

Programação Distribuída Com redes usando Linux e Python



Aluno: Ian Costa dos Santos

Turma: 24E2 4

1. Considere o encapsulamento de pacote e que você irá enviar a string "abcde" na Internet (TCP/IP) por meio de um socket e descreva:
 - O conteúdo do pacote enviado ao chegar na camada física de seu hospedeiro com todos os cabeçalhos e indicando o que é o segmento, o datagrama e o quadro;
 - O conteúdo do mesmo pacote assumindo o modelo de referência ISO/OSI.

R:

Modelo TCP/IP:

- Ao enviar dados pela Internet usando TCP/IP, os dados são encapsulados em diferentes camadas:
- Camada de Aplicação: A string "abcde" é enviada pela aplicação. O TCP/IP encapsula esses dados em um segmento TCP.
- Camada de Transporte: O segmento TCP é então encapsulado em um datagrama IP.
- Camada de Rede: O datagrama IP é encapsulado em um quadro de dados.
- Camada Física: O quadro de dados é então transmitido fisicamente pelo meio de transmissão, como fios ou ondas de rádio.

```
lansantos@lansantos:~/Desktop$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=55 time=8.68 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=55 time=8.49 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=55 time=9.44 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=55 time=8.34 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=55 time=8.29 ms
64 bytes from 8.8.8.8: icmp_seq=6 ttl=55 time=8.10 ms
64 bytes from 8.8.8.8: icmp_seq=7 ttl=55 time=8.03 ms
64 bytes from 8.8.8.8: icmp_seq=8 ttl=55 time=8.38 ms
64 bytes from 8.8.8.8: icmp_seq=9 ttl=55 time=8.32 ms
64 bytes from 8.8.8.8: icmp_seq=10 ttl=55 time=8.32 ms
^C
--- 8.8.8.8 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9028ms
rtt min/avg/max/mdev = 8.029/8.438/9.441/0.375 ms
```

Modelo OSI:

- Camada de Aplicação: Os dados são enviados pela aplicação.
- Camada de Apresentação: Os dados são formatados para transmissão, se necessário.
- Camada de Sessão: Estabelece, gerencia e encerra conexões entre aplicativos.
- Camada de Transporte: Os dados são encapsulados em segmentos.
- Camada de Rede: Os segmentos são encapsulados em datagramas.
- Camada de Enlace de Dados: Os datagramas são encapsulados em quadros.
- Camada Física: Os quadros são transmitidos fisicamente.

```
iansantos@iansantos:~/Desktop$ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  _gateway (10.0.2.2)  0.462 ms  0.260 ms  3.899 ms
 2  * * *
 3  * * *
 4  * * *
 5  * * *
 6  * * *
 7  * * *
 8  * * *
 9  * * *
10  * * *
11  * * *
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
```

Assim, ao enviar a string "abcde" pela Internet, ela é encapsulada em diferentes camadas do modelo TCP/IP e do modelo OSI, garantindo que os dados sejam transmitidos de forma confiável e eficiente.

2. Suponha um cenário que você perceba que a conexão do seu hospedeiro com a Internet foi perdida. A perda da conexão ocorreu devido a uma falha no ISP, porém, sem saber dessa informação, você realiza uma verificação

do seu hospedeiro para garantir que o problema não ocorreu com ele. Um dos passos da verificação é checar a configuração da tabela de rotas.

- Execute o comando para visualizar o IPv4 e IPv6 e indique o endereço IPv4 e máscara de subrede.
- Execute o comando para visualizar a tabela de rotas e indique qual rota deveria estar faltando, ou deveria estar mal configurada, para que a conexão com a Internet fosse perdida.
- A partir da máscara de subrede, indique a rota correspondente à subrede.

R:

a) Endereço IPv4: 10.0.2.15

Endereço IPV6: fe80::7ae1:b728:8b79:5aa6/64

Máscara de sub-rede: /24 (corresponde a 255.255.255.0)

```
iainsantos@iainsantos:~/Desktop$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:f2:9c:9e brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute enp0s3
        valid_lft 86094sec preferred_lft 86094sec
    inet6 fe80::7ae1:b728:8b79:5aa6/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:73:c8:2e:81 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
        valid_lft forever preferred_lft forever
```

b) Para que a conexão com a Internet seja perdida, a rota que deveria estar faltando ou mal configurada é a rota padrão (default route). Esta rota direciona o tráfego que não pertence à rede local para o gateway, que depois direciona para a Internet.

```
iansantos@iansantos:~/Desktop$ ip route show
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
169.254.0.0/16 dev enp0s3 scope link metric 1000
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
iansantos@iansantos:~/Desktop$
```

c) A rota correspondente à subrede 10.0.2.0/24 é: “10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100”. Isso indica que a subrede 10.0.2.0/24 está acessível através do dispositivo de rede enp0s3 com endereço IP de origem 10.0.2.15.

3. Forneça os endereços IP dos seguintes nomes de hospedeiros (nomes de rede), utilizado pelos menos dois comandos Linux diferentes:

www.google.com, www.yahoo.com, localhost.

- **Remova todas as rotas da tabela de roteamento do seu hospedeiro e mostre que a conexão para o Gateway Default e o domínio **www.google.com** foi perdida.**
- **A seguir, adicione uma rota para sub rede local e mostre que a conexão com o Gateway Default foi reestabelecida, porém a conexão para **www.google.com** ainda está perdida.**
- **Por fim, escolha uma sub rede arbitrária que contenha o endereço IP de **www.google.com**, adicione uma rota para esta sub rede e mostre que a conexão com **www.google.com** foi reestabelecida.**

R:

Obtendo endereços IP dos domínios:

Ping:

```

iansantos@iansantos:~/Desktop$ ping -c 1 www.google.com
ping -c 1 www.yahoo.com
ping -c 1 localhost
PING www.google.com (142.250.79.164) 56(84) bytes of data.
64 bytes from eze04s15-in-f4.1e100.net (142.250.79.164): icmp_seq=1 ttl=243 time=14.9 ms

--- www.google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 14.907/14.907/14.907/0.000 ms
PING me-ycpi-cf-www.g06.yahoodns.net (200.152.162.137) 56(84) bytes of data.
64 bytes from e4-ha.ycpi.bra.yahoo.com (200.152.162.137): icmp_seq=1 ttl=243 time=8.93 ms

--- me-ycpi-cf-www.g06.yahoodns.net ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 8.932/8.932/8.932/0.000 ms
PING localhost (127.0.0.1) 56(84) bytes of data.
64 bytes from localhost (127.0.0.1): icmp_seq=1 ttl=64 time=0.017 ms

--- localhost ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.017/0.017/0.017/0.000 ms

```

Nslookup:

```

iansantos@iansantos:~/Desktop$ nslookup www.google.com
nslookup www.yahoo.com
nslookup localhost
Server:      127.0.0.53
Address:     127.0.0.53#53

Non-authoritative answer:
Name:   www.google.com
Address: 142.250.79.164
Name:   www.google.com
Address: 2800:3f0:4001:821::2004

Server:      127.0.0.53
Address:     127.0.0.53#53

Non-authoritative answer:
www.yahoo.com canonical name = me-ycpi-cf-www.g06.yahoodns.net.
Name:   me-ycpi-cf-www.g06.yahoodns.net
Address: 200.152.162.136
Terminal me-ycpi-cf-www.g06.yahoodns.net
200.152.162.137
Name:   me-ycpi-cf-www.g06.yahoodns.net
Address: 200.152.162.143
Name:   me-ycpi-cf-www.g06.yahoodns.net
Address: 200.152.162.189
Name:   me-ycpi-cf-www.g06.yahoodns.net
Address: 2804:1bc:114::2000
Name:   me-ycpi-cf-www.g06.yahoodns.net
Address: 2804:1bc:114::2002
Name:   me-ycpi-cf-www.g06.yahoodns.net
Address: 2804:1bc:114::2006
Name:   me-ycpi-cf-www.g06.yahoodns.net
Address: 2804:1bc:114::2005

Server:      127.0.0.53
Address:     127.0.0.53#53

Name:   localhost
Address: 127.0.0.1
Name:   localhost
Address: ::1

