

Relatório

1. Introdução

O trabalho atual consiste em analisar o tráfego de uma rede para identificar e indicar a ocorrência de conexões, transferências de dados e finalizações de conexões, utilizando-se da ferramenta Wireshark.

Wireshark é um programa que analisa o tráfego de rede, e o organiza por protocolos. Tal ferramenta é capaz de registrar pacotes que trafegam pelas redes, assim como as informações dos pacotes. ^[1] O foco do trabalho corrente será no protocolo TCP e HTTP.

2. Descrição do Funcionamento

2.1. Estabelecimento da Conexão

Para estabelecer uma conexão, o TCP usa um handshake (aperto de mão) de três vias. Antes que o cliente tente se conectar com o servidor, o servidor deve primeiro ligar e escutar a sua própria porta, para só depois abri-la para conexões: isto é chamado de abertura passiva. Uma vez que a abertura passiva esteja estabelecida, um cliente pode iniciar uma abertura ativa. Para estabelecer uma conexão, o aperto de mão de três vias (ou 3 etapas) é realizado:

1. SYN: A abertura ativa é realizada por meio do envio de um SYN pelo cliente ao servidor. O cliente define o número de sequência de segmento como um valor aleatório A.

2. SYN+ACK: Em resposta, o servidor responde com um SYN-ACK. O número de reconhecimento (acknowledgment) é definido como sendo um a mais que o número de sequência recebido, i.e. A+1, e o número de sequência que o servidor escolhe para o pacote é outro número aleatório B.

3. ACK: Finalmente, o cliente envia um ACK de volta ao servidor. O número de sequência é definido ao valor de reconhecimento recebido, i.e. A+1, e o número de reconhecimento é definido como um a mais que o número de sequência recebido, i.e B+1. ^[2]

2.2. Transferência de Dados

O TCP é um protocolo de nível da camada de transporte (camada 4) do Modelo OSI (Figura 1) e é sobre o qual que se assentam a maioria das aplicações cibernéticas, como o SSH, FTP, HTTP — portanto, a World Wide Web. O Protocolo de controle de transmissão provê confiabilidade, entrega na sequência correta e verificação de erros dos pacotes de dados, entre os diferentes nós da rede, para a camada de aplicação. ^[2]

Existem permanentemente um par de números de sequência, doravante referidos como ACK. O emissor determina o seu próprio número de sequência e o receptor confirma o segmento usando como número ACK o número de sequência do emissor. Para manter a confiabilidade, o receptor confirma os segmentos indicando que recebeu um determinado número de bytes contíguos. ^[2]

A remontagem ordenada dos segmentos é feita usando os números de sequência, de 32 bit, que reiniciam a zero quando ultrapassam o valor máximo, $2^{31}-1$, tomando o valor da diferença. ^[2]

O campo checksum permite assegurar a integridade do segmento. Este campo é expresso em complemento para um consistindo na soma dos valores (em complemento para um) da trama. A escolha da operação de soma em complemento para um deve-se ao fato desta poder ser calculada da mesma forma para múltiplos desse comprimento - 16 bit, 32 bit, 64 bit... - e o resultado, quando encapsulado, será o mesmo. A verificação deste campo por parte do receptor é feita com a recálculo da soma em complemento para um que dará -0 caso o pacote tenha sido recebido intacto. ^[2]

A camada de Acesso à rede especifica a forma que os dados devem ser encaminhados independentemente do tipo de rede utilizado, a camada da Internet é encarregada fornecer o pacote de dados (datagrama), a camada do Transporte garante o encaminhamento dos dados, assim como os mecanismos que permitem conhecer o estado da transmissão e a camada de aplicação agrupa os a padrão da rede (Telnet, SMTP, FTP, etc.). ^[3]

O Hypertext Transfer Protocol, sigla HTTP (Protocolo de Transferência de Hipertexto) é um protocolo de comunicação na camada de aplicação segundo o Modelo OSI, (Figura 1) utilizado para sistemas de informação de hipermídia, distribuídos e colaborativos. Ele é a base para a comunicação de dados da World Wide Web. ^[4]

Hipertexto é o texto estruturado que utiliza ligações lógicas (hiperlinks) entre nós contendo texto. O HTTP é o protocolo para a troca ou transferência de hipertexto. ^[4]

