



MÓDULO DE:

## **SERVIÇOS DE REDE**

AUTORIA:

**Msc. PEDRO HENRIQUE MANNATO COUTINHO**

Copyright © 2011, ESAB – Escola Superior Aberta do Brasil

Módulo de: Serviços de Rede

Autoria: Pedro Henrique Mannato Coutinho

Primeira edição: 2011

## CITAÇÃO DE MARCAS NOTÓRIAS

Várias marcas registradas são citadas no conteúdo deste módulo. Mais do que simplesmente listar esses nomes e informar quem possui seus direitos de exploração ou ainda imprimir logotipos, o autor declara estar utilizando tais nomes apenas para fins editoriais acadêmicos.

Declara ainda, que sua utilização tem como objetivo, exclusivamente a aplicação didática, beneficiando e divulgando a marca do detentor, sem a intenção de infringir as regras básicas de autenticidade de sua utilização e direitos autorais.

E por fim, declara estar utilizando parte de alguns circuitos eletrônicos, os quais foram analisados em pesquisas de laboratório e de literaturas já editadas, que se encontram expostas ao comércio livre editorial.

Todos os direitos desta edição reservados à

ESAB – ESCOLA SUPERIOR ABERTA DO BRASIL LTDA

<http://www.esab.edu.br>

Av. Santa Leopoldina, nº 840/07

Bairro Itaparica – Vila Velha, ES

CEP: 29102-040

Copyright © 2011, ESAB – Escola Superior Aberta do Brasil

# Apresentação

Atualmente a Internet representa um papel importante em nossas vidas. Podemos dizer que a Web mudou o nosso modo de viver, de forma que é difícil imaginar como seria nosso dia a dia se a Internet deixasse de existir. Acessamos e-mails, transferimos arquivos, realizamos compras, conversamos via bate-papo, assistimos a vídeos, lemos notícias, pesquisamos, e até mesmo estudamos através do Ensino Online à Distância, como é o caso deste Módulo.

Tudo isso é possível graças à existência de aplicações clientes e servidoras, e também de protocolos e serviços adicionais que permitem que os dados sejam trocados via Internet. Portanto, o objetivo deste Módulo é apresentar os principais serviços com respectivos protocolos e aplicações que permitem nossa interação com essa fantástica tecnologia de comunicação.

# O bjetivo

Apresentar de forma dinâmica e agradável os conceitos dos principais serviços de redes, em conjunto com demonstrações práticas. Para atingir esse objetivo, foram intercaladas Unidades de conceituação com outras de exemplos reais, utilizando um analisador de pacotes para demonstrar o funcionamento e detalhes de alguns serviços apresentados aqui.

# Ementa

---

Apresentação dos seguintes serviços de rede: serviço de transporte (TCP/IP), segurança de comunicação (SSL e TLS), web (HTTP), transferência de arquivos (FTP), correio eletrônico (SMTP, POP3, IMAP), acesso remoto (Telnet e SSH), tradução de nomes para endereço IP (DNS), atribuição dinâmica de IP (DHCP), rede virtual privada (VPN) e o serviço para reduzir o tempo de resposta, economia de banda e controle de acesso (Proxy).

Demonstração de maneira prática do funcionamento de alguns dos serviços e respectivos protocolos e aplicações descritos, utilizando um analisador de pacotes de redes.

## Sobre o Autor

---

Mestre em Informática (UFES -2007), Graduado em Ciência da Computação (UFES-2004).

A Dissertação de Mestrado rendeu o terceiro lugar no prêmio de dissertações do SBIE 2008 (Simpósio Brasileiro de Informática na Educação).

Diretor Executivo e sócio fundador da empresa Projeta Sistemas de Informação. Vice-Presidente de Associativismo e Financeiro da ASSESPRO-ES.

Professor de Pós Graduação Lato Sensu em disciplinas presenciais e online. Faz parte do corpo de consultores de tecnologia do SEBRAE-ES. Possui experiência atuando como Gerente de Projeto, Analista de Sistema, Analista de Processos de Negócio (BPM), Desenvolvedor, Pesquisador de Novas Tecnologias, dentre outros. Atuou em projetos que tinham como clientes: Arcelor Mittal, Receita Federal, IBGE, Sebrae, Grupo Coimex, ESAB, dentre outros. Atuou como analista e desenvolvedor do software para o gerenciamento de empresas do mercado rent a car, ganhador do 1º Prêmio do Pólo de Software do Espírito Santo (22/01/2008) e um dos quatro finalistas do 6º Encontro Nacional de Tecnologia e Negócios - Rio Info 2008 (30/09/2008).

# SUMÁRIO

---

|  |           |
|--|-----------|
| <b>UNIDADE 1 .....</b>                                       | <b>7</b>  |
| Introdução .....   | 7         |
| <b>UNIDADE 2 .....</b>                                       | <b>12</b> |
| Breve Apresentação de Conceitos: Redes de Computadores ..... | 12        |
| <b>UNIDADE 3 .....</b>                                       | <b>16</b> |
| Serviços da Camada de Transporte: UDP e TCP .....            | 16        |
| <b>UNIDADE 4 .....</b>                                       | <b>23</b> |
| Serviços Seguros de Transporte: SSL e TLS.....               | 23        |
| <b>UNIDADE 5 .....</b>                                       | <b>30</b> |
| Analisador de Pacotes de Rede.....                           | 30        |
| <b>UNIDADE 6 .....</b>                                       | <b>38</b> |
| HTTP.....  | 38        |
| <b>UNIDADE 7 .....</b>                                       | <b>42</b> |
| HTTP (Continuação) .....                                     | 42        |
| <b>UNIDADE 8 .....</b>                                       | <b>48</b> |
| HTTPS .....  | 48        |
| <b>UNIDADE 9 .....</b>                                       | <b>52</b> |
| Analisador de Pacotes de Rede – HTTP x HTTPS .....           | 52        |
| <b>UNIDADE 10 .....</b>                                      | <b>57</b> |
| Proxy .....  | 57        |
| <b>UNIDADE 11 .....</b>                                      | <b>61</b> |
| Proxy (Continuação).....                                     | 61        |
| <b>UNIDADE 12 .....</b>                                      | <b>65</b> |
| DNS (Domain Name Server).....                                | 65        |
| <b>UNIDADE 13 .....</b>                                      | <b>72</b> |
| DNS (Continuação) .....                                      | 72        |
| <b>UNIDADE 14 .....</b>                                      | <b>78</b> |
| Analisador de Pacotes de Rede – Consulta DNS .....           | 78        |
| <b>UNIDADE 15 .....</b>                                      | <b>83</b> |
| Correio Eletrônico.....                                      | 83        |

|   |            |
|---|------------|
| <b>UNIDADE 16 .....</b>   | <b>88</b>  |
| Correio Eletrônico (Continuação) .....  | 88         |
| <b>UNIDADE 17 .....</b>   | <b>92</b>  |
| DHCP (Dynamic Host Configuration Protocol) .....                                      | 92         |
| <b>UNIDADE 18 .....</b>   | <b>97</b>  |
| DHCP (Continuação).....   | 97         |
| <b>UNIDADE 19 .....</b>   | <b>100</b> |
| Analisador de Pacotes de Rede – DHCP.....   | 100        |
| <b>UNIDADE 20 .....</b>   | <b>106</b> |
| VPN.....  | 106        |
| <b>UNIDADE 21 .....</b>   | <b>110</b> |
| VPN (Continuação) .....   | 110        |
| <b>UNIDADE 22 .....</b>   | <b>115</b> |
| VPN (Continuação) .....   | 115        |
| <b>UNIDADE 23 .....</b>   | <b>118</b> |
| Serviços e Protocolos de Acesso Remoto - Telnet .....                                 | 118        |
| <b>UNIDADE 24 .....</b>   | <b>121</b> |
| Serviços e Protocolos de Acesso Remoto - SSH (Secure Shell) .....                     | 121        |
| <b>UNIDADE 25 .....</b>   | <b>124</b> |
| Analisador de Pacotes de Rede – Telnet x SSH.....                                     | 124        |
| <b>UNIDADE 26 .....</b>   | <b>130</b> |
| Serviços e Protocolos de Transferência de Arquivos - FTP (File Transfer Protocol).... | 130        |
| <b>UNIDADE 27 .....</b>   | <b>136</b> |
| FTP (Continuação) .....   | 136        |
| <b>UNIDADE 28 .....</b>   | <b>140</b> |
| Analisador de Pacotes de Rede – FTP .....   | 140        |
| <b>UNIDADE 29 .....</b>   | <b>144</b> |
| TFTP (Trivial File Transfer Protocol) .....   | 144        |
| <b>UNIDADE 30 .....</b>   | <b>148</b> |
| Alternativas Seguras para realizar a Transferência de Arquivos.....                   | 148        |

# UNIDADE 1

*Objetivo: Realizar uma apresentação inicial do conteúdo deste Módulo de Serviços de Redes.*

## **Introdução**

Bem vindo ao Módulo de Serviços de Redes!

Atualmente a Internet representa um papel importante em nossas vidas. Podemos dizer que a Web mudou o nosso modo de viver, de forma que é difícil imaginar como seria nosso dia a dia se a Internet deixasse de existir. Acessamos e-mails, transferimos arquivos, realizamos compras, conversamos via bate-papo, assistimos a vídeos, lemos notícias, pesquisamos, e até mesmo estudamos através do Ensino Online à Distância, como é o caso deste Módulo.

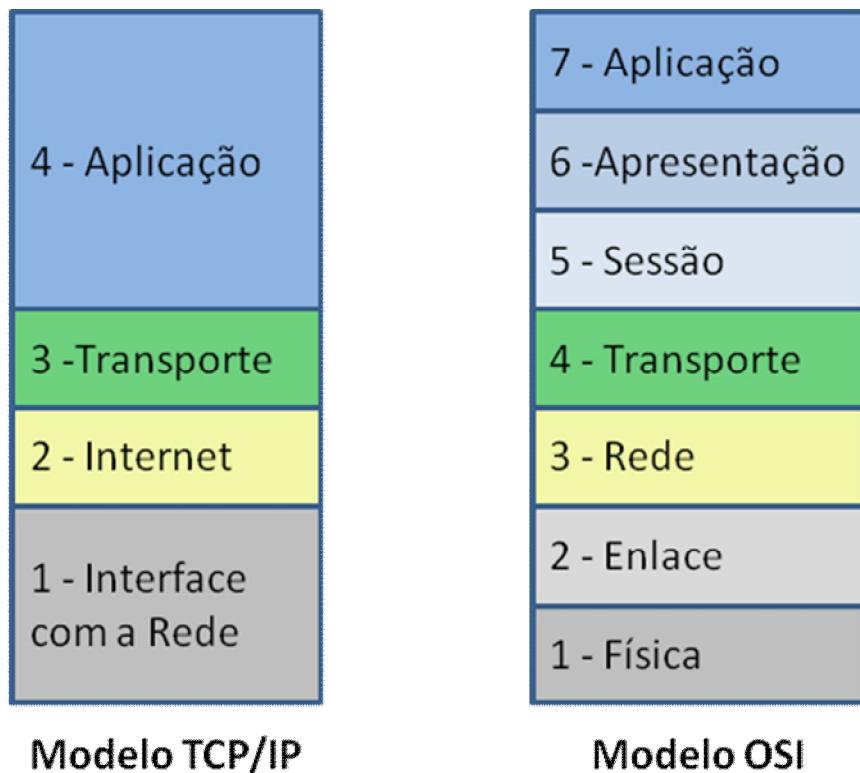
Tudo isso é possível graças à existência de aplicações clientes e servidoras, e também de protocolos e serviços adicionais que permitem que os dados sejam trocados via Internet. Portanto, o objetivo deste Módulo é apresentar os principais serviços com respectivos protocolos e aplicações que permitem nossa interação com essa fantástica tecnologia de comunicação.

Os conhecimentos apresentados aqui foram organizados para serem agradáveis e dinâmicos, apresentando os principais conceitos em conjunto com demonstrações práticas. Para atingir esse objetivo, intercalamos algumas Unidades de conceituação com outras de exemplos reais, onde utilizamos um analisador de pacotes para demonstrar o funcionamento e detalhes de alguns serviços apresentados aqui.

A maioria dos serviços apresentados se refere às aplicações e respectivos protocolos da Camada de Aplicação, que são os principais serviços e a “razão de ser” das redes de computadores. Mas antes de falarmos sobre os serviços, vamos relembrar em que contexto a Camada de Aplicação está inserida.

## ***Os Modelos de Referências e Respectivas Camadas***

As redes são projetadas em camadas empilhadas com o objetivo de reduzir sua complexidade, de forma que a camada inferior possa fornecer determinados serviços à sua camada superior, sem que esta última tenha que se preocupar com detalhes da implementação dos mesmos. Portanto, entre as camadas existe uma interface que define as operações e serviços disponibilizados para a camada superior, facilitando a substituição de implementação de uma camada por outra implementação distinta.



**Figura 1 – Os Modelos de Referências TCP/IP e OSI e suas respectivas Camadas**

As camadas correspondentes em diferentes máquinas são denominadas “pares”, e se comunicam através de protocolos, que são os padrões e regras convencionados para a comunicação. Se analisarmos mais de perto, perceberemos que os “pares” não se comunicam diretamente. Na verdade, a camada que deseja se comunicar com sua camada correspondente em outra máquina passa os dados e informações para a sua camada imediatamente inferior, que adota o mesmo procedimento até atingir o meio físico. Através do meio físico a mensagem é transportada até a camada mais baixa da

outra máquina, que transfere os dados para as camadas superiores até alcançar a camada desejada.

O nome, número, função e conteúdo de cada camada, diferem de uma arquitetura para outra. Duas importantes arquiteturas de redes em camadas são o modelo de referência TCP/IP e o modelo de referência OSI, apresentados na Figura 1. Vale destacar que o modelo de referência TCP/IP em quatro camadas apresentado na Figura 1 se baseia no padrão IETF definido na RFC 1122 (<http://tools.ietf.org/html/rfc1122>), embora seja possível encontrar na literatura autores que o apresentem em cinco camadas.

O modelo de referência TCP/IP recebeu esse nome devido aos seus dois principais protocolos, e seu início é antigo. A ARPANET (antecessora da Internet) era uma rede de pesquisa patrocinada pelo Departamento de Defesa dos Estados Unidos, que foi crescendo na medida em que universidades e repartições públicas foram se conectando. Para permitir que várias redes se conectassem de maneira uniforme evitando problemas e conflitos, foi definida em 1974 uma arquitetura de referência, que ficou conhecida como Modelo de Referência TCP/IP. Quatro versões do TCP/IP foram desenvolvidas na época. A migração da ARPANET para TCP/IP foi oficialmente completada em 1983.

Já o modelo de referência OSI (*Open Systems Interconnection*, em português, Interconexão de Sistemas Abertos) foi baseado em uma proposta de 1983 desenvolvida pela ISO (*International Standards Organization*, ou Organização Internacional de Padrões) para padronizar internacionalmente os protocolos utilizados nas diferentes camadas.

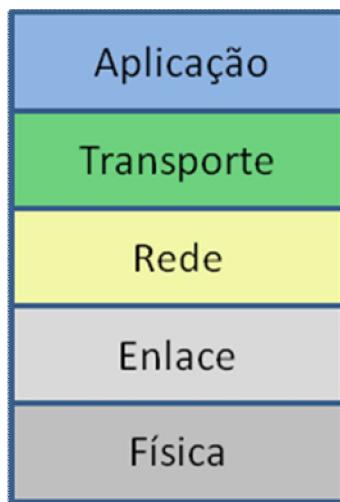
Apesar das diferenças entre os dois modelos, que não iremos abordar aqui, podemos perceber que ambos possuem características comuns, com camadas com funções equivalentes, como, por exemplo, a camada de transporte presente nos dois modelos.

A camada mais alta, a de aplicação, é a camada mais próxima do usuário. Ela também está presente em ambos os modelos. O modelo TCP/IP não possui as camadas de Apresentação e Sessão, e a experiência com o modelo OSI demonstrou que elas são pouco utilizadas pela maioria das aplicações.

O modelo OSI (sem as camadas de Apresentação e Sessão) é útil para discussão de redes de computadores, porém seus protocolos não se tornaram populares. Já o modelo

TCP/IP, no caminho inverso possui seus protocolos utilizados amplamente, porém o modelo praticamente não existe.

Como as camadas de Apresentação e Sessão do modelo OSI praticamente não são utilizadas, durante o texto faremos referência a um modelo adaptado conforme a Figura 2 abaixo, em que a camada de aplicação está acima da de transporte (assim como no modelo TCP/IP).



**Figura 2 – Modelo Adaptado**

### **A Camada de Aplicação**

A camada de aplicação contém a maioria dos programas utilizados para comunicação através da rede, contendo um conjunto de protocolos essenciais para os usuários. Portanto, grande parte dos serviços abordados neste módulo é baseada em protocolos da camada de aplicação, como: web (HTTP), transferência de arquivos (FTP), correio eletrônico (SMTP, POP3, IMAP), acesso remoto (Telnet e SSH), e tradução de nomes para endereço IP (DNS).

Além dos serviços da camada de aplicação, serão apresentados outros serviços de rede de fundamental importância para o funcionamento da Internet e dos serviços da camada de aplicação, como o serviço de transporte (TCP/IP), transmissão de dados criptografados oferecendo confidencialidade e integridade (SSL e TLS), atribuição

dinâmica de IP (DHCP), rede virtual privada (VPN) e o serviço para reduzir o tempo de resposta, economia de banda e controle de acesso (Proxy).

### ***Organização do Conteúdo***

Nas próximas quatro unidades deste Módulo, estudaremos conceitos que serão utilizados nas demais unidades. Portanto, a “Unidade 2” cita brevemente as redes de computadores, portas, protocolos e arquitetura cliente/servidor. A “Unidade 3” apresentará o UDP e TCP enquanto a “Unidade 4” o SSL/TLS. Ao longo do estudo visualizaremos que esses serviços são importantes para os serviços da camada de aplicação apresentados. Já a “Unidade 4” apresenta uma ferramenta para captura de pacotes chamada *Wireshark*, utilizada ao longo de vários pontos deste módulo para demonstrar de maneira prática o funcionamento de alguns dos serviços, aplicações e protocolos descritos. A partir da “Unidade 5”, os outros serviços citados anteriormente são apresentados, intercalando sempre que conveniente a demonstração prática com *Wireshark*.

# UNIDADE 2

*Objetivo: Relembrar de forma sucinta alguns conceitos relativos às redes de computadores que servirão como base para as próximas unidades.*

## Breve Apresentação de Conceitos: Redes de Computadores

Esta unidade apresenta de forma sucinta alguns conceitos gerais das redes de computadores, para que você relembre alguns conhecimentos prévios adquiridos em outros módulos do curso da ESAB ou em estudos anteriores

Uma rede pode ser definida como um conjunto de computadores e outros equipamentos interligados e capazes de comunicarem-se utilizando um conjunto pré-determinado de regras, ou “linguagem”, chamada de **protocolo**. Os protocolos são especificados por institutos de pesquisa e anunciados para toda a comunidade por meio de um memorando publicado pela **IETF** (*Internet Engineering Task Force – Força Tarefa de Engenharia da Internet*) denominado **RFC** (*Request for Comments – Solicitação de Comentários*). Este memorando deve descrever os métodos, padrões, pesquisas ou inovações aplicadas ao funcionamento da Internet.

Na comunicação via correios, para que uma carta chegue ao endereço de destino correto e possa ser respondida ao autor devidamente, o envelope deve conter todas as informações de como localizar remetente e destinatário. Da mesma forma, para um pacote de dados trafegar por uma rede, este deve conter todas as informações necessárias de endereço de origem e destino.

O **IP**, ou *Internet Protocol*, é um tipo de protocolo que foi projetado para criar ligações entre diferentes redes, possibilitando a intercomunicação entre dispositivos nelas presentes. Denomina-se normalmente de internet (com “i” minúsculo) uma interligação entre diversas redes. Como nos correios, para que não haja erros de entrega mensagens e encomendas (encomendas que em redes de computadores pode-se fazer uma analogia com os pacotes que trafegam nas redes), numa determinada internet o endereço de cada

equipamento de rede ativo, como *switches*, roteadores, e computadores deve ser único. Este endereço é chamado **endereço IP**.

Costuma-se designar de **Internet**, com a letra “i” maiúscula, à interligação de milhares de redes ao redor do mundo. Por volta de 1983, ela poderia ser considerada uma rede predominantemente acadêmica, com cerca de 200 computadores interligados. A partir de 1995, a Internet começa a se popularizar, chegando ao fantástico patamar que encontramos atualmente.

Sem as **aplicações de redes**, uma rede de computadores não faria muito sentido. Podemos citar como exemplos aplicações de redes, o acesso remoto a computadores, transferências de arquivos, correio eletrônico, grupos de discussão, navegação web, dentre outros. Os protocolos das aplicações de redes normalmente especificam as comunicações entre duas entidades: um **cliente** e um **servidor**. Portanto, normalmente temos as aplicações clientes e as aplicações servidoras. Podemos citar como exemplo a própria Web, em que do lado do cliente temos os navegadores (*browsers*) e do lado servidor um servidor Web (como por exemplo, o servidor Web Apache). Existem casos em que a mesma aplicação implementa tanto o lado cliente como o lado servidor. Um exemplo típico são aplicações FTP, em que durante uma sessão entre dois hospedeiros, ambos podem transferir arquivos. Entretanto, normalmente a parte que inicia a sessão é denominada de cliente.

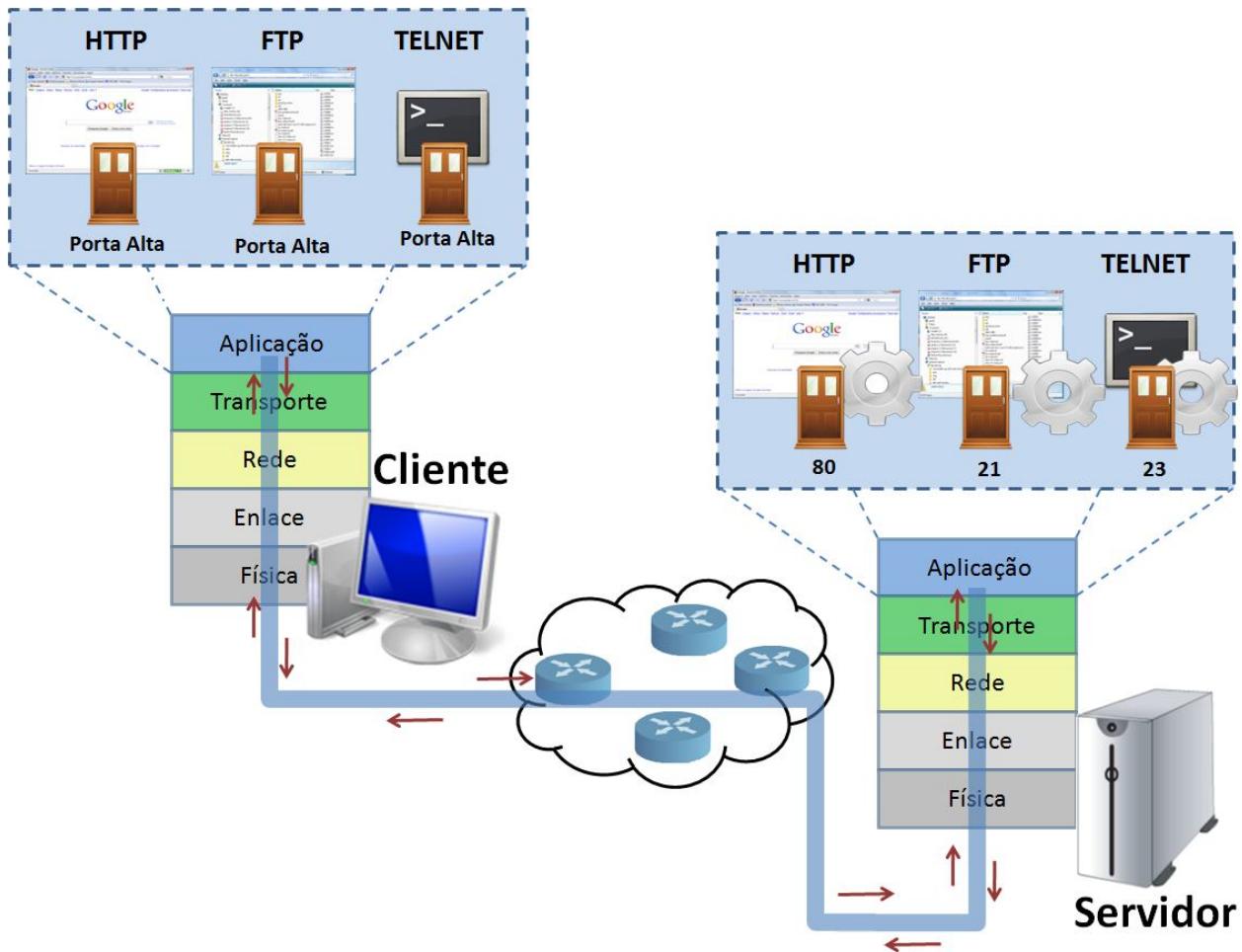
Existem dois tipos de aplicações cliente/servidor: aplicações proprietárias e aplicações que implementam protocolos padrões definidos em uma RFC. Nas aplicações proprietárias, os programas clientes e servidores podem ser desenvolvidos sem precisar se basear em protocolos de domínio público, de forma que pode acontecer de programadores independentes não conseguir desenvolver aplicações para se comunicar com as mesmas. No presente Módulo, as aplicações explicadas são baseadas em implementações de acordo com regras definidas nas RFCs.

A padronização dos protocolos mostra a sua relevância ao permitir que aplicações hospedadas em diferentes sistemas operacionais e desenvolvidas por equipes independentes, possam se comunicar sem problemas.

As aplicações presentes em máquinas distintas se comunicam através da troca de mensagens utilizando a rede de computadores. As aplicações enviam e recebem mensagens através de suas portas. Uma porta é a interface entre camada de aplicação e a camada de transporte, e pode ser entendida como um canal de entrada e saída. As principais aplicações de rede possuem números das portas definidas para o serviço fornecido pelo servidor. Essas portas são denominadas “portas bem conhecidas” (do inglês *well-known port numbers*), e tipicamente é um número de porta baixo (menor do que 1024). Por exemplo, um servidor HTTP, por padrão, escuta a porta 80. Em contrapartida, a aplicação cliente normalmente utiliza uma porta de número alto, definida dinamicamente e com objetivo de ter curta duração. Embora a maioria das aplicações possua número de portas definidas para a aplicação servidora e utilizem uma porta alta alocada dinamicamente para a aplicação cliente, existem casos em que o cliente e servidor utilizam portas determinadas, como por exemplo, o DHCP em que o servidor utiliza a porta 67 e o cliente a 68.

A aplicação cliente é responsável por iniciar o contato com a aplicação servidora, através de sua porta bem conhecida e endereço IP. Para o servidor receber o contato inicial e respondê-lo, sua aplicação tem que estar ativa (rodando) e escutando a sua porta padrão. É como se nós clientes tivéssemos um fornecedor (servidor), como por exemplo, de assinatura de revistas, responsável por atender nossas solicitações por telefone (ou seja, por uma rede). O número do telefone e ramal (no caso, endereço IP e número de porta) do prestador do serviço precisam ser conhecidos pelo cliente que deseja o serviço. Para alguém poder atender a demanda do cliente, precisa estar disponível (rodando) e escutando o seu ramal. Uma vez que o cliente telefona e o funcionário do fornecedor atende, a comunicação é estabelecida para a prestação de serviço poder ser realizada.

Em relação às portas, podemos também fazer uma analogia da camada de aplicação do servidor com um andar de atendimento de um hospital, com várias especialidades. Se você estiver procurando um cardiologista, terá que se dirigir à porta 443 (HTTPS). Um paciente que precisa de um neurologista terá que entrar na porta 22 (SSH).



**Figura 3 – Comunicação de Aplicações Cliente-Servidor através da Rede**

As portas “bem conhecidas” são definidas pelo IANA (*Internet Assigned Numbers Authority* – Autoridade Atribuidora de Números da Internet) e podem ser vistas em <http://www.iana.org/assignments/port-numbers>.

# UNIDADE 3

*Objetivo: Apresentar os dois principais protocolos da Camada de Transporte, o UDP e o TCP.*

## Serviços da Camada de Transporte: UDP e TCP

A **camada de transporte** fica situada em uma posição chave entre a **camada de aplicação** e a **camada de rede**, portanto, presta serviços para a camada de aplicação e utiliza serviços fornecidos pela camada de redes.

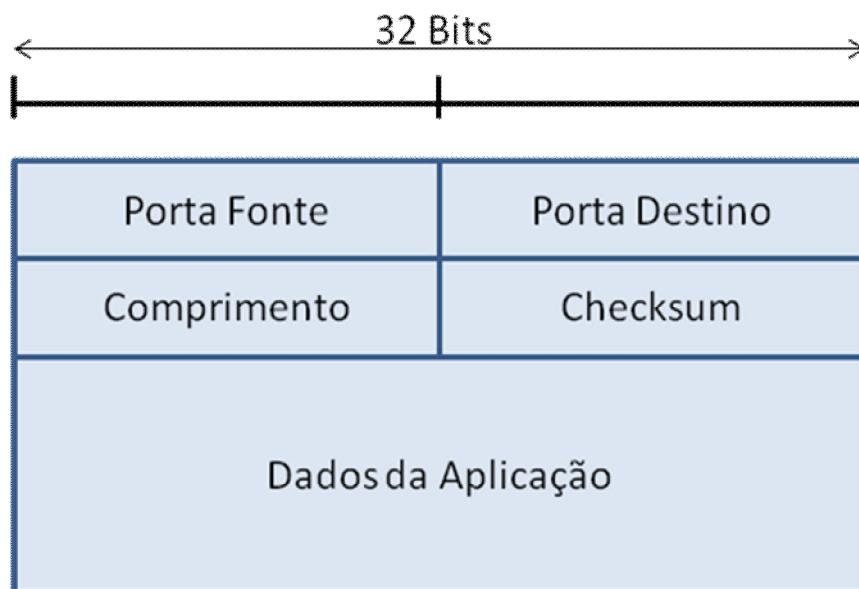
Vale ressaltar que os serviços fornecidos pela camada de redes não é confiável, visto que o serviço IP não garante a entrega, ordem e integridade dos datagramas. Com os serviços IP existe a possibilidade de datagramas transbordarem os *buffers* dos roteadores e não chegarem ao seu destino final, assim como podem chegar fora de ordem, ou até mesmo possuir os seus bits corrompidos (mudar de 0 para 1 ou de 1 para 0). Portanto o IP não foi projetado para ser confiável. Dessa forma, cabe a camada de transporte definir se vai fornecer ou não confiabilidade na entrega. Assim, uma das principais funções da camada de transporte é proteger as camadas superiores das imperfeições da rede. A separação dos serviços e ocultamento de complexidade das camadas inferiores são características que tornam os modelos em camadas tão úteis. A seguir serão apresentados de maneira breve os dois principais protocolos da camada de transporte: o UDP e o TCP.

### **UDP**

O **UDP** (*User Datagram Protocol* – Protocolo de Datagrama do Usuário) permite que as aplicações enviem mensagens sem realizar uma apresentação mútua do cliente-servidor, ou seja, quando um dos lados deseja enviar pacotes para outro, ele apenas os envia. Como não há uma sessão de apresentação entre o remetente e destinatário, o UDP é classificado como **não orientado à conexão**.

O UDP foi definido na RFC 768 em 1980 (<http://tools.ietf.org/html/rfc768>), e é quase uma extensão da camada de rede, visto que realiza somente o mínimo esperado por um

protocolo de transporte. A estrutura do seu segmento, apresentada na Figura 4, consiste em um cabeçalho de 8 bytes, sendo quatro campos de 2 bytes cada. Os campos que representam o número da **porta da fonte e do destino** servem para que a camada de transporte entregue o segmento para o processo da aplicação correta. O campo do comprimento informa o tamanho total do segmento UDP, incluindo os 8 bytes do cabeçalho. Já o *checksum*, ou soma de verificação, serve para o receptor verificar se ocorreu algum erro com o segmento.



**Figura 4 – Estrutura de um segmento UDP**

Portanto, o UDP ao receber a mensagem do processo de aplicação, adiciona as informações dos números das portas fonte e destino e outros dois pequenos campos. Esse segmento é então entregue à camada de rede, que o encapsulará em um datagrama IP e tentará via melhor esforço entregar ao destinatário, para que esse realize o processo inverso e o UDP possa entregar ao processo de aplicação correto.

O UDP **não fornece um serviço confiável de dados**, visto que não realiza garantia de entrega, controle de fluxo e nem controle de erros ou retransmissão. Ele simplesmente fornece um meio para que o protocolo IP possa ser entregue para aplicação correta.

Mas se o UDP não fornece um serviço de transferência confiável de dados, não seria melhor que as aplicações sempre escolhessem o TCP como protocolo da camada de transporte? Na realidade, não. E o motivo é porque existem aplicações em que pequenas

perdas são toleráveis de forma que uma entrega confiável não seja imprescindível. Por exemplo, as aplicações de vídeo conferência, transmissão de áudio e vídeo em tempo real normalmente rodam sobre o UDP devido a esse motivo. Outro exemplo é o protocolo da camada de aplicação DNS (veremos mais detalhes sobre o DNS na Unidade 12) que normalmente realiza suas requisições padrões na Internet utilizando o UDP.

Existe uma forma de ter uma transferência confiável de dados utilizando o UDP, que é realizar todo o controle na própria aplicação. Só que ela não é trivial e necessita de mais esforço dos desenvolvedores que precisam implementar todo controle. Entretanto, muitas aplicações proprietárias de transferência de vídeo e áudio que utilizam o UDP optam por essa estratégia de realizar o controle na aplicação para reduzir a perda de pacotes.

### **TCP**

O **TCP** (*Transmission Control Protocol* – Protocolo de Controle de Transmissão) é o protocolo da camada de transporte que fornece uma **transferência de dados confiável** e é **orientado à conexão**. Ele foi projetado para fornecer o serviço de transferência confiável entre redes com topologias, largura de banda, tamanho de pacotes, atrasos, entre outras configurações distintas. Portanto o objetivo do TCP é fornecer um serviço confiável em uma rede não confiável. O TCP foi apresentado em um artigo em maio de 1974, seguido pela primeira especificação na RFC 675 do mesmo ano (<http://tools.ietf.org/html/rfc675>). Posteriormente o TCP foi especificado no RFC 793 (<http://tools.ietf.org/html/rfc793>), e alguns esclarecimentos e soluções de bugs foram descritos em RFCs posteriores.

Do ponto de vista da aplicação uma conexão TCP é uma conexão virtual direta entre as portas das aplicações cliente e servidor. Entretanto, somente cliente e servidor têm conhecimento desta conexão “virtual”, visto que os elementos intermediários da rede (ex: roteadores) não mantém nenhuma informação da conexão TCP, tomando conhecimento apenas dos datagramas que passam por eles.

As conexões TCP são **full-duplex** porque o tráfego pode ser feito entre o cliente e servidor ao mesmo tempo, ou seja, os dados da camada de aplicação podem ser transferidos do cliente para o servidor no mesmo instante em que o servidor transfere dados para o cliente. As conexões TCP também são **ponto a ponto**, ou seja, ocorre

exatamente entre duas extremidades (um remetente e um destinatário), de forma que a transferência de um remetente para vários destinatários (o chamado *multicast*), não é fornecida pelo TCP. Já o UDP permite *multicast*.

A Figura 5 apresenta a estrutura de um segmento TCP. Normalmente, o cabeçalho TCP tem 20 bytes, ou seja, 12 a mais do que o cabeçalho UDP. Assim como no UDP, o cabeçalho TCP possui os campos com o número da **porta fonte e destino**, e o campo **checksum**, para realizar a soma de verificação.

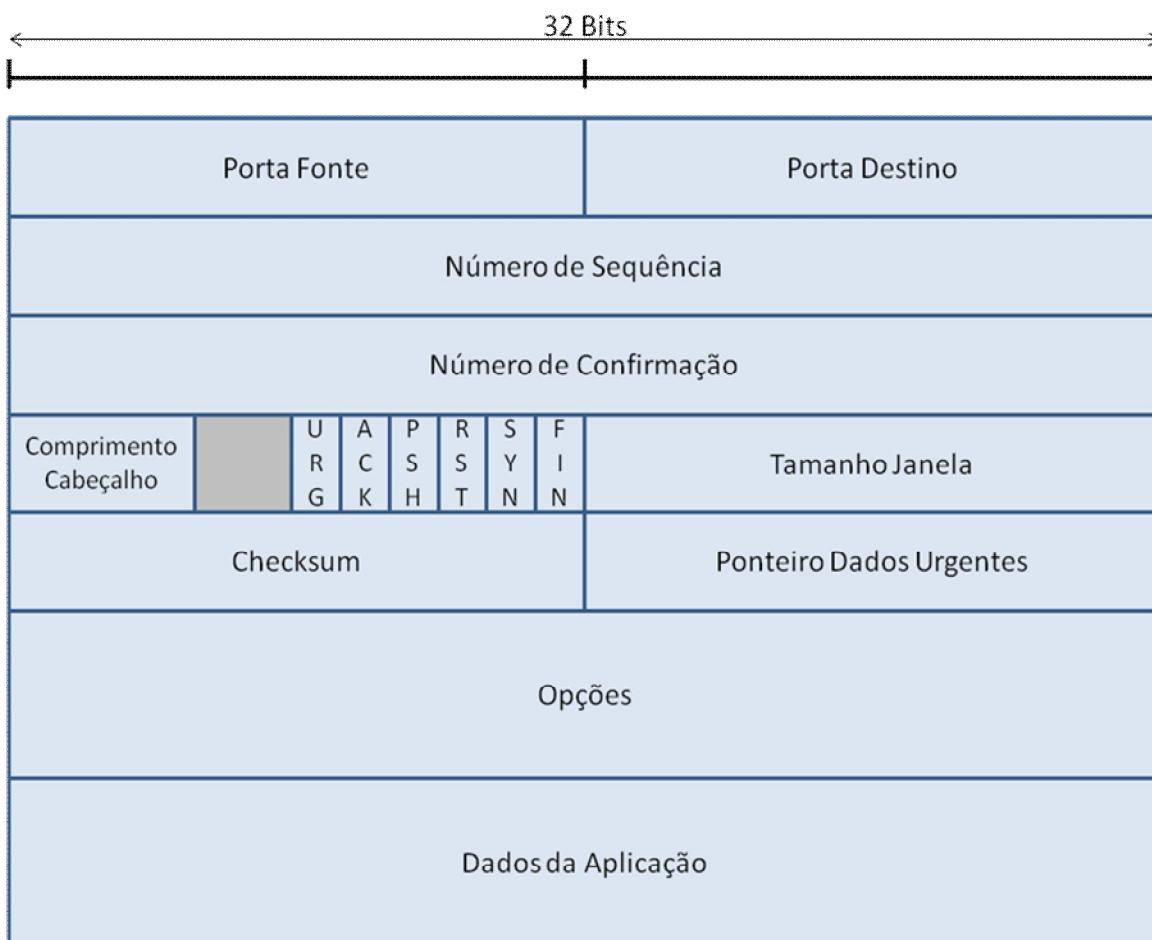


Figura 5 – Estrutura de um segmento TCP

- O campo **número de sequência** e **número de confirmação** possuem 32 bits cada, e são utilizados pelo TCP para fornecer o serviço confiável. O número de sequência indica o número sequencial do primeiro byte de dados contidos no segmento. Dessa forma, supondo que os dados fossem divididos em 1000 bytes, o

primeiro segmento teria número de sequência = 0, o segundo = 1000, o terceiro = 2000, e assim sucessivamente até atingir tamanho total da mensagem sendo transmitida. Para evitar possíveis conflitos, o número de sequência inicial pode ser escolhido aleatoriamente e os demais segmentos somam a quantidade de bytes a esse número inicial. Já o **número de confirmação** representa o próximo byte que o destinatário está aguardando. Dessa forma, se o destinatário envia um número de reconhecimento 3001, ele indica que todos os segmentos com número de sequência menores do que 3001 já foram recebidos, permitindo assim que os dados não recebidos sejam retransmitidos.

