Marcelo Leal de Araújo Barrêto

Protótipo de uma Aplicação de Autenticação Web para Redes *Wireless* para os Órgãos Públicos do Estado do Tocantins utilizando *FOSS*

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Orientador
Prof. Sandro Melo

Lavras
Minas Gerais - Brasil
2005

Marcelo Leal de Araújo Barrêto

Protótipo de uma Aplicação de Autenticação Web para Redes *Wireless* para os Órgãos Públicos do Estado do Tocantins utilizando *FOSS*

Monografia de Pós-Graduação "*Lato Sensu*" apresentada ao Departamento de Ciência da Computação para obtenção do título de Especialista em "Administração em Redes Linux"

Aprovada em *10 de Dezembro de 2005*

Prof. Fernando Silva Lozano

Prof. Heitor Augustus Xavier Costa

Prof. Sandro Melo
(Orientador)

Lavras
Minas Gerais - Brasil

Agradecimentos

Agradeço a Deus por ter me dado forças e meios para desenvolver este projeto, à minha família, amigos e ao professor Sandro Melo pela orientação no trabalho.

Resumo

Este trabalho tem como objetivo propor a implementação de um mecanismo de autenticação em redes de computadores a ser aplicado em redes *Wireless*. Para isso foi feito um estudo nas tecnologias de *Software Livre* e a implementação de um protótipo de uma aplicação de autenticação web para Redes *Wireless* para os Órgãos Públicos do Estado do Tocantins utilizando *FOSS*, criando assim, uma alternativa de autenticação de usuários baseada em *Software Livre*.

Este trabalho é dedicado às pessoas que fazem uso do Software Livre como instrumento para promover o crescimento intelectual coletivo no meio ao qual está inserido, seja ele uma megalópole ou uma aldeia indígena.

Sumário

Lista de Figuras

Capítulo 1

Introdução

A partir dos avanços tecnológicos observados no campo das telecomunicações, o advento da Internet e o uso cada vez mais freqüente dos computadores no meio social, acadêmico e comercial influenciou o uso de uma extensa gama de aplicações do campo da informática nos negócios e na vida pessoal dos brasileiros. As redes de computadores fornecem dentre outros, vários serviços como comunicação entre os computadores, troca de mensagens, compartilhamento de arquivos, correio eletrônico e acesso a Internet.

As formas de comunicação dos computadores vêm evoluindo ano após ano, e hoje é notável o crescimento da comunicação via rádio. As redes via rádio destinadas a fazer o papel das redes locais são denominadas Redes *Wireless*. Conhecidas como comunicação sem fio, as redes *Wireless* constituem uma forma flexível e passível de mudança contribuindo assim, para o barateamento na implantação de redes de computadores, pois em alguns casos dispensa o uso de infra-estrutura de cabeamento estruturado. Contudo, com o crescimento do uso das redes *Wireless*, cresce também a preocupação com a segurança dos ambientes computacionais, culminando assim, na busca e criação de mecanismos que auxiliam no fornecimento da segurança dessas redes.

O presente projeto tem como tema o Protótipo de uma Aplicação de Autenticação Web para Redes *Wireless* para os Órgãos Públicos do Estado do Tocantins utilizando *FOSS*[1] que consiste no estudo da utilização de um conjunto de ferrametas baseadas em *Software Livre*, e na implementação de uma aplicação web, que tem como objetivo criar uma "camada" de autenticação de usuários a ser usada em redes *Wireless*.

A idéia principal é criar um mecanismo de autenticação de usuários baseado na implementação de um controle de acesso aos serviços de Internet. Este mecanismo

[1] *FOSS - Free And Open Souce Software* - Uso de *Software Livre* e *Software de Código Aberto*

de controle redireciona a tentativa de conexão oriunda da rede *Wireless* para uma área de autenticação no servidor, e em seguida verifica-se o *login* e senha dos usuários cadastrados. Os usuários autenticados são liberados para uso dos recursos de Internet de acordo com o perfil disponível. Já os usuários não cadastrados são descartados.

1.1 Motivação

O desenvolvimento de uma solução baseada em *Software Livre* para promover a autenticação de usuários em redes *wireless* teve seu ponto inicial pautado no problema de gastos com cabeamento estruturado em órgãos que constantemente mudam de endereço.

