UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE SOFTWARE PARA MONITORAÇÃO DO CABEÇALHO DO PROTOCOLO HTTP EM UMA REDE TCP/IP

FERNANDO HILGENSTIELER

BLUMENAU
2003

2003/2-15

FERNANDO HILGENSTIELER

PROTÓTIPO DE SOFTWARE PARA MONITORAÇÃO DO CABEÇALHO DO PROTOCOLO HTTP EM UMA REDE TCP/IP

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Francisco Adell Péricas - Orientador

BLUMENAU
2003

2003/2-15

PROTÓTIPO DE SOFTWARE PARA MONITORAÇÃO DO CONTEÚDO DAS MENSAGENS EM UMA REDE TCP/IP

Por

FERNANDO HILGENSTIELER

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Francisco Adell Péricas
Membro:	Prof. Sérgio Stringari
Membro:	Prof. Mauro Marcelo Mattos

Blumenau, 15 de Dezembro de 2003

Dedico este trabalho as pessoas que estiveram ao meu lado em momentos de conquistas e de dificuldades.

AGRADECIMENTOS

Agradeço ao pai de todos, DEUS. Aquele que nos guia por caminhos nem sempre fáceis de serem percorridos.

Aos meus queridos pais, José Roberto e Ivete Maria, por terem me dado a oportunidade de me tornar uma pessoa melhor.

Não poderia deixar de agradecer aos meus grandes irmãos, Adriano e Ricardo, que sempre estiveram presentes ao longo destes anos.

A minha querida irmã Karla, que demonstra tanto carinho por tudo que faz.

A Fabíola, que me ensinou a ser melhor e acreditar em minhas qualidades como ser humano.

Ao meu orientador Franciso Adell Péricas, que soube me conduzir aos caminhos que pareciam escuros e cheios de obstáculos.

Aos meus verdadeiros amigos, aqueles que estiveram comigo, não somente em momentos de diversão, mas aqueles que se mostraram companheiros.

RESUMO

Este trabalho apresenta a especificação e implementação de um protótipo de software para monitoração do cabeçalho do protocolo HTTP da camada de aplicação em um computador conectado a uma rede *Ethernet.* Para este protótipo foi feito um estudo sobre a arquitetura TCP/IP e sobre segurança de redes de computadores. O protótipo foi desenvolvido para o ambiente Windows.

Palavras chaves: Segurança de Redes, Monitoração, TCP/IP, *sniffer*.

ABSTRACT

This work presents the specification and implementation of a software prototype for monitoring the application layer of the HTTP protocol in a computer connected to the *Ethernet*. For this purpose, it has been done a study about the TCP/IP architecture and security of computer networks. The software prototype was developed for the Windows platform.

Key-Words: Security Networking, Network Monitor, TCP/IP, sniffer.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

ARPANET – Advanced Research Projects Agency Network

BNF – Backus-Naur Form

CRC – Cyclic Redundancy Check

CSMA/CD – Carrier Sense Multiple Access/Collision Detection

DES – Data Encryption Standard

FTP – File Transfer Protocol

HTTP – Hyper Text Transfer Protocol

ICMP – Internet Control Message Protocol

IEEE – Institute of Electrical and Electronic Engineers

IP – Internet Protocol

LLC – Logical Link Control

MIME – Multipurpose Internet Mail Extensions

OSI – Open Systems Interconnection

RSA – Rivest, Shamir and Adleman

SMTP – Simple Mail Transfer Protocol

SSL - Secure Sockets Layer

TCP – Transmission Control Protocol

TCP/IP – Transmission Control Protocol/Internet Protocol

TELNET – Telecommunications Network

UDP – User Datagram Protocol

URI – Uniform Resource Identifiers

URL – Uniform Resource Locator

SUMÁRIO

1 INTRODUÇÃO

Com a constante evolução das tecnologias da informação, tem havido um crescimento proporcional na necessidade da disponibilização de dados de forma rápida e segura. Dados que são considerados mais valiosos do que os próprios equipamentos.

A questão da rapidez foi solucionada, mas não a da segurança. Sabendo que seus dados encontram-se em áreas abertas, as empresas preocuparam-se em ampliar seus métodos de proteção desses dados contra possíveis espiões industriais, contra seus concorrentes e, principalmente, contra a ação de *hackers* (STARLIN, 1999).

Os dados, para trafegarem em uma rede, passam por um processo de codificação, passando de informações alfanuméricas para *bits*. Esses *bits* ficam agrupados em forma de pacotes e então são transmitidos pela rede. Sendo assim, não se tem mais o controle de quais pacotes estão trafegando, qual o destino dos pacotes, qual sua origem e que dado está trafegando.

Segundo Silva (2001), ferramentas de monitoração são importantes, pois através delas pode-se determinar com grande rapidez se houve realmente uma invasão à rede e quais os danos causados por esta invasão, auxiliando o administrador de rede a tomar providências para que os danos causados pela invasão sejam contidos o mais rápido possível.

Monitores, segundo McClure (2000), capturam, interpretam e armazenam, para análise posterior, pacotes que viajam por uma rede. Ferramentas como esta são de grande valia para o administrador da rede, pois possibilitam uma série de informações e diagnósticos do que está trafegando pela rede. Ainda segundo McClure (2000), um *sniffer* é um programa monitor que funciona em conjunto com a placa de rede para extrair todo o tráfego da rede, em vez de somente o tráfego endereçado ao *host* onde ele estiver instalado.

Este trabalho propõe a implementação de um *sniffer* para monitoração e captura do cabeçalho do protocolo HTTP da camada de aplicação para gerar relatórios de acesso a *sites* na internet.

Já foram desenvolvidos TCC´s nesta área, tais como Pompermayer (2002) e Silva (2001), os quais fazem a monitoração de pacotes IP, com objetivo de verificar o tráfego da

rede em relação aos endereços da camada de rede e de transporte. Este trabalho visa a camada de aplicação, analisando o conteúdo da navegação da internet através do protocolo HTTP.

1.1 OBJETIVOS DO TRABALHO

O objetivo principal deste trabalho consiste em especificar e desenvolver um protótipo de software de monitoração do cabeçalho das mensagens que trafegam em uma rede *Ethernet*, na camada de aplicação, para navegação internet (HTTP), utilizando o Sistema Operacional Windows.

Os objetivos específicos do trabalho são:

a) interceptar e interpretar pacotes TCP/IP;
b) monitorar dados contidos nas mensagens: camada de rede, de transporte e de aplicação;
c) identificar situações suspeitas;
d) armazenar as informações monitoradas.

1.2 ESTRUTURA DO TRABALHO

O trabalho está organizado em cinco capítulos:

No capítulo 1, encontra-se a introdução, os objetivos e a organização do trabalho.

No capítulo 2, são apresentados fundamentos sobre a internet e a arquitetura TCP/IP, necessários para o desenvolvimento deste trabalho.

No capítulo 3, são apresentados os protocolos da arquitetura TCP/IP.

No capítulo 4, é apresentado um estudo sobre Segurança de Redes, bem como algumas técnicas de ataque e de defesa a redes de computadores.

O capítulo 5 apresenta a especificação e implementação do protótipo.

Após o capítulo 5 são apresentadas conclusões, sugestões para trabalhos futuros e referências bibliográficas utilizadas para a elaboração deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 A INTERNET E SUA HISTÓRIA

A internet surgiu em 1969, nos Estados Unidos. Interligava originalmente laboratórios de pesquisa e se chamava ARPANET.

Era uma rede do Departamento de Defesa norte-americano, no o auge da Guerra Fria, e os cientistas queriam uma rede que continuasse funcionando em caso de um bombardeio. Surgiu então o conceito central da internet: "Uma rede em que todos os pontos se equivalem e não há um comando central". Assim, se B deixa de funcionar, A e C continuam a poder se comunicar.

O nome internet propriamente dito surgiu bem mais tarde, quando a tecnologia da ARPANET passou a ser usada para conectar universidades e laboratórios, primeiro nos Estados Unidos e depois em outros países.

Por isso não há um único centro que "governa" a internet. Hoje ela é um conjunto de mais de 40 mil redes no mundo inteiro. O que essas redes têm em comum é a arquitetura de rede TCP/IP, que permite que elas se comuniquem umas com as outras. Os protocolos dessa arquitetura são a língua comum dos computadores que integram a internet.

Então, a internet pode ser definida como:

a) uma rede de redes baseadas na arquitetura TCP/IP;
b) uma comunidade de pessoas que usam e desenvolvem essa rede;
c) uma coleção de recursos que podem ser alcançados através desta rede.

Durante cerca de duas décadas a internet ficou restrita ao ambiente acadêmico e científico. Em 1987 pela primeira vez foi liberado seu uso comercial nos Estados Unidos.

Mas foi em 1992 que a rede virou moda. Começaram a aparecer nos Estados Unidos várias empresas provedoras de acesso à internet. Centenas de milhares de pessoas começaram a colocar informações na internet, que se tornou uma mania mundial (fig. 1).

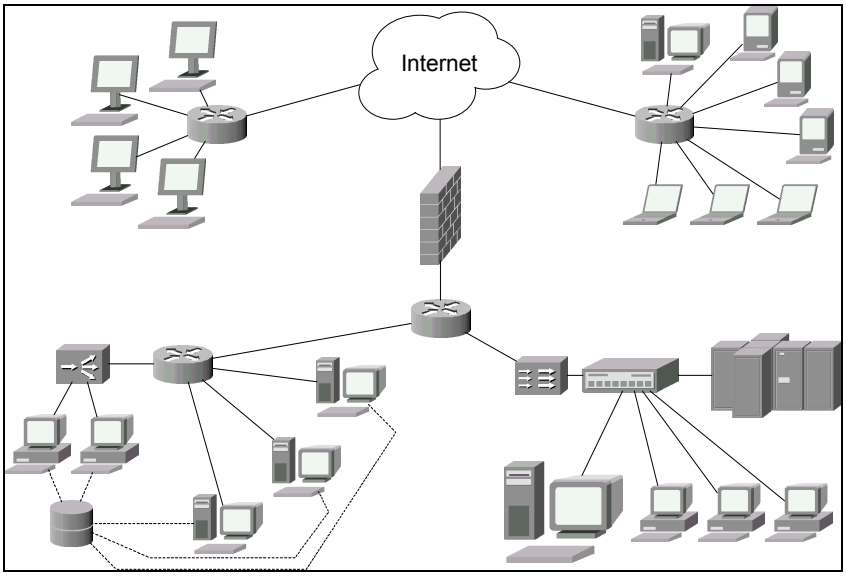


FIGURA 1 – Exemplo de um conjunto de redes de computadores

2.1.1 A ESTRUTURA DA INTERNET

Segundo Silva (2001), a estrutura da internet é formada basicamente por três elementos principais: os *backbones*, os provedores de acesso e os usuários finais.

Backbones são grandes redes de computadores de alta velocidade, projetadas para a transmissão de um grande volume de dados. Pode-se dizer que os *backbones* são a espinha dorsal das redes de computadores. A internet é formada por vários *backbones* espalhados pelo mundo, sendo que seu *backbone* principal encontra-se nos EUA.

Um provedor de acesso conecta-se diretamente aos *backbones* da internet, formando conexões dedicadas e permanentes. O objetivo do provedor de acesso é fornecer aos usuários finais acesso à internet e a outros serviços correlatos.

Os usuários finais podem ser particulares ou empresas que desejam obter acesso à internet. Os usuários finais conectam-se aos provedores de acesso através de uma rede local ou da rede telefônica, via modem, e podem utilizar todos os recursos disponíveis na internet.

A estrutura dos *backbones*, provedores de acesso ou usuários finais na internet é formada por *hosts,* que são computadores que estão conectados a uma única rede física e têm

o objetivo de executar programas aplicativos, e roteadores, que são equipamentos conectados a duas ou mais redes físicas e são responsáveis pela transmissão entre as redes.

2.2 A ARQUITETURA TCP/IP

Segundo Palma (2000), TCP/IP é um conjunto de protocolos usados em redes de computadores. TCP e IP são dois protocolos dessa família e por serem os mais conhecidos, tornou-se comum usar o termo TCP/IP para se referir à família inteira. É também usual falar em "suíte TCP/IP" para designar um pacote contendo o protocolo TCP/IP e alguns utilitários auxiliares.

