

# Segurança da Informação e Segurança em Redes de Computadores

Prof. Dr. Kelton Costa

[kelton.costa@gmail.com](mailto:kelton.costa@gmail.com)

[kelton.costa@unesp.br](mailto:kelton.costa@unesp.br)



- O tcpdump é um analisador de tráfego em modo texto (via terminal) muito utilizado.
- Baseado na libpcap, uma poderosa *API* para a captura de pacotes de rede durante o tráfego corrente.
- O tcpdump mostra as conexões estabelecidas e o tráfego correspondente.

# Introdução

- A vantagem de executar o tcpdump em um terminal é poder analisar tráfego em roteadores e outros dispositivos de rede a partir do acesso remoto (SSH).

# Introdução

- Com o uso de *hubs*, o tcpdump permite capturar todo o tráfego gerado em tempo real, isso se deve ao fato de que o *hub* se comporta como um barramento.
- Com o uso de *switches* o tráfego da rede local é segmentado entre as duas entidades que estão se comunicando, não sendo possível capturar o tráfego de terceiros.

# Introdução

- O tcpdump está disponível para os sistemas operacionais Unix, GNU/Linux, BSD, OS X, Solaris, Windows (*WinDump*)

# Introdução

- Outro programa similar ao tcpdump é o wireshark, o qual dispõe de uma interface gráfica para facilitar a análise o tráfego de rede.

# Introdução

- O tcpdump é uma ferramenta muito conveniente para analisar tráfegos em redes de computadores e solucionar possíveis problemas de rede, detectar máquinas infectadas por malware, entre outras.
- Entretanto, é possível também ser aplicado em atividades negativas, como capturar senhas de conexões não cifradas.

# Instalação

- `# apt-get install tcpdump`



# Chaves e Funções

- As chaves exibidas são as mais utilizadas. Existem outras disponíveis que podem ser acessadas via terminal (`$ man tcpdump`) ou pela página [http://tcpdump.org/tcpdump\\_man.html](http://tcpdump.org/tcpdump_man.html).

Chave	Função
-D	Mostra as interfaces de rede disponíveis.
-i iface	Determina qual interface de rede deverá ser utilizada. Caso nenhuma seja especificada, a primeira mostrada pela chave -D será utilizada. É possível utilizar qualquer uma mostrada pela chave -D, podendo citá-la pelo nome ou pelo número. Para escutar em todas as interfaces, utilize <b>any</b> como iface.
-n	Não faz resolução de nomes de hosts e nem de portas, acelerando a exibição dos resultados na tela (tempo real). É aconselhável sempre utilizar -n nas análises de tráfego.

-N	Ao resolver nomes, não mostra o domínio do host.
-A	Mostra cabeçalho e payload dos pacotes em ASCII.
-X	Idem, mas em hexadecimal e caracteres ASCII.
-x	Idem, mas somente em sequências em hexadecimal.
-v	Aumenta a quantidade de informações extraídas do cabeçalho do pacote.
-vv	Idem ao anterior, com mais informações ainda.
-vvv	Idem ao anterior, com mais informações.

-w arq	Grava o resultado da captura em um arquivo. É importante ressaltar que se nenhuma outra chave ou expressão de filtragem for utilizada, todo o tráfego passante será gravado. É aconselhável utilizar as chaves <b>-nv</b> para acelerar a gravação, por não resolver nomes, e para mostrar detalhes da captura em andamento.
-r arq	Lê um arquivo previamente gravado com w. Diversas chaves poderão ser utilizadas para depurar o resultado.
-t	Não mostra a data e a hora na tela.
-tttt	Mostra a data e a hora utilizando o padrão yyyy-mm-dd - hh:mm:ss.ssssss.

-e

Mostra também os dados referentes à camada 2 do Modelo OSI (enlace).

-S

Exibe os resultados TCP utilizando a sua sequência absoluta, em vez da sequência relativa. Recomendado na análise de sequências TCP.

Last login: Fri Oct 2 08:27:44 on ttys000

(base) keltoncosta@Keltons-MacBook-Air ~ % apt-get install tcpdump

Last login: Fri Oct 2 08:29:34 on ttys000

((base) keltoncosta@Keltons-MacBook-Air ~ % pwd  
/Users/keltoncosta

((base) keltoncosta@Keltons-MacBook-Air ~ % ls

Applications	Downloads	Movies
Desktop	Dropbox	Music
Documents	Library	Pictures

((base) keltoncosta@Keltons-MacBook-Air ~ % cd Desktop

((base) keltoncosta@Keltons-MacBook-Air Desktop % ls

1.png	Icon?	~\$ojeto metodologia-xai.docx
2. Técnicas Clássicas de Criptografia	defesa-mestrado-thiago	

((base) keltoncosta@Keltons-MacBook-Air Desktop %

Public  
Sites  
VirtualBox VMs

anaconda3  
googledrive  
iCloud Drive (Archive)

iCloud Drive (Archive) - 1  
opt  
seaborn-data

(base) keltoncosta@Keltons-MacBook-Air Desktop % tcpdump -D

1.en0 [Up, Running]

2.p2p0 [Up, Running]

3.awdl0 [Up, Running]

4.llw0 [Up, Running]

5.utun0 [Up, Running]

6.utun1 [Up, Running]

7.utun2 [Up, Running]

8.utun3 [Up, Running]

9.lo0 [Up, Running, Loopback]

10.bridge0 [Up, Running]

11.en1 [Up, Running]

12.en2 [Up, Running]

13.pktap0 [Up]

14.gif0 [none]

15.stf0 [none]

16.ap1 [none]

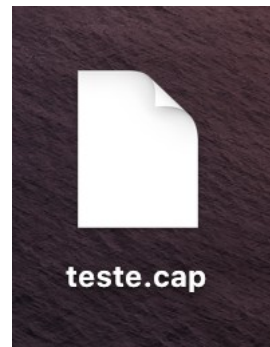
(base) keltoncosta@Keltons-MacBook-Air Desktop %