No Estado do Tocantins, em virtude de sua recente criação e não obstante o grande desenvolvimento alcançado ao longo dos anos, muitos órgãos do Poder Executivo e do Poder Judiciário Estadual não possuem sede própria, situação esta presente tanto no interior (sedes de Comarca e representações do Governo nos municípios), como na própria Capital (anexos das secretarias do Governo), valendo-se a Administração Pública da locação de prédios comercais e até mesmo residenciais para o abrigo desses órgãos. Neste contexto, um dos problemas encontrados é que os imóveis objeto de locação, na maioria das vezes, para não falar na totalidade, padecem de estrutura física de rede, tal como tubulação e cabeamento, necessária à implantação da rede física de dados, exigindo-se assim, o dispêndio de vultosas quantias nessa área por parte desses órgãos, com vistas a adequar o prédio ao fim a que se destina.

Nesse interim, observa-se a freqüente repetição dos gastos quando os contratos de locação expiram, sem renovação, uma vez que exsurge daí a necessidade de locação de um novo prédio e consequentemente, de novos investimentos com vistas à adequação fisica do ambiente computacional.

Objetivando a contensão de gastos com a adequação física para a rede de computadores, foi implantado em alguns órgãos[2] do Estado do Tocantins, o uso de redes *Wireless*, pois desta forma, quando houvesse necessidade da mudança de endereço, não haveria novamente o trabalho e os gastos com estruturas físicas, cabeamentos e tubulações, por se tratar de equipamentos móveis.

Um dos fatores críticos de sucesso para o uso de redes *Wireless* no interior do Estado está relacionado à falta de capacitação profissional para a configuração da segurança nestes equipamentos. Isso porque no primeiro momento, o objetivo inicial é levar a estrutura básica de conectividade, sendo que a preocupação com

[2]Por questões de privacidade os nomes dos órgãos não serão mencionados

questões de segurança e questões técnicas relacionadas à configuração refinada dos recursos dos equipamentos, são ofuscadas pela necessidade básica inicial, que é a de interligar os computadores da rede, ficando a segurança dessa forma, em segundo plano a ser implementada posteriormente.

O uso de criptografia, identificação de um computador por endereços MAC[3] e de ocultação de identificador são alguns dos exemplos de técnicas que visam inibir que usuários mal intencionados possam usufruir do sinal de rede *Wireless*.

Nota-se que em ambientes de rede *Wireless* desprotegidos é possivel, a uma determinada distância, ter acesso, via sinal de rádio, aos recursos do ambiente computacional que vão desde acessos aos computadores até acesso a Internet. As práticas mais comuns dos usuários mal intencionados para acessarem a Internet através de sinais de rede *wireless* são: a prática de *Wardriving*, que detecta o sinal de rede *Wireless* através de um *notebook* com uma placa de rede *Wireless* e posteriomente consegue o acesso a rede; e a prática denominada de *Warchalking* que consiste na demarcação de pontos de acesso a redes sem fios e desenho de símbolos no local, indicando a presença e o estado de uma rede sem fio, permitindo que outros utilizem aquele ponto.

Não obstante os problemas de segurança com o uso das redes *Wireless* e a falta de conhecimento técnico especializado dentro do cenário apresentado, observa-se que as tecnologias de rede sem fio como *Wi-Fi*, infravermelho, *bluetooth e Wi-Max* estão cada dia mais presentes em aparelhos como controle remoto, celulares, *notebooks*, PDAs, mouses e aparelhos de som.

Observa-se também que a comunicação sem fio em ambientes computacionais tem crescido e que as redes *Wireless*, em alguns casos, constituem em um alterntativa atraente no que diz respeito à relação custo-beneficio, flexibilidade e conforto ao usuário.

Considerando a preocupação com a segurança do ambiente computacional nos órgãos públicos do Estado do Tocantins e na busca de soluções alternativas de baixo custo à utilização de produtos proprietários é que se faz necessária a criação de mecanismos de autenticação de usuários em redes *Wireless*.

Diante das limitações expostas e da necessidade de implantação da segurança das redes dos anexos que usam tecnologia *Wireless*, foi proposto pelo Centro Operacional de Redes do Poder Executivo em conjunto com o Centro Operacional de Redes do Poder Judiciário a criação de uma alternativa de baixo custo baseada em *Software Livre* com intuito de fornecer um nível de segurança com base na autenticação de usuários.

