Practical 6

Programming in Security

1. Learn to code basic sockets for networking
2. Install Nmap ("Network Mapper"), a free and open source utility for network discovery and security auditing. Nmap uses raw IP packets to determine what hosts are available on the network, what services (application name and version) those hosts are offering, what operating systems (and OS versions) they are running, what type of packet filters/firewalls are in use, and dozens of other characteristics. It was designed to rapidly scan large networks, but works fine against single hosts.

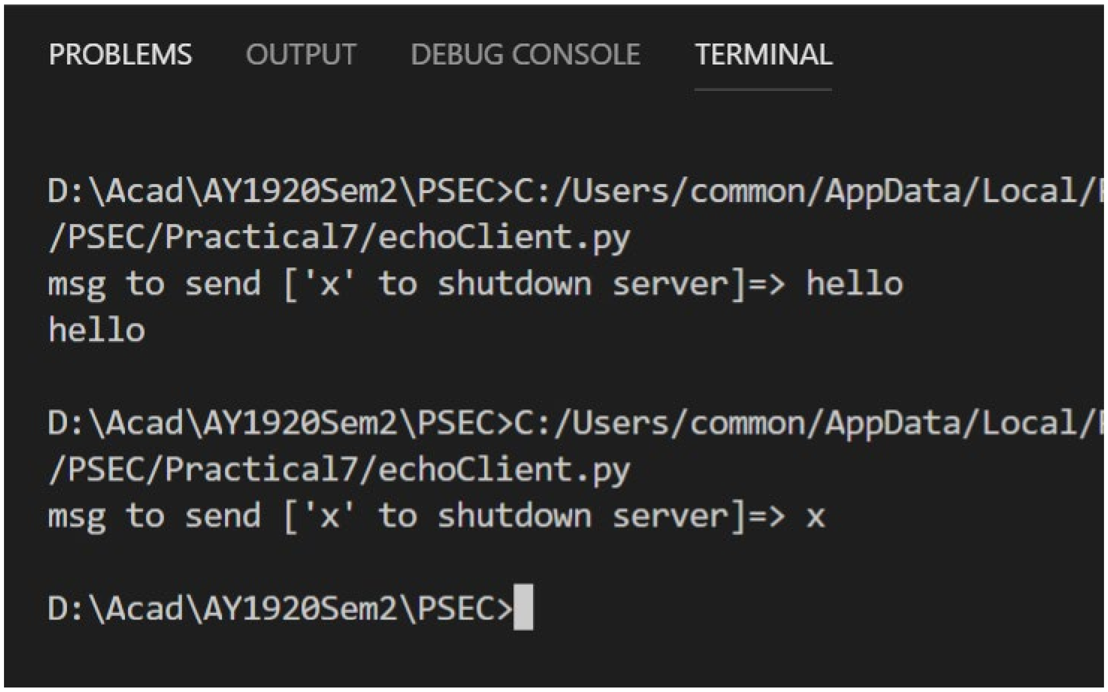
In this practical, you will …

- Import and code using python sockets
- Install Nmap
- use Python to execute Nmap scan

created by Junie Tan

Sockets

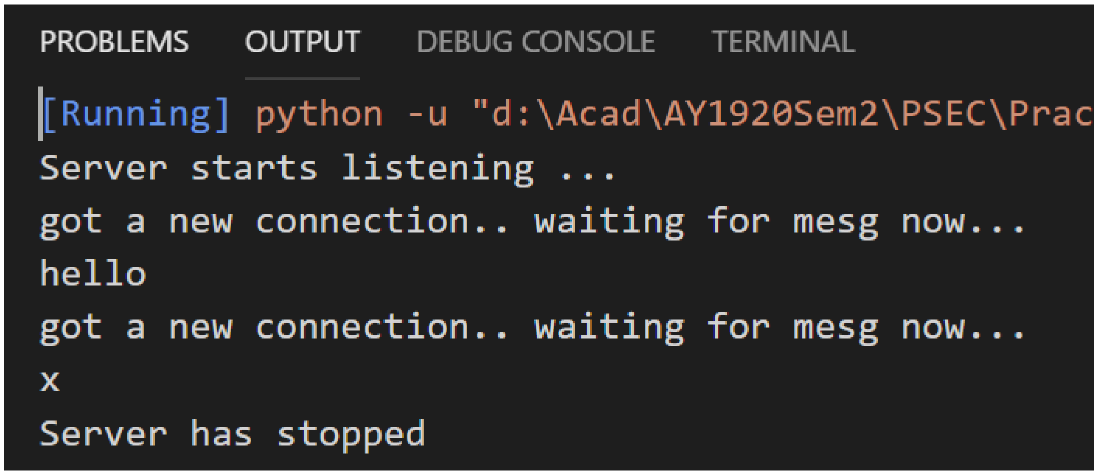
1. From slides 8-9 code a simple echoServer and echoClient pair that allows the server to echo messages sent from the client, and shutdown the server from the client end.



