



CYBERSECURITY

ANÁLISE DE PACOTES **EM REDES TCP/IP**

RENATO JARDIM PARDUCCI



6

LISTA DE FIGURAS

Figura 6.1 – Cenário para estudo do <i>port mirroring</i>	6
Figura 6.2 – Verificação do <i>port mirroring</i>	7
Figura 6.3 – Pacote ICMP capturado pelo <i>sniffer</i>	7
Figura 6.4 – Identificação das <i>interfaces</i> de rede	9
Figura 6.5 – TCPDump utilizando os nomes dos hosts	10
Figura 6.6 – TCPDump utilizando os endereços IPv4 dos hosts	11
Figura 6.7 – Geração de arquivo .pcap com TCPDump	12
Figura 6.8 – Captura de sessão HTTP.....	13
Figura 6.9 – Inspeção do arquivo http.pcap	13
Figura 6.10 – Edição e formatação manuais do arquivo http.txt	14
Figura 6.11 – Verificação do endereço IPv4 dos <i>hosts</i>	15
Figura 6.12 – Filtragem de datagramas com <i>flag</i> PUSH habilitada	16
Figura 6.13 – Descoberta do nome do <i>host</i> com TCPDump	16
Figura 6.14 – Ambiente para testes FTP.....	17
Figura 6.15 – Descompactação da Metasploitable2	18
Figura 6.16 – Criação da VM TargetSrv	18
Figura 6.17 – Criação do disco rígido da VM TargetSrv	19
Figura 6.18 – Configuração final da Metasploitable2 (TargetSrv)	20
Figura 6.19 – Criação de arquivos de controle	20
Figura 6.20 – Terminais para captura da sessão FTP.....	21
Figura 6.21 – Utilização do cliente FTP em modo texto	22
Figura 6.22 – Abertura de arquivo de captura com o Wireshark.....	23
Figura 6.23 – Tipo de sessão FTP	24
Figura 6.24 – Filtragem de pacotes baseada em <i>strings</i>	24
Figura 6.25 – Sequência da sessão FTP	25
Figura 6.26 – Conexão FTP-DATA	26
Figura 6.27 – Caminho para visualização do conteúdo do arquivo	27
Figura 6.28 – Gravação do conteúdo do arquivo apt.conf	28

LISTA DE CÓDIGOS-FONTE

Código-fonte 6.1 – Listagem 1: Configuração do <i>port mirroring</i>	6
--	---

EXEMPLO

SUMÁRIO

6 ANÁLISE DE PACOTES EM REDES TCP/IP	5
6.1 Implementando o <i>port mirroring</i>	6
6.2 Sniffers e Analisadores de Protocolos.....	8
6.2.1 Hands on TCPDump	8
6.3 Implementando a metasploitable2.....	17
6.4 Analisando uma sessão ftp	20
REFERÊNCIAS	29
GLOSSÁRIO	30

6 ANÁLISE DE PACOTES EM REDES TCP/IP

A análise de pacotes em uma rede, também por vezes referenciada como análise de tráfego, pode ter diferentes motivações, sendo, entretanto, uma das práticas mais amplamente difundidas e relevantes, não somente em relação à segurança (cibernética e da informação), como também em relação às condições operacionais da rede. Dentre outras, pode-se apontar como algumas das principais razões que justificariam a análise de pacotes em uma rede:

- Verificar o tráfego gerado por protocolo.
- Verificar a existência de gargalos na rede.
- Correlacionar eventos.
- Detectar intrusões.
- Implementar políticas de segurança.

Uma das melhores formas para obter amostras do tráfego da rede consiste na utilização do SWITCHED PORT ANALIZER (*span*), também conhecido como *port mirroring*, o qual pode ser resumidamente descrito como uma funcionalidade dos *switches* gerenciáveis que copia os pacotes que entram ou saem por uma porta do equipamento, enviando essas cópias a outra porta previamente configurada, à qual podem ser conectadas aplicações específicas, tais como analisadores de tráfego e de protocolos.

É importante salientar a relevância do espelhamento de portas para viabilizar a análise de tráfego em redes comutadas (baseadas em *switches*) uma vez que, sendo esses dispositivos (nativamente) de camada 2 (camada de Enlace de Dados do modelo OSI), têm como função estabelecer conexões fim a fim entre o *host* de origem e o *host* de destino, com base nos respectivos endereços MAC, o que em princípio impossibilita a captura de pacotes por um terceiro *host*.

Para fins forenses, a análise de pacotes permite ainda obter o conteúdo das mensagens trafegadas nas redes, tais como e-mails, possibilitando, inclusive, a obtenção de cópias dos arquivos enviados/recebidos, cabendo, entretanto, mais uma vez, reforçar que tais práticas podem ser consideradas ilegais se o devido alinhamento jurídico não for observado, conforme será demonstrado adiante.

6.1 Implementando o *port mirroring*

Com base no **Cisco Packet Tracer**, será implementado o cenário de referência representado pela figura a seguir, com intuito de observarmos o funcionamento básico do *port mirroring*.

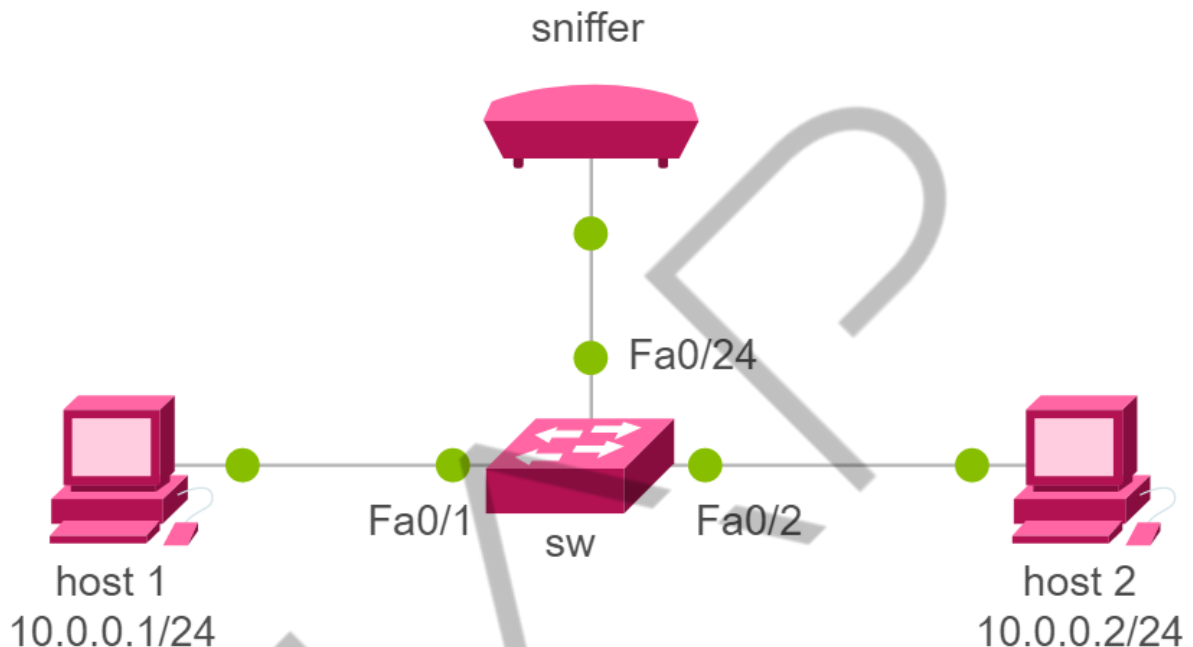


Figura 6.1 – Cenário para estudo do *port mirroring*
Fonte: Elaborado pelo autor (2020)

Montado o cenário da figura “Cenário para estudo do *port mirroring*”, e configurados os *hosts* 1 e 2 conforme ilustrado, configura-se então o switch SW, como mostra o código-fonte abaixo.

```
Switch>enable
Switch#configure terminal
Enter configuration commands, one per line. End with
CNTL/Z.
Switch(config)#monitor session 1 source interface fa0/1
Switch(config)#monitor session 1 source interface fa0/2
Switch(config)#monitor session 1 destination interface
fa0/24
Switch(config)#exit
```

Código-fonte 6.1 – Listagem 1: Configuração do *port mirroring*
Fonte: Elaborado pelo autor (2020)

A partir da listagem 1: Configuração do *port mirroring*, é possível observar que a primeira linha de comandos habilita o modo privilegiado do equipamento, passando-se na sequência ao modo de configuração global (notar a mudança no *prompt* de

comando). Como o *port mirroring* suporta múltiplas sessões, a primeira sessão foi configurada tendo as *interfaces* fa0/1 e fa0/2 como origens do tráfego a ser copiado, e a *interface* fa0/24 como destino desse tráfego, e em razão disso, conectada ao *sniffer*.

A figura “Verificação do *port mirroring*” exibe a saída do comando **show monitor**, permitindo confirmar que o *port mirroring* foi devidamente habilitado, e monitora as *interfaces* conforme especificado.

```
Switch#show monitor
Session 1
-----
Type                : Local Session
Description          : -
Source Ports        :
    Both             : Fa0/1, Fa0/2
Destination Ports   : Fa0/24
Encapsulation       : Native
    Ingress          : Disabled

Switch#
```

Figura 6.2 – Verificação do *port mirroring*
Fonte: Elaborado pelo autor (2020)

A partir do *host 1* (IP 10.0.0.1), uma sequência de pacotes ICMP foi enviada ao *host 2* (IP 10.0.0.2), sendo tais pacotes devidamente copiados para o *sniffer*, em função do SPAN. A figura adiante mostra os detalhes de um dos pacotes (ICMP) capturados.

