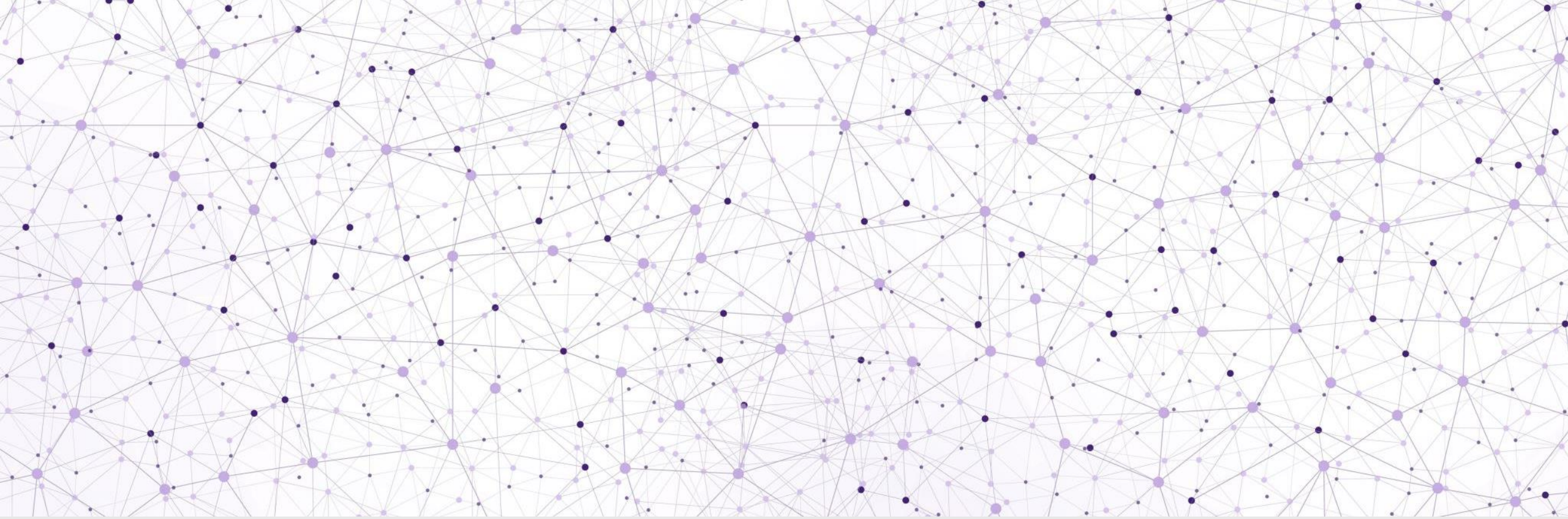

Escaneamento de Portas com nmap

Prof. Ricardo Mesquita

Introdução

- Instalação do nmap:

```
$ sudo apt-get install python3-setuptools
$ sudo pip3.10 install python-nmap
```

- Etapas:
  - Reconhecimento da ferramenta nmap para varredura de portas.
  - Principais tipos de varredura que suportadas pelo nmap.
  - Escaneamento para analisar portas e serviços executados em um host específico.
  - Depois de identificar diferentes hosts em sua rede, realizar uma varredura de portas em cada host identificado.

Introdução

- O Nmap é um projeto de grande importância para a segurança cibernética.
- A varrefura de portas é, geralmente, a primeira ação de um analista de segurança para avaliar o nível de exposição de um alvo potencial.
- O nmap é, atualmente, o melhor programa para realizar uma varredura de hosts em uma rede local.
- Para baixar a última versão disponível: https://nmap.org/download.html
- Para checar as oções:

```
$ nmap
Nmap 7.92 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}
TARGET SPECIFICATION:
        Can pass hostnames, IP addresses, networks, etc.
        Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0-255.1-254
```

Algumas Opções do nmap

sT (*TCP Connect Scan*):

- Opção normalmente usada para detectar se uma porta está aberta ou fechada.
- Com esta opção, uma porta é aberta se o servidor responder com um pacote contendo a flag ACK ao enviar um pacote com a flag SYN.

sS (*TCP Stealth Scan*):

- É um tipo de scan baseado no TCP Connect Scan com a diferença de que a conexão na porta não é feita completamente.
- Esta opção consiste em verificar o pacote de resposta do alvo antes de verificar o pacote com a flag SYN habilitada.
- Se o destino responder com um pacote que contém o sinalizador RST, você poderá verificar se a porta está aberta ou fechada.