O desenvolvimento da arquitetura TCP/IP começou com o Departamento Americano de Defesa, no final da década de 60, que resolveu criar um *software* de comunicação entre computadores de forma que fosse possível a comunicação remota ou local, entre sistemas operacionais iguais ou diferentes utilizando ou não o mesmo tipo de hardware. O protocolo permitiu que, no começo da década de 70, os computadores pudessem comunicar-se remotamente, sendo de suma importância em navios de guerra ou entre porta-aviões.

Segundo Starlin (1999), para que isto fosse possível, foi necessário criar um sistema de comunicação onde a rede como um todo fosse dividida em "pequenas" redes independentes, passando a usar como padrão de interconexão o TCP/IP.

A comutação por pacote (*packet-switching)*, segundo Redes (2001), é à base de sua tecnologia de comunicação e utiliza o pacote (*datagram*) como unidade de transmissão de dados. Cada *host* conectado à rede possui um endereço único, possibilitando que a máquina emitente reconheça a máquina receptora.

Os diversos tipos de protocolo disponíveis no TCP/IP trabalham em conjunto para que a comunicação possa ser efetuada. São organizados em um modelo de 4 camadas.

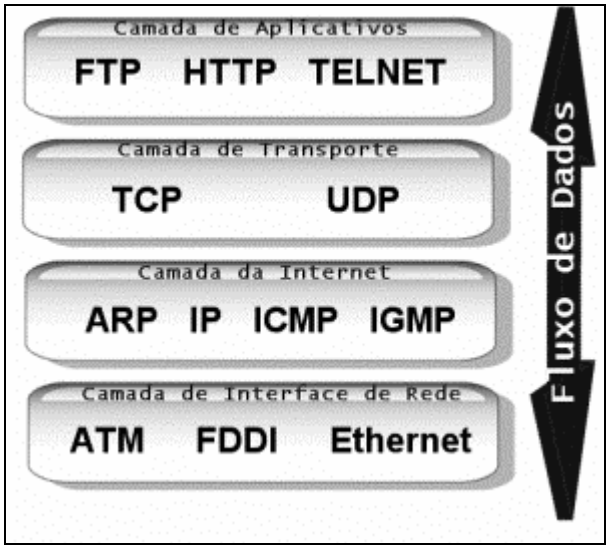


FIGURA 2 – Camadas da arquitetura TCP/IP

Conforme a fig. 2, as camadas são:

a) **camada de aplicação**: segundo Palma (2000), esta camada permite que aplicações tenham acesso às camadas inferiores através de portas TCP e UDP e define os protocolos utilizados por essas aplicações para envio e recepção de dados. Como exemplo temos os protocolos de HTTP e SMTP;

b) **camada de transporte**: os protocolos desta camada coordenam o envio de mensagens de um computador a outro, podendo ou não implementar algum mecanismo de controle para garantir a entrega das mensagens. Esses protocolos recebem pacotes da camada de aplicação e dividem-nos em *segmentos*, que são passados para a camada de rede;

c) **camada da internet (rede):** esta camada define o mecanismo utilizado para que o computador de origem localize o computador de destino, definindo a rota que as mensagens deverão percorrer. Os protocolos da camada de rede recebem datagramas da camada de transporte e analisam-nos para definir a rota que será utilizada;

d) **camada de interface de rede**: segundo Silva (2001), é responsável por encapsular os pacotes da camada da internet no formato específico da rede associada e extrair os pacotes dos quadros vindos da rede e encaminhá-los à camada da internet.

Quando as informações são enviadas, elas passam por todas essas camadas. Por exemplo, quando se transfere um arquivo para um servidor através do FTP, os dados e mais um cabeçalho contendo informações de controle do FTP serão enviados ao TCP.

Este adiciona suas informações de controle e repassa os dados ao IP. Este por sua vez adiciona seu próprio cabeçalho e repassa o bloco de informações (incluindo todos os cabeçalhos) para o *Ethernet*, que então codifica os dados em uma forma que possa trafegar no cabo da rede (fig. 3).

FIGURA 3 – Exemplo de envio de dados através do protocolo FTP

3 PROTOCOLOS DA ARQUITETURA TCP/IP

3.1 O PROTOCOLO HTTP

O protocolo HTTP é um protocolo do nível de aplicação que possui objetividade e rapidez necessárias para suportar sistemas de informação distribuídos, cooperativos e de hipermídia. Suas principais características são as seguintes:

a) propicia busca de informação e atualização;
b) as mensagens são enviadas em um formato similar aos utilizados pelo correio eletrônico da internet e pelo MIME;
c) comunicação entre os agentes usuários e *gateways*, permitindo acesso a hipermídia e a diversos protocolos do mundo internet;
d) obedece ao paradigma de cliente/servidor: um cliente estabelece uma conexão com um servidor e envia um pedido ao servidor, o qual o analisa e responde. A conexão deve ser estabelecida antes de cada pedido de cliente e encerrada após a resposta.

As mensagens seguem o formato da RFC 822 (RFC 822, 2003) e se apresentam na forma de:

a) pedidos enviados pelo cliente ao servidor;
b) respostas enviadas pelo servidor para o cliente.

3.1.1 PEDIDOS ENVIADOS PELO CLIENTE AO SERVIDOR

O formato da mensagem de pedido enviado do cliente ao servidor está descrito abaixo, de acordo com a notação BNF:

a) Pedido = Pedido Simples | Pedido Completo;
b) Pedido Simples = *Método* SP *Pedido-URI* CRLF;
   - SP: espaço;
   - CRLF: *carriage return* / *line fee*d.
c) Pedido Completo = Linha de pedido * (Cabeçalho geral | Cabeçalho do pedido | Cabeçalho da entidade) CRLF [Corpo da entidade];
d) Linha de Pedido = *Método* SP *Pedido-URI* SP Versão do protocolo CRLF.

Método indica a forma a ser aplicada para requisitar um recurso. Os métodos aceitos por um determinado recurso podem mudar dinamicamente. O cliente é notificado com o

código 501 quando o método é desconhecido ou não implementado. Os métodos são sensitivos ao caso.

Os principais métodos são:

a) GET: recupera todas as informações identificadas no recurso da rede. Se o recurso for um processo executável, ele retornará a resposta do processo e não o seu texto. Existe o GET condicional que traz o recurso apenas se o mesmo foi alterado depois da data da última transferência;
b) HEAD: semelhante ao método GET, só que neste caso não há a transferência da entidade para o cliente. Este método é utilizado para testar a validade e acessibilidade dos *links* de hipertexto;
c) POST: utilizado para solicitar que o servidor destino aceite a entidade constante no pedido como um novo subordinado ao recurso constante no URI. Suas principais funções são:
   - anotações de recursos existentes;
   - postar uma mensagem em um *bulletin board* , *newsgroup*, *mailing list*;
   - abastecer um processo com um bloco de dados;
   - estender uma base de dados com uma operação de *append*.

   A entidade é subordinada da mesma forma que um arquivo é subordinado ao diretório ou um registro a base de dados;
d) PUT: coloca a entidade abaixo do recurso especificado no pedido. Se esta entidade não existe é criada. Se existe, apenas é atualizada;
e) DELETE: solicita que o servidor origem apague o recurso identificado no URI;
f) LINK: estabelece uma ou mais relações de *links* entre o recurso identificado pelo URI e outros recursos existentes, não permitindo que o corpo da entidade enviada seja subordinada ao recurso;
g) UNLINK: remove uma ou mais relações de *links* existentes entre o recurso identificado no URI;

Pedido-URI identifica o recurso da rede. Por exemplo:

a) http://www.inf.furb.br/~pericas. Quando o caractere "*" aparece antes do recurso da rede, o mesmo indica que a requisição não se aplica ao recurso de rede especificado e sim ao seu servidor.

A notação BNF estabelece as seguintes regras de construção:

a) nome = definição: nome da regra, separada de sua definição através do sinal de “=”.
b) "literal": caracteres entre aspas indicam texto;
c) regra1 | regra2: elementos separados por barra indicam alternativas, isto é, apenas uma das regras é selecionada;
d) (regra1 regra2): elementos entre parênteses são tratados como elementos simples.
e) *regra: o caracter “*” quando precede um elemento indica repetição;
f) [regra]: elementos entre colchetes indicam elementos opcionais;
g) N regra: especifica quantas vezes exatamente o elemento deve ser repetido;
h) #regra: define listas de elementos. A forma "<n>#<m>elemento" indica no mínimo "n" e no máximo "m" elementos;
i) comentários: é indicado por “;”.

O pedido possui três cabeçalhos distintos:

a) cabeçalho geral: são dados complementares não relacionados com as partes que estão se comunicando nem com o conteúdo sendo transferido. Exemplo: data, versão do MIME;
b) cabeçalho do pedido: informações adicionais sobre o pedido e o cliente, como por exemplo, o intervalo de respostas esperadas no processamento da requisição;
c) cabeçalho da entidade: informações adicionais sobre a entidade, tais como: título da entidade, tamanho, linguagem utilizada.

3.1.2 RESPOSTAS ENVIADAS PELO SERVIDOR AO CLIENTE

O formato da mensagem da resposta enviada ao cliente é descrito abaixo, de acordo com a notação BNF:

a) Resposta = Resposta Simples | Resposta Completa.
b) Resposta Simples = [Corpo da entidade].
c) Resposta Completa = Linha de status * (Cabeçalho geral | Cabeçalho da resposta | Cabeçalho da entidade) CRLF [Corpo da entidade].
d) Linha de Status = Versão Protocolo SP Status Descrição Status CRLF.

A resposta possui três cabeçalhos distintos:

a) cabeçalho geral: dão dados complementares não relacionados com as partes que estão se comunicando nem com o conteúdo sendo transferido. Exemplo: data, versão do MIME;
b) cabeçalho da resposta: informações adicionais sobre a resposta e o servidor, como por exemplo, local do recurso, software utilizado no servidor;
c) cabeçalho da entidade: informações adicionais sobre a entidade, tais como: título da entidade, tamanho, linguagem utilizada.

O status representa o resultado do processamento executado pelo servidor. O status possui o formato NXX, onde:

a) N representa a classe da resposta;
b) XX representa a categoria da resposta.

Atualmente o protocolo possui cinco classes de respostas:

a) 1XX: não utilizada, reservada para utilização futura;
b) 2XX: a ação foi recebida, entendida e executada com sucesso. Com o método GET, a entidade correspondente é enviada com a resposta. Com o método HEAD, a resposta contém o cabeçalho da informação. Com o método POST, a resposta descreve/contém o resultado da ação. Nos restantes, a resposta descreve o resultado da ação. Exemplos:
   - 201 - Um novo recurso foi criado;
   - 202 - O pedido foi aceito para processamento, mas o mesmo não foi concluído;
c) 3XX: redirecionamento indica que as ações devem ser efetuadas em ordem para completar o pedido. Exemplos:
   - 300 - O recurso requisitado está disponível em mais de um local e o local preferido não pode ser determinado via negociação;
   - 302 - O recurso requisitado reside temporariamente em outro URI;
d) 4XX: erro no cliente. O pedido contém erro de sintaxe ou não pode ser efetuado. Exemplos:
   - 401 - O recurso requisitado necessita autenticação do usuário;
   - 404 - O servidor não encontrou o recurso definido no URI;
e) 5XX: erro no servidor. O servidor falhou ao executar um pedido aparentemente válido. Exemplos:
   - 500 - Erro interno no servidor;

- 501 - Recurso solicitado não implementado no servidor.

