'**, **'547'**: **'DHCP-SERVER'**, **'995'**: **'POP3-SSL'**, **'993'**: **'IMAP-SSL'**, **'2086'**: **'WHM/CPANEL'**, **'2087'**: **'WHM/CPANEL'**, **'2082'**: **'CPANEL'**, **'2083'**: **'CPANEL'**, **'3306'**: **'MYSQL'**, **'8443'**: **'PLESK'**, **'10000'**: **'VIRTUALMIN/WEBMIN'**}

starting\_time = time.time() *# Get the time at which the scan was started***print "+"** \* 40  
**print "\tSimple Port Scanner..!!!"  
print "+"** \* 40  
  
**if** (flag): *# The flag is set, that means the user did not provide any ports as argument* **print "Scanning for most common ports on %s"** % (host)  
**else**:  
 *# The user did specify a port range to scan* **print "Scanning %s from port %s - %s: "** % (host, start\_port, end\_port)  
**print "Scanning started at %s"** %(time.strftime(**"%I:%M:%S %p"**))  
  
  
*# This is the function that will connect to a port and will check  
# if it is open or closed***def** check\_port(host, port, result = 1):  
 *# The function takes 3 arguments  
 # host : the IP to scan  
 # port : the port number to connect* **try**:sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)  
 *# Setting socket timeout so that the socket does not wait forever to complete a connection* sock.settimeout(0.5) *# if the connection was successful, that means the port  
 # is open, and the output 'r' will be zero* r = sock.connect\_ex((host, port))  
 **if** r == 0:  
 result = r  
 sock.close() *# closing the socket* **except** Exception, e:  
 **pass  
 return** result *# returns the result of the scan.  
  
# This function reads the dictonary of ports and services and  
# Checks for the service name corresponding to a port.***def** get\_service(port):  
 port = str(port) *# converts the int to string* **if** port **in** common\_ports: *# check if the port is available in the common ports dictionary* **return** common\_ports[port] *# returns the service name if available* **else**:  
 **return** 0 *# return 0 if no service is identified*

**try**:  
 **print "Scan in progress.."  
 print "Connecting to Port: "**,  
 **if** flag: *# The flag is set, means the user did not give any port range* **for** p **in** sorted(common\_ports): *# So we will scan the common ports.* sys.stdout.flush() *# flush the stdout buffer.* p = int(p)  
 **print** p,  
 response = check\_port(host, p) **if** response == 0: *# The port is open* open\_ports.append(p) *# append it to the list of open ports  
 #if not p == end\_port:* sys.stdout.write(**'\b'** \* len(str(p))) *# This is just used to clear the port number displayed. This is not important at all* **else**:  
*# The user did provide a port range, now we have to scan through that range* **for** p **in** range(start\_port, end\_port+1):  
 sys.stdout.flush()  
 **print** p,  
 response = check\_port(host, p)**if** response == 0: *# Port is open* open\_ports.append(p) *# Append to the list of open ports* **if not** p == end\_port:  
 sys.stdout.write(**'\b'** \* len(str(p)))  
  
 **print "\nScanning completed at %s"** %(time.strftime(**"%I:%M:%S %p"**))  
 ending\_time = time.time()  
 total\_time = ending\_time - starting\_time **print "="** \* 40  
 **print "\tScan Report: %s"** %(host)  
 **print "="** \* 40  
 **if** total\_time <= 60:  
 total\_time = str(round(total\_time, 2))  
 **print "Scan Took %s seconds"** %(total\_time)  
 **else**:  
 total\_time = total\_time / 60  
 **print "Scan Took %s Minutes"** %(total\_time)  
  
 **if** open\_ports: *# There are open ports available* **print "Open Ports: "  
 for** i **in** sorted(open\_ports):  
 service = get\_service(i)  
 **if not** service: *# The service is not in the disctionary* service = **"Unknown service"  
 print "\t%s %s: Open"** % (i, service)  
 **else**:  
 *# No open ports were found* **print "Sorry, No open ports found.!!"  
  
except** KeyboardInterrupt: *# This is used in case the user press "Ctrl+C", it will show the following error instead of a python's scary error* **print "You pressed Ctrl+C. Exiting "** sys.exit(1)

# Output

In bruteforce progress



End of Scan