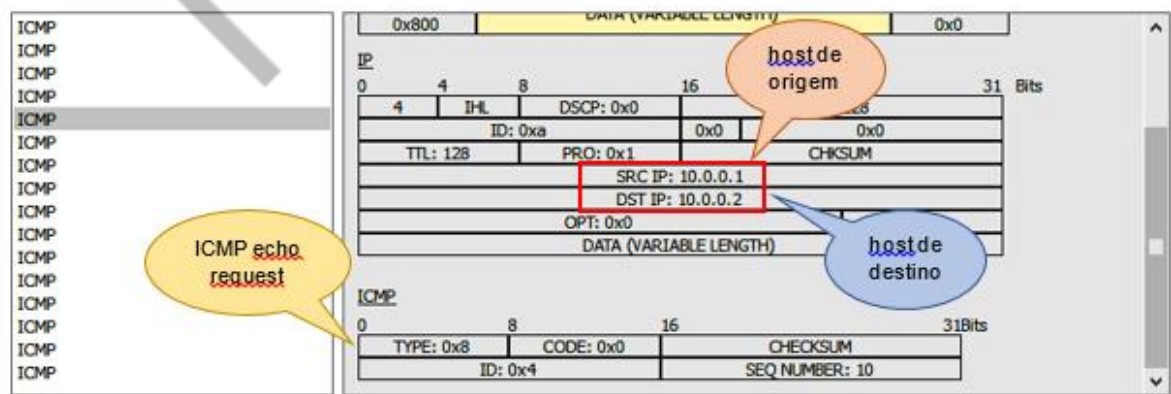


Figura 6.3 – Pacote ICMP capturado pelo *sniffer*
Fonte: Elaborado pelo autor (2020)

6.2 Sniffers e Analisadores de Protocolos

Sniffer é um termo popularmente utilizado para referenciar ferramentas capazes de capturar, e muitas vezes também analisar, os pacotes trafegados nas redes. O TCPDUMP é um dos *sniffers open source* em linha de comando mais conhecidos e amplamente utilizados, tendo como base a *libpcap*, uma poderosa API desenvolvida pelo próprio The TCPDUMP team.

A utilização desse tipo de ferramenta é relativamente simples, entretanto, a correta interpretação de seus resultados pode se tornar bastante desafiadora, na medida em que pode exigir sólidos conhecimentos no segmento das redes de computadores e protocolos de serviços e comunicações.

O Wireshark, por sua vez, é um dos aplicativos mais completos e amplamente utilizados para análise de protocolos. Dispõe de interface gráfica simples e adequadamente configurável, sendo capaz de capturar os pacotes a serem analisados, ou extraí-los de arquivos no formato *pcap*, como os gerados pelo TCPDump.

6.2.1 Hands on TCPDump

Considere agora um novo cenário, no qual o *host* Linux (Debian), que já virtualizamos anteriormente, será utilizado para navegação na Internet. Analogamente aos procedimentos adotados no referido capítulo para instalação do Wireshark, deverá, agora, ser instalado o TCPDump.

```
root@debian01:~# apt-get install -y tcpdump ftp
```

Finalizada a instalação, a ferramenta já estará apta à captura de pacotes. Entretanto, nem todos os sistemas terão apenas uma *interface* de rede, a exemplo do *host* de testes utilizado (VM Debian). Assim sendo, os trabalhos serão iniciados pela identificação de todas as interfaces de rede disponíveis no sistema, e aptas à captura de pacotes pelo TCPDump, o que será feito por meio da opção **-D** na linha de comando da ferramenta. Vale lembrar que a execução dessa ferramenta requer privilégios de superusuário (*root*).


```
root@debian01:~# tcpdump -D
1.enp0s3 [Up, Running]
2.any (Pseudo-device that captures on all interfaces) [Up, Running]
3.lo [Up, Running, Loopback]
4.nflog (Linux netfilter log (NFLOG) interface)
5.nfqueue (Linux netfilter queue (NFQUEUE) interface)
6.usbmon1 (USB bus number 1)
7.usbmon2 (USB bus number 2)
root@debian01:~#
```

Figura 6.4 – Identificação das *interfaces* de rede
Fonte: Elaborado pelo autor (2020)

Da resposta obtida após a execução da linha de comando em referência (figura “Identificação das *interfaces* de rede”), é possível observar que a *interface* de rede a ser utilizada pelo TCPDump é identificada como enp0s3, sendo então esse identificador especificado na linha de comando da ferramenta por meio da opção **-i <interface>**.

Ainda, com intuito de obter maior detalhamento dos pacotes capturados, será acrescentada também a opção **-v** (*verbose*) ficando, portanto, a linha de comando final:

```
root@debian01:~# tcpdump -i enp0s3 -v
```

Posto o TCPDump em execução, conforme a linha de comando descrita, a partir de um novo terminal aberto como usuário regular, é então enviado um pacote ICMP ao site do DEBIAN (www.debian.org), obtendo-se do TCPDump a resposta sintetizada a seguir.

```

!--- várias linhas omitidas
[1] 20:51:31.621596 IP (tos 0x0, ttl 64, id 39723, offset 0, flags [DF],
proto UDP (17), length 60)
  debian01.46875 > www.instaladorvivofibra.br.domain: 59779+ A?
  www.debian.org. (32)
!--- várias linhas omitidas
[2] 20:51:31.851657 IP (tos 0x0, ttl 64, id 33217, offset 0, flags [none],
proto UDP (17), length 76)
  www.instaladorvivofibra.br.domain > debian01.46875: 59779 1/0/0
  www.debian.org. A 200.17.202.197 (48)
!--- várias linhas omitidas
[3] 20:51:31.856931 IP (tos 0x0, ttl 64, id 45610, offset 0, flags [DF],
proto ICMP (1), length 84)
  debian01 > www.debian.org: ICMP echo request, id 1855, seq 1, length 64
!--- várias linhas omitidas
[4] 20:51:31.873340 IP (tos 0x0, ttl 54, id 33220, offset 0, flags [none],
proto ICMP (1), length 84)
  www.debian.org > debian01: ICMP echo reply, id 1855, seq 1, length 64

```

Figura 6.5 – TCPDump utilizando os nomes dos hosts

Fonte: Elaborado pelo autor (2020)

Analisando-se a saída do TCPDump, observa-se que:

- 1) Um datagrama IP foi enviado pelo *host* debian01 ao *host* www.instaladorvivofibra.com.br solicitando o endereço IPv4 do *host* www.debian.org.
- 2) O *host* www.instaladorvivofibra.com.br responde ao *host* debian01 que o endereço IPv4 do *host* www.debian.org é 200.17.202.197.
- 3) O *host* debian01 envia ao *host* www.debian.org outro datagrama IP, dessa vez com uma mensagem ICMP tipo *echo-request* (seq 1).
- 4) O *host* www.debian.org responde à mensagem do *host* debian01 enviando outro datagrama IP, com uma mensagem ICMP tipo *echo-reply* (seq 1).

Além das informações abordadas nos tópicos 1 a 4, observa-se também que a resposta do TCPDump traz o *timestamp* de cada pacote enviado/recebido, informação altamente relevante para auditorias e perícias forenses.

Há que se considerar, ainda, que por diferentes motivos, pode ser necessário ou preferível trabalhar diretamente com os endereços IP dos hosts, em vez de seus respectivos nomes, como por exemplo: www.debian.org. Geralmente, isto torna as respostas mais legíveis e fáceis de serem interpretadas.

Para instruir o TCPDump a não converter endereços, basta acrescentar à linha de comando a opção **-n**.

```
root@debian01:~# tcpdump -i enp0s3 -v -n
```

A figura a seguir sintetiza a resposta produzida pelo TCPDump ao utilizar essa nova opção em sua linha de comando, quando da captura de um novo pacote ICMP enviado ao site do DEBIAN.

Observa-se que, mesmo sendo essencialmente a mesma linha de comando utilizada da primeira vez, a saída produzida agora é um pouco mais fácil de compreender, uma vez que ao lidar-se diretamente com os endereços IPv4 dos *hosts* obtém-se maior legibilidade.

```
!---- várias linhas omitidas
[1] 22:37:15.476290 IP (tos 0x0, ttl 64, id 57449, offset 0, flags [DF],
proto UDP (17), length 60)
    10.0.2.15.52646 > 192.168.33.1.53: 26963+ A? www.debian.org. (32)
!---- várias linhas omitidas
[2] 22:37:15.480955 IP (tos 0x0, ttl 64, id 33231, offset 0, flags [none],
proto UDP (17), length 76)
    192.168.33.1.53 > 10.0.2.15.52646: 26963 1/0/0 www.debian.org. A
    200.17.202.197 (48)
!---- várias linhas omitidas
[3] 22:37:15.701571 IP (tos 0x0, ttl 64, id 25041, offset 0, flags [DF],
proto ICMP (1), length 84)
    10.0.2.15 > 200.17.202.197: ICMP echo request, id 2545, seq 1, length
    64
!---- várias linhas omitidas
[4] 22:37:15.718095 IP (tos 0x0, ttl 54, id 33233, offset 0, flags [none],
proto ICMP (1), length 84)
    200.17.202.197 > 10.0.2.15: ICMP echo reply, id 2545, seq 1, length 64
```

Figura 6.6 – TCPDump utilizando os endereços IPv4 dos hosts
Fonte: Elaborado pelo autor (2020)

Apesar dos ótimos resultados retornados pela ferramenta, as saídas obtidas ainda continuam sendo direcionadas exclusivamente à tela, o que geralmente não atende às necessidades de seu usuário. Entretanto, acrescentando-se a opção **-w <nome_arquivo.pcap>** à linha de comando, a saída do TCPDump será direcionada a um arquivo tipo *.pcap*, o qual poderá ser lido por qualquer aplicação apta à utilização desse tipo de arquivo, com o por exemplo, o Wireshark.

```
root@debian01:~# tcpdump -i enp0s3 -v -n -w captura.pcap
```

A figura a seguir oferece um exemplo de saída produzida pela ferramenta quando da utilização da opção `-w`.

```
root@debian01:~# tcpdump -i enp0s3 -v -n -w captura.pcap
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
^C13 packets captured
13 packets received by filter
0 packets dropped by kernel
root@debian01:~# file captura.pcap
captura.pcap: tcpdump capture file (little-endian) - version 2.4 (Ethernet, capture length 262144)
root@debian01:~#
```

Figura 6.7 – Geração de arquivo .pcap com TCPDump

Fonte: Elaborado pelo autor (2020)

Para finalizar a captura de pacotes e gravar o arquivo, basta digitar CTRL+C.