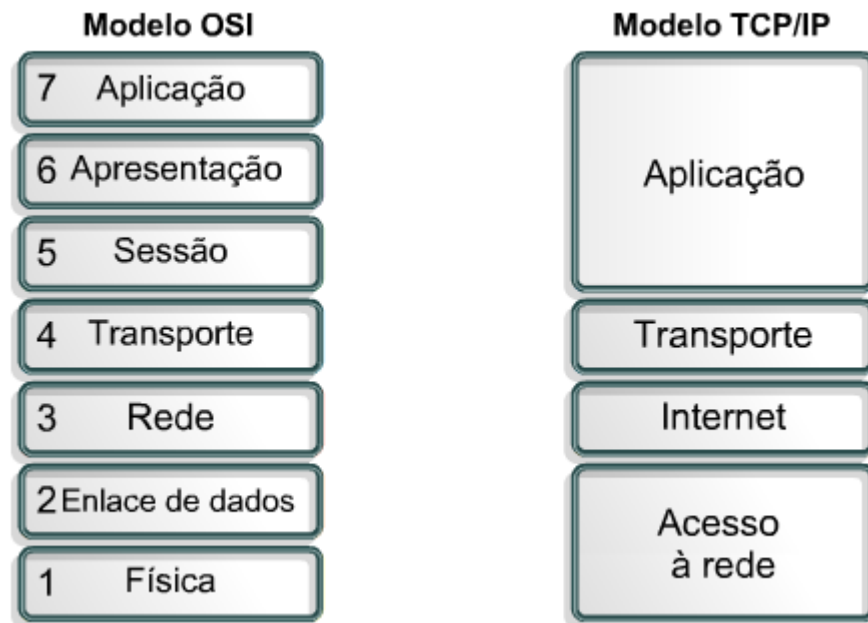


Figura 1 Modelo OSI e suas 7 camadas e modelo TCP/IP e suas 4 camadas.

2.3. Finaliza  o da Conex  o

A fase de encerramento da sess  o TCP    um processo de quatro fases, em que cada interlocutor se responsabiliza pelo encerramento do seu lado da liga  o. Quando um deles pretende finalizar a sess  o, envia um pacote com a flag FIN ativa, ao qual dever   receber uma resposta ACK. Por sua vez, o outro interlocutor ir   proceder da mesma forma, enviando um FIN ao qual dever   ser respondido um ACK. ^[2]

3. Desenvolvimento

Para a realiza  o do trabalho corrente, foi selecionado para an  lise a URL <http://ocsp.digicert.com/MFEwTzBNMEswSTAJBgUrDgMCGgUAABBTBL0V27RVZ7LBduom%2FnYB45SPUEwQU5Z1ZMIJHWMys%2BghUNoZ7OrUETfACEAiIzVJfGSRETRS1gpHeuVI%3d>.

Depois da execu  o do v  deo, Wireshark foi interrompido resultando em 120105 pacotes totais. Visto que est   sendo analisado os protocolos TCT e HTTP, usaremos filtro para explorar tais protocolos.

3.1. Estabelecimento da Conexão

Primeiramente há a conexão com o servidor, tal conexão pode ser vista na Figura 2. A primeira etapa consiste no pedido inicial (flecha verde) em seguida há a confirmação do pedido (flecha azul) e finalmente a reconfirmação, estabilizando a conexão (flecha vermelha).

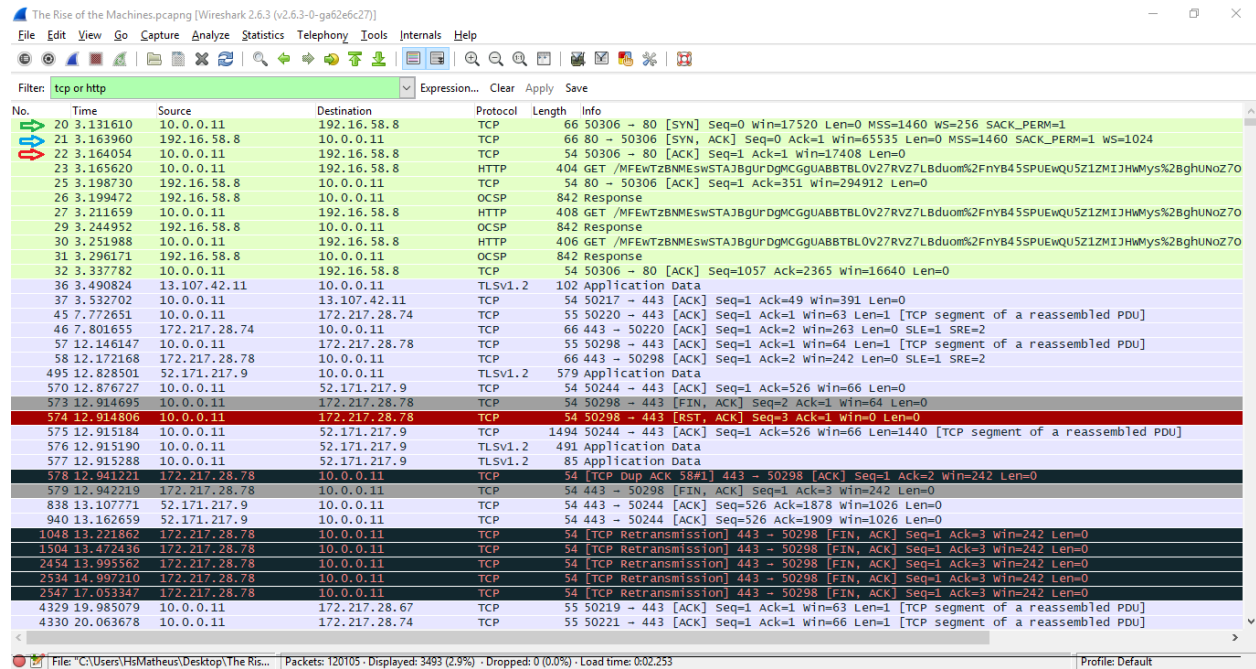


Figura 2 Estabilização da conexão.