```

Remover todas as rotas da tabela de roteamento:

```
iansantos@iansantos:~/Desktop$ ip route
default via 10.0.2.2 dev enp0s3 proto dhcp metric 100
10.0.2.0/24 dev enp0s3 proto kernel scope link src 10.0.2.15 metric 100
169.254.0.0/16 dev enp0s3 scope link metric 1000
172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1 linkdown
iansantos@iansantos:~/Desktop$ sudo ip route flush table main
[sudo] password for iansantos:
Sorry, try again.
[sudo] password for iansantos:
iansantos@iansantos:~/Desktop$ ip route
iansantos@iansantos:~/Desktop$
```

Tentando conexão com google e gateway padrão:

```
iansantos@iansantos:~/Desktop$ ip route
iansantos@iansantos:~/Desktop$ ping -c 1 10.0.2.2
ping: connect: Network is unreachable
iansantos@iansantos:~/Desktop$ ping -c 1 www.google.com
ping: www.google.com: Temporary failure in name resolution
iansantos@iansantos:~/Desktop$
```

Adicionar uma rota para a sub-rede local

```
iansantos@iansantos:~/Desktop$ sudo ip route add 10.0.2.0/24 dev enp0s3
iansantos@iansantos:~/Desktop$ ip route
10.0.2.0/24 dev enp0s3 scope link
iansantos@iansantos:~/Desktop$
```

Verificando conexão:

```
iansantos@iansantos:~/Desktop$ ping -c 1 10.0.2.2
PING 10.0.2.2 (10.0.2.2) 56(84) bytes of data.
64 bytes from 10.0.2.2: icmp_seq=1 ttl=64 time=1.12 ms

--- 10.0.2.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.119/1.119/1.119/0.000 ms
iansantos@iansantos:~/Desktop$ ping -c 1 www.google.com
ping: www.google.com: Temporary failure in name resolution
iansantos@iansantos:~/Desktop$ ~
```


Restabelecendo conexão com google.com

```
iansantos@iansantos:~/Desktop$ sudo ip route add 172.217.29.0/24 via 10.0.2.2 dev enp0s3
iansantos@iansantos:~/Desktop$
```

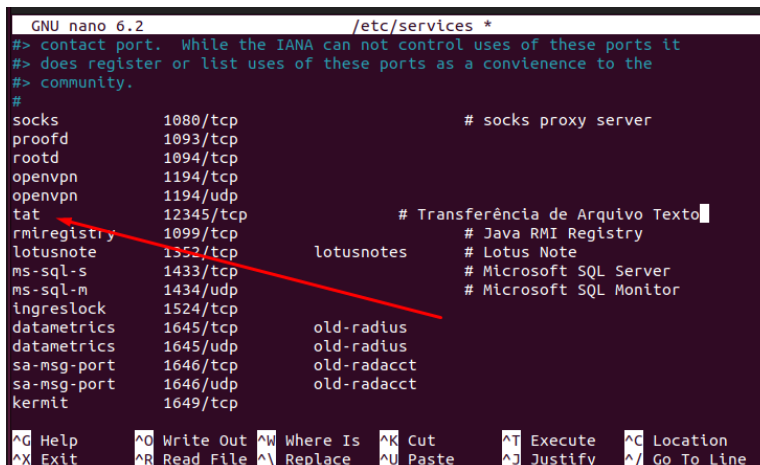
Verificando conexão com google.com

```
iansantos@iansantos:~/Desktop$ ping -c 1 www.google.com
PING www.google.com (142.250.79.164) 56(84) bytes of data:
64 bytes from rio07s12-in-f4.1e100.net (142.250.79.164): icmp_seq=1 ttl=243 time=16.0 ms

--- www.google.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 15.990/15.990/15.990/0.000 ms
iansantos@iansantos:~/Desktop$ ~
```

4. Considere que você criou um novo serviço de transferência de arquivos chamado Transferência de Arquivo Texto (TAT), o qual executa na porta 12345 do seu hospedeiro. Como tal hospedeiro é utilizado com muita frequência, associe o número de porta 12345 ao nome tat. Para testar sua atribuição de porta, execute o comando `lsf -i:tat` e mostre que o comando retorna sem nenhum erro.

R: Adicionando Tat:



The screenshot shows the `/etc/services` file being edited with nano. A red arrow points to the line `tat 12345/tcp`, which has been added to the file. The file contains a list of services and their corresponding ports and protocols. The new entry is placed between `openvpn` and `rmiregistry`.

```
GNU nano 6.2 /etc/services *
#> contact port. While the IANA can not control uses of these ports it
#> does register or list uses of these ports as a convenience to the
#> community.
#
socks 1080/tcp # socks proxy server
proofd 1093/tcp
rootd 1094/tcp
openvpn 1194/tcp
openvpn 1194/udp
tat 12345/tcp # Transferência de Arquivo Texto
rmiregistry 1099/tcp # Java RMI Registry
lotusnote 1352/tcp lotusnotes # Lotus Note
ms-sql-s 1433/tcp # Microsoft SQL Server
ms-sql-m 1434/udp # Microsoft SQL Monitor
ingreslock 1524/tcp
datametrics 1645/tcp old-radius
datametrics 1645/udp old-radius
sa-msg-port 1646/tcp old-radacct
sa-msg-port 1646/udp old-radacct
kermit 1649/tcp

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^_ Go To Line
```


Executando comando:

```
iansantos@iansantos:~/Desktop$ lsof -i:tat  
iansantos@iansantos:~/Desktop$
```

5. Descreva o funcionamento do protocolo DHCP e responda a seguinte pergunta:

- **Como um novo hospedeiro em uma LAN consegue obter endereço IP e máscara de subrede mesmo não conhecendo o endereço de nenhum outro nó e não conhecendo a subrede?**

R:

Solicitação DHCP: Quando um novo hospedeiro se conecta à rede, ele envia uma solicitação DHCP por broadcast na rede local, indicando sua necessidade de um endereço IP.

Oferta DHCP: O servidor DHCP da rede recebe a solicitação e oferece um endereço IP disponível do seu pool de endereços configurados para distribuição.

Aceitação da Oferta: O novo hospedeiro seleciona uma oferta DHCP e aceita, confirmando assim os parâmetros de configuração de rede oferecidos, incluindo o endereço IP e a máscara de sub-rede.

Configuração: O servidor DHCP reserva o endereço IP selecionado e configura o hospedeiro com os parâmetros de rede fornecidos na oferta DHCP, incluindo o endereço IP e a máscara de sub-rede.

Um novo hospedeiro em uma LAN obtém endereço IP e máscara de sub-rede através do protocolo DHCP, que atribui automaticamente esses parâmetros ao hospedeiro

quando ele se conecta à rede, sem que seja necessário conhecer outros nós ou a sub-rede previamente.

6. Descreva a diferença entre um hospedeiro linux configurado como roteador e um hospedeiro linux não configurado como roteador.meio

R:

Um hospedeiro Linux configurado como roteador possui a capacidade de encaminhar pacotes de dados entre diferentes redes, utilizando tabelas de roteamento para determinar os melhores caminhos. Isso envolve ter duas ou mais interfaces de rede configuradas com endereços IP pertencentes a diferentes sub-redes e a capacidade de realizar funções como NAT (Network Address Translation) para permitir o acesso à Internet para dispositivos em uma rede privada. Por outro lado, um hospedeiro Linux não configurado como roteador não possui essa capacidade de encaminhamento de pacotes entre redes, limitando-se à comunicação dentro da mesma rede local.