Vale ressaltar, ainda com base na figura “Geração de arquivo pcap com TCPDump”, que agora o TCPDump ecoa na tela um contador de pacotes em vez do conteúdo da captura, para que o usuário possa se orientar. O comando *file* utilizado serve para identificação do tipo de arquivo por ele analisado. Na próxima análise, será efetuada a captura de uma sessão *web*, na qual o usuário da VM Debian navega até a página inicial do site do Debian. Antes de iniciar o experimento, deve-se assegurar que o *cache* do navegador esteja limpo.

Lembrando-se de que o tráfego HTTP é caracterizado por uma conexão TCP à porta 80 do servidor, a nova linha de comando a ser utilizada pelo TCPDump será:

```
root@debian01:~# tcpdump -i enp0s3 -w http.pcap tcp port 80
```

Ao final dessa linha de comando, observa-se que a captura de pacotes deve ser limitada ao tráfego TCP que tiver como origem ou destino a porta 80 de um servidor HTTP, caracterizando, dessa forma, a navegação *web*, tendo-se, ainda, esse tráfego gravado em um arquivo denominado *http.pcap*.

A captura do tráfego será efetivada simplesmente disparando-se o TCPDump com a nova linha de comando, sendo o navegador *web* aberto em seguida e apontado para o site do Debian.

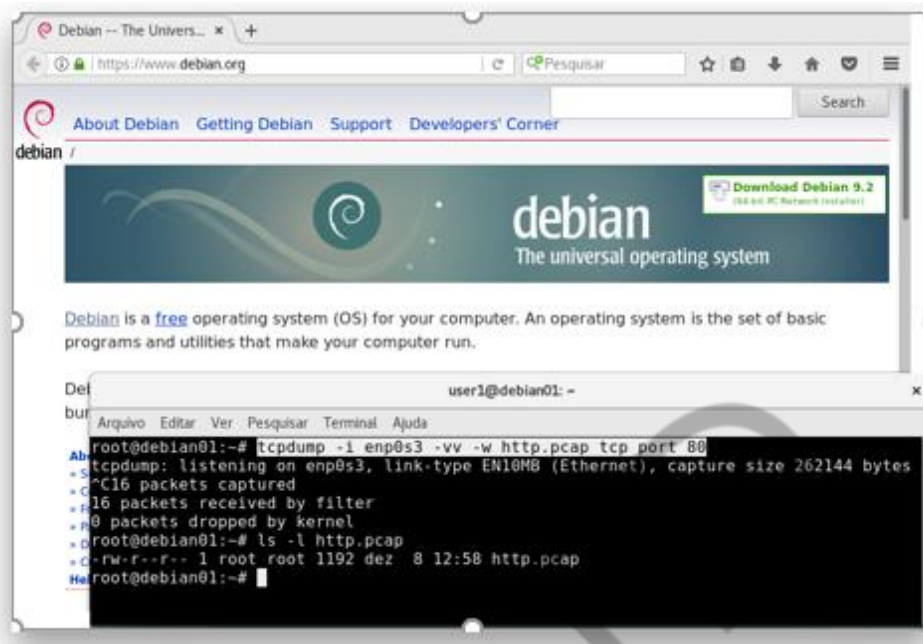


Figura 6.8 – Captura de sessão HTTP
Fonte: Elaborado pelo autor (2020)

Para que o arquivo http.pcap possa ser visualmente inspecionado, basta utilizar a linha de comando:

```
root@debian01:~# tcpdump -r http.pcap
```

Essa linha, fará com que o conteúdo do arquivo seja impresso na tela, como pode ser visto na figura adiante.

```
root@debian01:~# tcpdump -r http.pcap
reading from file http.pcap, link-type EN10MB (Ethernet)
13:12:57.662068 IP debian01.41294 > debiansec.c3sl.ufpr.br.http: Flags [S], seq 655710519
, win 29200, options [mss 1460,sackOK,TS val 4080925 ecr 0,nop,wscale 7], length 0
13:12:57.706230 IP debiansec.c3sl.ufpr.br.http > debian01.41294: Flags [S.], seq 23072000
1, ack 655710520, win 65535, options [mss 1460], length 0
13:12:57.706264 IP debian01.41294 > debiansec.c3sl.ufpr.br.http: Flags [.], ack 1, win 29
200, length 0
13:12:57.706532 IP debian01.41294 > debiansec.c3sl.ufpr.br.http: Flags [P.], seq 1:315, a
ck 1, win 29200, length 314: HTTP: GET / HTTP/1.1
13:12:57.706697 IP debiansec.c3sl.ufpr.br.http > debian01.41294: Flags [.], ack 315, win
65535, length 0
13:12:57.729876 IP debiansec.c3sl.ufpr.br.http > debian01.41294: Flags [P.], seq 1:617, a
ck 315, win 65535, length 616: HTTP: HTTP/1.1 302 Found
13:12:57.729890 IP debian01.41294 > debiansec.c3sl.ufpr.br.http: Flags [.], ack 617, win
```

Figura 6.9 – Inspeção do arquivo http.pcap
Fonte: Elaborado pelo autor (2020)

Entretanto, a impressão do conteúdo do arquivo exclusivamente na tela pode não ser suficiente. É possível observar na figura “Inspeção do arquivo http.pcap”, que a falta de formatação do arquivo, bem como, o reduzido controle sobre a navegação por seu conteúdo, podem dificultar sua análise. É possível, entretanto, redirecionar a

impressão do conteúdo do arquivo *pcap* para um arquivo texto, possibilitando, assim, a edição e formatação de seu conteúdo de maneira mais adequada. Para isso, basta ajustar essa linha de comando anterior, adicionando-se a ela um redirecionador para enviar a saída do comando a um arquivo texto, em vez da tela:

```
root@debian01:~# tcpdump -r http.pcap > http.txt
```

O símbolo maior (>) é o responsável pelo redirecionamento da saída do comando para o novo arquivo. A figura “Edição e formatação manuais do arquivo http.txt” exibe a saída do novo arquivo (http.txt) manualmente editado por meio do **VI** (pronuncia-se “VI Al”, não é o número 6 em romano) um dos editores de textos padrão do Debian, tornando-o consideravelmente mais fácil de ser lido e analisado.

```
[1] 13:12:57.662068 IP debian01.41294 > debiansec.c3sl.ufpr.br.http:
    Flags [S], seq 655710519, win 29200, options
    [mss 1460,sackOK,TS val 4080925 ecr 0,nop,wscale 7],
[2] 13:12:57.706230 IP debiansec.c3sl.ufpr.br.http > debian01.41294:
    Flags [S.], seq 230720001, ack 655710520, win 65535,
    options [mss 1460], length 0
[3] 13:12:57.706264 IP debian01.41294 > debiansec.c3sl.ufpr.br.http:
    Flags [.], ack 1, win 29200, length 0
[4] 13:12:57.706532 IP debian01.41294 > debiansec.c3sl.ufpr.br.http:
    Flags [P.], seq 1:315, ack 1, win 29200, length 314:
[5] 13:12:57.706697 IP debiansec.c3sl.ufpr.br.http > debian01.41294:
    Flags [.], ack 315, win 65535, length 0
[6] 13:12:57.729876 IP debiansec.c3sl.ufpr.br.http > debian01.41294:
    Flags [P.], seq 1:617, ack 315, win 65535, length 616:
    HTTP: HTTP/1.1 302 Found
[7] 13:12:57.729890 IP debian01.41294 > debiansec.c3sl.ufpr.br.http:
    Flags [.], ack 617, win 30184, length 0
!--- várias linhas omitidas
```

Figura 6.10 – Edição e formatação manuais do arquivo http.txt
Fonte: Elaborado pelo autor (2020)

No bloco [1] da listagem na figura “Edição e formatação manuais do arquivo http.txt”, observa-se que o *host* local (debian1) enviou uma solicitação de conexão (Flags [S]) à porta 80 do servidor (debiansec.c3sl.ufpr.br.http). Caso seja necessário apurar o endereço IPv4 destes *hosts*, basta executar a linha de comando:

```
root@debian01:~# tcpdump -n -r http.pcap | head -n1
```

Diferentemente do sinal de maior (>), o qual, conforme já discutido, redireciona a saída do TCPDump para um arquivo, o *pipe* (|) redirecionará essa saída para a

entrada do comando *head*, fazendo com que seja impressa na tela apenas a primeira linha (-n1) no início (*head*) da saída do TCPDump.

```
root@debian01:~# tcpdump -n -r http.pcap | head -n1
reading from file http.pcap, link-type EN10MB (Ethernet)
13:12:57.662068 IP 10.0.2.15.41294 > 200.17.202.197.80: Flags [S], seq 655710519,
win 29200, options [mss 1460,sackOK,TS val 4080925 ecr 0,nop,wscale 7], length 0
root@debian01:~#
```

Figura 6.11 – Verificação do endereço IPv4 dos *hosts*
Fonte: Elaborado pelo autor (2020)

Com base nas informações fornecidas pelas figuras “Edição e formatação manuais do arquivo http.txt” e “Verificação do endereço IPv4 dos *hosts*”, conclui-se, ao analisar o datagrama [1], que o endereço IPv4 do servidor é 200.17.202.197, tendo o datagrama IP *sequence number* de valor 655710519 e *window size* de 29200 bytes com tamanho máximo de segmento (mss) de 1460 bytes, não transportando nenhum dado (*length 0*).