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
D:\Acad\AY1920Sem2\PSEC>C:/Users/common/AppData/Local/
/PSEC/Practical7/echoClient.py
msg to send ['x' to shutdown server]=> hello
hello

D:\Acad\AY1920Sem2\PSEC>C:/Users/common/AppData/Local/
/PSEC/Practical7/echoClient.py
msg to send ['x' to shutdown server]=> x

D:\Acad\AY1920Sem2\PSEC>
```

Message at client end

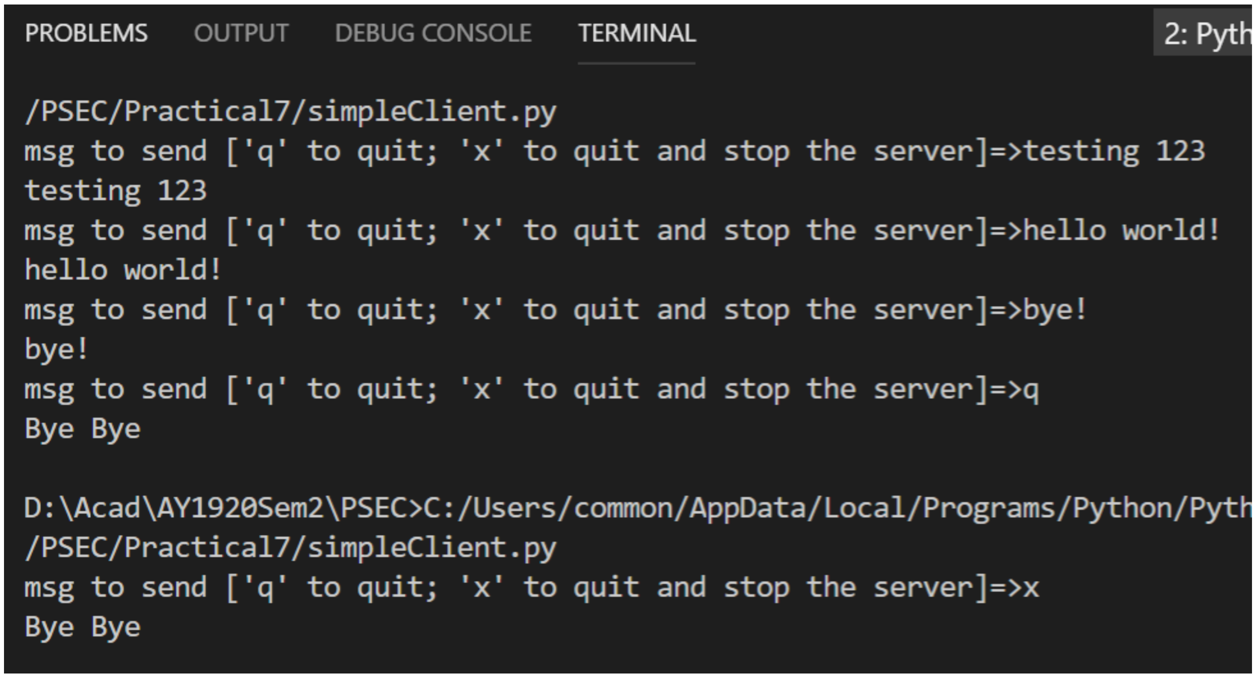


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
[Running] python -u "d:\Acad\AY1920Sem2\PSEC\Prac
Server starts listening ...
got a new connection.. waiting for mesg now...
hello
got a new connection.. waiting for mesg now...
x
Server has stopped
```