A partir da observação das técnicas de autenticação e soluções de *Software Livre* chegou-se a um modelo inicial que consiste na criação de uma “camada” de

[3]MAC Adress - É o endereço MAC - Medium Access Control

autenticação de usuários a ser usada em redes *Wireless*. Esta "camada" é representada por um computador localizado entre a Internet e a rede local do órgão que usa *Wireless*. Neste computador existe um banco de dados contendo informações do usuário como *login* e senha e seus respectivos privilégios de acesso. Desta forma, os usuários dos computadores da rede *Wireless* deste órgão terão que se autenticar com seus dados, para ter acesso aos recursos de rede que lhe são permitidos. Os serviços de rede para os usuários não cadastrados que tentarem acesso indevido serão negados.

O resultado final deste trabalho poderá ser usado por qualquer órgão público que queira ter o controle sobre um conjunto de usuários, sub-redes, departamentos e anexos.

1.2 Estado da Arte

Atualmente existem várias técnicas para promover a segurança em redes de computadores as quais podem ser aplicadas às redes *Wireless*. Nas redes *Wireless*, buscando-se o fornecimento da segurança, são empregadas técnicas como a configuração das bases de difusão de sinais, denominadas de *Access Point*, o cadastro de endereços MAC dos computadores e o uso de criptografia (ROSS, 2003). Não obstante a utilização dessas técnicas, o presente trabalho objetiva demonstrar que o uso de autenticação de usuário baseada no *login* e senha, também é capaz de proporcionar a segurança necessária ao ambiente computacional em que se utiliza rede *Wireless*.

A autenticação de usuários em redes de computadores pode ser implementada de várias formas, sendo a mais comum o uso de *firewall*. No caso das redes *Wireless*, o gerenciamento é feito através de ferramentas comerciais que acompanham o equipamento de rádio. Para *Marcelo* (MARCELO, 2003), em uma rede de computadores, um *firewall* é um conjunto de técnicas de segurança baseadas em *hardware* e/ou *software* com o objetivo de proteger as informações e/ou filtrar o acesso às mesmas. O *firewall* pode servir como elemento de ligação entre redes e fazer o gerenciamento das informações que trafegam entre elas.

Os *firewalls* podem ser implementados através de programas instalados em computadores ou roteadores. Os *firewalls* podem ser construidos através de um conjunto de regras de proteção e de filtragem de pacotes (MARCELO, 2003).

As técnicas de *firewall* construidos via *software* são implementadas de duas formas: *firewall* de pacote e/ou *firewall* de conteúdo.

***firewall* de pacote:** faz o controle de pacotes e diz se eles podem ou não passar de uma rede para outra. O controle de pacotes por exemplo, permite ou não,

acesso à Internet ou acesso ao um servidor FTP[4].

***firewall* de conteúdo:** implementado através de um servidor *proxy*, geralmente um *cache web* com o intuito de acelerar a conexão. Entretanto, o conteúdo do *cache* pode ser submetido a análise liberando ou negando os acessos de acordo com as regras definidas pelo administador.

A partir desse princípio existem técnicas onde é possível configurar um *firewall* com controle de pacotes e conteúdo. Neste *firewall* são configuradas regras de gerenciamento de acesso aos pacotes e ao conteúdo deles. Através da configuração de um *firewall* como o *IPFW2*, *PF* ou *iptables*[5] é possível controlar os pacotes e através da configuração do *proxy* como o *squid*[6] é possível controlar os conteúdos dos pacotes.

Uma das formas de implementar a autenticação dentro de uma rede, consiste em configurar um servidor *proxy* com autenticação. Através do cadastramento dos usuários no servidor *proxy* é possível dar ou não acesso aos recursos de rede.

A figura 1.1 ilustra um exemplo[7] de autenticação usando o *squid* em conjunto com um servidor Web *Apache*[8].

A autenticação seguindo este modelo faz com que seja necessário que o computador do usuário esteja usando o *proxy* na configuração do navegador Web. Dessa forma, o usuário ao tentar acessar a internet, terá que informar o *login* e senha cadastrados na base de dados do *Apache*.