3.2 O PROTOCOLO TCP

O TCP verifica a existência de erros em cada pacote que recebe para evitar a corrupção dos dados. A estrutura do TCP possui os seguintes campos principais:

a) porta de origem: uma porta define o aplicativo na camada mais acima que transmitiu e que deverá receber os dados na outra ponta. Neste campo, é especificada a porta de origem;
b) porta de destino: a porta (aplicativo) de destino;
c) número de seqüência: define o número de seqüência do pacote TCP. Ele serve para evitar que ocorra a corrupção dos dados se um pacote chegar ao destino antes de outro, ou para detectar algum pacote que porventura se perdeu no caminho entre a origem e o destino;
d) número de confirmação: este número é enviado pela estação de destino para a estação de origem, para confirmar o recebimento de pacote(s) recebido(s) anteriormente;
e) *offset* dos dados: indica o tamanho do cabeçalho TCP, em blocos de 32 bits.
f) reservado: um campo reservado para uso futuro, sempre planejando previamente as melhorias ao protocolo;
g) bits de controle: são os seguintes:
   - URG (urgente): envia uma mensagem ao destino de que dados urgentes estão esperando para serem enviados a ele;
   - ACK (confirmação): confirma o recebimento de um ou mais datagramas enviados anteriormente;
   - PSH (p*ush*): faz com que o TCP imediatamente envie os dados pendentes;
   - RST (r*eset*): reinicia a comunicação entre os *hosts*;
   - SYN: usado na inicialização e para estabelecer um número de seqüência;
   - FIN: mais nenhum dado está vindo da estação de origem;
h) janela: o número de octetos de dados que começam com o valor indicado no campo *Acknowledgement* que o remetente do segmento de dados está querendo aceitar. Este é o método de controle de fluxo do TCP e se o receptor quiser que a estação de origem pare de enviar dados, ele configurará este campo como zero;
i) soma de verificação: somente um método de detecção de erros;

j) ponteiro urgente: aponta para o número de seqüência do byte após os dados urgentes. Interpretado apenas quando URG é definido.

O TCP é um protocolo orientado à conexão porque os dois computadores participantes da transmissão de dados sabem da existência um do outro. Esta conexão virtual entre eles é chamada de sessão. Entre outras coisas, o *handshake* de três vias sincroniza os números de seqüência entre as duas estações de rede.

O processo ocorre da seguinte maneira:

a) o computador de origem inicia a conexão, transmitindo informações da sessão como número de seqüência e tamanho do pacote;
b) o computador de destino responde com suas informações sobre a sessão;
c) o computador de origem confirma o recebimento das informações e a sessão é estabelecida. Com os números de seqüência sincronizados, a transferência de dados pode ser efetuada sem erros.

A transferência de arquivos seria muito lenta se cada vez que o TCP enviasse um pacote, esperasse pela confirmação de recebimento para enviar o próximo: para evitar este problema, criou-se o "janelamento". Podemos definir este processo como sendo a quantia de dados que a estação de origem pode enviar sem receber confirmação de cada pacote.

Uma porta se refere ao aplicativo da camada de aplicação que irá processar os dados. Como o número de porta é de 16 bits, tem-se que o TCP/IP permite 65.536 números de porta diferentes, sendo as primeiras 1024 estáticas (pré-definidas) e as outras dinâmicas. Geralmente, a porta de origem é gerada aleatoriamente pela aplicação, enquanto, a porta de destino é fixa.

Serviço	Porta	Alias	# Comentário
ftp-data	20		# FTP, dados
ftp	21		# FTP, controle
telnet	23		
smtp	25	mail	# *Simple Mail Transfer Protocol*
http	80	www www-http	# *World Wide Web*
pop3	110		# *Post Office Protocol* - versão 3

fonte: \windows\system32\drivers\etc\service

QUADRO 1 – Algumas portas do TCP

Segundo Pompermayer (2002), para garantir a transferência confiável de dados, seu principal propósito, o TCP deve oferecer alguns serviços, tais como:

a) transferência de dados: pode ser feita tanto no modo *full-duplex* quanto *half-duplex*. Em cada conexão, é o protocolo que determina o momento mais conveniente de bloquear ou liberar o dado para transmissão. O processo do ajuste de tempo para a liberação é feito pela diferença do tempo em que foi enviado o segmento e a chegada da confirmação de recebimento desse segmento;
b) confiabilidade: o TCP garante a correção dos dados nos casos de alteração, perda, duplicação ou entrega fora de seqüência. Esse controle é feito basicamente com o número de seqüência. Pode ocorrer a perda de dados, segmentos chegarem fora de ordem, duplicação de segmentos, erros de conteúdo;
c) controle de fluxo: mecanismo chamado de janela deslizante de tamanho variável ou janela de crédito, que permite à estação receptora controlar a quantidade de dados enviados pela estação transmissora;
d) multiplexação: cada máquina possui um conjunto de portas e isto permite que vários processos dentro de uma mesma máquina façam uso simultâneo dos serviços de comunicação do protocolo;
e) conexão: o TCP, por ser orientado à conexão, caracteriza três fases de funcionamento: estabelecimento da conexão, onde são iniciadas as opções de *status* e a sincronização dos números de seqüência; transferência de dados, ocorrendo à verificação da recepção correta dos dados; e liberação da conexão, que deve ser feita para liberar recursos para outros usuários;
f) relatório de erros: erros não recuperáveis relacionados a pedidos de serviço ou a problemas ocasionados pelo funcionamento deficiente do ambiente interno da rede são relatados ao processo usuário pelo TCP.

O TCP é um protocolo complexo que promove a comunicação através de uma grande variedade de tecnologias de rede. Segundo Redes (2001), a generalidade do TCP não parece interferir no seu desempenho.

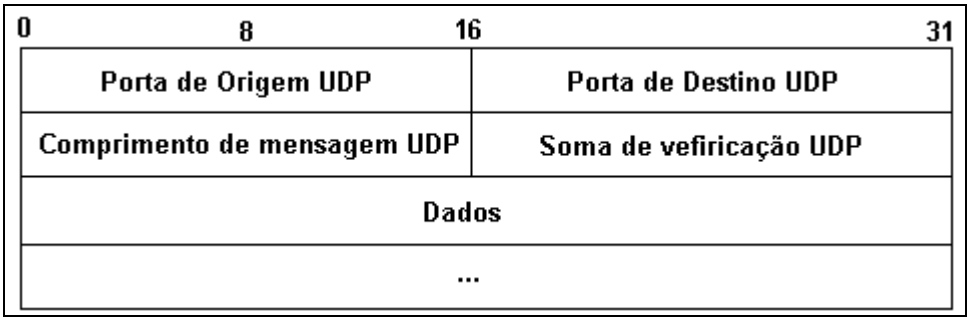
Informações mais detalhadas podem ser encontradas em Comer (1998).

3.3 O PROTOCOLO UDP

Segundo Kurose (2003), ao contrário do TCP, este protocolo não é confiável, não é baseado em sessão e não confirma cada pacote enviado - no entanto ele identifica o aplicativo da camada superior usando também um número de porta. Por não ter todos esses

procedimentos para detecção de erros, o UDP é um protocolo bem mais leve do que o TCP e adiciona muito menos informações ao cabeçalho.

O UDP é geralmente utilizado por aplicativos que ou implementam seu próprio mecanismo de entrega de dados confiável ou que simplesmente não necessitam dessa função. Um bom exemplo disso são aplicações de videoconferência: as informações devem chegar rapidamente ao destino, já que um pacote atrasado não terá mais serventia. O formato do datagrama está representado na Figura 4.



QUADRO 2 – Formato do datagrama UDP

O cabeçalho do UDP está subdividido em quatro campos de 16 bits:

a) porta de origem e porta de destino: contém os números de portas do protocolo UDP de 16 bits usados para demultiplexar os datagramas entre os processos que esperam para recebê-lo, tendo a porta de origem como opcional. Se não usada, deverá ser zero;
b) tamanho da mensagem (em blocos de 32 bits): possui como valor mínimo 8 bytes, que é o tamanho apenas do cabeçalho. O valor total é a contagem de octetos do cabeçalho e dos dados de usuário;
c) soma de verificação: campo opcional. Se não for usado significa que a soma de verificação não foi calculada.

Segundo Redes (2001), existe uma forte interação entre UDP e IP, violando a premissa básica de que a colocação em camadas reflete separação de funcionalidade. O UDP tem estado estreitamente integrado ao protocolo IP. Isso é aceitável pois é impossível identificar inteiramente um programa aplicativo de destino sem especificar a máquina de destino.

A definição de números de porta, é fundamental para permitir mais do que um destino em cada máquina. Num sistema multi-processo, cada processo que utiliza a rede requisita uma

porta que lhe fica associada. Todos os dados que chegam ao *host* com essa porta de destino são encaminhados para esse processo.

3.4 O PROTOCOLO IP

O IP é responsável por endereçar e entregar os dados em uma comunicação de rede TCP/IP. Ele sempre tenta entregar os dados, mas não garante que eles sejam entregues, que eles sejam entregues livres de erros ou que sejam entregues na mesma ordem pois isto é função de um protocolo da camada superior, como o TCP.

O IP não estabelece uma sessão (*link* virtual) entre o computador de origem e o de destino. Ele também é responsável por pegar os dados da camada de interface de rede e apresentá-los ao protocolo da camada acima que os solicitou.

O IP, quando os dados chegam a ele vindos da camada superior, adiciona suas informações de controle e um cabeçalho aos dados. A partir daí, aquele bloco de dados é chamado de *datagrama.*

O datagrama, ao receber as informações de cabeçalho da camada de interface de rede (como o protocolo *Ethernet*) é chamado de *pacote*.

O cabeçalho do IP tem vários campos, a saber:

a) versão: este campo define a versão do protocolo IP utilizada;
b) comprimento do cabeçalho: este campo define o tamanho, em blocos de 32 bits, do cabeçalho do IP;
c) tipo de serviço: este campo contém quatro subcampos. O primeiro é o campo de precedência, que permite configurar a prioridade com que estes dados devem ser enviados na rede. Ele ocupa três bits, e por isso, seus valores variam de 0 a 7. Os outros três campos são bits únicos e, teoricamente, deveriam controlar como o datagrama deve ser roteado na rede: são os bits de baixo Retardo (*Delay*), Rendimento (*Troughput*) e Confiabilidade (*Reliability*);
d) comprimento total: tamanho total do datagrama medido em bytes. Como este campo é composto por 16 bits, o tamanho máximo de um datagrama IP é de 65535 bytes;

e) identificação: este campo tem a ver com a fragmentação de datagramas IP. Ele identifica quais fragmentos pertencem a qual datagrama, para que o destino não se confunda na hora de remontá-lo;
f) sinalizadores: indicam, em uma fragmentação de datagramas, se mais fragmentos chegarão ou se não serão enviados mais dados pertencentes ao datagrama identificado no campo acima. Também indicam se um datagrama pode ou não ser fragmentado. Caso o bit de não fragmentação for definido, o roteador que receber o datagrama e precisar fragmentá-lo para seguir adiante simplesmente descartará aquele datagrama e enviará uma mensagem de erro ao computador de origem;
g) *offset* do fragmento: indica que parte do datagrama à estação de destino deve continuar a remontá-lo para determinado fragmento. Por exemplo, se o primeiro fragmento fosse de 576 bytes e este fragmento fosse o segundo, o campo *offset* conteria 577, indicando que este fragmento inicia o 577º byte do datagrama;
h) tempo de vida: define o tempo máximo que um datagrama pode trafegar na rede antes de ser descartado. Cada vez que o datagrama passa por um roteador, este campo é diminuído. Foi criado para que um pacote não fique eternamente trafegando na rede. Ele é definido pela estação de origem e é de 8 bits, ou seja, seus valores variam de 0 a 255;
i) protocolo: este campo o protocolo da camada superior para o qual os dados devem ser repassados;
j) soma de verificação: aqui está contida uma CRC do cabeçalho IP com o objetivo de evitar erros na transmissão. O CRC é um número que é calculado pelo computador de origem com base no conteúdo do cabeçalho. Quando a estação de destino receber o datagrama, ela calculará a sua CRC com base nos dados de cabeçalho que foram recebidos e caso os valores sejam diferentes dos contidos no campo soma de verificação, um erro na transmissão ocorreu e o datagrama será descartado. Note que este campo é recalculado cada vez que o datagrama passa por um roteador, pois este altera o campo TTL do cabeçalho;
k) endereço IP de origem e destino: estes campos dispensam explicações. O endereçamento IP será discutido posteriormente;
l) opções do IP: pode ou não estar presente (pouco utilizado atualmente). Especifica diversas opções do datagrama IP, como a que permite que os pontos por onde o

datagrama passa sejam "ecoados" para o computador de origem e a que configura uma rota restrita, limitando a liberdade dos roteadores em encaminhar o datagrama.