Pelo datagrama [4], observa-se que *host* local (Debian1 – Ipv4 10.0.2.15) faz uma requisição ao servidor HTTP (HTTP: GET / HTTP/1.1), a qual é atendida pelo servidor no datagrama 6 (HTTP: HTTP/1.1 302 Found).

Uma maneira de melhor visualizar essa requisição é isolando seus respectivos datagramas do demais. Uma das diferentes maneiras possíveis de fazê-lo, é por intermédio de um filtro que isole todos os datagramas com a *flag* PUSH habilitada, uma vez que ela indica o envio imediato dos dados do *host* transmissor ao *host* receptor.

A nova linha de comando do TCPDump empregará a ferramenta *grep* (nativa dos sistemas UNIX-LIKE), a qual, de forma bastante sucinta, cria um filtro que busca a expressão fornecida como argumento no arquivo especificado. Assim sendo, a nova linha de comando para o TCPDump fica:

```
root@debian01:~# tcpdump -r http.pcap | grep -w P.
```

A figura “Filtragem de datagramas com *flag* PUSH habilitada” ilustra a saída da nova linha de comando.

```
root@debian01:~# tcpdump -r http.pcap | grep -w P.  
reading from file http.pcap, link-type EN10MB (Ethernet)  
13:12:57.706532 IP debian01.41294 > debiansec.c3sl.ufpr.br.http: Flags [P.],  
seq 1:315, ack 1, win 29200, length 314: HTTP: GET / HTTP/1.1  
13:12:57.729876 IP debiansec.c3sl.ufpr.br.http > debian01.41294: Flags [P.],  
seq 1:617, ack 315, win 65535, length 616: HTTP: HTTP/1.1 302 Found  
root@debian01:~# █
```

Figura 6.12 – Filtragem de datagramas com *flag* PUSH habilitada
Fonte: Elaborado pelo autor (2017)

Uma forma simples de identificar o host de hospedagem do *site* acessado na sessão http capturada, consiste na alteração da última linha de comando utilizada conforme indicado no destaque:

```
root@debian01:~# tcpdump -A -r http.pcap | grep -i host
```

O Linux é um sistema operacional “*case sensitive*”, ou seja, ele diferencia maiúsculas e minúsculas. A opção **-i** na ferramenta *grep* inibe essa característica, tornando a busca insensitiva, ou seja, ela validará as ocorrências da *string* *host* independentemente de haverem sido grafadas em letras maiúsculas ou minúsculas. A opção **-A** do TCPDump é bastante conveniente para análise da captura de páginas *web*, uma vez que essa opção faz com que cada pacote seja impresso em ASCII, ampliando, assim, o alcance da busca.

A figura “Descoberta do nome do *host* com TCPDump” ilustra a saída resultante do novo ajuste.

```
root@debian01:~# tcpdump -A -r http.pcap | grep -i host  
reading from file http.pcap, link-type EN10MB (Ethernet)  
Host: www.debian.org  
root@debian01:~# █
```

Figura 6.13 – Descoberta do nome do *host* com TCPDump
Fonte: Elaborado pelo autor (2020)

Entrando agora em um novo contexto, tome-se como referência o tráfego de uma sessão FTP. Para melhor percepção dos conceitos e comandos desse protocolo, sua abordagem será feita por meio da ferramenta **ftp**, anteriormente instalada com o TCPDump.

Conforme o próprio nome transparece, essa ferramenta é um cliente FTP em modo texto, daí a observação em relação à maior percepção do ocorrido durante a

sessão FTP. Entretanto, antes de se iniciar o experimento, o novo ambiente deverá ser preparado. Ele contemplará, além da VM Debian utilizada até o momento, uma nova VM: a Metasploitable2, a qual consiste de uma máquina virtual **intencionalmente vulnerável**, construída com base no UBUNTU 8.04 – codinome *hardy*, disponibilizada pela RAPID7 (www.rapid7.com) para testes e estudo de ferramentas voltadas à segurança.

A figura “Ambiente para testes FTP” representa o novo ambiente.

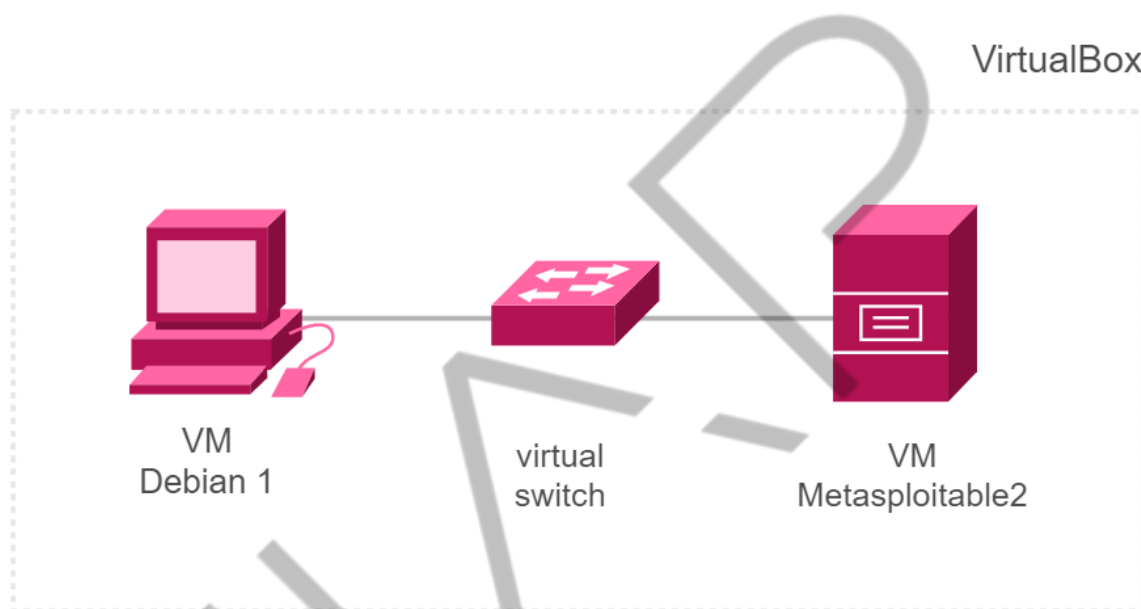


Figura 6.14 – Ambiente para testes FTP
Fonte: Elaborado pelo autor (2020)

Dadas suas vulnerabilidades características, a VM do Metasploitable 2 nunca deverá ser utilizada em produção, nem ser exposta à rede não confiável (Internet).

6.3 Implementando a metasploitable2

O procedimento para instalação da Metasploitable2 (PC2) é bastante simples:

- Criar um subdiretório denominado Metasploitable2, e tendo esse como destino, proceder ao download da VM **Metasploitable-linux-2.0.0.zip**, disponível em: <https://sourceforge.net/projects/metasploitable/files/Metasploitable2/>.
- Antes de descompactar o arquivo baixado, utilizar o aplicativo WinMD5 ou outro similar, para verificar o *hash* md5 do referido arquivo, comparando-o

ao *hash* do arquivo original fornecido pelo fabricante e geralmente disponível no *site* de *download* (8825f2509a9b9a58ec66bd65ef83167f).

- Verificada a integridade do arquivo baixado, proceder à sua descompactação no subdiretório recém-criado e de mesmo nome, confirmando a existência dos arquivos listados na figura “Descompactação da Metasploitable2”.

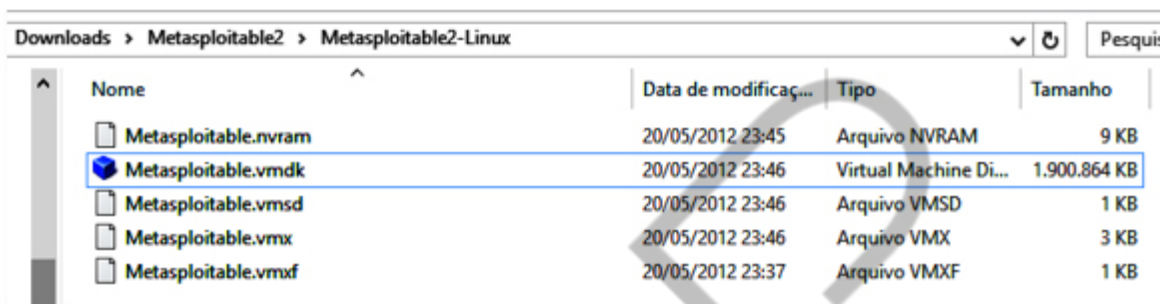


Figura 6.15 – Descompactação da Metasploitable2

Fonte: Elaborado pelo autor (2020)

- Abrir o VirtualBox, e no menu principal clicar na opção “Novo”.
- Na caixa de diálogo “Criar Máquina Virtual”, preencher os campos: [Nome] com TargetSrv, [Tipo] com Linux e Versão com “Ubuntu (64bit)”, clicando então no botão “Próximo (N)” conforme demonstrado pela figura a seguir.