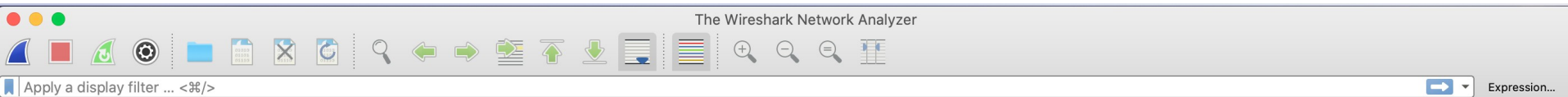


\_\_\_\_\_



```
(base) keltoncosta@Keltons-MacBook-Air Desktop % tcpdump -w teste.cap
tcpdump: ioctl(SIOCIFCREATE): Operation not permitted
(base) keltoncosta@Keltons-MacBook-Air Desktop % tcpdump -i en0 -w teste.cap
tcpdump: listening on en0, link-type EN10MB (Ethernet), capture size 262144 bytes
^C148 packets captured
148 packets received by filter
0 packets dropped by kernel
(base) keltoncosta@Keltons-MacBook-Air Desktop %
```





# Welcome to Wireshark

## Open

/Users/keltoncosta/Desktop/teste.cap (2347 Bytes)

/Users/keltoncosta/Desktop/primeiracaptura.pcap (not found)

*/Users/keltoncosta/Desktop/21.pcap (not found)*

*/Users/keltoncosta/Dropbox/fatec/2019/segurança em redes de computadores/captura-senha-ftp-tcpdump/1194.pcap (not found)*

*/Users/keltoncosta/Dropbox/fatec/2019/segurança em redes de computadores/captura-senha-ftp-tcpdump/80.pcap (not found)*

*/Users/keltoncosta/Dropbox/fatec/2019/segurança em redes de computadores/captura-senha-ftp-tcpdump/21.pcap (not found)*

*/Users/keltoncosta/Desktop/teste.pcap (not found)*

/Users/keltoncosta/Desktop/captura.pcap (not found)



*/Users/keltoncosta/Desktop/massae0captura/captura.pcap (not found)*

```
//Users/keltoncosta/Desktop/gregiocantura.ncan (not found)
```

## Capture

...using this filter:

All interfaces shown

Wi-Fi: en0	
p2p0	—
awdl0	—
llw0	—
utun0	—
utun1	—
utun2	

## Learn

[User's Guide](#) · [Wiki](#) · [Questions and Answers](#) · [Mailing Lists](#)

You are running Wireshark 3.0.1 (v3.0.1-0-gea351cd8).

## Wireshark · Open Capture File

Look in: /Users/keltoncosta/Desktop

Computer

keltoncosta

Name	Size	Kind	Date Modified
2.png	16...KiB	png File	02/10/20 08:30
3.png	02...KiB	png File	02/10/20 08:31
4.png	8...iB	png File	02/10/20 08:32
5.png	61,...KiB	png File	02/10/20 08:34
defesa-mestrado-thiago	--	Folder	30/09/20 13:38
Screen Shot 2020... at 08.37.34.png	84,...KiB	png File	02/10/20 08:37
Screen Shot 2020-...2 at 08.37.47.png	57,...KiB	png File	02/10/20 08:37
Screen Shot 2020... at 08.57.33.png	156...KiB	png File	02/10/20 08:57
teste.cap	2,29 KiB	cap File	21/06/16 07:26

File name: teste.cap

Open

Cancel

Files of type: All Files

Help

Automatically detect file type

Format: Wireshark/tcpdump/... - pcap  
Size: 2347 bytes, 36 data records  
Start / elapsed: 2016-05-02 20:14:02 / 00:00:10

Read filter: Apply a read filter ...









teste.cap

Apply a display filter ... < %/ >

Expression... +

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.6	192.168.10.3	TCP	52	52064 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1368
2	0.000475	192.168.10.3	192.168.0.6	TCP	52	21 → 52064 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0
3	0.010769	192.168.0.6	192.168.10.3	TCP	40	52064 → 21 [ACK] Seq=1 Ack=1 Win=65536 Len=0
4	0.012277	192.168.10.3	192.168.0.6	FTP	60	Response: 220 (vsFTPd 2.2.2)
5	0.023432	192.168.0.6	192.168.10.3	FTP	56	Request: USER anonymous
6	0.023906	192.168.10.3	192.168.0.6	TCP	40	21 → 52064 [ACK] Seq=21 Ack=17 Win=14720 Len=0
7	0.023926	192.168.10.3	192.168.0.6	FTP	74	Response: 331 Please specify the password.
8	0.027283	192.168.0.6	192.168.10.3	FTP	65	Request: PASS chrome@example.com
9	0.067525	192.168.10.3	192.168.0.6	TCP	40	21 → 52064 [ACK] Seq=55 Ack=42 Win=14720 Len=0
10	3.446327	192.168.10.3	192.168.0.6	FTP	62	Response: 530 Login incorrect.
11	3.475304	192.168.0.6	192.168.10.3	FTP	46	Request: QUIT
12	3.475726	192.168.10.3	192.168.0.6	TCP	40	21 → 52064 [ACK] Seq=77 Ack=48 Win=14720 Len=0
13	3.475743	192.168.10.3	192.168.0.6	FTP	54	Response: 221 Goodbye.
14	3.475754	192.168.10.3	192.168.0.6	TCP	40	21 → 52064 [FIN, ACK] Seq=91 Ack=48 Win=14720 Len=0
15	3.486905	192.168.0.6	192.168.10.3	TCP	40	52064 → 21 [ACK] Seq=48 Ack=91 Win=65536 Len=0

> Frame 8: 65 bytes on wire (520 bits), 65 bytes captured (520 bits)

Raw packet data

> Internet Protocol Version 4, Src: 192.168.0.6, Dst: 192.168.10.3

> Transmission Control Protocol, Src Port: 52064, Dst Port: 21, Seq: 17, Ack: 55, Len: 25

> File Transfer Protocol (FTP)

[Current working directory: ]

0000

45 00 00 41 32 b7 40 00 80 06 3c a6 c0 a8 00 06

E A2 @ . . . . .