O endereçamento lógico no TCP/IP é definido pelo protocolo IP, que determina que cada computador da rede deve ter um endereço IP único na rede, composto de 32 bits. Esse endereço define a rede à qual o computador pertence e o identifica nessa rede. Essa informações permitem ao protocolo IP implementar o mecanismo de roteamento das redes TCP/IP.

4 SEGURANÇA EM REDES DE COMPUTADORES

Nos dias atuais, uma conexão permanente à internet está ao alcance de qualquer organização e de praticamente qualquer indivíduo. Ao mesmo tempo, as conexões temporárias (discadas e similares) permitem níveis de serviço impensáveis há poucos anos.

Paralelamente a isso, muitos dos sistemas operacionais utilizados nos computadores conectados diretamente à rede continuam vindo com uma série de funcionalidades e facilidades pré-ativadas que não contribuem para a criação de um ambiente seguro. É bom lembrar que o projeto de segurança trabalha com a premissa de que a segurança e a conveniência (do ponto de vista do usuário final) são inversamente proporcionais.

Além desses fatores, há um terceiro que deve ser considerado. Hoje, comprar um computador e colocá-lo na rede está ao alcance de um público muito grande, que geralmente não vai contar com a ajuda de um profissional do ramo antes de conectar-se à rede de seu condomínio ou conjunto empresarial.

Por mais seguro que um sistema operacional seja, as atitudes de um administrador sem treinamento formal ou até mesmo informal podem transformá-lo em uma fortaleza de portas escancaradas, pronto para receber todo tipo de ataque e invasões. Segurança de rede é uma preocupação que deve ser compartilhada por todos, inclusive pelo usuário final.

Vários são os motivos que levam *hackers* e *crackers* a invadirem sistemas. Existe uma diferença entre *hackers* e *crackers*: os primeiros invadem sistemas apenas para o aprimoramento dos seus conhecimentos ou para testar a segurança dos softwares e buscar vulnerabilidades. Não há o roubo de informações ou qualquer outro dano. Já *crackers* invadem com o objetivo de danificar e/ou roubar informações, projetos ou verbas.

Segundo Bernstein (1997), *crackers* revelaram os seguintes motivos para seus atos:

a) ganhos financeiros: geralmente os intrusos são funcionários que obtém acesso a sistemas financeiros para roubar dinheiro através de desvios ou para reunir informações que poderão ser vendidas;
b) vingança: funcionários descontentes com sua situação na empresa ou que foram demitidos podem ser os responsáveis por danos físicos ou de dados;

c) necessidade de aceitação ou respeito: raramente ataques com esse motivo acarretam perdas. Geralmente são efetuados como teste para entrar em grupos de *hackers* ou para promover seu conhecimento aos olhos alheios;
d) idealismo: alguns intrusos acham-se como heróis protegendo o mundo do governo ou sentem-se verdadeiros "*Robin Wood"* virtuais;
e) curiosidade ou busca de emoção: alguns intrusos simplesmente querem saber "o que tem dentro", ou são viciados na adrenalina que o ato ilegal causa;
f) anarquia: têm como objetivo promover a discórdia ou a confusão e são motivados pelo ato ilegal de tentar invadir sistemas;
g) aprendizado: invadem sistemas para aprimorar suas técnicas e conhecimentos sobre os diversos sistemas e formas de proteção;
h) ignorância: alguns intrusos não têm consciência da ilegalidade dos seus atos, nem que podem ser punidos por suas ações;
i) espionagem industrial: ocorre quando uma empresa ou organização tem como objetivo atacar outras empresas ou organizações com o objetivo de roubar informações confidenciais ou até mesmo produtos;
j) espionagem internacional: semelhante à espionagem industrial, com a diferença de que os ataques são contra outros países. As informações adquiridas podem ser de natureza estratégica, onde guerras e conflitos podem ser descobertos.

4.1 POLÍTICAS E PROCEDIMENTOS DE SEGURANÇA

Segundo Bernstein (1997), políticas de segurança com boas informações são o fundamento para um programa de segurança de informações eficaz. Em termos gerais, as políticas são diretivas de gerenciamento que estabelecem as metas comerciais da organização, fornecem uma estrutura de implementação para alcançar os objetivos dessas metas e atribuem responsabilidades e domínios ao processo. Mais especificamente, as políticas de segurança são projetadas para gerenciar o risco em que a empresa incorre ao buscar esses objetivos comerciais. O termo risco nesse contexto se refere às informações, representando um risco tanto comercial quanto de segurança.

Os procedimentos incorporam os objetivos de segurança para sistemas ou aplicações específicas que definem como os sistemas deverão ser operados sob determinadas circunstâncias para que os objetivos sejam alcançados. Os procedimentos de resposta a

incidentes, por exemplo, especificam o que usuários, administradores de sistema e gerentes devem fazer se alguém descobrir algum indício de um incidente relacionado a segurança. Os procedimentos de segurança das informações quase sempre derivam de políticas da empresa.

4.2 ALGUMAS TÉCNICAS DE INVASÃO

Segundo Bernstein (1997), um ataque é qualquer ação não autorizada empreendida com a intenção de impedir, danificar, incapacitar ou quebrar a segurança de um servidor conectado à internet. O ataque pode ser uma simples recusa de serviços ou até a captura ou destruição de todas as informações contidas no servidor.

As técnicas mais utilizadas são:

a) *footprinting*: o *footprinting* de uma empresa permite que invasores criem um perfil completo da postura de segurança dessa organização. Usando uma combinação de ferramentas e técnicas – atacantes podem empregar um fator desconhecido (a conexão à internet da organização) e convertê-lo em um conjunto específico de nomes de domínio, blocos de rede e endereços IP individuais de sistemas conectados diretamente à internet. Embora haja diversas técnicas diferentes de *footprinting*, seu objetivo primário é descobrir informações relacionadas a tecnologias de internet, intranet, acesso remoto e extranet;
b) varreduras de rede: um dos passos mais básicos no mapeamento de uma rede é realizar uma varredura *ping* automatizada em um intervalo de endereços de IP e blocos de rede para determinar se sistemas individuais estão ativos. O *ping* é tradicionalmente usado para enviar pacotes ICMP_ECHO (tipo 8) para um sistema-alvo, em uma tentativa de obter um ICMP_ECHO_REPLY (tipo 0), que indica que o sistema-alvo está ativo. Embora o *ping* seja aceitável para determinar o número de sistemas ativos em uma rede de tamanho médio, ele é pouco eficiente para redes corporativas, maiores;
c) enumeração: a diferença-chave entre as técnicas de coleta de informações (*footprinting e varredura*) e a enumeração está no nível de invasividade – a enumeração envolve conexões ativas a sistemas e consultas dirigidas. Assim, elas podem ser registradas ou sinalizadas de alguma forma;
d) DoS (*Denial of Service*): segundo McClure (2000), um ataque DoS interrompe ou nega completamente serviço a usuários legítimos, redes, sistemas ou outros

recursos. O objetivo de tais ataques é normalmente de natureza mal-intensionada e muitas vezes exige pouca habilidade, pois as ferramentas que os possibilitam são facilmente encontradas;

e) cavalo de tróia: consiste em falsificar programas normalmente utilizados como: *ftp, ifconfig* e assim por diante. O intruso, após conseguir acesso de superusuário, troca um dos programas utilizados por uma versão modificada que pode desde roubar senhas até apropriar-se indevidamente de dados;

Segundo McClure (2000), muitas das informações obtidas por meio de enumeração podem parecer inofensivas à primeira vista. No entanto, as informações que vazam desses buracos podem ser sua ruína. Em geral, uma vez que um nome de usuário ou de compartilhamento válido seja enumerado, normalmente é só uma questão de tempo antes que o invasor adivinhe a senha correspondente ou identifique algum ponto fraco associado ao protocolo de compartilhamento do recurso.

4.3 FORMAS DE DEFESA

As formas de defesa mais eficientes dizem respeito à implementação e adoção de políticas de segurança, utilização de *firewalls*, criptografia ou outro esquema que sirva para proteger ao máximo o sistema. Mas as dificuldades não param por aí. Além de implementar as técnicas de segurança, faz-se necessário identificar onde está a vulnerabilidade para que seja possível corrigí-la, fazer o levantamento dos possíveis prejuízos e identificar os invasores.

Ao se implementar uma política de segurança, segundo Silva (2001), deve-se visar principalmente três aspectos:

a) privacidade: que visa proteger as informações contra acesso de qualquer pessoa não explicitamente autorizada pelo proprietário da informação, ou seja, o acesso aos recursos deve ser acessível somente às pessoas devidamente autorizadas;
b) integridade de dados: evitando que dados sejam apagados ou de alguma forma alterados, sem a permissão do proprietário da informação. Obter a garantia de que pessoas não autorizadas ou de má índole não realizem ‘estragos’ à rede;
c) disponibilidade: proteger os serviços prestados aos usuários autorizados de forma que nenhuma pessoa externa ou interna sem autorização danifique os processos que mantém a operacionabilidade.

4.3.1 FIREWALLS

Segundo Oliveira (2000), pode ser chamado de *firewall* qualquer dispositivo que impeça que estranhos tenham acesso à rede. Eles comumente auxiliam no planejamento e nas regras da rede. Um *firewall* é um dispositivo de hardware (roteadores, servidores) ou software. A localização do *firewall* depende do projeto da rede que se quer proteger, ficando entre a rede interna e a internet.

O papel fundamental de um *firewall* é monitorar todo tráfego que flui entre duas redes e bloquear certos tipos de tráfegos completamente: por exemplo, pode-se configurar o *firewall* para bloquear protocolos como o UDP, limitar o acesso às portas do servidor, limitar a saída e a entrada dos dados.

Um *firewall* é o ponto de entrada de uma rede e por isso, quando tem suas regras bem implementadas e mantidas, torna-se uma barreira extremamente funcional contra ataques. Mas os invasores sabem disso e a alternativa é tentar entrar por algum outro caminho que não passe pelo *firewall*, como um acesso discado.

4.3.2 CRIPTOGRAFIA

Criptografia, segundo Oliveira (2000), é a arte ou ciência de escrever em cifra ou em códigos, ou seja, um conjunto de técnicas que tornam uma mensagem incompreensível permitindo apenas que o destinatário, que conhece a chave de encriptação, consiga desencriptar e ler a mensagem com clareza. É uma das melhores formas de segurança de dados e uma possibilidade que não deve ser esquecida, principalmente se trabalharmos com dados confidenciais e envio de mensagens via internet/intranet.

A criptografia dos dados é feita através de algoritmos que codificam e decodificam os dados, enviando-os de forma ilegível para que, se forem detectados por algum *sniffer* ou algum outro dispositivo que intercepte os pacotes, seja praticamente impossível a sua decodificação. Os algoritmos devem ser conhecidos e testados, e a segurança reside totalmente na chave secreta, a qual deve ter tamanho suficiente para evitar sua descoberta por teste exaustivo. Estes algoritmos podem ser classificados em:

a) algoritmos simétricos: esses algoritmos usam a mesma chave para encriptar e decriptar. Pode-se citar o DES;
b) algoritmos assimétricos: são utilizadas chaves diferentes para a encriptação e

decriptação dos dados, existindo uma relação entre as chaves.

Dependendo do tipo de encriptação que está sendo utilizado, pode-se ter uma queda na performance do sistema, pois quanto mais complexo for o algoritmo de criptografia utilizado, mais processamento irá exigir da máquina.

4.3.3 OUTRAS MEDIDAS DE SEGURANÇA

Por maior que seja o aparato de equipamentos e de softwares de segurança de uma empresa, algumas medidas consideradas simples podem ser adotadas para dificultar o acesso não autorizado às informações e sistemas. Algumas dessas medidas são:

a) fazer com que os usuários passem a utilizar e criar senhas mais seguras, não sendo tão óbvias e promover a troca periódica das senhas;
b) deixar claro o perigo contido em arquivos anexados em e-mails de procedência desconhecida e a instalação de qualquer tipo de *software;*
c) manter uma atualização constante de *drivers* e aplicativos, buscando versões com correções contra vulnerabilidades nas versões antigas;
d) eliminar serviços que não sejam realmente necessários para a empresa;
e) fazer um levantamento periódico de contas de usuários inativos e eliminá-las;
f) estar sempre atualizado em questões de segurança inscrevendo-se em listas de discussão sobre o assunto e fazer visitas periódicas a endereços na internet que tratam do assunto.