- O campo **comprimento de cabeçalho** possui 4 bits e serve para indicar o tamanho do cabeçalho, uma vez que o campo **opções** poder ter comprimento variável. Por essa razão, o cabeçalho TCP poder ter comprimento variável. Como o campo opções normalmente é vazio, o comprimento de um cabeçalho TCP típico é 20 bytes.
- O próximo campo de 6 bits não é utilizado, ele havia sido reservado para corrigir possíveis erros do projeto original, mas como o TCP foi bem organizado, isso não foi preciso.
- Em seguida o campo flag contém 6 bits. O campo **URG** indica que o campo do **ponteiro de dados urgentes** está sendo usado, indicando que a camada superior do remetente enviou dados urgentes. O campo **ACK** (de *acknowledgement*, ou confirmação/reconhecimento em português) é utilizado para indicar que o segmento possui um número de confirmação que deve ser analisado. Já o bit **PSH** tem o objetivo de informar que o destinatário deve passar os dados para a camada superior na medida em que eles chegarem, ao invés de aguardar que o *buffer* completo tenha sido recebido para realizar essa passagem. O bit **RST** indica que a conexão deve ser resetada devido algum tipo de problema. O **SYN** é utilizado para estabelecer conexões, como veremos a seguir. E por fim, o bit **FIN** indica que a conexão deve ser encerrada já que o remetente não possui mais dados para enviar.
- O campo **tamanho da janela** possui 16 bits, e indica o número de bytes que o destinatário deseja receber a partir do último byte confirmado, servindo para

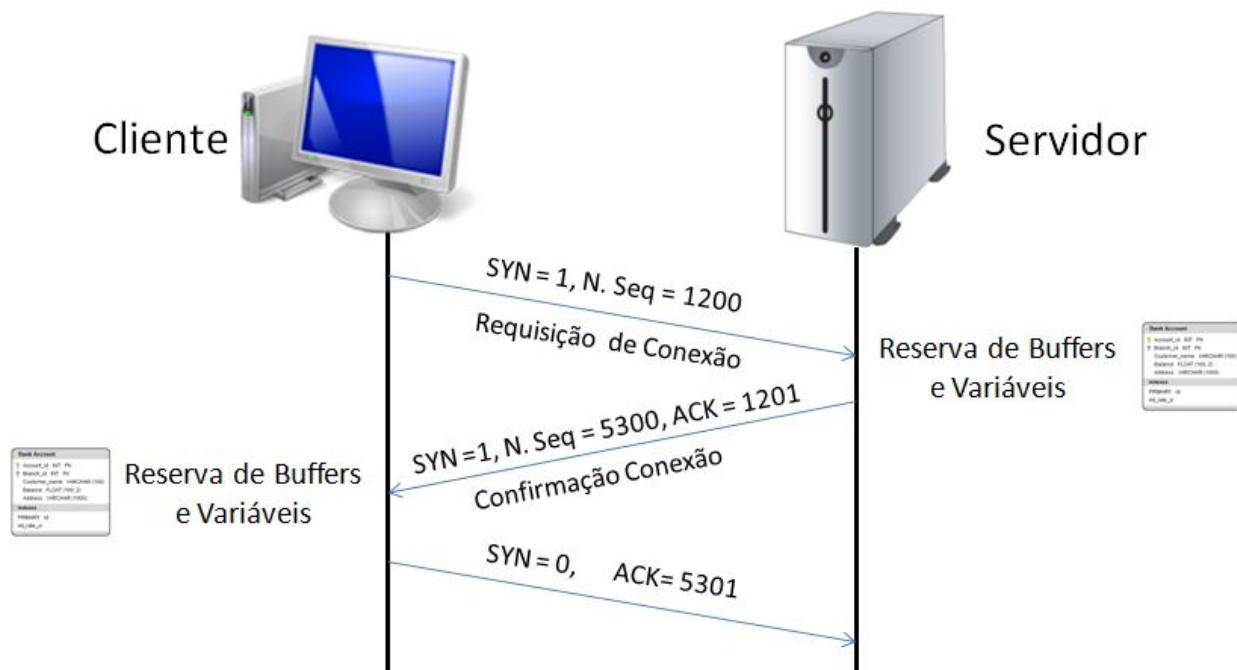
controle de fluxo. Quando esse campo possui o valor 0, o destinatário informa que recebeu os dados de acordo com o campo “número de confirmação” e que não deseja receber mais dados no momento. Para restabelecer a transmissão, o destinatário envia um segmento com o tamanho de janela diferente de 0, e com o mesmo número de reconhecimento anterior.

- O campo **opções** possui tamanho variável, é opcional, e foi projetado para oferecer recursos extras.

Em uma conexão TCP, antes do processo da aplicação enviar dados ao destinatário é preciso estabelecer uma conexão, ou seja, o remetente e destinatário precisam trocar alguns segmentos preliminares para acordar sobre parâmetros de transferência. Pelo fato do estabelecimento de conexão ser necessário, dizemos que o TCP é **orientado à conexão**. Essa conexão é estabelecida com um procedimento “*Handshake*” (em português “aperto de mãos”, ou seja, fase de estabelecimento do acordo) de 3 passos (*three-way handshake*). Nesta etapa, o lado do cliente envia um segmento especial ao servidor, sem nenhum dado de aplicação e com o bit SYN igual a 1, indicando que deseja estabelecer uma conexão TCP. Adicionalmente, esse segmento contém o número definido pelo cliente como valor inicial para o campo número de sequência (conforme vimos anteriormente), que pode ser zero. O servidor, ao receber o segmento, reserva buffers e variáveis TCP e em seguida envia um segmento TCP ao cliente como forma de confirmar a conexão. Esse segundo segmento, assim como o primeiro, não possui dados de aplicação e o bit SYN também é igual a 1. Já o campo número de sequência possui um valor inicial definido pelo servidor (também pode ser zero) e o campo número de confirmação é o campo de sequência do cliente + 1. Para finalizar, após o cliente receber o segmento de confirmação a terceira etapa consiste em o cliente reservar buffers e variáveis e enviar ao servidor o último segmento, com SYN = 0 e confirmação = sequência do servidor + 1. Esse último segmento pode conter dados da aplicação. Um *handshake* de 3 passos é apresentado na Figura 6, utilizando como exemplo o número de sequência inicial do cliente = 1200 e o do servidor = 5300.

Portanto, uma conexão TCP consiste em variáveis, buffers, porta para aplicação no lado cliente e servidor, mas nada nos elementos de rede intermediários (ex: roteadores). Através dos campos do cabeçalho apresentado o TCP fornece o serviço confiável de

entrega, oferecendo controle de fluxo, controle de congestionamento, retransmissão de segmentos perdidos e ordenação dos segmentos que chegam fora de ordem. Assim, o TCP oculta os erros existentes na camada de redes, dando a impressão à aplicação que existe um fluxo contínuo de dados, garantindo que a mensagem entregue é a mesma enviada pelo remetente e não está corrompida, está na ordem e não possui lacunas.



**Figura 6 – Estabelecimento de Conexão TCP em Três Passos (*Three-way Handshake*)**

Aplicações que precisam de um serviço de transferência confiável, como por exemplo, transferência de arquivos, correio eletrônico e web, rodam sobre TCP.

# UNIDADE 4

*Objetivo: Apresentar os protocolos SSL e TLS, que podem ser utilizados pelos protocolos da camada de aplicação para transmissão de dados com confidencialidade e integridade.*

## Serviços Seguros de Transporte: SSL e TLS

O **SSL** (*Secure Sockets Layer*, ou em português, Camada de Sockets Segura) e o seu sucessor **TLS** (*Transport Layer Security*, ou Segurança da Camada de Transporte) são protocolos destinados a prover privacidade e integridade para comunicação entre duas aplicações.

O SSL surgiu da necessidade de utilização da WEB para trocar informações confidenciais, como por exemplo, utilização para fins comerciais. Inicialmente a WEB era utilizada apenas para apresentar páginas estáticas, mas logo seu potencial para permitir transações financeiras (ex: compra de mercadoria com cartão de crédito) foi descoberto, introduzindo a necessidade de conexões seguras. Como o protocolo HTTP (apresentado na Unidade 6), utilizado como protocolo para transferir páginas WEB, não criptografa as mensagens e também não protege os dados confidenciais, várias tecnologias foram propostas para sanar esta necessidade. O protocolo que rapidamente se tornou popular para esta finalidade foi o SSL.

### Histórico

O SSL foi apresentado pela Nestcape Communications Corp. em 1994. Na época a Netscape era a empresa referência em navegadores. A primeira versão comercial, versão 2.0 (a versão 1.0 não chegou a ser lançada publicamente), foi apresentada no *Netscape Navigator* no final de 1994 e início de 1995. Posteriormente em 1995, a Microsoft lançou sua versão inicial do Internet Explorer com uma tecnologia própria de criptografia, apresentando melhorias em relação ao SSL 2.0. Em seguida (1996) a Netscape lançou o SSL 3.0 que tornou a tecnologia da Microsoft irrelevante. Na medida em que o SSL 3.0 se tornava o padrão “de facto”, a IETF iniciou a sua padronização, concluindo-a em 1999 (RFC 2246 - <http://tools.ietf.org/html/rfc2246>) de forma a torná-lo o protocolo oficial para

segurança das comunicações Web, porém com o nome de *Transport Layer Security* (TLS).

As diferenças entre o TLS 1.0 e SSL 3.0 são pequenas, mas significantes o suficiente para fazer com que as duas não interoperem. O protocolo SSL 3.0 também é referenciado como SSL3, e o TLS 1.0 como TLS1 ou também como SSL3.1. Após o lançamento do TLS 1.0, foram lançados os TLS 1.1 (RFC 4346 - <http://tools.ietf.org/html/rfc4346>) em 2006 e TLS 1.2 (RFC 5246 - <http://tools.ietf.org/html/rfc5246>) em 2008, para sanar algumas falhas de segurança identificadas nas versões anteriores.

Ao longo deste módulo o termo SSL será utilizado para designar tanto o SSL quanto o TLS, exceto quando for explicitado o uso específico

## Objetivos

Os principais objetivos do SSL são: confidencialidade da comunicação, integridade dos dados, autenticação do servidor e autenticação do cliente (embora esse último seja raramente utilizado, ele é útil para VPN SSL).

A confidencialidade da comunicação é proporcionada na medida em que apenas os dois computadores que estão participando de troca de mensagens com SSL devem ser capazes de entender as mesmas, que são transferidas criptografadas. Ou seja, o SSL evita que a comunicação seja violada. O SSL atua como se estivesse criando um túnel seguro entre as duas máquinas que se comunicam em um ambiente inseguro. Esse túnel virtual protege de forma que se alguém conseguir interceptar as mensagens trocadas, não seria capaz de decifrá-las.

A integridade dos dados é proporcionada pela proteção dada às mensagens transferidas na sessão, de forma que não possam ser modificadas desde o momento do envio pelo remetente até o recebimento pelo destinatário.

Uma sessão SSL é segura apenas se existir uma “Terceira Parte Confiável” para assegurar a autenticidade do cliente/servidor, ou seja, garantir que as partes são quem realmente afirmam ser. Essa função é exercida pelas Autoridades Certificadoras, que emitem o certificado digital do servidor ou cliente.

O protocolo SSL/TLS fica situado entre a aplicação e a camada de transporte, permitindo criar túneis criptografados para vários protocolos de aplicações que se situam acima, como por exemplo, HTTP, FTP, SMTP, dentre outros. A utilização do HTTP com SSL/TLS forma o HTTPS. Primeiramente o SSL foi utilizado sobre protocolos confiáveis da camada de transporte, como o TCP, mas também pode ser implementado para outros protocolos desta camada, como o UDP por exemplo.

### Criptografia de Chaves Simétricas e Assimétricas

Antes de prosseguirmos no entendimento do SSL, é importante conhecermos os conceitos básicos dos algoritmos de criptografia de chave **simétrica** e **assimétrica**, ambos utilizados no SSL. A parte de criptografia é bem extensa e está fora do escopo deste módulo. O objetivo é mostrar de maneira geral o conceito do funcionamento para entender o modo de operação do SSL.

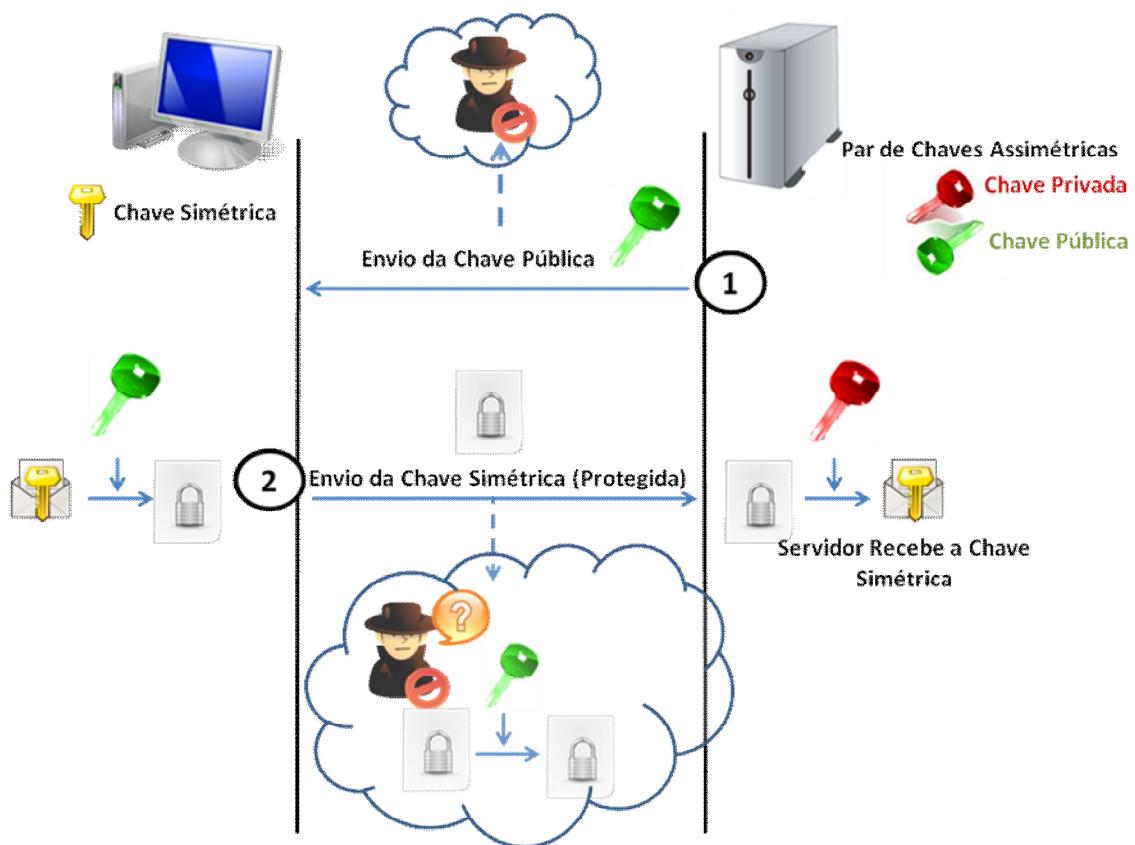
O algoritmo de **chaves simétricas** utiliza a mesma chave para criptografar e descriptografar, de forma que as partes envolvidas na comunicação devem compartilhar uma chave comum. Todos que tiverem acesso à chave podem descriptografar a mensagem. Já o algoritmo de **chaves assimétricas** utiliza duas chaves: a **chave privada secreta** e a **chave pública**. Uma mensagem criptografada com a chave pública só pode ser descriptografada com a chave privada correspondente, ou seja, o detentor da chave privada envia a chave pública para o seu interlocutor para criptografar as mensagens e mantém a chave privada somente sob o seu domínio, de forma que só ele consiga descriptografar. A chave pública, como o nome já diz, não é secreta e é compartilhada com o público. Se a mensagem criptografada com a chave pública for interceptada por um invasor, o mesmo não conseguirá decifrá-la, visto que somente o detentor da chave privada poderá descriptografá-la.

O algoritmo de chaves simétricas exige menos processamento de CPU do que o de chaves assimétricas. O algoritmo de chaves assimétricas é muito lento para codificar grandes volumes de dados. Entretanto, o algoritmo de chaves simétricas apresenta uma desvantagem, visto que a chave precisa ser compartilhada entre as partes que desejam se comunicar. E como compartilhar uma chave de forma segura em uma rede insegura como a Internet? Esse segundo problema o algoritmo de chaves assimétricas não enfrenta, visto que a sua chave pública pode ser conhecida por todo mundo.

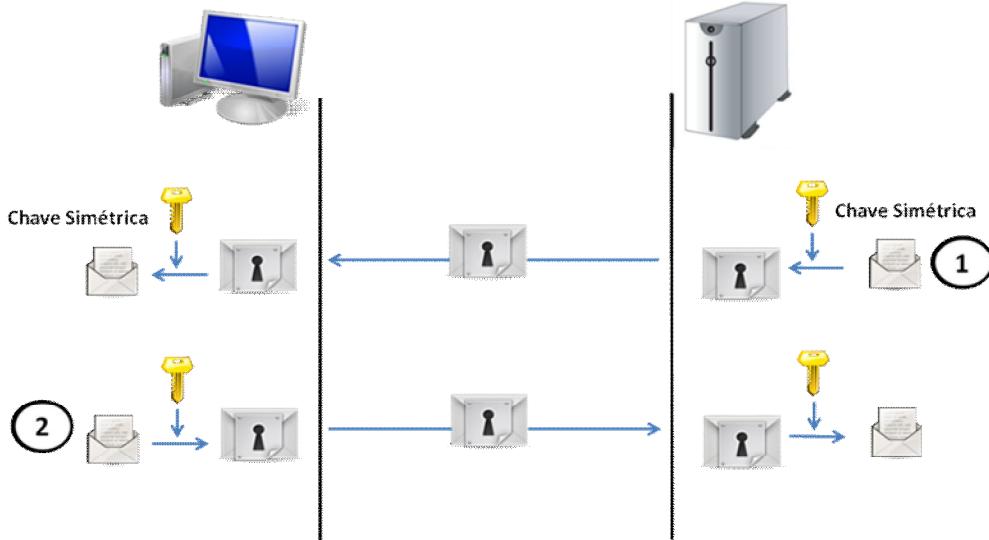
Para utilizar os benefícios e sanar as dificuldades de cada um dos dois algoritmos, surge uma combinação muito interessante: a utilização do algoritmo de chaves assimétricas para compartilhar a chave simétrica entre o cliente/servidor, de forma que todas as mensagens trocadas em seguida sejam criptografadas com a chave simétrica. Nesse modelo, o cliente obtém a chave pública do servidor com o qual deseja se comunicar, gera uma chave simétrica aleatória, criptografa-a com a chave pública obtida e a envia ao servidor. O servidor descriptografa a mensagem utilizando sua chave privada e fica de posse da chave simétrica enviada pelo cliente. Pronto, a chave simétrica foi compartilhada de forma segura e as próximas mensagens poderão ser criptografadas utilizando o algoritmo simétrico, que requer menos processamento.

A Figura 7 abaixo apresenta o compartilhamento da chave simétrica de maneira segura, demonstrando que no “passo 1” o servidor que possui um par de chaves assimétricas envia sua chave pública para o cliente através de uma rede que está “grampeada” por um invasor. Esse invasor intercepta a mensagem e também fica de posse da chave pública do servidor. Em seguida, o cliente pega a sua chave simétrica e gera uma mensagem, criptografando-a com a chave pública do servidor. No “passo 2” o cliente envia essa mensagem criptografada ao servidor, e o invasor a intercepta. Por fim, o invasor tenta descriptografar a mensagem com a chave pública do servidor, mas não consegue, visto que só o servidor com sua chave privada realizará essa descriptografia.

Dando sequência ao processo de comunicação, uma que o servidor está de posse da chave simétrica todas as mensagens trocadas entre o cliente e servidor serão criptografadas com essa chave, como apresenta a Figura 8.



**Figura 7 – Envio da Chave Simétrica criptografando-a com a Chave Assimétrica Pública**



**Figura 8 – Troca de Mensagens Criptografadas com a Chave Simétrica**

Portanto, cada sessão SSL/TLS utiliza dois tipos de criptografia, assimétrica e simétrica. A criptografia de chaves assimétricas é utilizada no "handshake" para permitir trocar de forma segura a chave simétrica. Após essa fase inicial, toda comunicação de uma sessão SSL/TLS é criptografada utilizando as chaves simétricas.

Vale citar que existe outra forma de utilização de um par de chaves privada/pública para garantir a autenticidade, só que sem garantir a confidencialidade. Neste caso, o proprietário utiliza sua chave privada para criptografar as mensagens que enviará. Qualquer um de posse da chave pública poderá descriptografar a mensagem (por isso não garante a confidencialidade) tendo certeza que ela foi “assinada digitalmente” pelo remetente que possui a chave privada. Um exemplo de utilização de autenticidade através de chaves assimétricas é pelo DNSSEC, uma extensão de segurança do DNS que estudaremos mais adiante.

### **HandShake**

O início de uma sessão SSL se dá através de um procedimento *handshake*, quando um cliente tenta se conectar utilizando SSL a um servidor que forneça suporte a esse protocolo. Nesta etapa, o servidor e o cliente “negociam” alguns parâmetros, dentre eles a versão do SSL/TLS a ser utilizada e o algoritmo de criptografia. O SSL fornece suporte a diferentes tipos de algoritmos de criptografia. Os algoritmos disponíveis para uma sessão SSL podem variar de acordo com a versão do SSL/TLS, restrições governamentais, restrições da aplicação cliente e políticas empresariais. Ainda na fase de *handshake*, após o acordo do algoritmo de criptografia, o servidor envia o seu certificado digital como forma de identificação. O certificado normalmente contém o nome do servidor, a Autoridade Certificadora (que deve ser confiável) e a chave pública de criptografia do servidor. O cliente pode entrar em contato com Autoridade Certificadora para confirmar a validade do certificado antes de prosseguir. Uma vez que o certificado é validado, o cliente gera uma chave para a comunicação (chave simétrica), criptografa-a com a chave pública do servidor, e envia para o servidor. Como a chave foi criptografada com a chave pública do servidor, apenas esse deve ser capaz de descriptografá-la, utilizando sua chave privada. Se a autenticação do cliente for necessária, durante o *handshake* o servidor requisitará o certificado do cliente.

Assim o *handshake* é concluído, a conexão segura iniciada e todas as mensagens trocadas serão criptografadas com a chave simétrica.

# UNIDADE 5

*Objetivo: Explicar o funcionamento e utilidade de um analisador de pacotes de rede, utilizando como ferramenta o software open source Wireshark. O Analisador de pacotes será útil para demonstrar nas próximas unidades os pacotes de alguns dos serviços de redes descritos neste módulo.*

## Analizador de Pacotes de Rede

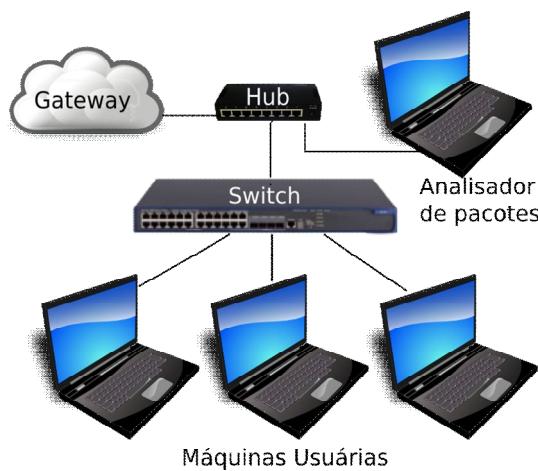
### Introdução

Em uma rede com vários serviços como DHCP, DNS, Servidor de E-mail e inúmeras aplicações sendo executadas por diversos usuários como identificar problemas? Como saber quais protocolos e portas dos protocolos de transporte estão sendo utilizadas a fim de se configurar um *firewall* corretamente?

Uma das formas de se fazer um levantamento do que está sendo utilizado em uma rede é utilizar um *software* analisador de pacotes. Há inúmeros programas capazes de fazer a captura de pacotes disponíveis na Internet, tais como tcpdump, tshark, Wireshark, além de diversos *sniffers* disponíveis. Normalmente utilizam das bibliotecas de *software* libpcap/Winpcap, que devem ser instaladas junto com os programas. O funcionamento adequado destes programas depende de alguns fatores. O primeiro deles é que estes sejam executados por um usuário administrador da máquina em que estão instalados. O motivo desta necessidade é que a placa de rede deve ser colocada em modo “promíscuo” fazendo com que todos os pacotes que cheguem a esta placa de rede sejam capturados, mesmo que estes não sejam endereçados a ela. Em outro modo de funcionamento, pacotes que não são endereçados (via MAC, *Media Access Control*, endereço da camada de enlace em uma rede *Ethernet*) a uma interface são descartados.

Outro fator importante é fazer com que todos os pacotes da rede cheguem a esta placa de rede. Há várias topologias de composição de uma rede *Ethernet* que utilizam de elementos como *hubs*, *switches* e roteadores para realizar as interconexões. Os roteadores atuam na interconexão de duas redes *Ethernet* atuando na camada IP. Os outros dois trabalham em camada de enlace, sendo a diferença básica entre eles o

acesso às informações. Os *hubs* copiam as informações que entram em uma interface a todas as outras, e os equipamentos conectados a estas verificam se estas informações são endereçadas a eles ou não. Os *switches* possuem uma inteligência no encaminhamento de pacotes. Primeiramente, através do protocolo ARP, verifica em qual porta está a máquina de destino e a partir de então, os pacotes com mesmo endereço são copiados apenas para esta interface. Então pode ser necessária a utilização de um *hub* para capturar os pacotes de uma parte da rede.



**Figura 9 - Exemplo de topologia de rede**

Na Figura 9 mostrada acima temos uma topologia de rede para melhor exemplificar os pacotes capturados. Neste caso a máquina com o analisador de pacotes capturaria todos os pacotes destinados ao *gateway*, mas não capturaria pacotes entre duas máquinas usuárias. Outras técnicas podem ser utilizadas para isto, como espelhamento de tráfego no *switch* e um tipo de ataque chamado *ARP spoofing*.

Note que as informações disponíveis nesta Unidade são para fins acadêmicos e **não devem ser utilizadas para violar a privacidade e realizar ataques a redes de empresas**, ficando o aluno responsável pela autorização de utilizar estas ferramentas em redes privadas e pelas consequências ocasionadas por elas.

Por ser *open source* e possuir uma interface muito amigável, um decodificador de protocolos muito eficiente oferecendo suporte a filtros de captura e análise de fluxos e a

possibilidade de capturar em tempo real pacotes de uma rede, utilizaremos o programa Wireshark como padrão de analisador de pacotes neste Módulo.

Para *download* e procedimentos de instalação do software visite a página do projeto: [www.wireshark.org](http://www.wireshark.org). Neste site você encontrará os arquivos de instalação e uma documentação completa que poderá sanar dúvidas quanto a utilização do programa.

## Wireshark

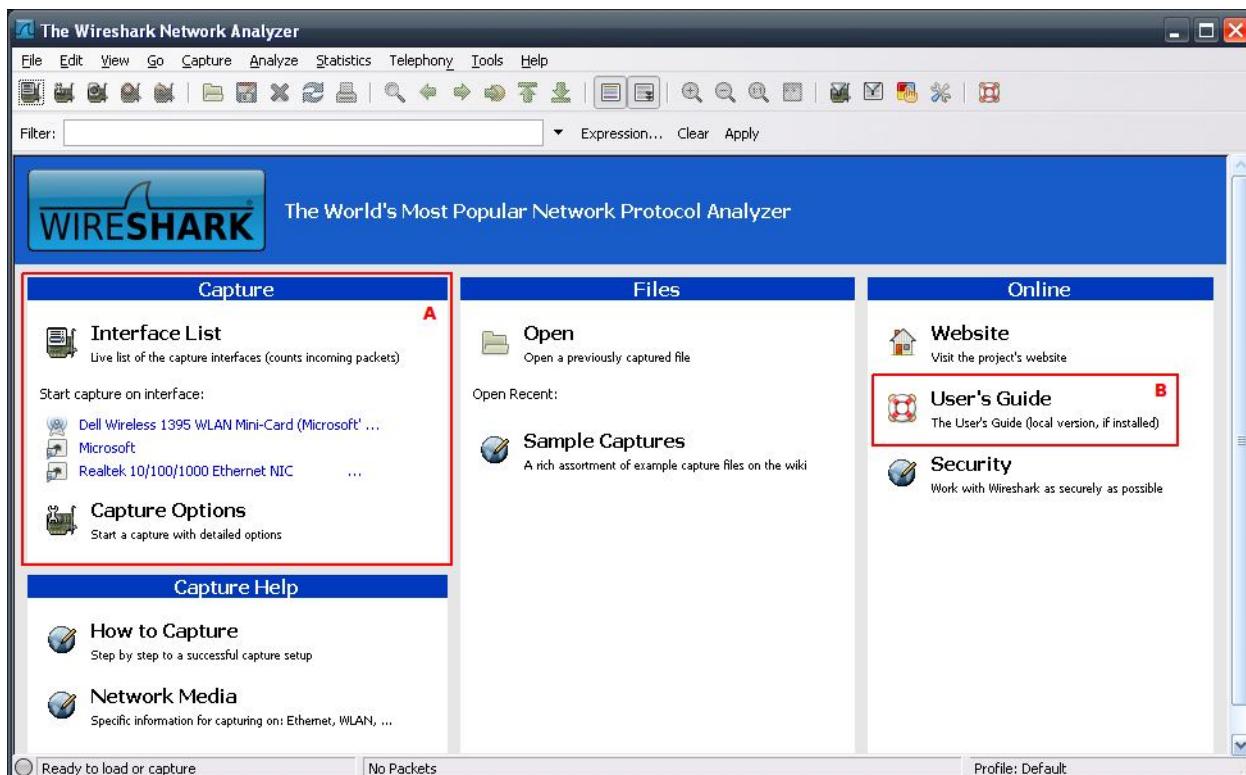
Dependendo sistema operacional (SO) no qual o programa Wireshark for instalado, o software terá suporte a diferentes tipos de rede. Isso depende principalmente da versão da biblioteca libpcap/Winpcap utilizada. O suporte do Wireshark aos diversos tipos de rede é listado de forma atualizada no site <http://wiki.wireshark.org/CaptureSetup/NetworkMedia> de acordo com o sistema operacional utilizado. Abaixo, na Figura 10, é mostrado o suporte oferecido em dezembro de 2010.

|                            | AIX     | FreeBSD | HP-UX   | Irix    | Linux            | Mac OS X | NetBSD  | OpenBSD | Solaris | Tru64 UNIX | Windows          |
|----------------------------|---------|---------|---------|---------|------------------|----------|---------|---------|---------|------------|------------------|
| <b>Physical Interfaces</b> |         |         |         |         |                  |          |         |         |         |            |                  |
| <b>ATM</b>                 | Unknown | Unknown | Unknown | Unknown | Yes              | No       | Unknown | Unknown | Yes     | Unknown    | Unknown          |
| <b>Bluetooth</b>           | No      | No      | No      | No      | Yes <sup>1</sup> | No       | No      | No      | No      | No         | No               |
| <b>CiscoHDLC</b>           | Unknown | Yes     | Unknown | Unknown | Yes              | Unknown  | Yes     | Yes     | Unknown | Unknown    | Unknown          |
| <b>Ethernet</b>            | Yes     | Yes     | Yes     | Yes     | Yes              | Yes      | Yes     | Yes     | Yes     | Yes        | Yes              |
| <b>FDDI</b>                | Unknown | Unknown | Unknown | Unknown | Yes              | No       | Unknown | Unknown | Yes     | Unknown    | Unknown          |
| <b>FrameRelay</b>          | Unknown | Unknown | No      | No      | Yes              | No       | Unknown | Unknown | No      | No         | No               |
| <b>IrDA</b>                | No      | No      | No      | No      | Yes              | No       | No      | No      | No      | No         | No               |
| <b>PPP<sup>2</sup></b>     | Unknown | Unknown | Unknown | Unknown | Yes              | Yes      | Unknown | Unknown | No      | Unknown    | Yes              |
| <b>TokenRing</b>           | Yes     | Yes     | Unknown | No      | Yes              | No       | Yes     | Yes     | Yes     | Unknown    | Yes              |
| <b>USB</b>                 | No      | No      | No      | No      | Yes <sup>3</sup> | No       | No      | No      | No      | No         | No               |
| <b>WLAN<sup>4</sup></b>    | Unknown | Yes     | Unknown | Unknown | Yes              | Yes      | Yes     | Yes     | Unknown | Unknown    | Yes              |
| <b>Virtual Interfaces</b>  |         |         |         |         |                  |          |         |         |         |            |                  |
| <b>Loopback</b>            | Unknown | Yes     | No      | Unknown | Yes              | Yes      | Yes     | Yes     | No      | Yes        | N/A <sup>5</sup> |
| <b>VLAN Tags</b>           | Yes     | Yes     | Yes     | Unknown | Yes              | Yes      | Yes     | Yes     | Yes     | Yes        | Yes              |

**Figura 10 - Suporte do Wireshark a diferentes tipos de rede por SO**

Nota-se que o suporte completo só é conseguido utilizando-se distribuições de sistema operacional Linux. Porém para nosso estudo será necessário apenas suporte a redes *Ethernet*, *WLAN*.

Quando se executa o programa, a janela principal, mostrada na Figura 11, é apresentada ao usuário e oferece diversos recursos, tais como a janela de Captura de Tráfego (em inglês, *Capture*) indicada no quadro A, e um Guia de usuário (em inglês, *User's Guide*) indicado no quadro B. A partir do quadro A pode-se iniciar diretamente a captura de tráfego clicando-se numa das interfaces listadas na Lista de Interfaces (*Interface List*). Desta forma a captura se dará de acordo com as configurações padrão do Wireshark, que não incluem filtro de captura.



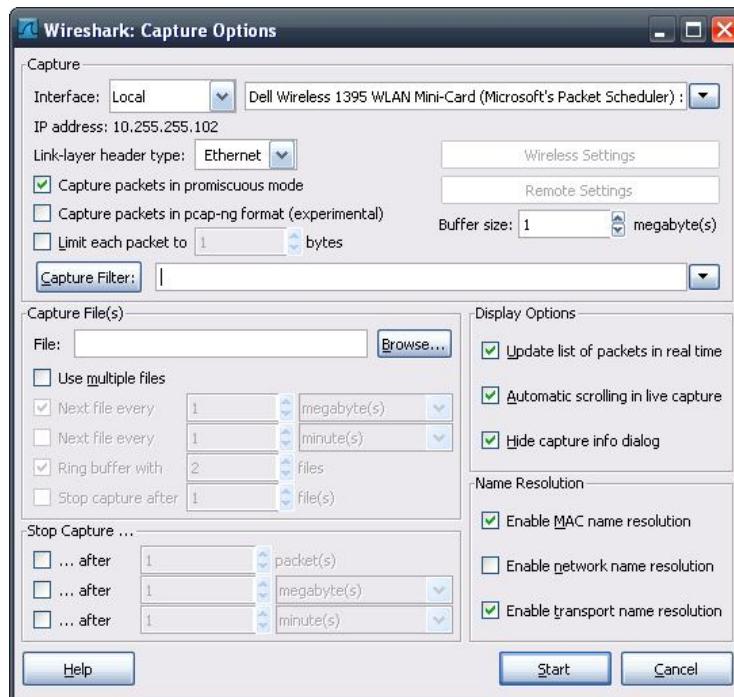
**Figura 11 - Janela principal do Wireshark**

Outra forma de se iniciar a captura de tráfego é clicar no ícone *Options* da Barra de Ferramentas principal do Wireshark, mostrado na Figura 12. Ao se fazer isto a janela de Opções de Captura de Tráfego (*Capture Options*), apresentada na Figura 13, será apresentada.



**Figura 12 - Barra de Ferramentas Principal**

A janela de Opções de Captura de Tráfego mostra os dados da interface de rede selecionada e as opções de captura de tráfego, tais como resolução de nomes do endereço MAC (*Enable MAC name resolution*) e campo para inclusão de um filtro de captura, onde apenas os pacotes que satisfizerem as condições explicitadas serão registrados. A princípio não utilizaremos os filtros de captura, mas sim filtros de exibição, que serão abordados mais a seguir. Uma vez selecionadas as opções desejadas, basta iniciar a coleta de tráfego clicando no botão *Start* e os pacotes serão capturados e mostrados em tempo real. Para parar a captura basta clicar em *Stop* na Barra de Ferramentas Principal.



**Figura 13 - Janela de Opções de Captura de Tráfego**

A janela de Captura de tráfego, Figura 14, mostra a janela de Captura de Análise de Tráfego. No quadro A dessa figura, mostram-se as informações principais dos pacotes coletados, tais como, tempo (*time*), Endereço de Origem (*Source*) e Destino (*Destination*), Protocolo (*Protocol*). Cada protocolo é marcado com uma cor diferente facilitando sua localização dentro dos inúmeros pacotes capturados.

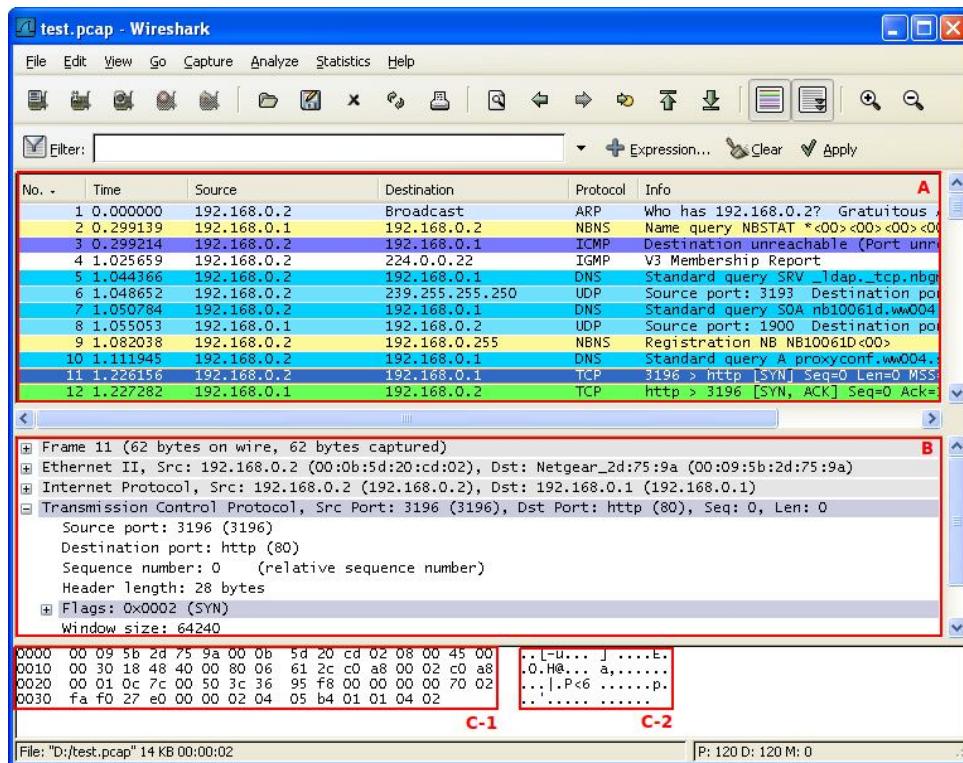


Figura 14 - Janela de Captura e Análise de Tráfego

No quadro B da Figura 14, é apresentado o detalhamento dos cabeçalhos dos pacotes, podendo o usuário verificar cada campo destes. Quando se clica em um campo do cabeçalho em B é evidenciado em C-1 os bits, convertidos em hexadecimais, referentes aquele campo. C-1 mostra todos os bits do pacote capturado. Em C-2 é mostrado os bits do pacote codificados em ASCII. Cada par de algarismos hexadecimais (8 bits) é representado por um caractere. Com isto pode-se ver as *strings* contidas nos campos de cabeçalhos e dados em codificação ASCII, já que se tornaria muito difícil a sua interpretação em hexadecimal. É em C-2 que se consegue visualizar os dados não criptografados de uma conexão. Normalmente são apresentados pontos e outros

caracteres que não têm valor semântico. Isto acontece, pois os campos de cabeçalho não possuem sempre o mesmo número de bits de um caractere ASCII.

## Filtros

Após iniciada a captura de tráfego é possível a inclusão de filtros de visualização, úteis para restringir os pacotes mostrados, facilitando a visualização de pacotes de protocolos / hosts específicos. A configuração de um filtro é feita na caixa de texto, *Filter*, que pode ser vista na Figura 13. Um exemplo de filtro seria incluir nesta caixa a string “tcp”. Com isto apenas pacotes do protocolo TCP serão mostrados. Filtros mais complexos podem ser utilizados para restringir ainda mais os pacotes como a string “tcp.dstporttcp.dstport == 21”. Neste caso só pacotes TCP com porta de destino 21 serão apresentados.

A combinação de várias strings é possível através para de funções “and”, “or” e “not” (“e”, “ou” e “não”). Um exemplo seria o seguinte filtro: “host 10.0.0.22 and not (port 80 or port 25)”. Neste filtro só serão visualizados pacotes com origem e destino no IP 10.0.0.22 e que não utilizam as portas 80 (HTTP) e 25 (SMTP).

Para auxílio na criação de filtros mais complexos ainda que este, utilize o botão “+ Expression...” na frente do campo do texto de edição de filtros, *Filter*, ou acessar o site: <http://wiki.wireshark.org/CaptureFilters>.

## Conclusão

Nesta unidade foram apresentados conceitos básicos da utilização de um analisador de pacotes com uma interface amigável e muito útil para administradores de rede em geral, o Wireshark. Aprendemos a como capturar tráfego de diferentes tipos de interface, visualizar os campos dos cabeçalhos de pacotes e criar filtros para facilitar visualização destes pacotes.



## Para sua reflexão

Note que as informações disponíveis nesta Unidade são para fins de estudo e não devem ser utilizadas para violar a privacidade e realizar ataques a redes, ficando o aluno responsável pela autorização de utilizar estas ferramentas em redes privadas e pelas consequências ocasionadas por elas.

Uma ferramenta para captura de pacotes é de valor inestimável para permitir que administradores de redes realizem o gerenciamento e monitoramento das mesmas. Mas infelizmente também são utilizadas por criminosos.

Sabemos que a apresentação das informações desta Unidade não fornecem conhecimento para uma pessoa se tornar um hacker, que precisa de muito mais conhecimento para realizar ataques. Entretanto, não custa reforçar: Aja sempre de forma ética.