0010

c0 a8 0a 03 cb 60 00 15 0f 26 20 6f e5 d3 e6 73

. . . . . & o . . s

0020

50 18 01 00 e3 f1 00 00 50 41 53 53 20 63 68 72

P . . . . . PASS chr

0030

6f 6d 65 40 65 78 61 6d 70 6c 65 2e 63 6f 6d 0d

ome@exam ple.com .

0040

0a

.

teste.cap

Packets: 36 · Displayed: 36 (100.0%)

Profile: Default







teste.cap

Apply a display filter ... < % / >

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
16	3.496942	192.168.0.6	192.168.10.3	TCP	40	52064 → 21 [ACK] Seq=48 Ack=92 Win=65536 Len=0
17	3.496971	192.168.0.6	192.168.10.3	TCP	40	52064 → 21 [FIN, ACK] Seq=48 Ack=92 Win=65536 Len=0
18	3.497346	192.168.10.3	192.168.0.6	TCP	40	21 → 52064 [ACK] Seq=92 Ack=49 Win=14720 Len=0
19	7.516624	192.168.0.6	192.168.10.3	TCP	52	52065 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1368
20	7.517118	192.168.10.3	192.168.0.6	TCP	52	21 → 52065 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0
21	7.531328	192.168.0.6	192.168.10.3	TCP	40	52065 → 21 [ACK] Seq=1 Ack=1 Win=65536 Len=0
22	7.532779	192.168.10.3	192.168.0.6	FTP	60	Response: 220 (vsFTPd 2.2.2)
23	7.546625	192.168.0.6	192.168.10.3	FTP	51	Request: USER caio
24	7.547082	192.168.10.3	192.168.0.6	TCP	40	21 → 52065 [ACK] Seq=21 Ack=12 Win=14720 Len=0
25	7.547102	192.168.10.3	192.168.0.6	FTP	74	Response: 331 Please specify the password.
26	7.554274	192.168.0.6	192.168.10.3	FTP	55	Request: PASS teste123
27	7.594524	192.168.10.3	192.168.0.6	TCP	40	21 → 52065 [ACK] Seq=55 Ack=27 Win=14720 Len=0
28	10.429894	192.168.10.3	192.168.0.6	FTP	62	Response: 530 Login incorrect.
29	10.466224	192.168.0.6	192.168.10.3	FTP	46	Request: QUIT

▶ Frame 22: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)

Raw packet data

▶ Internet Protocol Version 4, Src: 192.168.10.3, Dst: 192.168.0.6

▶ Transmission Control Protocol, Src Port: 21, Dst Port: 52065, Seq: 1, Ack: 1, Len: 20

▶ File Transfer Protocol (FTP)

[Current working directory: ]

0000 45 00 00 3c 20 fc 40 00 3f 06 8f 66 c0 a8 0a 03 E...<.@?..f....

0010 c0 a8 00 06 00 15 cb 61 da f0 b4 79 bd 75 13 eb .....a...y.u...

0020 50 18 00 73 9d 9a 00 00 32 32 30 20 28 76 73 46 P..s....220(vsF

0030 54 50 64 20 32 2e 32 2e 32 29 0d 0a TPd 2.2. 2)...

teste.cap

Packets: 36 · Displayed: 36 (100.0%)

Profile: Default

teste.cap

Apply a display filter ... <%%/>

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
16	3.496942	192.168.0.6	192.168.10.3	TCP	40	52064 → 21 [ACK] Seq=48 Ack=92 Win=65536 Len=0
17	3.496971	192.168.0.6	192.168.10.3	TCP	40	52064 → 21 [FIN, ACK] Seq=48 Ack=92 Win=65536 Len=0
18	3.497346	192.168.10.3	192.168.0.6	TCP	40	21 → 52064 [ACK] Seq=92 Ack=49 Win=14720 Len=0
19	7.516624	192.168.0.6	192.168.10.3	TCP	52	52065 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1368
20	7.517118	192.168.10.3	192.168.0.6	TCP	52	21 → 52065 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0
21	7.531328	192.168.0.6	192.168.10.3	TCP	40	52065 → 21 [ACK] Seq=1 Ack=1 Win=65536 Len=0
22	7.532779	192.168.10.3	192.168.0.6	FTP	60	Response: 220 (vsFTPD 2.2.2)
23	7.546625	192.168.0.6	192.168.10.3	FTP	51	Request: USER caio
24	7.547082	192.168.10.3	192.168.0.6	TCP	40	21 → 52065 [ACK] Seq=21 Ack=12 Win=14720 Len=0
25	7.547102	192.168.10.3	192.168.0.6	FTP	74	Response: 331 Please specify the password.
26	7.554274	192.168.0.6	192.168.10.3	FTP	55	Request: PASS teste123
27	7.594524	192.168.10.3	192.168.0.6	TCP	40	21 → 52065 [ACK] Seq=55 Ack=27 Win=14720 Len=0
28	10.429894	192.168.10.3	192.168.0.6	FTP	62	Response: 530 Login incorrect.
29	10.466224	192.168.0.6	192.168.10.3	FTP	46	Request: QUIT

▶ Frame 23: 51 bytes on wire (408 bits), 51 bytes captured (408 bits)

Raw packet data

▶ Internet Protocol Version 4, Src: 192.168.0.6, Dst: 192.168.10.3

▶ Transmission Control Protocol, Src Port: 52065, Dst Port: 21, Seq: 1, Ack: 21, Len: 11

▶ File Transfer Protocol (FTP)

[Current working directory: ]

0000 45 00 00 33 32 be 40 00 80 06 3c ad c0 a8 00 06 E..32.@...<....