Message at server end

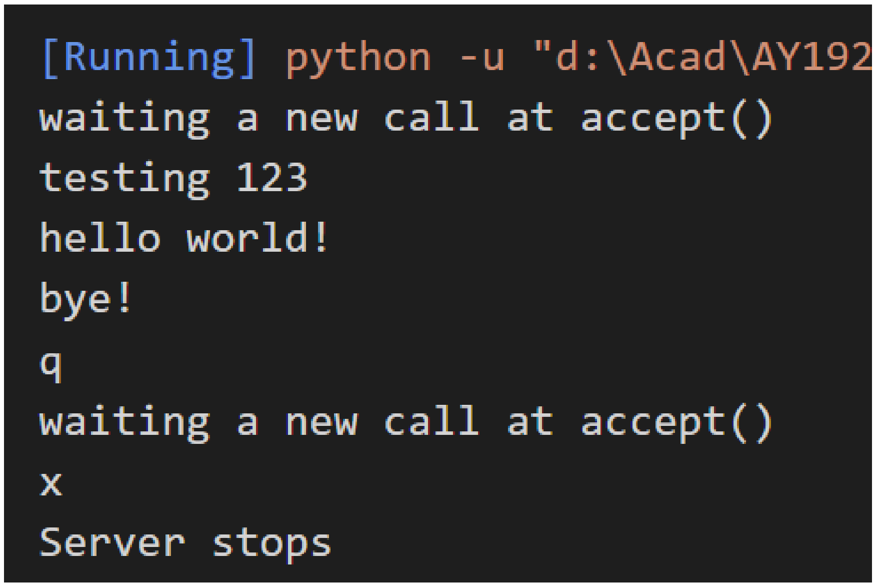
2. From slides 10-11 code a simple simServer and simClient pair that loops to allow the server to echo multiple messages sent from the client, and quit, leaving the server running. There can be a separate command to shutdown the server from the client end.



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: Pyth

```
/PSEC/Practical7/simpleClient.py
msg to send ['q' to quit; 'x' to quit and stop the server]=>testing 123
testing 123
msg to send ['q' to quit; 'x' to quit and stop the server]=>hello world!
hello world!
msg to send ['q' to quit; 'x' to quit and stop the server]=>bye!
bye!
msg to send ['q' to quit; 'x' to quit and stop the server]=>q
Bye Bye

D:\Acad\AY1920Sem2\PSEC>C:/Users/common/AppData/Local/Programs/Python/Pyth
/PSEC/Practical7/simpleClient.py
msg to send ['q' to quit; 'x' to quit and stop the server]=>x
Bye Bye
```

Message at client end

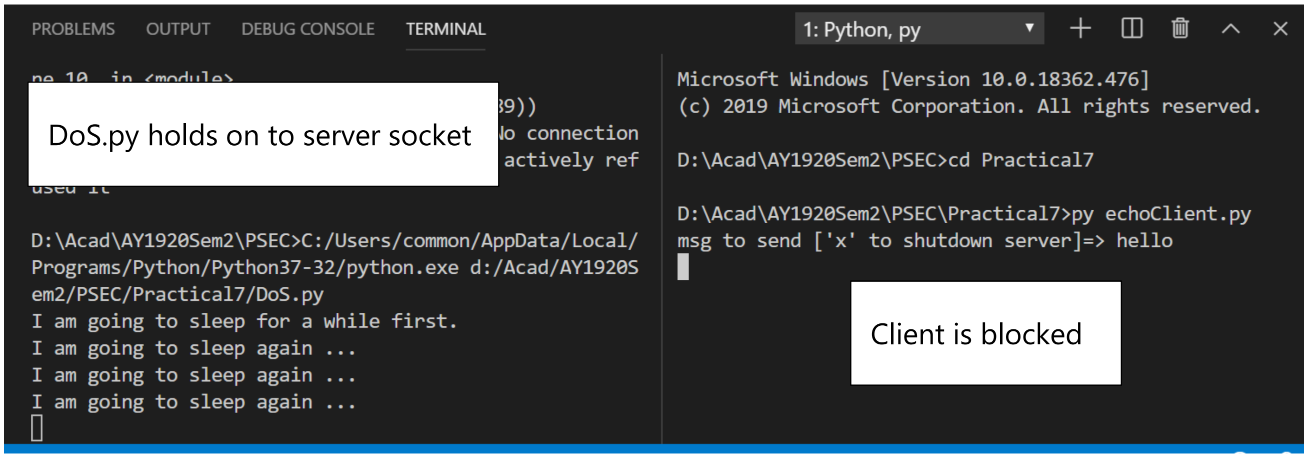


```
[Running] python -u "d:\Acad\AY192
waiting a new call at accept()
testing 123
hello world!
bye!
q
waiting a new call at accept()
x
Server stops
```