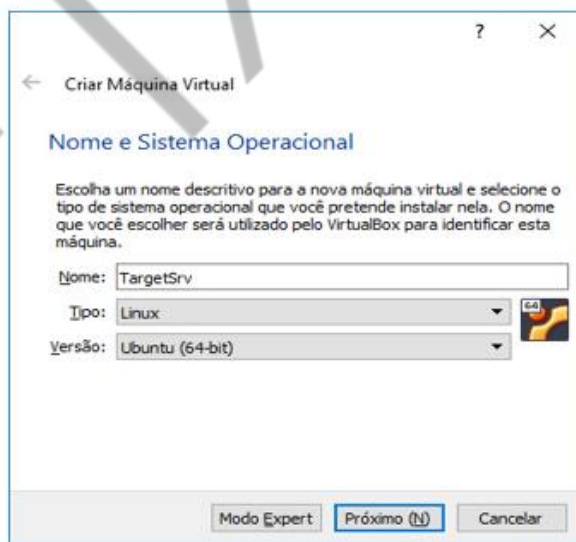


Figura 6.16 – Criação da VM TargetSrv

Fonte: Elaborado pelo autor (2020)

Ainda caixa de diálogo “Criar Máquina Virtual”, agora na página “Tamanho da memória” manter o valor sugerido de 1GB (1024MB), clicando então no botão “Próximo (N)” para prosseguir para a próxima página.

Já na página “Disco Rígido”, alterar as opções padrão conforme a figura adiante, de forma a **importar o arquivo.VMDK** disponibilizado após a descompactação do arquivo baixado. Feito isso, finalizar então, a criação da VM TargetSrv clicando no botão “Criar”.

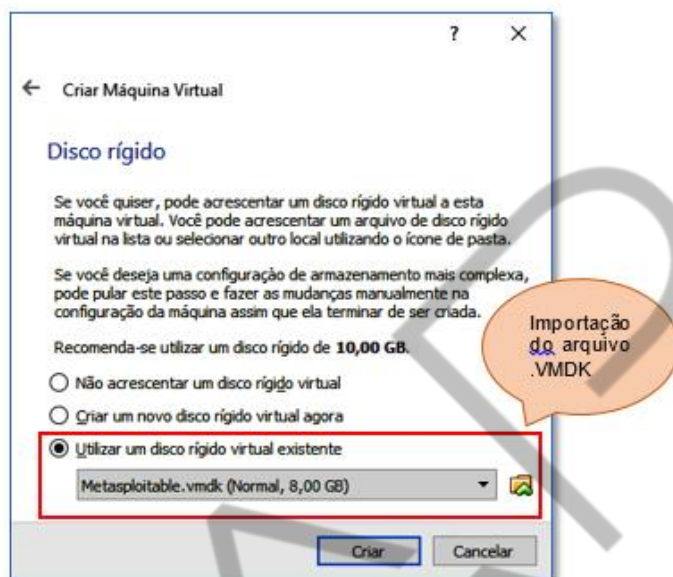


Figura 6.17 – Criação do disco rígido da VM TargetSrv
Fonte: Elaborado pelo autor (2020)

Efetivados os procedimentos anteriores, clicar no botão “Configurações”, na barra de menu e, então, desabilitar a opção “Áudio” e ajustar a opção “Rede” do adaptador 1 para modo **Bridge**. A figura a seguir mostra as configurações para provisionamento da VM TargetSrv, a qual poderá então ser inicializada, tendo-se como credenciais para login: msfadmin / msfadmin, respectivamente, como username e senha.

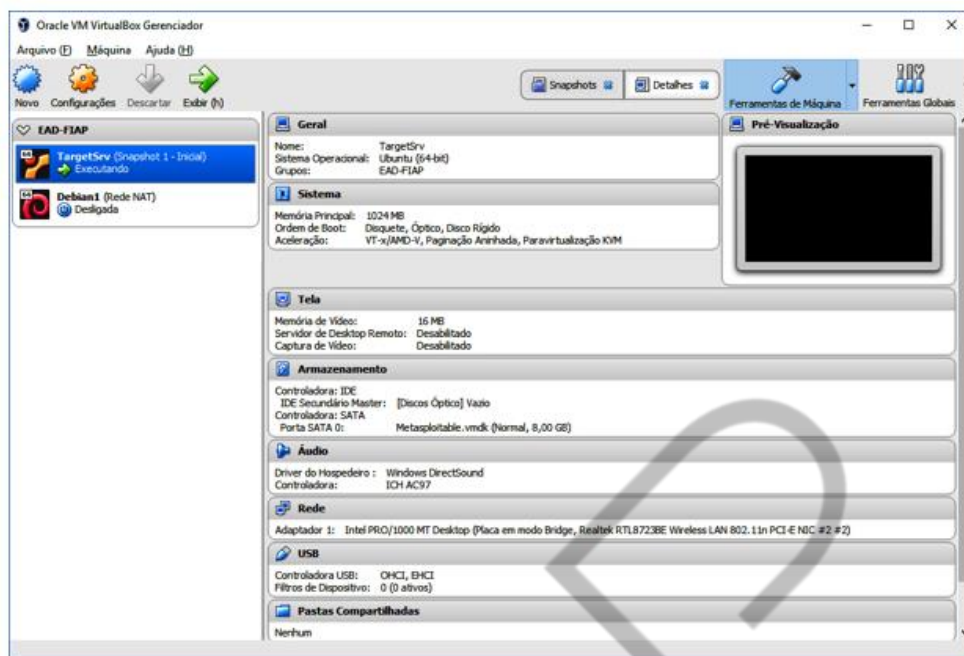


Figura 6.18 – Configuração final da Metasploitable2 (TargetSrv)
Fonte: Elaborado pelo autor (2020)

Como a Metasploitable2 se baseia no Ubuntu, **o usuário root encontra-se inicialmente desabilitado**, assim sendo, procedimentos administrativos deverão ser executados mediante o uso do comando **sudo** pelo usuário msfadmin.

6.4 Analisando uma sessão ftp

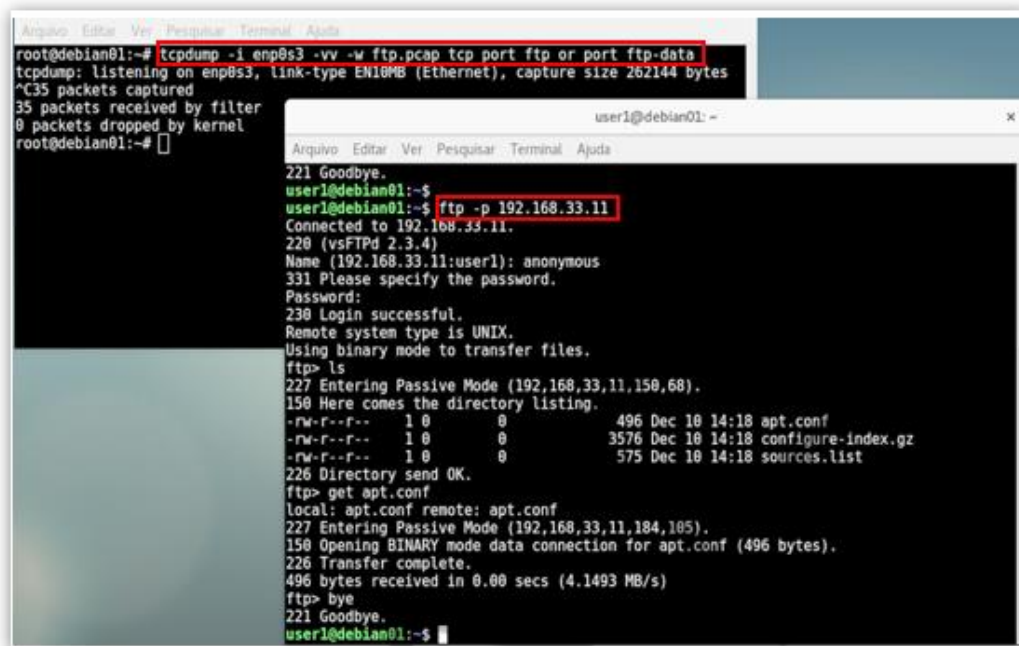
Antecedendo o início dos procedimentos para captura dos pacotes, alguns arquivos de controle deverão ser criados para utilização pelo cliente ftp. Para tal, no console do servidor TargetSRV (VM Metasploitable), como usuário desprivilegiado, executar os comandos ilustrados no destaque da figura “Criação de arquivos de controle”.

```
msfadmin@metasploitable:~$ sudo cp /usr/share/doc/apt/examples/* /home/ftp/
msfadmin@metasploitable:~$ ls -l /home/ftp/
total 12
-rw-r--r-- 1 root root 496 2017-12-10 09:18 apt.conf
-rw-r--r-- 1 root root 3576 2017-12-10 09:18 configure-index.gz
-rw-r--r-- 1 root root 575 2017-12-10 09:18 sources.list
msfadmin@metasploitable:~$ _
```

Figura 6.19 – Criação de arquivos de controle
Fonte: Elaborado pelo autor (2020)