Primeira etapa (Figura 3): realizado o pedido inicial com a flag SYN, com um valor aleatório. Retângulo verde é a origem do pedido, o cliente. Retângulo azul é o destino do pacote, o servidor. Retângulo vermelho é o protocolo utilizado. Retângulo laranja é a porta utilizada pela camada de aplicação TCP/IP. Retângulo amarelo é o valor da flag SYN.

```
20 3.131610 10.0.0.11 192.16.58.8 TCP 66 50306 → 80 [SYN] Seq=0 Win=17520 Len=0 MSS=1460 WS=256 SACK_PERM=1
+ Frame 20: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
+ Ethernet II, Src: IntelCor_f2:56:5c (60:57:18:f2:56:5c), Dst: Netgear_58:62:b8 (44:94:fc:58:62:b8)
+ Internet Protocol Version 4, Src: 10.0.0.11, Dst: 192.16.58.8
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
+ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 52
  Identification: 0x5dc8 (24008)
+ Flags: 0x4000, Don't fragment
  Time to live: 128
  Protocol: TCP (6)
  Header checksum: 0x98d8 [validation disabled]
  [Header checksum status: Unverified]
  Source: 10.0.0.11
  Destination: 192.16.58.8
+ Transmission Control Protocol, Src Port: 50306, Dst Port: 80, Seq: 0, Len: 0
  Source Port: 50306
  Destination Port: 80
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence number: 0 (relative sequence number)
  [Next sequence number: 0 (relative sequence number)]
  Acknowledgment number: 0
  1000 .... = Header Length: 32 bytes (8)
```

Figura 3 Primeira etapa.

Segunda etapa (Figura 4): Caso disponível, o servidor confirma o pedido enviando um SYN e um ACK, onde o valor do ACK é o valor recebido mais um. Retângulo verde é a origem do pacote, o servidor. Retângulo azul é o destino do pacote, o cliente. Retângulo vermelho é o protocolo utilizado. Retângulo laranja é a porta utilizada pela camada de aplicação TCP/IP. Retângulo amarelo é o valor da flag SYN. Retângulo roxo é o valor da flag ACK.

```
21 3.163960 192.16.58.8 10.0.0.11 TCP 66 80 → 50306 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1 WS=1024
+ Frame 21: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
+ Ethernet II, Src: Netgear_58:62:b8 (44:94:fc:58:62:b8), Dst: IntelCor_f2:56:5c (60:57:18:f2:56:5c)
+ Internet Protocol Version 4, Src: 192.16.58.8, Dst: 10.0.0.11
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
+ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 52
  Identification: 0x0dce (3534)
+ Flags: 0x0000
  Time to live: 53
  Protocol: TCP (6)
  Header checksum: 0x73d3 [validation disabled]
  [Header checksum status: Unverified]
  Source: 192.16.58.8
  Destination: 10.0.0.11
+ Transmission Control Protocol, Src Port: 80, Dst Port: 50306, Seq: 0, Ack: 1, Len: 0
  Source Port: 80
  Destination Port: 50306
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence number: 0 (relative sequence number)
  [Next sequence number: 0 (relative sequence number)]
  Acknowledgment number: 1 (relative ack number)
  1000 .... = Header Length: 32 bytes (8)
```

Figura 4 Segunda etapa.

Terceira etapa (Figura 5): Para estabelecer a conexão o cliente retorna um ACK, com SYN possuindo o mesmo número do ACK recebido pelo servidor. Retângulo verde é a origem do

pacote, o servidor. Retângulo azul é o destino do pacote, o cliente. Retângulo vermelho é o protocolo utilizado. Retângulo laranja é a porta utilizada pela camada de aplicação TCP/IP. Retângulo amarelo é o valor da flag SYN. Retângulo roxo é o valor da flag ACK.