Message at server end

3. Run the following DoS.py script to connect to echoServer.py and deny client sockets access to the server.

```python
#similar to simClient.py, but this client never sends nor recv to/from
#the server
#after the connection is established. The trouble it made is to hang
#the echo
#server . kind of DOS effect for the rest of the clients (if  any)
import socket
import time
def getnewsocket():
        return socket.socket(socket.AF_INET, socket.SOCK_STREAM)

clientsocket = getnewsocket()
clientsocket.connect(('localhost', 8089))
print("I am going to sleep for a while first.")
while True:
        time.sleep(10) # sleep for 30 seconds
        print("I am going to sleep again ...")
clientsocket.close()
print("Bye Bye")
```



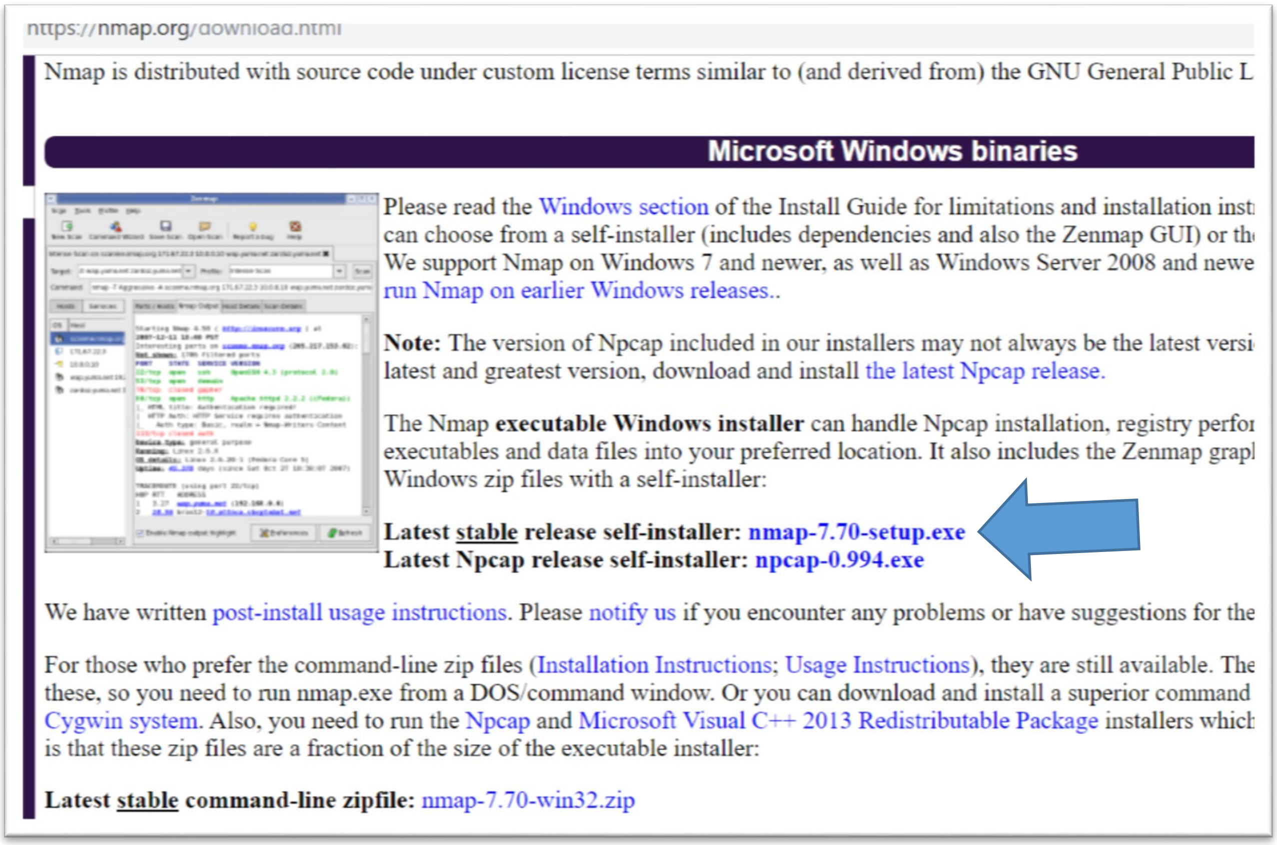
```
ne 10, in <module>
                              39))
                              No connection
                              actively ref
used it

D:\Acad\AY1920Sem2\PSEC>C:/Users/common/AppData/Local/
Programs/Python/Python37-32/python.exe d:/Acad/AY1920S
em2/PSEC/Practical7/DoS.py
I am going to sleep for a while first.
I am going to sleep again ...
I am going to sleep again ...
I am going to sleep again ...
```

```
Microsoft Windows [Version 10.0.18362.476]
(c) 2019 Microsoft Corporation. All rights reserved.

D:\Acad\AY1920Sem2\PSEC>cd Practical7

D:\Acad\AY1920Sem2\PSEC\Practical7>py echoClient.py
msg to send ['x' to shutdown server]=> hello
```