Criados os arquivos, a partir agora, da VM Debian01, serão executados os procedimentos para captura da sessão FTP do usuário user1. Para isso, deverão ser abertos dois terminais: o primeiro como superusuário (*root*), para que o TCPDump

possa ser utilizado para captura da sessão FTP; e o segundo, como usuário regular, para uso do cliente FTP.



```
root@debian01:~# tcpdump -i enp0s3 -vv -w ftp.pcap tcp port ftp or port ftp-data
tcpdump: listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
^C35 packets captured
35 packets received by filter
0 packets dropped by kernel
root@debian01:~#

user1@debian01:~# ftp -p 192.168.33.11
Connected to 192.168.33.11.
220 (vsFTPd 2.3.4)
Name (192.168.33.11:user1): anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
227 Entering Passive Mode (192,168,33,11,150,68).
150 Here comes the directory listing.
-rw-r--r-- 1 0 0 496 Dec 10 14:18 apt.conf
-rw-r--r-- 1 0 0 3576 Dec 10 14:18 configure-index.gz
-rw-r--r-- 1 0 0 575 Dec 10 14:18 sources.list
226 Directory send OK.
ftp> get apt.conf
local: apt.conf remote: apt.conf
227 Entering Passive Mode (192,168,33,11,184,105).
150 Opening BINARY mode data connection for apt.conf (496 bytes).
226 Transfer complete.
496 bytes received in 0.00 secs (4.1493 MB/s)
ftp> bye
221 Goodbye.
user1@debian01:~#
```

Figura 6.20 – Terminais para captura da sessão FTP

Fonte: Elaborado pelo autor (2020)

Uma das maneiras para captura do tráfego FTP com o TCPDump consiste em, a partir do terminal do `root` (figura “Terminais para captura da sessão FTP”), executar a linha de comando:

```
root@debian01:~# tcpdump -i enp0s3 -vv -w ftp.pcap tcp port ftp or port ftp-data
```

Enquanto no caso de estabelecimento da sessão FTP, executam-se os comandos destacados na figura a seguir, a partir de outro terminal de usuário.

```
[1] user1@debian01:~$ ftp -p 192.168.33.11
[2] Connected to 192.168.33.11.
[3] 220 (vsFTPD 2.3.4)
[4] Name (192.168.33.11:user1): anonymous
[5] 331 Please specify the password.
[6] Password:
[7] 230 Login successful.
[8] Remote system type is UNIX.
[9] Using binary mode to transfer files.
[10] ftp> ls
[11] 227 Entering Passive Mode (192.168.33.11,150.68).
[12] 150 Here comes the directory listing.
[13] -rw-r--r-- 1 0 0 496 Dec 10 14:18 apt.conf
[14] -rw-r--r-- 1 0 0 3576 Dec 10 14:18 configure-index.gz
[15] -rw-r--r-- 1 0 0 575 Dec 10 14:18 sources.list
[16] 226 Directory send OK.
[17] ftp> get apt.conf
[18] local: apt.conf remote: apt.conf
[19] 227 Entering Passive Mode (192.168.33.11,184.105).
[20] 150 Opening BINARY mode data connection for apt.conf (496 bytes).
[21] 226 Transfer complete.
[22] 496 bytes received in 0.00 secs (4.1493 MB/s)
[23] ftp> bye
[24] 221 Goodbye.
user1@debian01:~$
```

Figura 6.21 – Utilização do cliente FTP em modo texto

Fonte: Elaborado pelo autor (2020)

Na linha [1], o usuário *user1* inicia uma sessão *ftp* passiva (*-p*) com o *host* 192.168.33.11. Pela linha [4] é possível verificar que o servidor está configurado como *FTP* anônimo e, portanto, o usuário autenticou-se no serviço com *username* *anonymous*, sem necessidade de senha. Havendo logado com sucesso, conforme mensagem do servidor na linha [7], o usuário lista o conteúdo do diretório [10] e faz *download* do arquivo *apt.conf* [17], finalizando a sessão em seguida [23].

Finalizada a sessão do usuário, o arquivo de captura deverá ser gravado no disco, devendo retornar ao terminal do *root* e digitar, então, CTRL+C.

Gravado o arquivo *ftp.pcap*, uma cópia dele deverá ser feita no diretório do usuário (*/home/user1*), e o terminal do *root* fechado, uma vez que a análise do arquivo será executada nessa oportunidade, por meio do Wireshark – cujo uso não é recomendado com privilégios administrativos (do *root*).

Embora o *TCPDump* também seja capaz de fazer uma análise do arquivo, geralmente, o Wireshark acaba se mostrando mais produtivo e amigável para o usuário, dada sua interface gráfica e as múltiplas funcionalidades específicas dos analisadores dos protocolos. Cabe observar aqui, que em casos em que a captura do tráfego precisar ser minuciosa, o *TCPDump* poderá ser a opção mais indicada;

enquanto, no caso de a análise do tráfego tiver de ser mais aprofundada e ágil, o Wireshark poderá se mostrar mais adequado.

A partir do terminal do usuário, o Wireshark poderá ser invocado para análise do arquivo *ftp.pcap*, simplesmente digitando-se:

```
root@debian01:~# wireshark ftp.pcap &
```

A figura a seguir ilustra os pacotes capturados.

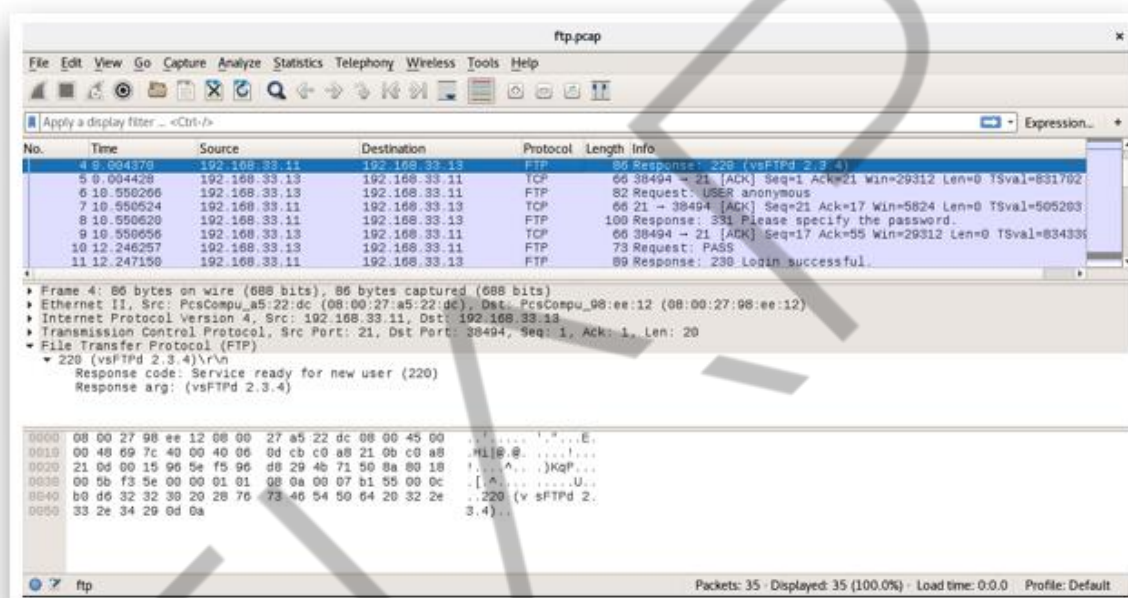


Figura 6.22 – Abertura de arquivo de captura com o Wireshark
Fonte: Elaborado pelo autor (2020)

A análise do pacote 6 indica que o *username* utilizado para a sessão foi *anonymous* (Request: USER *anonymous*), e ainda, no pacote 10, observa-se que o *login* foi feito sem senha (Request: PASS), ou seja, como já abordado, o FTP é um protocolo conhecido por graves vulnerabilidades, decorrentes, especialmente, do fato de ser um protocolo originalmente *plain-text*.

Entretanto, a captura das credenciais do usuário não é a única ameaça que esse protocolo oferece à segurança da informação. O pacote 16 indica que o modo assumido para a sessão é o ftp passivo, o que é confirmado pelo pacote 17; podendo ser observado ainda, que a primeira solicitação do usuário ao serviço, foi a listagem do conteúdo do diretório (figura “Tipo de sessão FTP”).

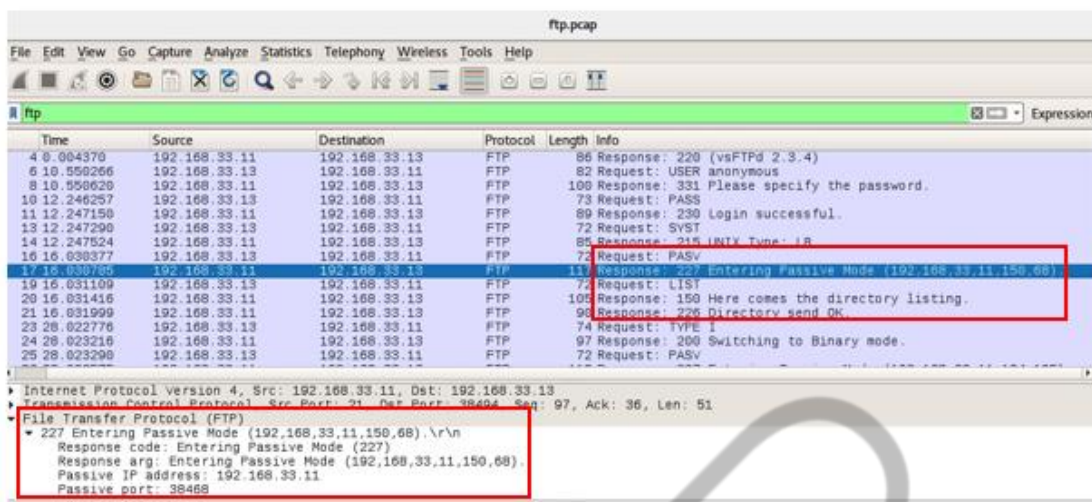


Figura 6.23 – Tipo de sessão FTP
Fonte: Elaborado pelo autor (2020)

Uma das maneiras para verificar se o usuário procedeu ao *download* de algum arquivo é, a partir da opção Edit - na barra do menu principal, clicar em Find Packet, e na barra de menu secundário que surge, clicar no (terceiro) menu *pull-down* e escolher a opção String.