0010 c0 a8 0a 03 cb 61 00 15 bd 75 13 eb da f0 b4 8d .....a...u.....

0020 50 18 01 00 61 92 00 00 55 53 45 52 20 63 61 69 P...a... USER cai

0030 6f 0d 0a o...

teste.capPackets: 36 · Displayed: 36 (100.0%)Profile: Default



teste.cap

Apply a display filter ... <?/?>

Expression...

No.	Time	Source	Destination	Protocol	Length	Info
16	3.496942	192.168.0.6	192.168.10.3	TCP	40	52064 → 21 [ACK] Seq=48 Ack=92 Win=65536 Len=0
17	3.496971	192.168.0.6	192.168.10.3	TCP	40	52064 → 21 [FIN, ACK] Seq=48 Ack=92 Win=65536 Len=0
18	3.497346	192.168.10.3	192.168.0.6	TCP	40	21 → 52064 [ACK] Seq=92 Ack=49 Win=14720 Len=0
19	7.516624	192.168.0.6	192.168.10.3	TCP	52	52065 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1368
20	7.517118	192.168.10.3	192.168.0.6	TCP	52	21 → 52065 [SYN, ACK] Seq=0 Ack=1 Win=14600 Len=0
21	7.531328	192.168.0.6	192.168.10.3	TCP	40	52065 → 21 [ACK] Seq=1 Ack=1 Win=65536 Len=0
22	7.532779	192.168.10.3	192.168.0.6	FTP	60	Response: 220 (vsFTPd 2.2.2)
23	7.546625	192.168.0.6	192.168.10.3	FTP	51	Request: USER caio
24	7.547082	192.168.10.3	192.168.0.6	TCP	40	21 → 52065 [ACK] Seq=21 Ack=12 Win=14720 Len=0
25	7.547102	192.168.10.3	192.168.0.6	FTP	74	Response: 331 Please specify the password.
26	7.554274	192.168.0.6	192.168.10.3	FTP	55	Request: PASS teste123
27	7.594524	192.168.10.3	192.168.0.6	TCP	40	21 → 52065 [ACK] Seq=55 Ack=27 Win=14720 Len=0
28	10.429894	192.168.10.3	192.168.0.6	FTP	62	Response: 530 Login incorrect.
29	10.466224	192.168.0.6	192.168.10.3	FTP	46	Request: QUIT

▶ Frame 28: 62 bytes on wire (496 bits), 62 bytes captured (496 bits)

Raw packet data

▶ Internet Protocol Version 4, Src: 192.168.10.3, Dst: 192.168.0.6

▶ Transmission Control Protocol, Src Port: 21, Dst Port: 52065, Seq: 55, Ack: 27, Len: 22

▶ File Transfer Protocol (FTP)

[Current working directory: ]

0000 45 00 00 3e 21 00 40 00 3f 06 8f 60 c0 a8 0a 03 E...!..@ ?... ..

0010 c0 a8 00 06 00 15 cb 61 da f0 b4 af bd 75 14 05 .....a .....u..

0020 50 18 00 73 4a 1f 00 00 35 33 30 20 4c 6f 67 69 P...sJ... 530 Logi

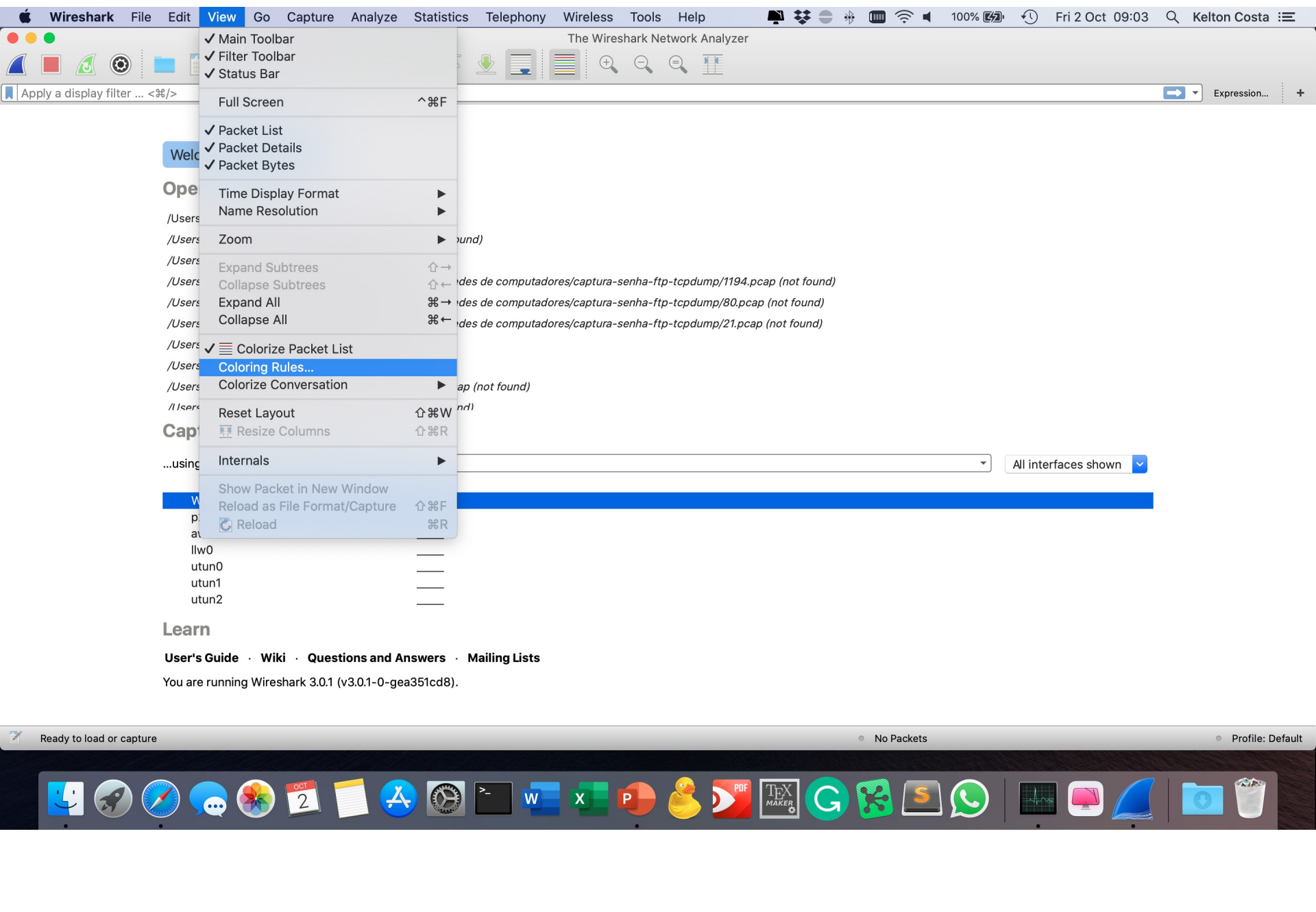
0030 6e 20 69 6e 63 6f 72 72 65 63 74 2e 0d 0a n incorr ect...