Medidas preventivas são a melhor forma de manter as informações seguras das invasões a uma rede de computadores.

5 DESENVOLVIMENTO DO TRABALHO

Em síntese, para a realização deste trabalho foram executados procedimentos de especificação e implementação visando o cumprimento dos objetivos utilizando-se de metodologias de desenvolvimento de software, ferramenta de especificação e ambiente de programação já consagrados tecnologicamente de acordo com a categoria em que o trabalho se enquadra.

5.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Foram levantados alguns requisitos que devem estar presentes no protótipo. Estes requisitos demonstram algumas características que o protótipo implementa para sua funcionalidade.

Os requisitos principais do protótipo são:

a) operar no ambiente (sistema operacional) Windows;
b) localizar o adaptador de rede;
c) iniciar a captura de pacotes de uma rede *Ethernet*;
d) monitorar os protocolos da arquitetura TCP/IP;
e) identificar o procotolo TCP e/ou UDP;
f) capturar e analisar os pacotes que trafegam pela camada de aplicação;
g) filtrar por um determinado *Host* (opcional);
h) possibilitar a mudança dos filtros durante a execução do monitor;
i) no decorrer da execução, mostrar o conteúdo capturado ao usuário;
j) permitir que os dados capturados fossem salvos.

5.2 ESPECIFICAÇÃO

A especificação do protótipo, é constituída de um único módulo, o qual implementa todas as funcionalidades necessárias para o correto funcionamento do software deste trabalho.

A metodologia de especificação utilizada para representar os processos que formam o protótipo foi a estruturada usando a fluxogramação, que segundo Demarco (1989), é uma forma rápida e simples de se ter uma visão macro dos problemas em questão. Para auxiliar na construção dos fluxogramas utilizados na especificação do protótipo foi utilizada a ferramenta EDGE *Diagramer* (SOFTWARE 2003).

5.2.1 MONITOR DE APLICAÇÕES

Ao ser executado, o protótipo faz a inicialização do *driver* do adaptador de rede, para verificar se o mesmo pode ser aberto, caso contrário, o Monitor de Aplicações não poderá capturar os pacotes que trafegam pela rede. A representação macro deste processo, está ilustrada na fig. 5.

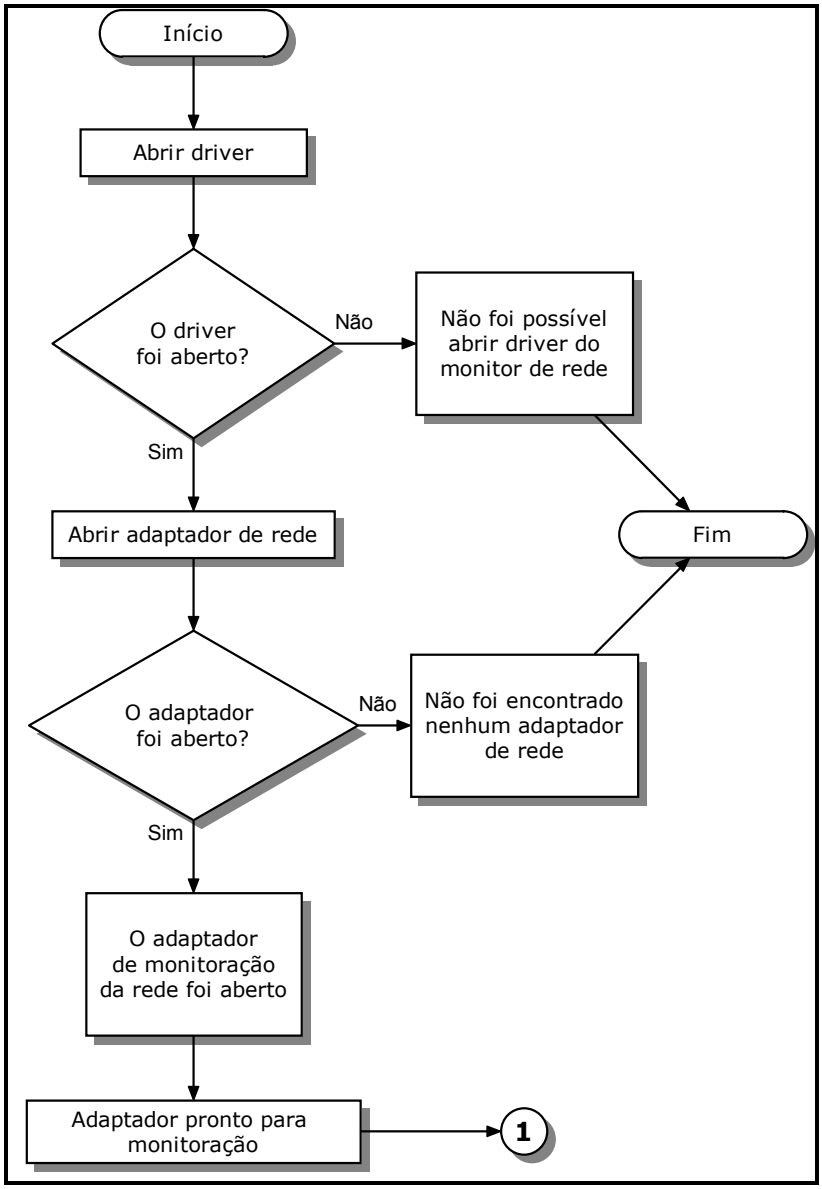


FIGURA 4 – Procedimento de inicialização do adaptador de rede

Após a inicialização do adaptador de rede, o Monitor de Aplicações está pronto para capturar os pacotes que trafegam pela rede. Assim que iniciada está captura, é verificada a existência de pacotes. Ao capturar estes pacotes, se identificado o protocolo IP, é armazenado

o IP de origem e de destino e verificado se o protocolo é TCP ou UDP, para que os dados possam ser processados. A representação macro deste processo, está ilustrada na fig. 6.

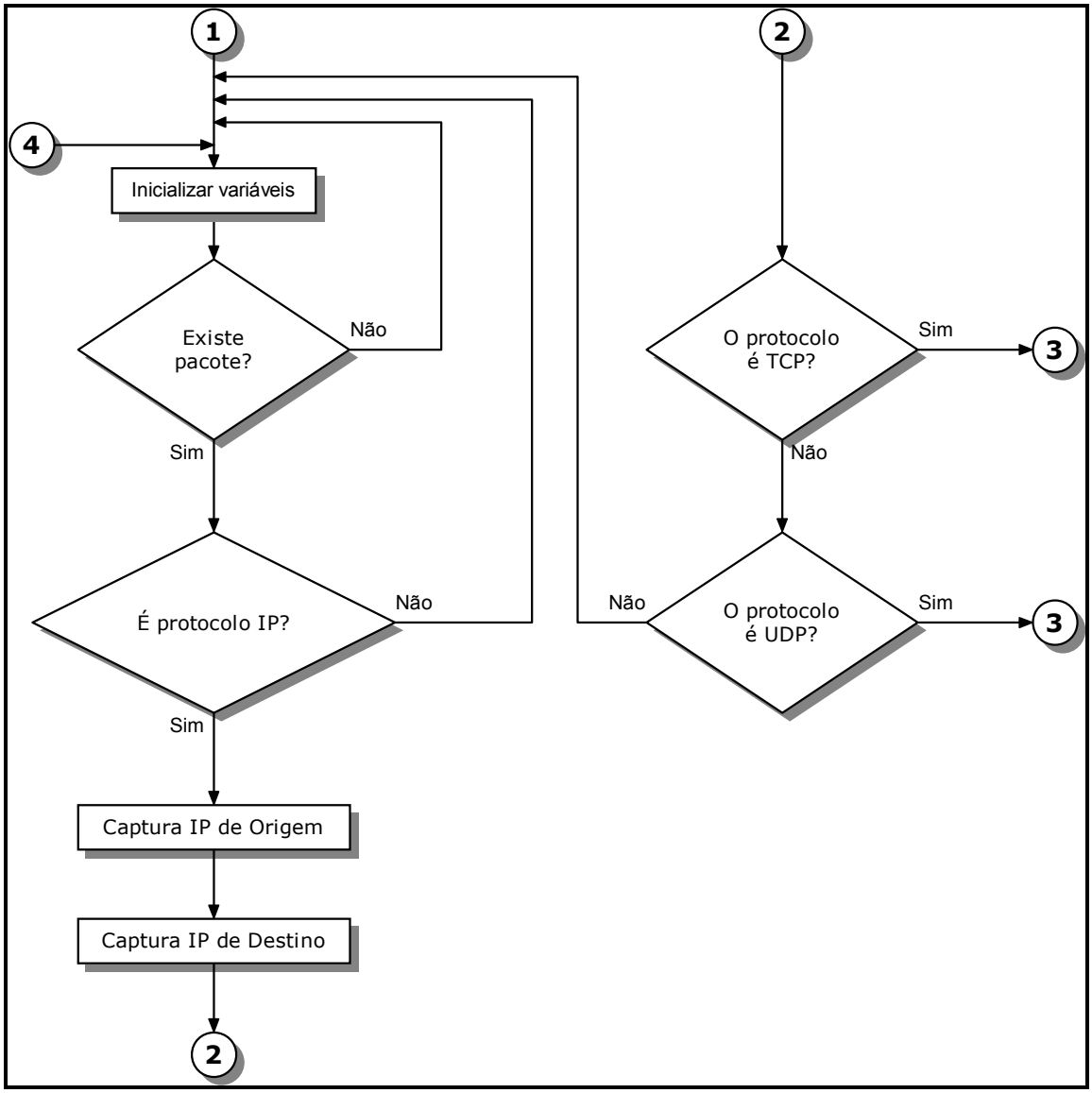


FIGURA 5 – Monitoração dos protocolos

Quando um pacote TCP é identificado, o mesmo é tratado para que se obtenha o conteúdo dos pacotes. Se este conteúdo for referente ao protocolo HTTP, os dados são tratados e apresentados ao usuário. Se o protocolo identificado for o UDP, as únicas informações que serão mostradas são os IP´s de origem e de destino. A representação macro deste processo, está ilustrada na fig. 7.