Algumas Opções do nmap

sU (*UDP Scan*):

- É um tipo de varredura onde um pacote UDP é enviado para determinar se a porta está aberta.
- Se a resposta for outro pacote UDP, significa que a porta está aberta.
- Se a resposta retornar um pacote ICMP (Internet Control Message Protocol) do tipo 3 (destino inacessível), a porta não estará aberta.

sA (*TCP ACK Scan*):

- É uma varredura que nos permite saber se a máquina alvo possui algum tipo de firewall em execução.
- Esta opção de varredura envia um pacote com o sinalizador ACK ativado para a máquina de destino.
- Se a máquina remota responder com um pacote onde o sinalizador RST está ativado, pode-se determinar que a porta não está filtrada por nenhum firewall.
- Se não obtivermos uma resposta da máquina remota, pode-se determinar que existe um firewall filtrando os pacotes enviados para a porta especificada.

Algumas Opções do nmap

sN (*TCP NULL Scan*):

- Envia um pacote TCP para a máquina de destino sem nenhum sinalizador.
- Se a máquina remota retornar uma resposta válida, poderá ser determinado que a porta está aberta.
- Caso contrário, se a máquina remota retornar um flag RST, podemos dizer que a porta está fechada.

sF (*TCP FIN Scan*):

- Envia um pacote TCP para a máquina de destino com o sinalizador FIN.
- Se a máquina remota retornar uma resposta, poderá ser determinado que a porta está aberta.
- Se a máquina remota retornar um flag RST, podemos dizer que a porta está fechada.

sX (*TCP XMAS Scan*):

- Envia um pacote TCP para a máquina de destino com o sinalizador PSH, FIN ou URG.
- Se a máquina remota retornar uma resposta válida, poderá ser determinado que a porta está aberta.
- Se a máquina remota retornar um flag RST, podemos dizer que a porta está fechada.
- Se obtivermos um pacote ICMP tipo 3 na resposta, a porta será filtrada.

Uso das Opções de Varredura

- O tipo de varredura padrão pode variar dependendo do usuário que a executa, devido às permissões para o envio dos pacotes durante a varredura.
- As diferenças entre os tipos de varredura são os pacotes retornados da máquina alvo e sua capacidade de evitar que sejam detectados por sistemas de segurança, como firewalls ou sistemas de detecção de invasões.
  - Por exemplo, o comando com a opção -sS (TCP SYN scan) requer a execução do nmap de forma privilegiada, pois esse tipo de varredura requer privilégios de socket bruto/pacote bruto.
  - Por outro lado, o comando com a opção -sT (varredura de conexão TCP) não requer sockets brutos e -nmap pode ser executado sem necessidade de privilégios.

Funcionamento do nmap

- O comportamento padrão do Nmap executa uma varredura de porta usando uma lista padrão de portas comumente usadas.
- Para cada uma das portas, ele retorna informações sobre o estado da porta e o serviço que está sendo executado nessa porta.
- Neste ponto, o Nmap categoriza as portas nos seguintes estados:
  - **Aberto**: um serviço está escutando conexões nesta porta.
  - **Fechado**: não há nenhum serviço em execução nesta porta.
  - **Filtrado**: nenhum pacote foi recebido e o estado não pôde ser estabelecido.
  - **Não filtrado**: os pacotes foram recebidos, mas não foi possível estabelecer um estado.

Escaneamento de Portas com Python-nmap

- É uma ferramenta cuja principal funcionalidade é descobrir quais portas ou serviços estão abertos para escuta em um host específico.
- É uma ferramenta para administradores de sistema ou consultores de segurança de computadores quando se trata de automatizar processos de teste de penetração e solução de problemas de rede.
- Oferece a possibilidade de saber qual versão de um determinado serviço, como SSH ou FTP, está sendo utilizada pela máquina alvo.
- Permite executar scripts avançados graças ao **Nmap Scripting Engine** (**NSE**) para automatizar diferentes tipos de ataques ou detectar serviços vulneráveis na máquina alvo.

Escaneamento de Portas com Python-nmap

- Com os comandos a seguir, invocamos o interpretador Python para revisar os vários métodos e funções que python-nmap tem a oferecer:

```
>>> import nmap
>>> nmap.__version__
'0.7.1’
>>> dir(nmap)
['ET', 'PortScanner', 'PortScannerAsync', 'PortScannerError',
'PortScannerHostDict', 'PortScannerTimeout', 'PortScannerYield',
'Process', '__author__', '__builtins__', '__cached__', '__doc__',
'__file__', '__last_modification__', '__loader__', '__name__',
'__package__', '__path__', '__spec__', '__version__',
'convert_nmap_output_to_encoding', 'csv', 'io', 'nmap', 'os', 're',
'shlex', 'subprocess', 'sys']
```

Escaneamento de Portas com Python-nmap

- Depois de verificarmos a instalação, podemos começar a escanear um host específico.
- Precisamos instanciar um objeto da classe PortScanner para podermos acessar o método scan().
- Uma boa prática para entender como funciona um processo, método ou objeto é utilizar o método dir() para descobrir os métodos disponíveis nesta classe:

```
>>> port_scan = nmap.PortScanner()
>>> dir(port_scan)
['_PortScanner__process', '__class__', '__delattr__', '__dict__', '__dir__', '__doc__',
'__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__',
'__init__', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__',
'__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__',
'__subclasshook__', '__weakref__', '_nmap_last_output', '_nmap_path',
'_nmap_subversion_number', '_nmap_version_number', '_scan_result', 'all_hosts',
'analyse_nmap_xml_scan', 'command_line', 'csv', 'get_nmap_last_output', 'has_host',
'listscan', 'nmap_version', 'scan', 'scaninfo', 'scanstats']
```