teste.cap

Packets: 36 · Displayed: 36 (100.0%)

Profile: Default







Name	Filter
<input checked="" type="checkbox"/> Bad TCP	tcp.analysis.flags && !tcp.analysis.window_update
<input checked="" type="checkbox"/> HSRP State Change	hsrp.state != 8 && hsrp.state != 16
<input checked="" type="checkbox"/> Spanning Tree Topology Change	stp.type == 0x80
<input checked="" type="checkbox"/> OSPF State Change	ospf.msg != 1
<input checked="" type="checkbox"/> ICMP errors	icmp.type eq 3    icmp.type eq 4    icmp.type eq 5    icmp.type eq 11    icmpv6.type eq 1    icmpv6.type eq 2    ic
<input checked="" type="checkbox"/> ARP	arp
<input checked="" type="checkbox"/> ICMP	icmp    icmpv6
<input checked="" type="checkbox"/> TCP RST	tcp.flags.reset eq 1
<input checked="" type="checkbox"/> SCTP ABORT	sctp.chunk_type eq ABORT
<input checked="" type="checkbox"/> TTL low or unexpected	( ! ip.dst == 224.0.0.0/4 && ip.ttl < 5 && !pim && !ospf )    ( ip.dst == 224.0.0.0/24 && ip.dst != 224.0.0.251 && ip.t
<input checked="" type="checkbox"/> Checksum Errors	eth.fcs.status=="Bad"    ip.checksum.status=="Bad"    tcp.checksum.status=="Bad"    udp.checksum.status=="
<input checked="" type="checkbox"/> SMB	smb    nbss    nbns    netbios
<input checked="" type="checkbox"/> HTTP	http    tcp.port == 80    http2
<input checked="" type="checkbox"/> DCERPC	dcerpc
<input checked="" type="checkbox"/> Routing	hsrp    eigrp    ospf    bgp    cdp    vrrp    carp    gvrp    igmp    ismp
<input checked="" type="checkbox"/> TCP SYN/FIN	tcp.flags & 0x02    tcp.flags.fin == 1
<input checked="" type="checkbox"/> TCP	tcp
<input checked="" type="checkbox"/> UDP	udp
<input checked="" type="checkbox"/> Broadcast	eth[0] & 1
<input checked="" type="checkbox"/> System Event	systemd_journal    sysdig

Double click to edit. Drag to move. Rules are processed in order until a match is found.



Help

Import...

Export...

Copy from



Cancel

OK

## Field Name

## ▼ TCP · Transmission Control Protocol

mptcp.connection.echoed\_key\_mismatch · The echoed key in the ...  
mptcp.connection.missing\_algorithm · No crypto algorithm specifi...  
mptcp.connection.unsupported\_algorithm · Unsupported algorithm  
mptcp.dss.infinite\_mapping · Fallback to infinite mapping  
mptcp.dss.missing\_mapping · No mapping available  
tcp.ack · Acknowledgment number  
tcp.ack.nonzero · The acknowledgment number field is nonzero w...  
tcp.analysis · SEQ/ACK analysis  
tcp.analysis.ack\_lost\_segment · ACKed segment that wasn't capt...  
tcp.analysis.ack\_rtt · The RTT to ACK the segment was  
tcp.analysis.acks\_frame · This is an ACK to the segment in frame  
tcp.analysis.bytes\_in\_flight · Bytes in flight  
tcp.analysis.duplicate\_ack · Duplicate ACK  
tcp.analysis.duplicate\_ack\_frame · Duplicate to the ACK in frame  
tcp.analysis.duplicate\_ack\_num · Duplicate ACK #  
tcp.analysis.fast\_retransmission · This frame is a (suspected) fast ...  
tcp.analysis.flags · TCP Analysis Flags  
tcp.analysis.initial\_rtt · iRTT  
tcp.analysis.keep\_alive · TCP keep-alive segment  
tcp.analysis.keep\_alive\_ack · ACK to a TCP keep-alive segment  
tcp.analysis.lost\_segment · Previous segment(s) not captured (co...  
tcp.analysis.out\_of\_order · This frame is a (suspected) out-of-ord...  
tcp.analysis.push\_bytes\_sent · Bytes sent since last PSH flag  
tcp.analysis.retransmission · This frame is a (suspected) retransm...  
tcp.analysis.reused\_ports · A new tcp session is started with the s...  
tcp.analysis.rto · The RTO for this segment was  
tcp.analysis.rto\_frame · RTO based on delta from frame  
tcp.analysis.spurious\_retransmission · This frame is a (suspected)...  
tcp.analysis.tfo\_syn · TCP SYN with TFO Cookie  
tcp.analysis.window\_full · TCP window specified by the receiver is...  
tcp.analysis.window\_update · TCP window update