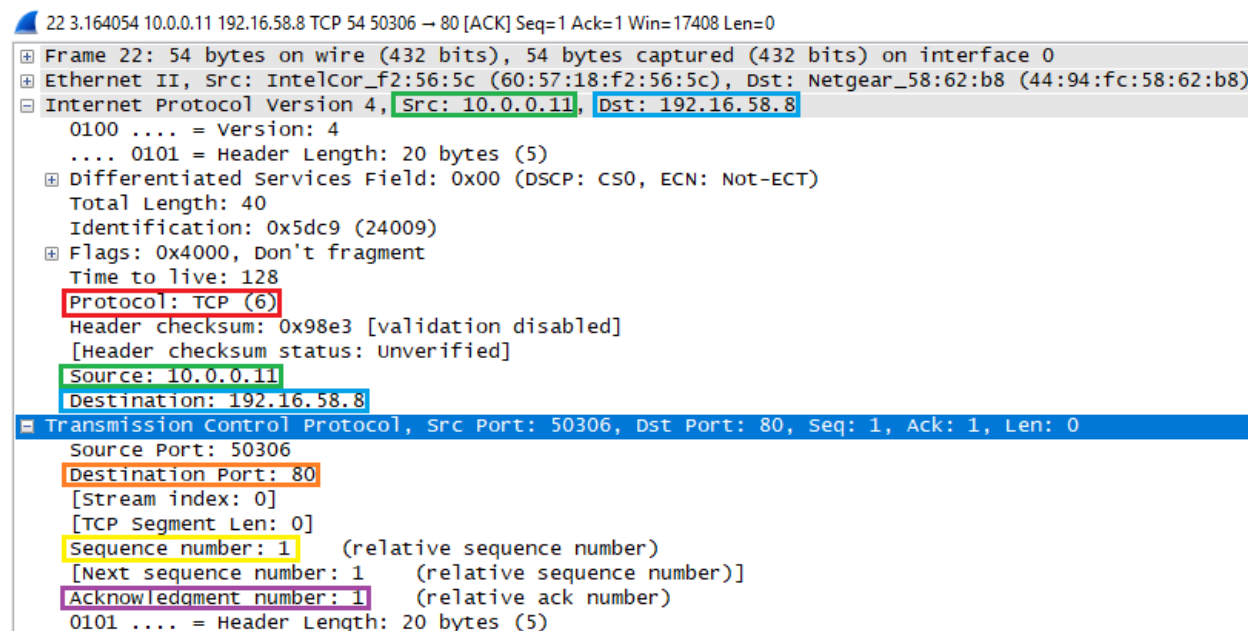


Figura 5 Terceira etapa.

3.2. Transferência de Dados

Logo que a conexão é estabilizada entre o navegador e o servidor, o browser realiza o pedido com a estrutura demonstrada na Figura 6. Retângulo verde mostra o método utilizado. Retângulo amarelo mostra a URL. Retângulo azul mostra o protocolo utilizado. Retângulo vermelho mostra a versão do protocolo utilizado.

No.	Time	Source	Destination	Protocol	Length	Info
20	3.131610	10.0.0.11	192.16.58.8	TCP	66	50306 → 80 [SYN] Seq=0 Win=17520 Len=0 MSS=1460 WS=256 SACK_PERM=1
21	3.163960	192.16.58.8	10.0.0.11	TCP	66	80 → 50306 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 SACK_PERM=1 WS=1024
22	3.164054	10.0.0.11	192.16.58.8	TCP	54	50306 → 80 [ACK] Seq=1 Ack=1 Win=17408 Len=0
23	3.165620	10.0.0.11	192.16.58.8	HTTP	404	GET /WPContent/Images/TAJBaU-Da%CGaUBBTL8VZ7RVZ7L8duom%2FnyB455PUeW0USZ1ZHIJHh#ys%28eHUnoZ70rUETFACEA11zVJfG5RETRS1apHeuV1%30 HTTP/1.1
25	3.198730	192.16.58.8	10.0.0.11	TCP	54	80 → 50306 [ACK] Seq=1 Ack=351 Win=294912 Len=0

Figura 6 Estrutura pacote HTTP.

Detalhes sobre o pacote HTTP da Figura 6 acima, podem ser visualizados na Figura 7 abaixo. Retângulo verde mostra que ao selecionar uma das linhas de comando do pacote, na divisão inferior da Figura 7 é destacado a informação sobre tal linha, em hexadecimal (retângulo azul).

```

> Frame 23: 404 bytes on wire (3232 bits), 404 bytes captured (3232 bits) on interface 0
> Ethernet II, Src: IntelCor_f2:56:5c (60:57:18:f2:56:5c), Dst: Netgear_58:62:b8 (44:94:fc:58:62:b8)
> Internet Protocol Version 4, Src: 10.0.0.11, Dst: 192.16.58.8
> Transmission Control Protocol, Src Port: 50306, Dst Port: 80, Seq: 1, Ack: 1, Len: 350
  Hypertext Transfer Protocol
    > GET /MFewTzBNMEswSTAjBgUrDg/MCGgUA8BTBL0V27RVZ7LBduom%2FnyB455PUeWQU5Z1ZMIJHhMys%2BghUNoZ70rUETfACEAiIzVJfG5RETRSlgpHeuVI%3D HTTP/1.1\r\n
      Cache-Control: max-age = 157771\r\n
      Connection: Keep-Alive\r\n
      Accept: */*\r\n