Escaneamento de Portas com Python-nmap

- Se executarmos o comando help(port_scan.scan), podemos ver que o método scan da classe PortScanner recebe três argumentos, o(s) host(s), as portas e os argumentos relacionados ao tipo de digitalização:

```
>>> help(port_scan.scan)
Help on method scan in module nmap.nmap:
scan(hosts='127.0.0.1', ports=None, arguments='-sV', sudo=False) method of
nmap.nmap.PortScanner instance
    Scan given hosts
    May raise PortScannerError exception if nmap output was not xml
    Test existance of the following key to know if something went wrong :
                                                        ['nmap']['scaninfo']['error’]
    If not present, everything was ok.
    :param hosts: string for hosts as nmap use it 'scanme.nmap.org' or '198.116.0-255.1-127’
                                                                  or '216.163.128.20/20’
    :param ports: string for ports as nmap use it '22,53,110,143-4564’
    :param arguments: string of arguments for nmap '-sU -sX -sC’
    :param sudo: launch nmap with sudo if True
    :returns: scan_result as dictionary
```

Escaneamento de Portas com Python-nmap

- Vamos executar um primeiro scan com o método scan('ip', 'ports'), onde o primeiro parâmetro é o endereço IP, o segundo é uma lista de portas e o terceiro, que é opcional,são as opções de escaneamento.
- No exemplo a seguir, uma varredura é executada no domínio scanme.nmap.org em portas no intervalo 22-443.
- Com o argumento -sV, estamos executando o nmap para detectar serviços e versões ao invocar a varredura:

```
>>> portScanner = nmap.PortScanner()
>>> results = portScanner.scan('scanme.nmap.org', '22-443','-sV’)
>>> results
{'nmap': {'command_line': 'nmap -oX - -p 22-443 -sV scanme.nmap.org', 'scaninfo':
{'tcp’: ... 'scan': {'45.33.32.156': {'hostnames’: ...
 'tcp': {22: {'state': 'open’... 80: {'state': 'open’...
'(Ubuntu)', 'conf': '10', 'cpe': 'cpe:/a:apache:http_server:2.4.7'}}}}}
```

Note: IP, SO, portas 22 e 80 abertas.

Extraindo Informações com nmap

- O Nmap fornece funções para extrair informações com mais eficiência.
- Por exemplo, podemos obter informações sobre nomes de host, endereços IP, resultados de varredura, protocolos e status de host:


```
>>> portScanner.all_hosts() ['45.33.32.156’]
>>> portScanner.scaninfo()
{'tcp': {'method': 'connect', 'services': '22-443’}}
>>> portScanner['45.33.32.156'].all_protocols()
['tcp’]
>>> portScanner['45.33.32.156'].hostnames()
[{'name': 'scanme.nmap.org', 'type': 'user'}, {'name': 'scanme.nmap.org', 'type': 'PTR’}]
>>> portScanner['45.33.32.156'].state()
'up'
```

Extraindo Informações com nmap

- O método command_line(), permite ver qual comando nmap foi executado:

```
>>> portScanner.command_line()
‘nmap -oX - -p 22-443 -sV scanme.nmap.org’
```

Extraindo Informações com nmap

- O Nmap fornece uma opção --open para exibir portas abertas:

```
>>> portScanner.scan('scanme.nmap.org','21,22,80,443','-v --open’)
{'nmap': {'command_line': 'nmap -oX - -p 21,22,80,443 -v --open scanme.nmap.org',
'scaninfo': {'tcp': {'method': 'connect', 'services': '21-22,80,443'}}, 'scanstats':
{'timestr': 'Sun Jan 15 23:36:01 2023', 'elapsed': '0.63', 'uphosts': '1',
'downhosts': '0', 'totalhosts': '1'}}, 'scan': {'45.33.32.156': {'hostnames':
[{'name': 'scanme.nmap.org', 'type': 'user'}, {'name': 'scanme.nmap.org', 'type':
'PTR'}], 'addresses': {'ipv4': '45.33.32.156'}, 'vendor': {}, 'status': {'state':
'up', 'reason': 'syn-ack'}, 'tcp': {22: {'state': 'open', 'reason': 'syn-ack',
'name': 'ssh', 'product': '', 'version': '', 'extrainfo': '', 'conf': '3', 'cpe':
''}, 80: {'state': 'open', 'reason': 'syn-ack', 'name': 'http', 'product': '',
'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}}}}}
```