As redes dos Poderes Executivo, Legislativo e Judiciário Tocantinense utilizam como padrão o sistema operacional *FreeBSD* em conjunto com o *Firewall Aker*[9] para implantar a segurança e gerência de rede. Dessa forma, sugere-se que os serviços de rede e aplicativos sejam desenvolvidos tendo como base o sistema operacional *FreeBSD*.

Uma outra ferramenta que implementa soluções de *firewall* é o *pfSense* que é um *firewall* de código aberto baseado na plataforma do *m0n0wall*[10]. O *pfSense* tem como objetivo fornecer um gerenciamento via web para ferramentas de *firewall* baseadas nos sistemas operacionais *OpenBSD*[11] e *FreeBSD*. O *pfSense* encontra-se

[4] FTP - File Transport Protocol - Protocolo de Transferência de Arquivos

[5] Iptables, IPFW2 e PF são *softwares* que implementam técnicas de *firewall*

[6] Squid - *sfotware* que implementa o serviço de proxy - www.squid-cache.org

[7] Para o exemplo ilustrado considera-e que tanto o Servidor Web *Apache* e o *Squid* estejam instalados

[8] Apache - www.apache.org - servidor web

[9] Aker - www.aker.com.br empresa especializada segurança.

[10] O *m0n0wall* é um *firewall* com gerenciamento via web usando *FreeBSD* e *PHP* disponível em: http://m0n0.ch/wall/

[11] Sistema Operacinal OpenBSD - www.openbsd.org/

```
1- Acessar a pasta do codigo fonte do squid e dentro dela ir para o diretorio helpers/basic_auth/NCSA
        # cd /usr/src/squid-2.5.STABLE6/helpers/basic_auth/NCSA

2- Compilar o programa de autenticacao e instala-lo:
        # make
        # make install

3- Editar o arquivo de çãconfigurao do squid (squid.conf):
        3.1- Configurar o programa de autenticacao, apontando para um arquivo de usuarios
        auth_param basic program /usr/local/squid/libexec/ncsa\auth /usr/local/squid/etc/passwd

        3.2- Criar UM acl para autenticacao
        acl password proxy_auth REQUIRED

        3.3- Liberar este acl para navegacao
        http_access allow password

        3.4- Bloquear todos os demais
        http_access deny all

4.) Criar o arquivo de áusurios com um programa que vem com o servidor web apache
        #/usr/local/apache/bin/htpasswd -c /usr/local/squid/etc/passwd mleal
        // O âparmetro -c é para criar o arquivo, para a çãcriao dos proximos
        // áusurios usar o comando sem o parametro -c

5.) Reiniciar o squid:
        # killall -9 squid
        # /usr/local/sbin/squid
```

Figura 1.1: Exemplo de autenticação usando *Squid e Apache*

em desenvolvimento e está na versão 1.0 BETA 3 e fornece um ambiente amigável para a gerência de *firewall* (PFSENSE, 2006).

Com relação aos recursos usados nos equipamentos de rádio para redes *Wireless* pode-se citar:

Ocultação da identificação de serviço : a configuração do ponto de acesso oculta o indentificador, evitando a identificação na varredura por invasores;

Cadastro de endereço MAC : cadastro dos endereços MAC dos computadores que podem se conectar ao ponto de acesso;

Uso de protocolos de criptografia (WEP) : criação de chaves criptográficas para identificação dos computadores válidos.

É possível implementar o controle das conexões dos usuários utilizando ferramentas baseadas em *Software Livre* como *IPFW2*, *PF*, *Iptables*, *pfSense* como também com ferramentas comerciais proprietárias como o *Autenticador de Usuários Aker* ou os diversos aplicativos proprietários fornecidos pelos fabricantes dos equipamentos de rádio como o *Access Point*. Apesar de existirem inúmeras técnicas baseadas em *software* proprietário e em *Software Livre*, existe a necessidade da criação de uma solução própria do Centro Operacional de Redes do Poder Judiciário e Poder Executivo, customizada de acordo com a filosofia e realidades locais.

Não obstante as técnicas mencionadas fornecerem soluções de autenticação que poderiam ser usadas para autenticar os usuários e computadores na rede *Wireless*, elas carecem de uma qualificação técnica por parte do administrador da Rede. Porém, nada impede que possam ser utilizadas em conjunto. A praticidade, flexibilidade e independência tecnológica são aspectos que se deseja obter com o modelo proposto, reduzindo assim os esforços referentes a busca de conhecimento técnico específico de cada aparelho de rádio que o mercado oferece.