Install Nmap

Installing Nmap

a) Visit https://nmap.org/download.html to download the latest stable release self-installer for Windows (or your respective OS): **nmap-7.70-setup.exe**.



https://nmap.org/download.html

Nmap is distributed with source code under custom license terms similar to (and derived from) the GNU General Public L

Microsoft Windows binaries

Please read the Windows section of the Install Guide for limitations and installation inst can choose from a self-installer (includes dependencies and also the Zenmap GUI) or th We support Nmap on Windows 7 and newer, as well as Windows Server 2008 and newe run Nmap on earlier Windows releases..

Note: The version of Npcap included in our installers may not always be the latest versi latest and greatest version, download and install the latest Npcap release.

The Nmap **executable Windows installer** can handle Npcap installation, registry perfo executables and data files into your preferred location. It also includes the Zenmap grapl Windows zip files with a self-installer:

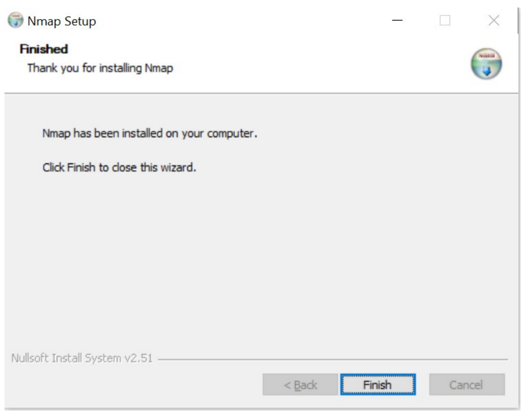
Latest stable release self-installer: nmap-7.70-setup.exe
Latest Npcap release self-installer: npcap-0.994.exe

We have written post-install usage instructions. Please notify us if you encounter any problems or have suggestions for the

For those who prefer the command-line zip files (Installation Instructions; Usage Instructions), they are still available. The these, so you need to run nmap.exe from a DOS/command window. Or you can download and install a superior command Cygwin system. Also, you need to run the Npcap and Microsoft Visual C++ 2013 Redistributable Package installers whicl is that these zip files are a fraction of the size of the executable installer:

Latest stable command-line zipfile: nmap-7.70-win32.zip

b) Double click the installer to install. Accept the license agreement and install using the default options.



Nmap Setup

Finished

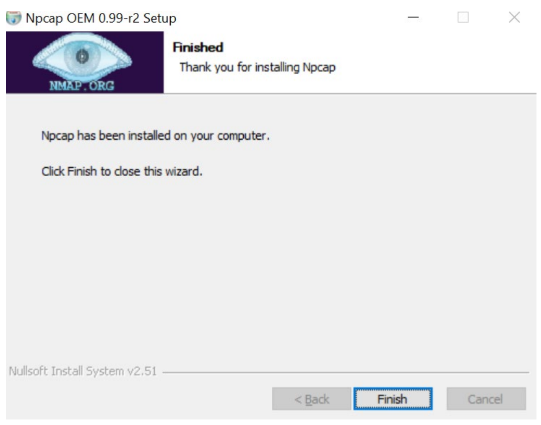
Thank you for installing Nmap

Nmap has been installed on your computer.