Extraindo Informações com nmap

- Também poderíamos obter todos esses dados em um formato mais legível através do método csv().

```
>>> portScanner.csv()
'host;hostname;hostname_type;protocol;port;name;state;product;extrai
nfo;reason;version;conf;cpe\r\n45.33.32.156;scanme.nmap.org;user;tcp
;22;ssh;open;;;syn-
ack;;3;\r\n45.33.32.156;scanme.nmap.org;PTR;tcp;22;ssh;open;;;syn-
ack;;3;\r\n45.33.32.156;scanme.nmap.org;user;tcp;80;http;open;;;syn-
ack;;3;\r\n45.33.32.156;scanme.nmap.org;PTR;tcp;80;http;open;;;syn-
ack;;3;\r\n'
```

Extraindo Informações com nmap

- O código a seguir é para realizar uma varredura com python-nmap com as seguintes condições nos argumentos:
  - Lista de portas: 21, 22, 23, 25, 80
  - Opção -n (não aplicar uma resolução DNS)

```
import nmap
portScanner = nmap.PortScanner()
host_scan = input('Host scan: ‘)
portlist="21,22,23,25,80"
portScanner.scan(hosts=host_scan, arguments='-n -p'+portlist)
print(portScanner.command_line())
hosts_list = [(x, portScanner[x]['status']['state’])
                                    for x in portScanner.all_hosts()]
for host, status in hosts_list:
    print(host, status)
for protocol in portScanner[host].all_protocols():
    print('Protocol : %s' % protocol)
    listport = portScanner[host]['tcp'].keys()
    for port in listport:
        print('Port : %s State : %s’ %
              (port,portScanner[host][protocol][port]['state']))
```

Estamos usando o método all_protocols() para analisar cada protocolo encontrado nos resultados do portScanner.

Extraindo Informações com nmap

- Execução do código anterior:

```
$ python Nmap_port_scanner.py
Host scan: scanme.nmap.org
nmap -oX - -n -p21,22,23,25,80 scanme.nmap.org
45.33.32.156 up
Protocol : tcp
Port : 21 State : closed
Port : 22 State : open
Port : 23 State : closed
Port : 25 State : closed
Port : 80 State : open
```

Podemos ver o estado das portas que estamos analisando.

```
import nmap
import socket
print("-----------" * 6)
print(' Scanner with Nmap: ‘)
print("-----------" * 6)
domain = input ('Domain: ‘)
port_range = input ('Port range: ‘)
ip_address = socket.gethostbyname(domain)
print("-----------" * 6)
print(" Scanning the host with ip address: " + ip_address)
print("-----------" * 6)
nm = nmap.PortScanner()
nm.scan(ip_address, port_range)
for host in nm.all_hosts():
   print(" Host : %s (%s)" % (host,ip_address))
   print(" State : %s" % nm[host].state())
   for protocol in nm[host].all_protocols():
      print("-----------" * 6)
      print(" Protocols : %s" % protocol)
      lport = nm[host][protocol].keys()
      for port in lport:
         print(" Port : %s \t State : %s" %(port, nm[host][protocol][port]['state']))
```

Vamos realizar a varredura especificando um nome de domínio e indicando um intervalo de portas.

```
$ python PortScannerRange.py
---------------------------------------------------------------------
      Scanner with Nmap:
---------------------------------------------------------------------
Domain: scanme.nmap.org
Port range: 70-80
---------------------------------------------------------------------
      Scanning the host with ip address: 45.33.32.156
---------------------------------------------------------------------
   Host : 45.33.32.156 (45.33.32.156)
   State : up
---------------------------------------------------------------------
   Protocols : tcp
   Port : 70 State : closed
   Port : 71 State : closed
   Port : 72 State : closed
   Port : 73 State : closed
   Port : 74 State : closed
   Port : 75 State : closed
   Port : 76 State : closed
   Port : 77 State : closed
   Port : 78 State : closed
   Port : 79 State : closed
   Port : 80 State : open
```

Escaneamento Síncrono e Assíncrono

- No **modo síncrono**, toda vez que a varredura é feita em uma porta, ela precisa ser concluída para prosseguir para a próxima porta.
- No **modo assíncrono**, podemos realizar varreduras em diferentes portas simultaneamente e podemos definir uma função que será executada quando uma varredura for concluída em uma porta específica.
    - Dentro desta função podemos realizar operações adicionais como verificar o estado da porta ou lançar um script Nmap para um serviço específico (HTTP, FTP ou MySQL).

Escaneamento Síncrono

```
import optparse
import nmap
class NmapScanner:
    def __init__(self):
        self.portScanner = nmap.PortScanner()
    def nmapScan(self, ip_address, port):
        self.portScanner.scan(ip_address, port)
        self.state = self.portScanner[ip_address]['tcp'][int(port)]['state’]
        print(" [+] Executing command: ", self.portScanner.command_line())
        print(" [+] "+ ip_address + " tcp/" + port + " " + self.state)
```

- A classe NmapScanner permite varrer um endereço IP e uma lista de portas que são passadas como parâmetro.
- Realizamos um loop que processa cada porta enviada pelo parâmetro e chamamos o método nmapScan(ip, port) da classe NmapScanner.