# UNIDADE 6

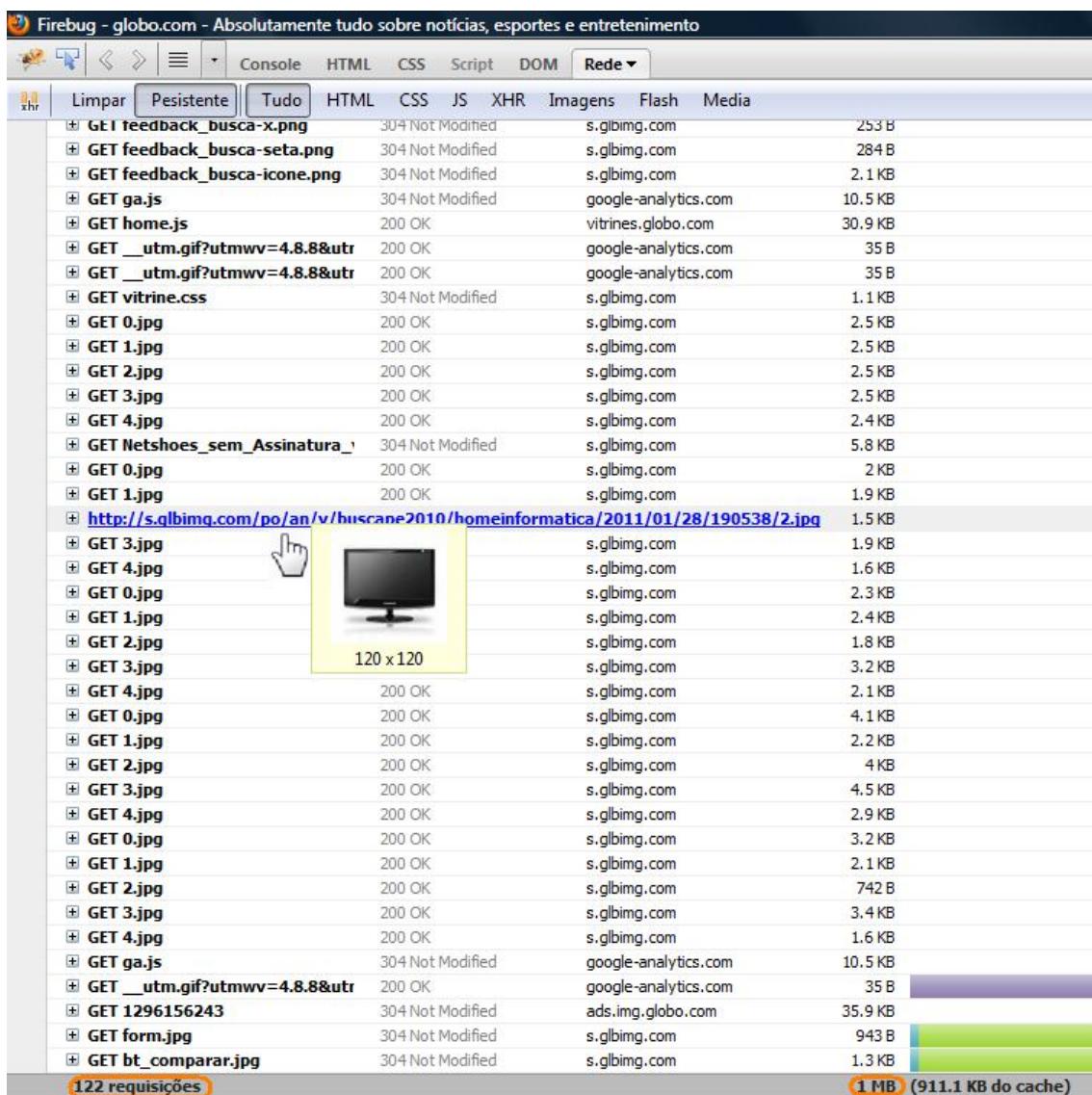
*Objetivo: Conhecer o protocolo HTTP, utilizado para transferência de páginas Web.*

## HTTP

O **HTTP** (*Hypertext Transfer Protocol*) é um protocolo cliente-servidor utilizado para transferência de páginas Web (e respectivos objetos) em toda World Wide Web, especificando como os clientes (navegadores, conhecidos também como *browsers*) devem solicitar as páginas e como os servidores devem transferi-las de volta aos clientes. O HTTP tem sido utilizado pela World Wide Web desde 1990 a partir da sua primeira versão, a de número 0.9. A sua versão mais atual, HTTP 1.1, foi oficialmente lançada em janeiro de 1997, porém atualizações e melhorias foram especificadas no RFC 2616 (<http://tools.ietf.org/html/rfc2616>) em junho de 1999. A partir do seu lançamento, essa versão foi amplamente adotada e mantém compatibilidade com a versão 1.0.

Antes de aprofundarmos sobre o HTTP, é importante entender as páginas Web. As páginas Web são acessadas através de uma URL e são compostas por **objetos**, que por sua vez são arquivos, como arquivos HTML, imagens JPEG, arquivos de áudio, etc. Na maioria dos casos, as páginas Web possuem um arquivo HTML como base e diversos objetos relacionados através das suas respectivas URLs. Uma página Web que possui texto HTML e três imagens JPEG, possui quatro objetos ao todo.

Muitas vezes acessamos sites na Internet e não nos damos conta do mecanismo que funciona por trás, como por exemplo, a quantidade de objetos transferidos. A Figura 15 a seguir apresenta a quantidade de objetos retornados ao acessarmos um portal de conteúdo (no caso abaixo utilizamos [www.globo.com](http://www.globo.com) para exemplificação). Repare que cada objeto existente na página foi retornado em um método GET (estudaremos o GET na próxima unidade), completando 122 requisições. Para efeito de demonstração de um objeto, destacamos a imagem uma televisão. A Figura 15 foi obtida da ferramenta *add-on* do Firefox chamada Firebug (mais detalhes sobre o Firebug no final desta unidade).



**Figura 15 – Demonstração de 122 Requisições de Objetos com o Firebug**

O protocolo de transporte utilizado pelo HTTP é o TCP. Como o TCP fornece um serviço confiável de transferência de dados, recuperando os dados perdidos ou reordenando-os dentro da rede, o HTTP não precisa se preocupar com os mesmos. O processo servidor, normalmente, escuta a porta padrão 80, aguardando as solicitações dos clientes que são realizadas através de uma URL.

O servidor responde às solicitações enviando os arquivos requisitados pelo cliente sem manter nenhuma informação sobre o estado do cliente, ou seja, se um cliente específico solicitar duas vezes o mesmo objeto com poucos segundos de diferença, o servidor

reenviará o arquivo visto que não possui a informação de que havia o enviado há pouco tempo. Por esse motivo, o HTTP é chamado de **protocolo sem estado**.

### Conexões Persistentes e Conexões Não Persistentes

No HTTP 1.0, a conexão TCP era encerrada após o recebimento de cada resposta pelo cliente, ou seja, caso uma página possuísse dez objetos (um arquivo base HTML e nove arquivos relacionados), era necessário estabelecer dez conexões TCP. Dessa forma, cada conexão transporta apenas uma única mensagem de requisição e uma mensagem de resposta. Como a conexão não persiste para outros objetos, ela é chamada de **conexão não persistente**. As conexões TCP podiam ser sequenciais ou paralelas, de acordo com a configuração definida no navegador.

Na época do lançamento da versão 1.0 as páginas consistiam praticamente de textos HTML, tornando aceitável esse tipo de comportamento ocasionado pelas conexões não persistentes. Entretanto, no decorrer da evolução da WEB as páginas passaram a conter vários objetos, tornando muito custoso estabelecer uma conexão TCP para transportar um único ícone.

Observando que esse modo de operação gerava *overhead* desnecessário, o lançamento do HTTP 1.1 permitiu o estabelecimento de **conexões persistentes**, inclusive esse sendo o seu modo default. Desta maneira, o servidor deixa a conexão TCP aberta após enviar uma resposta permitindo que as requisições e respostas subsequentes entre o mesmo cliente e servidor sejam enviadas por essa conexão. Caso a conexão fique um tempo sem ser utilizada, o servidor HTTP a fecha. Esse período é chamado de intervalo máximo de pausa (*timeout*), e normalmente pode ser configurado. Além do encerramento por *timeout*, essa conexão também pode ser encerrada por solicitação do cliente. As conexões persistentes também podem ser **sem paralelismo** ou **com paralelismo**. Nas conexões sem paralelismo, o cliente lança a nova requisição apenas quando a resposta anterior é recebida. Uma desvantagem deste tipo é que após o servidor enviar uma resposta, ele fica ocioso até chegar a próxima requisição, representando um desperdício de recursos do servidor. Nas conexões com paralelismo o cliente pode fazer requisições completas para os objetos relacionados, permitindo que o servidor envie todos os objetos.



## Dica

O Firebug (disponível em: <https://addons.mozilla.org/pt-br/firefox/addon/firebug/>) é uma ferramenta de desenvolvimento gratuita muito útil que pode ser integrada ao Firefox. Esta ferramenta permite editar, debugar e monitorar HTML, JavaScript e CSS enquanto você navega em uma página.



# UNIDADE 7

*Objetivo: Conhecer o protocolo HTTP, utilizado para transferência de páginas Web.*

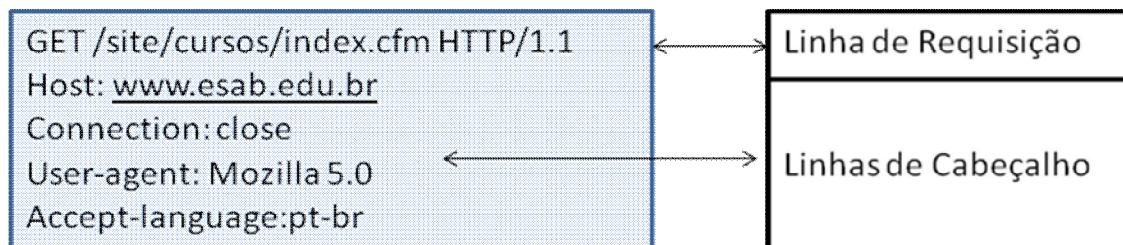
## HTTP (Continuação)

### Mensagens HTTP

Existem dois tipos de mensagens HTTP, as **Mensagens de Requisição** - enviadas pelo cliente ao servidor, e as **Mensagens de Resposta** – enviadas do servidor para o cliente. Os formatos das mensagens são definidos nos RFCs do HTTP e serão apresentados de maneira sucinta a seguir.

### Mensagens de Requisição HTTP

A seguir é apresentada uma mensagem de requisição típica, utilizando como exemplo a URL <http://www.esab.edu.br/site/cursos/index.cfm>:



**Figura 16 – Exemplo de uma Mensagem de Requisição HTTP**

Analisando a mensagem, percebemos que ela está escrita em ASCII, permitindo a identificação das palavras por seres humanos. Na imagem acima, a primeira linha é uma **linha de requisição** e as demais são **linhas de cabeçalho**. A linha de requisição possui três itens:

1. Método: Pode assumir os valores GET, HEAD, POST, PUT, dentre outros;
2. Caminho do objeto (URL);
3. Versão do HTTP.

A maioria das mensagens de requisição HTTP utiliza o método GET, que tem como finalidade solicitar um objeto (que pode ser uma página). No exemplo acima, o navegador solicita o objeto /site/cursos/index.cfm utilizando a versão 1.1 do HTTP.

Se ao invés de GET, o método fosse o HEAD, apenas o cabeçalho da página seria retornado, permitindo obter de maneira mais rápida algumas informações, como por exemplo, a data de última modificação da página (essa informação será útil para o “Cache Proxy” apresentado na Unidade 10 de “Proxy”). Já o método PUT grava páginas, permitindo criar páginas WEB em um servidor remoto. As linhas que seguem o método PUT podem conter cabeçalho de autenticação para permitir acesso ao servidor remoto.

O método POST envia dados para serem processados pelo servidor, como por exemplo, dados fornecidos pelo usuário em um formulário HTML. Também é possível passar dados através de GET, porém a diferença é que com GET os dados são exibidos ao usuário na URL e com POST os dados ficam dentro da mensagem de requisição, como por exemplo:

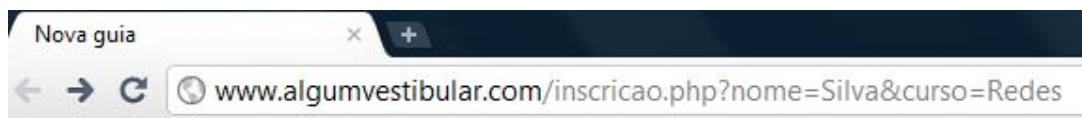


Figura 17 – Parâmetros passados via GET

```
POST /inscricao.php HTTP 1.1
...
Content-Length:50
Nome=Silva&Curso=Redes
```

Figura 18 – Parâmetros passados via Post

Portanto o método GET não é recomendado para passar dados sensíveis (confidenciais, críticos, etc).

Voltando a Figura 16, a mensagem exemplificada possui cinco linhas, mas as mensagens de requisição podem ter mais ou menos linhas, de acordo com os cabeçalhos utilizados. O cabeçalho Host é retirado da URL e tem como objetivo identificar o servidor. Ele é

obrigatório, com exceção de quando a requisição é realizada para um Proxy. Nesse último caso a primeira linha já possui a URL completa, como por exemplo:

```
GET http://www.esab.edu.br/site/cursos/index.cfm HTTP/1.1
```

### Mensagens de Resposta HTTP

As mensagens de resposta HTTP são compostas por uma **linha de status** e possivelmente por **linhas de cabeçalho** e o **corpo da mensagem**. A figura a seguir ilustra uma mensagem de resposta.

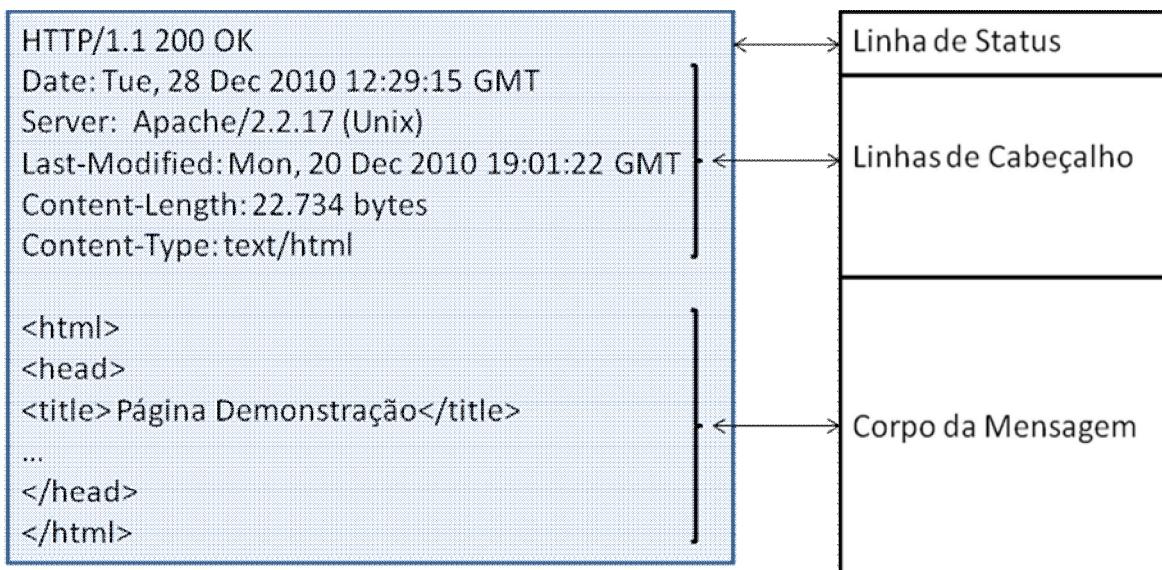


Figura 19 – Exemplo de uma Mensagem de Resposta HTTP

A linha de status possui três campos: a versão do protocolo, o código do status (três dígitos) e mensagem de status correspondente. Os três dígitos do status são utilizados para indicar o resultado da requisição (se foi atendida ou não, e o respectivo motivo). Existem cinco possíveis grupos de status, cada um deles representado pelo primeiro dígito do código do status. Por exemplo, o código de status que começa com o número 5 indica que ocorreu um problema no servidor, seja por sobrecarga, erro no código fonte, dentre outros. A Figura 20 apresenta de forma resumida os grupos com respectivo significado e exemplos.

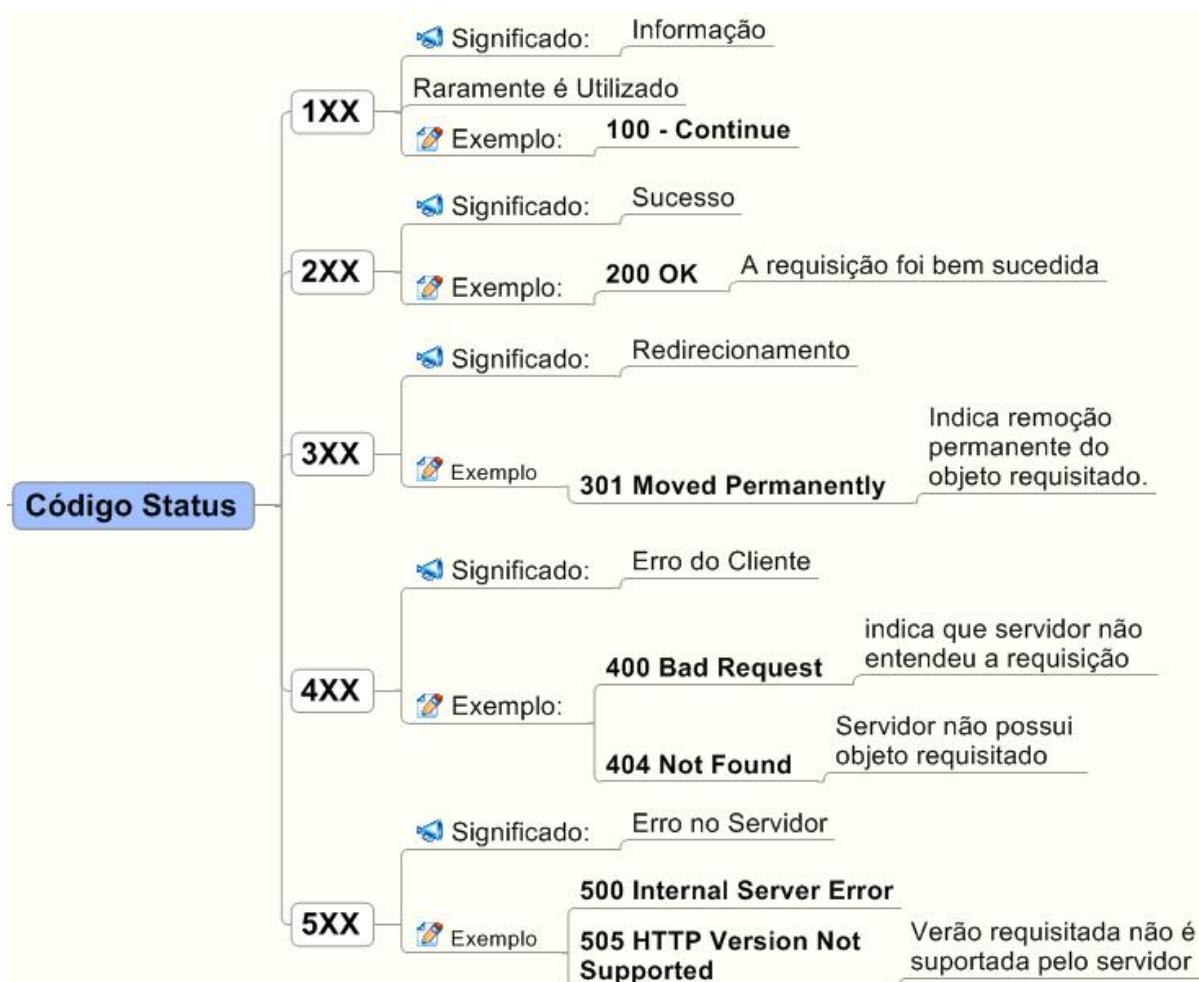


Figura 20 – Grupos de Status da Mensagem de Resposta HTTP

Voltando para a Figura 19 as linhas que seguem a **linha de status** são as **linhas de cabeçalho**. As linhas de cabeçalho são do tipo MIME e apresenta informações sobre o objeto (metadados) e sobre o servidor. A linha *Date* indica a data e a hora em que a resposta HTTP foi enviada pelo servidor, e não tem nenhuma relação com a data e hora de criação ou alteração mais recente do objeto. Essa informação está na linha *Last-Modified*, de fundamental importância para realizar cache das páginas. O cabeçalho *Server*, indica que a resposta foi gerada por um servidor Web Apache. Os cabeçalhos do tipo *Content-* tem a função de informar sobre propriedades do objeto que está sendo enviado. *Content-Length* apresenta o número de bytes e *Content-Type* o tipo do objeto, que no caso da figura é um texto HTML. A figura apresenta apenas alguns possíveis

cabeçalhos a título de representação. Para uma lista completa dos possíveis cabeçalhos, consulte o RFC 2616 que especifica o HTTP 1.1.

O corpo da mensagem contém o conteúdo em si, no caso da mensagem ilustrada, o HTML da página. O corpo da mensagem não é apresentado na mensagem de resposta a uma requisição do tipo HEAD.

### Aplicações HTTP

O protocolo HTTP serve como base para a tecnologia de comunicação que está mudando o modo de vida das pessoas, a Web. Esse protocolo da camada de aplicação WEB é implementando por um programa cliente e um programa servidor, definindo como os clientes (navegadores) solicitam páginas aos servidores Web. Os navegadores (ou *browsers*) mais utilizados atualmente são: Microsoft Internet Explorer, Mozilla Firefox, Apple Safari, Google Chrome e Opera.



Segundo a pesquisa de Servidores Web da NetCraft (<http://news.netcraft.com>) em dezembro de 2010, os principais servidores Web em utilização eram o Apache (com expressiva utilização por 59,35%) e o Microsoft Internet Information Server - IIS(22,22%). O servidor web Apache é um projeto *open-source* compatível com os protocolos padrão da internet, como o HTTP/1.1. É o servidor web mais popular da Internet desde Abril de 1996. Dentre alguns dos recursos disponíveis, podemos citar:

- Autenticação
- Suporte à conexão encriptada (SSL e TLS)
- Domínio Virtual – único endereço IP pode suportar múltiplos sites
- Conteúdo Dinâmico (ex: PHP, ASP)
- Compressão de Conteúdo – permite economizar largura de banda ao enviar conteúdo comprimido ao usuário

- Configuração de Limite de Usuários e Largura de Banda – evita saturar o servidor ou a rede



## Dica

O livro “O Mundo é Plano” (Thomas L. Friedman) é uma leitura muito interessante e apresenta como a convergência de tecnologias (em destaque redes, Internet e Web) permitiu transformações na globalização, niveling a competição entre países industrializados e países emergentes como Índia, China, dentre outros.



# UNIDADE 8

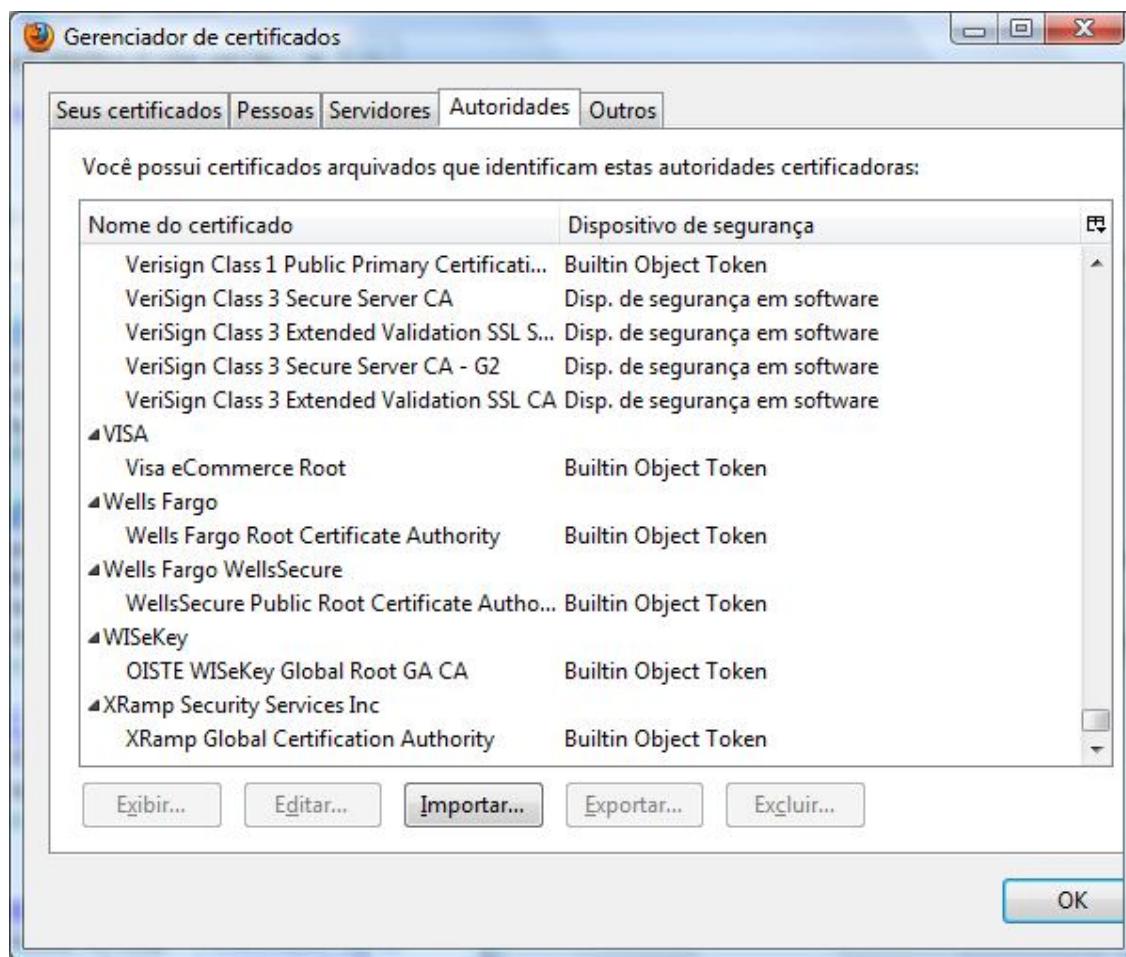
*Objetivo: Conhecer o HTTPS, utilização do HTTP sobre SSL/TLS.*

## HTTPS

O HTTPS consiste na utilização do SSL/TLS para criptografar o HTTP. É amplamente utilizado para prover segurança nas comunicações via Web. Sua utilização foi ganhando cada vez mais importância na medida em que dados sensíveis e confidenciais foram aumentando na Web, como por exemplo, dados provenientes de compras com cartões de crédito em sites. A RFC 2818 (<http://tools.ietf.org/html/rfc2818>) explica como utilizar o TLS, para prover conexões HTTP seguras. O principal objetivo é criar um “túnel” seguro em uma rede insegura, como a Internet.

Enquanto uma conexão típica HTTP é normalmente realizada para a porta 80 do servidor, uma conexão HTTPS por padrão é realizada para a porta 443. A aplicação cliente HTTPS (normalmente os navegadores) começa a conexão com o servidor iniciando o *handshake SSL* (apresentado na Unidade 4). Uma vez estabelecido com sucesso o *handshake*, o cliente pode iniciar a primeira requisição HTTP.

Assim como vimos na unidade de SSL, a confiabilidade do HTTPS depende de uma Autoridade Certificadora. Os certificados de Autoridades Certificadoras geralmente vêm pré-instalados nos navegadores. Você mesmo pode verificar no seu navegador (geralmente fica em Ferramentas > Opções de Internet – o caminho daí para frente depende do navegador que estiver utilizando). A Figura 21 apresenta alguns certificados apresentados pelo navegador Firefox.

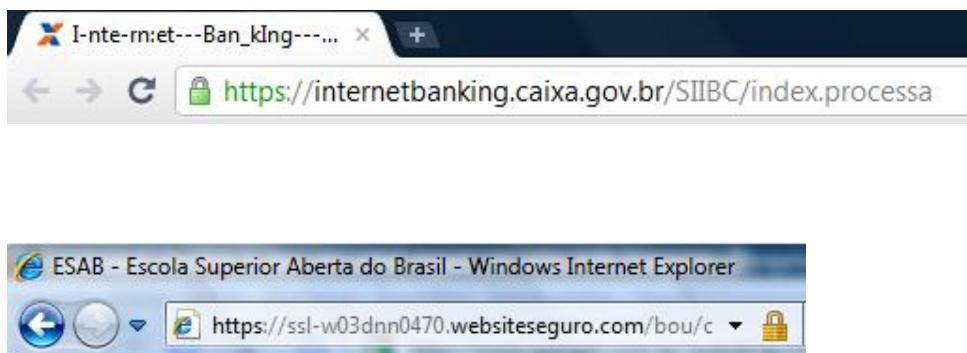


**Figura 21 – Alguns Certificados do Firefox**

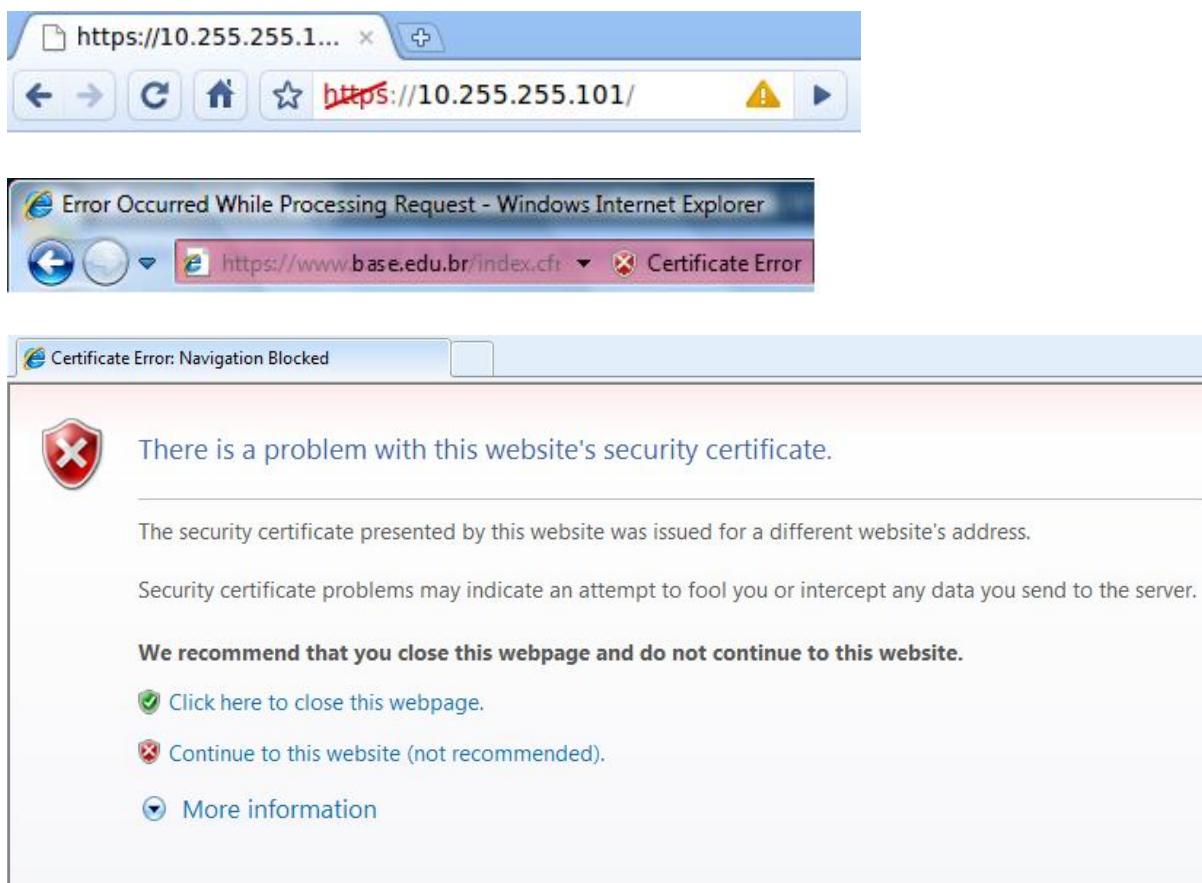
Para uma conexão HTTPS ser considerada segura, algumas condições devem ser satisfeitas:

1. O software navegador implementar corretamente o HTTPS e vir corretamente preparado com os certificados das Autoridades Certificadoras;
2. O website apresentar um certificado válido e que o identifique corretamente;
3. A Autoridade Certificadora autenticar apenas websites legítimos, sem confundir nomes;
4. A versão do protocolo SSL/TLS utilizado ser segura e livre de falhas.

A utilização do HTTPS pode ser identificada visualmente através das URLs que se diferenciam das do HTTP pela adição do “s” (<https://>) e também através de identificadores visuais (ícones e cores) disponibilizados pelos navegadores.



**Figura 22 – Indicadores Visuais de HTTPS sem problemas nos navegadores**

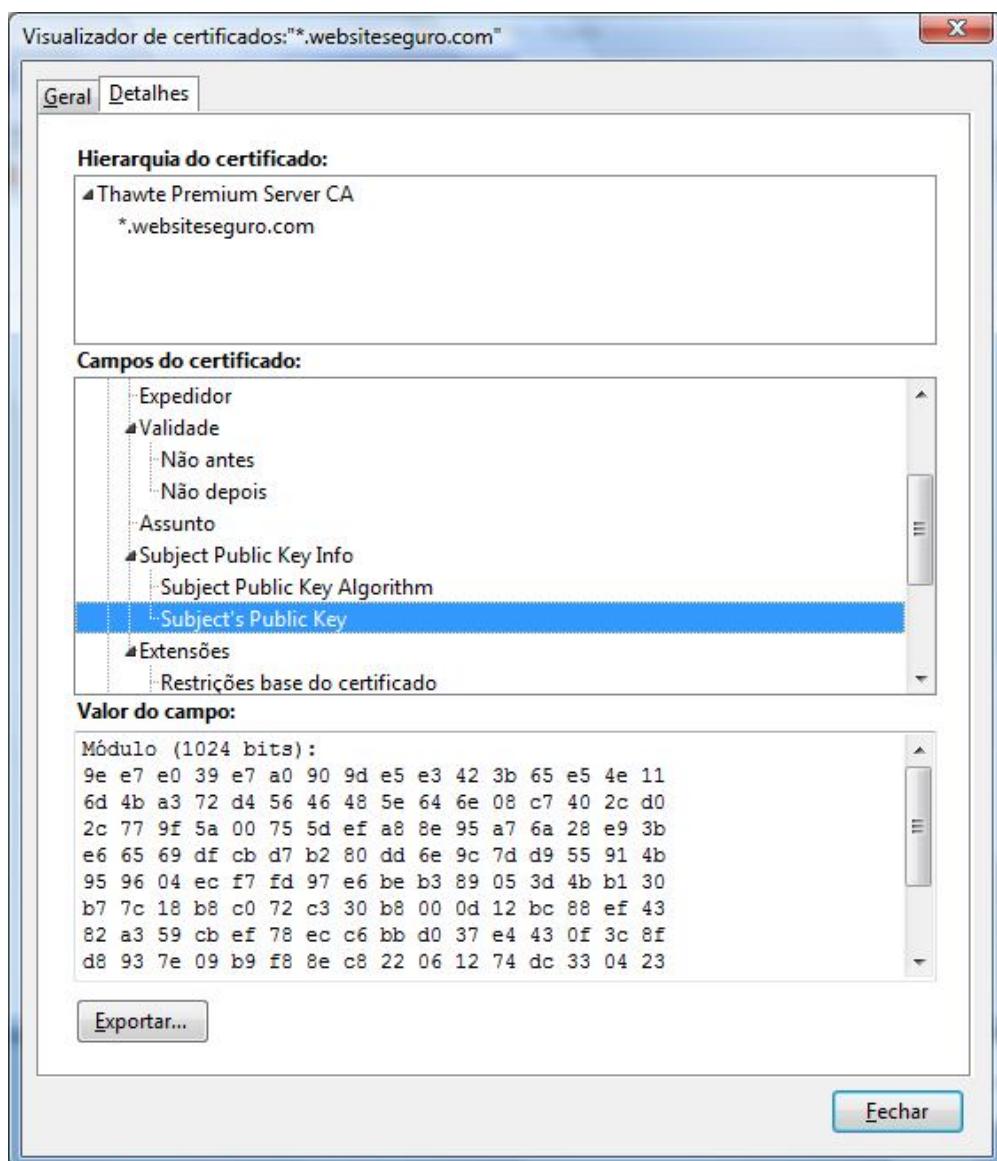


**Figura 23 – Indicadores Visuais de HTTPS que apresentam problemas nos navegadores**

A Figura 22 acima apresenta alguns indicadores visuais apresentados quando o HTTPS está adequado. Já a Figura 23 apresenta os indicadores visuais de problemas, bem como um exemplo de mensagem de erro apresentada. Podemos citar como exemplos de

problemas alertados pelos navegadores, os certificados não assinados apropriadamente, inválidos, vencidos, etc.

É possível consultar detalhes do certificado de segurança de um site que utiliza HTTPS a partir do seu navegador. A Figura abaixo apresenta alguns dados de um certificado de segurança visualizados no Firefox (Ferramentas>Propriedades da Página>Segurança>Exibir Certificado). Podemos notar na Figura que a autoridade certificadora é a Thawte, e também visualizar a chave pública do servidor no formato hexadecimal.



**Figura 24 – Alguns dados de um certificado consultado no Firefox**

# UNIDADE 9

*Objetivo: Mostrar a confidencialidade gerada pelo uso de HTTPS nos servidores web.*

## **Analizador de Pacotes de Rede – HTTP x HTTPS**

### **Introdução**

O protocolo HTTP utiliza a porta padrão TCP 80 e deixa totalmente aparente os dados que trafegam entre o cliente/servidor. Já a porta TCP 443 é utilizada por padrão para conexões HTTPS. Neste tipo de conexão os dados dos pacotes são encriptados trazendo a confidencialidade como maior benefício. Isto pode ser importante para sites de comércio eletrônico, *chat*, e-mail, páginas de *login* de sistemas, dentre outros.

Nossa abordagem prática consiste em acessar uma página em um servidor *web*, via os dois protocolos, capturar os pacotes, e tentar visualizar dentro destes o conteúdo da página.

A descrição da rede:

Rede Local: 10.255.255.0/24

Servidor HTTP e HTTPS: 10.255.255.101

Máquina cliente que acessará os sites: 10.255.255.100

Código fonte da Página: <html><body>Teste ESAB!<h1></h1></body></html>\n

O Wireshark será utilizado na máquina que acessará os sites.

### **HTTP**

Na Figura 25 é mostrada a exibição do site em HTTP visualizado em um navegador. Em seguida, na Figura 26, é apresentada a tela de captura de pacotes relativa à transferência de dados entre o servidor *web* e a máquina que está acessando o site. Nela é possível perceber nos pacotes 1, 2 e 3 o estabelecimento de uma sessão TCP, pois as flags TCP, SYN, SYN ACK, ACK, que formam o *three-way handshake* aparecem nestes pacotes respectivamente.

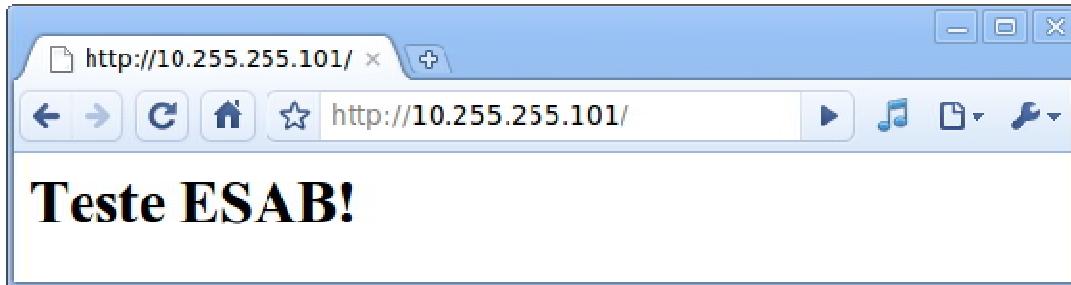
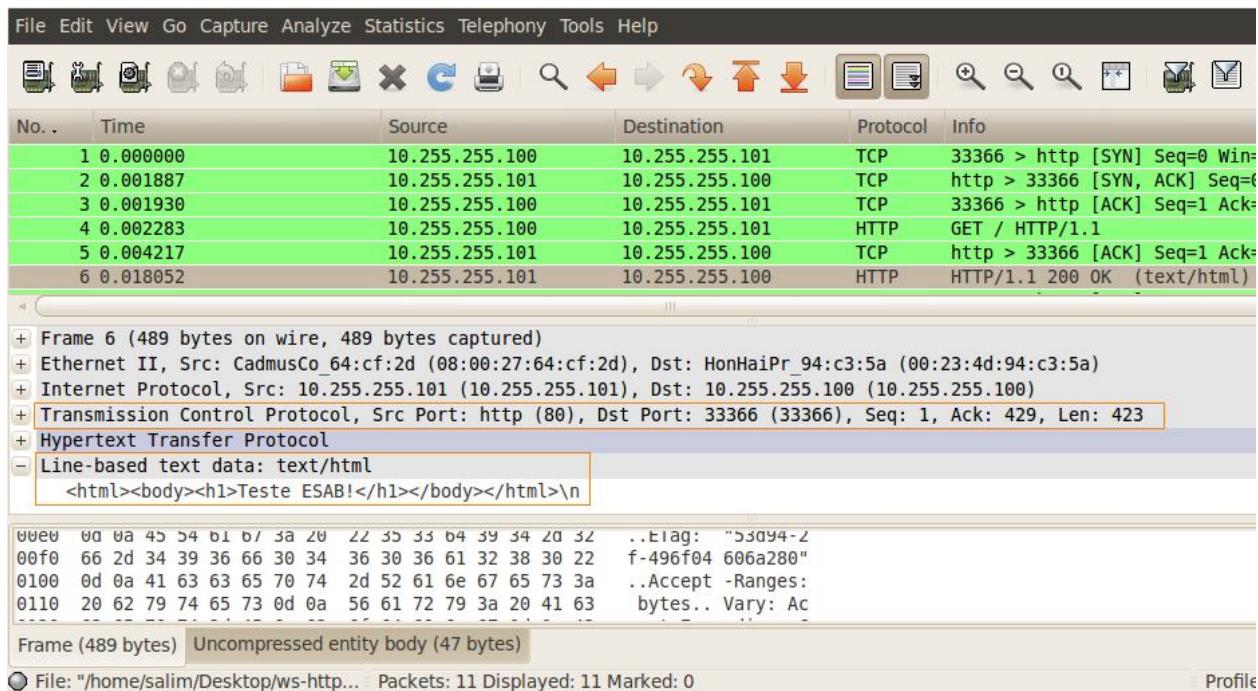


Figura 25 - Site de teste acessado em HTTP

A princípio, vemos no pacote 4 a linha de requisição HTTP para baixar a página, “GET / HTTP/1.1”. Como a página existe o servidor envia um pacote, 6, contendo o código da página junto com a linha de status da resposta “HTTP/1.1 200 OK”.

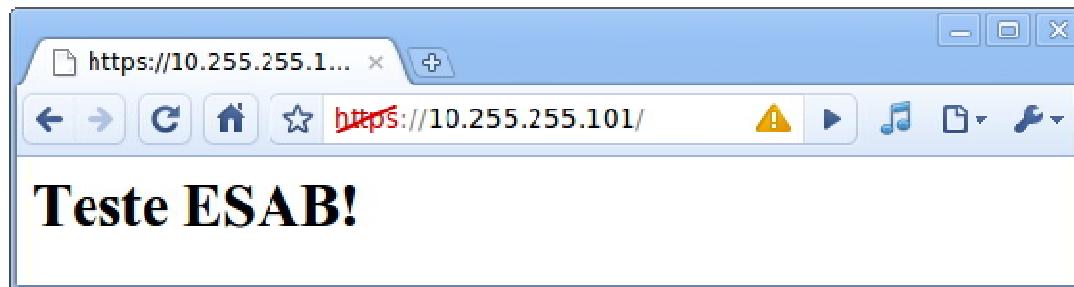
O primeiro quadro destacado em laranja na parte inferior da Figura 26 mostra o uso do protocolo TCP, sendo que a porta do servidor é a 80 (porta padrão do HTTP) e a do cliente uma porta alta gerada para essa conexão (no caso a porta 33366). No quadro seguinte, destaca-se a apresentação dos dados do pacote HTTP com o código fonte da página padrão adotada. Isto mostra a não confidencialidade deste protocolo não sendo indicado para algumas aplicações.