Click Finish to close this wizard.

Nullsoft Install System v2.51

or



Npcap OEM 0.99-r2 Setup

Finished

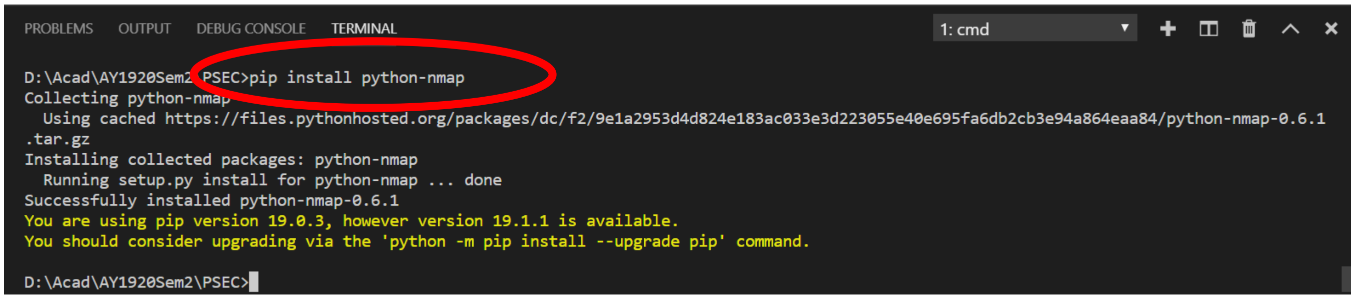
Thank you for installing Npcap

Npcap has been installed on your computer.

Click Finish to close this wizard.

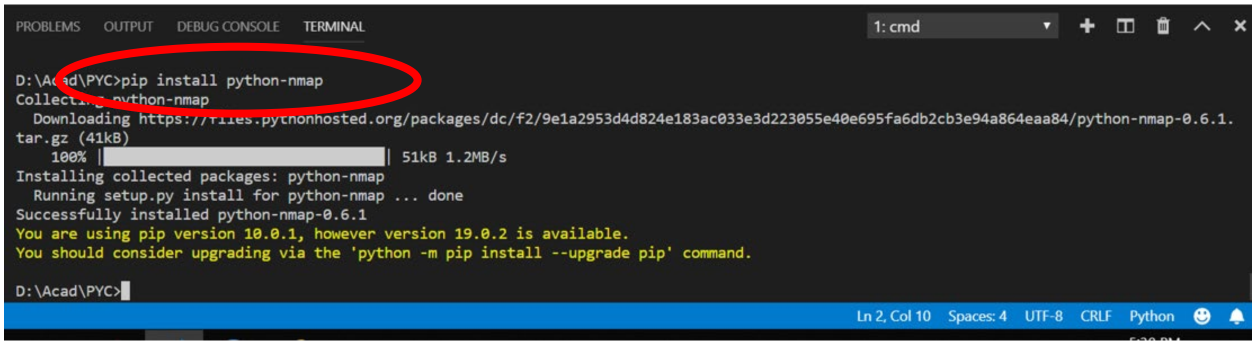
Nullsoft Install System v2.51

c) In Visual Studio Code's Terminal, type ***pip install python-nmap***.



```
D:\Acad\AY1920Sem2\PSEC>pip install python-nmap
Collecting python-nmap
  Using cached https://files.pythonhosted.org/packages/dc/f2/9e1a2953d4d824e183ac033e3d223055e40e695fa6db2cb3e94a864eaa84/python-nmap-0.6.1
.tar.gz
Installing collected packages: python-nmap
  Running setup.py install for python-nmap ... done
Successfully installed python-nmap-0.6.1
You are using pip version 19.0.3, however version 19.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

D:\Acad\AY1920Sem2\PSEC>
```