## Relation

is present

==

!=

&gt;

&lt;

&gt;=

&lt;=

contains

matches

in

Value (Protocol)

Predefined Values

Range (offset:length)

Search:

tc\_nv

Click OK to insert this filter

Help

Cancel

OK

# Expressões de Filtragem

- Além das chaves, o tcpdump admite as expressões de filtragem fornecidas pela libpcap. As expressões mais comuns são:
- As expressões de filtragem são fornecidas pela libpcap (\$ man pcap-filter) ou pela página <http://www.manpagez.com/man/7/pcap-filter>.

host <i>nome-ip</i>	Especifica que somente o tráfego envolvendo a máquina em questão, referenciada pelo seu nome ou IP, será mostrado.
net <i>rede/CIDR</i>	Idem ao anterior. No entanto, a filtragem é em relação a uma faixa de rede, em vez de uma máquina única. A expressão de filtragem poderá ser com CIDR, como em 192.168.1.0/24, ou com máscara de rede, como em 192.168.0.16 mask 255.255.255.0.
ether host <i>MAC</i>	Idem, referindo-se a um endereço MAC.
port <i>porta</i>	Idem, referindo-se a uma porta
src	Idem, referindo-se ao range de portas de 20 a 90.
dst	Delimita à origem. Pode ser associado a host, net, port, portrange e ether host. Exemplos: src host, src net, src port, ether src host.

not ou !	Operador lógico NOT. Utilizado para excluir algo do resultado da pesquisa. Ex.: ! port 80.
and ou &&	Operador lógico AND. Utilizado para associar duas ou mais expressões, tornando-as obrigatórias no resultado da pesquisa.
or ou	Operador lógico OR. Utilizado para declarar duas ou mais expressões, fazendo com que, pelo menos uma, apareça no resultado da pesquisa.
ip	Mostra somente o tráfego IPv4.
ip6	Mostra somente o tráfego IPv6.
tcp	Mostra somente o tráfego TCP.
udp	Mostra somente o tráfego UDP.

icmp / icmp6	Mostra apenas tráfego ICMP ou ICMP6.
arp	Mostra somente tráfego ARP.
stp	Apenas tráfego do tipo Spanning Tree Protocol
less <i>tam</i>	Mostra apenas pacotes com tamanho $\leq$ <i>tam</i> .
greater <i>tam</i>	Mostra apenas pacotes com tamanho $\geq$ <i>tam</i> .
vlan <i>id</i>	Mostra apenas o tráfego relativo à vlan que possui a identificação <i>id</i> .

# Exemplos de Uso

## Exemplo 1

- Mostrar todo o tráfego de rede, que passa pela primeira interface listada com `# tcpdump -D`, sem resolver nomes. Isso permitirá a visualização do tráfego em tempo real.
- `# tcpdump -n`

Tipos de anomalias para análise de assinatura

<https://www.ll.mit.edu/ideval/docs/attackDB.html>

# Exemplo 2

- Tráfego UDP no adaptador eth1, incluindo o payload (área de dados) em ASCII, sem resolver nomes.
- `# tcpdump -nAi eth1 udp`



# Exemplo 3

- Tráfego UDP com o host 10.1.1.25, sem resolver nomes.
- `# tcpdump -n host 10.1.1.25 and udp`

# Exemplo 4

- Tráfego com o host 10.1.1.2, que seja UDP, e que tenha como origem ou destino a porta 53, sem resolver nomes.
- `# tcpdump -n host 10.1.1.2 and udp and port 53`

# Exemplo 5

- Tráfego que envolva o host 10.1.1.25, que seja UDP, e que esteja relacionado a qualquer porta, exceto a 53, sem resolver nomes. Também será mostrado o cabeçalho referente à camada de enlace.
- `# tcpdump -ne host 10.1.1.25 and udp and port ! 53`

# Exemplo 6

- Tráfego TCP que seja oriundo ou destinado à porta 80 ou que seja apenas oriundo da 110, sem resolver nomes. Os apóstrofos foram utilizados para evitar a interpretação errônea dos parênteses pelo shell.
- `# tcpdump -n tcp and '(port 80 or src port 110)'`

# Exemplo 7

- Tráfego ICMP referentes a qualquer host que pertença à rede 10.1.0.0/16, sem resolver nomes.
- `# tcpdump -n icmp and net 10.1.0.0/16`

# Exemplo 8

- Tráfego referente ao host que possua o endereço MAC especificado. Não resolve nomes.
- `# tcpdump -n ether host 00:ff:31:22:2d:11`

# Exemplo 9

- É possível realizar filtragens, procurando por situações específicas no TCP. Para isso, é necessário conhecer a estrutura de cabeçalho do protocolo (RFC 793).

# Exemplo 9

- Filtrar apenas o tráfego que contenha as flags ACK e RST ativadas. Segundo a RFC 793, as flags TCP CWR, ECE, URG, ACK, PSH, RST, SYN e FIN, nesta ordem, estão no 14º byte do cabeçalho. Como a contagem inicia em zero, o 14º byte é o campo 13. Assim, será necessário marcar 1 na flag RST e 0 nas restantes. Pela ordem das flags, o resultado final será 00010100 que, em decimal, representa 20.