Escaneamento Síncrono

```
def main():
   parser = optparse.OptionParser("usage%prog " + "--ip_address <target ip
                                                  address> --ports <target port>")
   parser.add_option('--ip_address', dest = 'ip_address', type = 'string', help =
                                              'Please, specify the target ip address.')
   parser.add_option('--ports', dest = 'ports', type = 'string', help = 'Please,
                                          specify the target port(s) separated by comma.')
   (options, args) = parser.parse_args()
   if (options.ip_address == None) | (options.ports == None):
      print('[-] You must specify a target ip_address and a target port(s).')
      exit(0)
   ip_address = options.ip_address
   ports = options.ports.split(',')
   for port in ports:
      NmapScanner().nmapScan(ip_address, port)
if __name__ == "__main__":
   main()
```

Escaneamento Síncrono

- Com a opção -h, podemos ver quais opções estão sendo aceitas pelo script:

```
$ python NmapScanner.py -h
Usage: usageNmapScanner.py --ip_address <target ip address> --ports <target port>
Options:
    -h, --help          show this help message and exit
    --ip_address=IP_ADDRESS
                        Please, specify the target ip address.
    --ports=PORTS       Please, specify the target port(s) separated by comma.
```

Escaneamento Síncrono

- Esta poderia ser a saída se executarmos o script anterior sobre o host 45.33.32.15, correspondente ao domínio scanme.nmap.org e às portas 21, 22, 23, 25, 80:

```
$ python NmapScanner.py --ip_address 45.33.32.156 --ports 21,22,23,25,80
[+] Executing command: nmap -oX - -p 21 -sV 45.33.32.156
[+] 45.33.32.156 tcp/21 closed
[+] Executing command: nmap -oX - -p 22 -sV 45.33.32.156
[+] 45.33.32.156 tcp/22 open
[+] Executing command: nmap -oX - -p 23 -sV 45.33.32.156
[+] 45.33.32.156 tcp/23 closed
[+] Executing command: nmap -oX - -p 25 -sV 45.33.32.156
[+] 45.33.32.156 tcp/25 closed
[+] Executing command: nmap -oX - -p 80 -sV 45.33.32.156
[+] 45.33.32.156 tcp/80 open
```

```
import optparse
import nmap
import csv
class NmapScannerCSV:
   def __init__(self):
      self.portScanner = nmap.PortScanner()
   def nmapScanCSV(self, host, ports):
      try:
         print("Checking ports "+ str(ports) +" ..........")
         self.portScanner.scan(host, arguments='-n -p'+ports)
         print("[*] Executing command: %s" % self.portScanner.command_line())
         print(self.portScanner.csv())
         print("Summary for host",host)
         with open('csv_file.csv', mode='w') as csv_file:
            csv_writer = csv.writer(csv_file, delimiter=',')
            csv_writer.writerow(['Host', 'Protocol', 'Port', 'State'])
            for x in self.portScanner.csv().split("\n")[1:-1]:
               splited_line = x.split(";")
               host = splited_line[0]
               protocol = splited_line[5]
               port = splited_line[4]
               state = splited_line[6]
               print("Protocol:",protocol,"Port:",port,"State:",state)
               csv_writer.writerow([host, protocol, port, state])
      except Exception as exception:
         print("Error to connect with " + host + " for port scanning" ,exception)
```

Fazendo a varredura de portas e gerando o resultado no formato csv.

- Estamos usando o método csv() do objeto portScanner, que retorna os resultados da varredura em um formato fácil para coletar as informações.
- A ideia é fazer com que cada linha CSV obtenha informações sobre o host, protocolo, porta e estado.

Escaneamento Síncrono

- Próxima parte do código: para gerenciar os argumentos do script:

```
def main():
   parser = optparse.OptionParser("usage%prog " + "--host <target host> --ports <target
                                                                        port>")
   parser.add_option('--host', dest = 'host', type = 'string', help = 'Please, specify the
                                                                    target host.’)
   parser.add_option('--ports', dest = 'ports', type = 'string', help = 'Please, specify the
                                                        target port(s) separated by comma.’)
   (options, args) = parser.parse_args()
   if (options.host == None) | (options.ports == None):
      print('[-] You must specify a target host and a target port(s).’)
      exit(0)
   host = options.host
   ports = options.ports
   NmapScannerCSV().nmapScanCSV(host,ports)
if __name__ == "__main__":
   main()
```

Estamos gerenciando os argumentos utilizados pelo script e chamando o método nmapScanCSV(host,ports), passando o endereço IP e a lista de portas como parâmetros.