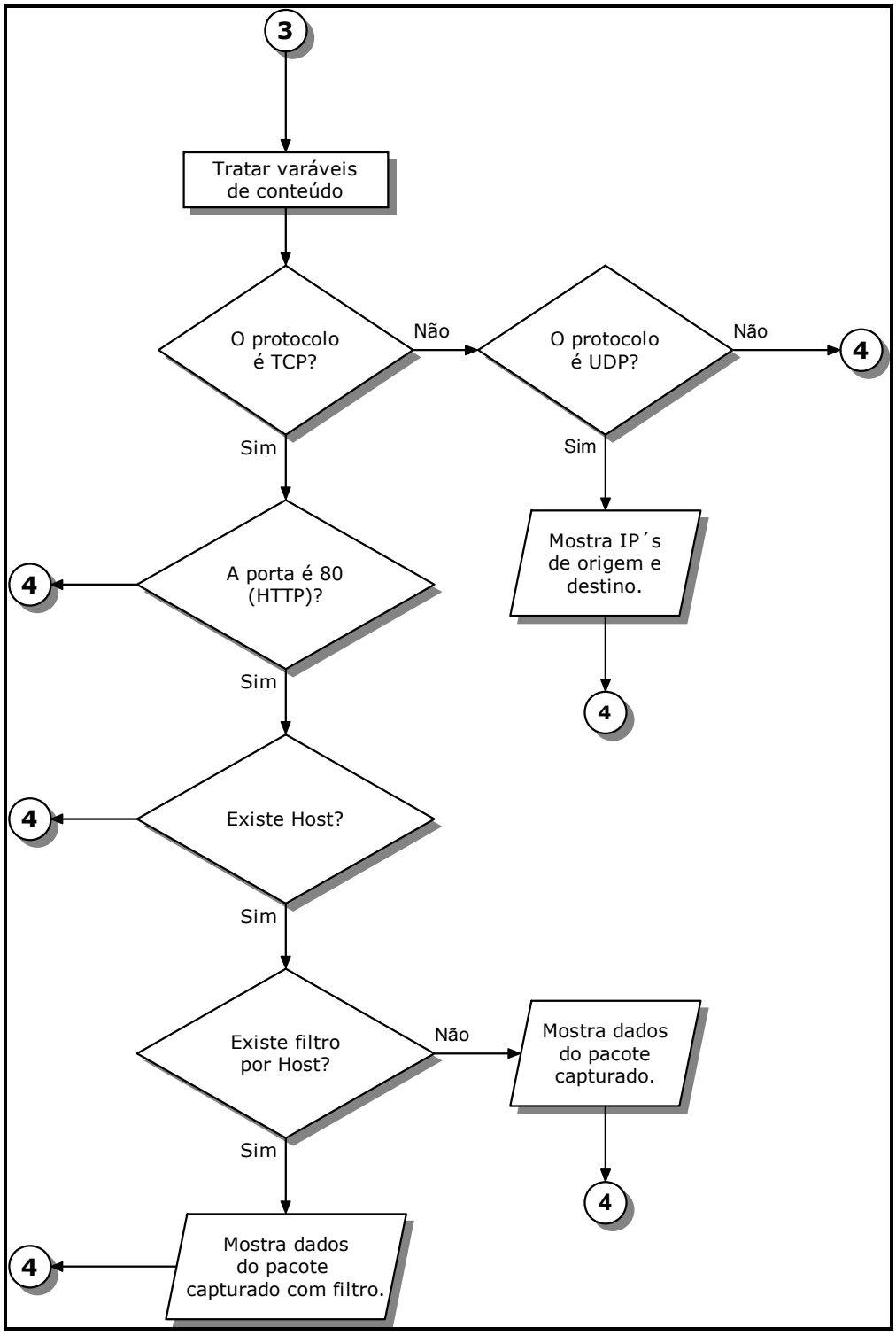


FIGURA 6 – Apresenta conteúdo capturado

5.3 IMPLEMENTAÇÃO

Nos itens seguintes serão abordadas as técnicas e ferramentas utilizadas para a elaboração e desenvolvimento deste trabalho e detalhadamente o funcionamento do protótipo.

5.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

Para a implementação do protótipo foi utilizada a linguagem visual de programação Borland C++ Builder 6.0 baseando-se na biblioteca *WinPcap* escrita por *Politecnico di Torino* (2003) e o componente baseado na biblioteca *WinPcap*, *TJesNetMonitor* escrito por Reichler (2002).

5.3.1.1 BORLAND C++

Segundo descrito por Dias (2000), Borland C++ é uma linguagem visual descendente do C que possibilita o uso e criação de novas bibliotecas de funções. Por se tratar de uma ferramenta que utiliza os conceitos de orientação a objetos como definições de classes e objetos, onde as variáveis e funções são agrupadas dentro das classes, é que o Borland C++ foi escolhido para ser utilizado na implementação deste protótipo.

A facilidade de um ambiente gráfico, que o Borland C++ incluí em seu funcionamento, foi fundamental para a modelagem do protótipo, assim disponibilizando um ambiente de interação de fácil entendimento por parte do usuário.

5.3.1.2 WINPCAP

WinPcap, é uma biblioteca para acesso a interface de rede baseadas na plataforma Windows. Ela implementa o acesso direto ao tráfego da rede, sem a intermediação das entidades relacionadas, como a pilha de protocolos.

As funcionalidades que a biblioteca fornece, são:

a) captura de pacotes, diretamente do adaptador de rede, destinados ao microcomputador onde está instalada;
b) filtro de pacotes de acordo com as regras pré-estabelecidas pelo usuário;
c) transmite pacotes diretamente à rede;
d) análise de valores estatísticos no tráfego da rede.

WinPcap pode ser usada por diferentes tipos de aplicação de análise de rede, pesquisando falhas na segurança e monitorando-a.

Aplicações que podem se beneficiar das vantagens da biblioteca são:

a) analisadores de rede e de protocolos;

b) monitores de rede;
c) *loggers* de tráfego;
d) geradores de tráfego;
e) pontes e *routers* em nível de usuário;
f) sistemas de detecção de intrusos da rede;
g) "varredores" da rede;
h) ferramentas de segurança.

São duas as bibliotecas responsáveis pelo filtro de pacotes, sendo a de nível mais baixo (*packet.dll*) responsável pela ligação dinâmica, ou seja, pode ser usada para acessar diretamente as funções do *driver* com uma programação de interface independente do sistema operacional Windows e uma de alto nível independente do adaptador de rede e do sistema operacional (*wpcap.dll*), conforme representado na fig. 8.

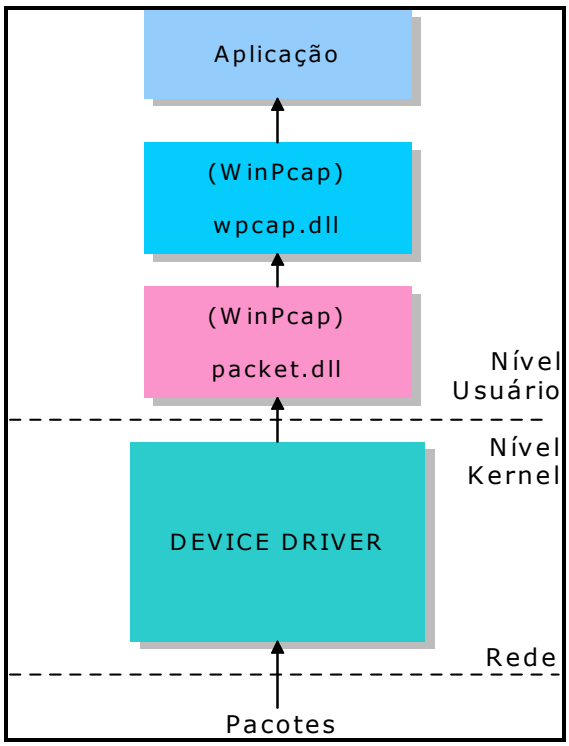


FIGURA 7 – Arquitetura da biblioteca *WinPcap*

5.3.1.3 TJESNETMONITOR

O *TJesNetMonitor*, é um componente que encapsulou as funcionalidades e facilidades da biblioteca *WinPcap*. Para este protótipo, foram usados alguns métodos de acesso ao adaptador de rede, bem como a captura de pacotes.

Os métodos utilizados são:

a) open: inicializa o monitor de rede e lista os adaptadores de rede disponíveis. Este método deve ser utilizado apenas uma vez pela aplicação antes da captura dos pacotes;
b) close: fecha o monitor de rede e deve ser utilizado apenas uma vez antes de sair da aplicação;
c) selectadapter: seleciona o adaptador a ser utilizado;
d) get_adaptername: retorna o adaptador selecionado;
e) startcapture: inicia a captura dos pacotes;
f) stopcapture: para a captura de pacotes;
g) notifythreadstart: responsável pela chama do método TakeTimeToProcess, responsável pelo "tempo" de checagem por novos pacotes;
h) notifythreadend: finaliza ou suspende a notificação de novos pacotes;
i) get_packetcount: retorna o número de pacotes que estão esperando no *buffer* local;
j) getnextpacket: retorna o próximo pacote da fila em espera;
k) get_ppacket: ponteiro para o conteúdo do pacote.

5.3.2 O PROTÓTIPO

A primeira função a ser executada é a de verificação da existência do adaptador de rede e se é possível estabelecer conexão com o mesmo.

Para que o protótipo funcione corretamente, é necessário que o administrador, caso exista um, se "logue" a primeira vez no sistema operacional, porque a biblioteca *WinPcap* só é instanciada quando o procedimento é feito pelo administrador. Depois deste processo, qualquer usuário que estiver utilizando o sistema, poderá executar o protótipo.

Caso não exista um administrador no sistema operacional, qualquer usuário que for utilizar o microcomputador poderá executar o protótipo normalmente.

A primeira etapa do funcionamento do protótipo é mostrada no quadro 3.

```
__fastcall Tfrm_Monitor::Tfrm_Monitor(TComponent* Owner): TForm(Owner)
{
  Parado = true;
  Iniciou = false;
  bb_Iniciar->Enabled = true;
  bb_Parar->Enabled = false;
  bb_Limpar->Enabled = true;
  AtualizaTempo->Interval = REFRESH_INTERVAL_S * 1000; // 1 segundo
  if (!MonitorAplicacao->Open())
     BarraStatus->SimpleText = "O Monitor de Aplicação não pode ser
                               inicializado!";
  else if (!MonitorAplicacao->SelectAdapter(0))
     BarraStatus->SimpleText = "Adaptador de rede não pode ser
                               selecionado!";
  else
     edt_Adaptador->Text = MonitorAplicacao->get_adaptername(0);
}
```

QUADRO 3 – Estabelecer conexão com o adaptador de rede

Após a execução do procedimento de inicialização do adaptador de rede, o protótipo está pronto para iniciar a monitoração dos pacotes. Ao ser iniciado a captura, o procedimento entra em estado de execução, e fica por tempo indeterminado, até que o usuário pare o procedimento, executando a captura dos pacotes que trafegam pela rede. Caso as condições sejam atendidas, o código listado no quadro 4, é responsável pela captura dos pacotes e pelos filtros de IP de origem e destino assim como os protocolo TCP e UDP.

```
void __fastcall Tfrm_Monitor::CapturaPacote(TObject *Sender)
{
  BYTE *ptr;
  BYTE *ipAddr;
  ETHERNET_FRAME *ePtr;
  char IP_Origem[20], IP_Destino[20];
  char EndOrigem[25], EndDestino[25];
  char PortaOrigem[6], PortaDestino[6];
  TJesNetPacket Pacote;
  while (MonitorAplicacao->get_packetcount() > 0)
  {
    short int etherHdrLen, ipHdrLen, tcpHdrLen, udpHdrLen;
    MonitorAplicacao->GetNextPacket(&Pacote);
    ptr = Pacote.get_pPacket();
    ePtr = (ETHERNET_FRAME *) ptr;
    if (XCHG(ePtr->FrameType) == 0x0800) // Verifica se o protocolo é IP
    {
      {
        etherHdrLen = 14; // Tamanho do cabeçalho Ethernet
        IP_HEADER *ipPtr = (IP_HEADER *) &ptr[etherHdrLen];
        ipHdrLen = (ipPtr->x & (BYTE) 0x0f) << 2;
        ipAddr = (BYTE *) &ipPtr->src;
        sprintf(IP_Origem, "%d.%d.%d.%d", ipAddr[0], ipAddr[1], ipAddr[2],
                           ipAddr[3]); // Identifica IP de origem
        ipAddr = (BYTE *) &ipPtr->dest;
        sprintf(IP_Destino, "%d.%d.%d.%d", ipAddr[0], ipAddr[1], ipAddr[2],
                            ipAddr[3]); // Identifica IP de destino
        if (ipPtr->protocol == 0x06) // Verifica se o protocolo é TCP
        {
          TCP *pTCP = (TCP *) &ptr[etherHdrLen + ipHdrLen];
          tcpHdrLen = TCP_HdrLen(pTCP); // Tamanho do cabeçalho TCP
          sprintf(EndOrigem, "%s:%d", IP_Origem, TCP_SrcPort(pTCP));
          sprintf(PortaOrigem, "%d", TCP_SrcPort(pTCP));
          sprintf(EndDestino, "%s:%d", IP_Destino, TCP_DstPort(pTCP));
          sprintf(PortaDestino, "%d", TCP_DstPort(pTCP));
          frm_Monitor->Append(ipPtr->protocol, EndOrigem, EndDestino,
                              PortaOrigem, PortaDestino,
                              &ptr[etherHdrLen + ipHdrLen + tcpHdrLen]);
        }
        else if (ipPtr->protocol == 0x11) // Verifica se o protocolo é UDP
        {
          UDP_HEADER *pUDP = (UDP_HEADER *) &ptr[etherHdrLen + ipHdrLen];
          udpHdrLen = 8; // Tamanho do cabeçalho UDP
          sprintf(EndOrigem, "%s:%d", IP_Origem, XCHG(pUDP->src_port));
          sprintf(PortaOrigem, "%d", XCHG(pUDP->src_port));
          sprintf(EndDestino, "%s:%d", IP_Destino, XCHG(pUDP->dest_port));
          sprintf(PortaDestino, "%d", XCHG(pUDP->dest_port));
          frm_Monitor->Append(ipPtr->protocol, EndOrigem, EndDestino,
                              PortaOrigem, PortaDestino,
                              &ptr[etherHdrLen + ipHdrLen + udpHdrLen]);
        }
      }
    }
  }
}
```