# Exemplo 9

- Resultado:
- `# tcpdump -n tcp[13] = 20`
- Para ver o tráfego que NÃO contenha ACK e RST, utilize:
- `# tcpdump -n tcp[13] != 20`

# Exemplo 10

- Utilizando *tcpdump* para capturar as senhas dos serviços de e-mail (POP3/IMAP/SMTP) e/ou de aplicações em HTTP, e mostrar o usuário e senha no através do terminal.
- **`tcpdump -i eth0 port smtp or port imap or port pop3 or port http -l -A | egrep -i 'pass=|pwd=|log=|login=|user=|username=|pw=|passw=|passwd=|password=|pass:|user:|username:|password:|login:|pass|user'`**

```
E...?u.0.....g.....n....{...P.0.w...USER xxxxx.yyyyyy
```

```
E...4u.0.....g.....n....{...P.0..u...PASS gol11123
```

```
E...>...0.....M.n...7\2...P...
```

# Exemplo 11

- Gravar os pacotes capturados em um arquivo de nome captura.cap:
- **sudo tcpdump -w captura.cap**

# Exemplo 12

- Ler os pacotes capturados a partir do arquivo `captura.cap`:
- **`sudo tcpdump -r captura.cap`**

# Exemplo 13

- Capturar somente o tráfego associado ao protocolo ICMP, na interface eth0:
- **`sudo tcpdump -i eth0 icmp`**

# Exemplo 14

- Capturar somente o tráfego associado ao protocolo ARP, na interface eth0:
- **`sudo tcpdump -i eth0 arp`**

# Exemplo 15

- Capturar somente 50 pacotes a partir da interface eth0:
- **`sudo tcpdump -c 50 -i eth0`**

# Exemplo 16

- Mostrar os pacotes capturados tanto em ASCII quanto em HEX, incluindo cabeçalho Ethernet:
- **`sudo tcpdump -XX -i eth0`**



# Exemplo 17

- Capturar pacotes mostrando IPs em vez de nomes:
- **`sudo tcpdump -n -i eth0`**

# Exemplo 18

- Capturar somente pacotes maiores que 100 bytes:
- **`sudo tcpdump -i eth0 greater 100`**
- Neste exemplo, se emitirmos um comando ping a partir de outra janela de terminal, os pacotes não serão capturados, pois são menores que 100 bytes.

# Exemplo 19

- Capturar somente pacotes destinados à porta 53:
- **`sudo tcpdump -i eth0 port 53`**
- Para testar, abrimos um navegador e acessamos uma página qualquer da Web.

# Exemplo 20

- Usando filtros de condições: Capturar pacotes que usam o protocolo e cujo endereço de destino seja 64.233.186.121
- **`sudo tcpdump -i eth0 dst 64.233.186.121 and icmp`**
- Para testar, abrimos outra janela de terminal e emitimos o comando ping para vários endereços; somente serão capturados pacotes ao ser usado o endereço discriminado no comando. Se abrirmos um navegador e tentarmos acessar esse mesmo endereço (ou o site, [www.planetaunix.com.br](http://www.planetaunix.com.br)), os pacotes não serão capturados, por conta do protocolo utilizado (http em vez de icmp), mostrando que ambas as condições (AND) precisam ser satisfeitas para que essa captura tenha efeito.

# Exemplo 21

- Capturar somente os pacotes ICMP Echo Request enviados pelo programa ping da máquina local, cujo IP é 192.169.1.105, para um endereço remoto, como 8.8.8.8
- **`sudo tcpdump -i eth0 icmp and src 192.168.1.105 and dst 8.8.8.8`**
- Manual tcpdump:  
**`http://www.tcpdump.org/tcpdump\_man.html`**

# Case Study

# Outros Exemplos (1)

- Faz com que o sniffer capture apenas os pacotes que chegarem ou saírem da interface eth0.
- `tcpdump -i eth0`

# Outros Exemplos (2.1)

- Filtro por host. Capturar apenas o tráfego gerado/recebido pelo host 192.168.1.100.
- `tcpdump -nn -ni eth0 host 192.168.1.100`



## Outros Exemplos (2.2)

- Assim, em qualquer pacote no qual o host 192.168.1.100 esteja envolvido (seja recebendo, seja enviando) o tcpdump irá capturar o pacote e exibí-lo. Mas se desejar somente os pacotes destinados ao host 192.168.1.100 será também necessário o comando *src*.
- `# tcpdump -nn -ni eth0 src host 192.168.1.100`

## Outros Exemplos (2.3)

- Pronto, assim a captura se restringe exclusivamente aos pacotes originados no host 192.168.1.100. Para inverter esta lógica, ou seja, capturar pacotes cujo destino é o 192.168.1.100:
- `# tcpdump -nn -ni eth0 dst host 192.168.1.100`

## Outros Exemplos (2.4)

- Estas duas opções já diminuem muito a quantidade de pacotes capturados. Mas podemos ser ainda mais específicos. Por exemplo, usando port, dst port e src port podemos especificar também as portas. Assim, para capturar pacotes originados no host 192.168.1.100 com destino à porta 80 de qualquer outro host:
- **# tcpdump -nn -ni eth0 src host 192.168.1.100 and dst port 80**

# Outros Exemplos (2.5)

- Note que, neste caso, começamos a precisar dos operadores lógicos (and, or). O “and” faz com que o TCPDump só capture os pacotes onde ambas as condições forem verdadeiras (o pacote deve ter saído de 192.168.1.100 E ter como destino a porta 80). Se usássemos o “or”, qualquer pacote onde pelo menos uma das duas condições fossem verdadeiras seria capturado. Por exemplo, a linha abaixo captura pacotes originados no host 192.168.1.100 OU com destino à porta 80:
- **# tcpdump -nn -ni eth0 src host 192.168.1.100 or dst port 80**

## Outros Exemplos (2.6)

- Bem simples, não? Para especificar um host é a mesma coisa. Por exemplo, a linha abaixo vai capturar qualquer pacote que envolva os hosts 192.168.1.100 ou 192.168.1.101:
- **# tcpdump -nn -ni eth0 host 192.168.1.100 or host 192.168.1.101**

# Outros Exemplos (2.7)

- **# tcpdump -i eth0 dst host  
www.google.com.br**
- Assim, iremos capturar todos os pacotes com destino ao site google.com.br.