Escaneamento Síncrono

- Na saída a seguir, podemos ver a execução do script anterior:

```
$ python NmapScannerCSV.py --host 45.33.32.156 --ports 21,22,23,25,80
Checking ports 21,22,23,25,80 ..........
[*] Executing command: nmap -oX - -n -p21,22,23,25,80 45.33.32.156
host;hostname;hostname_type;protocol;port;name;state;product;extrainfo;reason;versio
n;conf;cpe
45.33.32.156;;;tcp;21;ftp;closed;;;conn-refused;;3;
45.33.32.156;;;tcp;22;ssh;open;;;syn-ack;;3;
45.33.32.156;;;tcp;23;telnet;closed;;;conn-refused;;3;
45.33.32.156;;;tcp;25;smtp;closed;;;conn-refused;;3;
45.33.32.156;;;tcp;80;http;open;;;syn-ack;;3;
Summary for host 45.33.32.156
Protocol: ftp Port: 21 State: closed
Protocol: ssh Port: 22 State: open
Protocol: telnet Port: 23 State: closed
Protocol: smtp Port: 25 State: closed
Protocol: http Port: 80 State: open
```

```
import nmap, sys
command="nmap_operating_system.py <IP_address>"
if len(sys.argv) == 1:
   print(command)
   sys.exit()
host = sys.argv[1]
portScanner = nmap.PortScanner()
open_ports_dict = portScanner.scan(host, arguments="-O -v")
if open_ports_dict is not None:
   open_ports_dict = open_ports_dict.get("scan").get(host).get("tcp")
   print("Open port-->Service")
   port_list = open_ports_dict.keys()
   for port in port_list:
      print(port, "-->",open_ports_dict.get(port)['name’])
   print("\n--------------Operating System details--------------------\n")
   print("Details about the scanned host are: \t", portScanner[host]['osmatch'][0]['osclass'][0]['cpe’])
   print("Operating system family is: \t\t", portScanner[host]['osmatch'][0]['osclass'][0]['osfamily’])
   print("Type of OS is: \t\t\t\t", portScanner[host]['osmatch'][0]['osclass'][0]['type’])
   print("Generation of Operating System :\t", portScanner[host]['osmatch'][0]['osclass'][0]['osgen’])
   print("Operating System Vendor is:\t\t", portScanner[host]['osmatch'][0]['osclass'][0]['vendor’])
   print("Accuracy of detection is:\t\t", portScanner[host]['osmatch'][0]['osclass'][0]['accuracy'])
```

Usando o comando nmap para detectar portas abertas e obter informações sobre o sistema operacional.

- Estamos usando o método scan() do objeto portScanner, tendo como argumento o flag -O para detectar o sistema operacional ao executar a varredura.
- Para obter informações sobre detalhes do sistema operacional, precisamos acessar o dicionário portScanner[host] que contém essas informações na chave osmatch.

Escaneamento Síncrono

- Na saída a seguir, podemos ver a execução do script anterior:

```
$ sudo python nmap_operating_system.py 45.33.32.156
Open port-->Service
22 --> ssh
80 --> http
9929 --> nping-echo
31337 --> Elite

--------------Operating System details--------------------

Details about the scanned host are:      ['cpe:/o:linux:linux_kernel:5']
Operating system family is:              Linux
Type of OS is:                           general purpose
Generation of Operating System:          5.X
Operating System Vendor is:              Linux
Accuracy of detection is:                95
```

Escaneamento Assíncrono

- Embora a classe PortScanner seja a mais utilizada, também é possível executar a varredura em segundo plano enquanto o script executa outras atividades.
- Isto é conseguido com a classe PortScannerAsync:

```
>>> def nmap_callback(host,result):
... print(result)
...
>>> nma = nmap.PortScannerAsync()
>>> nma.scan('scanme.nmap.org',arguments="-Pn",callback=nmap_callback)
>>> nma.still_scanning()
True
>>> {'nmap': {'command_line': 'nmap -oX - -Pn 45.33.32.156', 'scaninfo': {'tcp': {'method': 'connect'}},
'scanstats': {'timestr': 'Wed Jan 11 22:39:28 2023', 'elapsed': '48.25', 'uphosts': '1', 'downhosts':
'0', 'totalhosts': '1'}}, 'scan': {'45.33.32.156': {'hostnames': [{'name': 'scanme.nmap.org', 'type':
'PTR'}], 'addresses': {'ipv4': '45.33.32.156'}, 'vendor': {}, 'status': {'state': 'up', 'reason': 'user-
set'}, 'tcp': {22: {'state': 'open', 'reason': 'syn-ack', 'name': 'ssh', 'product': '', 'version': '',
'extrainfo': '', 'conf': '3', 'cpe': ''}, 80: {'state': 'open', 'reason': 'syn-ack', 'name': 'http',
'product': '', 'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}, 9929: {'state': 'open',
'reason': 'syn-ack', 'name': 'nping-echo', 'product': '', 'version': '', 'extrainfo': '', 'conf': '3',
'cpe': ''}, 31337: {'state': 'open', 'reason': 'syn-ack', 'name': 'Elite', 'product': '', 'version': '',
'extrainfo': '', 'conf': '3', 'cpe': ''}}}}}
```