QUADRO 4 – Captura de pacotes e filtros IP, TCP e UDP

Assim que os dados são capturados e filtrados, são enviados a uma outra função que tem o papel de apresentar estes dados capturados de forma fácil de se compreender. Podemos visualizar está funcionalidade no quadro 5.

```
switch (prot) {
case 0x06:
  if (!frm_Monitor->cb_TCP->Checked)
     return;
  if ((PortaDestino == 80) && (HOST > "")) {
     if ((Filtro) || (edt_FiltrarHost->Text == "")) {
       frm_Monitor->lv_Protocolos->Items->Add();
       aux = frm_Monitor->lv_Protocolos->Items->Count - 1;
       frm_Monitor->lv_Protocolos->Items->Item[aux]->Caption = aux;
       frm_Monitor->lv_Protocolos->Items->Item[aux]->SubItems->Add("TCP");
       frm_Monitor->lv_Protocolos->Items->Item[aux]->SubItems->Add(EndOrigem);
       frm_Monitor->lv_Protocolos->Items->Item[aux]->SubItems->Add(EndDestino);
       frm_Monitor->lv_Protocolos->Items->Item[aux]->SubItems->Add(HOST);
       frm_Monitor->lv_Protocolos->Items->Item[aux]->SubItems->Add(GET);
       frm_Monitor->lv_Protocolos->Items->Item[aux]->Update(); } } break;
case 0x11:
  if (!frm_Monitor->cb_UDP->Checked)
     return;
  frm_Monitor->lv_Protocolos->Items->Add();
  aux = frm_Monitor->lv_Protocolos->Items->Count - 1;
  frm_Monitor->lv_Protocolos->Items->Item[aux]->Caption = aux;
  frm_Monitor->lv_Protocolos->Items->Item[aux]->SubItems->Add("UDP");
  frm_Monitor->lv_Protocolos->Items->Item[aux]->SubItems->Add(EndOrigem);
  frm_Monitor->lv_Protocolos->Items->Item[aux]->SubItems->Add(EndDestino);
  frm_Monitor->lv_Protocolos->Items->Item[aux]->SubItems->Add(HOST);
  frm_Monitor->lv_Protocolos->Items->Item[aux]->SubItems->Add(GET);
  frm_Monitor->lv_Protocolos->Items->Item[aux]->Update();
  break;
default:
  return; }

if (len) {
   if ((Filtro) || (edt_FiltrarHost->Text == "") || ((auxPR == "HTTP") &&
       (StrPos(NomFiltro.c_str(), edt_FiltrarHost->Text.c_str())))) {
      if (auxPR == "GET ") {
        frm_Monitor->memo_Origem->Lines->Append("------> " + AnsiString(aux));
        frm_Monitor->memo_Origem->Lines->Append(asAux);
      else if (auxPR == "HTTP")
        frm_Monitor->memo_Destino->Lines->Append("--------------------------");
        frm_Monitor->memo_Destino->Lines->Append(asAux); } } }
Filtro = false; }
```

QUADRO 5 – Visualização do conteúdo capturado

5.3.3 OPERACIONALIDADE DA IMPLEMENTAÇÃO

A simples utilização do Monitor de Aplicações, o torna uma ferramenta de acesso rápido e eficaz aos dados capturados. O protótipo apresenta apenas uma tela ao usuário, de forma que o mesmo tenha todas a informações necessárias de uma só vez.

A visualização do protótipo pode ser observada na fig. 9.

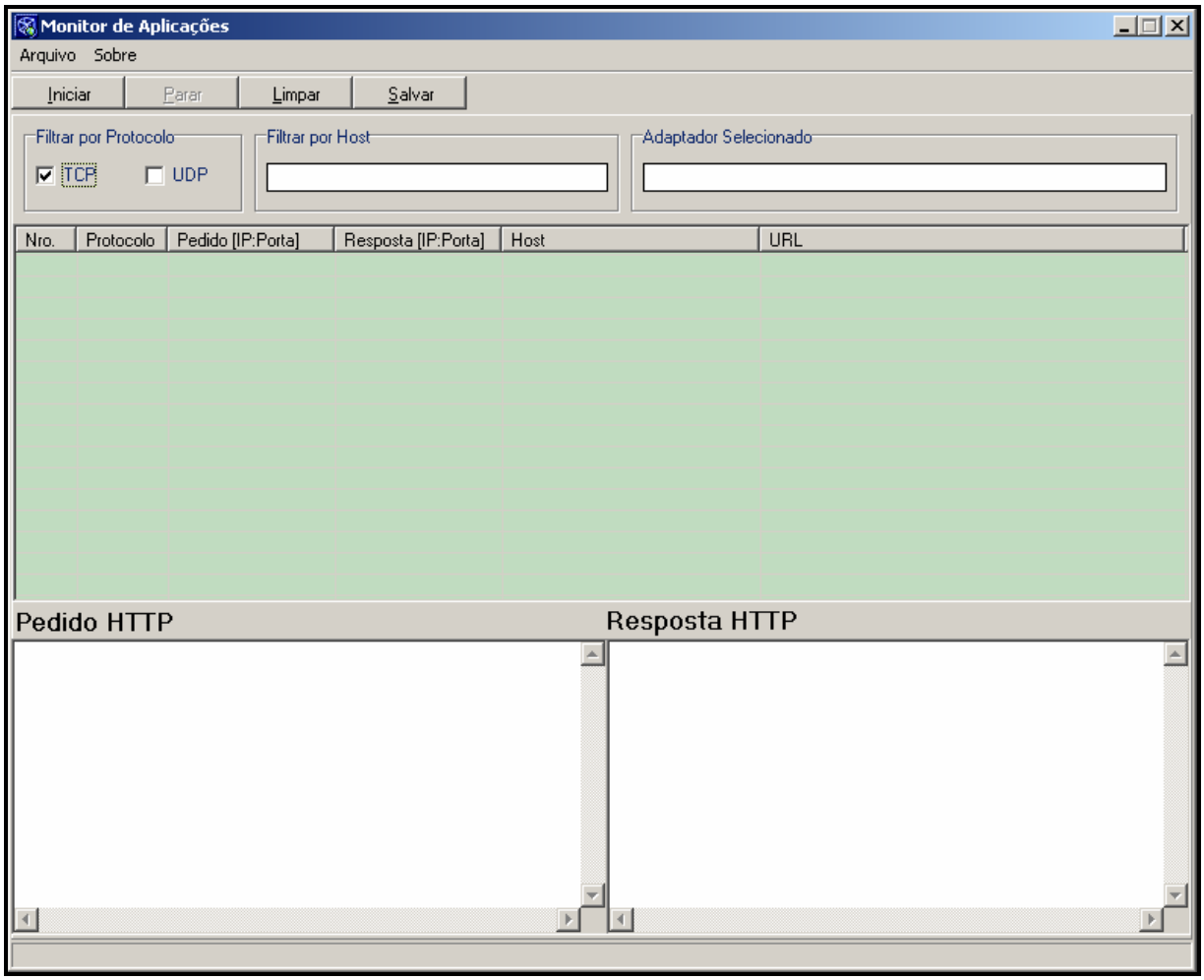


FIGURA 8 – Botões de execução do protótipo

Os botões que estão localizados no canto superior esquerdo são responsáveis pela execução do protótipo, sendo eles (fig. 10):

a) iniciar: inicia a captura de todos os pacotes que estão trafegando pela rede;
b) parar: permite que o protótipo interrompa o processo de visualização dos pacotes, mas não encerra o processo de captura destes pacotes;
c) limpar: limpa o conteúdo das visualizações apresentadas ao usuário;
d) salvar: armazena o conteúdo monitorado, para uma futura consulta.

FIGURA 9 – Botões de execução do protótipo

Logo abaixo dos botões de execução, estão listados os possíveis filtros por protocolo. Se o protocolo estiver marcado, o protótipo irá interceptar os dados referentes a este

protocolo. Estes filtros podem ser ativados ou desativados durante a execução do programa conforme a fig. 11.

FIGURA 10 – Filtro por protocolo

A caixa de texto, que pode ser visualizada na fig. 12, que está ao lado dos filtros por protocolo, pode ser usada para monitorar aplicações HTTP que atendam a um determinado *site* ou mesmo a uma palavra específica. Se o endereço completo do *site* for digitado, o monitor passa a apresentar apenas o conteúdo referente a este *site*. Caso o usuário preencha a caixa de texto com uma determinada palavra, todo o pacote que contiver a palavra digitada, será apresentado ao usuário (fig. 12).

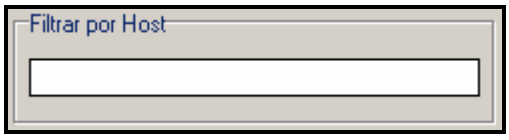


FIGURA 11 – Filtro por *Host*

E ainda, a caixa de texto mais à direita localizada no canto superior, mostra ao usuário, o adaptador de rede que foi selecionado, caso exista (fig. 13).

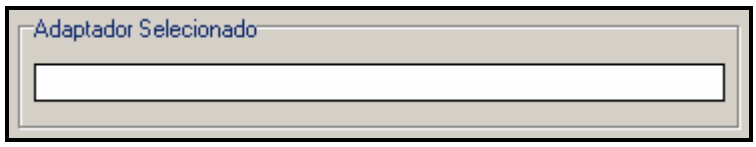


FIGURA 12 – Adaptador de rede selecionado

Os pacotes capturados pelo protótipo são mostrados ao usuário, conforme eles são capturados. Depois de tratados pelo sistema, são apresentados ao usuário de forma simples de se entender.

As colunas e campos listados ao usuário são:

a) nro.: apresenta a seqüência em que os pacotes são capturados;
b) protocolo: identifica o protocolo referente aquela captura;
c) pedido [ip:porta]: identifica qual foi o IP que originou o pedido HTTP e sua porta correspondente (cliente);

d) resposta [ip:porta]: identifica o IP que respondeu ao pedido HTTP e sua porta (servidor);
e) *host*: identifica o *site* que está tendo seus dados monitorados;
f) url: apresenta em detalhes todos os arquivos que serão armazenados na máquina local, com sua estrutura de diretórios do servidor.
g) pedido HTTP: são os dados requisitados pelo cliente ao servidor;
h) resposta HTTP: é a resposta do servidor ao cliente, notificando se houve atualização no pacote ou não, e se o mesmo pode ser enviado.

Estas colunas e campos estão apresentados na fig. 14.