**Figura 26 – Pacotes capturados em HTTP e código fonte do site mostrado**

## HTTPS

Agora refaremos o acesso a página, porém utilizando o protocolo HTTPS. A página exibida no navegador é mostrada na Figura 27. Vale notar que nessa figura, o início “https” está em vermelho com um traço indicando problema no HTTPS. Esse comportamento ocorre visto que esta página de teste não possui um certificado digital assinado por uma Autoridade Certificadora.



**Figura 27 - Site de teste acessado em HTTPS**

| No. . | Time     | Source         | Destination    | Protocol | Info                           |
|-------|----------|----------------|----------------|----------|--------------------------------|
| 1     | 0.000000 | 10.255.255.100 | 10.255.255.101 | TCP      | 36066 > https [SYN] Seq=0 Win= |
| 2     | 0.001952 | 10.255.255.101 | 10.255.255.100 | TCP      | https > 36066 [SYN, ACK] Seq=0 |
| 3     | 0.001990 | 10.255.255.100 | 10.255.255.101 | TCP      | 36066 > https [ACK] Seq=1 Ack= |
| 4     | 0.003250 | 10.255.255.100 | 10.255.255.101 | SSL      | Client Hello                   |
| 5     | 0.003755 | 10.255.255.101 | 10.255.255.100 | TCP      | https > 36066 [ACK] Seq=1 Ack= |
| 6     | 0.019385 | 10.255.255.101 | 10.255.255.100 | TLSv1    | Server Hello, Certificate,     |
| 7     | 0.019416 | 10.255.255.100 | 10.255.255.101 | TCP      | 36066 > https [ACK] Seq=183 Ac |
| 8     | 0.023004 | 10.255.255.101 | 10.255.255.100 | TLSv1    | Server Key Exchange, Server He |
| 9     | 0.023030 | 10.255.255.100 | 10.255.255.101 | TCP      | 36066 > https [ACK] Seq=183 Ac |
| 10    | 0.028421 | 10.255.255.100 | 10.255.255.101 | TLSv1    | Client Key Exchange, Change Ci |
| 11    | 0.043269 | 10.255.255.101 | 10.255.255.100 | TLSv1    | Encrypted Handshake Message, C |
| 12    | 0.044652 | 10.255.255.100 | 10.255.255.101 | TLSv1    | Application Data               |
| 13    | 0.048962 | 10.255.255.101 | 10.255.255.100 | TLSv1    | Application Data, Application  |
| 14    | 0.085112 | 10.255.255.100 | 10.255.255.101 | TCP      | 36066 > https [ACK] Seq=706 Ac |
| 15    | 0.108303 | 10.255.255.100 | 10.255.255.101 | TLSv1    | Application Data               |
| 16    | 0.111330 | 10.255.255.101 | 10.255.255.100 | TLSv1    | Application Data, Application  |

+ Frame 16 (534 bytes on wire, 534 bytes captured)  
+ Ethernet II, Src: CadmusCo\_64:cf:2d (08:00:27:64:cf:2d), Dst: HonHaiPr\_94:c3:5a (00:23:4d:94:c3:5a)  
+ Internet Protocol, Src: 10.255.255.101 (10.255.255.101), Dst: 10.255.255.100 (10.255.255.100)  
+ Transmission Control Protocol, Src Port: https (443), Dst Port: 36066 (36066), Seq: 1855, Ack: 759, Len: 468  
- Secure Socket Layer  
- TLSv1 Record Layer: Application Data Protocol: http  
 Content Type: Application Data (23)  
 Version: TLS 1.0 (0x0301)  
 Length: 96  
 Encrypted Application Data: 4354F694065135AD82F79747217375DA3F3D4F7F8D6FFB70...  
+ TLSv1 Record Layer: Application Data Protocol: http  
+ TLSv1 Record Layer: Application Data Protocol: http

**Figura 28 – Pacotes capturados em HTTPS, dados da aplicação encriptados**

Na Figura 28 é apresentada toda a sequência de pacotes capturada enquanto se transferia a página padrão em HTTPS. Assim como demonstrado para o HTTP, os três primeiros pacotes se referem ao estabelecimento de uma conexão TCP. É possível ver também, nos pacotes de 8 a 11, o procedimento *handshake* do TLS realizado para efetuar a troca de chaves de criptografia, *Key Exchange*.

Seguindo a análise da Figura 28, os quadros em destaque na parte inferior desta apresentam nesta sequência o uso do protocolo TCP com a porta padrão 443 (HTTPS) como porta de origem de um pacote originado no servidor web, com destino à porta 36066 do cliente. Logo abaixo, podemos visualizar a camada de aplicação segura, *TLS Record Layer*, contendo os dados HTTP criptografados (*Encrypted Application Data*).

Os pacotes seguintes mantêm o mesmo formato, e assim não se consegue localizar o pacote que contém o código fonte da página, tornando evidente a confidencialidade dos dados trafegados a partir de então.

Mostraram-se, assim, nesta unidade os fluxos HTTP e HTTPS capturados e o padrão de mensagens trocadas para que se crie uma troca de informações confidenciais, tornando clara a utilidade da camada TLS (poderia também ser SSL) na solução de transporte de dados seguros.

# UNIDADE 10

*Objetivo: Entender o funcionamento e tipos de servidores proxy, suas vantagens e desvantagens.*

## Proxy

### Introdução

No contexto do início da popularização da Internet nos anos 90, o custo por kbps (kilobit por segundo) era muito alto, se compararmos com a oferta que temos hoje. Formas de tornar o serviço de Internet mais rápido para os usuários e ao mesmo tempo tornar o uso da banda contratada mais racional foram alvos de pesquisas. Varias soluções foram encontradas, tais como limitação do conteúdo permitido dentro de uma rede, armazenamento inteligente e transparente aos usuários de dados frequentemente baixados de uma mesma rede, serviços para compartilhamento de uma conexão com a Internet contratada com os demais computadores.

O ponto comum entre todas essas soluções, que inclusive podem ser combinadas para alcance de melhor desempenho, é a necessidade de um equipamento de rede intermediário entre as máquinas de uma rede e a conexão com a Internet disponível. Este equipamento pode ser um computador que repasse as requisições de dados e serviços de uma máquina cliente e forneça a ela os dados de resposta de interesse caso não haja alguma regra pré-configurada que impeça esta operação. Denominou-se o serviço executado neste equipamento que intermedeia as conexões entre máquinas diferentes de proxy. O serviço de proxy pode ser implementado dentro de diversos equipamentos como computadores, servidores, roteadores, entre outros.

Os proxies atuam na camada de aplicação e dependendo das funcionalidades suportadas ganham um nome diferente. Proxies focados no tráfego WWW (World Wide Web), HTTP, HTTPS, são chamados de **Web Proxies**. Um proxy que armazena conteúdos frequentemente requisitados recebe o nome de **Cache Proxy**. Há diversos tipos de proxies que serão estudados a frente.

## Tipos e funcionalidades

Um servidor proxy recebe as requisições de uma máquina usuária e pode repassá-las aos servidores de destino, bloqueá-las, e opcionalmente, alterar a requisição do cliente ou a resposta do servidor. Em alguns casos pode responder informações para uma solicitação sem nem mesmo se conectar ao servidor especificado, fornecendo informações armazenadas previamente. Um servidor que armazena dados em forma de cache em redes de computadores é denominado *cache proxy*. São instalados em máquinas com recursos computacionais tipicamente superiores às dos clientes e com poder de armazenamento elevado.

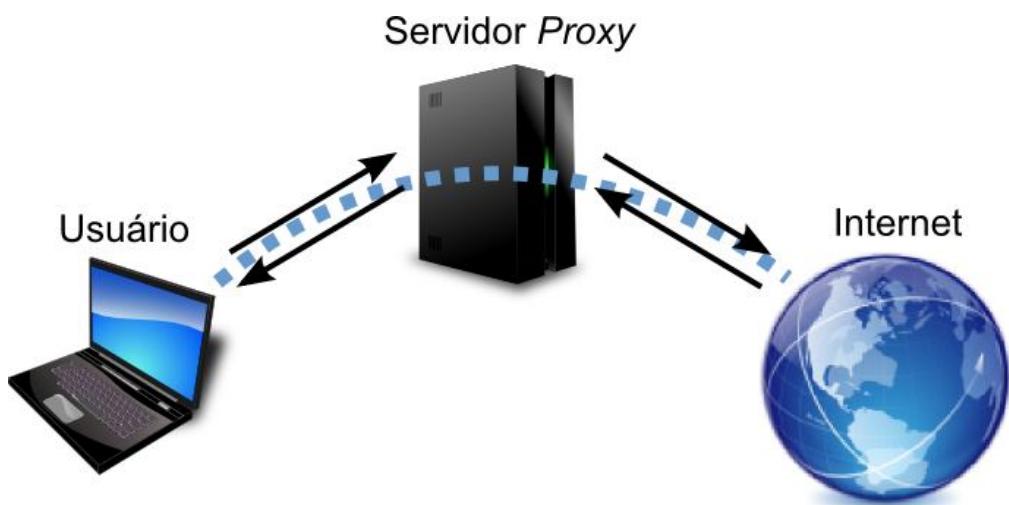


Figura 29 - Conexões realizadas através de servidor *Proxy*

## ***Cache Proxy***

Há grandes vantagens do uso de servidores *cache proxy*, uma vez que muitos dos recursos da Internet são utilizados por diversos usuários. Ainda, em um grupo, como uma empresa, muitos arquivos e sites disponíveis são comumente acessados e baixados pelos funcionários. Tradicionalmente, *sem o uso do cache proxy*, a cada requisição de usuário uma nova conexão é feita, mesmo que o recurso já tenha sido solicitado minutos atrás. O uso de servidores *cache proxy* permite que requisições já feitas anteriormente não necessitem de novas conexões com a Internet, otimizando o uso de banda tornando mais rápido o acesso à rede e diminuindo a latência de acesso, do ponto de vista do usuário.

Quanto ao modelo de funcionamento um *proxy* deve agir tanto como um servidor, quanto como um cliente. Age como servidor já que aceita requisições de usuários, como por exemplo, requisições de páginas *web*, através do protocolo HTTP. E como cliente, uma vez que tem que se conectar à máquina ou ao serviço de destino para conseguir retornar ou atualizar os documentos para seus clientes.

Uma requisição através de um *proxy* é um pouco diferente do que uma requisição normal. Vamos tomar como exemplo requisições HTTP. Uma conexão sem a utilização de um *proxy*, diretamente com um servidor WWW é do tipo:

```
GET /index.html HTTP/1.1
Host: http://www.esab.edu.br
Accept: */*
```

Já uma mesma conexão, porém realizada através de um servidor proxy seria feita da seguinte forma:

```
GET http://servidorcache/www.esab.edu.br/index.html HTTP/1.1
Accept: */*
```

Estas seriam as alterações realizadas pelos navegadores quando configurados para trabalhar com um servidor *cache proxy*. O navegador faz a requisição das mesmas URLs atribuídas pelo cliente adicionando o nome do servidor proxy de destino, e com isso, este servidor tem todas as informações necessárias para verificar se possui a página em seu cachê ou então fazer a requisição ao servidor remoto especificado na URL, em caso negativo.

Em casos de requisições utilizando outros protocolos, como por exemplo, o FTP, as alterações são um pouco mais complicadas. Além de o servidor utilizar o protocolo FTP para requisitar o arquivo do servidor, ele deve transformar esta requisição FTP em uma resposta HTTP para o cliente. Isto significa incluir os cabeçalhos *Content-Length*, *Last-Modified*, e *Content-Type*. Um comando FTP *list*, por exemplo, é retornado como um documento HTML.

```
GET ftp://ftp.ircache.net/download/README HTTP/1.1
Accept: */*
```

Um ponto que pode preocupar os administradores de uma rede é a garantia de consistência do conteúdo acessado. Para isto, uma questão importante é determinar quando os objetos serão atualizados ou removidos visto que alguns arquivos podem

permanecer estáveis por um longo tempo e de repente mudar, enquanto outros mudam diariamente.

Quando um conteúdo é acessado, um objeto é armazenado e a ele é atribuído um tempo de vida em um cabeçalho. Se o tempo do objeto expirou, o servidor original será consultado para revalidá-lo.

Um objeto armazenado no cache contém em seu cabeçalho o campo *Last-Modified* (em português: “Última Modificação”), que indica quando ele sofreu a última modificação. O servidor *proxy* pode usá-lo para fazer uma requisição *If-Modified-Since* (“Se Modificado Desde a Data”) ao servidor *web* remoto e, a partir da comparação das datas, saber se o objeto foi alterado. Se o documento não foi modificado, não será retornado pelo servidor *web*. O servidor receberá como resposta, informações como a nova data de expiração. Se o objeto foi modificado, este será retornado.

Mesmo sendo máquinas superiores às utilizadas pelos usuários, o armazenamento de informações possui um limite. Quando os diretórios de *cache* estão cheios, alguns objetos devem ser removidos para que novos objetos sejam armazenados. Apenas a escolha de exclusão dos objetos mais antigos não seria suficiente uma vez que este objeto pode não sofrer alterações desde a sua publicação na rede. Sendo assim, a escolha é feita usando algoritmos de substituição, em conjunto com algumas regras configuradas pelo administrador.

Um *cache proxy* especializado em requisições *web* são chamados de *web proxy*.

# UNIDADE 11

*Objetivo: Entender o funcionamento e tipos de servidores proxy, suas vantagens e desvantagens.*

## Proxy (Continuação)

### Web Proxy com Filtro

É comum o uso associado de servidores *proxy* com um *firewall*, aumentando a segurança e o controle de acesso de conteúdos disponíveis na Internet. Um *web proxy* com filtro provê um controle administrativo ainda maior sobre o conteúdo que poderá passar pelo servidor até o cliente. É comumente utilizado para adequar o uso da Internet por parte dos funcionários a uma política estabelecida pela empresa que os contratou.

Sua forma de ação é análoga aos filtros de camada 3 e 4. No entanto, além de poder bloquear acesso a determinado IP ou não permitir o uso de determinado protocolo dentro da rede, tal como o ICMP (ping), também funcionam analisando o tráfego da camada de aplicações, examinando todo o conteúdo dos pacotes. Alguns métodos usados para filtrar o conteúdo da camada de aplicação incluem: listas negras de URL e DNS, filtro por expressão regular de URL, filtros MIME (*Multipurpose Internet Mail Extensions*).

Um exemplo comum de uso deste tipo de proxy é para bloquear o acesso a conteúdos eróticos. Pode-se incluir uma regra para que todas as URLs que contenham palavras como “sexy”, “playboy”, entre outras, sejam bloqueadas, e no lugar do *website* de destino, seja apresentado ao usuário uma página de advertência.

Normalmente, este tipo de *proxy* produz *logs* contendo informações detalhadas sobre as URLs acessadas por usuários específicos, podendo também gerar estatísticas da banda utilizada e auditorias sobre o uso dos recursos de rede contratados. Estas informações podem também ser utilizadas para incrementar os filtros existentes, tornando-os ainda mais eficazes.

## Classificação quanto à forma de utilização

Como dito anteriormente, um servidor *proxy* atua na camada de aplicação e pode ser utilizado pelos usuários finais de diversas maneiras. Uma dessas formas é configurar a aplicação que deverá utilizar o *proxy* com as informações necessárias para a utilização do serviço. Por exemplo, para acesso à web normalmente utilizamos os navegadores. Assim, caso um administrador de rede queira poupar banda da sua rede com um sistema de *cache* e filtrar o conteúdo que pode ser acessado, ele pode conseguir isto configurando um *proxy* em todos os navegadores disponíveis nas máquinas dentro de sua rede.

Esta forma de utilização não é escalável. É custosa demais a configuração e manutenção desta em redes maiores, pois a configuração é feita de forma manual e no nível de usuário de computador “não administrador”. Seria necessário configurar os navegadores de todas as contas de usuário em todas as máquinas da rede. Além disso, nada impede que um usuário desconfigure o uso do *proxy* escolhido pelo administrador para burlar um filtro de acesso de conteúdo por exemplo. Para solução deste problema surgiu o **Proxy de Intercepção** (*Interception Proxy*).

### **Proxy de Intercepção**

Este tipo de *proxy* é erroneamente chamado de **Proxy Transparente**, embora isto ocorra de forma frequente. A RFC 2616 que especifica o HTTP/1.1 (<http://tools.ietf.org/html/rfc2616>) define um *proxy* transparente como aquele que não modifica a requisição ou resposta além do que é necessário para a autenticação do *proxy* e identificação do cliente. *Proxy* não transparente é um *proxy* que modifica o pedido ou resposta, a fim de fornecer alguns serviços adicionados ao agente usuário, tais como serviços de grupo de anotação, a transformação de tipo de mídia, a redução de protocolo, ou anônimo filtro.

Este erro de chamar de **Proxy Transparente** ocorre, pois, num **proxy de intercepção**, definido pela RFC 3040 (*Internet Web Replication and Caching Taxonomy*, <http://tools.ietf.org/html/rfc3040>) o cliente não precisa configurar um *proxy* em seu navegador e não detecta diretamente que suas requisições estão sendo enviadas via *proxy*.

Um proxy de interceptação combina um **servidor proxy** com um **gateway** ou **roteador** (geralmente com recursos NAT). As conexões feitas pelos navegadores do cliente através do gateway são desviadas para o proxy sem configuração do lado do cliente (e muitas vezes sem seu conhecimento).

Porém a utilização de um *proxy* de interceptação não resolve todos os problemas de limitação de acesso. Suponha que um determinado *proxy* só aceite requisições HTTP e HTTPS que não contenham a palavra “sexo” em sua URL. Um usuário poderia burlar este controle de acesso utilizando um **web proxy anônimo** que funciona com HTTPS por exemplo. Este *proxy* irá camuflar as URLs de sites que contêm a palavra sexo utilizando criptografia. Uma solução para isso seria bloquear também as palavras mais comuns que compõem URLs de proxies anônimos. Mas isto ainda não impede que um usuário avançado configure um *proxy* em sua casa e o utilize.

### **Proxy Anônimos**

Um servidor *proxy* anônimo geralmente tenta anonimizar a navegação na *web*. Como esses são normalmente difíceis de controlar, são especialmente úteis para aqueles que procuram o anonimato online. Alguns usuários estão interessados no anonimato apenas para maior segurança, escondendo suas identidades a partir de sites potencialmente maliciosos, ou para facilitar direitos constitucionais como a liberdade de expressão. O servidor de destino recebe solicitações a partir do servidor *proxy* anônimo, e, portanto, não recebe informações sobre o endereço do usuário que o requisitou. No entanto, os pedidos não serão anônimos para o servidor *proxy*, e assim um grau de confiança deve estar presente entre o servidor *proxy* e o usuário. Para aumentar esta relação de confiança pode ser requisitada a autenticação do cliente, antes que o *proxy* possa ser efetivamente utilizado.

Porém quando não há essa relação de confiança, o *proxy* pode oferecer riscos à privacidade do usuário, substituindo os dados retornados para difusão de softwares maliciosos, envio de mensagens não autorizadas, dentre outros.

### **Conclusão**

O controle de acesso é cada vez mais necessário, em empresas de qualquer tamanho. Os benefícios proporcionados pela maior difusão da Internet são evidentes, como:

- Agilidade na troca de informações com outras empresas, funcionários e clientes;
- Relacionamentos pessoais e familiares;
- Comércio eletrônico;
- Distribuição e compartilhamento de conteúdo, etc.

Aliado a estes fatores, há uma ampliação constante da largura de banda por parte de empresas e usuários domésticos, incentivando-os a utilizar serviços antes inviáveis. Este cenário gera uma demanda sempre crescente de largura de banda.

Deste modo, é extremamente necessária, a definição de uma política de uso da rede, em que todos os envolvidos tenham consciência do modo como os recursos disponibilizados devem ser utilizados. Faz-se necessário um controle de registro de todos os acessos, bloqueando aqueles considerados indevidos, reduzindo a utilização de banda em ações estranhas às atividades de uma empresa, mantendo-a disponível e melhor preparada para a prática de atividades legítimas. Isto diminui a circulação de vírus, worms, programas piratas e outros males que apresentam riscos às empresas.

Uma das formas de se obter todo esse controle, adicionando ainda economia de banda, via uso de *cache* de conteúdos frequentemente acessados, é utilizando *proxies*.

# UNIDADE 12

*Objetivo: Entender a utilidade e funcionamento do serviço de tradução de “nome-para-endereço IP”, o DNS.*

## DNS (Domain Name Server)

### Introdução

Como visto anteriormente, na Internet ou em redes privadas, computadores ou quaisquer equipamentos ativos que as compõe, como *hubs*, *switches*, roteadores, *firewalls*, *proxies*, entre outros, são identificados através de pelo menos um endereço único na camada de rede (camada 3 do modelo OSI), o endereço IP. Em uma rede pequena é fácil a memorização dos endereços de cada máquina. Porém, quando este número de máquinas cresce a tarefa se torna humanamente impossível e insustentável.

Além disto, nos deparamos com outro problema. Suponha que por motivos da administração de rede, ou por motivo de manutenção de servidores, precisássemos trocar determinados serviços, como, por exemplo, um *website*, de servidor. Imagine o seguinte caso: alunos acessam diariamente o *website* da ESAB pelo IP 201.76.48.129. Porém, a instituição decide alojar o *website* em outro servidor de IP 201.76.48.128, que já possui outros serviços internos, utilizados por seus funcionários. Se o acesso fosse feito diretamente pelo endereço IP, o quanto dispendioso seria avisar a todos os alunos ou a todos os funcionários que utilizam estes serviços internos que o endereço do servidor mudou? Não seria muito mais fácil dar um nome a estes serviços? O aluno passaria a acessar o *website* da ESAB pelo nome [www.esab.edu.br](http://www.esab.edu.br) e os funcionários a rede interna pelo nome [intranet.esab.edu.br](http://intranet.esab.edu.br). No entanto, os equipamentos da rede só reconhecem endereços numéricos. Assim, seria necessário algum mecanismo para converter estes nomes em endereços numéricos.

No início do desenvolvimento das redes de computadores, na ARPANET (*Advanced Research Projects Agency Network*, ou Agência de Projetos de Pesquisa Avançada em Redes), havia simplesmente um arquivo texto chamado “hosts.txt” mantido em cada

máquina que listava todos os nomes dos computadores da rede (*hosts*) e seus endereços IP. Toda dia, todos os usuários acessavam um arquivo central no local em que era mantido, atualizavam e o copiavam. Para uma rede de algumas centenas de grandes máquinas, essa estratégia funcionava razoavelmente bem.

No entanto, quando as redes começaram a crescer e milhares de computadores domésticos foram conectados, todos perceberam que essa estratégia não poderia continuar a ser utilizada para sempre. Por um lado, o arquivo de *hosts* se tornaria grandes demais e as consultas seriam difíceis. Por outro, todos estariam frequentemente sujeitos a conflitos de nomes de *hosts*, o que é muito preocupante. A menos que este arquivo de nomes fosse gerenciado de uma forma centralizada, algo totalmente fora de cogitação em uma enorme rede internacional, como a Internet, este problema não seria solucionado. Para resolver esses problemas, foi criado o **DNS** (*Domain Name System* — Sistema de Nomes de Domínios).

Portanto, o objetivo principal do DNS é traduzir um nome para um endereço IP. O nome é conveniente para associação dos seres humanos, enquanto o endereço numérico (IP) é necessário para acessar as máquinas. Funciona de forma parecida a uma lista telefônica. Se você sabe o nome de uma pessoa, mas não o seu telefone, você recorre à lista e através do nome obtém o número desejado.

A essência do DNS é a criação de um esquema hierárquico de atribuição de nomes baseado em domínios e de um sistema de bancos de dados distribuídos para armazenamento e consulta desse esquema de nomenclatura. O protocolo do DNS é definido nas RFCs 1034 (<http://tools.ietf.org/html/rfc1034>) e 1035 (<http://tools.ietf.org/html/rfc2616>). Assim, de forma resumida, o DNS é um protocolo da camada de aplicação e sua implementação contempla um banco de dados distribuído para traduzir nomes para endereço IP, de forma a permitir que um cliente acesse o servidor desejado.

Diferentemente dos outros protocolos da camada de aplicação, o usuário não interage diretamente com os serviços do DNS. Entretanto, ele é utilizado por outros protocolos da camada de aplicação, como o FTP, SMTP e HTTP, fornecendo “serviços internos” na Internet. Por exemplo, quando um navegador (um cliente HTTP) precisa acessar o site [www.esab.edu.br](http://www.esab.edu.br), antes de conseguir estabelecer a conexão ele precisará obter o

endereço IP do respectivo servidor. Dessa forma, a aplicação cliente DNS presente na mesma máquina do navegador vai providenciar essa tradução, conforme explicaremos nesta Unidade. Uma vez obtido o endereço IP, o cliente se conectará ao servidor realizando o *three-way handshake* do TCP.

Podemos perceber que o DNS precisa de um tempo para realizar o seu serviço, adicionando o que pode ser considerado um “atraso” nas aplicações Internet. De forma geral, esse atraso é praticamente imperceptível, mas em alguns casos pode ser considerável. Para minimizar esse risco e o tráfego DNS na rede, são utilizados caches em servidores DNS próximos, conforme veremos mais adiante.

### **Domínios e Hierarquia de Servidores**

Para entendermos a definição do DNS, é preciso que antes entendamos o que é um domínio. Um exemplo de domínio é: **esab.edu.br**. Cada parte desse nome de domínio (esab, edu, br) representa um nível de servidores. Pode ser que exista um servidor por nível, mas é fortemente recomendado que haja mais que um servidor por nível visando tornar o serviço de DNS um serviço de alta disponibilidade.

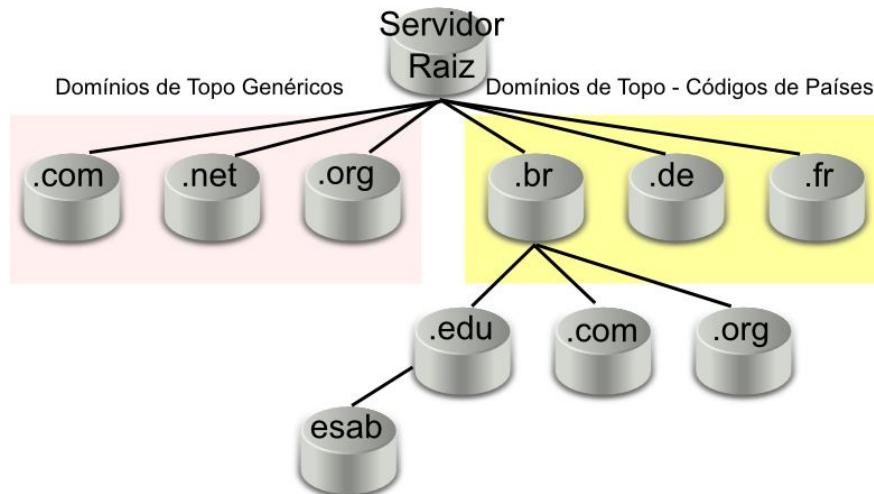
A cadeia começa com os servidores raiz, ou *root servers*. Eles são representados por um ponto no fim do domínio, ou por uma string vazia (""). O domínio **esab.edu.br** representa a mesma coisa que **esab.edu.br**. para as aplicações que utilizaram esse tipo de identificação. A seguir é mostrada na Figura 30 a distribuição geográfica destes servidores raiz.



**Figura 30 - Localização dos root-servers no mundo - <http://root-servers.org>**

As bases de dados destes servidores são pequenas (em torno de 200KB), dividida em domínios genéricos e geográficos. Porém por serem o início da cadeia de registro de nomes, são os servidores de maior responsabilidade. A ICANN (*Internet Corporation for Assigned Names and Numbers*) é uma corporação sem fins lucrativos habilitada a fiscalizar tarefas relacionadas com a Internet em nome do governo dos EUA e de outras organizações notórias como IANA (*Internet Assigned Numbers Authority*). Há um comitê da ICANN denominado *DNS Root Server System Advisory Committee* que é responsável pela autorização de operação de servidores de nomes raiz.

Em nosso exemplo, esab.edu.br, o servidor raiz indicaria que o próximo nível da base de dados que deve ser consultado seria o do servidor de domínio geográfico **br**, indicando os IPs destes servidores. Por sua vez, eles indicariam os IPs dos servidores responsáveis pelo nome **edu**, utilizados por instituições de educação, que indicariam os **servidores de nome** da ESAB. Assim, podemos perceber que uma URL é analisada da direita para esquerda, na ordem “.”, “br”, “edu” e “esab”. No Brasil os registros dos servidores de primeiro nível (com.br, gov.br, org.br, ...), são de responsabilidade do Registro.br. A hierarquia dos servidores pode ser representada como na Figura 31 abaixo.



**Figura 31 - Hierarquia de Servidores DNS**

Em nosso exemplo chegamos até o nível do servidor DNS da ESAB. Este servidor é o responsável pelos nomes dos *hosts* desta instituição de ensino. Quando digitamos em nosso navegador [www.esab.edu.br](http://www.esab.edu.br) para acessar o site da ESAB, aquele aponta no IP do servidor web (www), onde o site está acessível. Cada nome a esquerda do domínio pode representar uma máquina, ou um subdomínio. Um exemplo de subdomínio é [intranet.esab.edu.br](http://intranet.esab.edu.br), dentro deste domínio podem estar servidores que só são acessados de dentro da rede local da ESAB.

Para prosseguirmos com nossos estudos apresentaremos definições de dois elementos do DNS da RFC1034.

### Elementos do DNS

Segundo a RFC1034 o DNS é composto de **NAME SERVERS** (Servidores de Nome) e **RESOLVERS** (Resolvedores).

Os **NAME SERVERS** são programas servidores que detêm as informações sobre a estrutura da árvore de domínios. Um servidor de nomes pode fazer cache da estrutura ou definir informações sobre qualquer parte da árvore de domínios, mas em geral um servidor de nomes particular tem a informação completa do subconjunto do seu espaço de domínio e ponteiros para outros servidores de nomes que podem ser utilizados para

conseguir informações de qualquer parte da árvore de domínios. Os tipos de servidores são:

- **Servidores Recursivos:** Consultam servidores Autoritativos
- **Servidores Autoritativos:** Respondem requisições com o endereço IP (caso conheça), com uma referência para outro servidor, ou com uma negação (caso não tenha informações sobre o nome consultado).

Visando alta disponibilidade do sistema de tradução de nomes, é recomendado que se tenha sempre mais que um servidor DNS configurado respondendo para as mesmas zonas. Para isto servidores recursivos e autoritativos podem ser configurados como servidores primários ou secundários.

- **Servidor Primário:** Servidor principal. Pode ser tanto recursivo quanto autoritativo.
- **Servidor Secundário:** Atua como backup do servidor Primário em casos de falhas.

Um servidor que conhece todas as informações de parte da árvore de domínios é a **autoridade** para esse espaço de nomes. Informações autoritativas são organizadas em unidades chamadas **zonas**. Estas zonas podem ser automaticamente distribuídas para outros servidores a fim de prover redundância para os dados das zonas. Para descrever as características das zonas (ou domínios) os servidores DNS possuem os DNS Resource Records (RRs, ou em português Registro de Recursos). Os RRs têm um formato binário, que é usado para requisições e respostas destas, e um formato de texto usado nos arquivos das zonas.

Exemplos de RRs:

|               |   |
|---------------|---|
| <b>SOA:</b>   | Indica onde começa a autoridade da zona |
| <b>NS:</b>    | Indica um servidor de nomes para a zona |
| <b>A:</b>     | Mapeamento de nome para endereço ipv4   |
| <b>AAAA:</b>  | Mapeamento de nome para endereço ipv6   |
| <b>MX:</b>    | Indica um mail exchanger para um nome   |
| <b>CNAME:</b> | Mapeia um nome alternativo a outro nome |

Um exemplo de uso destas RRs para registro de um nome na configuração de uma zona em um servidor DNS é:

|        |    |       |              |
|--------|----|-------|--------------|
| www    | IN | A     | 192.168.1.99 |
| portal | IN | CNAME | www          |

No exemplo acima na primeira linha utilizou-se o RR “A” para mapear o nome www dentro de uma zona para o endereço ipv4 192.168.1.99. Em seguida atribuiu-se um nome alternativo (como se fosse um apelido) para o mesmo IP via o RR “CNAME” de forma que tanto o nome www ou o nome portal estão mapeados para o mesmo IP. Na prática, isto significa uma configuração semelhante a esta fosse aplicada na zona do domínio esab.edu.br, tanto o nome www.esab.edu.br, quanto o nome portal.esab.edu.br seriam traduzidos para o mesmo ipv4.

Resumidamente, as zonas de um servidor autoritativo contêm todas as informações para tradução de IP – NOMES, para determinada área de domínios.

Nos servidores de nomes, cada nome de domínio possui um registro de zona de autoridade. Este contém informações do domínio e da zona em que o domínio está inserido, tais como, nome do servidor de nomes primário, e-mail do responsável pelo domínio, número de série da zona, entre outros.

Os *Resolvers* (ou, resolvedores), são programas que extraem as informações de um servidor de nomes em resposta a uma requisição do cliente. Mas o resolvedor não existe como um processo distinto executado no computador. Ele é uma biblioteca de rotinas ligada a qualquer aplicação que deseja traduzir endereços. Por exemplo, o sistema operacional possui um resolvedor que é consultado pelos navegadores toda vez que o usuário tenta acessar um site. Assim, não é necessário nenhum protocolo entre o resolvedor e um programa utilizado pelo usuário.

# UNIDADE 13

*Objetivo: Entender a utilidade e funcionamento do serviço de tradução de “nome-para-endereço ip”, o DNS.*

## DNS (Continuação)

### Funcionamento

O DNS é um serviço de tradução nomes que funciona utilizando os protocolos de transporte TCP e UDP na porta padrão 53. Dentro da especificação do serviço cada tipo de protocolo de transporte é utilizado de acordo com o tipo de mensagem a ser transmitido. O protocolo UDP não é aceitável para transferência de informações para sincronização de zonas, pois estas devem ser transmitidas entre dois servidores com garantia de integridade, ficando estas a cargo do protocolo TCP, mas é recomendável para requisições padrão de tradução de nomes na Internet. Requisições UDP podem ser perdidas na rede, então há estratégias de retransmissão destas pelos *resolvers*.

O mapeamento de nomes - IPs pode ser feito de forma direta ou reversa. O **mapeamento direto** ocorre da seguinte maneira. O computador do usuário encaminha uma requisição de tradução de nome ao servidor DNS local da sua rede. Caso este servidor saiba a tradução, ou possua cache desta, ele responde a requisição com o IP relacionado ao nome perguntado. Caso o servidor local não saiba, ele perguntará aos servidores DNS do domínio começando da direita para esquerda. Voltando ao nosso exemplo: um aluno quer acessar de casa o site da ESAB, [www.esab.edu.br](http://www.esab.edu.br). Então o servidor perguntará ao servidor raiz, quem é **www.esab.edu.br**? E este servidor retornaria uma referência ao IP dos servidores **br**. O servidor local prosseguirá perguntando a um dos servidores **br** quem é [www.esab.edu.br](http://www.esab.edu.br)? Esse, por sua vez, prosseguirá respondendo qual é o próximo servidor que contém as informações sobre o domínio em questão. No Brasil são os servidores **br** os responsáveis pelos subdomínios **.com.br**, **.edu.br**, e assim por diante. O processo de pesquisa acaba quando a consulta (*query*) é respondida com uma referência ao servidor que responde pelo domínio [esab.edu.br](http://esab.edu.br). Este servidor apontará, por sua vez

que IP da máquina www é o 201.76.48.129. Este tipo de consulta é chamada de **consulta recursiva**, mostrada na Figura 32.

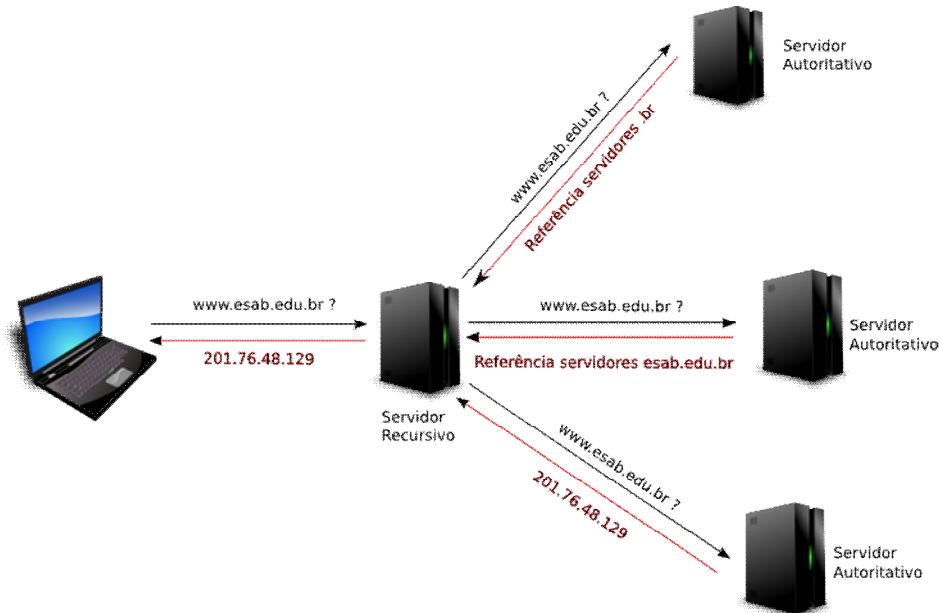


Figura 32 - Consulta Recursiva

O **mapeamento reverso** foi posteriormente definido na RFC 1912 (<http://tools.ietf.org/html/rfc1912>), pois não era o objetivo inicial do serviço de tradução de nomes. É utilizado quando se deseja fazer a tradução IP – nome. O servidor DNS local consulta um servidor DNS raiz e em seguida o DNS da rede de destino. A requisição é então respondida ao computador que a realizou. A tradução nome – IP nós já entendemos sua necessidade, mas qual é a função do mapeamento reverso? Esse tipo de resolução é muito importante para que os servidores de e-mail entreguem mensagens a outros domínios. A conferência do domínio explicitado no remetente do e-mail com o IP registrado do servidor de correio eletrônico registrado no DNS é realizada para combater o famoso *spam*, uma vez que estas mensagens são enviadas através de servidores de e-mail com falhas de segurança e utilizando como remetentes fora do domínio destes. Também, existem servidores FTP que fazem a verificação de reverso antes de permitir acesso a seus arquivos.

## CACHE DNS

Para diminuir o tempo de resolução dos nomes e o tráfego DNS, pode ser utilizado um *cache* DNS. O funcionamento do serviço continua da mesma forma, porém a resposta do registro pode ser dada por um servidor que não tenha a zona de autoridade. Seguindo o mesmo exemplo da consulta pelo nome **www.esab.edu.br**, vamos partir do princípio que os servidores agora armazenam *cache*. A consulta “quem é **www.esab.edu.br?**” será feita primeiramente num servidor raiz. O raiz examinaria os registros armazenados em *cache* e somente no caso em que não localize o IP ele indica as referências dos servidores **br**. Em seguida, a consulta passa para o servidor **br** “quem é **www.esab.edu.br?**”. Caso este nome tenha sido acessado recentemente, ele estará no *cache* do servidor e este retornará “**www.esab.edu.br** é o IP 201.76.48.129” sem dar as informações do próximo servidor da hierarquia.

Este processo agiliza a consulta do ponto de vista que menos servidores serão consultados, mas faz com que haja um tempo entre a alteração de um registro e a visibilidade desta alteração em toda a Internet. Isto ocorre visto que um servidor hierarquicamente superior ao que sofreu alteração de um registro responderá as consultas de acordo com os registros de seu *cache* até que estes registros expirem e sejam atualizados. Para evitar problemas com as alterações, o registro no *cache* expira em um curto período (normalmente dois dias).

## DNSSEC

Como vimos o DNS tem com funcionalidade de realizar traduções nomes – IPs e IP – nomes, tornando mais amigável aos humanos a localização de endereços na rede. Utiliza uma estrutura hierárquica de árvore para realizar consultas. Na criação do DNS, não foram analisados diversos aspectos de segurança que o protocolo poderia trazer, e por isso foi criado uma extensão, denominada DNSSEC.

O DNSSEC (*Domain Name System Security Extensions*), definido na RFC 2065 (<http://tools.ietf.org/html/rfc2065>), adiciona um sistema de resolução de nomes mais seguro, solucionando problemas encontrados na atual tecnologia DNS. A segurança em servidores DNS é de alta importância, pois sem ela falsas informações de tradução de nomes podem criar oportunidades para roubo de informações de terceiros ou alteração de

dados em diversos tipos de transações, como compras eletrônicas. Um atacante poderia, por exemplo, responder com uma informação de redirecionamento para uma falsa página de cadastro de cartão de crédito em uma loja virtual a uma pessoa que acessa o site através de consulta recursiva. O cadastro de todos os dados de seu cartão estaria sob a posse de uma pessoa não confiável, que poderia fazer o uso indevido dele.

Na tecnologia DNS, um ataque deste tipo é extremamente difícil de ser detectado, e na prática, de prevenção impossível. O objetivo da extensão é assegurar o conteúdo do DNS, por meio de criptografia, e impedir estes ataques, garantindo assim a origem das informações. Como em outros serviços com exigências de privacidade que utilizam criptografia, a extensão garante: autenticidade, integridade, e a não existência de um nome. Porém, não garante confidencialidade ou proteção contra ataques de negação de serviço (DoS).