1.3 Objetivos

A construção de um mecanismo de Autenticação de Usuários em Redes baseada em *Software Livre* sendo aplicada em redes *Wireless* representa uma solução alternativa e de baixo custo para minimizar os riscos de segurança em ambientes de redes *Wireless* dos órgãos do Executivo e Judiciário Tocantinense.

1.3.1 Objetivo Geral

O objetivo geral deste trabalho é fazer um estudo sobre as tecnologias de comunicação sem fio, observando suas peculiaridades no tocante à segurança computacional, no intuito de fornecer mecanismos alternativos para a implantação da segurança em ambientes computacionais de redes *Wireless*.

Pretende-se então, nesse Projeto, implementar um Protótipo de uma Aplicação de Autenticação Web para Redes *Wireless* para os Órgãos Públicos do Estado do Tocantins utilizando *FOSS*, através do uso de um conjunto de ferrametas baseadas em *Software Livre*.

1.3.2 Objetivos Específicos

Os objetivos específicos estão relacionados no fornecimento de alternativas para implantação de técnicas de segurança através da implementação de uma autenticação de usuários a ser aplicada nas redes *Wireless*. A autenticação será feita através da criação de um banco de dados usando o SGBD[12] *Postgresql* contendo informações do usuário e suas regras de *firewall* e seus recursos de rede disponíveis. Após a normatização e a criação deste banco de dados, será criada uma aplicação web escrita em PHP que terá a função de fazer a autenticação deste usuário quando solicitar uma conexão para Internet.

O ponto principal deste trabalho é estudar meios para desenvolver uma aplicação que consiga redirecionar qualquer conexão oriunda da rede *Wireless* em dire-

[12] SGBD - Sistema de Gerenciamento de Banco de Dados

ção a Internet para uma área de autenticação. Quando o usuário for autenticar-se no sistema, a aplicação verifica no banco de dados quais são as suas regras de *firewall* e cria em tempo de execução, as respectivas regras de *firewall* para este usuário, liberando os recursos cabíveis. As tentantivas de conexões de usuários não autenticados são descartadas e é gerado o *log* da origem da conexão.

Em conjunto com o desenvolvimento da aplicação de autenticação, da modelagem do banco de dados, da criação dos scripts de manipulação dos arquivos de *firewall*, serão configurados os serviços de rede como: Servidor Web, banco de dados e *Firewall*.

1.3.3 Apresentação do Trabalho

Este Projeto está organizado da seguinda forma: No capítulo 2, será apresentada a revisão de literatura juntamente com a apresentação e contextualização do modelo da Aplicação de Autenticação Web para Redes *Wireless* utilizando *FOSS*, dando uma visão geral das suas funcionalidades. O capitulo 3 descreve o material e método usado para o desenvolvimento do Projeto. O capítulo 4 apresenta os resultados e discussões e no capítulo 5 serão expostas as considerações sobre o Projeto.

Capítulo 2

Revisão de Literatura

2.1 Redes de computadores

Com a evolução tecnológica e a redução dos custos dos computadores, tornou cada vez mais interessante a distribuição do poder computacional em módulos processadores localizados em diversos pontos de uma organização. A necessidade de interconexão desses módulos processadores para permitir compartilhamento de recursos de *hardware* e *software* e a troca de informações entre seus usuários, criou um ambiente propício para o desenvolvimento das *redes de computadores* (GOMES SOARES, 1995).

2.1.1 Conceito de Redes de Computadores

É um conjunto de computadores e periféricos que podem estar interconectados através de cabos, linha telefônica, rádio e infra-vermelho. O sistema de comunicação de uma rede será constituído de um arranjo topológico interligando os módulos processadores através de enlaces físicos, que são os meios de transmissão, e de um conjunto de regras, protocolos com a finalidade de organizar a comunicação (GOMES SOARES, 1995).

A partir das redes, os computadores podem trocar informações e compartilhar recursos. A figura 2.1 ilustra um exemplo do arranjo topológico que pode representar uma rede de computadores.

2.2 Redes sem Fio

Os meios de transmissão são os mecanismos de comunicação entre dois ou mais pontos. Eles se diferem com relação a banda passante, potencial para comunica-

Figura 2.1: Sistema computacional

ção, *ponto-a-ponto* ou *multiponto*, limitação devido a atenuação, característica do meio, imunidade a ruído, custo, disponibilidade de componentes e confiabilidade.