      If-Modified-Since: Tue, 09 Oct 2018 09:50:35 GMT\r\n
      If-None-Match: "5bbc79eb-1d7"\r\n
      User-Agent: Microsoft-CryptoAPI/10.0\r\n
      Host: ocsf.digicert.com\r\n
      \r\n
      [Full request URI: http://ocsf.digicert.com/MFewTzBNMEswSTAjBgUrDg/MCGgUA8BTBL0V27RVZ7LBduom%2FnyB455PUeWQU5Z1ZMIJHhMys%2BghUNoZ70rUETfACEAiIzVJfG5RETRSlgpHeuVI%3D]
      [HTTP request 1/3]
      [Response in frame: 26]
0000 44 94 fc 58 62 b8 60 57 18 f2 56 5c 08 00 45 00 D...Xb...W...V...E...
0010 01 86 5d ca 40 00 80 06 97 84 0a 00 00 0b c0 10 ...]@... ..
0020 3a 08 c4 82 00 50 8c 3f e8 4d ab 87 3b 8b 50 18 :...P?..M...;P...
0030 00 44 73 f8 00 00 47 45 54 20 2f 4d 46 45 77 54 ..Ds...GET /MFewT
0040 7a 42 4e 4d 45 73 77 53 54 41 4a 42 67 55 72 4d zBNMEswSTAjBgUrD
0050 67 4d 43 47 67 55 41 42 42 54 42 4c 30 56 32 37 gKCGgUAB 8TBL0V27
0060 52 56 5a 37 4c 42 64 75 6f 6d 25 32 46 6e 59 42 RVZ7LBdu om%2FnyB
0070 34 35 53 50 55 45 77 51 55 35 5a 31 5a 4d 49 4a 45SPUEwQ U5Z1ZMIJ
0080 48 57 4d 79 73 25 32 42 67 68 55 4e 6f 5a 37 4f HhMys%2B ghUNoZ70
0090 72 55 45 54 66 41 43 45 41 69 49 7a 56 4a 66 47 rUETfACE AiIzVJfG
00a0 53 52 45 54 52 53 6c 67 70 48 65 75 56 49 25 33 SRETRSlg pHeuVI%3D
00b0 44 20 48 54 54 50 2f 31 2e 31 0d 0a 43 61 63 68 g HTTP/1.1...Cach
00c0 65 2d 43 6f 6e 74 72 6f 6c 3a 20 6d 61 78 2d 61 e-Contro l: max-a
00d0 67 65 20 3d 20 31 35 37 37 31 0d 0a 43 6f 6e ge = 157 771...Con
00e0 6e 65 63 74 69 6f 6e 3a 20 4b 65 65 70 2d 41 6c nection: Keep-Al
00f0 69 76 65 0d 0a 41 63 63 65 70 74 3a 20 2a 2f 2a ive-Accept: */*
0100 0d 0a 49 66 2d 4d 6f 64 69 66 69 65 64 2d 53 69 ..If-Mod ified-Si
0110 6e 63 65 3a 20 54 75 65 2c 20 30 39 20 4f 63 74 nce: Tue , 09 Oct
0120 20 32 30 31 38 20 30 39 3a 35 30 3a 33 35 20 47 2018 09 :50:35 G
0130 4d 54 0d 0a 49 66 2d 4e 6f 6e 65 2d 4d 61 74 63 MT...If-N one-Matc
0140 68 3a 20 22 35 62 62 63 37 39 65 62 2d 31 64 37 h: "5bbc 79eb-1d7
0150 22 0d 0a 55 73 65 72 2d 41 67 65 6e 74 3a 20 4d "...User- Agent: M