A confidencialidade não é garantida, pois o mecanismo utilizado pelo DNSSEC é baseado na tecnologia de criptografia com chaves assimétricas apenas para assegurar a autenticidade do servidor. Isso significa que o servidor DNS com suporte a DNSSEC possui um par de chaves eletrônicas, uma chave privada para encriptar suas respostas a consultas e uma chave pública divulgada na Internet para que os servidores que realizem consultas possam decriptar as respostas e garantir que elas são do servidor correto. O administrador da zona usa a chave privada para assinar digitalmente sua própria zona no DNS, e todos com acesso a chave pública desta zona podem verificar que os dados transferidos desta zona estão intactos e são autênticos. Porém as chaves nunca expiram, e por isto deve haver políticas locais (em cada zona) para a troca e atualização de chaves com frequência determinada, como por exemplo, de três em três meses. Assim todos os servidores que assinam determinada zona devem obter a nova chave de período em período.

A partir do dia 16/07/2010, a *zona root* (primeiros servidores a serem consultados em uma resolução de nome) passou a fornecer suporte a DNSSEC, disponibilizando uma chave pública para que os servidores recursivos possam se autenticar. Dessa forma, a autenticação pode ser feita desde o início da consulta. Porém, é altamente recomendado que os servidores locais estejam sempre atualizados de forma a conter todos os módulos da extensão.

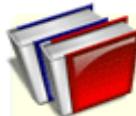
No DNSSEC foram adicionados novos RRs para descrever as zonas seguras.



## Estudo Complementar

Consulte: “Falha de segurança em servidores de DNS preocupam” disponível neste blog  
<http://www.ovelho.com/node/4801>.

Em seguida acesse: “OpenDNS - Navegação melhor e mais segura” disponível em  
<http://www.ovelho.com/node/4797>.



## Estudo Complementar

O registro.br, Registro de Domínios para a Internet do Brasil, vem incentivando o uso de DNSSEC no Brasil. Acesse o tutorial desta extensão publicado por eles.

<ftp://ftp.registro.br/pub/doc/tutorial-dnssec.pdf>





## Atividades

### Tarefa Dissertativa

Acesse a sua sala de aula para realizar a seguinte tarefa dissertativa:

Pesquise em livros e sites e faça uma síntese com suas próprias palavras sobre ataques DoS (*Denial of Service*).



# UNIDADE 14

*Objetivo: Apresentar, utilizando o Wireshark, as mensagens trocadas entre cliente e servidor DNS.*

## **Analizador de Pacotes de Rede – Consulta DNS**

### **Introdução**

Nesta unidade apresentaremos os pacotes capturados durante uma consulta de DNS. Utilizaremos como exemplo o nome: www.esab.edu.br. Verificaremos que a consulta do *Resolver* é feita através do protocolo de transporte UDP, utilizando a porta 53 no servidor.

Nossa abordagem prática consiste em acessar os nomes através do comando de terminal linux “dig +trace <nome consultado>”, utilizando o Wireshark para capturar pacotes na máquina cliente e não no servidor DNS. Como resposta ele nos dará toda a sequência de servidores consultadas durante a resolução de nomes. Note que com o comando “dig +trace” a máquina usuária fará a consulta iterativamente em todos os servidores DNS da hierarquia a fim de saber todos os RRs destes servidores. Numa consulta padrão, como a utilizada por um navegador de Internet, a máquina cliente só troca pacotes com servidor recursivo, configurado como DNS Primário. Só conseguiríamos capturar os pacotes de uma consulta padrão instalando o Wireshark em um servidor DNS.

A descrição da rede:

Rede Local: 10.255.255.0/24

Servidor DNS primário: 8.8.8.8

Máquina cliente, Resolver: 10.255.255.100

Para resolução do primeiro nome, mostrado na Figura 33, o *Resolver* pergunta ao servidor primário, 8.8.8.8, quem é www.esab.edu.br? O servidor primário consulta os RRs do servidor raiz que faz referência a todos os nomes dos servidores raiz. Estes nomes são do tipo a.root-servers.net.

Escolhendo um destes servidores, no caso o b.root-server.net. de IP 192.228.79.201, o servidor primário repete a pergunta. Quem é www.esab.edu.br? Este servidor aponta os RRAs dos servidores .br, que são do tipo a.dns.br.

```

File Edit View Terminal Help
salim@salim-note:~$ dig +trace www.esab.edu.br

; <>> DiG 9.7.0-P1 <>> +trace www.esab.edu.br
;; global options: +cmd
.
        4886      IN      NS      a.root-servers.net.
.
        4886      IN      NS      b.root-servers.net.
.
        4886      IN      NS      c.root-servers.net.
.
        4886      IN      NS      d.root-servers.net.
.
        4886      IN      NS      e.root-servers.net.
.
        4886      IN      NS      f.root-servers.net.
.
        4886      IN      NS      g.root-servers.net.
.
        4886      IN      NS      h.root-servers.net.
.
        4886      IN      NS      i.root-servers.net.
.
        4886      IN      NS      j.root-servers.net.
.
        4886      IN      NS      k.root-servers.net.
.
        4886      IN      NS      l.root-servers.net.
.
        4886      IN      NS      m.root-servers.net.

;; Received 228 bytes from 8.8.8.8#53(8.8.8.8) in 27 ms

br.          172800   IN      NS      d.dns.br.
br.          172800   IN      NS      b.dns.br.
br.          172800   IN      NS      e.dns.br.
br.          172800   IN      NS      f.dns.br.
br.          172800   IN      NS      a.dns.br.
br.          172800   IN      NS      c.dns.br.

;; Received 285 bytes from 192.228.79.201#53(b.root-servers.net) in 331 ms

esab.edu.br.     86400   IN      NS      ns1.locaweb.com.br.
esab.edu.br.     86400   IN      NS      ns2.locaweb.com.br.
esab.edu.br.     86400   IN      NS      ns3.locaweb.com.br.

;; Received 99 bytes from 200.219.159.10#53(f.dns.br) in 19 ms

www.esab.edu.br. 3600     IN      CNAME   esab.edu.br.
esab.edu.br.      3600     IN      A       201.76.48.129

;; Received 63 bytes from 201.76.40.2#53(ns2.locaweb.com.br) in 22 ms

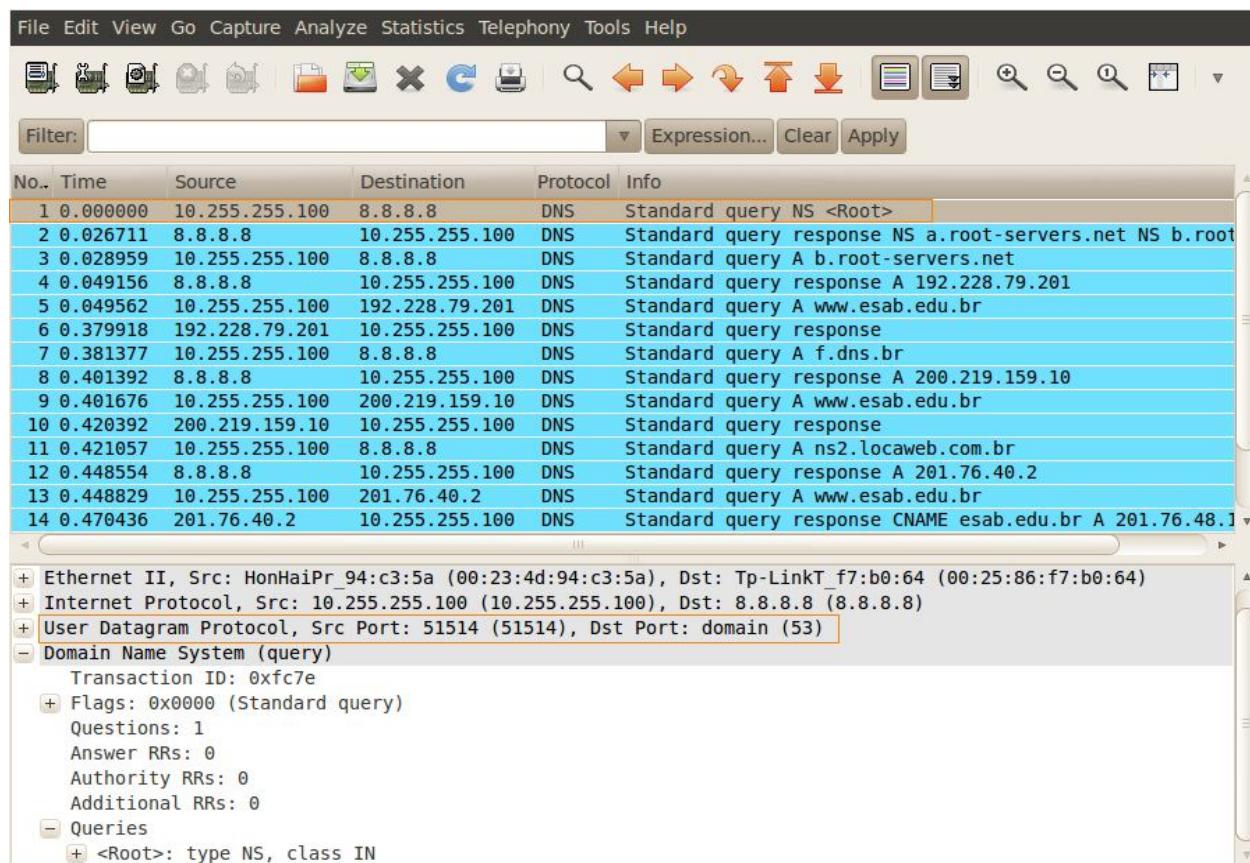
```

Figura 33 - Servidores DNS consultados durante a resolução do nome www.esab.edu.br

A sequência do processo é dada quando o servidor primário pergunta ao servidor f.dns.br, IP 200.219.159.10, quem é www.esab.edu.br? Agora este servidor responde que os servidores autoritativos do domínio esab.edu.br são os nsx.locaweb.com.br, informando

os RRs deles. Por último, o servidor primário pergunta ao servidor ns2.locaweb.com.br, IP 201.76.40.2, quem é www.esab.edu.br? Este, por sua vez, responde que www.esab.edu.br é um CNAME para esab.edu.br que tem IP 201.76.48.129. Vale perceber que existe mais de um servidor por nível, aumentando a disponibilidade do serviço de DNS.

De posse desta informação as aplicações podem chegar até a máquina web que responde pelo site da ESAB naquele IP. Os pacotes capturados na máquina que originou a consulta são apresentados na Figura 34. Nela pode-se perceber claramente no quadro inferior em destaque, o uso do protocolo UDP com porta de destino, apontando para servidor 8.8.8.8, número 53. Também é possível ver toda a sequência de perguntas (*queries*) realizadas aos servidores autoritativos, pacotes azuis. É evidenciado o pacote número 1, cinza, que é a primeira query realizada para resolução do nome, que tem IP de destino no servidor primário, 8.8.8.8.



**Figura 34 – Pacotes capturados durante a consulta do nome www.esab.edu.br**

Em seguida são apresentados na Figura 35 os RRs dos servidores raiz. E na Figura 36 o padrão da query (requisição) enviada ao servidor b.root-server.net. Nela, o quadro em destaque mostra o objetivo é conseguir o RR A (IP) do nome www.esab.edu.br.

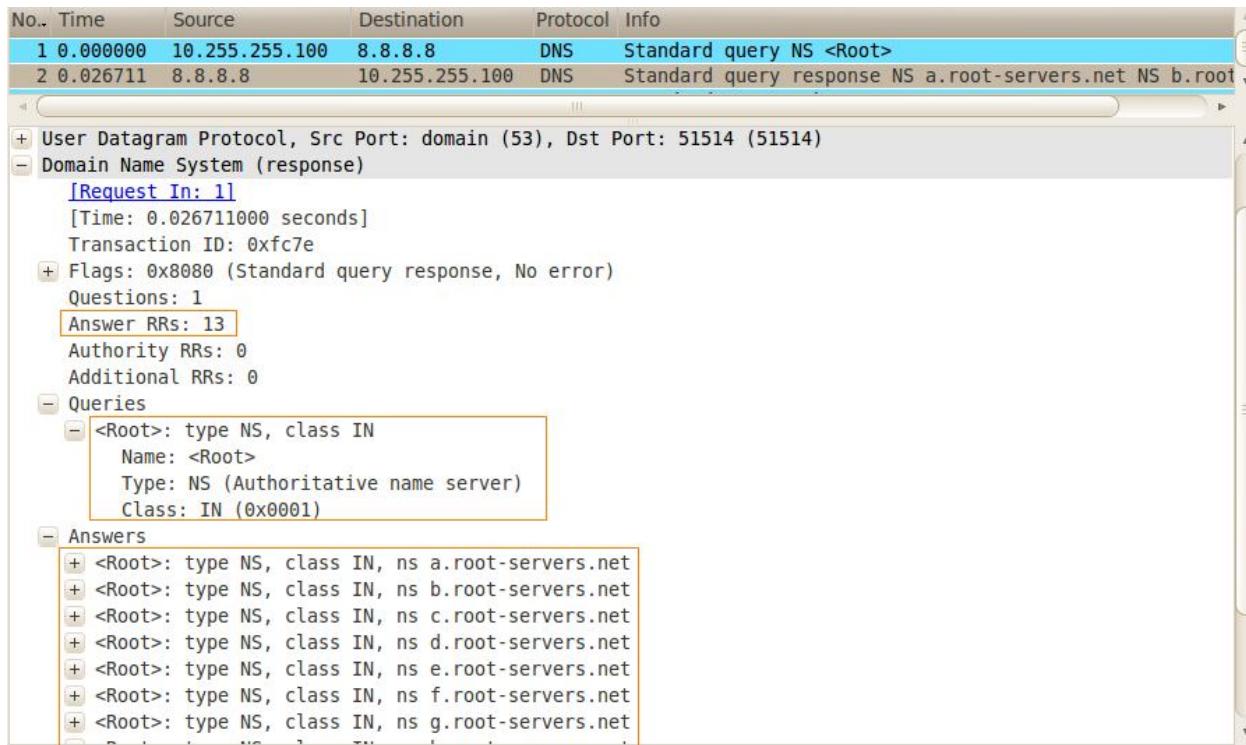


Figura 35 – RRs dos servidores raiz

Por fim, na Figura 37, é mostrada a resposta dada pelo servidor 201.76.40.2, ns2.locaweb.com.br, para o nome www.esab.edu.br, apontando que este nome é um CNAME para o nome esab.edu.br, e tem IP 201.76.48.129.

Nesta unidade acompanhamos todas as *queries* e suas respostas realizadas para obtenção do IP responsável pelo nome www.esab.edu.br pelo comando Linux `dig +trace`, mostrando os pacotes e formato das mensagens envolvidos capturados no Wireshark. Estes passos são a descrição de um procedimento para verificar se a resolução de nomes de um domínio particular está correta, e caso falhe, em qual servidor DNS ela parou. Com esta informação, o administrador pode reportar ao registro do servidor correto para resolução do problema.

| No. | Time     | Source         | Destination    | Protocol | Info                                     |
|-----|----------|----------------|----------------|----------|--|
| 3   | 0.028959 | 10.255.255.100 | 8.8.8.8        | DNS      | Standard query A b.root-servers.net      |
| 4   | 0.049156 | 8.8.8.8        | 10.255.255.100 | DNS      | Standard query response A 192.228.79.201 |
| 5   | 0.049562 | 10.255.255.100 | 192.228.79.201 | DNS      | Standard query A www.esab.edu.br         |
| 6   | 0.379918 | 192.228.79.201 | 10.255.255.100 | DNS      | Standard query response                  |
| 7   | 0.381377 | 10.255.255.100 | 8.8.8.8        | DNS      | Standard query A f.dns.br                |
| 8   | 0.401392 | 8.8.8.8        | 10.255.255.100 | DNS      | Standard query response A 200.219.159.10 |

+ Frame 5 (75 bytes on wire, 75 bytes captured)  
 + Ethernet II, Src: HonHaiPr\_94:c3:5a (00:23:4d:94:c3:5a), Dst: Tp-LinkT\_f7:b0:64 (00:25:86:f7:b0:64)  
 + Internet Protocol, Src: 10.255.255.100 (10.255.255.100), Dst: 192.228.79.201 (192.228.79.201)  
 + User Datagram Protocol, Src Port: 54926 (54926), Dst Port: domain (53)  
 - Domain Name System (query)  
 [Response In: 6]  
 Transaction ID: 0x27f6  
 + Flags: 0x0000 (Standard query)  
 Questions: 1  
 Answer RRs: 0  
 Authority RRs: 0  
 Additional RRs: 0  
 - Queries  
 - www.esab.edu.br: type A, class IN  
 Name: www.esab.edu.br  
 Type: A (Host address)  
 Class: IN (0x0001)

Figura 36 - Query realizada ao servidor raiz b.root-server.net.

| No. | Time     | Source         | Destination    | Protocol | Info  |
|-----|----------|----------------|----------------|----------|---|
| 12  | 0.448554 | 8.8.8.8        | 10.255.255.100 | DNS      | Standard query response A 201.76.40.2                     |
| 13  | 0.448829 | 10.255.255.100 | 201.76.40.2    | DNS      | Standard query A www.esab.edu.br                          |
| 14  | 0.470436 | 201.76.40.2    | 10.255.255.100 | DNS      | Standard query response CNAME esab.edu.br A 201.76.48.129 |

+ www.esab.edu.br: type CNAME, class IN, cname esab.edu.br  
 + esab.edu.br: type A, class IN, addr 201.76.48.129

Figura 37 - Resposta dada ao Resolver apontando o IP do nome www.esab.edu.br

# UNIDADE 15

*Objetivo: Apresentar a arquitetura do serviço de correio eletrônico descrevendo os principais protocolos envolvidos.*

## Correio Eletrônico

### Introdução

O correio eletrônico, popularizado com o nome *e-mail*, é um dos serviços mais usados na Internet. Permite os diversos usuários conectados através da Internet troarem mensagens e arquivos de diversos tipos, tais como documentos, fotos, músicas, etc.

Porém, se engana quem pensa que este é um serviço novo. Mesmo antes da popularização da Internet, o correio eletrônico já era utilizado. Surgiu em meados dos anos 60, e originalmente foi criado para troca de mensagens entre usuários de computadores de grande porte e logo foi adaptado para troca de mensagens entre terminais remotos. Em 1969, com o início da ARPANet (Advanced Research and Projects Agency Network – Rede da Agência de Pesquisas em Projetos Avançados), surgiu a utilização do símbolo “@” como separador entre nome de usuário e domínio. A partir de sua implantação pelo governo dos Estados Unidos, pesquisadores em universidades e outros locais passaram a trocar dados eletronicamente uns com os outros. A popularização do serviço foi ocorrendo de acordo com que ocorria a popularização dos computadores e da Internet.

Em 1982, as propostas relativas a correio eletrônico da ARPANet foram publicadas como a RFC 821 (<http://tools.ietf.org/html/rfc821>, define o protocolo de transmissão SMTP) e a RFC 822 (<http://tools.ietf.org/html/rfc822>, define o formato de mensagens). Revisões menores, publicadas nas RFCs 2821 e 2822, se tornaram padrões da Internet, mas todas ainda se referem ao correio eletrônico da Internet como a RFC 822.

## Arquitetura

A arquitetura dos correios eletrônicos é dividida em dois tipos de programas: o de transporte e o de usuário. O primeiro cuida do transporte das mensagens já escritas entre origem e destino. A aplicação que cuida do transporte destas mensagens é chamada de Agente de Transferência de Mensagens (MTA, do inglês *Mail Transfer Agent*). Esta aplicação (*daemon*) executa numa máquina que será o servidor de correio eletrônico e deve possuir alta disponibilidade, uma vez que tem que estar apta a receber mensagens de outros servidores o tempo todo. Como exemplo deste tipo de aplicações podemos citar o Sendmail, Postfix, Qmail e o MSExchange.

O segundo é responsável por prover ao usuário os meios de ler e escrever suas mensagens, são os Agentes de Mensagem de Usuário, (MUA, *Mail User Agent*). Podem tanto ser programas executados no computador do usuário, como o Eudora, Evolution e o Outlook, ou programas disponíveis através de interfaces web, os *webmails*, como os utilizados pelo Gmail, Hotmail que possuem desenvolvimento proprietário e não disponível. Há programas que possibilitam uma empresa possuir um webmail próprio como o SquirrelMail, NeoMail e o Webmial.

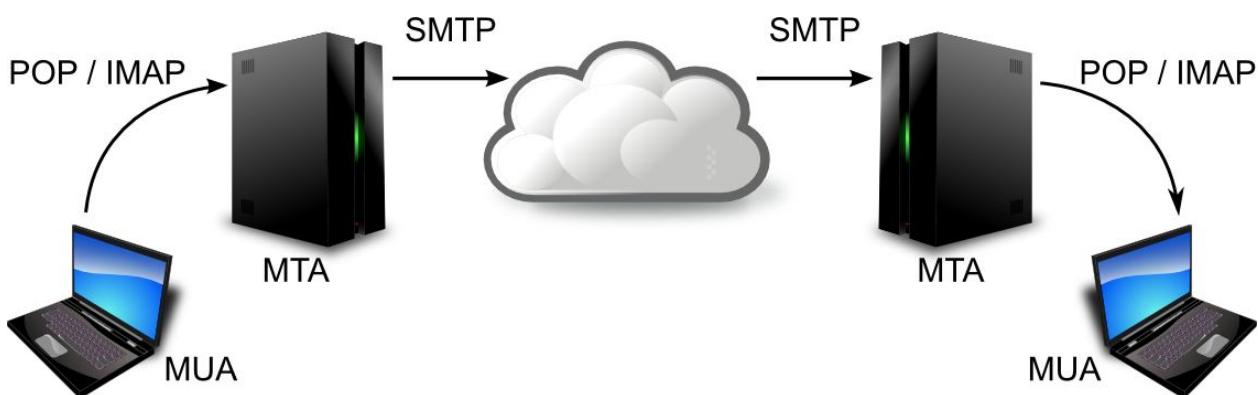


Figura 38 – Arquitetura de conexão dos agentes SMTP e possíveis protocolos envolvidos

Estes dois tipos de agente utilizam diferentes protocolos para envio e recebimento de mensagens. Os MTAs utilizam o protocolo SMTP definido pelas RFCs 821 e 822 para a troca de mensagens entre os servidores de e-mail, e os MUAs podem utilizar os protocolos IMAP (*Internet Message Access Protocol*) e POP (*Post Office Protocol*), definidos nas RFCs 1064 (<http://tools.ietf.org/html/rfc1064>) e 1081

(<http://tools.ietf.org/html/rfc1081>) respectivamente, mas há revisões para cada versão destes protocolos. Há outros tipos de protocolos proprietários para MUAs como o Microsoft Exchange e Lotus Notes/Domino.

## **SMTP**

O SMTP (*Simple Mail Transfer Protocol*) é um protocolo padrão Internet para envio de e-mails entre redes. Foi inicialmente definido pela RFC 821 e sua última revisão foi publicada em 2008 na RCP 5321 (<http://tools.ietf.org/html/rfc5321>).

Este protocolo atua puramente na camada de aplicação e é independente do subsistema de transmissão. Foi projetado para transferir mensagens de maneira confiável e eficiente. Note que as mensagens trocadas não são confidenciais uma vez que não são criptografadas, isto se dá apenas com a inserção de uma camada SSL/TLS que pode ser configurada pelo administrador. Em sua especificação original, o SMTP utiliza a porta 25 do protocolo de transporte TCP quando a comunicação entre dois servidores (MTAs) é baseada em redes de longo alcance. Em novas submissões tem sido utilizada a porta TCP 587. Apesar de não usual, existe a possibilidade de se utilizar um canal com um mecanismo qualquer para a comunicação entre processos.

Seu funcionamento é orientado à conexão. O resultado de uma requisição de um cliente (MUA) é o estabelecimento de uma conexão entre o servidor SMTP de origem e um servidor SMTP de destino, que pode ser tanto o de destino final, quanto um servidor intermediário. Cada parte da mensagem eletrônica só é enviada após uma negociação inicial a respeito de quem é o emissor original, e de quem receberá a mensagem em questão.

Quando um servidor SMTP aceita uma correspondência eletrônica, assume a responsabilidade de entregar a mensagem ao usuário final, caso ele seja local, ou repassá-la caso o destinatário seja um usuário remoto. Caso ocorra falha na entrega, este servidor deve notificar o emissor da mensagem.

Há várias mensagens de controle dentro do protocolo SMTP. Para envio de uma mensagem, após uma conexão ser estabelecida, o servidor SMTP de origem envia um comando MAIL indicando a origem da mensagem e caso a mensagem seja aceita, o servidor de destino envia uma mensagem OK (código 250). O próximo passo é o envio do

nome do destinatário da mensagem ao servidor de destino. Caso o usuário exista, novamente a mensagem OK é enviada. Porém, se o usuário não existir uma mensagem de ERRO (Código 550) é enviada ao servidor de origem. Para uma única mensagem, múltiplos usuários podem ser aceitos.

Para início da transmissão da correspondência eletrônica o servidor de origem envia o comando DATA. E em seguida, informa a mensagem SMTP de código 354, que indica o início da transferência da correspondência e que no fim desta será indicado com uma linha constando somente um ponto, "<CR><LF>.<CR><LF>". Após o recebimento dos dados, o servidor de destino confirma com uma mensagem OK (Código 250).

O formato padrão das correspondências eletrônicas, *e-mails*, é definido na RFC822. Em qualquer sistema, estas mensagens são divididas em duas partes: cabeçalho e corpo. As duas partes são separadas por uma única linha em branco.

Dentro da especificação, há o formato para a composição do cabeçalho de *e-mails*, assim como a semântica para cada um de seus campos. Algumas palavras-chaves são obrigatórias, outras não. Uma palavra-chave deve ser seguida de dois-pontos, ":", e de um valor para este campo do cabeçalho. Um exemplo de mensagem é apresentado abaixo:

```
From: secretario@esab.edu.br
To: aluno@<domínio>
Cc: supervisor@esab.edu.br
      professor@esab.edu.br
Subject: Atualização de dados cadastrais
Date: Mon, 14 December 2010 14:12:00

Prezado aluno,
Foram encontradas inconsistências no seu cadastro.
Por favor, atualize seu cadastro no site da ESAB
(www.esab.edu.br)

Atenciosamente,
Secretario
```

**Figura 39 - Formato padrão de correspondência eletrônica, RFC822**

## SMTP/MIME

Na primeira versão do protocolo SMTP eram suportados apenas 7-bit de caracteres ASCII, limitando-se, assim, ao uso de apenas caracteres de língua inglesa para composição das mensagens de e-mails. Extensões foram criadas a fim de resolver esta limitação. Estas extensões do protocolo recebem o nome de MIME (*Multipurpose Internet Mail Extensions*, ou em português, Extensões Multifunção para Mensagens de Internet). As MIME são uma norma da Internet para o formato das mensagens de correio eletrônico e, atualmente, grande parte das mensagens de correio eletrônico utilizam o protocolo SMTP e o formato de mensagens MIME.

Estas extensões também permitem transferir dados em codificação diferentes do ASCII. É a partir delas que surge a possibilidade de envio de arquivos binários contendo imagens, sons, filmes e programas de computador.

Outra grande importância do formato das mensagens SMTP/MIME é que este foi fundamental para o desenvolvimento dos *webmails* tão populares hoje em dia. São as extensões MIME que possibilitam que aplicações HTTP possam trocar tráfego com servidores de e-mail.

# UNIDADE 16

*Objetivo: Apresentar a arquitetura do serviço de correio eletrônico descrevendo os principais protocolos envolvidos.*

## Correio Eletrônico (Continuação)

Até agora vimos a arquitetura do correio eletrônico e os conceitos das trocas de mensagem SMTP. Porém, para que as mensagens pudessem ser escritas e lidas pelos usuários é necessária a existência de uma aplicação conectada ao servidor. Denominamos estas aplicações de MUA.

Hoje, comumente acessamos nossas contas de e-mail via webmails, que utilizam as extensões do protocolo SMTP, MIME, para que as aplicações HTTP possam se comunicar com os servidores de e-mail aos quais chamamos de MTA. No entanto, também podemos acessar nossas mensagens através de outras aplicações gerenciadoras de e-mail como, por exemplo, o Microsoft Outlook e o Thunderbird. A partir destas aplicações é possível baixar as mensagens para uma estação de trabalho. Após o download das mensagens é possível trabalhar escrevendo suas respostas mesmo sem estar conectado na Internet e só depois transmiti-las pela Internet.

Para que os gerenciadores de e-mail possam acessar e enviar mensagens aos servidores SMTP foram desenvolvidos dois protocolos aos quais trataremos nesta Unidade: o POP e o IMAP.

### POP3

O principal motivador do surgimento do protocolo POP (*Post Office Protocol*) foi prover um mecanismo para que estações que não estavam conectadas permanentemente a uma rede IP pudessem receber e tratar suas mensagens. Com este protocolo, as estações podem buscar suas mensagens armazenadas temporariamente em servidores de correio eletrônico.

A primeira versão do protocolo POP foi publicada na RFC 918. A versão atual, POP3, foi publicada na RFC1081 que já sofreu algumas revisões e atualizações para atualização do mecanismo de autenticação.

O serviço POP funciona sob o modelo cliente/servidor. O cliente é a entidade que faz uso deste serviço, e o servidor o disponibiliza para uso. Seu funcionamento se dá da seguinte maneira: o servidor POP em execução escuta a porta TCP 110 e quando um cliente deseja utilizar o serviço, este inicia uma conexão com o servidor que envia uma saudação. Em seguida as requisições (comandos) são trocadas e respondidas até que se encerre esta conexão.

As requisições são realizadas através de palavras-chaves seguidos de um ou mais argumentos. Já as respostas são compostas de um indicador de status que podem ser “+OK” quando positivas e “-ERR” quando negativas, uma palavra-chave e de informações adicionais. As respostas podem conter mais de uma linha que são terminadas com a sequência <CR><LF> e o fim de uma resposta é indicado com o código do caractere “.” seguido de um par de <CR><LF>.

Os principais comandos POP3 utilizados são:

USER *name* – identifica o usuário

PASS *string* – senha do usuário

QUIT – finaliza a sessão POP

APOP – identifica a caixa de correio e a *string* MD5 de autorização

STAT – lista o tamanho das mensagens

RETR *msg* – solicita o envio da mensagem número “*msg*”

RSET – desmarca a mensagem

DELE *msg* – marca uma mensagem para remoção

LIST – Lista as mensagens disponíveis numa caixa de correio

+OK – resposta positiva a comandos

-ERR – resposta negativa a comandos

## IMAPv4

O protocolo IMAP, assim como o POP, tem o objetivo de manipular mensagens entre o servidor de correio eletrônico (*e-mail*) e o cliente. O IMAP é mais recente que o POP e

possui mais funções que este protocolo. A última versão do protocolo IMAP, versão 4, foi publicada na RFC1730, e revisada pela RFC2060.

O POP3 foi projetado para manipular as mensagens de forma *off-line* a partir de um único cliente. Deste modo, após efetuar o download da mensagem esta é removida do servidor. Já o IMAP, possui a opção de leitura das mensagens sem que estas sejam necessariamente removidas. Isto traz a possibilidade de leitura de mensagens a partir de múltiplos MUAs instalados. Por exemplo, um usuário pode configurar um agente em um computador em seu trabalho e outro em sua residência.

Uma das principais vantagens do uso do protocolo IMAP ao invés do POP3 é o modo de operação conectado ou desconectado. Os clientes que utilizam POP se conectam ao servidor para fazer download das mensagens e logo após esta conexão é fechada. Isto deve ser feito de tempos em tempos a fim de que se verifique a existência de novas mensagens. No IMAP o cliente fica conectado ao servidor constantemente e baixa novas mensagens sob demanda. Isto proporciona um tempo de resposta mais rápido, principalmente para clientes que recebem muitas mensagens.

Outras vantagens são os recursos para manipulação de mensagens no servidor, tais como: possibilidade de criar, remover e renomear pastas, ativar ou remover marcadores (*flags*) e o acesso MIME *parsing*, que permite a busca e a seleção de mensagens por dados do cabeçalho, corpo ou anexos.

O protocolo IMAPv4, denominado assim em sua última versão, é orientado a conexão e utiliza a porta 143 do protocolo de transporte TCP para isto. Para qualquer possível operação, como recuperar uma mensagem, inicialmente uma conexão TCP é estabelecida. Logo após ocorre uma operação de saudação inicial (*server greeting*) entre os agentes, e em seguida a conexão muda de estado, podendo estar não autenticada (*not authenticated*), ou autenticada (*authenticated*). Se a conexão for não autenticada o cliente deve fornecer as credenciais (*login* e senha) antes do envio de qualquer comando. Já se o estado é de autenticado, basta o cliente selecionar uma caixa de mensagem antes de emitir comandos relacionados a mensagens. Após selecionar uma caixa de mensagens, o cliente muda para o estado para selecionado (*selected*). O estado *logout* é alcançado após uma operação de LOGOUT do cliente, ou por ações unilaterais do cliente ou servidor.

A fim de exemplificação, devido aos inúmeros comandos existentes dentro deste protocolo, alguns comandos IMAP são:

CAPABILITY – solicita lista de recursos suportados pelo servidor.

AUTHENTICATE – comando para iniciar autenticação (SASL) no servidor.

LOGIN – utilizado para identificar o cliente, sendo informados usuário e senha em texto plano.

STARTTLS – utilizado para iniciar autenticação em TLS.

SELECT – seleciona uma caixa de mensagem.

EXAMINE – similar ao SELECT, porém acessa a caixa de mensagens em modo de somente leitura.

SEARCH – busca por mensagens de acordo com um critério especificado.

CHECK – solicita a atualização da caixa de mensagens atual.

CLOSE – Remove as mensagens marcadas para exclusão.

### **Configurações do Servidor**

Para funcionamento adequado de um servidor de e-mail alguns itens devem ser configurados adequadamente, tais como o *hostname*, a data do sistema (*system time*) e as entradas no servidor de DNS que o identifica.

Um servidor de correio eletrônico deve possuir um nome de domínio completamente qualificado (do inglês, *Fully Qualified Domain Name* – FQDN), como por exemplo, mail.esab.edu.br. Este *hostname* é utilizado para saudação a outros servidores de e-mail que o verificam utilizando uma consulta de DNS reverso, descartando a mensagem caso a origem não seja adequadamente confirmada.

O relógio dos servidores deve ser sincronizado, pois mensagens entre servidores com diferenças muito grandes de data e hora podem ser descartadas. É recomendado o uso de servidores NTP (*Network Time Protocol*) para sincronização dos relógios automaticamente.

Outras recomendações são a configuração adequada do sistema de *logs* a fim de depuração de erros, liberação das portas de entrada e saída utilizadas pelo servidor de correio eletrônico no *firewall*. As portas a serem liberadas dependem dos recursos disponibilizados pelo servidor.

# UNIDADE 17

*Objetivo: Entender o funcionamento do protocolo de endereçamento dinâmico e sua importância.*

## DHCP (Dynamic Host Configuration Protocol)

### Introdução

Antes de discutir o funcionamento do protocolo DHCP em si, vamos relembrar alguns pontos sobre o funcionamento das redes e estudar qual a motivação para se criar este protocolo.

Como vimos anteriormente, por se tratar de uma rede global, na Internet é necessário que alguns recursos sejam controlados de forma centralizada, como é o caso dos endereços IP, para que não haja possibilidade de duplicação. A entidade que controla os números IP é o IANA (*Internet Assigned Numbers Authority*), que hoje é parte da ICANN (*Internet Corporation for Assigned Names and Numbers*). A autoridade sobre os números IP é delegada regionalmente para outras entidades. Na América Latina e Caribe a entidade responsável é o LACNIC, e, no Brasil, o NIC.br.

Conforme vimos também, a partir da década de 90 a Internet começou a se popularizar. Empresas de todo mundo investem cada vez mais em equipamentos de rede capazes de se comunicar via IP. Isso torna a administração das redes cada vez mais complicada, motivando o desenvolvimento de novos protocolos e serviços capazes de utilizar os endereços IP de forma mais eficiente. Dentre eles está o DHCP, publicado na RFC 2131 (<http://tools.ietf.org/html/rfc2131>) em 1997.

O DHCP permite a alocação dinâmica de endereços IP, implicando na possibilidade de reutilização dos endereços da Internet fornecidos pelas operadoras aos clientes em conexões não permanentes, como as realizadas através de linhas discadas ou ADSL. Junto a um estudo estatístico do número de endereços IPs usados simultaneamente em uma operadora, este protocolo viabiliza o atendimento de um maior número de clientes com um mesmo bloco. Suponha que uma empresa utilize o bloco privado classe B em

sua rede interna, por exemplo, o 172.20.0.0/16. Isto equivale a 65025 endereços que poderiam atender a mais de 65025 usuários, dado que nem todas as máquinas de uma empresa ficam ligadas ao mesmo tempo.

O protocolo também traz vantagem para os administradores de redes privadas. É difícil e oneroso manter as configurações de endereçamento manualmente, quando temos um grande número de estações de trabalho em rede. Redes com o número de estações de trabalho acima de 30 já exigem um controle mais rigoroso dos endereços utilizados, para evitar problema de duplicação de endereço de destino. Imagine que determinada máquina não consiga utilizar a rede por conta de um endereço duplicado e um técnico tivesse que visitar máquina por máquina para conferir todas as configurações. Além disto, se tornaria necessária uma política de verificação de endereços, onde o técnico faria essas visitas frequentemente, a fim de minimizar problemas. Qual seria o custo disto à medida que a rede cresce? Sem dúvida seria alto.

É importante dizer que existiram outros protocolos para configuração dinâmica de endereços IP, como o RARP e o BOOTP, porém com o aumento do número de máquinas, necessidade de mais parâmetros de configuração, e com o advento da computação móvel esses protocolos se tornaram ineficientes e isto também foi um grande motivador da criação do DHCP.

## O Protocolo

O protocolo DHCP utiliza o modelo cliente - servidor, onde o servidor responde às requisições das estações clientes com endereços de rede disponíveis e máscara de rede. Opcionalmente, o servidor DHCP pode incluir nesta resposta o endereço do roteador padrão (default gateway), e de servidores de nomes WINS e DNS, definidos nas RFCs 1001 (<http://tools.ietf.org/html/rfc1001>), 1002 (<http://tools.ietf.org/html/rfc1002>), 2929 (<http://tools.ietf.org/html/rfc2929>) e 5395 (<http://tools.ietf.org/html/rfc5395>).

Existem três métodos para alocação de endereços via DHCP:

- **Alocação dinâmica:** atribuição temporária de um endereço a um cliente. Este é o método mais utilizado e motivador do desenvolvimento desta tecnologia;

- **Alocação automática:** atribuição preferencial de um endereço a um cliente. É semelhante à alocação dinâmica, porém o servidor mantém um histórico dos IPs alocados anteriormente e preferencialmente aloca o mesmo endereço a um cliente;
- **Alocação estática ou manual:** feita pelo administrador, sendo o servidor DHCP apenas usado para transporte do endereço ao cliente. Neste caso, apenas a estação cliente com MAC (*Media Access Control*) cadastrado pelo administrador na tabela do servidor pode receber um determinado endereço.

Estes métodos de alocação podem ser usados de forma combinada ou individualmente em uma determinada rede, sendo o método de alocação dinâmica o mais empregado.

Uma estação cliente deve ser capaz de descobrir os parâmetros necessários ao seu funcionamento, inserindo-os automaticamente em seu sistema sem a intervenção manual. Do mesmo modo, o servidor deve ser capaz de funcionar de maneira automática sem a necessidade de intervenção manual de um administrador para o funcionamento de cada estação cliente, exceto nos casos em que o administrador tem que intervir no funcionamento do servidor para fixar determinado endereço a um cliente (“Alocação Estática ou Manual”).

Para isto o protocolo DHCP funciona em quatro fases: Descoberta de IP (*IP discovery*), Oferta de IP (*IP lease offer*), Requisição de IP (*IP request*), e Reconhecimento de Locação (*IP lease acknowledgement*). Todas essas fases utilizam de mensagens que seguem o formato a seguir. O formato da mensagem será importante para o estudo da Unidade 19, onde apresentaremos como se dá o processo de troca de mensagens para alocação dinâmica de IP utilizando o *WireShark*, software analisador de pacotes de rede.

|                                 |              |                |             |  |  |
|---------------------------------|--------------|----------------|-------------|--|--|
| OP(1byte)                       | HTYPE(1byte) | HLEN(1byte)    | HOPS(1byte) |  |  |
| ID DE TRANSAÇÕES (4bytes)       |              |                |             |  |  |
| SEGUNDOS (2bytes)               |              | FLAGS (2bytes) |             |  |  |
| ENDEREÇO IP DO CLIENTE (4bytes) |              |                |             |  |  |
| SEU ENDEREÇO IP (4bytes)        |              |                |             |  |  |

|   |
|---|
| ENDEREÇO IP DO SERVIDOR (4bytes)                |
| ENDEREÇO IP DO ROTEADOR (4bytes)                |
| ENDEREÇO DE HARDWARE (MAC) DO CLIENTE (16bytes) |
| NOME DO HOST DO SERVIDOR (64bytes)              |
| NOME DO ARQUIVO DE PARTIDA (128bytes)           |
| OPÇÕES (variável)                               |

**Figura 40 - Formato de uma mensagem DHCP**

**OP** - Numa mensagem DHCP, uma solicitação e uma resposta possuem os mesmos campos. O que as diferenciam é o conteúdo deste campo. A informação **um** indica uma solicitação, a informação **dois** indica uma resposta

**HTYPE** - Informa o padrão de rede utilizado pelo adaptador de rede

**HLEN** - Informa o tamanho do endereço MAC do adaptador de rede

**HOPS** - Quantidade de roteadores pelos quais a mensagem deverá passar. Parâmetro configurado no servidor para limitar o alcance das mensagens.

**ID DE TRANSAÇÕES** - Número de identificação da mensagem

**SEGUNDOS** - Quantidade de tempo em segundos desde que o cliente fez a inicialização do processo para conseguir a alocação de um endereço IP. Utilizado para limitar a espera de uma requisição.

**FLAGS** - Utilizado para "setar" opções especiais de resposta às solicitações. Este campo não é utilizado atualmente, mas ficou reservado na criação do protocolo.

**ENDEREÇO IP DO CLIENTE** - Em uma solicitação o cliente informa o seu endereço IP (possível quando o cliente conhece o seu endereço)

**SEU ENDEREÇO IP** - Utilizado pelo servidor para enviar informação de um endereço IP disponível ao cliente que solicitou.