Os meios de transmissão podem ser vistos de duas formas: os guiados (cabos) e os não guiados (sinais eletromagnéticos)(CORREIA LUIZ HENRIQUE ANDRADE, 2002).

Este trabalho destina-se a fazer um breve estudo sobre comunicação via rádio, sobre os protocolos que são usados para a comunicação em redes *Wireless*.

Ao tecer comentário sobre comunicação sem fio, oberva-se a analogia com a comunicação natural. Na comunicação natural, as pessoas conversam, ou seja, trocam informações, o ar é o meio de transmissão e o idioma, língua Portuguesa, é o protocolo de comunicação. Nas redes sem fio, tem-se os mesmos elementos: o meio de comunicação é o ar, os equipamentos de rádio são os elementos de comunicação e o protocolo de comunicação para redes *Ethernet* é o padrão IEEE 802.11.

Observação: Nota-se portanto que a evolução da informática também pode ser baseada na observação das coisas simples da natureza.

2.2.1 Redes *Wireless*

As redes via rádio destinadas a fazer o papel de redes locais são denominadas *Redes Wireless*, ou *Wi-Fi(802.11)*[1]. A comunicação entre os elementos computacionais se dá por ondas de rádio, dispensando o uso de cabeamento estruturado.

O transporte de dados de uma comunicação via rádio envolve três elementos distintos: os sinais de rádio, o formato dos dados e a estrutura da rede. Segundo

[1] *WECA Wireless Ethernet Compability Alliance*, grupo industrial que fornece equipamentos com padrão 812.11b, achou mais amigável usar nome Wi-Fi - *Wireless Fidelity*

John (ROSS, 2003), cada um desses elementos é independente dos outros dois. Portanto, em redes *Wireless* tem-se a definição destes elementos.

No tocante ao modelo **OSI**[2], o sinal de rádio opera na camada física, enquanto que o formato dos dados trabalham nas camadas mais elevadas. A estrutura de rede inclui os adaptadores, interfaces e estações bases para envio e recebimento de sinal (ROSS, 2003).

O IEEE[3] produziu um conjunto de padrões e especificações para redes *Wireless* denominados de "IEEE 802.11", que norteia o formato e a estrutura dos sinais. Atualmente existem vários padrões sendo desenvolvidos, sendo que alguns já estão em uso comercial como:

IEEE 802.11a: descreve redes *Wireless* que operam em velocidades mais elevadas com diferentes freqüências de rádio;

IEEE 802.11b: fornece especificações para redes *Ethernet* (sem fio). Este padrão é usado em qualquer tipo de rede sem fio com velocidade de 11 Mbps, desde as redes domésticas às redes públicas. As redes 802.11b operam em uma freqüência em torno dos 2,4 GHz que é reservada para o serviço **ponto-a-ponto** ou de **espalhamento de espectro** não licenciado[4];

IEEE 802.11g: seque o padrão de rede *Wi-Fi* com velocidade similar ao 802.11a que é de 54 Mbps e é compatível com redes 802.11b;

IEEE 802.11i: é uma atualização de segurança do padrão 802.11 e conta com o novo protocolo *AES* (Advanced Encryption System), que oferece maior proteção que o *WPA*.

No modo de operação **ponto-a-ponto** o sinal é transportado de um ponto emissor a outro ponto receptor através de um canal, o contrário disso é modo de radiodifusão onde um envia o sinal para vários. O modo **espalhamento de espectros** é definido como "uma família de métodos para transmissão de um único sinal de rádio, usando um segmento relativamente amplo de espectros de rádio"(ROSS, 2003).

[2]OSI - *Open Systems Interconnectiion* é o modelo de referência criado pela ISO *International Standard Organizartion*

[3]IEEE - Institute of Eletrical and Eletronics Engineers

[4]Não licenciado implica dizer que pode-se enviar e receber sinais usando essa freqüência sem necessidade de uma licença de uso de rádio

2.2.2 Estabelecimento de uma conexão - *Handshaking*

Para ocorrer o início da troca de dados via rádio, é necessário o estabelecimento da comunicação entre a origem e o destino e, segundo *John* (ROSS, 2003), essa comunicação é representada na figura 2.2.