```

Figura 7 Detalhes sobre o pacote HTTP.

Após o envio de um pedido, será gerado uma resposta na estrutura apresentada na figura 8. Retângulo verde mostra o protocolo/versão do código da mensagem. Retângulo azul mostra o campo de rubricas da reposta. Quando a operação é concluída com sucesso 200 é mostrado; caso não exista é mostrado 404; caso não houver conteúdo é mostrado 204, entre outros. O campo de rubricas contém informações adicionais sobre a reposta e/ou sobre o servidor.

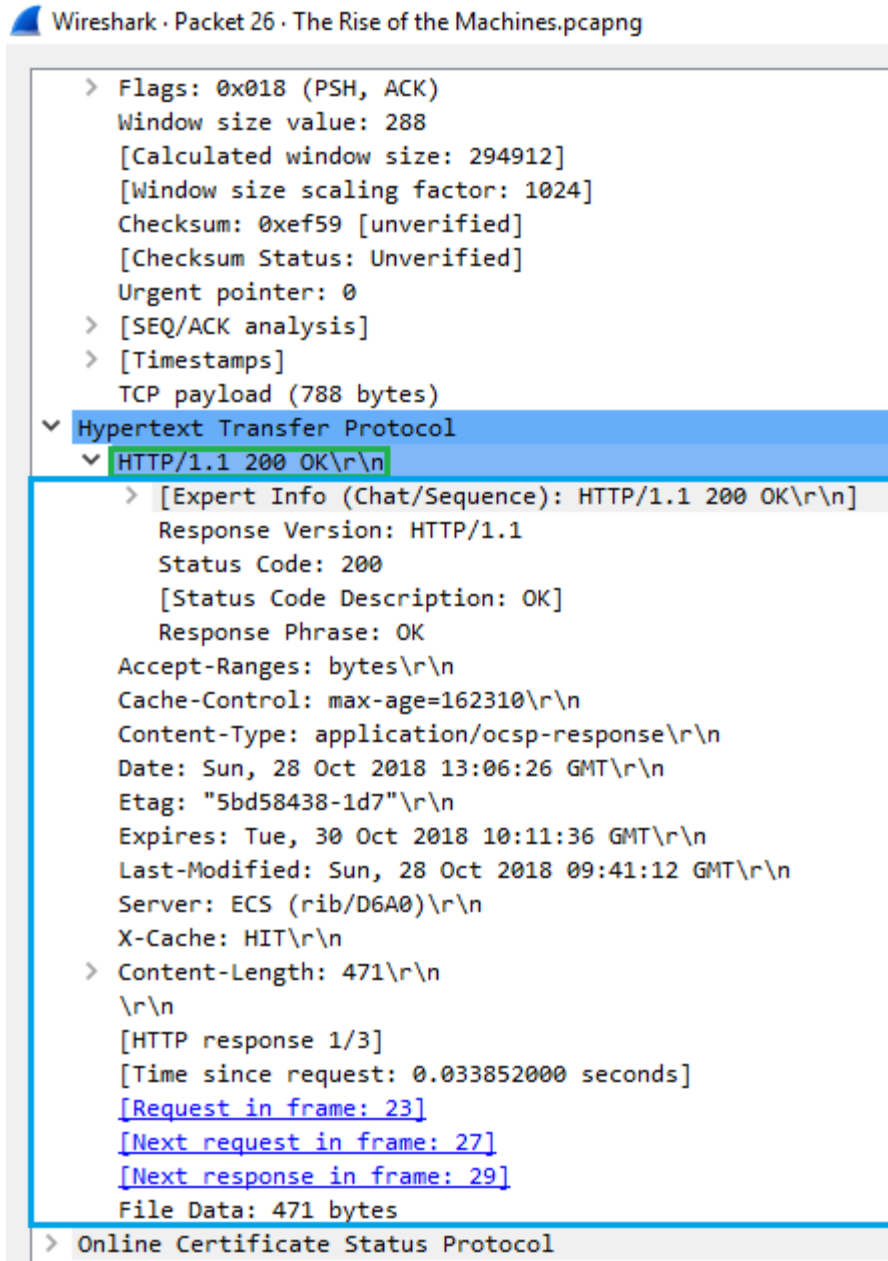


Figura 8 Exemplo de resposta, pacote 26

3.3. Finalização da Conexão

Para a finalização da conexão são necessárias três etapas, que são executadas em três pacotes distintos (Figura 9).

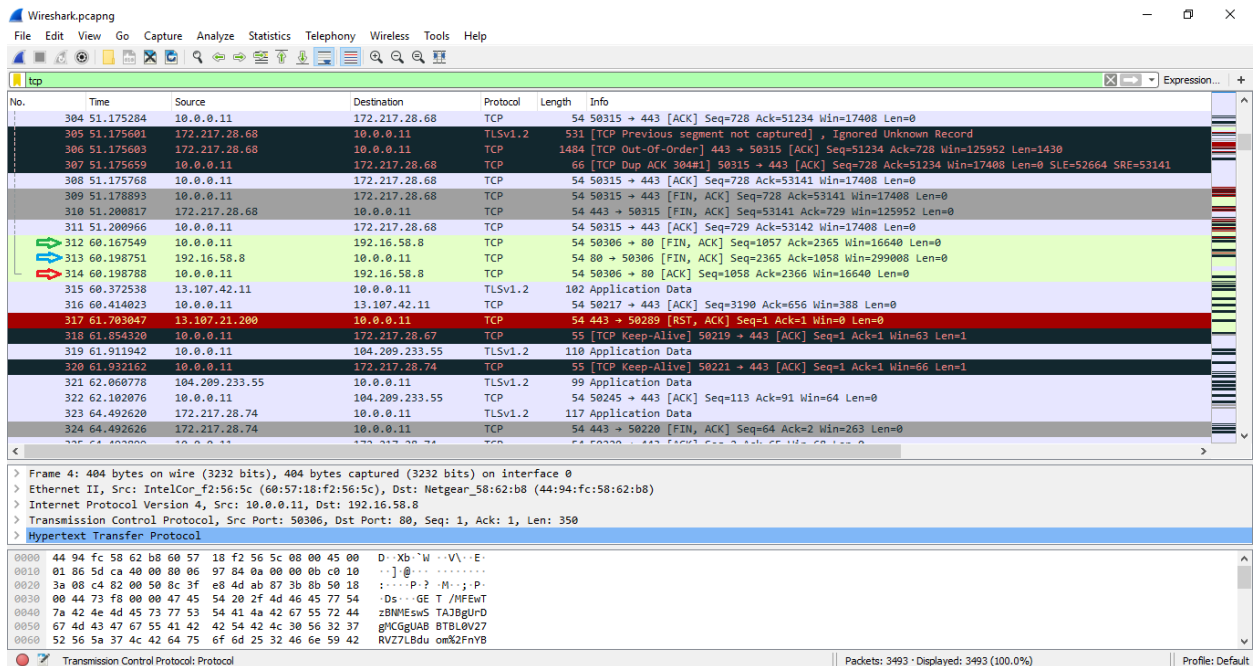


Figura 9 Pacotes para finalização da conexão.

O primeiro pacote para o término da conexão que o servidor envia contém a flag [FIN] e [ACK]. (Figura 10). Retângulo verde é a origem do pacote, o servidor. Retângulo azul é o destino do pacote, o cliente. Retângulo vermelho é o valor da flag: [FIN] e [ACK].