**ENDEREÇO IP DO SERVIDOR** - Preenchido pelo cliente quando ele quer obter uma informação de um servidor específico.

**ENDEREÇO IP DO ROTEADOR** - Preenchido pelo servidor para informar ao cliente o endereço IP do roteador da rede local

**END. DE HARDWARE DO CLIENTE** - Informação do endereço MAC do cliente

**NOME DO HOST DO SERVIDOR** - Quando esses campos não são utilizados para enviar as informações pertinentes a cada um (nome do servidor e informação do sistema operacional que será inicializado no cliente) o DHCP utiliza-o remetendo informações adicionais transformando-os em campo de **OPÇÕES**, otimizando assim a utilização da mensagem.

**NOME DO ARQUIVO DE PARTIDA** - Nome do arquivo que contém a imagem de memória da(s) estação (ões) correspondente(s). A estação tem a memória de quais IPs já lhe foram atribuídos e guarda isso em arquivo. Este campo é nulo quando não há memória, ou caso contrário, contém o caminho indicando o diretório deste arquivo.

**OPÇÕES** - Esse campo é utilizado para informar que tipo de resposta ou solicitação DHCP (DHCPDISCOVER, DHCPOFFER etc.) está sendo enviada para o cliente ou para o servidor.

# UNIDADE 18

*Objetivo: Entender o funcionamento do protocolo de endereçamento dinâmico e sua importância.*

## DHCP (Continuação)

### Funcionamento

O DHCP utiliza um modelo de funcionamento cliente/servidor. Suas mensagens utilizam o protocolo de transporte UDP. As mensagens do cliente para o servidor são enviadas para a porta 67 do servidor DHCP. Já as que são originadas no servidor são enviadas para a porta 68 do cliente DHCP.

O administrador da rede pode instalar e configurar um ou mais servidores DHCP, de acordo com a necessidade. Cada servidor pode responder a requisições de uma ou mais redes. Cada rede deve ser configurada em um escopo. As informações de configuração, tais como, escopos de endereços IP a serem ofertados, reservados e outras configurações, são armazenadas no banco de dados dos servidores DHCP. O banco de dados do servidor inclui os seguintes itens:

- **Parâmetros de configuração válidos para todos os clientes na rede:** IP do Default Gateway, IP de um ou mais servidores DNS e assim por diante. Estas configurações podem ser diferentes para cada escopo.
- **Endereços a serem atribuídos:** Faixa de IPs disponíveis para configuração automática, ou reservados para estações específicas previamente cadastradas.
- **Duração das concessões oferecidas pelo servidor.** Este tempo define o período durante o qual o endereço IP atribuído pode ser utilizado pelo cliente. Antes que este tempo expire o cliente deve renovar no servidor a concessão de seu IP.

Com o servidor em operação na rede, as máquinas da rede com um cliente DHCP podem configurar todas as informações necessárias ao funcionamento dos protocolos TCP/IP

dinamicamente sempre que forem inicializadas. Esse procedimento é realizado toda vez que a máquina é inicializada como critério de confirmação, para não haver duplicação na rede. Os servidores DHCP fornecem essa configuração na forma de uma oferta de concessão de endereço para clientes solicitantes.

Durante a obtenção dos parâmetros de configuração necessários, algumas mensagens são trocadas entre o servidor e o cliente DHCP. A seguir é descrita a sequência destas mensagens com seus respectivos significados.

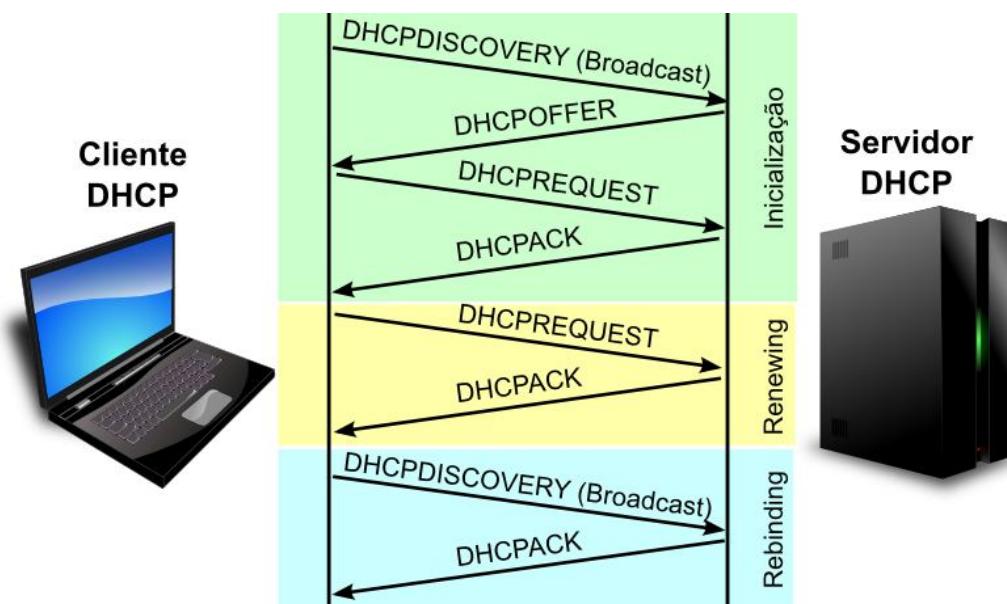


Figura 41 - Mensagens DHCP para configuração automática de endereços IP

A sequência de mensagens para inicialização da configuração dos parâmetros de rede são:

1. O cliente envia mensagem *broadcast* **DHCPDISCOVERY**. Uma mensagem *broadcast* em camada 2 significa que o cliente inclui seu endereço de hardware (MAC) como endereço de origem e o endereço ff:ff:ff:ff:ff:ff como MAC de destino. Esta mensagem será recebida por todos os computadores de uma mesma LAN. No **DHCPDISCOVERY** o cliente pode incluir uma sugestão de endereço IP, no campo NOME DO ARQUIVO DE PARTIDA, além da duração máxima da requisição em segundos. Após este tempo, considera-se a requisição perdida.

2. O servidor pode responder com uma mensagem DHCPOFFER, que inclui um endereço IP disponível no campo SEU ENDEREÇO IP além de outros parâmetros nas OPÇÕES. O servidor verifica a disponibilidade do endereço IP antes de disponibilizá-lo.
3. O cliente envia uma mensagem DHCPREQUEST que inclui o identificador do servidor DHCP que lhe enviou a primeira mensagem. Isto é necessário para o caso de o cliente receber respostas de mais de um servidor DHCP.
4. O servidor, após receber a mensagem de REQUEST, salva as configurações e responde com uma mensagem DHCPACK confirmando as configurações ofertadas anteriormente.
5. O cliente efetua uma verificação se o endereço recebido está em uso na rede através do protocolo ARP. O protocolo ARP envia uma mensagem *broadcast*, que é propagada para toda a rede interna (LAN) perguntando qual é o endereço de hardware do IP em questão. Caso receba uma resposta de alguma estação dentro do tempo determinado de duração da requisição ARP (tempo de *timeout*) é constatado que o endereço está em uso e então o cliente DHCP envia uma mensagem DHCPDECLINE ao servidor. Assim, caso o cliente receba uma mensagem DHCPNACK o processo é reiniciado.

No caso de o cliente já ter um endereço IP alocado, desejando apenas renová-lo (*Renewing*), enviará diretamente um DHCPREQUEST. Caso o cliente não receba resposta do servidor DHCP ele entrará em estado de *Rebinding* enviando novamente a mensagem de DHCPDISCOVERY, mas agora preenchendo os campos da mensagem padrão com as configurações desejadas. Um servidor pode ou não aceitar, sendo que no último caso, enviará uma mensagem de DHCPOFFER para o cliente.

## Conclusão

Nesta unidade aprendemos a importância de um servidor DHCP quando as redes crescem, já que fica inviável administrativamente manter a configuração de rede, para todas as máquinas do parque de uma empresa. Conhecemos também, detalhes do funcionamento do protocolo e suas mensagens padrões.

# UNIDADE 19

*Objetivo: Utilizando o analisador de pacotes Wireshark, mostrar a troca de mensagens para alocação dinâmica de endereços utilizando DHCP.*

## Analisador de Pacotes de Rede – DHCP

### Introdução

Como estudado, o DHCP utiliza um modelo de funcionamento cliente/servidor. Suas mensagens utilizam o protocolo de transporte UDP. As mensagens do cliente para o servidor são enviadas para a porta 67. Já as que são originadas no servidor são enviadas para a porta 68 do cliente DHCP.

Nossa abordagem prática mostrará as mensagens de inicialização de um cliente: DHCPDISCOVER, DHCPOFFER, DHCPREQUEST, DHCPACK.

Neste teste o Wireshark foi instalado na máquina cliente, sem que houvesse necessidade do uso de qualquer outro dispositivo para que pudéssemos interceptar pacotes de outras máquinas. A máquina cliente, neste caso, tem sistema operacional Linux, Ubuntu 10.04.

A descrição da rede:

Rede Local: 10.255.255.0/24

Router: 10.255.255.1

Servidor DHCP: 10.255.255.1

IP da máquina cliente armazenado de conexões anteriores: 192.168.2.135

Descrição do experimento: iniciou-se a captura de pacotes no Wireshark na interface de rede da máquina cliente, executando-se em seguida o software cliente DHCP (dhclient) em terminal. Assim que os pacotes foram capturados, a coleta de pacotes foi parada.

Na Figura 42 é apresentada a tela de captura de pacotes do Wireshark. Podemos ver marcado em cinza o pacote 10, pacote inicial enviado pelo cliente DHCP, e em azul a sequência de pacotes 12, 13 e 14, que dão prosseguimento à configuração da rede da

máquina cliente. Nesta figura, no quadro em evidência é mostrado o detalhamento do pacote 10, na área inferior da janela do Wireshark. Nela podemos ver claramente que para a transmissão da mensagem DHCPDISCOVER utilizou-se o protocolo UDP (User Datagram Protocol) com porta de origem UDP 68 (máquina cliente) e com porta de destino 67 (servidor DHCP)

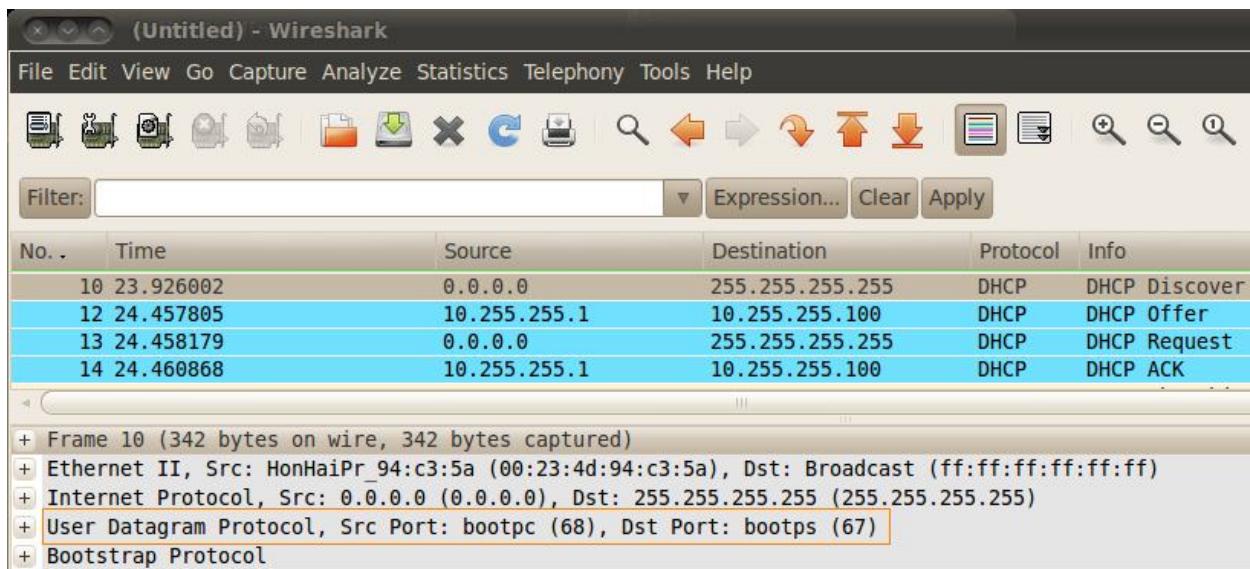


Figura 42 - Mensagens DHCP capturadas

O detalhamento da mensagem DHCPDISCOVER é mostrado na Figura 43, em caso de dúvida no formato da mensagem e qual é a função dos campos desta revise a unidade de DHCP. Note que os campos da mensagem Client IP Address (ENDEREÇO IP DO CLIENTE) e Your IP Address (SEU ENDEREÇO IP) são preenchidos com 0.0.0.0, já que a máquina cliente está requisitando um IP da rede em questão. Porém no quadro inferior o campo Requested IP Address (campo de opções (Options) que em português significa Endereço IP requisitado) tem valor 192.168.2.135, IP utilizado e armazenado na última rede acessada pela máquina cliente. O cliente DHCP tenta conseguir o mesmo endereço da última configuração válida. Também é possível ver nesta figura, na linha do pacote 10, destacado em cinza, que esta mensagem é de broadcast, enviada para o endereço 255.255.255.255. Note que o endereço MAC do cliente é explicitado nesta mensagem e é a partir dele que a mensagem DHCPOFFER chega até a máquina cliente através da

camada de enlace, camada 2 do modelo OSI, uma vez que esta máquina ainda não tem endereço IP, camada 3, configurado.

| No. | Time      | Source       | Destination     | Protocol | Info          |
|-----|-----------|--------------|-----------------|----------|---------------|
| 10  | 23.926002 | 0.0.0.0      | 255.255.255.255 | DHCP     | DHCP Discover |
| 12  | 24.457805 | 10.255.255.1 | 10.255.255.100  | DHCP     | DHCP Offer    |
| 13  | 24.458179 | 0.0.0.0      | 255.255.255.255 | DHCP     | DHCP Request  |
| 14  | 24.460868 | 10.255.255.1 | 10.255.255.100  | DHCP     | DHCP ACK      |

- Bootstrap Protocol

Message type: Boot Request (1)  
 Hardware type: Ethernet  
 Hardware address length: 6  
 Hops: 0  
 Transaction ID: 0xb16c0b67  
 Seconds elapsed: 0  
 + Bootp flags: 0x0000 (Unicast)  
 Client IP address: 0.0.0.0 (0.0.0.0)  
 Your (client) IP address: 0.0.0.0 (0.0.0.0)  
 Next server IP address: 0.0.0.0 (0.0.0.0)  
 Relay agent IP address: 0.0.0.0 (0.0.0.0)  
 Client MAC address: HonHaiPr 94:c3:5a (00:23:4d:94:c3:5a)  
 Client hardware address padding: 000000000000000000000000  
 Server host name not given  
 Boot file name not given  
 Magic cookie: (OK)  
 + Option: (t=53,l=1) DHCP Message Type = DHCP Discover  
 + Option: (t=50,l=4) Requested IP Address = 192.168.2.135  
 + Option: (t=12,l=10) Host Name = "salim-note"  
 + Option: (t=55,l=13) Parameter Request List  
 End Option

**Figura 43 - Detalhamento da mensagem DHCPDISCOVER**

A próxima mensagem a ser detalhada é a DHCPOFFER. As informações enviadas começam pelo campo Your IP Address, onde é preenchido o IP que foi oferecido para a máquina cliente. Também é comunicado ao cliente, no campo de Opções, o IP do servidor DHCP no campo de Opções (Options) a máscara de rede 255.255.255.0, o IP do roteador gateway 10.255.255.1 e o endereço de broadcast da rede 10.255.255.255.

| No. | Time      | Source       | Destination     | Protocol | Info         |
|-----|-----------|--------------|-----------------|----------|--------------|
| 12  | 24.457805 | 10.255.255.1 | 10.255.255.100  | DHCP     | DHCP Offer   |
| 13  | 24.458179 | 0.0.0.0      | 255.255.255.255 | DHCP     | DHCP Request |
| 14  | 24.460868 | 10.255.255.1 | 10.255.255.100  | DHCP     | DHCP ACK     |

**Bootstrap Protocol**

- Message type: Boot Reply (2)
- Hardware type: Ethernet
- Hardware address length: 6
- Hops: 0
- Transaction ID: 0xb16c0b67
- Seconds elapsed: 0
- + Bootp flags: 0x0000 (Unicast)
  - Client IP address: 0.0.0.0 (0.0.0.0)
  - Your (client) IP address: 10.255.255.100 (10.255.255.100)
  - Next server IP address: 0.0.0.0 (0.0.0.0)
  - Relay agent IP address: 0.0.0.0 (0.0.0.0)
  - Client MAC address: HonHaiPr\_94:c3:5a (00:23:4d:94:c3:5a)
  - Client hardware address padding: 000000000000000000000000
  - Server name option overloaded by DHCP
  - Boot file name option overloaded by DHCP
  - Magic cookie: (OK)
  - + Option: (t=53,l=1) DHCP Message Type = DHCP Offer
  - + Option: (t=54,l=4) DHCP Server Identifier = 10.255.255.1
  - + Option: (t=1,l=4) Subnet Mask = 255.255.255.0
  - + Option: (t=51,l=4) IP Address Lease Time = 2 hours
  - + Option: (t=52,l=1) Option Overload = Boot file and server host names hold options
  - Boot file name option overload
  - End Option (overload)
  - Server host name option overload
  - End Option (overload)
  - + Option: (t=28,l=4) Broadcast Address = 255.255.255.255
  - + Option: (t=3,l=4) Router = 10.255.255.1
  - + Option: (t=6,l=8) Domain Name Server
  - + Option: (t=26,l=2) Interface MTU = 576
  - End Option

**Figura 44 – Detalhamento da Mensagem DHCPOFFER**

O cliente procede com o processo enviando a mensagem DHCPREQUEST, apresentada na Figura 45, em broadcast ainda utilizando o IP de origem 0.0.0.0. Nela é identificado nas opções o IP do servidor DHCP que o cliente utilizará, pois ele poderia ter recebido mais de uma resposta quando enviou a primeira mensagem do processo. Neste ponto o cliente já tem as informações necessárias para a configuração da rede IP e aguardará a mensagem de confirmação DHCPACK, mostrada na Figura 46.

| No. | Time      | Source       | Destination     | Protocol | Info         |
|-----|-----------|--------------|-----------------|----------|--------------|
| 13  | 24.458179 | 0.0.0.0      | 255.255.255.255 | DHCP     | DHCP Request |
| 14  | 24.460868 | 10.255.255.1 | 10.255.255.100  | DHCP     | DHCP ACK     |

**Bootstrap Protocol**

- Message type: Boot Request (1)
- Hardware type: Ethernet
- Hardware address length: 6
- Hops: 0
- Transaction ID: 0xb16c0b67
- Seconds elapsed: 0
- + Bootp flags: 0x0000 (Unicast)
  - Client IP address: 0.0.0.0 (0.0.0.0)
  - Your (client) IP address: 0.0.0.0 (0.0.0.0)
  - Next server IP address: 0.0.0.0 (0.0.0.0)
  - Relay agent IP address: 0.0.0.0 (0.0.0.0)
  - Client MAC address: HonHaiPr\_94:c3:5a (00:23:4d:94:c3:5a)
  - Client hardware address padding: 000000000000000000000000
  - Server host name not given
  - Boot file name not given
  - Magic cookie: (OK)
- + Option: (t=53,l=1) DHCP Message Type = DHCP Request
- + Option: (t=54,l=4) DHCP Server Identifier = 10.255.255.1
- + Option: (t=50,l=4) Requested IP Address = 10.255.255.100
- + Option: (t=12,l=10) Host Name = "salim-note"
- + Option: (t=55,l=13) Parameter Request List
- End Option
- Padding

**Figura 45 - Detalhamento da mensagem DHCPREQUEST**

A mensagem DHCPACK já é recebida pelo cliente com o IP de destino correto, apesar de o pacote ter sido encaminhado novamente em camada 2. A partir desta tem sua configuração de rede confirmada.

Assim nossa experiência termina. Para *Renewing* e *Rebinding* as mensagens mostradas aqui são semelhantes e por isso estes dois tipos não são tratados nesta unidade.

| No.   | Time      | Source       | Destination    | Protocol | Info     |
|---|-----------|--------------|----------------|----------|----------|
| 14  | 24.460868 | 10.255.255.1 | 10.255.255.100 | DHCP     | DHCP ACK |
|   |           |              |                |          |          |
| - Bootstrap Protocol  |           |              |                |          |          |
| Message type: Boot Reply (2)<br>Hardware type: Ethernet<br>Hardware address length: 6<br>Hops: 0<br>Transaction ID: 0xb16c0b67<br>Seconds elapsed: 0  |           |              |                |          |          |
| + Bootp flags: 0x0000 (Unicast)<br>Client IP address: 0.0.0.0 (0.0.0.0)<br>Your (client) IP address: 10.255.255.100 (10.255.255.100)  |           |              |                |          |          |
| Next server IP address: 0.0.0.0 (0.0.0.0)<br>Relay agent IP address: 0.0.0.0 (0.0.0.0)<br>Client MAC address: HonHaiPr_94:c3:5a (00:23:4d:94:c3:5a)<br>Client hardware address padding: 000000000000000000000000<br>Server name option overloaded by DHCP<br>Boot file name option overloaded by DHCP<br>Magic cookie: (OK) |           |              |                |          |          |
| + Option: (t=53,l=1) DHCP Message Type = DHCP ACK<br>+ Option: (t=54,l=4) DHCP Server Identifier = 10.255.255.1<br>+ Option: (t=1,l=4) Subnet Mask = 255.255.255.0  |           |              |                |          |          |
| + Option: (t=51,l=4) IP Address Lease Time = 2 hours<br>+ Option: (t=52,l=1) Option Overload = Boot file and server host names hold options<br>Boot file name option overload<br>End Option (overload)<br>Server host name option overload<br>End Option (overload)   |           |              |                |          |          |
| + Option: (t=28,l=4) Broadcast Address = 255.255.255.255<br>+ Option: (t=3,l=4) Router = 10.255.255.1<br>+ Option: (t=6,l=8) Domain Name Server<br>+ Option: (t=26,l=2) Interface MTU = 576<br>End Option   |           |              |                |          |          |

**Figura 46 - Detalhamento da Mensagem DHCPACK**

# UNIDADE 20

*Objetivo: Entender a motivação, o funcionamento e os conceitos de uma VPN (Virtual Private Network, ou em português: Rede Virtual Privada)*

## VPN

### Introdução

Com a globalização do mundo dos negócios, um ambiente de trabalho mais dinâmico é exigido. Ao invés de lidar só com assuntos locais e regionais, muitos negócios têm de levar em conta o mercado e a logística globais. Muitas empresas têm instalações espalhadas por todo o país ou mesmo pelo mundo, e até mesmo escritórios dentro da casa de seus funcionários, conhecidos como *home office*. Além disto, executivos, funcionários ou vendedores de uma empresa, durante uma viagem de negócios, podem precisar de acesso a arquivos e documentos só disponíveis dentro da rede privada de uma empresa. Em suma, o que todos precisam é de uma maneira de manter uma comunicação rápida, segura e confiável onde quer que seus escritórios estejam.

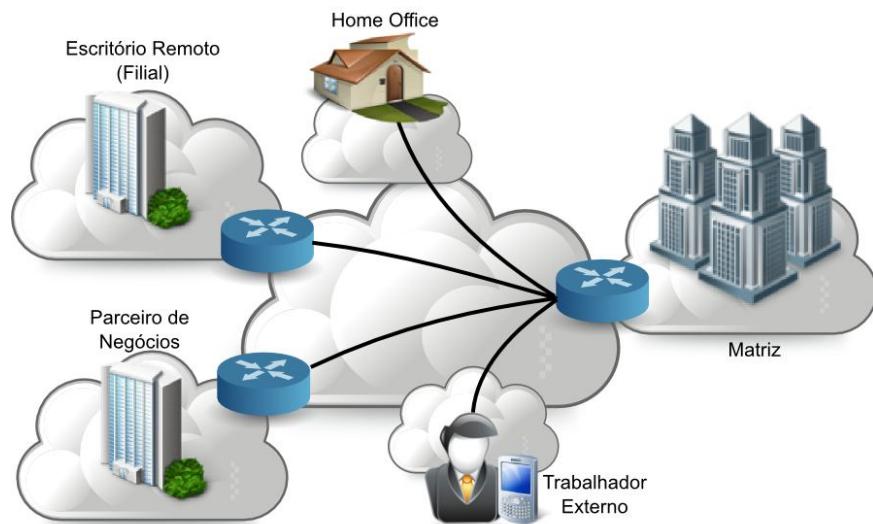
Até recentemente, isso significou o uso de linhas dedicadas para manter uma rede de longa distância (WAN, *Wide Area Network*). Canais de dados dedicados, variando desde ISDN (*Integrated Services Digital Network*, ou em português serviços de rede digital com integração) com conexão de 128 Kbps, até enlaces contratados com fibras óticas com banda variando de 34Mbps até 10Gbps, utilizando serviços como o Frame Relay, ATM, SDH e WDM das operadoras, proveem a empresa com um meio de expandir suas redes privadas além de suas áreas vizinhas. A grande vantagem de uma rede WAN sobre redes públicas é quando se trata de confiabilidade, desempenho e segurança. Porém, manter uma rede WAN, especialmente quando usamos linhas dedicadas, pode se tornar um tanto quanto custoso, com o aumento da capilaridade e das distâncias entre os escritórios.

As VPNs começaram a serem desenvolvidas em meados dos anos 90, por duas das maiores empresas de tecnologia, a Microsoft e a Cisco. Estas empresas tentaram

funcionar com suas respectivas implementações de VPNs na esperança de que essas se tornariam um “padrão industrial”.

A Microsoft, devido à própria natureza da empresa, chegou ao mercado VPN por meio do ponto de vista dos sistemas operacionais, possibilitando o uso através de computadores comuns. Desenvolveu o *Point-to-Point Tunneling Protocol* (PPTP, ou em português: Protocolo de Tunelamento Ponto a Ponto). Ao mesmo tempo a Cisco tinha a visão estritamente do ponto de vista de equipamentos de rede com um protocolo chamado L2F (*Layer 2 Forwarding*, Encaminhamento Camada 2). As duas implementações tinham suas vantagens e desvantagens. A partir de um acordo de cooperação entre as duas empresas foi desenvolvido um novo protocolo o L2TP (*Layer 2 Tunneling Protocol*, Protocolo de Tunelamento Camada 2).

As primeiras VPNs necessitavam de mão-de-obra especializada para sua implantação e manutenção. Hoje, a tecnologia se desenvolveu e tornou-se mais simples e adequada ao uso em negócios de quaisquer tamanhos, possibilitando assim seu uso em pequenas empresas, antes afastadas deste tipo de solução.



**Figura 47 - Exemplo de uma VPN interligando diferentes tipos de usuário**

Como em outros casos, o crescimento da popularidade da Internet fez com que os escritórios se voltassem para as VPNs como forma de ampliar suas próprias redes. Porém redes públicas são consideradas não confiáveis, pois os dados que nelas trafegam estão sujeitos à interceptação e à captura. Para contornar esse problema, primeiro

chegaram as *intranets*, que tinham acesso controlado por senhas com uso específico, somente liberado para funcionários das empresas. Isto limitava o acesso, porém não garantia que os pacotes não pudessem ser interceptados. Atualmente, as técnicas de VPN possibilitam a criação de grandes e capilarizadas redes.

Utilizando a Internet, as VPNs possibilitam as empresas reduzirem de forma imediata o custo com as conexões de longa distância (principalmente as companhias com atuação internacional), aluguel de linhas privadas e equipamentos de conexão, como banco de modems e servidores de autenticação.

Há várias formas de se implementar uma VPN utilizando diferentes mecanismos de transporte de rede. Cada forma será mais adequada a um caso de uso. Utilizando como base o modelo OSI, quanto mais baixa é a camada mais próxima de ser a escolha mais adequada a unidades de grandes empresas, pois estas possuem gigantesco volume de dados trafegando e dispões de recursos suficientes para a implantação de enlaces dedicados para interconexão de suas unidades. À medida que nos voltamos para camadas superiores, a solução se torna mais viável a pequenas empresas e usuários remotos. Exemplos de técnicas utilizadas de acordo com as camadas OSI são: VPNs camada 2 (camada de enlace), utilizando protocolos como PPTP e L2TP; camada 3 (camada de rede) utilizando IPSec (*IP Security Tunnel Mode*); e até camada 7 (camada de aplicação), usando o protocolos SSL (Security Sockets Layer) ou TLS (*Transport Layer Security*) para encriptação e segurança dos dados. Alguns tipos de VPN e outros conceitos de utilização serão abordados nesta Unidade.

## Requisitos Básicos

A utilização de redes públicas, como a Internet, tende a reduzir drasticamente os custos com a implantação de redes privadas, sendo este o grande motivador para o uso de VPNs. No entanto, para que esta abordagem se torne efetiva, a VPN deve prover um conjunto de funções que garantam: Confidencialidade, Integridade, Autenticidade.

O requisito de **confidencialidade** requer que todo o tráfego circulado em uma VPN seja privado, permitindo que apenas usuários autorizados entendam as informações circuladas. Do ponto de vista da utilização de uma rede pública, como a Internet, é uma tarefa relativamente simples a de interceptar uma sequência de dados. Assim, as VPNs

terão que utilizar de mecanismos de criptografia, tais como o SSL e TLS, visando que mesmos que estes dados sejam capturados, não possam ser entendidos. As chaves de criptografia utilizadas por estes mecanismos devem ser gerenciadas de forma a garantir sua alteração de forma periódica, visando sempre à manutenção de uma comunicação segura.

Também é preciso que as informações transmitidas entre dois pontos sejam íntegras. Atacantes como os crackers não devem obter sucesso em tentativas de captura e adulteração de dados. Algoritmos como o SHA-1 (*Secure Hash Algorithm 1*) e o MD5 (*Message Digest Algorithm 5*) são utilizados para garantir a **integridade** dos dados. Para evitar estes tipos de ataque, também é necessário que os endereços atribuídos aos clientes em uma rede privada não sejam divulgados. Neste sentido, devem ser adotados endereços diferentes para tráfego externo.

Por fim, é necessário que apenas usuários e equipamentos autorizados possam trocar dados entre si. A **autenticidade** não garante a propriedade de confidencialidade, porém é suficiente para alguns tipos de usuários. Serviços como o Radius, Active Directory, LDAP, podem ser utilizados para autenticar usuários.

Outros requisitos são desejáveis em uma rede VPN. Atualmente nas empresas, existem inúmeras aplicações que utilizam de diferentes tipos protocolos para executar suas finalidades, como acesso a um sistema de estoque, vendas, etc. Desta forma, tecnologias VPNs com suporte a multi protocolos são bastante interessantes. Muitas vezes este suporte chega a ser necessário, uma vez que uma empresa normalmente não deseja realizar investimentos já efetuados para o desenvolvimento de novas aplicações com a mesma utilidade das que já possuem para que alcancem os objetivos de acesso confidencial, íntegro e seguro.

# UNIDADE 21

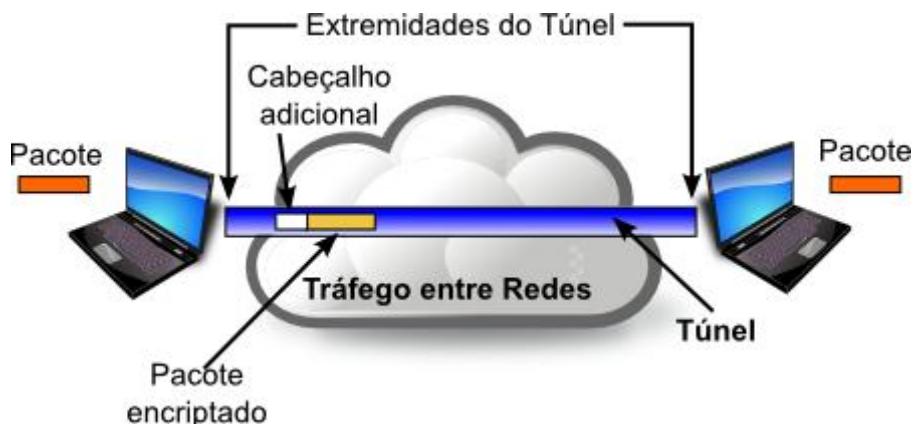
*Objetivo: Entender a motivação, o funcionamento e os conceitos de uma VPN (Virtual Private Network, ou em português: Rede Virtual Privada).*

## VPN (Continuação)

### Soluções com Tunelamento

A tecnologia de tunelamento é anterior ao desenvolvimento das redes virtuais privadas. O processo para a realização de túneis consiste no encapsulamento de um protocolo em outro. O uso desta técnica nas VPNs permite que não só os dados de um pacote que circula em redes públicas sejam criptografados, mas também todo o conteúdo dos cabeçalhos dos protocolos utilizados pelas aplicações destas redes. Isto traz uma segurança ainda maior para os usuários interessados em utilizar uma rede privada.

Ao entrar no túnel um pacote é criptografado e adicionado como dados em outro pacote com informações suficientes para que este seja encaminhado por toda a rede, ou Internet, até a outra extremidade do túnel. Lá este é descriptografado e retorna ao formato original. É isto que permite o encapsulamento de datagramas de um protocolo em outro. Por exemplo, pacotes de protocolo IPX podem ser encapsulados e transportados dentro de pacotes TCP/IP.



**Figura 48 - Encaminhamento de um pacote em um túnel**

Os túneis podem ser criados de duas diferentes formas: voluntárias e compulsórias. Em um **Túnel Voluntário** o usuário emite uma solicitação VPN através de um *software* para configurar e criar um túnel voluntário. Neste caso, o computador do usuário funciona como uma das extremidades do túnel e, também, como cliente do túnel. Já em um **Túnel Compulsório**, um servidor de acesso VPN, que pode estar associado a um *proxy* por exemplo, configura e cria o túnel de forma compulsória, isto é, o túnel para acesso a VPN é criado independentemente da vontade do usuário. Neste caso, este servidor funcionará como *gateway* para acesso às estações dentro da VPN e é nele que estará a extremidade do túnel do ponto de vista do usuário que solicitou o acesso a uma estação dentro da VPN.

Desta forma, quando se utiliza um servidor VPN, o computador cliente trafega seus dados com destino a outra estação do outro lado do túnel primeiramente por uma rede local sem criptografia e a partir da passagem dos dados pelo servidor VPN os dados passam a ser criptografados por ele.

Vimos duas formas de criação de uma das extremidades de um túnel VPN através de servidor VPN ou através da solicitação de um usuário remoto. Desta forma, dependendo de como se dá a criação dos túneis VPN, as topologias de acesso são nomeadas como:

- Topologia **gateway-gateway**: dois servidores de acesso estabelecem um túnel entre duas redes locais
- Topologia **gateway-usuário**: o túnel se inicia em um servidor de acesso e tem fim em uma máquina cliente remota;
- Topologia **usuário-usuário**: um túnel é configurado entre duas máquinas clientes através de solicitação dos usuários.

### Protocolos Utilizados em Diferentes Camadas OSI

Para se estabelecer um túnel é necessário que as suas extremidades utilizem o mesmo protocolo de tunelamento.

O tunelamento pode ocorrer na camada 2 ou 3 (respectivamente enlace e rede) do modelo de referência OSI (*Open Systems Interconnection*).

## Tunelamento em Camada 2

Protocolos de camada 2 têm por objetivo controlar o acesso ao meio, e encaminhar pacotes das camadas superiores de uma máquina a outros dispositivos de rede conectados. Podemos citar o IP e IPX como exemplo de protocolos camada 3 transportados pelo protocolo ARP que funciona na camada 2.

Resumidamente, um túnel camada 2 deve transportar os pacotes de protocolos de níveis superiores no nível de rede local, LAN, criptografando e encapsulando-os em quadros de camada 2. Neste caso, o acesso se dá de forma compulsória e transparente ao usuário. Como exemplos, podemos citar os seguintes protocolos:

- **PPTP:** Considerado uma extensão do protocolo PPP uma vez que faz uso dos mecanismos de conexão, autenticação e compactação deste protocolo. Foi desenvolvido pelo Fórum PPTP, um consórcio que inclui US Robotics, Microsoft, 3Com, Ascend e ECI Telematics, e entregue 2 anos antes dos protocolos L2TP. Sua implementação mais conhecida é a da Microsoft, que é amplamente utilizada em sistemas Windows. As premissas deste protocolo era ser simples, possuir suporte a múltiplos protocolos e ser versátil, podendo ser utilizado para percorrer caminhos entre diversas redes IP.
- **L2TP:** Projeto da Cisco Systems e Microsoft, foi homologado pela Internet Engineering Task Force (IETF) como protocolo padrão, combina características dos protocolos L2F e PPTP. Conforme definido na RFC2661 (<http://tools.ietf.org/html/rfc2661>) o L2TP estende as funcionalidades do PPP permitindo que L2TP e PPP estejam em equipamentos diferentes e mesmo assim interligados pela rede. As conexões feitas pelos clientes remotos chegam até um Concentrado de Acesso (LAC, *L2TP Access Concentrator*), mas as conexões PPP se concluem em um Servidor de Rede L2TP (LNS, *L2TP Network Server*). O L2TP trabalha com os dois modos de tunelamento, voluntário e compulsório e permite a autenticação de túneis utilizando os mesmos protocolos inclusos no PPP que o PPTP faz uso. Porém servidores Radius e Tacacs são também suportados. A

versão 3 do protocolo que é definida em 2005 na RFC3031 (<http://tools.ietf.org/html/rfc3931>) acrescenta funcionalidades adicionais de segurança, melhora da habilidade de encapsulamento, e a habilidade de ser utilizado sobre outros tipos de enlace que não o PPP, como por exemplo, Frame Relay, Ethernet, ATM, etc. Apesar de ser atual, o L2TP tem a desvantagem de possuir falhas de confidencialidade. Para suprir esta deficiência ele é normalmente associado ao protocolo IPSec.

### Tunelamento em Camada 3

O tunelamento em camada 3 encapsula e criptografa os pacotes IP que trafegam entre diferentes redes através de um cabeçalho adicional utilizando o IPSec (*Internet Protocol Security Tunnel Mode*, ou em português, Modo de Túnel Seguro do Protocolo de Internet). O **IPSec** é um conjunto de protocolos com estrutura desenvolvida em padrões abertos para proteger o tráfego dos pacotes IP garantindo confidencialidade, integridade e autenticidade através da criação de túneis, tornando possível seu uso para assegurar dados em uma rede pública. Implementa a criptografia e autenticação na camada de rede, permitindo aos aplicativos finais se aproveitarem desta vantagem de forte segurança sem alterações ou configurações adicionais. Os algoritmos de criptografia utilizados são HMAC-SHA1, para proteção de integridade e autenticação, o TripleDES-CBC e o AES-CBC para confidencialidade, para detalhes consulte a RFC4835 (<http://tools.ietf.org/html/rfc4835>). O IPSec foi desenvolvido pelo Grupo de Trabalho de Segurança do IP da IETF com o intuito de ser o protocolo padrão de endereçamento para a próxima versão do IP, chamado IPv6. Por enquanto, o IPSec é apenas opcional para o IPv4. Por ser um protocolo padrão, existem muitas implementações disponíveis. Devido a isto, a segurança provida pelo IPSec depende muito dos recursos utilizados em cada implementação. Vale lembrar que manter os sistemas operacionais de todos os equipamentos de rede atualizados previnem as redes de serem vítimas de vulnerabilidades divulgadas e já resolvidas.



## Dica

Toda a lista de RFCs utilizadas em VPNs é apresentada neste site:  
<http://www.vpnc.org/vpn-standards.html>. O site está em inglês, porém isso não dificulta a visualização dos RFCs.



# UNIDADE 22

*Objetivo: Entender a motivação, o funcionamento e os conceitos de uma VPN (Virtual Private Network, ou em português: Rede Virtual Privada).*

## VPN (Continuação)

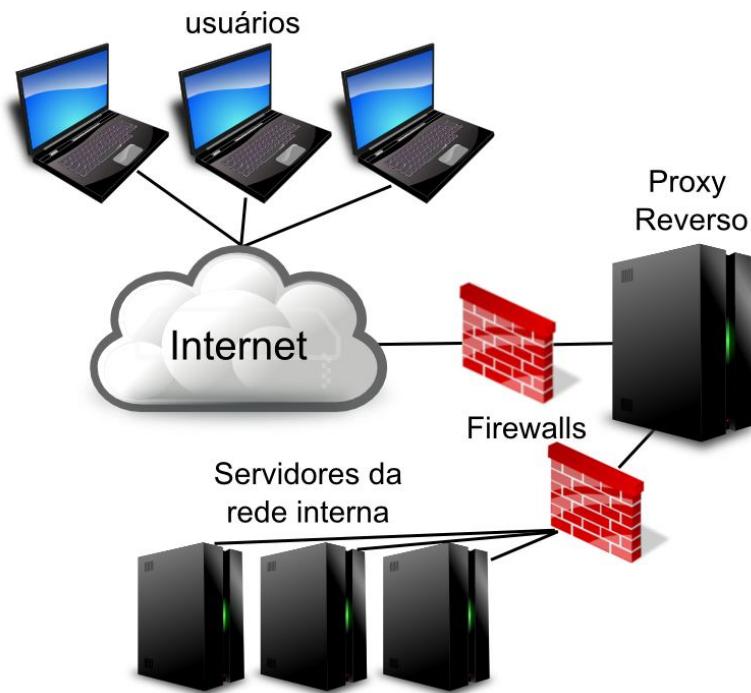
### VPNs em Camada 7

As VPNs que vimos até agora usam algum mecanismo de transporte de rede. Para um usuário remoto ou para pequenas empresas a solução pode ser utilizar uma VPN com base em uma aplicação (camada 7 do modelo OSI), possibilitando assim o uso da Internet disponível em qualquer lugar, em um computador ou *smartphone* qualquer. Para isso as aplicações utilizam da técnica de SSL/TLS, apresentada na Unidade 4.

As **VPNs SSL** atuam basicamente como um *proxy* reverso que funciona sobre HTTPS e consequentemente utiliza a camada SSL/TLS para prover segurança. A maior parte delas pode tunelar qualquer tipo de tráfego. Mas fique atento, uma VPN SSL é diferente de acessar uma página em HTTPS. A diferença básica é que uma VPN SSL depende de um servidor similar a um *proxy* reverso e permite acesso a basicamente qualquer tipo de aplicação e não só a aplicação HTTP.