# Outros Exemplos (2.8)

- Você também pode negar um host, excluindo apenas os hosts que você especificar da captura. A seguinte linha:
- **# tcpdump -nn -ni eth0 not host 192.168.1.101**
- Não irá capturar nada relacionado ao host 192.168.1.101, porém irá capturar todo o resto já que não adicionamos mais nenhum outro filtro. Para fazer um filtro melhor, utilize os outros comandos que já vimos anteriormente.
- Claro que esta [lista não contém todas as possibilidades que o TCPCDump oferece, mas vai te dar uma boa base para começar a brincar com ele.](#)

# Outros Exemplos (2.9)

- **Armazenando os pacotes capturados em um arquivo**
- Não vai te ajudar muito apenas olhar o stream de pacotes todo na sua tela. Por isso, é importante que você armazene todos esses dados em um arquivo para que você possa analisar tudo com calma depois, identificando qual o problema.
- No TCPDump isso é feito simplesmente usando a flag -w:



# Outros Exemplos (2.10)

- **# tcpdump -nn -ni eth0 not host 192.168.1.101 -w /tmp/captura.pcap**
- Isso irá escrever todos os pacotes capturados no arquivo /tmp/captura.pcap. Este é um arquivo binário (escrever em um arquivo binário é mais rápido que escrever em um arquivo texto puro, evitando que o sniffer perca pacotes enquanto espera o fluxo ser salvo no arquivo), portanto não adianta você tentar lê-lo utilizando um editor de textos normal ou mesmo um comando como o cat, por exemplo. Você precisa de um software que saiba ler este arquivo como o próprio TCPDump ou o Wireshark (o qual não vou ensiná-lo a usar neste texto. Porém, existe muita documentação na Internet).

# Outros Exemplos (2.11)

- Depois que você escrever um arquivo binário com a captura, pode lê-lo utilizando a opção -r do TCPDump:
- **# tcpdump -r /tmp/captura.pcap**
- Lembre-se que aí estão os pacotes em si, não apenas aquele output que aparece na sua tela quando você executa o TCPDump.

## Outros Exemplos (2.12)

Quando você fizer o TCPDump ler um arquivo, você também pode aplicar alguns switches para mudar um pouco o comportamento dele. Por exemplo, você pode usar a opção -XX para imprimir tanto o conteúdo do pacote e também o seu cabeçalho. Por exemplo:

# Outros Exemplos (2.12)

- **# tcpdump -r /tmp/captura.pcap -XX**
- *reading from file /tmp/captura.pcap, link-type EN10MB (Ethernet)21:46:37.127671 IP 192.168.1.100.43532 > 95.215.62.5.http: UDP, length 160x0000: d85d 4cd9 70f6 001e 90fa d599 0800 4500 .]L.p.....E.0x0010: 002c 0000 4000 4011 dad8 c0a8 0164 5fd7 ,...@.@.....d\_.0x0020: 3e05 aa0c 0050 0018 f0dc 0000 0417 2710 >....P.....'. 0x0030: 1980 0000 0000 8519 3adb .....:21:46:37.385254 IP 192.168.1.100.43532 > 95.215.62.5.http: UDP, length 1760x0000: d85d 4cd9 70f6 001e 90fa d599 0800 4500 .]L.p.....E.0x0010: 00cc 0000 4000 4011 da38 c0a8 0164 5fd7 ....@.@..8...d\_. 0x0020: 3e05 aa0c 0050 00b8 8ac8 2342 0517 de41 >....P....#B...A0x0030: 50ff 0000 0002 8519 3adc a901 95a9 900a P.....:.....0x0040: d620 8dfe 9bc9 3787 3cfd fce8 0fcc dfa7 .....7.<.....0x0050: 7e6f 43ed 4f83 14ad 5950 933d 67ca 4d96 ~oC.O...YP.=g.M. 0x0060: fe73 adec 2d0f 8dfc 776d 07f4 9587 d05c .s..-...wm.....\0x0070: 454a a436 4109 ccce 06e4 2f01 022a 9b1a EJ.6A..../..\*..0x0080: 71e7 ea62 4e3f 8255 0093 60dc 2504 e250 q..bN?.U..`.%..P0x0090: f906 c69e b4e9 de96 754b 6fe1 d939 1fc1 .....uKo..9.. 0x00a0: 4ff8 45f8 72d5 a8da 34d3 5309 3265 c54f O.E.r...4.S.2e.O0x00b0: 8d3f 6d67 2035 937c 12f2 1dde 14ef 351d .?mg.5.|.....5.0x00c0: 96da 0ed2 39c1 94aa 4e7a fa2a a8d1 4341 ....9...Nz.\*..CA0x00d0: 3111 d2cc 3fe9 8eb3 a608 1...?.....*

## Outros Exemplos (2.13)

Assim você pode fazer uma análise ainda mais detalhada dos pacotes que você capturou. A opção -XX (ou -X, que não imprime alguns cabeçalhos de nível mais baixo) também pode ser utilizada diretamente na linha de comando, não apenas quando se está lendo um arquivo.

# Referências

- Filho, J. E. M. Análise de Tráfego em Redes TCP/IP. São Paulo: Novatec Editora, 2013.
- [tcpdump.org](http://tcpdump.org). Página oficial da ferramenta.