Escaneamento Assíncrono

- No exemplo, ao realizar a varredura, especificamos um parâmetro adicional de callback onde definimos a função que deverá ser executada ao final da varredura.

```
import nmap
portScannerAsync = nmap.PortScannerAsync()
def callback_result(host, scan_result):
    print(host, scan_result)
portScannerAsync.scan(hosts='scanme.nmap.org', arguments='-p'21', callback=callback_result)
portScannerAsync.scan(hosts='scanme.nmap.org', arguments='-p'22', callback=callback_result)
portScannerAsync.scan(hosts='scanme.nmap.org', arguments='-p'23', callback=callback_result)
portScannerAsync.scan(hosts='scanme.nmap.org', arguments='-p'80', callback=callback_result)
while portScannerAsync.still_scanning():
    print("Scanning >>>")
    portScannerAsync.wait(5)
```

- Definimos uma função callback_result(), que é executada quando o Nmap termina o processo de escaneamento com os argumentos especificados.
- O loop while definido é executado enquanto o processo de escaneamento ainda está em andamento.

Escaneamento Assíncrono

- Saída do script anterior:

Observe que os resultados de cada porta não são retornados necessariamente em ordem sequencial.

```
$ python PortScannerAsync.py
Scanning >>>
45.33.32.156 {'nmap': {'command_line': 'nmap -oX - -p 21 45.33.32.156', 'scaninfo': {'tcp': {'method':
'connect', 'services': '21'}}, 'scanstats': {'timestr': 'Thu Oct 1 23:11:55 2020', 'elapsed': '0.38',
'uphosts': '1', 'downhosts': '0', 'totalhosts': '1'}}, 'scan': {'45.33.32.156': {'hostnames': [{'name':
'scanme.nmap.org', 'type': 'PTR'}], 'addresses': {'ipv4': '45.33.32.156'}, 'vendor': {}, 'status':
{'state': 'up', 'reason': 'conn-refused'}, 'tcp': {21: {'state': 'closed', 'reason': 'conn-refused',
'name': 'ftp', 'product': '', 'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}}}}}
45.33.32.156 {'nmap': {'command_line': 'nmap -oX - -p 23 45.33.32.156', 'scaninfo': {'tcp': {'method':
'connect', 'services': '23'}}, 'scanstats': {'timestr': 'Thu Oct 1 23:11:55 2020', 'elapsed': '0.38',
'uphosts': '1', 'downhosts': '0', 'totalhosts': '1'}}, 'scan': {'45.33.32.156': {'hostnames': [{'name':
'scanme.nmap.org', 'type': 'PTR'}], 'addresses': {'ipv4': '45.33.32.156'}, 'vendor': {}, 'status':
{'state': 'up', 'reason': 'syn-ack'}, 'tcp': {23: {'state': 'closed', 'reason': 'conn-refused', 'name':
'telnet', 'product': '', 'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}}}}}
```

Escaneamento Assíncrono

- No exemplo a seguir, implementamos a classe NmapScannerAsync, que nos permite executar uma varredura assíncrona com um endereço IP e uma lista de portas que são passadas como parâmetros.

```
import nmap
import argparse
def callbackResult(host, scan_result):
   #print(host, scan_result)
   port_state = scan_result['scan'][host]['tcp’]
   print("Command line:"+ scan_result['nmap']['command_line’])
   for key, value in port_state.items():
       print('Port {0} --> {1}'.format(key, value))

Continua...
```

- Definimos um método callback_result() que é executado quando o Nmap termina o processo de digitalização.
- Esta função mostra informações sobre o comando executado e o estado de cada porta que estamos analisando.

Escaneamento Assíncrono

```
class NmapScannerAsync:
   def __init__(self):
      self.portScannerAsync = nmap.PortScannerAsync()
   def scanning(self):
      while self.portScannerAsync.still_scanning():
         print("Scanning >>>")
         self.portScannerAsync.wait(5)
   def nmapScanAsync(self, hostname, port):
      try:
         print("Checking port "+ port +" ..........")
         self.portScannerAsync.scan(hostname, arguments="-A -sV -p"+port
                                                    ,callback=callbackResult)
         self.scanning()
      except Exception as exception:
         print("Error to connect with " + hostname + " for port scanning",str(exception))
```

O método nmapScanAsync(self, hostname, port) verifica cada porta passada como parâmetro e chama a função callbackResult ao finalizar a varredura na porta.