7. Assuma que você necessita logar em um hospedeiro que não possui interface gráfica. Tal hospedeiro necessita baixar a página inicial do Google em espanhol (es) e as únicas ferramentas disponíveis para realizar esta tarefa são o telnet e o curl. Para simular tal situação, realize o login remoto no localhost com o usuário alice e baixe com curl o conteúdo da URL www.google.com para o arquivo google_es.html.

R:

Acessando usuário Alice:

```

iansantos@iansantos:~/Desktop$ ssh alice@localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ED25519 key fingerprint is SHA256:1Gl26/HRwqXeivLaTLlCbi5K3p37NDHNE+YMPCECQGs.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'localhost' (ED25519) to the list of known hosts.
alice@localhost's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-35-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

130 updates can be applied immediately.
73 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

1 additional security update can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

```

Baixando conteudo:

```

alice@iansantos:~$ curl -o google_es.html https://www.google.com/?hl=es
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 20506    0 20506    0     0   70752      0 --:--:-- --:--:-- --:--:-- 70955
alice@iansantos:~$ ~

```

Conteudo gerado:

```
GNU nano 6.2 google_es.html
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="es">
var h=this||self;function l(){return window.google!==void 0&&window.google.koPI>
function t(a,b,c,d,k){var e="";b.search("&ei=")===-1&&(e="&ei="+p(d),b.search(">
document.documentElement.addEventListener("submit",function(b){var a;if(a=b.tar>
</style><style>body,td,a,p,.h{font-family:arial,sans-serif}body{margin:0;overfl>
var h=this||self;var k,l=(k=h.mei)!=null?k:1,n,p=(n=h.sdo)!=null?n:!0,q=0,r,t=g>
"&bver="+b(t.bv);var f=a.lineNumber;f!==void 0&&(c+="&line="+f);var g=a.fileNam>
if (!iesg){document.f&&document.f.q.focus();document.gbqf&&document.gbqf.q.focu>
}
})();</script><div id="mngb"><div id=gbar><nobr><b class=gb1>Bosqueda</b> <a cl>
else top.location='/doodles/';}}})();</script><input value="AL9hbdgAAAAZmsVUYd>
if(a&&b&&(a!=google.cdo.width||b!=google.cdo.height)){var e=google,f=e.log,g="/>
var e=this||self,f=function(a){return a};var g;var h=function(a){this.g=a};h.pr>
function l(a,b){a.src=b instanceof h&&b.constructor===h?b.g:"type_error:Trusted>
setTimeout(function(){google&&google.tick&&google.timers&&google.timers.load&&g>
(function(){google.jl={bfl:0,dw:false,ine:false,ubm:false,uwp:true,vs:false};})>
var e=this||self;var g,h;a:{for(var k=["CLOSURE_FLAGS"],l=e,n=0;n<k.length;n++)>
a.closest("[data-ved]"))?G(f)||""":"";f=f||"";if(a.hasAttribute("jsname"))a=a.ge>
```

8. Execute as seguintes tarefas com o usuário alicé e executando comandos ssh remotamente em dois terminais. No primeiro terminal, execute um servidor TCP com netcat no localhost na porta 12345. No segundo terminal, exiba a conexão do servidor TCP utilizando as ferramentas netstat e lsof.

R:

Abrindo porta no primeiro terminal:

```
alice@iansantos: ~ x alice@iansantos: ~ x
alice@iansantos:~$ nc -l 12345
```

Exibindo conexão no segundo terminal:

```
alice@iansamtos:~$ netstat -tuln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:36375         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.53:53          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306         0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:12345          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:22             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:33060        0.0.0.0:*               LISTEN
tcp6       0      0 :::1:631               :::*                    LISTEN
tcp6       0      0 :::22                  :::*                    LISTEN
udp        0      0 0.0.0.0:631            0.0.0.0:*               *
udp        0      0 127.0.0.53:53          0.0.0.0:*               *
udp        0      0 0.0.0.0:48330          0.0.0.0:*               *
udp        0      0 0.0.0.0:5353           0.0.0.0:*               *
udp6       0      0 :::42093               :::*                    *
udp6       0      0 :::5353                :::*                    *
```

```
alice@iansamtos:~$ lsof -iTCP -n -P | grep LISTEN
nc          5044 alice      3u  IPv4  43751      0t0  TCP *:12345 (LISTEN)
```

9. Considere que você utilizou o comando netstat e verificou que o número de pacotes chegando no seu hospedeiro está muito maior do que o normal. Pela quantidade de dados, você suspeita que o tráfego pode ser HTTP ou uma conexão remota com telnet ou ssh. Descreva um comando tcpdump que seja capaz de capturar tais tipos de tráfego.

R:

Para capturar tráfego HTTP, conexões remotas com telnet ou SSH usando o comando tcpdump, podemos usar filtros para especificar os tipos de tráfego que deseja capturar.

Exemplo para HTTP:

```
alice@iansantos:~$ sudo tcpdump -i enp0s3 port 80
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

Exemplo Telnet:

```
alice@iansantos:~$ sudo tcpdump -i enp0s3 port telnet
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
```

10. Você desenvolveu um cliente UDP na porta 12345 e deseja testá-lo rapidamente para corrigir alguns erros. Execute um servidor UDP com netcat que seja capaz de se comunicar com o cliente UDP.

R:

Server:

Rodando Servidor:

```
iansantos@iansantos:~/Desktop$ nc -u -l 12345
Testando o cliente UDPola
```

Client:

```
iansantos@iansantos:~/Desktop$ python3 client-udp.py
Enviando b'Testando o cliente UDP' para ('localhost', 12345)
Aguardando resposta
Recebido: b'ola\n'
iansantos@iansantos:~/Desktop$ ~
```

```
GNU nano 6.2 client-udp.py *
import socket

server_address = ('localhost', 12345)
message = b'Testando o cliente UDP'

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

try:
    print(f'Enviando {message} para {server_address}')
    sent = sock.sendto(message, server_address)

    print('Aguardando resposta')
    data, server = sock.recvfrom(4096)
    print(f'Recebido: {data}')

finally:
    sock.close()
```

11. Considere que sua empresa possui um servidor que envia arquivos-texto automaticamente para um outro hospedeiro localizado em uma rede externa. O arquivo `schedule.txt` contém a agenda de envio, onde cada linha possui os seguintes campos separados por vírgula: nome do arquivo a ser enviado sem espaços em branco, data de envio no formato DD-MM-AAAA, e horário de envio no formato mm:ss. Para que um arquivo-texto seja enviado, basta que ele seja armazenado na pasta `send_files`, no diretório home do usuário bob, e que contenha uma entrada em `schedule.txt`.
- Implemente uma aplicação Cliente-Servidor para transferência de arquivo-texto em que o cliente transfere um arquivo-texto para o

Servidor, incluindo na mensagem a data e horário. Ao receber o arquivo, o Servidor salva-o no diretório send_files e cria uma entrada em schedule.txt para que o arquivo possa ser transferido para a rede externa.

- **A aplicação deve utilizar apenas a biblioteca socket do Python e o código do servidor e do cliente devem ser implementados em servidor_v0.txt e cliente_v0.txt, respectivamente.**
- **O arquivo armazenado na pasta send_files deve possuir exatamente o mesmo conteúdo do arquivo enviado pelo cliente.**

Obs.: Você não deve implementar o servidor de envio para rede externa, apenas a aplicação Cliente-Servidor.

R:

Servidor_v0.py:

```
1 import socket
2 import os
3
4 HOST = 'localhost'
5 PORT = 65432
6 SEND_FILES_DIR = '/home/bob/send_files'
7 SCHEDULE_FILE = 'schedule.txt'
8
9 def handle_client(conn, addr):
10     file_name_length = int.from_bytes(conn.recv(4), 'big')
11     file_name = conn.recv(file_name_length).decode()
12
13     date_length = int.from_bytes(conn.recv(4), 'big')
14     date = conn.recv(date_length).decode()
15
16     time_length = int.from_bytes(conn.recv(4), 'big')
17     time = conn.recv(time_length).decode()
18
19     file_size = int.from_bytes(conn.recv(4), 'big')
20     file_content = conn.recv(file_size).decode()
21
22     with open(os.path.join(SEND_FILES_DIR, file_name), 'w') as f:
23         f.write(file_content)
24
25     with open(SCHEDULE_FILE, 'a') as f:
26         f.write(f"{file_name},{date},{time}\n")
27
28     conn.close()
29
30 def main():
31     if not os.path.exists(SEND_FILES_DIR):
32         os.makedirs(SEND_FILES_DIR)
33
34     with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
35         s.bind((HOST, PORT))
36         s.listen()
37
38         while True:
39             conn, addr = s.accept()
40             handle_client(conn, addr)
41
42 if __name__ == "__main__":
43     main()
```

Rodando Servidor:

```
bob@iansantos: ~/Desktop x bob@iansantos: ~/Desktop x v
bob@iansantos:~/Desktop$ python3 server_v0.py
```

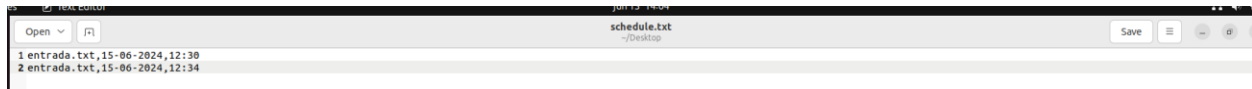
Cliente_v0.py:

```
1 import socket
2 import sys
3 import os
4
5 HOST = 'localhost'
6 PORT = 65432
7
8 def send_file(file_path, date, time):
9     file_name = os.path.basename(file_path)
10
11     with open(file_path, 'r') as f:
12         file_content = f.read()
13
14     with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
15         s.connect((HOST, PORT))
16
17         s.sendall(len(file_name).to_bytes(4, 'big'))
18         s.sendall(file_name.encode())
19
20         s.sendall(len(date).to_bytes(4, 'big'))
21         s.sendall(date.encode())
22
23         s.sendall(len(time).to_bytes(4, 'big'))
24         s.sendall(time.encode())
25
26         s.sendall(len(file_content).to_bytes(4, 'big'))
27         s.sendall(file_content.encode())
28
29     print(f"Arquivo {file_name} enviado com sucesso.")
30
31 if __name__ == "__main__":
32     if len(sys.argv) != 4:
33         print("Uso: python cliente_v0.py <caminho_arquivo> <data> <hora>")
34         sys.exit(1)
35
36     file_path = sys.argv[1]
37     date = sys.argv[2]
38     time = sys.argv[3]
39
40     if not os.path.isfile(file_path):
41         print(f"Erro: Arquivo {file_path} não encontrado.")
42         sys.exit(1)
43
44     send_file(file_path, date, time)
```

Rodando Client:

```
bob@iansantos:~/Desktop$ python3 cliente_v0.py entrada.txt 15-06-2024 12:30
Arquivo entrada.txt enviado com sucesso.
bob@iansantos:~/Desktop$ python3 cliente_v0.py entrada.txt 15-06-2024 12:34
Arquivo entrada.txt enviado com sucesso.
bob@iansantos:~/Desktop$
```

Resultado:



The screenshot shows a text editor window titled 'schedule.txt' with the following content:

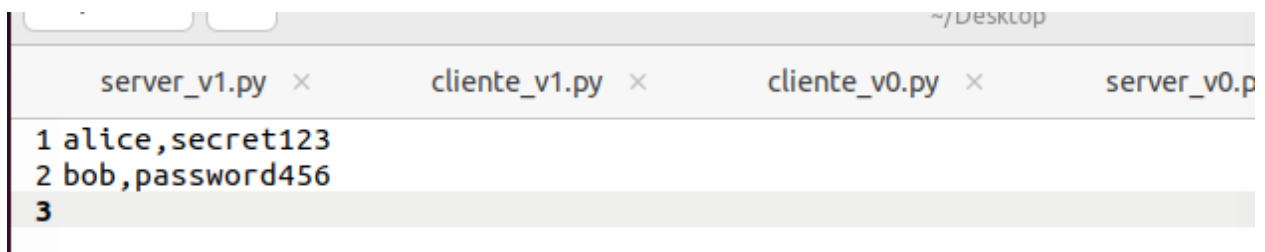
```
1 entrada.txt,15-06-2024,12:30
2 entrada.txt,15-06-2024,12:34
```

12. A partir do exercício anterior, implemente um serviço de autenticação na aplicação Cliente-Servidor nos arquivos servidor_v1.py e cliente_v1.py. O servidor possui um arquivo chamado users.txt em que cada linha contém o nome do usuário e a senha separados por vírgula. Antes de enviar o arquivo, o cliente deve enviar suas credenciais (login e senha) para o servidor autenticá-lo. O servidor deve procurar as credenciais do cliente em users.txt e enviar uma mensagem autorizando ou negando acesso ao cliente.

Obs.: A única biblioteca permitida é a socket do Python.

R:

Arquivo de usuarios:



The screenshot shows a text editor window with the following content:

```
1 alice,secret123
2 bob,password456
3
```

Servidor_v1.py

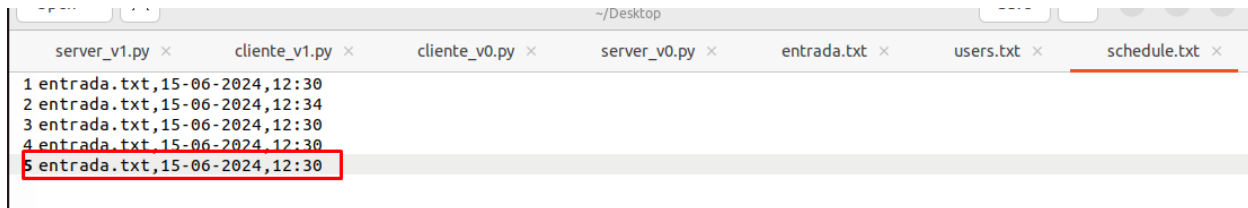
```
server_v1.py x cliente_v1.py x cliente_v0.py x server_v0.py x entrada.txt x users.txt
1 import socket
2 import os
3
4 HOST = 'localhost'
5 PORT = 65432
6 SEND_FILES_DIR = '/home/bob/send_files'
7 SCHEDULE_FILE = 'schedule.txt'
8 USERS_FILE = 'users.txt'
9
10 def authenticate(conn):
11     username_length = int.from_bytes(conn.recv(4), 'big')
12     username = conn.recv(username_length).decode()
13
14     password_length = int.from_bytes(conn.recv(4), 'big')
15     password = conn.recv(password_length).decode()
16
17     with open(USERS_FILE, 'r') as f:
18         for line in f:
19             user, passw = line.strip().split(',')
20             if user == username and passw == password:
21                 return True
22     return False
23
24 def handle_client(conn, addr):
25     if not authenticate(conn):
26         print(f"Autenticação falhou para {addr}")
27         conn.sendall(b'Authentication failed')
28         conn.close()
29         return
30
31     conn.sendall(b'Authentication successful')
32
33     file_name_length = int.from_bytes(conn.recv(4), 'big')
34     file_name = conn.recv(file_name_length).decode()
35
36     date_length = int.from_bytes(conn.recv(4), 'big')
37     date = conn.recv(date_length).decode()
38
39     time_length = int.from_bytes(conn.recv(4), 'big')
40     time = conn.recv(time_length).decode()
41
42     file_size = int.from_bytes(conn.recv(4), 'big')
43     file_content = conn.recv(file_size).decode()
44
45     file_size = int.from_bytes(conn.recv(4), 'big')
46     file_content = conn.recv(file_size).decode()
47
48     with open(os.path.join(SEND_FILES_DIR, file_name), 'w') as f:
49         f.write(file_content)
50
51     with open(SCHEDULE_FILE, 'a') as f:
52         f.write(f"{file_name},{date},{time}\n")
53
54     conn.close()
55
56 def main():
57     if not os.path.exists(SEND_FILES_DIR):
58         os.makedirs(SEND_FILES_DIR)
59
60     with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
61         s.bind((HOST, PORT))
62         s.listen()
63
64         while True:
65             conn, addr = s.accept()
66             handle_client(conn, addr)
67
68 if __name__ == "__main__":
69     main()
```

Cliente_v1.py:

```
server_v1.py × cliente_v1.py × cliente_v0.py × server_v0.py × entrada.txt ×
1 import socket
2 import sys
3 import os
4
5 HOST = 'localhost'
6 PORT = 65432
7
8 def send_credentials(s, username, password):
9     s.sendall(len(username).to_bytes(4, 'big'))
10    s.sendall(username.encode())
11
12    s.sendall(len(password).to_bytes(4, 'big'))
13    s.sendall(password.encode())
14
15 def send_file(file_path, date, time, username, password):
16     file_name = os.path.basename(file_path)
17
18     with open(file_path, 'r') as f:
19         file_content = f.read()
20
21     with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
22         s.connect((HOST, PORT))
23
24         send_credentials(s, username, password)
25         auth_message = s.recv(1024)
26         print(auth_message.decode())
27
28         s.sendall(len(file_name).to_bytes(4, 'big'))
29         s.sendall(file_name.encode())
30
31         s.sendall(len(date).to_bytes(4, 'big'))
32         s.sendall(date.encode())
33
34         s.sendall(len(time).to_bytes(4, 'big'))
35         s.sendall(time.encode())
36
37         s.sendall(len(file_content).to_bytes(4, 'big'))
38         s.sendall(file_content.encode())
39
40         print(f"Arquivo {file_name} enviado com sucesso.")
41
42 if __name__ == "__main__":
43     if len(sys.argv) != 6:
44         print("Uso: python cliente_v1.py <caminho_arquivo> <data> <hora> <username> <password>")
45
46
47     file_path = sys.argv[1]
48     date = sys.argv[2]
49     time = sys.argv[3]
50     username = sys.argv[4]
51     password = sys.argv[5]
52
53     if not os.path.isfile(file_path):
54         print(f"Erro: Arquivo {file_path} não encontrado.")
55         sys.exit(1)
56
57     send_file(file_path, date, time, username, password)
58
```

Resultado:

```
ConnectionRefusedError: [Errno 111] Connection refused
bob@iansantos:~/Desktop$ python3 cliente_v1.py entrada.txt 15-06-2024 12:30 alic
e secret123
Authentication successful
Arquivo entrada.txt enviado com sucesso.
bob@iansantos:~/Desktop$
```



Falhando na autenticação

```
bob@iansantos:~/Desktop$ python3 cliente_v1.py entrada.txt 15-06-2024 12:30 alic
e secret1
Authentication failed
```

13. Considere que sua empresa possui 10 servidores que armazenam na pasta `internet_data` arquivos HTML contendo dados tabulados obtidos de fontes na Internet. Para um usuário visualizar tais arquivos, ele baixa os arquivos de algum dos 10 servidores e os abre no navegador. Para otimizar tal processo, você foi encarregado de implementar um servidor http no arquivo `check_servers.py` rodando na porta 12345 e deixá-lo executando na pasta `internet_data`. Dessa forma, quando alguém quer acessar tal dado basta digitar no navegador `endereço_ip_servidor:12345/nome_arquivo.html`. Implemente o servidor HTTP utilizando apenas a biblioteca `socket` do python. Teste o Servidor HTTP com o navegador utilizando o arquivo `test.html` cujo conteúdo é descrito abaixo:

```
<html><body>
<h1>Dado Teste</h1>
<table>
```

```

<tr><td>titulo1</td><td>titulo2</td><td>titulo3</td></tr>
<tr><td>dado11</td><td>dado12</td><td>dado13</td></tr>
</table>
</body></html>

```

R:

Check-servers.py:

```

1 import socket
2 import os
3
4 BASE_DIR = 'internet_data'
5
6 def handle_request(client_socket, request):
7     request_lines = request.split(b'\r\n')
8     if len(request_lines) < 1:
9         return
10
11     request_line = request_lines[0].decode('utf-8')
12
13     parts = request_line.split()
14     if len(parts) < 2:
15         return
16
17     filename = parts[1]
18
19     file_path = os.path.join(BASE_DIR, filename[1:])
20
21     if os.path.exists(file_path):
22         with open(file_path, 'rb') as file:
23             content = file.read()
24
25         response = b'HTTP/1.1 200 OK\r\n\r\n' + content
26     else:
27         response = b'HTTP/1.1 404 Not Found\r\n\r\n<html><body><h1>404 Not Found</h1></body></html>'
28
29     client_socket.sendall(response)
30     client_socket.close()
31
32 def run_server():
33     host = 'localhost'
34     port = 12345
35
36     server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
37
38     try:
39         server_socket.bind((host, port))
40         server_socket.listen(5)

```



```
def run_server():
    host = 'localhost'
    port = 12345

    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    try:
        server_socket.bind((host, port))
        server_socket.listen(5)

        print(f'Servidor HTTP rodando em http://{host}:{port}')

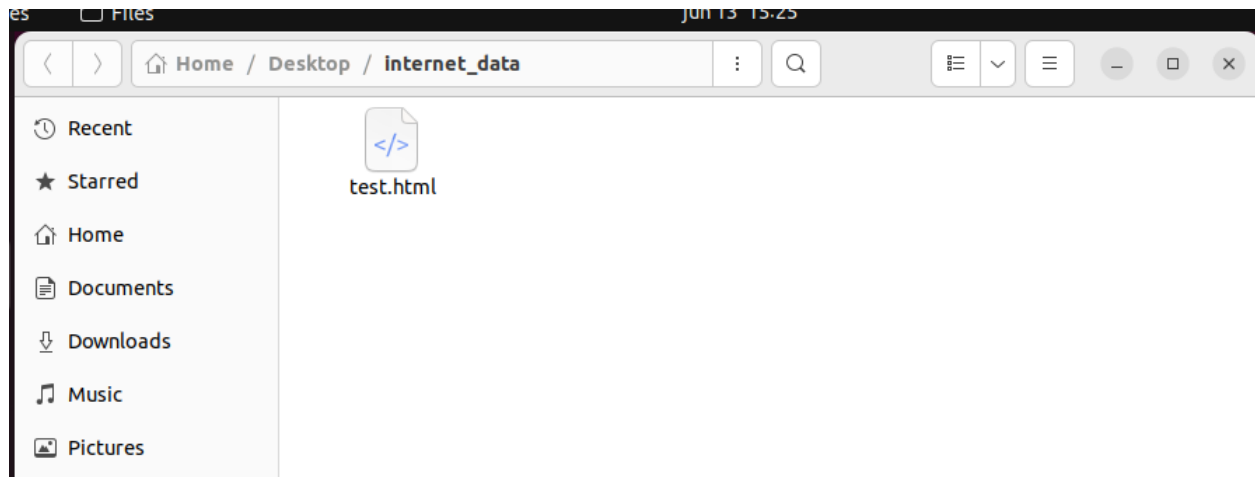
        while True:
            client_socket, addr = server_socket.accept()
            request = client_socket.recv(4096)

            handle_request(client_socket, request)