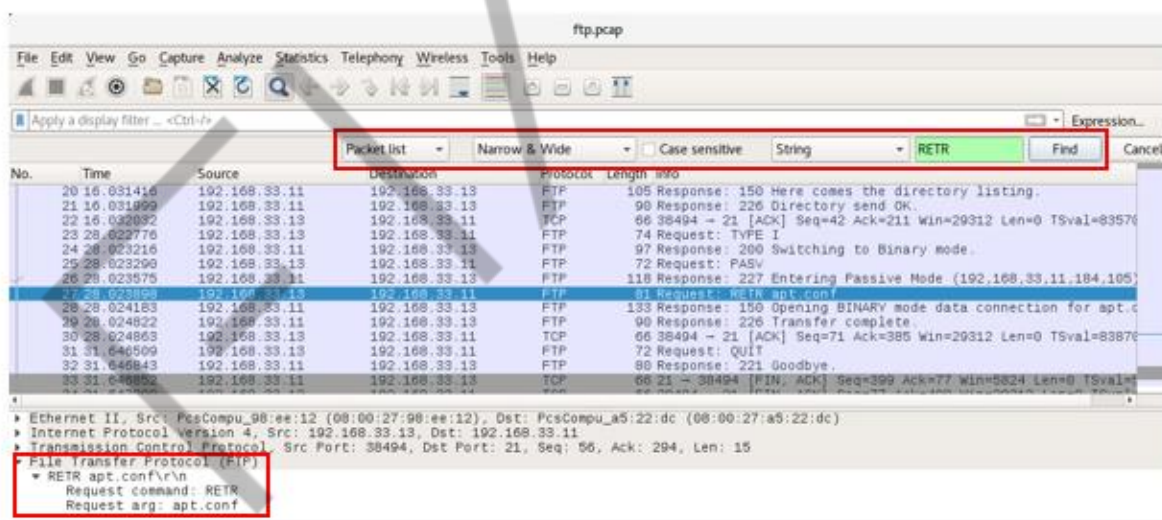


Figura 6.24 – Filtragem de pacotes baseada em strings
Fonte: Elaborado pelo autor (2020)

O comando RETR é o responsável por recuperar (baixar) arquivos do servidor, assim sendo, ao ser fornecido como *string* de busca do filtro construído, ele destaca o pacote 27, o qual claramente indica que o usuário solicitou o arquivo apt.conf (Request: RETR apt.conf).

Clicando-se com o botão direito sobre o pacote 28, e depois nas opções **Follow > TCP Stream**, uma nova caixa de diálogo surgirá, detalhando em ASCII toda a sequência do ocorrido (figura “Sequência da sessão FTP”).

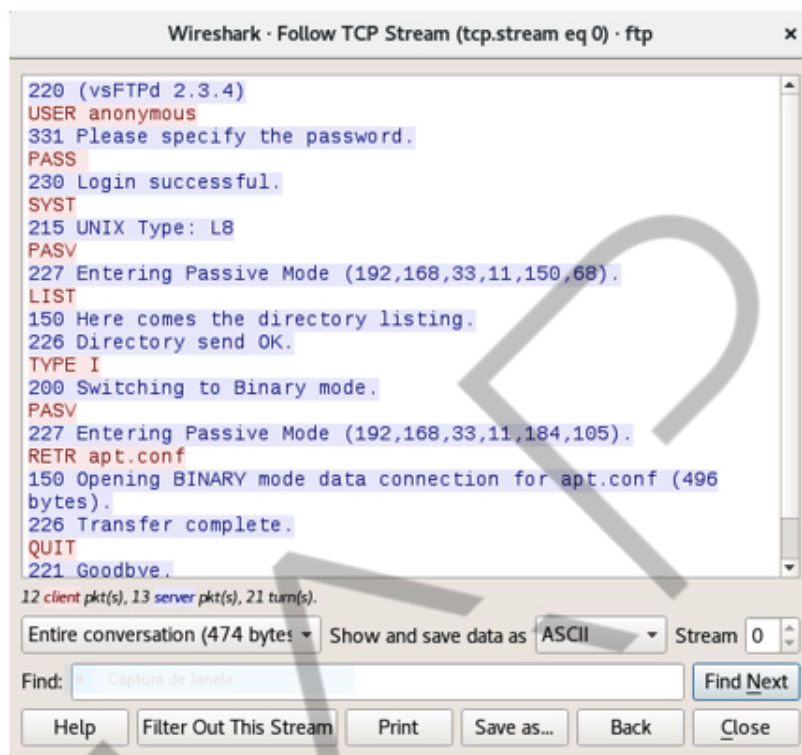


Figura 6.25 – Sequência da sessão FTP

Fonte: Elaborado pelo autor (2020)

Se necessário, é possível inclusive a recuperação de cópia do conteúdo do arquivo baixado pelo usuário, mais uma vez ressaltando, entretanto, que isso deve ser feito com o devido cuidado para que não seja caracterizada alguma conduta antijurídica (por exemplo, invasão da privacidade do usuário) ou, mesmo, alguma conduta que venha a ferir as políticas de segurança da empresa (por exemplo, decorrente da quebra de confidencialidade).

Para isso, basta uma pequena alteração na última linha de comando utilizada:

```
root@debian01:~# tcpdump -i enp0s3 -vv -w ftp.pcap tcp &
```

Ou seja, como a sessão FTP foi acordada entre os *hosts* para modo passivo, é sabido (do Capítulo 5) que o canal de transferência de dados (FTP-DATA) será estabelecido por iniciativa do cliente, o qual para download do arquivo, estabelecerá uma nova conexão para o servidor, em porta desprivilegiada a ser por ele especificada, sem utilização da porta 20, reservada pelo IANA para tal finalidade.

Assim sendo, a linha de comando originalmente aplicada à captura do tráfego garante, nesse caso específico, a aquisição, unicamente, dos dados de controle da sessão FTP (port ftp or ftp-data), portanto, impedindo que a confidencialidade da mensagem e a privacidade do usuário, venham a ser feridas. Exceção, naturalmente, às credenciais do usuário, as quais, entretanto, constituem objetos naturais de qualquer auditoria, e dessa forma, portanto, normalmente contempladas pelas políticas de segurança.

Já a última linha de comando proposta, elimina esse “mecanismo de proteção” involuntariamente implementado, na medida em que ocorrerá a captura de todo o tráfego TCP, incluindo o que fluir por portas desprivilegiadas (a serem empregadas pelo canal de dados).

Do exposto, tem-se uma breve, porém relevante, demonstração da importância do sólido conhecimento dos protocolos de serviço e comunicações. Procedimentos que coletam informações de tráfego que possam, eventualmente, impactar sobre as políticas de segurança ou dispositivos jurídicos devem, em princípio, ser adotados predominantemente para fins forenses.

A figura adiante ilustra a captura de pacotes feita com a nova linha de comando.

No.	Time	Source	Destination	Protocol	Length	Info
35	30.958101	192.168.33.13	192.168.33.11	FTP	72	Request: PASV
36	30.958358	192.168.33.11	192.168.33.13	FTP	118	Response: 227 Entering Passive Mode (192.168.33.11, 47591)
37	30.958434	192.168.33.13	192.168.33.11	TCP	74	34212 → 47591 [SYN] Seq=0 Win=29200 Len=0 MSS=1460
38	30.958595	192.168.33.11	192.168.33.13	TCP	74	47591 → 34212 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0
39	30.958607	192.168.33.13	192.168.33.11	TCP	66	34212 → 47591 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSv...
40	30.958686	192.168.33.13	192.168.33.11	FTP	81	Request: RETR apt.conf
41	30.959025	192.168.33.11	192.168.33.13	FTP	133	Response: 150 Opening BINARY mode data connection f...
42	31.001912	192.168.33.13	192.168.33.11	TCP	66	38676 → 21 [ACK] Seq=71 Ack=362 Win=29312 Len=0 TSv...
43	31.040572	192.168.33.11	192.168.33.13	FTP-DATA	562	FTP Data: 496 bytes
44	31.040588	192.168.33.11	192.168.33.13	TCP	66	47591 → 34212 [FIN, ACK] Seq=497 Ack=1 Win=5824 Len=0
45	31.040648	192.168.33.13	192.168.33.11	TCP	66	34212 → 47591 [ACK] Seq=1 Ack=497 Win=30336 Len=0 TS...
46	31.040806	192.168.33.13	192.168.33.11	TCP	66	34212 → 47591 [FIN, ACK] Seq=1 Ack=498 Win=30336 Len=0
47	31.040971	192.168.33.11	192.168.33.13	TCP	66	47591 → 34212 [ACK] Seq=498 Ack=2 Win=5824 Len=0 TS...
48	31.041158	192.168.33.11	192.168.33.13	FTP	90	Response: 226 Transfer complete.
49	31.041200	192.168.33.13	192.168.33.11	TCP	66	38676 → 21 [ACK] Seq=71 Ack=386 Win=29312 Len=0 TSv...
50	36.528262	192.168.33.13	192.168.33.11	FTP	72	Request: QUIT
51	36.528560	192.168.33.11	192.168.33.13	FTP	80	Response: 221 Goodbye.
52	36.528568	192.168.33.11	192.168.33.13	TCP	66	21 → 38676 [FIN, ACK] Seq=400 Ack=77 Win=5824 Len=0
53	36.528665	192.168.33.13	192.168.33.11	TCP	66	38676 → 21 [ACK] Seq=77 Ack=400 Win=29312 Len=0 TSv...
54	36.528932	192.168.33.13	192.168.33.11	TCP	66	38676 → 21 [FIN, ACK] Seq=77 Ack=401 Win=29312 Len=0
55	36.529093	192.168.33.11	192.168.33.13	TCP	66	21 → 38676 [ACK] Seq=401 Ack=78 Win=5824 Len=0 TSv...