Escaneamento Assíncrono

- O código a seguir representa o programa principal que solicita host e portas como parâmetros e chama a função nmapScanAsync(host,port) para cada porta que o usuário introduziu para verificação:

```
if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='Asynchronous Nmap scanner’)
    parser.add_argument("--host", dest="host", help="target IP / domain", required=True)
    parser.add_argument("-ports", dest="ports", help="Please, specify the target port(s)
                                    separated by comma[80,8080 by default]", default="80,8080")
    parsed_args = parser.parse_args()
    port_list = parsed_args.ports.split(',’)
    host = parsed_args.host
    for port in port_list:
        NmapScannerAsync().nmapScanAsync(host, port)
```

```
$ python NmapScannerAsync.py --host scanme.nmap.org -ports 21,22,23,25,80
Checking port 21 ..........
Checking port 22 ..........
Scanning >>>
Scanning >>>
Command line:nmap -oX - -A -sV -p22 45.33.32.156
Port 22 --> {'state': 'open', 'reason': 'syn-ack', 'name': 'ssh', 'product': 'OpenSSH',
'version': '6.6.1p1 Ubuntu 2ubuntu2.13', 'extrainfo': 'Ubuntu Linux; protocol 2.0', 'conf':
'10', 'cpe': 'cpe:/o:linux:linux_kernel', 'script': {'ssh-hostkey': '\n 1024
ac:00:a0:1a:82:ff:cc:55:99:dc:67:2b:34:97:6b:75 (DSA)\n 2048
20:3d:2d:44:62:2a:b0:5a:9d:b5:b3:05:14:c2:a6:b2 (RSA)\n 256
96:02:bb:5e:57:54:1c:4e:45:2f:56:4c:4a:24:b2:57 (ECDSA)\n 256
33:fa:91:0f:e0:e1:7b:1f:6d:05:a2:b0:f1:54:41:56 (EdDSA)'}}
Checking port 23 ..........
Checking port 25 ..........
Scanning >>>
Command line:nmap -oX - -A -sV -p25 45.33.32.156
Port 25 --> {'state': 'closed', 'reason': 'conn-refused', 'name': 'smtp', 'product': '',
'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ‘’}
Checking port 80 ..........
Scanning >>>
Command line:nmap -oX - -A -sV -p80 45.33.32.156
Port 80 --> {'state': 'open', 'reason': 'syn-ack', 'name': 'http', 'product': 'Apache
httpd', 'version': '2.4.7', 'extrainfo': '(Ubuntu)', 'conf': '10', 'cpe':
'cpe:/a:apache:http_server:2.4.7', 'script': {'http-server-header': 'Apache/2.4.7 (Ubuntu)',
'http-title': 'Go ahead and ScanMe!'}}
```

Escaneamento Assíncrono

Observações

- Como resultado da execução anterior, podemos observar que o processo analisou as portas que foram passadas por parâmetro e para cada porta escaneada foram apresentadas informações sobre o comando executado e o resultado em formato de dicionário.
- Por exemplo, foi retornado que as portas 22 e 80 estão abertas, e, na propriedade *extrainfo* retornada no dicionário, pode-se ver informações relacionadas ao servidor que está executando o serviço em cada porta.

Escaneamento Assíncrono

- Além das classes PortScanner e PortScannerAsync, existe outra classe que permite executar scans com Nmap, neste caso de forma progressiva.
- A classe PortScannerYield fornece a capacidade de executar a varredura do Nmap e retornar cada resultado que a ferramenta gera.
- Isso pode ser útil ao analisar um ambiente de rede completo e você não deseja esperar até que a varredura termine para ver os resultados, mas sim vê-los progressivamente à medida que o Nmap gera informações.

Escaneamento Assíncrono

Exemplo:

```
>>> nmy = nmap.PortScannerYield()
>>> for progress in nmy.scan('scanme.nmap.org',arguments="-Pn"):
...     print(progress)
...,
('45.33.32.156', {'nmap': {'command_line': 'nmap -oX - -Pn 45.33.32.156', 'scaninfo': {'tcp':
{'method': 'connect'}}, 'scanstats': {'timestr': 'Wed Jan 11 22:51:22 2023', 'elapsed': '41.75',
'uphosts': '1', 'downhosts': '0', 'totalhosts': '1'}}, 'scan': {'45.33.32.156': {'hostnames':
[{'name': 'scanme.nmap.org', 'type': 'PTR'}], 'addresses': {'ipv4': '45.33.32.156'}, 'vendor': {},
'status': {'state': 'up', 'reason': 'user-set'}, 'tcp': {22: {'state': 'open', 'reason': 'syn-
ack', 'name': 'ssh', 'product': '', 'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}, 80:
{'state': 'open', 'reason': 'syn-ack', 'name': 'http', 'product': '', 'version': '', 'extrainfo':
'', 'conf': '3', 'cpe': ''}, 9929: {'state': 'open', 'reason': 'syn-ack', 'name': 'nping-echo',
'product': '', 'version': '', 'extrainfo': '', 'conf': '3', 'cpe': ''}, 31337: {'state': 'open',
'reason': 'syn-ack', 'name': 'Elite', 'product': '', 'version': '', 'extrainfo': '', 'conf': '3',
'cpe': ''}}}}})
```

Praticar!!