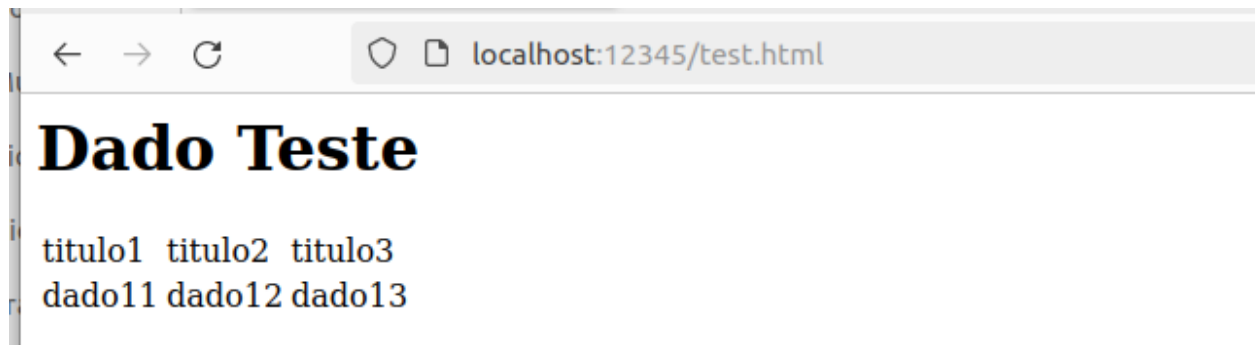
    finally:
        server_socket.close()

if __name__ == '__main__':
    run_server()
```

Criando arquivo html na pasta selecionada e conteudo fornecido:



Resultado:



14. Implemente no arquivo `https.py` uma função chamada `get_https_page` que receba como entrada uma url e retorna o documento HTML correspondente à url. Seu script Python deve utilizar apenas as bibliotecas `socket` e `ssl` e deve baixar o documento com o protocolo HTTPS na porta 443.

R:

`https.py`:

```
Open  https.py
~/Desktop

1 import socket
2 import ssl
3 import chardet
4
5 def get_https_page(url):
6     url_components = url.split('/')
7     host = url_components[2]
8     path = '/' + '/'.join(url_components[3:])
9
10    context = ssl.create_default_context()
11    with socket.create_connection((host, 443)) as sock:
12        with context.wrap_socket(sock, server_hostname=host) as ssock:
13            request = f"GET {path} HTTP/1.1\r\nHost: {host}\r\nConnection: close\r\n\r\n"
14            ssock.sendall(request.encode())
15
16            response = b""
17            while True:
18                data = ssock.recv(4096)
19                if not data:
20                    break
21                response += data
22
23    headers, html = response.split(b'\r\n\r\n', 1)
24
25    encoding = chardet.detect(html)['encoding']
26    if encoding:
27        return html.decode(encoding)
28    else:
29        return html.decode('utf-8', 'ignore')
30
31 if __name__ == "__main__":
32     url = "https://www.example.com"
33     html = get_https_page(url)
34     print(html)
35
```

Resultado:

```
bob@iansantos:~/Desktop$ python3 https.py
<!doctype html>
<html>
<head>
  <title>Example Domain</title>

  <meta charset="utf-8" />
  <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <style type="text/css">
    body {
      background-color: #f0f0f2;
      margin: 0;
      padding: 0;
      font-family: -apple-system, system-ui, BlinkMacSystemFont, "Segoe UI", "Open Sans", "Helvetica Neue", Helvetica, Arial, sans-serif;
    }
    div {
      width: 600px;
      margin: 5em auto;
      padding: 2em;
      background-color: #fdfdff;
      border-radius: 0.5em;
      box-shadow: 2px 3px 7px 2px rgba(0,0,0,0.02);
    }
    a:link, a:visited {
      color: #38488f;
      text-decoration: none;
    }
    @media (max-width: 700px) {
      div {
        margin: 0 auto;
        width: auto;
      }
    }
  </style>
</head>

<body>
<div>
  <h1>Example Domain</h1>
  <p>This domain is for use in illustrative examples in documents. You may use this domain in literature without prior coordination or asking for permission.</p>
</div>
</body>
</html>
```

15. Considere que você gerencia um cluster de servidores e com frequência necessita verificar se eles estão ativos na rede e realizando login remoto com ssh. Para automatizar este processo, você resolve implementar um script python que utiliza a biblioteca scapy para verificar se todos os hospedeiros em uma lista de endereços IP estão ativos e se a porta 22 (ssh) está aberta.

- **Implemente tal script e teste-o utilizando uma lista contendo o IP local e o IP do Gateway Padrão. Seu script deve ser implementado no arquivo check_hosts.py e deve utilizar apenas a biblioteca scapy.**
- **Descreva qual a principal diferença da implementação do seu script com scapy para uma implementação com a biblioteca pcap-ng.**

R:

Check_Hosts.py

```
1 from scapy.all import *
2
3 def check_ssh(ip):
4     icmp = IP(dst=ip)/ICMP()
5     resp = sr1(icmp, timeout=2, verbose=False)
6
7     if resp:
8         tcp = IP(dst=ip)/TCP(dport=22, flags='S')
9         resp_tcp = sr1(tcp, timeout=1, verbose=False)
10
11         if resp_tcp and resp_tcp.haslayer(TCP):
12             if resp_tcp[TCP].flags == 18:
13                 return True
14         return False
15
16 def main():
17     ips = ['192.168.0.1', '192.168.0.254']
18
19     for ip in ips:
20         if check_ssh(ip):
21             print(f"Host {ip} está ativo e a porta SSH (22) está aberta.")
22         else:
23             print(f"Host {ip} não está ativo ou a porta SSH (22) está fechada.")
24
25 if __name__ == "__main__":
26     main()
27
```

Resultado:

```
etc of directory
bob@iansamtos:~/Desktop$ sudo python3 check_posts.py
Host 192.168.0.1 não está ativo ou a porta SSH (22) está fechada.
Host 192.168.0.254 não está ativo ou a porta SSH (22) está fechada.
bob@iansamtos:~/Desktop$ ~
```

Diferença com a Biblioteca pcap-ng:

A principal diferença entre Scapy e pcap-ng é a seguinte:

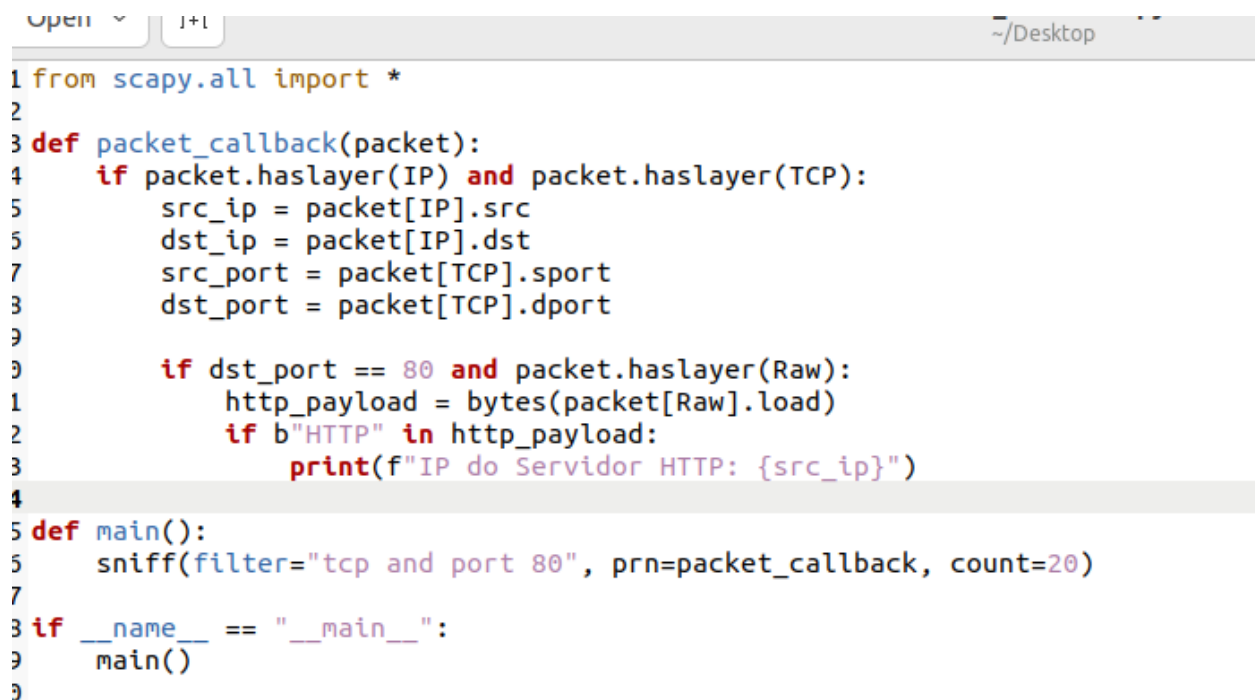
Scapy: É uma biblioteca Python usada para manipulação de pacotes de rede de forma detalhada e flexível. Scapy permite construir, enviar, capturar e decodificar pacotes de diversos protocolos de rede.

pcapy-ng: Por outro lado, pcapy-ng é uma biblioteca que fornece uma interface Python para a biblioteca C libpcap, que é amplamente utilizada para captura de pacotes de rede no nível do sistema operacional. pcapy-ng não possui as funcionalidades de construção e envio de pacotes como Scapy; ela é focada principalmente na captura de pacotes.

16. Escreva um programa Python no arquivo check_download.py que verifica se usuários da LAN estão fazendo download de conteúdo Web por meio do protocolo HTTP. Seu script deve capturar apenas os pacotes que chegam de algum Servidor Web HTTP para os hospedeiros da LAN. Ao capturar um pacote, você deve imprimir o endereço IP do servidor. Seu script deve utilizar apenas a biblioteca scapy.

R:

Check_download.py



```
Open 1+1 ~/Desktop
1 from scapy.all import *
2
3 def packet_callback(packet):
4     if packet.haslayer(IP) and packet.haslayer(TCP):
5         src_ip = packet[IP].src
6         dst_ip = packet[IP].dst
7         src_port = packet[TCP].sport
8         dst_port = packet[TCP].dport
9
10        if dst_port == 80 and packet.haslayer(Raw):
11            http_payload = bytes(packet[Raw].load)
12            if b"HTTP" in http_payload:
13                print(f"IP do Servidor HTTP: {src_ip}")
14
15 def main():
16     sniff(filter="tcp and port 80", prn=packet_callback, count=20)
17
18 if __name__ == "__main__":
19     main()
20
```

Resultado:

```
bob@iansantos:~/Desktop$ sudo python3 check_download.py
IP do Servidor HTTP: 10.0.2.15
IP do Servidor HTTP: 10.0.2.15
bob@iansantos:~/Desktop$
```

17. Você suspeita que seu Gateway está com a Tabela ARP falsificada.

Descreva como você pode identificar tal Falsificação ARP utilizando um script Python com scapy.

R:

Para identificar uma falsificação ARP (ARP spoofing) no seu Gateway utilizando um script Python com a biblioteca scapy, você pode seguir os passos abaixo. O objetivo é monitorar e analisar os pacotes ARP na rede para detectar possíveis discrepâncias nos endereços MAC associados aos IPs.

Exemplo:

```
1 from scapy.all import ARP, sniff
2
3 arp_table = {}
4
5 def detect_arp_spoof(packet):
6     if packet.haslayer(ARP) and packet[ARP].op == 2:
7         ip_src = packet[ARP].psrc
8         mac_src = packet[ARP].hwsrc
9
10        if ip_src in arp_table:
11            if arp_table[ip_src] != mac_src:
12                print(f"[ALERTA] Possível ataque ARP Spoofing detectado!")
13                print(f"IP: {ip_src} tem MAC diferente.")
14                print(f"MAC atual: {mac_src}, MAC esperado: {arp_table[ip_src]}")
15            else:
16                # Adicionar nova entrada na tabela ARP
17                arp_table[ip_src] = mac_src
18
19 # Iniciar a captura de pacotes ARP na rede
20 print("Monitorando pacotes ARP...")
21 sniff(store=False, prn=detect_arp_spoof, filter="arp")
22
```

O código acima detectando possíveis pacotes ARP:

```
bob@iansantos:~/Desktop$ sudo python3 arp_example.py
Monitorando pacotes ARP...
```

18. Com o objetivo de possuir um módulo em Python que forneça endereços IPv4 e nomes de servidores de e-mail para uma dada URL. Implemente o script `resolver.py` o qual possui as funções `url2ip` e `url2email`, as quais recebem uma URL como entrada e retornam o endereço IPv4 e os nomes de servidores de email (MX), respectivamente. A função `url2ip` deve ser implementada com a biblioteca `dnspython` e a função `url2email` deve ser implementada com a biblioteca `dnsrecon`.

R:

Resolver.py

```
1 import dns.resolver
2 import subprocess
3 import json
4 import tempfile
5 import os
6
7 def url2ip(url):
8     try:
9         result = dns.resolver.resolve(url, 'A')
10        return [ip.address for ip in result]
11    except Exception as e:
12        print(f"Erro ao resolver URL para IPv4: {e}")
13        return []
14
```



```

5 def url2email(url):
6     try:
7         with tempfile.NamedTemporaryFile(delete=False, suffix='.json') as temp_file:
8             temp_file_name = temp_file.name
9
10            command = ['dnsrecon', '-d', url, '-t', 'std', '-j', temp_file_name]
11            print(f"Executando comando: {' '.join(command)}")
12
13            result = subprocess.run(command, capture_output=True, text=True)
14            if result.returncode != 0:
15                print(f"Erro ao executar dnsrecon: {result.stderr}")
16                os.remove(temp_file_name)
17                return []
18
19            with open(temp_file_name, 'r') as temp_file:
20                recon_output = json.load(temp_file)
21
22            mx_records = []
23            if 'MX' in recon_output:
24                for mx_record in recon_output['MX']:
25                    mx_records.append(mx_record['data'])
26
27            os.remove(temp_file_name)
28            return mx_records
29        except Exception as e:
30            print(f"Erro ao resolver URL para servidores de email: {e}")
31            return []
32
33 if __name__ == "__main__":
34     url = "google.com"
35     print(f"Endereços IPv4 para {url}: {url2ip(url)}")
36     print(f"Servidores de email (MX) para {url}: {url2email(url)}")

```

Resultado:

```

bob@iansantos:~/Desktop$ python3 resolver.py
Endereços IPv4 para google.com: ['142.251.16.102', '142.251.16.101', '142.251.16.138', '142.251.16.139', '142.251.16.100', '142.251.16.113']
Executando comando: dnsrecon -d google.com -t std -j /tmp/tmpta3668sd.json
Servidores de email (MX) para google.com: []
bob@iansantos:~/Desktop$

```

19. Implemente no arquivo fuzzer.py um script de pentest que testa todas as URLs previsíveis para o domínio de teste testphp.vulnweb.com e com caminhos contidos no arquivo deste link “<https://github.com/fuzzdb-project/fuzzdb/blob/master/discovery/predictable-filepaths/login-file-locations/Logins.txt>”.

R: Fuzzer.py:

```
1 import requests
2
3 base_url = "http://testphp.vulnweb.com"
4 paths_file_url = "https://raw.githubusercontent.com/fuzzdb-project/fuzzdb/master/discovery/predictable-filepaths/login-file-locations/Logins.txt"
5
6 def load_paths_from_file(url):
7     try:
8         response = requests.get(url)
9         if response.status_code == 200:
10             paths = response.text.splitlines()
11             return paths
12         else:
13             print(f"Erro ao carregar caminhos do arquivo. Código de status: {response.status_code}")
14             return []
15     except Exception as e:
16         print(f"Erro ao carregar caminhos do arquivo: {e}")
17         return []
18
19 def test_paths(paths):
20     for path in paths:
21         url = f"{base_url}{path}"
22         try:
23             response = requests.get(url)
24             if response.status_code == 200:
25                 print(f"URL acessível: {url}")
26             elif response.status_code == 404:
27                 print(f"URL não encontrada: {url}")
28             else:
29                 print(f"Erro ao acessar URL {url}. Código de status: {response.status_code}")
30         except requests.RequestException as e:
31             print(f"Erro de requisição para URL {url}: {e}")
32
33 if __name__ == "__main__":
34     paths = load_paths_from_file(paths_file_url)
35
36     if paths:
37         print(f"Foram carregados {len(paths)} caminhos previsíveis.")
38         print("Iniciando teste de penetração...\n")
39         test_paths(paths)
40     else:
41         print("Nenhum caminho previsível foi carregado. Verifique a URL do arquivo ou a conexão com a internet.")
42
```

Resultado:

```
bob@lansantos:~/Desktop$ python3 fuzzer.py
Foram carregados 71 caminhos previsíveis.
Iniciando teste de penetração...

URL acessível: http://testphp.vulnweb.com/admin
URL não encontrada: http://testphp.vulnweb.com/Admin
URL não encontrada: http://testphp.vulnweb.com/admin.asp
URL não encontrada: http://testphp.vulnweb.com/admin.aspx
URL não encontrada: http://testphp.vulnweb.com/admin.cfm
URL não encontrada: http://testphp.vulnweb.com/admin.jsp
URL não encontrada: http://testphp.vulnweb.com/admin.php
URL não encontrada: http://testphp.vulnweb.com/Admin.php
URL não encontrada: http://testphp.vulnweb.com/admin.php4
URL não encontrada: http://testphp.vulnweb.com/admin.pl
URL não encontrada: http://testphp.vulnweb.com/Admin.pl
URL não encontrada: http://testphp.vulnweb.com/admin.py
URL não encontrada: http://testphp.vulnweb.com/admin.rb
URL não encontrada: http://testphp.vulnweb.com/administrator
URL não encontrada: http://testphp.vulnweb.com/Administrator
URL não encontrada: http://testphp.vulnweb.com/administrator.asp
URL não encontrada: http://testphp.vulnweb.com/administrator.aspx
URL não encontrada: http://testphp.vulnweb.com/administrator.cfm
URL não encontrada: http://testphp.vulnweb.com/administrator.jsp
```

20. Implemente no arquivo `asyncnmap.py` um script de varredura de portas assíncrono utilizando a biblioteca `nmap` do Python. Você deve executar o script para realizar a varredura nas seguintes URLs: `www.example.com` e `testphp.vulnweb.com`.

R:

Asyncnmap.py:

```
1 import asyncio
2 import nmap
3
4 async def scan_ports(hostname):
5     nm = nmap.PortScanner()
6
7     # Scanning top 1000 ports asynchronously
8     scan_result = nm.scan(hosts=hostname, arguments='-p 1-1000', sudo=True)
9
10    # Extracting open ports from the scan result
11    open_ports = []
12    for host, result in scan_result['scan'].items():
13        if 'tcp' in result:
14            for port, port_info in result['tcp'].items():
15                if port_info['state'] == 'open':
16                    open_ports.append((port, port_info['name']))
17
18    return open_ports
19
20 async def main():
21     hosts = ['www.example.com', 'testphp.vulnweb.com']
22
23     tasks = []
24     for host in hosts:
25         tasks.append(scan_ports(host))
26
27     results = await asyncio.gather(*tasks)
28
29     # Printing results
30     for i, host in enumerate(hosts):
31         print(f"Open ports for {host}:")
32         if results[i]:
33             for port, service in results[i]:
34                 print(f"  {port}/{service}")
35         else:
36             print("  No open ports found.")
37
38 if __name__ == "__main__":
39     asyncio.run(main())
40
```

Resultado:

```
bob@iansantos:~/Desktop$ python3 asyncnmap.py  
[sudo] password for bob:
```

```
Open ports for www.example.com:  
80/http  
443/https  
Open ports for testphp.vulnweb.com:  
80/http
```

21. Considere que você suspeita que seu hospedeiro pode ter sido acessado indevidamente por outro usuário e decide checar se existem conexões abertas por código malicioso ou backdoors. Execute o comando nmap para realizar tal verificação.

R:

Comando: `sudo nmap -sV -p- localhost`

```
bob@iansantos:~/Desktop$ sudo nmap -sV -p- localhost  
Starting Nmap 7.80 ( https://nmap.org ) at 2024-06-13 16:21 -03  
Nmap scan report for localhost (127.0.0.1)  
Host is up (0.0000030s latency).  
Not shown: 65530 closed ports  
PORT      STATE SERVICE      VERSION  
22/tcp    open  ssh          OpenSSH 8.9p1 Ubuntu 3ubuntu0.7 (Ubuntu Linux; proto  
col 2.0)  
631/tcp   open  ipp          CUPS 2.4  
3306/tcp   open  nagios-nsc   Nagios NSCA  
33060/tcp  open  mysqlx?        
39913/tcp  open  unknown        
2 services unrecognized despite returning data. If you know the service/version,  
please submit the following fingerprints at https://nmap.org/cgi-bin/submit.cgi  
?new-service :  
=====NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)=====  
SF-Port33060-TCP:V=7.80%I=7%D=6/13%Time=666B46DA%P=x86_64-pc-linux-gnu%(N  
SF:ULL,9,"%x05%0%0%0%x0b%x08%x05%x1a%0")%r(GenericLines,9,"%x05%0%0%0%x0b\  
SF:x08%x05%x1a%0")%r(GetRequest,9,"%x05%0%0%0%x0b%x08%x05%x1a%0")%r(HTTPop  
SF:tions,9,"%x05%0%0%0%x0b%x08%x05%x1a%0")%r(RTSPRequest,9,"%x05%0%0%0%x0b  
SF:%x08%x05%x1a%0")%r(RPCCheck,9,"%x05%0%0%0%x0b%x08%x05%x1a%0")%r(DNSVers  
SF:ionBindReqTCP,9,"%x05%0%0%0%x0b%x08%x05%x1a%0")%r(DNSStatusRequestTCP,2  
SF:B,"%x05%0%0%0%x0b%x08%x05%x1a%0%x1e%0%0%0%01%x08%01%x10%x88'x1a%0fI  
SF:nyvalid%x20message\"%x05HY0000\")%r(Hello,9,"%x05%0%0%0%x0b%x08%x05%x1a%0")
```

Bibliografia:

<https://docs.python.org/3/library/socket.html>

<https://nmap.org/>

<https://pypi.org/project/python-nmap/>

<https://ubuntu.com/tutorials>

<https://docs.python.org/pt-br/3/library/asyncio.html>

[https://www.vivaolinux.com.br/dica/O-que-e-um-Fuzzer-em-Penetration-Testing-\(Pentesting\)](https://www.vivaolinux.com.br/dica/O-que-e-um-Fuzzer-em-Penetration-Testing-(Pentesting))

<https://docs.python.org/3/library/os.html>

<https://docs.python.org/3/library/tempfile.html>

<https://scapy.net/>

<https://docs.python.org/3/library/ssl.html>

<https://pypi.org/project/dnspython/>

<https://docs.python.org/3/library/subprocess.html>