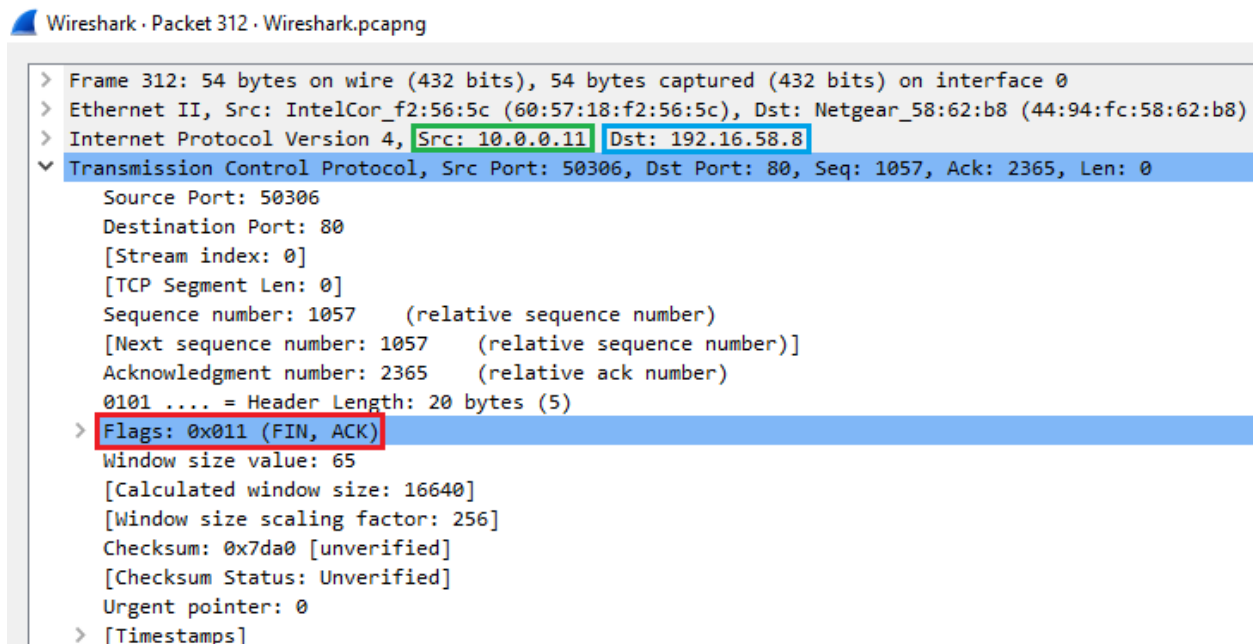
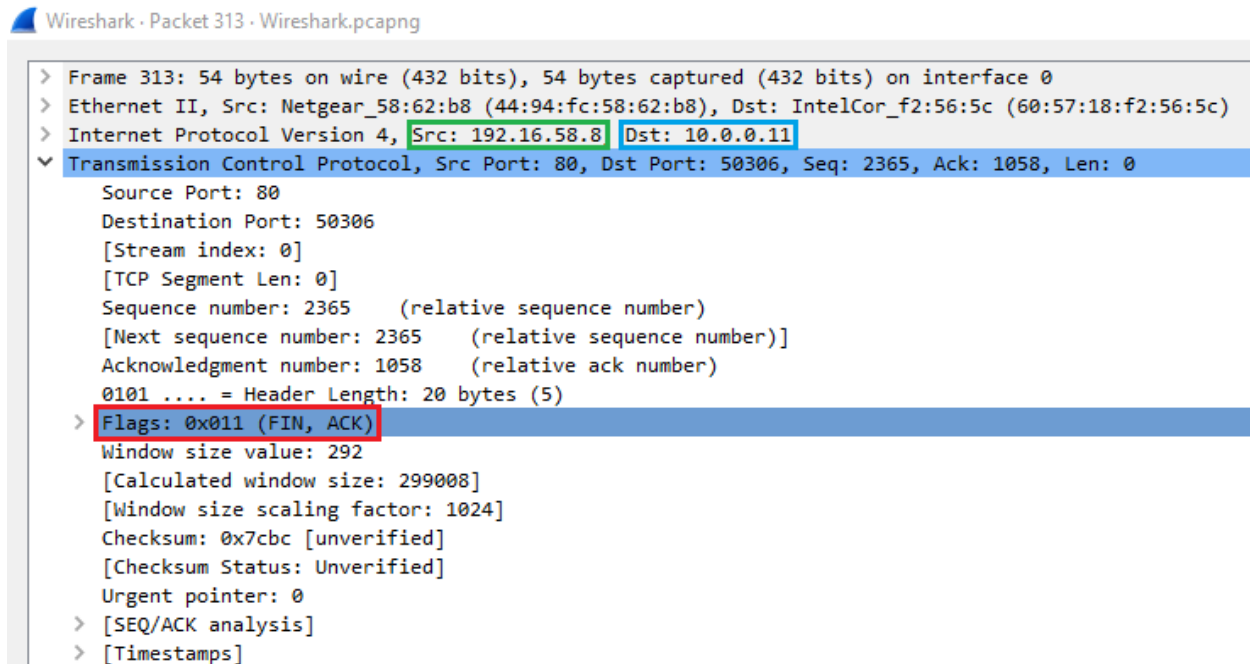


Figura 10 Último pacote com dados.

Em seguida o usuário confirma ao servidor o pedido de finalização, enviando a flag [FIN] e [ACK], e aguarda a verificação do servidor (Figura 11). Retângulo verde é a origem do pacote, o