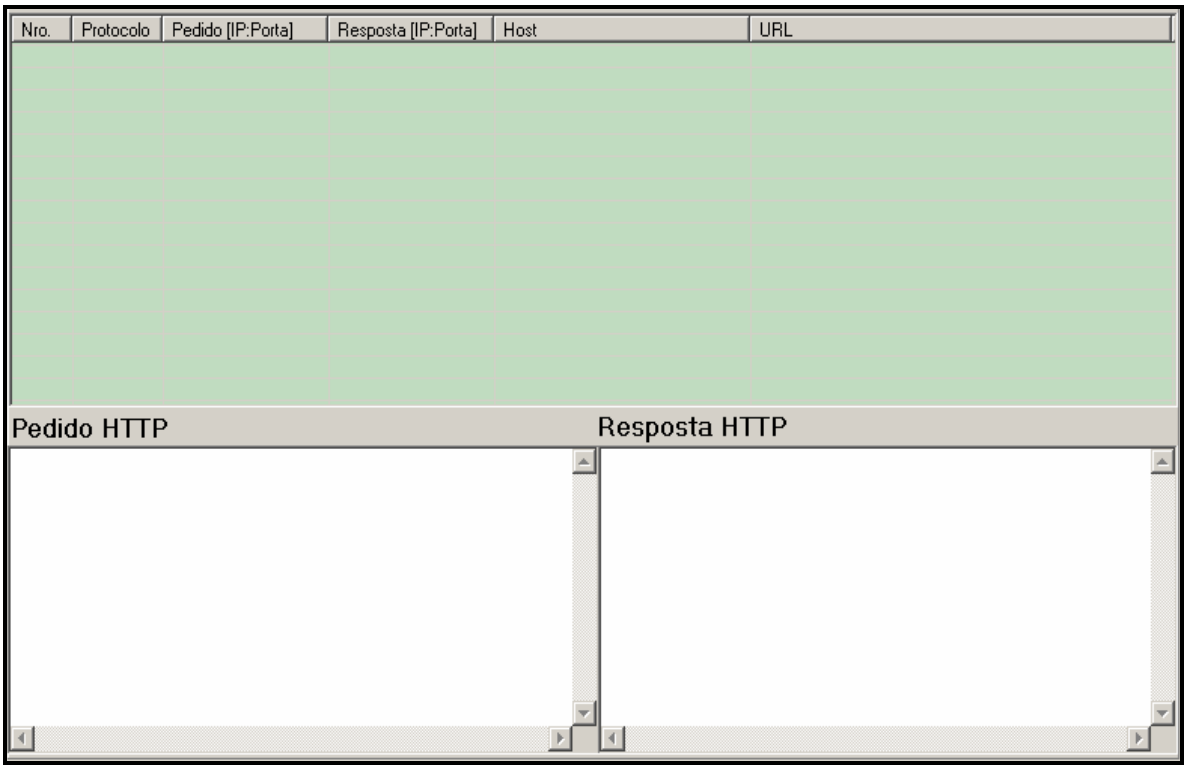


FIGURA 13 – Visualização dos pacotes capturados e tratados.

6 CONCLUSÕES

Ao longo do desenvolvimento deste trabalho, foi possível constatar a grande preocupação que se tem desprendido em torno da segurança de redes. Não se tem segurança sem uma política de segurança, envolvendo regras que possam garantir o bom funcionamento de uma rede assim como garantir a integridade das informações que por ela trafegam.

São inúmeras as técnicas utilizadas para a exploração de falhas em sistemas. Mas a maior parcela dos problemas ocorridos em relação à segurança das redes, são os próprios usuários, que de dentro da própria empresa onde trabalham, já conhecendo as falhas na segurança, se apropriam de informações confidenciais.

O protótipo desenvolvido neste trabalho demonstrou ser uma ferramenta de monitoração eficaz para redes *Ethernet*, utilizando a estrutura da arquitetura TCP/IP, sendo capaz de monitorar, extrair e armazenar informações do cabeçalho do protocolo HTTP, úteis para análise do administrador, podendo tomar decisões mais rápidas e eficazes quanto a um possível ataque.

Nesta pesquisa, foi possível aplicar na prática, conceitos que até então eram vistos apenas em livros. Ao implementar o protótipo, foi possível entender o funcionamento dos protocolos, bem como sua forma de encapsulamento dos dados e a forma como estão disponibilizadas as informações dentro do mesmo.

A utilização da linguagem de programação C++ mostrou-se adequada para o desenvolvimento do protótipo, por tratar de forma mais abstrata a implementação dos conceitos da arquitetura TCP/IP e por utilizar a *WinPcap*, uma biblioteca que tem como principal objetivo, acessar diretamente o adaptador rede, sem a intervenção do sistema operacional, para que os pacotes possam ser tratados pelo protótipo. E o componente *TJesNetMonitor*, que tem algumas funcionalidades da *WinPcap*, encapsuladas, tornando mais fácil o desenvolvimento.

As redes de computadores são totalmente seguras, até que ocorra a primeira invasão. Não se pode esperar que isso aconteça, para que se estabeleçam regras capazes de definir restrições no uso da rede.

6.1 EXTENSÕES

Em relação a futuros trabalho que possam tomar como base o Monitor de Aplicações, poderiam ser desenvolvidos trabalhos para:

a) implementar os filtros para os outros protocolos da camada de aplicação, como: SMTP de correio eletrônico, FTP de transferência de arquivos;
b) um sistema que consiga filtrar protocolos seguros, como SSL;
c) um sistema de bloqueios a páginas de internet ou endereços de e-mail, com base nas informações monitoradas;
d) a continuidade deste trabalho, visando a criação de listas de filtros, controles mais amplos sobre as camadas referentes à arquitetura TCP/IP.

6.2 RESULTADOS E DISCUSSÃO

O funcionamento do protótipo é simples e trás informações importantes para que um administrador de sistemas ou mesmo um usuário comum, possa identificar quais são os *sites* que estão sendo “visitados”.

Diferente de trabalhos anteriores, tais como Pompermayer (2002) e Silva (2001), os quais fazem a monitoração de pacotes IP, com objetivo de verificar o tráfego da rede em relação aos endereços da camada de rede e de transporte, este protótipo, além de monitorar pacotes, extraí do cabeçalho HTTP informações que são relevantes para a segurança de uma rede de computadores.

A captura é feita de forma simples e rápida, onde o conteúdo monitorado é mostrado em tempo real, possibilitando uma ação mais rápida em caso de suspeita de invasão do sistema. São informações que podem ser salvas para uma futura análise, de como os usuários estão utilizando a internet. Um exemplo de monitoração do cabeçalho HTTP pode ser visualizado na fig. 14.

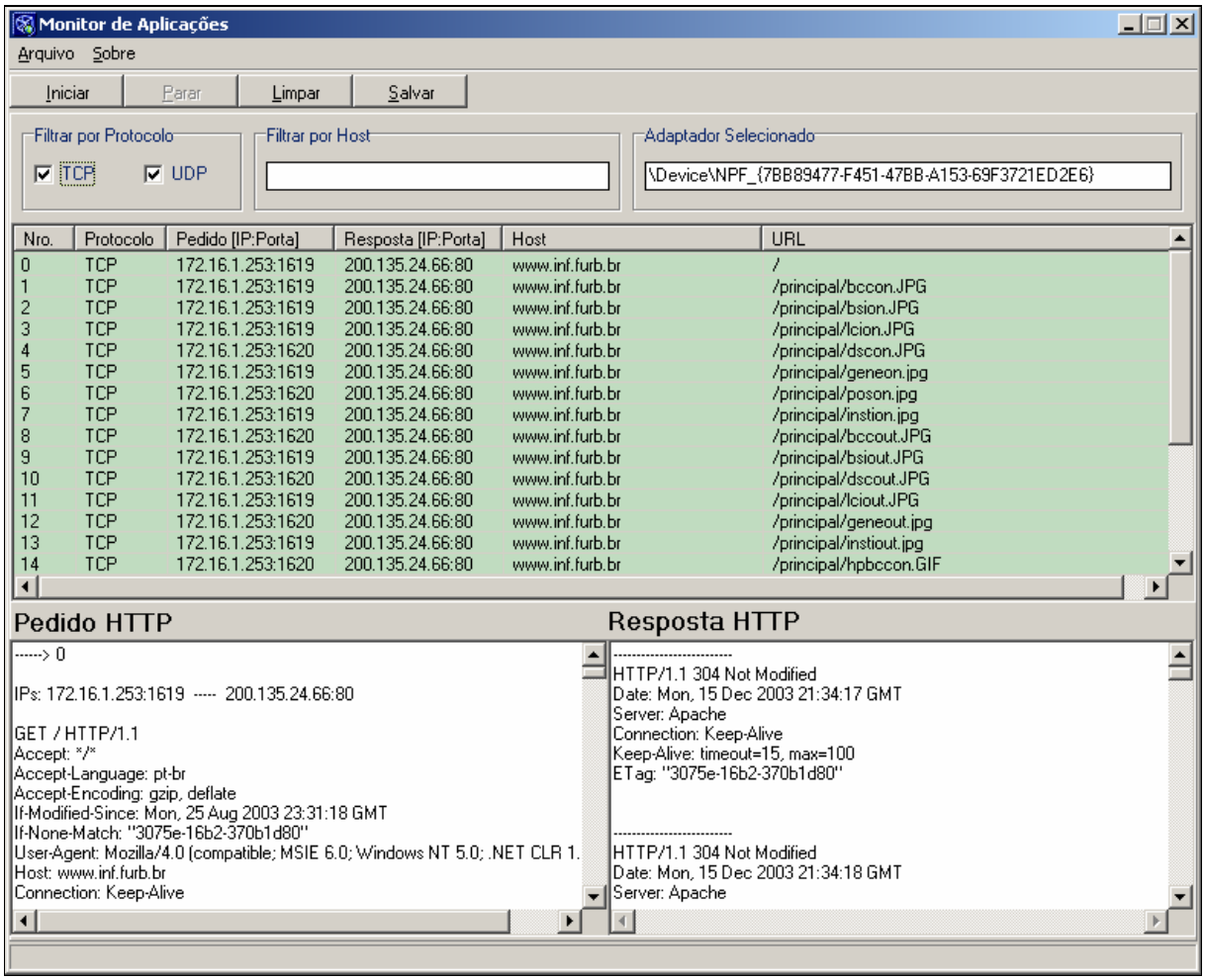


FIGURA 14 – Protótipo em execução

7 REFERÊNCIAS BIBLIOGRÁFICAS

BERNSTEIN, Terry; et al. **Segurança na internet**. Rio de Janeiro: Campus, 1997.

COMER, Douglas E. **Interligação em rede com TCP/IP**. Rio de Janeiro: Campus, 1998.

DEMARCO, Tom. **Análise estruturada e especificação de sistema**. Rio de Janeiro: Campus, 1989.

DIAS, Adilson de Souza. **Desenvolvendo em Borland C++ Builder 5.0**. São Paulo: Ciência Moderna, 2000.

KUROSE, James F. **Redes de computadores e a internet:** uma nova abordagem. Tradução de Arlete Simille Marques. São Paulo: Pearson Brasil, 2003.

MCCLURE, Stuart; SCAMBRAY, Joel; KURTZ, George. **Hackers expostos:** segredos e soluções para a segurança de redes. São Paulo: Makron Books, 2000.

OLIVERIA, Wilson José de; **Hacker:** invasão e proteção. Florianópolis: Visual Books, 2000.

PALMA, Luciano; PRATES Rubens. **TCP/IP:** guia de consulta rápida. São Paulo: Novatec, 2000.

POMPERMAYER, Jorge L. **Protótipo de software para a monitoração de pacotes em uma rede TCP/IP em ambiente Linux**. 2002. 60 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

REDES, CURSO DE. **Trabalho sobre as camadas de rede.** Proença, [2000?]. Disponível em: <http://proenca.uel.br/curso-redes-graduacao/1998/trab-08>. Acesso em: 16 nov. 2003.

REICHLER, Jesse, **TJESNETMONITOR**, [2002]. Disponível em: <http://delcomyn2.life.uiuc.edu/~reichler/TJesComponents/>. Acesso em: 16 nov 2003.

RFC 822. **Internet protocol**, Califórnia, [2003]. Disponível em: <http://www.netsys.com/rfc/rfc791.txt>. Acesso em: 16 nov 2003.

SILVA, Paulo Fernando da. **Protótipo de software de segurança em redes para a monitoração de pacotes em uma conexão TCP/IP**. 2001. 110 f. Trabalho de Conclusão de

Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SOFTWARE, PACESTAR. ***EDGE DIAGRAMER***, nov. 2003. Disponível em: <http://www.pacestar.com>. Acesso em: 16 nov. 2003.

STARLIN, Gorki. **Manual completo do *Hacker*:** como ser e evitá-los. Rio de Janeiro: Book Express, 1999.

TORINO, POLITECNICO DI. **WINPCAP**, set. 2003. Disponível em: <http://winpcap.polito.it>. Acesso em: 16 nov. 2003.