- Origem: “Alô destino! Eu tenho dados para você”;
- Destino: “Ok, origem, vá em frente. Eu estou pronto”;
- Origem: “Aqui estão os dados”;
- Origem: “Dados dados dados”;
- Destino: “Eu recebi algo, mas aparenta estar danificado”;
- Origem: “Então vai novamente”;
- Origem: “Dados dados dados”;
- Origem: “Você recebeu dessa vez?”;
- Destino: “Dessa vez sim! Estou pronto para receber novos dados”;

Figura 2.2: Estabelecimento de uma conexão - *Handshaking*

2.2.3 Controles de redes *Wireless* 802.11

O padrão 802.11b controla como os dados são transportados através da camada física (o link de rádio), definindo uma camada *Média Access Control (MAC)* que manipula a interface entre a camada física e o restante da estrutura da rede. Segundo *John* (ROSS, 2003) essas camadas podem ser definidas da seguinte forma:

Camada Física: O controle nesta camada é feito pelo preâmbulo *PHY*, o emissor adiciona um preâmbulo de 144 bits a cada pacote, incluindo 128 utilizados pelo receptor para efetuarem a sincronização, e um campo *start-of-frame* de 16 bits.

Camada MAC: A camada *MAC* controla o tráfego que ocorre na rede de rádio, evitando as colisões e conflitos de dados, utilizando para isso o CSMA/CA[5]

[5]CSMA/CA - *Carrier Sense Multiple Access witch Collision Avoidance*

A camada *MAC* ainda pode suportar dois tipos de autenticação para saber se um dispositivo está autorizado a associar-se à rede: *autenticação aberta* e *autenticação de chave compartilhada*. A rede suporta todas essas funções da camada *MAC*, trocando uma série de *frames* de controle antes de que as camadas mais elevadas enviem os dados para as camadas superiores. Toda a atividade especificada no padrão 802.11 ocorre nas camadas física e *MAC*. As questões relacionadas a endereçamento, roteamento, integridade, sintaxe e formato dos dados contidos no pacote são feitos pelas camadas superiores, pois é indiferente para as camadas se os dados vierem através de fio de cobre ou de sinal de rádio. Portanto, o padrão 802.11b pode ser usado em qualquer tipo de LAN[6]. Os mesmos rádios podem manipular TPC/IP[7], Novel NetWare[8] bem como serem os outros protocolos construídos para *Windows*, *MAC*, *Linux* e *Unix* com a mesma eficiência (ROSS, 2003).

2.2.4 Dispositivos de Redes

Para os dispositivos de redes, vale ressaltar que foi comentado sobre *link* de rádio e o formato, sendo agora, abordado a infra-estrutura de rede. As redes 802.11x incluem duas categorias de rádios: **estação** e **pontos de acesso** conhecidos como *Access Point*.

Para *John* (ROSS, 2003), os adaptadores *Wireless* para estações se mostram de diversas formas físicas como:

- *PC Cards conectadas ao soquete PCMCIA*[9] *usados geralmente em laptops*;
- Adaptador de rede interno em placas PCI[10] que se encaixam dentro do computador;
- Adaptador USB[11] externos;
- Adaptadores *Wireless* embutidos em *laptops*;
- Adaptadors embutidos em PDAs.

[6] LAN - *Local Area Network*

[7] TCP/IP - *Transmission Control Protcol / Internet Protocol* - Modelo e referência usando na internet

[8] Novel NetWare - http://www.novell.com/pt-br/

[9] PCMCIA - *Personal Computer Memory Card International Association*

[10] PCI - *Peripheral Component Interconnect*

[11] USB - *Universal Serial Bus*

2.2.5 Access Point

Um *Access Point* é uma estação base que pode trabalhar de várias formas. É possível encontrar um *Access Point Stand-alone* que estando conectado a *LAN* já provê o sinal para os computadores. Porém, segundo *John* (ROSS, 2003) existem muitas outras opções de configuração e uso do *Access Point*, são elas:

- Estação base simples: com uma porta *Ethernet* para comunicação com uma *LAN*;
- Estação base com *switch*, *hub* ou roteador, uma ou mais portas *Ethernet* com fio, junto com o ponto de acesso *Wireless*;
- Roteador de banda larga que porporcione uma ponte entre o modem a cabo e o ponto de acesso *Wireless*;
- *Gateway* residenciais que suportam uma quantidade limitada de canais operacionais.