Um *proxy* reverso funciona como um *gateway* para servidores. É muito utilizado para balanceamento de tráfego. Ele habilita um servidor *web* para prover conteúdo de forma transparente e atua de forma a fornecer controle de acesso ao conteúdo destes servidores. Máquinas externas a rede local acessarão o *proxy* reverso como se ele fosse o provedor de serviços e conteúdo. Assim que uma requisição é feita ao *proxy* reverso, este se encarrega de buscar as informações necessárias para a resposta desta requisição em um dos servidores internos e as retorna à máquina que fez a requisição.

O uso deste tipo de especial de *proxy* fortalece a segurança, pois as conexões não são realizadas diretamente aos servidores, e possibilitam o registro de *logs* permitindo a análise de tráfego para mitigar o problema de ataques a rede abaixo do *proxy*.



**Figura 49 - Exemplo de proxy reverso**

Numa VPN SSL, um servidor similar a um *proxy reverso* deve criar os túneis SSL até os clientes que se autenticarem nele. Também há a possibilidade de não ser obrigatória esta autenticação. Assim, o controle de acesso é de responsabilidade deste *servidor*. O servidor de uma VPN SSL pode suportar quaisquer tipos de aplicações e não só aplicações web, como, por exemplo, um serviço de impressão, de acesso a arquivos e outros recursos de rede.

Numa VPN SSL, aplicações como HTTP e FTP, disponíveis na maior parte dos navegadores de Internet, podem então solicitar uma conexão segura imediatamente, não havendo a necessidade de um software de cliente adicional ao que já está no sistema operacional do usuário remoto.

Quando usar uma VPN SSL? A escolha de um ou outro tipo de VPN não deve ser baseada unicamente nas características técnicas. A melhor opção será aquela que melhor atender aos requisitos que você tem, às funcionalidades de que o negócio de seu cliente precisa. As duas soluções têm o mesmo nível de segurança. VPNs SSL permitem um controle bem mais específico sobre os tipos de recursos, aplicações e serviços a que

o usuário remoto tem acesso, pois o suporte a essas aplicações deve ser liberado no servidor e o acesso a este pode gerar logs que trazem a possibilidade da análise do tráfego solicitado. Já o usuário de uma VPN IPSec tem, por exemplo, após conectado, ampla visibilidade e acesso aos recursos da rede, como se estivesse num ponto físico da rede local.

## **Conclusão**

As VPNs são uma alternativa viável para a transmissão segura de informações através de redes públicas ou privadas. Fornecem mecanismos de autenticação, integridade e criptografia com variados níveis de segurança em diferentes camadas do modelo OSI de comunicação de dados. Permite que de unidades pequenas e distantes de empresas sejam conectadas de forma segura sem a contratação de enlaces dedicados, muitas vezes onerosos demais para serem viáveis.

Entretanto, sua aplicação em ambientes onde as aplicações têm elevados requisitos de tempo de transmissão o seu uso deve ser analisado com cuidado. Pois a sequência de algoritmos utilizados para prover segurança demanda tempo de processamento dos dados, podendo ocorrer problemas de desempenho e atrasos na transmissão sobre os quais a organização não tem nenhum tipo de gerência ou controle, comprometendo a qualidade desejada nos serviços corporativos. A decisão de utilização de uma VPN deve levar em consideração aspectos relacionados à segurança, custos, qualidade de serviço e facilidade de uso que variam de acordo com o negócio de cada organização.

# UNIDADE 23

*Objetivo: Conhecer os Protocolos e Respectivos Serviços da Camada de Aplicação Responsáveis pelo Acesso Remoto.*

## Serviços e Protocolos de Acesso Remoto - Telnet

Imagine que você é responsável por administrar um servidor que está dentro de uma salacofre (ambiente estanque que protege os equipamentos contidos em seu interior contra diversas ameaças físicas como incêndio, roubo, dentre outros) onde o acesso é bem restrito. Ou então, responsável por administrar vários roteadores e servidores localizados em diversos prédios, destinados a distribuir Internet em cada um dos prédios. Como você faria para acessar todos esses equipamentos?

Esta unidade apresenta os protocolos da camada de aplicação e respectivos serviços responsáveis por fornecer o acesso remoto a equipamentos ligados em rede, estejam eles fisicamente perto de você, distantes, em ambientes restritos, dentre outras localidades. Vale ressaltar que apenas os serviços provindos de protocolos aprovados em RFCs são apresentados.

### Telnet

O **Telnet** é um protocolo cliente-servidor cujo objetivo principal é a **conexão e acesso às máquinas remotas**, de forma a permitir a comunicação entre computadores que fazem parte de uma rede, seja ela uma rede local ou a Internet. Foi desenvolvido em 1969 começando com o RFC 15 (<http://tools.ietf.org/html/rfc15>) e posteriormente estendido no RFC 854 (<http://tools.ietf.org/html/rfc854>) e 855 (<http://tools.ietf.org/html/rfc855>), tornando-se um padrão IETF (*Internet Engineering Task Force*) em 1983, sendo um dos primeiros padrões da Internet.

O termo Telnet também se refere ao software que implementa a parte cliente do protocolo. O Telnet tipicamente estabelece uma conexão pela porta 23 do protocolo de transporte TCP, sendo, portanto, orientado à conexão. A aplicação servidora do serviço

Telnet “escuta” essa porta e recebe as teclas acionadas no terminal remoto localizado no cliente. Dessa forma, através de uma sessão de terminal, o Telnet permite o acesso e execução de aplicações e comandos em uma máquina remota.

O servidor é a máquina que fornece algum serviço, sendo o cliente o lado que utiliza esse serviço. Uma sessão Telnet permite acessar computadores, servidores, roteadores, switches, dentre outros. A sessão pode ser estabelecida independente dos sistemas operacionais utilizados pelas máquinas que hospedam as aplicações Telnet.

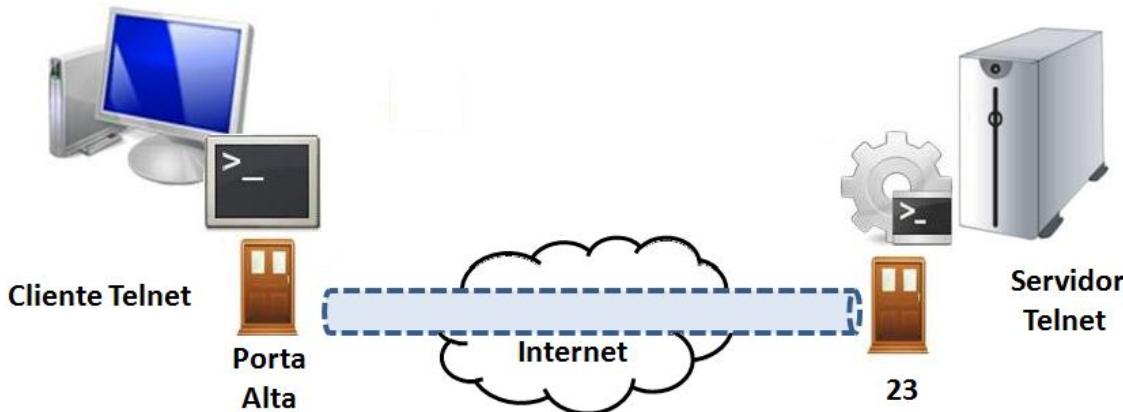


Figura 50 – Acesso Remoto via Telnet

Para acessar remotamente um equipamento, normalmente um login e senha são necessários. No Telnet, todo o tráfego na rede é realizado em texto plano (ASCII), ou seja, os dados transmitidos podem ser interceptados (caso haja “escuta” na rede) e visualizados normalmente (incluindo o login e a senha). Portanto, um usuário avançado com acesso ao roteador, switch, hub ou gateway localizados na rede entre os dois equipamentos utilizando Telnet pode interceptar os pacotes transmitidos (através de aplicativos comuns, como o *Wireshark* apresentado neste Módulo) e ter acesso ao login, senha ou qualquer outro caractere digitado.

Devido a essa grande desvantagem o uso do Telnet tem sido desaconselhado, recomendando-se a utilização do SSH. Assim como em outros protocolos mais antigos da Internet, extensões do protocolo Telnet fornecem segurança através do TLS e autenticação através do SASL (*Simple Authentication and Security Layer*, em português, Autenticação Simples e Camada de Segurança). Entretanto, poucas implementações do

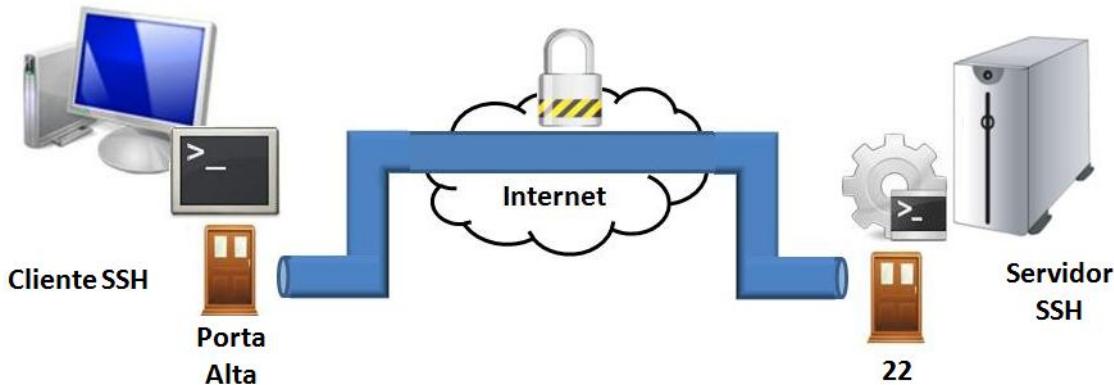
Telnet suportam essas extensões até mesmo porque como o SSH é adequado para a maioria das necessidades pouco interesse foi demonstrado em realizar as adequações citadas. Uma das poucas possibilidades de utilização do Telnet (sem a extensão citada) de forma “segura” é em redes internas seguras sem acesso de pessoas não confiáveis, ou através de VPN (Virtual Private Network).

# UNIDADE 24

*Objetivo: Conhecer o SSH, um Protocolo e Serviço para Acesso Remoto.*

## Serviços e Protocolos de Acesso Remoto - SSH (Secure Shell)

O SSH (Secure Shell), assim como Telnet, é um protocolo no modelo cliente-servidor que permite estabelecer conexão remota entre máquinas de forma transparente dando a impressão ao usuário que ele está operando diretamente o terminal de comandos da máquina servidora. Também utiliza como protocolo de transporte o TCP, sendo a sua porta padrão a 22.



**Figura 51 – Acesso Remoto via SSH**

O protocolo SSH foi criado devido às vulnerabilidades encontradas no Telnet. Dessa forma, sua principal diferença é a encriptação dos dados trafegados entre as máquinas conectadas, dificultando a descoberta dos conteúdos dos pacotes caso a transmissão seja interceptada. Portanto, de forma resumida, o SSH é um protocolo que permite estabelecer conexão remota entre máquinas fornecendo o intercâmbio de dados encriptados entre dois dispositivos em rede. A encriptação utilizada pelo SSH tem como objetivo fornecer integridade e confidencialidade aos dados transmitidos em uma rede insegura, como por exemplo, a Internet.

O SSH utiliza a **criptografia de chave pública** para autenticar o computador remoto e permitir que o computador remoto autentique o cliente. A criptografia de chave pública utiliza o **algoritmo de chaves assimétricas** para gerar duas chaves: a **chave privada secreta** e a **chave pública**, conforme vimos na Unidade 4. Em uma conexão SSH, o cliente e servidor trocam suas respectivas chaves públicas para realizar a autenticação de forma que possam se comunicar de forma segura. Uma vez realizada a autenticação, as mensagens são criptografadas com algoritmo de chaves simétricas.

Apesar de ser uma ferramenta nativa do Unix, existem clientes SSH para Windows e outras plataformas. O SSH permite gerenciar máquinas remotas através de aplicações em modo texto, via terminal, ou aplicações gráficas.

O SSH enfatiza a segurança, permitindo detectar quando a máquina remota não é a desejada, ou seja, é uma máquina impostora que foi substituída para tentar tirar proveito de uma conexão aberta para injetar dados na conexão. Outro ponto de segurança provido pelo SSH é a técnica de “despiste” que consiste em complicar a descoberta do pacote criptografado que possui a senha de acesso, dificultando a ação de quem pretende obter a senha para tentar descriptografá-la utilizando técnicas de força bruta.

A utilização típica do SSH é para efetuar login em uma máquina remota para executar comandos, mas também pode ser utilizado para: tunelamento, redirecionamento de portas TCP (*port fowarding*), X11 (protocolo e respectivo software associado que possibilita o emprego de uma interface gráfica) e para a transferência de arquivos utilizando os protocolos SFTP e SCP.

A primeira versão do protocolo (atualmente chamada de SSH-1) foi projetada em 1995 na Universidade Tecnológica de Helsinki após um ataque utilizando *sniffer* na rede da instituição. Em julho de 1995 sua implementação foi liberada como *freeware* e em pouco menos de seis meses a ferramenta rapidamente ganhou popularidade, atingindo a marca de 20.000 usuários em 50 países. Uma estimativa indica que no ano 2000 o SSH possuía 2 milhões de usuários.

Entre 1998 e 2001, algumas vulnerabilidades foram encontradas na versão 1 do SSH (a primeira foi identificada na versão 1.5), como por exemplo: inserção de conteúdo não autorizado no fluxo de dados, execução de código arbitrário com privilégios

administrativos no SSH, alteração do último bloco de uma sessão IDEA encriptada e permissão para um servidor malicioso redirecionar a autenticação de um cliente para outro servidor.

A segunda versão do protocolo, atualmente chamada de SSH-2, foi adotada como padrão em 2006 sendo que o seu núcleo foi publicado nos RFC de 4250 até 4254 (<http://tools.ietf.org/html/rfc4250>) para fornecer melhorias de segurança e funcionalidades em relação ao SSH-1. Essa versão do protocolo é incompatível com a SSH-1, porém as aplicações SSH podem fornecer suporte às duas versões. A versão SSH-2 mesmo sendo mais robusta do que a versão 1, teve uma vulnerabilidade descoberta em novembro de 2008 para o caso de uma configuração específica de uso da aplicação.

Devido ao fato do SSH-1 apresentar falhas de projeto que a tornam vulneráveis, geralmente é considerada obsoleta e deve ser evitada desabilitando explicitamente a compatibilidade na aplicação (quando o software suporta o SSH-2).

O OpenSSH (<http://www.openssh.com>) é uma ferramenta gratuita que suporta as versões 1.x e 2.x, sendo uma das implementações mais populares do SSH .



## Dica

Existem aplicações proprietárias com versões gratuitas que permitem acessar remotamente uma máquina através do modo gráfico. Dentre elas, podemos destacar o LogMeIn (<https://secure.logmein.com/BR/>) e o TeamViewer (<http://www.teamviewer.com/pt/index.aspx>).



# UNIDADE 25

*Objetivo: Apresentar as diferenças de confidencialidade entre os protocolos SSH e Telnet utilizando o analisador de pacotes de rede Wireshark.*

## **Analizador de Pacotes de Rede – Telnet x SSH**

### **Introdução**

Como estudado, o Telnet e o SSH utilizam o modelo de funcionamento cliente/servidor. Suas mensagens são transmitidas via protocolo de transporte TCP. O servidor destes protocolos envia e recebe mensagem nas portas 23 e 22 respectivamente

Nossa abordagem prática mostrará como se pode capturar o usuário e senha em uma conexão Telnet. Apresentará em seguida os pacotes com dados confidenciais numa conexão SSH.

Neste teste o Wireshark foi instalado na máquina cliente, sem que houvesse necessidade do uso de qualquer outro dispositivo para que pudéssemos interceptar pacotes de outras máquinas. A máquina cliente, neste caso, tem sistema operacional Linux, Ubuntu 10.04.

A descrição da rede:

Rede Local: 10.255.255.0/24

Servidor Telnet e SSH: 10.255.255.101

Usuário para acesso ao servidor: usuario

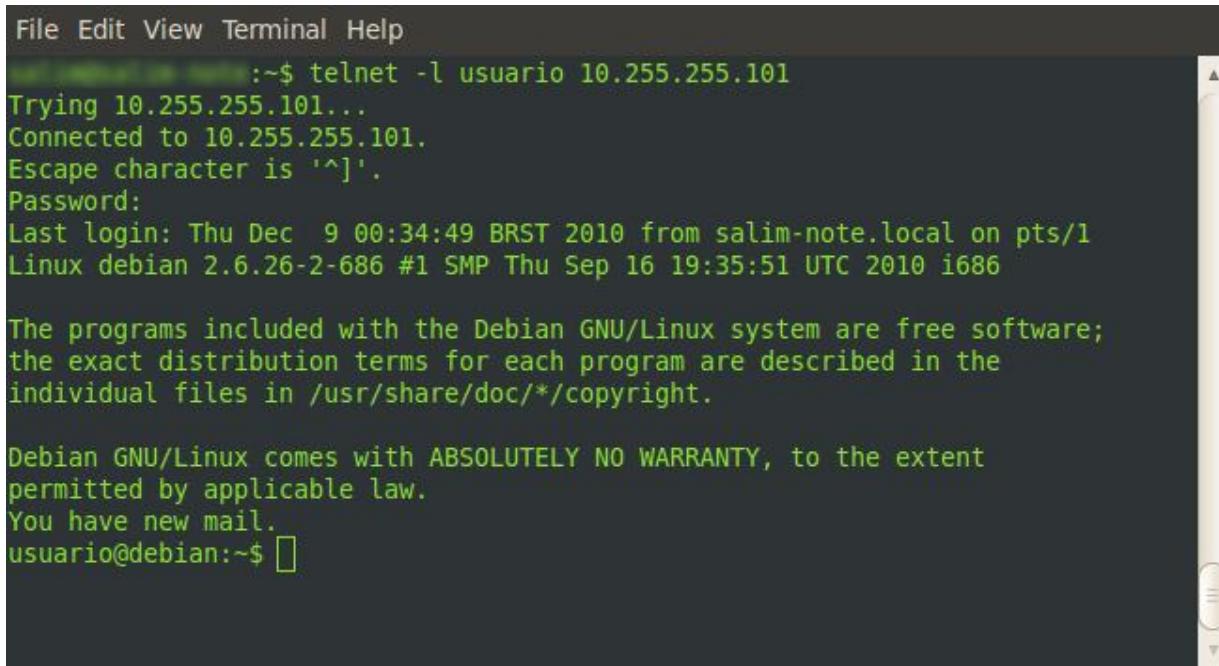
Senha do usuário para autenticação: esab123

Máquina cliente das conexões remotas: 10.255.255.100

Descrição do experimento: iniciou-se a captura de pacotes no Wireshark na interface de rede da máquina cliente, executando em uma conexão telnet com o usuário “usuario” com o comando telnet -l usuario 10.255.255.101 em terminal. Em seguida iniciamos uma conexão ssh com o mesmo usuário

## Telnet

A Figura 52 mostra o terminal de onde foi executada a conexão remota ficando explícito o êxito na conexão.



```

File Edit View Terminal Help
:~$ telnet -l usuario 10.255.255.101
Trying 10.255.255.101...
Connected to 10.255.255.101.
Escape character is '^]'.
Password:
Last login: Thu Dec  9 00:34:49 BRST 2010 from salim-note.local on pts/1
Linux debian 2.6.26-2-686 #1 SMP Thu Sep 16 19:35:51 UTC 2010 i686

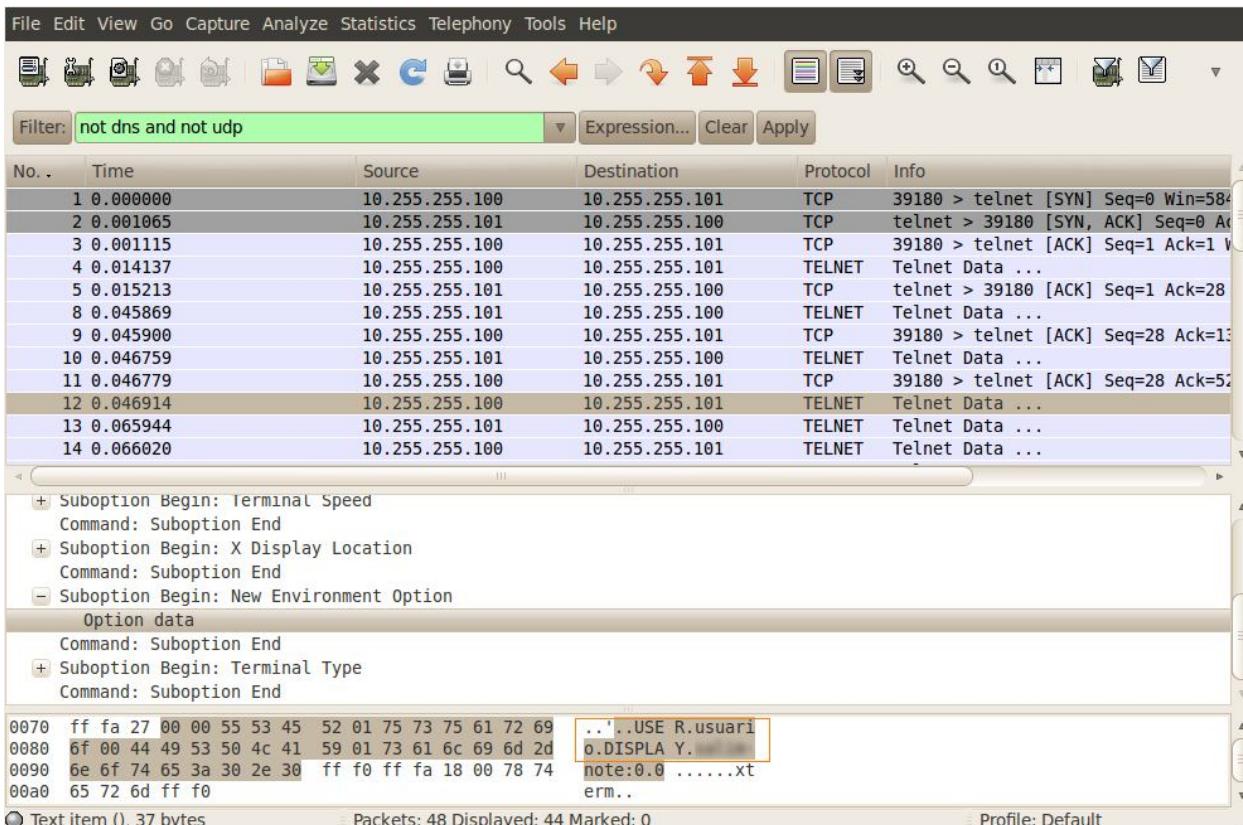
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
You have new mail.
usuario@debian:~$ 
    
```

Figura 52 - Terminal onde foi estabelecida uma conexão Telnet com êxito

A Figura 53 mostra os pacotes trocados numa conexão Telnet desde a iniciação desta no *three-way handshake* do TCP. O número 39180 nos pacotes do TCP indica a porta do cliente. Como o usuário já foi inserido no comando utilizado para início da conexão este aparece no pacote capturado número 12, *Telnet Data...* (dados telnet), em destaque. Podemos ver o usuário utilizado no quadro em destaque, na expressão do campo de opções (*Option data*) “USER.usuario”.

Supondo que uma pessoa má intencionada tenha capturado estes pacotes. Com esta informação, basta que se descubra a senha para que ela inicie uma conexão remota com o servidor em questão e aja no que ela deseje fazer, como, por exemplo, conseguir informações do sistema ou tentar ganhar um terminal *root* explorando outras vulnerabilidades locais.



**Figura 53 - Pacotes de uma conexão Telnet, usuário capturado**

Note que utilizamos um filtro de exibição na Figura 53: “*not dns and not udp*”. Isto foi realizado para retirar pacotes destes dois protocolos da tela, facilitando assim a análise dos pacotes de interesse.

O próximo pacote a ser analisado é o de número 17, enviado do servidor Telnet para o cliente. Na Figura 54 é apresentada a decodificação do campo de dados telnet. É neste pacote que o servidor envia a *string* “password:” ao cliente, indicando que o servidor ficará esperando o envio de dados referente à senha do usuário. Nesta figura também está em destaque a camada de transporte TCP (*Transmission Control Protocol*) indicando a porta padrão número 23 para o servidor e a porta 39180 para o cliente. O pacote 18 é a confirmação (ACK) de recebimento de dados TCP realizada pelo cliente.

| No.   | Time     | Source         | Destination    | Protocol | Info                 |
|---|----------|----------------|----------------|----------|----------------------|
| 17  | 0.087247 | 10.255.255.101 | 10.255.255.100 | TELNET   | Telnet Data ...      |
| 18  | 0.129002 | 10.255.255.100 | 10.255.255.101 | TCP      | 39180 > telnet [ACK] |
| Frame 17 (76 bytes on wire, 76 bytes captured)  |          |                |                |          |                      |
| Ethernet II, Src: CadmusCo_64:cf:2d (08:00:27:64:cf:2d), Dst: HonHaiPr_94:c3:5a (00:23:4d:94:c3:5a)       |          |                |                |          |                      |
| Internet Protocol, Src: 10.255.255.101 (10.255.255.101), Dst: 10.255.255.100 (10.255.255.100)             |          |                |                |          |                      |
| Transmission Control Protocol, Src Port: telnet (23), Dst Port: 39180 (39180), Seq: 58, Ack: 133, Len: 10 |          |                |                |          |                      |
| Telnet  |          |                |                |          |                      |
| Data: Password:   |          |                |                |          |                      |

**Figura 54 - Solicitação de senha pelo servidor Telnet**

O pacote 19 é o primeiro pacote de dados enviado pelo cliente após a solicitação de senha realizada pelo servidor. Vemos claramente na Figura 55 que nele é enviado o primeiro caractere da senha, a letra “e”. Na Figura 56 é mostrada a sequência de pacotes e dados Telnet seguintes. Juntando todos os pacotes vemos que fica evidente a senha do usuário em questão, destacada no quadro marcado.

| No.  | Time     | Source         | Destination    | Protocol | Info                 |
|--|----------|----------------|----------------|----------|----------------------|
| 18   | 0.129002 | 10.255.255.100 | 10.255.255.101 | TCP      | 39180 > telnet [ACK] |
| 19   | 1.342243 | 10.255.255.100 | 10.255.255.101 | TELNET   | Telnet Data ...      |
| Frame 19 (76 bytes on wire, 76 bytes captured)   |          |                |                |          |                      |
| Ethernet II, Src: HonHaiPr_94:c3:5a (00:23:4d:94:c3:5a), Dst: CadmusCo_64:cf:2d (08:00:27:64:cf:2d)      |          |                |                |          |                      |
| Internet Protocol, Src: 10.255.255.100 (10.255.255.100), Dst: 10.255.255.101 (10.255.255.101)            |          |                |                |          |                      |
| Transmission Control Protocol, Src Port: 39180 (39180), Dst Port: telnet (23), Seq: 133, Ack: 68, Len: 1 |          |                |                |          |                      |
| Telnet   |          |                |                |          |                      |
| Data: e  |          |                |                |          |                      |

**Figura 55 – Primeiro pacote da senha do usuário Telnet**

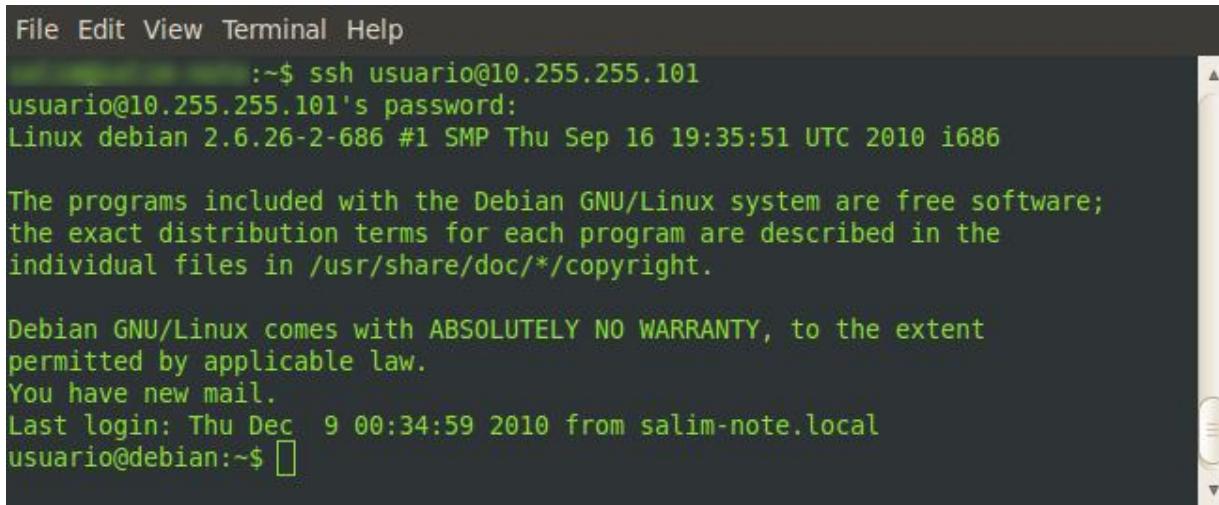
Desta forma concluímos a parte de conexão Telnet, ficando evidente sua fragilidade no quesito segurança e confidencialidade de informações. Partiremos a seguir para uma conexão SSH com o mesmo usuário e senha a fim de mostrar sua maior robustez nestes mesmos quesitos.

| No. | Time     | Source         | Destination    | Protocol | Info            |
|-----|----------|----------------|----------------|----------|-----------------|
| 19  | 1.342243 | 10.255.255.100 | 10.255.255.101 | TELNET   | Telnet Data ... |
| -   | Telnet   |                |                |          |                 |
|     | Data: e  |                |                |          |                 |
| 21  | 1.546572 | 10.255.255.100 | 10.255.255.101 | TELNET   | Telnet Data ... |
| -   | Telnet   |                |                |          |                 |
|     | Data: s  |                |                |          |                 |
| 25  | 1.701115 | 10.255.255.100 | 10.255.255.101 | TELNET   | Telnet Data ... |
| -   | Telnet   |                |                |          |                 |
|     | Data: a  |                |                |          |                 |
| 27  | 1.885235 | 10.255.255.100 | 10.255.255.101 | TELNET   | Telnet Data ... |
| -   | Telnet   |                |                |          |                 |
|     | Data: b  |                |                |          |                 |
| 29  | 2.289115 | 10.255.255.100 | 10.255.255.101 | TELNET   | Telnet Data ... |
| -   | Telnet   |                |                |          |                 |
|     | Data: 1  |                |                |          |                 |
| 31  | 2.449532 | 10.255.255.100 | 10.255.255.101 | TELNET   | Telnet Data ... |
| -   | Telnet   |                |                |          |                 |
|     | Data: 2  |                |                |          |                 |
| 33  | 2.589201 | 10.255.255.100 | 10.255.255.101 | TELNET   | Telnet Data ... |
| -   | Telnet   |                |                |          |                 |
|     | Data: 3  |                |                |          |                 |

Figura 56 – Em destaque a senha do usuário “usuario” para conexão Telnet

## SSH

Como realizado anteriormente, iniciaremos apresentando, na Figura 57, uma imagem do terminal onde a conexão SSH foi realizada com sucesso.



```

File Edit View Terminal Help
:~$ ssh usuario@10.255.255.101
usuario@10.255.255.101's password:
Linux debian 2.6.26-2-686 #1 SMP Thu Sep 16 19:35:51 UTC 2010 i686

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
You have new mail.
Last login: Thu Dec  9 00:34:59 2010 from salim-note.local
usuario@debian:~$ 
```

Figura 57 - Terminal onde foi estabelecida uma conexão SSH com êxito

Apresentamos em seguida, na Figura 58 a sequência de pacotes capturada durante esta conexão ssh. Seguindo esta sequência vemos o estabelecimento da conexão tcp do cliente para o servidor SSH, pacotes 1, 2 e 3, da porta 45632 para a porta ssh, 22. Prosseguindo conseguimos ver a troca de chaves (*Key Exchange*), nos pacotes 9 a 14.

Em destaque vemos o pacote 18, sendo este um pacote SSH versão 2 de resposta encriptada (*Encrypted response*). No primeiro quadro em destaque, confirmamos o uso do TCP na porta 22 pelo servidor. Em seguida, no próximo quadro, é mostrado o pacote encriptado. Percebe-se que não conseguiremos visualizar os dados transmitidos como no Telnet, não se fazendo necessária a análise de mais pacotes aqui, pois todos terão o mesmo formato. A pessoa mal intencionada de posse destes pacotes não conseguirá descobrir a senha nem o usuário para acessar o servidor, concluindo assim nossa experiência.

| No. . | Time     | Source         | Destination    | Protocol | Info                      |
|-------|----------|----------------|----------------|----------|---------------------------|
| 1     | 0.000000 | 10.255.255.100 | 10.255.255.101 | TCP      | 45632 > ssh [SYN] Seq=0 W |
| 2     | 0.008234 | 10.255.255.101 | 10.255.255.100 | TCP      | ssh > 45632 [SYN, ACK] Se |
| 3     | 0.008269 | 10.255.255.100 | 10.255.255.101 | TCP      | 45632 > ssh [ACK] Seq=1 A |
| 4     | 0.054111 | 10.255.255.101 | 10.255.255.100 | SSH      | Server Protocol: SSH-2.0- |
| 5     | 0.054162 | 10.255.255.100 | 10.255.255.101 | TCP      | 45632 > ssh [ACK] Seq=1 A |
| 6     | 0.054273 | 10.255.255.100 | 10.255.255.101 | SSH      | Client Protocol: SSH-2.0- |
| 7     | 0.054440 | 10.255.255.100 | 10.255.255.101 | TCP      | [TCP segment of a reassem |
| 8     | 0.055433 | 10.255.255.101 | 10.255.255.100 | TCP      | ssh > 45632 [ACK] Seq=33  |
| 9     | 0.055461 | 10.255.255.100 | 10.255.255.101 | SShv2    | Client: Key Exchange Init |
| 10    | 0.055539 | 10.255.255.101 | 10.255.255.100 | TCP      | ssh > 45632 [ACK] Seq=33  |
| 11    | 0.055866 | 10.255.255.101 | 10.255.255.100 | TCP      | ssh > 45632 [ACK] Seq=33  |
| 12    | 0.062028 | 10.255.255.101 | 10.255.255.100 | TCP      | [TCP segment of a reassem |
| 13    | 0.062103 | 10.255.255.101 | 10.255.255.100 | SShv2    | Server: Key Exchange Init |
| 14    | 0.062117 | 10.255.255.100 | 10.255.255.101 | TCP      | 45632 > ssh [ACK] Seq=832 |
| 15    | 0.062249 | 10.255.255.100 | 10.255.255.101 | SShv2    | Client: Diffie-Hellman GE |
| 16    | 0.068877 | 10.255.255.101 | 10.255.255.100 | SShv2    | Server: Diffie-Hellman Ke |
| 17    | 0.071124 | 10.255.255.100 | 10.255.255.101 | SShv2    | Client: Diffie-Hellman GE |
| 18    | 0.093398 | 10.255.255.101 | 10.255.255.100 | SShv2    | Encrypted response packet |
| 19    | 0.093524 | 10.255.255.101 | 10.255.255.100 | SShv2    | Encrypted response packet |
| 20    | 0.093537 | 10.255.255.100 | 10.255.255.101 | TCP      | 45632 > ssh [ACK] Seq=100 |
| 21    | 0.096443 | 10.255.255.100 | 10.255.255.101 | SShv2    | Encrypted request packet  |
| 22    | 0.125542 | 10.255.255.101 | 10.255.255.100 | TCP      | ssh > 45632 [ACK] Seq=169 |

```
+ Frame 18 (590 bytes on wire, 590 bytes captured)
+ Ethernet II, Src: CadmusCo_64:cf:2d (08:00:27:64:cf:2d), Dst: HonHaiPr_94:c3:5a (00:23:4d:94:c3:5a)
+ Internet Protocol, Src: 10.255.255.101 (10.255.255.101), Dst: 10.255.255.100 (10.255.255.100)
+ Transmission Control Protocol, Src Port: ssh (22), Dst Port: 45632 (45632), Seq: 969, Ack: 1000, Len: 524
- SSH Protocol
  - SSH Version 2
    Encrypted Packet: 000002BC0A2100000115000000077373682D727361000000...
```

**Figura 58 – Pacotes SS Hv2 com dados criptografados capturados**

# UNIDADE 26

*Objetivo: Entender a importância do protocolo para transferência de arquivos FTP e suas principais características.*

## Serviços e Protocolos de Transferência de Arquivos - FTP (File Transfer Protocol)

O **FTP** é um protocolo cliente-servidor utilizado para **transferir arquivos** entre computadores em uma rede baseada em TCP/IP, como a Internet. Esse protocolo foi publicado em 1971 na RFC 114 (<http://tools.ietf.org/html/rfc114>) e após substituições, está descrito no RFC 959 (<http://tools.ietf.org/html/rfc959>) de 1985. Mesmo sendo antigo está entre os meios mais populares para transferência de arquivos. Diversos servidores utilizam o FTP para disponibilizar arquivos (conteúdo, documentos, programas, etc) tanto na Internet quanto em redes locais. O protocolo FTP é independente de hardware e de sistema operacional.

Os aplicativos FTP permitem que o usuário navegue pela estrutura de diretório remoto como se estivesse acessando o sistema de arquivos de sua máquina, podendo realizar ações como criação, alteração e remoção de arquivos e pastas (de acordo com as permissões concedidas pelo administrador). A facilidade de acesso aos dados e às pastas, a possibilidade de transferência de arquivos grandes e a confiabilidade fornecida pelo TCP contribuem fortemente para a alta popularidade dos aplicativos FTP.



**Figura 59 – Transferência de Arquivos com o FTP**

A Figura 59 acima representa uma típica sessão FTP em que um usuário sentado em sua máquina, pretende transferir arquivos de ou para o servidor remoto, ou seja, realizar um upload (transferência de arquivo para um servidor) ou download (transferência de arquivo de um servidor para máquina local). Para acessar o servidor, o usuário primeiramente informa o endereço do mesmo (ex: [ftp.empresadocom.br](http://ftp.empresadocom.br)) e a aplicação cliente FTP presente em seu computador estabelece uma conexão TCP com o processo servidor FTP que está no computador remoto. Em seguida, caso não seja um usuário anônimo, ele informará o login e senha (enviados pela conexão TCP como parte dos comandos FTP) para acessar a conta remota. Após autenticação do login e senha, o usuário poderá realizar a transferência de arquivos armazenados no sistema de arquivos local para o sistema de arquivos remoto (ou vice-versa), de acordo com as permissões de acesso que possuir.

O FTP pode ser configurado para restringir os privilégios de acesso (leitura, escrita, criação ou remoção de arquivos) de determinados usuários em determinadas pastas do servidor. Dessa forma, o administrador de um servidor FTP tem pleno controle sobre as ações e pastas que cada usuário pode acessar. Por exemplo, em uma instituição de ensino o administrador poderia criar uma pasta destinada aos materiais do professor de forma que ele pudesse criar e remover arquivos enquanto os alunos só poderiam ler os arquivos disponibilizados pelo tutor, evitando que os conteúdos dos mesmos fossem alterados pelos alunos. O administrador pode inclusive liberar ou negar acesso de

usuários anônimos, que não precisam de login e senha para se conectar e transferir arquivos. De um modo geral os administradores costumam conceder aos usuários anônimos somente permissão de leitura em pastas públicas. Entretanto, recomenda-se bloquear os usuários anônimos caso sua participação não seja necessária.

## Conexões FTP

Uma característica do protocolo FTP é o estabelecimento de dois tipos de conexões TCP paralelas entre o cliente e servidor, uma para controle e outra para transferência de dados.

- A **conexão de controle**: é utilizada para enviar informações e comandos. Nenhum arquivo é transferido por essa conexão. As informações como login e senha, e os comandos para “baixar” arquivos, inserir arquivos, listar diretórios, dentre outros, são passados por essa conexão que tem características “administrativas”, e por isso recebe o nome “de controle”. A conexão de controle é estabelecida quando o cliente solicita o início da sessão FTP, e é mantida durante todo o tempo da sessão
- A **conexão de dados**: é utilizada para transferência do arquivo, sendo estabelecida a cada solicitação de transferência. O FTP envia exatamente um arquivo pela conexão de dados e em seguida fecha essa conexão, ou seja, as conexões de dados **não são persistentes**. Se durante a mesma sessão outros arquivos forem transferidos pelo usuário, uma nova conexão de dados para cada arquivo será aberta.

Uma característica adicional da **conexão de controle** é que ela utiliza ou se baseia no protocolo Telnet de acordo com as duas formas especificadas na RFC 959 (enquanto a **conexão de dados** é uma conexão TCP que não se baseia em nenhum protocolo adicional). Segundo essa RFC, a aplicação que implementa o protocolo FTP pode implementar diretamente em seus procedimentos as regras do protocolo Telnet ou então pode utilizar-se do módulo Telnet existente no sistema. A segunda opção apresenta as vantagens de facilidade de implementação, compartilhamento de código e programação modular, enquanto a primeira apresenta as vantagens de independência e eficiência. Na prática, a primeira opção não necessita de muito código visto que o FTP baseia-se em poucas definições do protocolo Telnet.