▶ Frame 36: 118 bytes on wire (944 bits), 118 bytes captured (944 bits)
 ▶ Ethernet II, Src: PcsCompu_a5:22:dc (08:00:27:a5:22:dc), Dst: PcsCompu_98:ee:12 (08:00:27:98:ee:12)
 ▶ Internet Protocol Version 4, Src: 192.168.33.13, Dst: 192.168.33.11
 ▶ Transmission Control Protocol, Src Port: 21, Dst Port: 38676, Seq: 243, Ack: 56, Len: 52
 ▶ File Transfer Protocol (FTP)
 ▶ 227 Entering Passive Mode (192.168.33.11, 47591).
 Response code: Entering Passive Mode (227)
 Response arg: Entering Passive Mode (192.168.33.11, 47591).
 Passive IP address: 192.168.33.11
 Passive port: 47591

Figura 6.26 – Conexão FTP-DATA

Fonte: Elaborado pelo autor (2020)

Da figura “Conexão FTP-DATA”, observa-se que o cliente (192.168.33.13) envia ao servidor (192.168.33.11) um pacote (35) com a instrução PASV (modo passivo), recebendo como resposta do servidor o número da porta à qual deverá se

conectar: 47591 (ver destaque na figura “Conexão FTP-DATA”). Os pacotes 37, 38 e 39 descrevem o 3WAY Handshake que estabelece a conexão do canal de dados (FTP-DATA). O pacote 40 exibe a solicitação de *download* do arquivo apt.conf (Request: RETR apt.conf) feita pelo cliente, o qual é enviado pelo servidor, tendo tamanho de 496 bytes, conforme pode-se verificar pelo pacote 43.

Finalizada a transmissão do arquivo, os pacotes 44 a 47 mostram o fechamento da conexão dados. De forma análoga ao anteriormente feito, ao clicar com o botão direito sobre algum desses pacotes (44 a 47), e na sequência, nas opções Follow > TCP Stream (figura “Caminho para visualização do conteúdo do arquivo”) das novas caixas de diálogo que surgem, é possível visualizar o conteúdo do arquivo baixado pelo usuário (figura “Gravação do conteúdo do arquivo apt.conf”).

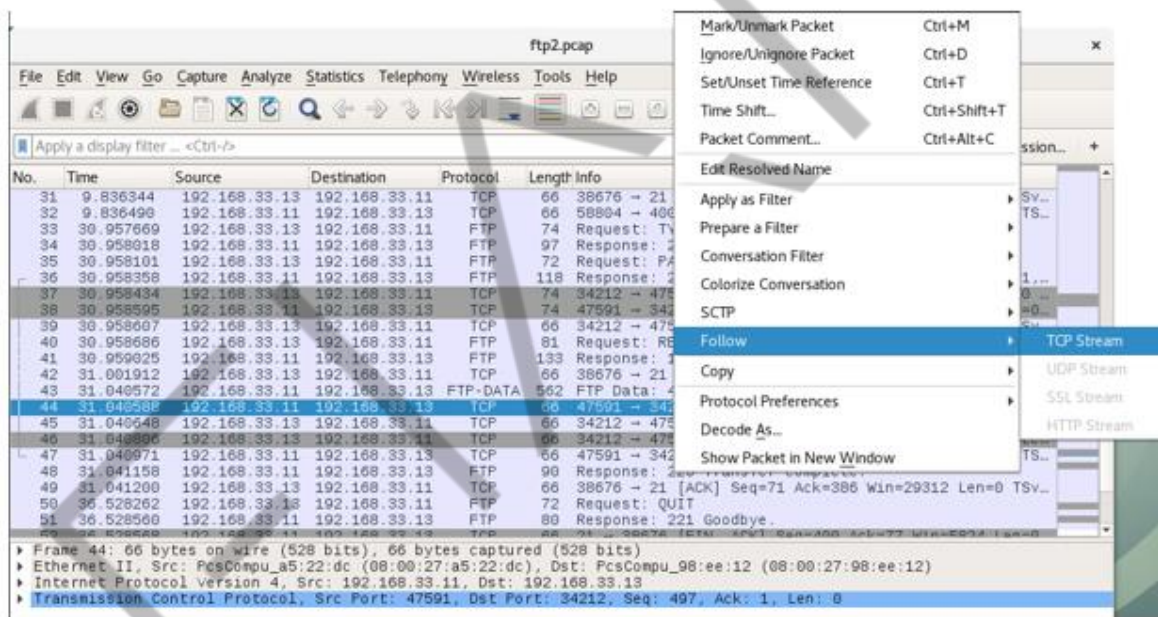


Figura 6.27 – Caminho para visualização do conteúdo do arquivo

Fonte: Elaborado pelo autor (2020)

A figura “Gravação do conteúdo do arquivo apt.conf” ilustra parcialmente o conteúdo do arquivo apt.conf, baixado pelo usuário em sua sessão FTP. Para salvar o conteúdo do arquivo, basta clicar no botão Save As, porém, alterando a opção ASCII para UTF-8, conforme exibido no destaque.

Observe-se que a verdadeira relevância do estudado neste capítulo não recai sobre o uso das ferramentas propriamente dito, mas, sim, sobre a estruturação lógica de suas linhas de comando e a correta interpretação dos resultados por elas apresentados.

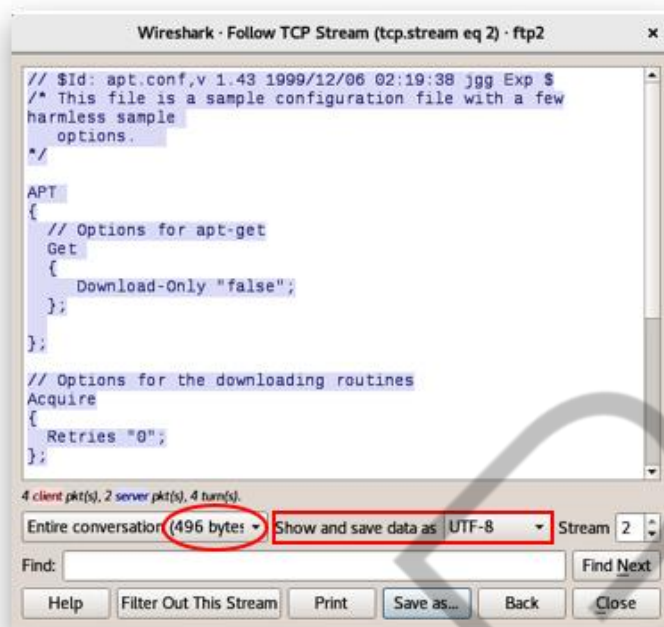


Figura 6.28 – Gravação do conteúdo do arquivo apt.conf
Fonte: Elaborado pelo autor (2020)

REFERÊNCIAS

BLANK, A.G. **TCP/IP Foundations**. Alameda: Sybex Inc., 2004.

FARREL, A. **A Internet e seus protocolos - Uma análise comparativa**. Rio de Janeiro: Elsevier, 2005.

FILHO, João Eriberto Mota. **Análise de tráfego em redes TCP/IP**. São Paulo: Novatec, 2013.

HUNT, C. **TCP/IP Network Administration**. 3. ed. Sebastopol: O'Reilly Media, 2010.

JUNIPER. **Understanding Port Mirroring**. 2016. Disponível em: <https://www.juniper.net/documentation/en_US/junos/topics/concept/port-mirroring-qfx-series-understanding.html>. Acesso em: 20 abr. 2020.

_____. **Analysis of Network Packets**. [s.d.]. Disponível em: <http://www.iitg.ernet.in/cse/ISEA/isea_PPT/ISEA_02_09/Analysis%20of%20Network%20Packets.pdf>. Acesso em: 20 abr. 2020.

KESSLER, Gary C. **On Teaching TCP/IP Protocol Analysis to Computer Forensics Examiners**. 2008. Disponível em: <<https://commons.erau.edu/cgi/viewcontent.cgi?referer=https://www.google.com.br/&httpsredir=1&article=1003&context=db-applied-aviation>>. Acesso em: 20 abr. 2020.

NORTHCUTT, Stephen. **Traffic Analysis**. 2007. Disponível em: <<https://www.sans.edu/cyber-research/security-laboratory/article/traffic-analysis>>. Acesso em: 20 abr. 2020.

PETERSON, L. L.; Davie, R. S. **Redes de Computadores – Uma abordagem de sistemas**. 3. ed. Rio de Janeiro: Elsevier, 2004.

GLOSSÁRIO

timestamp	Sequência de caracteres ou informações codificadas que identificam quando um determinado evento ocorreu, com precisão variável, em função da necessidade.
API	Acrônimo de "Application Programming Interface" (Interface de Programação de Aplicativos) consiste de um conjunto de rotinas e padrões de programação para acesso a um aplicativo de software ou plataforma baseado na Web.
Open Source	Open Source (ou Código Aberto) é um modelo de desenvolvimento que promove um licenciamento livre para o design ou esquematização de um produto, e a redistribuição universal desse design ou esquema, dando a possibilidade para que qualquer um consulte, examine ou modifique o produto.
UTF-8	Unicode Transformation Format 8-bit é uma codificação de comprimento variável capaz de representar cada caractere no conjunto de caracteres Unicode, sendo também compatível com o ASCII. Está gradativamente sendo empregado como tipo de codificação padrão para e-mails, páginas web, e outros.