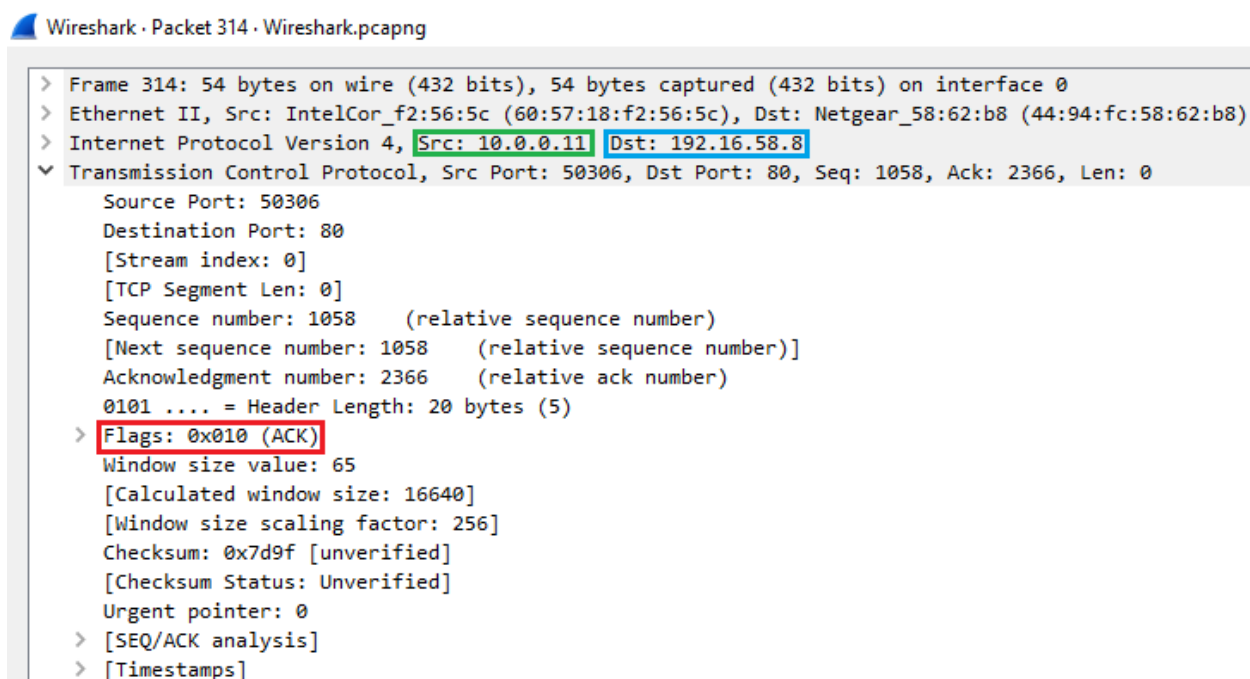
servidor. Retângulo azul é o destino do pacote, o cliente. Retângulo vermelho é o valor da flag: [FIN] e [ACK].



```
Wireshark · Packet 313 · Wireshark.pcapng
> Frame 313: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
> Ethernet II, Src: Netgear_58:62:b8 (44:94:fc:58:62:b8), Dst: IntelCor_f2:56:5c (60:57:18:f2:56:5c)
> Internet Protocol Version 4, Src: 192.16.58.8, Dst: 10.0.0.11
> Transmission Control Protocol, Src Port: 80, Dst Port: 50306, Seq: 2365, Ack: 1058, Len: 0
  Source Port: 80
  Destination Port: 50306
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence number: 2365 (relative sequence number)
  [Next sequence number: 2365 (relative sequence number)]
  Acknowledgment number: 1058 (relative ack number)
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x011 (FIN, ACK)
  Window size value: 292
  [Calculated window size: 299008]
  [Window size scaling factor: 1024]
  Checksum: 0x7cbc [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
```

Figura 11 Usuário confirma ao servidor o pedido de finalização.

Por fim, o servidor confirma o pedido, enviando a flag [ACK], e a conexão é finalizada (Figura 12). Retângulo verde é a origem do pacote, o servidor. Retângulo azul é o destino do pacote, o cliente. Retângulo vermelho é o valor da flag: [ACK].



```
Wireshark · Packet 314 · Wireshark.pcapng
> Frame 314: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface 0
> Ethernet II, Src: IntelCor_f2:56:5c (60:57:18:f2:56:5c), Dst: Netgear_58:62:b8 (44:94:fc:58:62:b8)
> Internet Protocol Version 4, Src: 10.0.0.11, Dst: 192.16.58.8
> Transmission Control Protocol, Src Port: 50306, Dst Port: 80, Seq: 1058, Ack: 2366, Len: 0
  Source Port: 50306
  Destination Port: 80
  [Stream index: 0]
  [TCP Segment Len: 0]
  Sequence number: 1058 (relative sequence number)
  [Next sequence number: 1058 (relative sequence number)]
  Acknowledgment number: 2366 (relative ack number)
  0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
  Window size value: 65
  [Calculated window size: 16640]
  [Window size scaling factor: 256]
  Checksum: 0x7d9f [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
```

Figura 12 Servidor confirma o pedido de finalização.

4. Conclusões

Este trabalho prático identifica a ocorrência de conexões e transferências de dados, envolvendo as camadas de aplicação, transporte e rede. Os protocolos abordados foram o HTTP e o TCP. Foram identificados os pacotes de estabelecimento de conexão, transferência de dados e finalização da conexão. A ferramenta Wireshark foi utilizada para analisar o tráfego da rede e a organização dos protocolos.

5. Referências Bibliográficas

^[1] <https://pt.wikipedia.org/wiki/Wireshark>

^[2] https://pt.wikipedia.org/wiki/Transmission_Control_Protocol

^[3] <https://br.ccm.net/contents/285-o-que-e-o-protocolo-tcp-ip>

^[4] https://pt.wikipedia.org/wiki/Hypertext_Transfer_Protocol