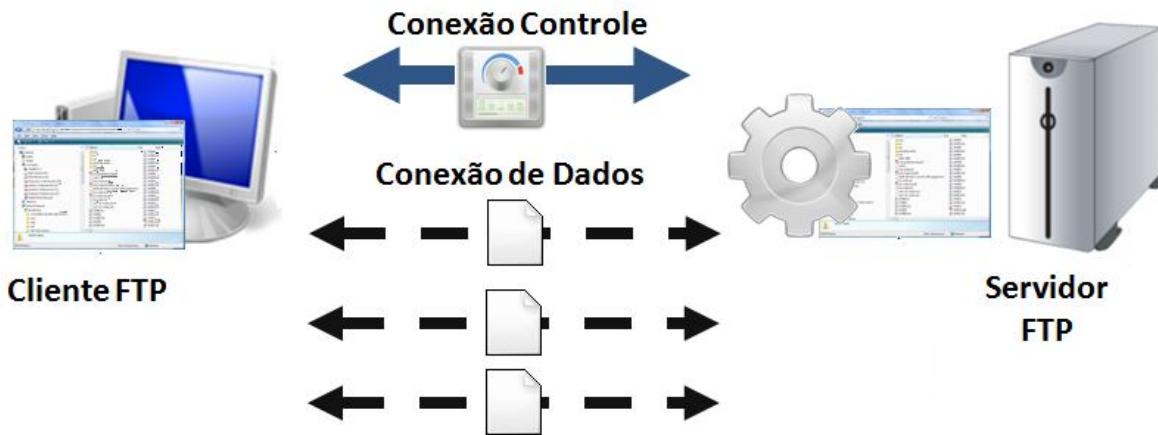


Figura 60 – Conexão de Controle e Conexão de Dados do FTP entre Cliente e Servidor

Como o FTP utiliza uma conexão exclusiva para troca de informações (a **conexão de controle**), dizemos que suas informações de controle são enviadas “**fora da banda**”. Essa denominação é diferente da do protocolo HTTP que transfere o cabeçalho, requisição e resposta pela mesma conexão TCP utilizada para o arquivo transferido, ou seja, envia suas informações de controle “**dentro da banda**”.

Uma transferência de arquivo sendo realizada pela **conexão de dados** pode ser abortada através do envio de um comando de interrupção enviado pela **conexão de controle**.

### Modos Ativo e Passivo

Um servidor FTP pode funcionar no modo ativo ou passivo, sendo a diferença entre eles a parte que origina a conexão de dados. Independentemente, em ambos os casos existem as conexões de controle e as conexões de dados. Primeiramente, vamos entender como funciona a conexão de controle, para então diferenciar a conexão de dados do servidor no modo ativo e passivo. Quando o cliente inicia uma sessão FTP com o servidor, uma **conexão TCP de controle** é estabelecida com a porta 21 do servidor. Esse procedimento é feito tanto no servidor modo ativo como passivo.

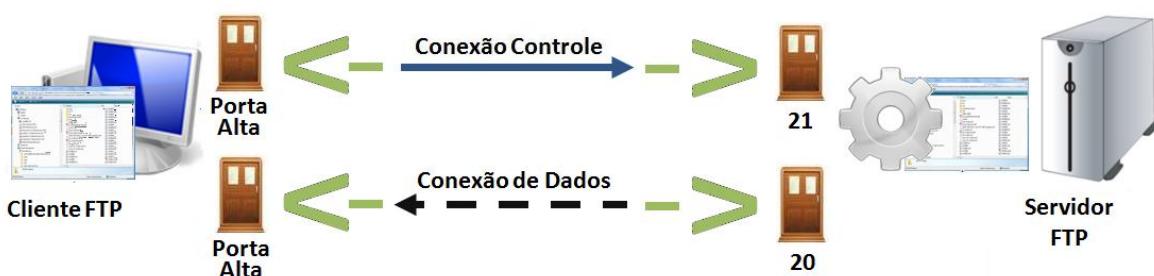
Em relação à conexão de dados:

- No modo **ativo** quando é solicitado o envio de um arquivo, o cliente envia ao servidor seu endereço IP e porta de escuta, e o servidor inicia uma conexão de

dados TCP que se origina na porta 20 (do servidor). Ou seja, no modo ativo o servidor que inicia a conexão de dados. Esse modo pode causar bloqueio pelo firewall da rede porque uma conexão iniciada por uma máquina fora da rede é destinada a uma máquina na rede interna.

- Para vencer essa restrição, no modo **passivo** a conexão é iniciada pelo cliente e o servidor nunca inicia uma conexão TCP. Nesse modo, o cliente envia um comando ao servidor e recebe como resposta o endereço IP e porta de escuta, para então iniciar a conexão para o servidor.

### Ftp Ativo



### Ftp Passivo

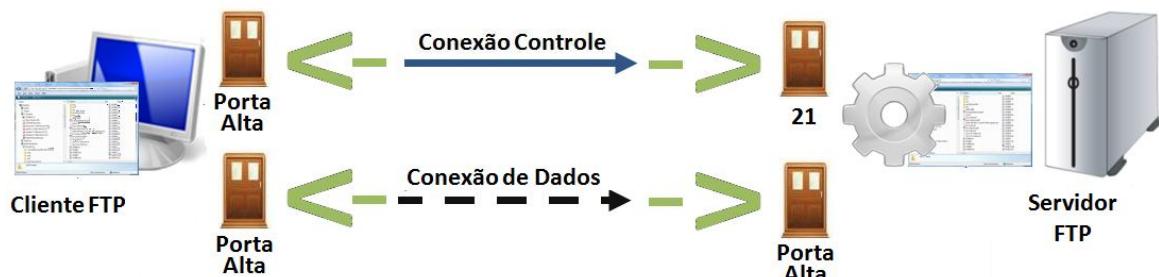


Figura 61 – Diferenças entre modo Ativo e modo Passivo no FTP



## Estudo Complementar

Acesse através do programa explorador de pastas do seu sistema operacional (ex: Windows Explorer) ou então de um navegador web (Firefox, Internet Explorer, etc) o FTP público da IETF disponível em <ftp://ftp.ietf.org/>. Se você utilizar através do explorador de pastas, poderá reparar que a visualização das pastas e arquivos dá impressão que os mesmos estão no seu computador (exceto pela velocidade de consulta e transferência). Como era de se esperar, você pode consultar e baixar arquivos, porém não pode excluir e nem inserir arquivos. Dentre outras pastas disponibilizadas nesse FTP, existem os drafts (rascunhos) e os RFCs citados neste Módulo. (rascunhos) e os RFCs citados neste Módulo.



# UNIDADE 27

*Objetivo: Entender a importância do protocolo para transferência de arquivos FTP e suas principais características*

## FTP (Continuação)

### Tipos de Dados Transferidos e Modos de Transferência

Na transferência realizada pelo FTP, quatro tipos de dados podem ser utilizados:

- ASCII: tipo *default*. Deve ser aceito por todas as implementações que se baseiam no protocolo FTP. É utilizado para a transferência de texto, sendo inapropriado para arquivos que contenham dados que não sejam desse tipo.
- EBCDIC: destinado para transferência eficiente de texto entre hosts utilizando o conjunto de caracteres EBCDIC. Esse tipo é semelhante ao tipo ASCII.
- Binário (ou tipo Imagem): destinado à transferência de dados binários sem nenhuma conversão. Os arquivos são enviados byte por byte e a máquina destinatária armazena esse fluxo na medida em que os recebe. A RFC 959 recomenda que todas as implementações FTP aceitem esse tipo.
- Local: destinado para transferência de dados em um modo proprietário entre computadores com configurações idênticas.

Como pode ser observado, o FTP fornece poucas opções de representações de dados. Transformações desejadas diferentes das oferecidas devem ser realizadas diretamente pelo cliente.

Em relação ao modo de transferência, existem três possibilidades:

- Fluxo: neste modo os dados são enviados em um fluxo contínuo de forma que o FTP não realiza nenhum tipo de processamento, que é completamente realizado pelo TCP.
- Blocos: Antes de passar os dados para o TCP, o FTP os divide em vários blocos: bloco cabeçalho, contagem de bytes e campo de dados.
- Compressão: os dados são comprimidos pela utilização de algoritmos simples.

## Comandos e Respostas FTP

Os **comandos** do cliente para o servidor, e as respectivas **respostas** enviadas pelo servidor são enviados através da conexão de controle em **modo texto (ASCII)**. Os comandos são compostos por caracteres ASCII sendo que alguns possuem argumentos opcionais. Os comandos mais comuns são descritos a seguir (para uma lista completa consulte a RFC 959):

- **USER *username***: envia a identificação do usuário (parâmetro *username*) ao servidor.
- **PASS *password***: envia a senha do usuário para o servidor.
- **RETR *filename***: obtém um arquivo do diretório atual do servidor.
- **STOR *filename***: insere um arquivo no diretório atual do hospedeiro remoto.
- **LIST**: solicita ao servidor a lista com todos os arquivos existentes no atual diretório remoto. A lista dos arquivos é enviada através de uma nova conexão de dados (e não pela conexão de controle).

A comunicação entre o cliente e servidor é semelhante a um diálogo. Cada **comando** enviado é seguido de uma **resposta** enviada pelo servidor. As respostas são números de três dígitos que podem possuir uma mensagem opcional após o número. Os três dígitos são utilizados para interpretação das aplicações, enquanto o texto é para compreensão do usuário. O texto da mensagem opcional pode variar de acordo com a aplicação servidora, mas independentemente, o significado dos 3 dígitos são padronizados. Se você reparar, a estrutura das respostas se assemelha à mensagem de resposta HTTP

(apresentado na Unidade 6). Esse fato não é uma coincidência, os criadores do HTTP incluíram intencionalmente essa similaridade. A seguir, algumas respostas típicas são apresentadas como critério de exemplificação (lista completa disponível na RFC 959):

- *200 (ou 200 ok)*: indica que o último comando foi executado com sucesso.
- *220 Service ready for new user* (Serviço pronto para novo usuário)
- *331 Username Ok, password required* (Nome do usuário ok, senha requisitada).
- *425 Can't open data connection* (Não é possível abrir a conexão de dados).
- *452 Insufficient storage space in system* (Espaço de armazenamento insuficiente no sistema).

## Segurança

Assim como ocorre com o Telnet, no FTP o tráfego na rede não é encriptado, permitindo que as informações transmitidas (login, senha e dados) possam ser facilmente visualizadas em caso de captura de pacotes (*sniffing*) na rede. Na verdade, essa fraqueza de segurança é comum em vários protocolos da internet, como por exemplo, SMTP, POP e IMAP, que foram especificados antes da criação de mecanismos de encriptação como o TLS ou SSL. Uma alternativa mais segura é a utilização de protocolos como FTPS e SFTP, apresentados na Unidade 30. Algumas falhas de segurança do FTP e soluções para minimizar os riscos são apresentadas no RFC 2577 (<http://tools.ietf.org/html/rfc2577>), a saber: ataques de força bruta, captura de pacotes (*sniffing*), roubo de portas, ataques bounce, descoberta de logins e senhas, roubo de porta e ataques spoof.



## Dica

Caso você precise compartilhar um arquivo grande que não possa ser enviado por e-mail e não possua acesso ou possibilidade de montar um servidor FTP, você pode utilizar alguns serviços proprietários com versões gratuitas na Internet. Dentre eles, citaremos o DropBox ([www.dropbox.com/](http://www.dropbox.com/)) e o MegaUpload ([www.megaupload.com/](http://www.megaupload.com/)). A versão gratuita do DropBox oferece 2GB de armazenamento online, permitindo compartilhar arquivos entre computadores com Linux, Mac e Windows. Você pode utilizar a ferramenta online ou baixar um programa para o seu computador, que compartilhará uma pasta criada e permitirá que você arraste/crie diretórios e arquivos. Caso esses arquivos sejam alterados, eles são atualizados no DropBox automaticamente. Já a associação grátis do MegaUpload permite armazenamento online de 200 GB.



# UNIDADE 28

*Objetivo: Apresentar, utilizando o Wireshark, as mensagens trocadas entre cliente e servidor FTP, mostrando a falta de confidencialidade quando não se utiliza a camada SSL/TLS (FTPS).*

## **Analizador de Pacotes de Rede – FTP**

### **Introdução**

O protocolo FTP estabelece dois tipos de conexão: Conexão de Controle e Conexão de Dados. Para estas conexões o servidor FTP utiliza a porta padrão TCP 21 para conexão de controle, trocando com o cliente as informações de autenticação, usuário e senha, de requisição de transferência de arquivos, entre outras. E a porta TCP 20 é destinada as conexões de dados (no modo ativo), para recebimento e envio de arquivos.

Porém no FTP não há nenhum mecanismo de encriptação, tornando a autenticação no servidor vulnerável a um programa de captura de tráfego na rede, como demonstraremos aqui utilizando o Wireshark.

A demonstração consistirá em conectar-se a um servidor ftp e transferir um arquivo texto para ser armazenado por este servidor.

A descrição dos parâmetros do serviço:

Rede Local: 10.255.255.0/24

Servidor FTP: 10.255.255.101

Máquina cliente: 10.255.255.100

Usuário FTP: “usuario”

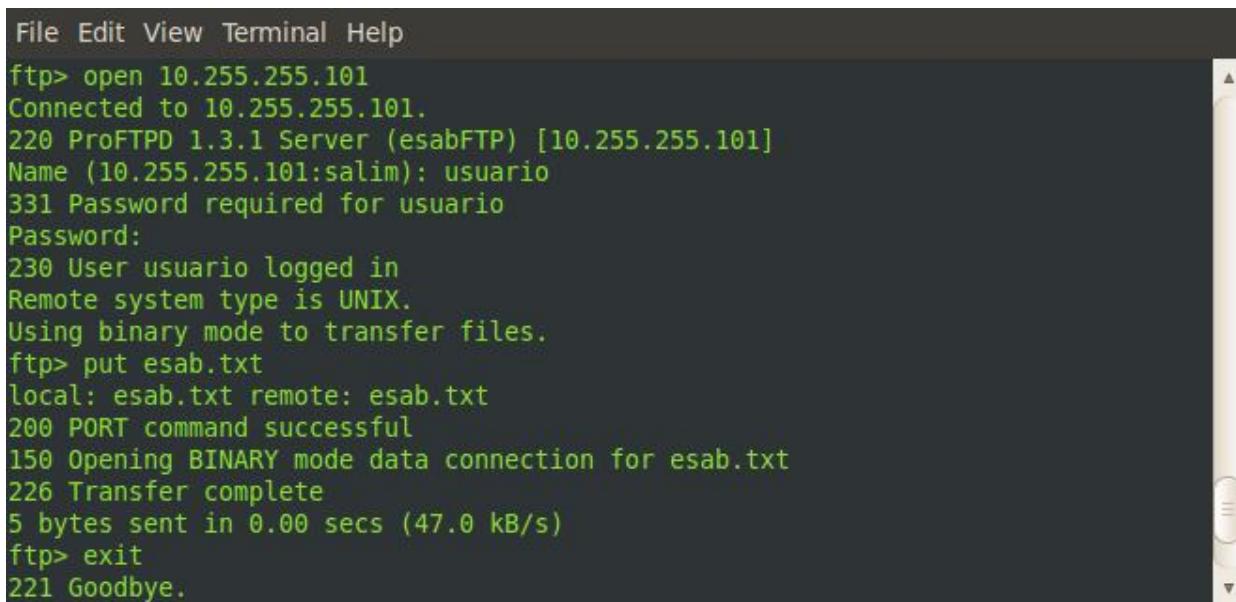
Senha: “esab123”

Nome do arquivo transferido: esab.txt

Conteúdo do arquivo: “esab”

O Wireshark será utilizado na máquina cliente FTP por quesitos de praticidade.

Na Figura 62 é mostrado um terminal cliente FTP apresentando todas as informações e comandos utilizados para transferência do arquivo “esab.txt”, desde a autenticação. Nele podemos ver a solicitação do cliente para a conexão com o servidor, “open 10.255.255.101”, e a de requisição de upload de arquivo, “put esab.txt” ( o comando “put” foi criado por muitos aplicativos de linha de comando FTP para implementar o comando “STOR” especificado no RFC 959 do FTP) . Também podemos ver a confirmação de conclusão de transferência, mensagem “226 Transfer complete” e de requisição de término de conexão, “exit”. Utilizamos um terminal cliente ao invés de uma interface gráfica para demonstrar alguns comandos e mensagens de resposta FTP.



```

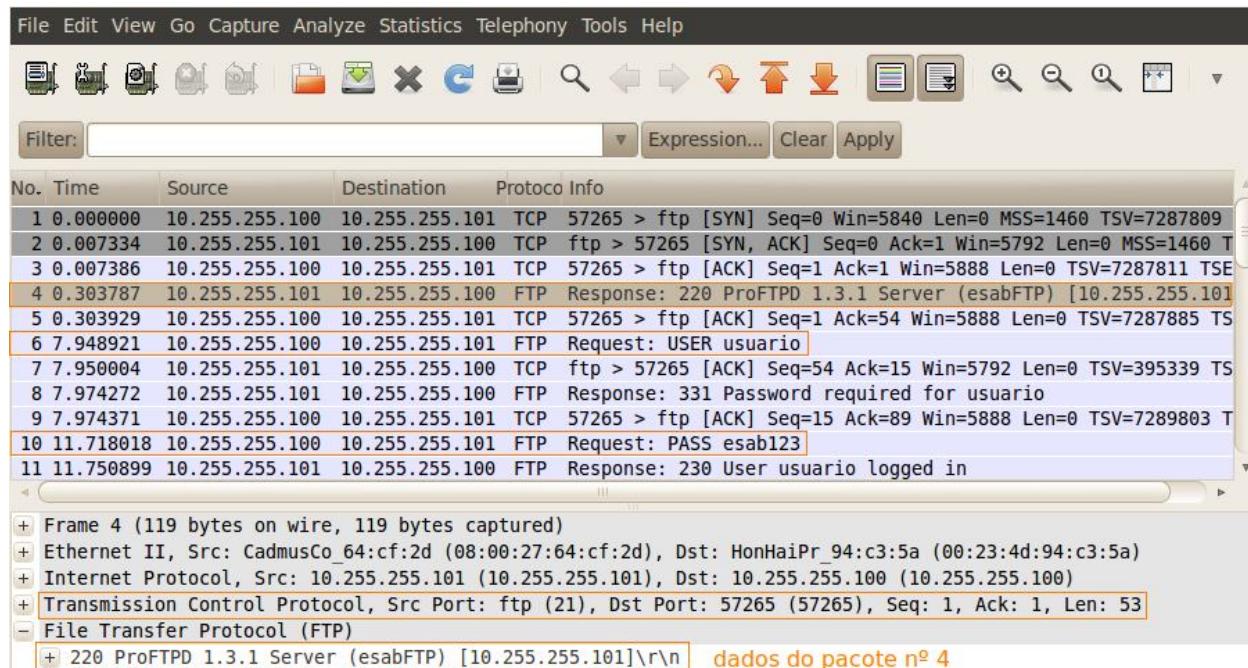
File Edit View Terminal Help
ftp> open 10.255.255.101
Connected to 10.255.255.101.
220 ProFTPD 1.3.1 Server (esabFTP) [10.255.255.101]
Name (10.255.255.101:salim): usuario
331 Password required for usuario
Password:
230 User usuario logged in
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> put esab.txt
local: esab.txt remote: esab.txt
200 PORT command successful
150 Opening BINARY mode data connection for esab.txt
226 Transfer complete
5 bytes sent in 0.00 secs (47.0 kB/s)
ftp> exit
221 Goodbye.
    
```

Figura 62 - Terminal cliente FTP

Agora verificaremos como são os pacotes trocados entre o servidor e cliente FTP para a execução dos comandos mostrados na Figura 62. Os primeiros pacotes capturados pelo Wireshark referem-se à abertura da conexão (*three-way handshake* do TCP) com o servidor FTP e a autenticação neste. Eles são apresentados na Figura 63**Erro! Fonte de referência não encontrada..**

No pacote de número 4 podemos ver a mensagem de código 220, que confirma a conexão com o servidor. Repare que o texto mostrado no terminal cliente é transferido neste pacote. Outra informação importante mostrada na parte de decodificação de

pacotes do Wireshark é o uso do protocolo TCP, com porta de origem (“Src Port”), 21. O cliente utiliza uma porta aleatória.

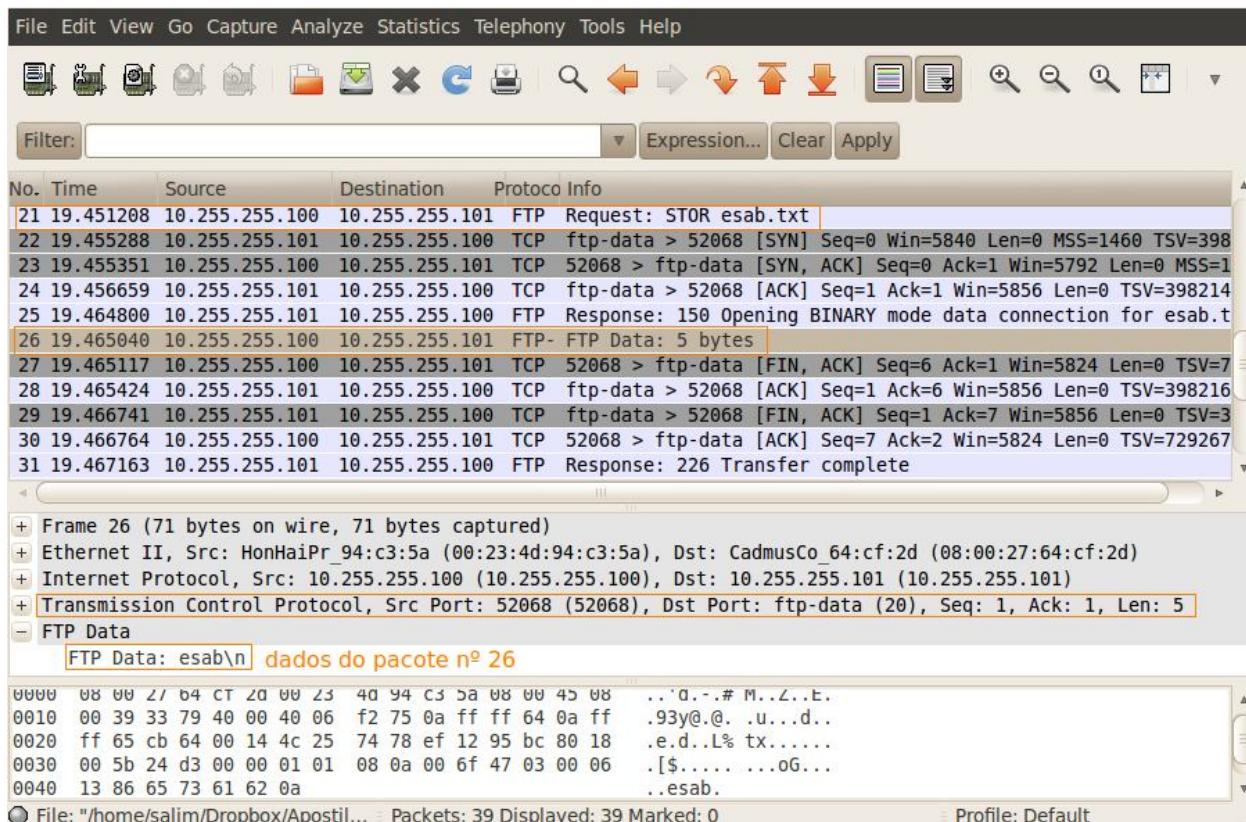


**Figura 63 - Pacotes de conexão e autenticação no servidor FTP**

As informações enviadas pelo cliente para autenticação no servidor são apresentadas nos pacotes 6 e 10, na Figura 63. Note que os dados ficam totalmente aparentes e fáceis de serem visualizados. No pacote 6 o cliente envia a informação “Request: USER usuario” e no 10 fica clara a senha utilizada para este usuário, “Request: PASS esab123”. De posse destas informações capturadas na rede é possível baixar e enviar arquivos para o servidor.

Agora apresentaremos os pacotes da conexão de dados FTP na Figura 64. Note que no pacote número 21 o cliente, IP 10.255.255.100, solicita o envio do “arquivo esab.txt” com a mensagem “Request: STOR esab.txt”. Conforme vimos na Unidade do FTP, esse comando é passado através da conexão de controle. Em seguida, nos pacotes 22, 23 e 24, é mostrado o estabelecimento de uma nova conexão de dados pelo servidor FTP. Conforme estudamos, podemos perceber que o modo “ativo” está sendo utilizado neste exemplo, visto que o servidor inicia a conexão para o cliente após a solicitação de transferência do arquivo. A string “ftp-data > 52068 [SYN]” indica que o pacote 22 é o

primeiro do *three-way handshake* do TCP, e que a conexão será feita através da porta ftp-data (20) no servidor para a porta 52068 no cliente. O pacote 25 apresenta a utilização do modo de transferência binária.



**Figura 64 - Pacotes referentes à transferência do arquivo esab.txt**

Outro pacote que demonstra a exposição de dados do protocolo FTP é o de número 26, apresentado na Figura 64. É possível reparar que os dados transferidos somam 5 bytes, e no campo de decodificação deste pacote no Wireshark vemos que o conteúdo do arquivo "esab.txt", é mostrado: "FTP Data: esab\n".

Desta forma, fica clara a não confidencialidade do protocolo FTP podendo acarretar vulnerabilidade de acesso e privacidade no serviço. Isto pode ser corrigido utilizando a camada SSL/TLS acima do protocolo de transporte TCP, caracterizando um novo serviço, o FTPS. Nele os pacotes serão encriptados da mesma forma que no protocolo HTTPS, apresentado na Unidade 8.

# UNIDADE 29

*Objetivo: Conhecer o Protocolo TFTP em conjunto com suas vantagens e desvantagens*

## **TFTP (Trivial File Transfer Protocol)**

O TFTP é um protocolo de transferência de arquivos simples, por isso no seu nome possui a letra T de “Trivial”. Sua primeira definição foi apresentada em 1980, 9 anos após a apresentação da primeira definição do FTP. A revisão 2 está especificada no RFC 1350 (<http://tools.ietf.org/html/rfc1350>). O TFTP foi projetado para ser pequeno e fácil de implementar e por isso não possui a maioria das funcionalidades do FTP. O TFTP apenas lê e escreve arquivos (ou mensagens) de/para servidores remotos. Não permite listar diretórios e não fornece suporte para autenticação de usuário (até o presente momento). Além de não ter autenticação via login e senha, também não fornece suporte à encriptação das mensagens (assim como o FTP). Portanto, devido à sua falta de segurança é perigoso utilizá-lo em redes abertas (como a Internet), e esse protocolo geralmente é utilizado apenas em redes locais privadas.

Mas se o TFTP possui menos funcionalidades do que o FTP, não é recomendado que seja usado em redes abertas, não tem autenticação de usuário e não possui mecanismos de segurança, para que seria utilizado?

Bem, devido a sua simplicidade, o TFTP pode ser implementado utilizando uma quantidade pequena de memória, sendo útil para dar *boot* em equipamentos que não possuem dispositivos de armazenamento, como por exemplo, roteadores. Também é utilizado em alguns estágios iniciais de instalação de sistemas via rede (como por exemplo, Red Hat Kickstart e Serviço de Instalação Remota do Windows NT).

O TFTP tipicamente utiliza o UDP (porta 69), mas pode ser implementado para utilizar outros protocolos de transporte.

## Funcionamento da Transferência de Arquivo

A transferência TFTP começa com uma solicitação para ler ou escrever um arquivo em outro equipamento. Nessa mesma solicitação é realizada uma requisição para conexão, que caso aprovada pelo servidor, é aberta para a transferência. A solicitação para transferência de dados é iniciada na porta 69, mas as portas para transferência dos dados em si, são definidas pelo remetente e destinatário durante a inicialização da conexão.

Os arquivos são enviados em blocos fixos de 512 bytes, sendo que, apenas o pacote que indica o término da transmissão é menor do que 512. Cada **pacote de dados** deve ter o recebimento confirmado por um **pacote de confirmação** (pacote de “*acknowledgment*” – “ACK”) antes do envio do próximo pacote. Cada **pacote de dados** possui um número do bloco para identificação, que começa do número 1 e é incrementado consecutivamente. Em geral, o **pacote de confirmação** contém o número do bloco do **pacote de dados** cujo recebimento está sendo confirmado.

Em relação à primeira solicitação, caso ela seja para escrita e o servidor aceite, a resposta será um pacote de confirmação. Caso seja para leitura, a resposta será o primeiro pacote de dados.

Se um pacote se perder na rede, após o tempo de espera necessário, o destinatário desse pacote vai retransmitir o seu último pacote (seja ele um pacote de dados ou de confirmação), indicando ao remetente que ele deve retransmitir o seu último pacote. Com esse mecanismo, o remetente precisa manter apenas o seu último pacote para caso seja necessária uma retransmissão, visto que o pacote de confirmação garante que os pacotes anteriores foram recebidos. Note que as duas máquinas que participam da transferência podem ser consideradas destinatárias e remetentes, visto que uma envia os dados e recebe confirmação, e a outra envia confirmação e recebe os dados.

A forma de transferência de “enviar-esperar confirmação” faz com que apenas um pacote novo (seja ele de dados ou confirmação) esteja em trânsito por vez, ocasionando uma baixa vazão e alta latência. Algumas técnicas podem ser adicionadas para melhorar consideravelmente esses pontos, como por exemplo, a janela de 8 pacotes introduzida no Windows 2008.

A Figura 65 apresenta um exemplo de transferência de arquivos com TFTP.

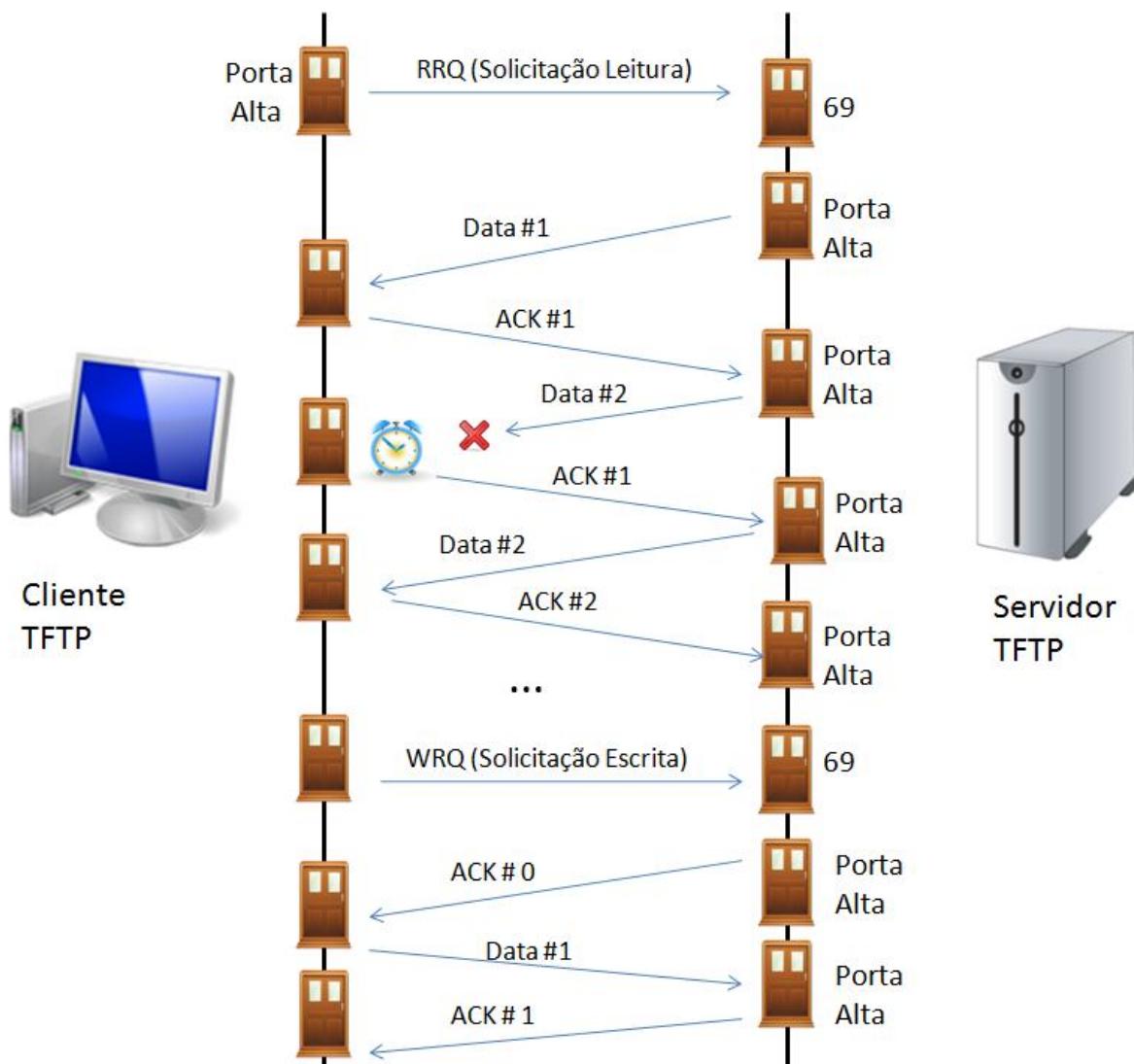


Figura 65 – Exemplo de Troca de Pacotes TFTP

A ocorrência de um erro é sinalizada por um **pacote de erro**, que não é seguido por um pacote de confirmação e nem gera uma retransmissão. Os erros são causados por três tipos de eventos:

- Impossibilidade de satisfazer a requisição (ex: arquivo não encontrado).
- Recebimento de um arquivo que não pode ser entendido (ex: pacote incorretamente formado).
- Perda de acesso ao recurso necessário (ex: disco cheio).

A maioria dos erros ocasiona a finalização da conexão. Apenas quando a porta de origem de um pacote recebido está incorreta, um pacote de erro é enviado ao remetente, e a conexão não é finalizada.

Conforme citado anteriormente, o TFTP oferece cinco tipos de pacotes:

1. *Read request (RRQ)* – Requisição de Leitura
2. *Write request (WRQ)* – Requisição de Escrita
3. *Acknowledgment (ACK)* – Confirmação
4. *Data (DATA)* – Dados
5. *Error (ERROR)* - Erro

O TFTP pode transferir os dados em três formatos:

- NetASCII – Correspondente ao tipo ASCII do FTP
- Octeto – Correspondente ao tipo Binário do FTP
- *Mail* – Envio de caracteres NetASCII ao invés de enviar o arquivo. Esse modo se tornou obsoleto na RFC 1350.

Modos adicionais podem ser definidos entre as duas máquinas conectadas para a transferência.

# UNIDADE 30

*Objetivo: Conhecer alternativas para transferir arquivos de forma segura.*

## **Alternativas Seguras para realizar a Transferência de Arquivos**

Nesta unidade, conheceremos formas para prover segurança na transferência de arquivos, visualizando as limitações de cada alternativa. Serão apresentados:

- SCP
- SFTP
- FTPS
- FTP sobre SSH

Perceberemos que o FTPS e FTP são formas de uso do protocolo FTP. Já o SCP e SFTP utilizam serviços SSH e não possuem relação com o FTP.

### **SCP (Secure Copy)**

O **SCP** (*Secure Copy*) é um meio para transferência de arquivos de forma segura entre dois computadores. O **SCP** é baseado no **RCP**, um comando Unix para realizar cópia remota. O **RCP** é inseguro para rodar em uma rede visto que as informações enviadas não são encriptadas, e por esse motivo tem sido amplamente substituído pelo **SCP**.

O SCP cria um túnel (*Tunnelling*) através do protocolo SSH para dar suporte à encriptação e autenticação. Algumas vezes encontramos na literatura e páginas da internet o SCP sendo tratado como protocolo, porém ele pode não ser considerado um protocolo visto que é apenas uma combinação do comando RCP com o protocolo SSH, ou seja, enquanto o RCP transfere o arquivo o SSH realiza a autenticação e encriptação. Dessa forma, o SCP protege a autenticidade e confidencialidade dos dados em trânsito, dificultando a extração de informações dos pacotes de dados em caso de captura através de *sniffers*.

No SCP, para realizar um *upload* a aplicação cliente informa ao servidor os arquivos e opcionalmente atributos básicos como permissões e *timestamps* (data do arquivo no servidor). Para realizar o *download* de um diretório, o servidor passa pra o cliente seus subdiretórios e arquivos. Vale ressaltar que mesmo a conexão sendo segura, sempre vai existir risco de segurança caso você se conecte a um servidor malicioso.

O programa SCP mais utilizado é a linha de comando *scp* que acompanha a maioria das implementações SSH. Sua linha de comando é como a linha de comando *rcp*, porém de forma segura. Algumas implementações do SSH oferecem o programa *scp2*, que utiliza o SFTP (apresentado a seguir) ao invés do SCP, mas mantém a mesma linha de comando.

Como o “protocolo” SCP implementa apenas transferência de arquivos e nenhum outro comando, é raro encontrar Interfaces Gráficas (*GUI – Graphical User Interface*) visto que a implementação requer funcionalidades adicionais, como no mínimo a listagem de diretórios. Alguns programas, como o WinSCP, acabam utilizando o protocolo SFTP para fornecer suporte à interfaces gráficas. Mesmo quando rodam no modo SCP tipicamente não são “clientes SCP puros” uma vez que precisam utilizar outros meios para implementar funcionalidades adicionais. Essa questão traz a tona problemas de dependência de plataforma.

O SCP tem sido substituído pelo SFTP cujo protocolo e respectivo aplicativo oferecem ferramentas mais abrangentes para gerenciar transferência de arquivos.

### SFTP (SSH File Transfer Protocol)

O **SFTP** (SSH File Transfer Protocol) é uma alternativa segura ao FTP. O SFTP Foi projetado pela IETF (*Internet Engineering Task Force*) como uma extensão do protocolo SSH-2 para fornecer transferência segura de arquivos. Entretanto, pode ser utilizado com outros protocolos que disponibilizam um canal seguro. É importante não confundir SFTP com “FTP sobre SSH” (apresentado mais adiante nesta unidade), que são mecanismos distintos. O “*Internet Draft*” (“rascunho”) do IETF declara que mesmo sendo descrito no contexto do SSH-2, esse protocolo é geral e independente do restante das funcionalidades do protocolo SSH-2, de forma que possa ser utilizado em diferentes aplicações, como por exemplo, transferência de informações de gerenciamento em aplicações VPN.

Comparado ao SCP que apenas permite a transferência de arquivos, o protocolo SFTP disponibiliza uma quantidade de operações em arquivos remotos. A aplicação SFTP fornece a mais do que o SCP: continuação de uma transferência interrompida, listagem de diretórios e remoção remota de arquivos. Essas características tornam a implementação de uma interface gráfica do SFTP relativamente simples se comparada ao SCP. Enquanto o SCP é mais implementado para plataformas Unix, os servidores SFTP são comumente disponibilizados na maioria das plataformas.

O protocolo SFTP, por si só, não fornece autenticação e segurança, e conta com o protocolo subjacente para se encarregar dessas questões. Assim como o SCP, também transfere os atributos básicos do arquivo, como data/*timestamp*. Diferentemente do SCP, uma transferência no SFTP pode ser cancelada sem terminar a sessão.

O aplicativo cliente SFTP pode ser um programa de linha de comando ou interface gráfica interativa semelhante aos clientes tradicionais FTP. A utilização de um cliente SFTP com interface gráfica simplifica a transferência de arquivos visto que permite realizá-la apenas arrastando e soltando os arquivos entre as janelas.

O SFTP possui um conjunto de comandos similar ao FTP. Podemos deduzir que essa foi uma estratégia boa e conveniente visto que muitos usuários já estavam familiarizados com FTP. Ao contrário do FTP padrão, SFTP encripta os comandos e dados fornecendo segurança para transferência de senhas e informações confidenciais. Mesmo possuindo as características de transferir arquivos e possuir um conjunto de comandos similares, os protocolos SFTP e FTP não se relacionam, ou seja, você não consegue utilizar um cliente SFTP para se conectar a um servidor FTP e vice-versa.

A última versão produzida pelo grupo do IETF responsável pelo SFTP é de 2006 (<http://tools.ietf.org/html/draft-ietf-secsh-filexfer-13>). O SFTP não se tornou um padrão IETF e seus *drafts* foram paralisados porque alguns membros do comitê visualizaram o SFTP como um protocolo de sistema de arquivos e não um protocolo de transferência de arquivos. Mesmo não sendo um padrão IETF, o SFTP ainda é amplamente utilizado.

### **FTPS (FTP Seguro ou FTP-SSL)**

O **FTPS** (FTP Seguro ou FTP-SSL) é uma extensão do FTP que fornece suporte aos protocolos TLS(*Transport Layer Security*) e SSL(*Secure Sockets Layer*).

Conforme explicado anteriormente, o FTP utiliza uma porta dinâmica para cada conexão de dados, sendo que essa conexão é criada para cada transferência de arquivo. Por esse motivo, muitos *firewalls* foram projetados para “bisbilhotar” as mensagens de controle do FTP para identificar as conexões de dados que devem ser liberadas. Porém, se a conexão de controle for encriptada utilizando TLS/SSL, o *firewall* não poderá definir e liberar a porta TCP da conexão de dados. Portanto, em muitas redes com a proteção de um *firewall* o serviço FTP funcionará enquanto ao FTPS não. Uma forma de viabilizar a utilização de um serviço FTPS através de um *firewall* é determinando um intervalo limitado de portas e configurando o *firewall* para abrir essas portas.

### FTP sobre SSH

Utilizar FTP sobre SSH consiste em “tunelar” (*tunneling*) um FTP normal através de uma conexão SSH. A característica especificada no protocolo FTP de criar múltiplas conexões TCP (as conexões de dados) dificulta tunelar o FTP através do SSH. Na maioria das aplicações clientes SSH, o “túnel” será criado apenas para a conexão de controle, já que a sua porta é conhecida (porta 21). Na transferência de arquivo, uma nova conexão de dados será criada por fora do túnel e, portanto, não será encriptada.

Se o objetivo for apenas proteger o *username* e senha, essa abordagem funcionará visto que esses dados são passados através da conexão de controle que estará segura pelo SSH. Porém, se os arquivos também forem confidenciais, apenas algumas aplicações clientes SSH implementadas com especificidades para o protocolo FTP, servirão para o propósito. Essas aplicações devem monitorar e re-escrever o controle das conexões de dados de forma a criá-las dentro do túnel SSH.



## Atividades

Antes de iniciar sua Avaliação Online, é fundamental que você acesse sua SALA DE AULA e faça a Atividade 3 no “link” ATIVIDADES.





## Atividades

### ***Atividades dissertativas***

Acesse sua sala de aula, no link “Atividade Dissertativa” e faça o exercício proposto.

Bons Estudos!



# GLOSSÁRIO

---

Caso haja dúvidas sobre algum termo ou sigla utilizada, consulte o link Glossário em sua sala de aula, no site da ESAB.

# BIBLIOGRAFIA

---

Caso haja dúvidas sobre algum termo ou sigla utilizada, consulte o link Bibliografia em sua sala de aula, no site da ESAB.