Or

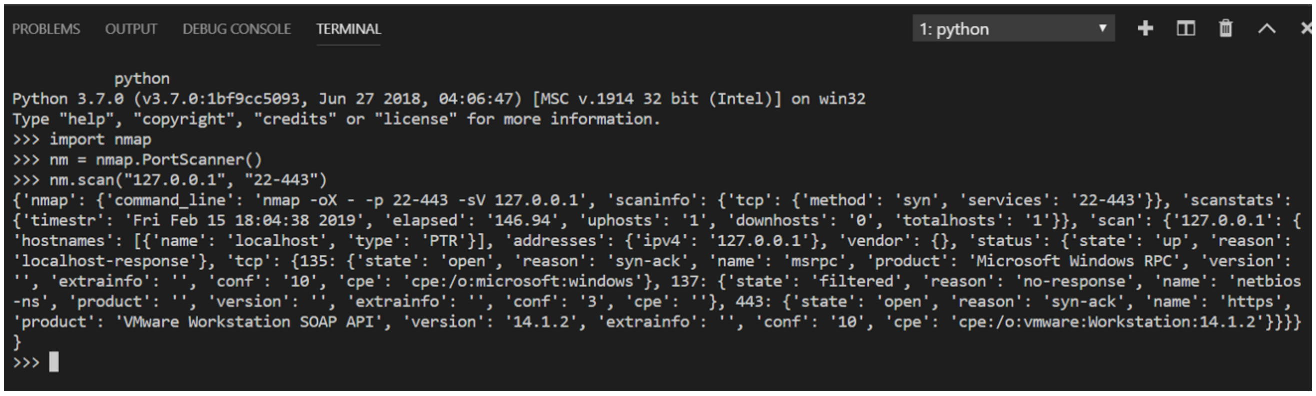


```
D:\Acad\PYC>pip install python-nmap
Collecting python-nmap
  Downloading https://files.pythonhosted.org/packages/dc/f2/9e1a2953d4d824e183ac033e3d223055e40e695fa6db2cb3e94a864eaa84/python-nmap-0.6.1.
tar.gz (41kB)
    100% |████████████████████████████████| 51kB 1.2MB/s
Installing collected packages: python-nmap
  Running setup.py install for python-nmap ... done
Successfully installed python-nmap-0.6.1
You are using pip version 10.0.1, however version 19.0.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

D:\Acad\PYC>
```

Use Python to execute Nmap scan

Test your Nmap installation.

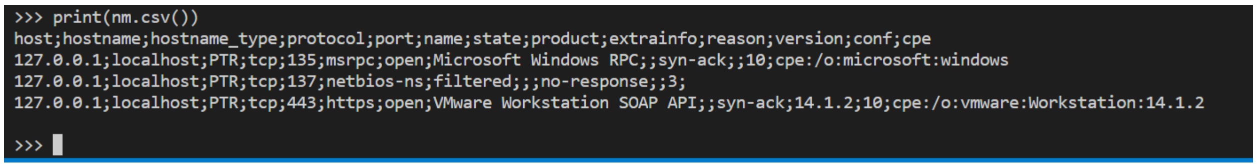
a) Run the Python interactive interpreter and test your Nmap installation with the following lines of Python code.



```
           python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import nmap
>>> nm = nmap.PortScanner()
>>> nm.scan("127.0.0.1", "22-443")
{'nmap': {'command_line': 'nmap -oX - -p 22-443 -sV 127.0.0.1', 'scaninfo': {'tcp': {'method': 'syn', 'services': '22-443'}}, 'scanstats':
{'timestr': 'Fri Feb 15 18:04:38 2019', 'elapsed': '146.94', 'uphosts': '1', 'downhosts': '0', 'totalhosts': '1'}}, 'scan': {'127.0.0.1': {
'hostnames': [{'name': 'localhost', 'type': 'PTR'}], 'addresses': {'ipv4': '127.0.0.1'}, 'vendor': {}, 'status': {'state': 'up', 'reason':
'localhost-response'}, 'tcp': {135: {'state': 'open', 'reason': 'syn-ack', 'name': 'msrpc', 'product': 'Microsoft Windows RPC', 'version':
'', 'extrainfo': '', 'conf': '10', 'cpe': 'cpe:/o:microsoft:windows'}, 137: {'state': 'filtered', 'reason': 'no-response', 'name': 'netbios
-ns', 'product': '', 'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}, 443: {'state': 'open', 'reason': 'syn-ack', 'name': 'https',
'product': 'VMware Workstation SOAP API', 'version': '14.1.2', 'extrainfo': '', 'conf': '10', 'cpe': 'cpe:/o:vmware:Workstation:14.1.2'}}}}
}
>>> 
```

```
>>> import nmap
>>> nm = nmap.PortScanner()
>>> nm.scan('127.0.0.1', '22-443')

{'nmap': {'command_line': 'nmap -oX - -p 22-443 -sV 127.0.0.1', 'scaninfo': {'tcp': {'method':
'syn', 'services': '22-443'}}, 'scanstats':

{'timestr': 'Mon May 13 10:06:34 2019', 'elapsed': '146.88', 'uphosts': '1', 'downhosts': '0',
'totalhosts': '1'}}, 'scan': {'127.0.0.1': {'hostnames': [{'name': 'localhost', 'type':
'PTR'}], 'addresses': {'ipv4': '127.0.0.1'}, 'vendor': {}, 'status': {'state': 'up', 'reason':

'localhost-response'}, 'tcp': {135: {'state': 'open', 'reason': 'syn-ack', 'name': 'msrpc',
'product': 'Microsoft Windows RPC', 'version':

[...]
```

b) Use print(nm.csv()) to view the scanned results in a more readable format.



```
>>> print(nm.csv())
host;hostname;hostname_type;protocol;port;name;state;product;extrainfo;reason;version;conf;cpe
127.0.0.1;localhost;PTR;tcp;135;msrpc;open;Microsoft Windows RPC;;syn-ack;;10;cpe:/o:microsoft:windows
127.0.0.1;localhost;PTR;tcp;137;netbios-ns;filtered;;;no-response;;3;
127.0.0.1;localhost;PTR;tcp;443;https;open;VMware Workstation SOAP API;;syn-ack;14.1.2;10;cpe:/o:vmware:Workstation:14.1.2

>>> 
```

```
>>> print(nm.csv())
host;protocol;port;name;state;product;extrainfo;reason;version;conf
127.0.0.1;tcp;22;ssh;open;OpenSSH;protocol 2.0;syn-ack;5.9p1 Debian 5ubuntu1;10
127.0.0.1;tcp;25;smtp;open;Exim smtpd;;syn-ack;4.76;10
127.0.0.1;tcp;53;domain;open;dnsmasq;;syn-ack;2.59;10
```

created by Junie Tan

c) Other commands you can test using your nmap can include:

```
>>> import nmap
>>> nm = nmap.PortScanner()
>>> nm.scan('127.0.0.1', '22-443')
>>> nm.command_line()
'nmap -oX - -p 22-443 -sV 127.0.0.1'
>>> nm.scaninfo()
{'tcp': {'services': '22-443', 'method': 'connect'}}
>>> nm.all_hosts()
['127.0.0.1']
>>> nm['127.0.0.1'].hostname()
'localhost'
>>> nm['127.0.0.1'].state()
'up'
>>> nm['127.0.0.1'].all_protocols()
['tcp']
>>> nm['127.0.0.1']['tcp'].keys()
[80, 25, 443, 22, 111]
>>> nm['127.0.0.1'].has_tcp(22)
True
>>> nm['127.0.0.1'].has_tcp(23)
False
>>> nm['127.0.0.1']['tcp'][22]
{'state': 'open', 'reason': 'syn-ack', 'name': 'ssh'}
>>> nm['127.0.0.1'].tcp(22)
{'state': 'open', 'reason': 'syn-ack', 'name': 'ssh'}
>>> nm['127.0.0.1']['tcp'][22]['state']
'open'

>>> nm.scan(hosts='192.168.1.0/24', arguments='-n -sP -PE -
PA21,23,80,3389')
>>> hosts_list = [(x, nm[x]['status']['state']) for x in nm.all_hosts()]
>>> for host, status in hosts_list:
>>>     print(f'{host}:{status}')
192.168.1.0:down
192.168.1.1:up
192.168.1.10:down
192.168.1.101:down
192.168.1.102:down
192.168.1.103:down
```

Or use any resources available on the internet.

d) Other sites you can use to test your nmap commands include, www.hackthissite.org and scanme.nmap.org. You can also use the servers you accessed in your earlier semesters.

Pls do not test public sites that did not give prior consent to allow your scanning!

-- End --

created by Junie Tan