2.2.6 Modos de Operação

Segundo Matos (MATOS, 2005) a "comunicação *Wireless* na aquitetura 802.11x pode ser ponto-a-ponto, onde cada dispositivo da rede comunica-se diretamente com outro, ou em um esquema em que uma determinada área ou célula abrange os dispositivos envolvidos, os quais dependem de um controlador ou ponto central para comunicar-se entre si". Estes dois modos podem ser definidos como:

Redes *ad-roc*: formada por um grupo de estações autocontidas sem conexões com uma *LAN* ou com internet. Uma rede *ad-roc* é caracterizada pela conexão de uma, duas ou mais estações de *Wireless* sem a presença de um *Access Point*. As redes *ad-roc* são baseadas em redes ponto-a-ponto e são chamadas de IBSS[12]. A figura 2.3 ilutra um modelo de rede *ad-roc* (ROSS, 2003);

Redes *infra-estrutura*: é formada por um ou mais *Access Points* geralmente conectados a uma rede com fio. Cada estação *Wireless* troca mensagens e dados com o *Access Point*, que troca com outros nós na rede *Wireless* ou com *LAN* com fio. Uma *infra-estrutura* com uma única estação base é conhecida como BSS[13], já quando existem mais de um *Access Point* é conhecida como ESS[14](ROSS, 2003). A figura 2.4 representa uma rede *infra-estrutura*.

[12] IBSS - *Independent Basic Service Sets*
[13] BSS - *Basic Service Set*
[14] ESS - *Extended Service Set*

Figura 2.3: Exemplo de rede wireless *ad-roc* com tres computadores

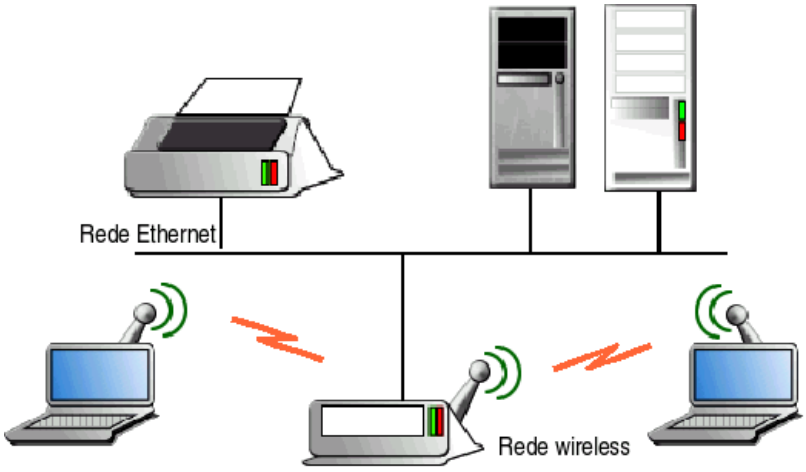

Figura 2.4: Exemplo de rede *Wireless* infra-estrutura

2.2.7 Segurança em Redes *Wireless*

Na especificação 802.11b, existe um esquema de segurança denominado *WEP (Wireless Equivalence Private)* que utiliza a chave criptográfica de até 128 bits, levando em consideração a data de criação deste documento. Um dos problemas atuais é a diferença de tratamento da criptografia de acordo com as marcas dos produtos, pois o tamanho da chave e os modelos de adaptadores nem sempre trocam dados criptografados com o *Access Point* (ROSS, 2003).

O *Wardriving* e *Warkchalking* são técnicas usadas para obter acesso e demarcar a área sujeita à invasão. O *Wardriving* consiste em efetuar um roubo de acesso a

internet a partir de uma rede *Wireless* que não faz uso de criptografia *WEP* ou outros mecanismos de proteção. O *Warkchalking* são símbolos que servem para identificar supostos locais onde o sinal de rede *Wireless* está presente mostrando o nível de segurança (ROSS, 2003).

Os simbolos de *Warkchalking* incialmente foram propostos pelo *Web Designer* ***Mat Jones***(ROSS, 2003). Estes símbolos fazem analogia a uma linguagem criada por andarilhos norte-americanos desempregados, onde marcavam com giz símbolos indicando onde havia comida. No *warkchalking* os símbolos indicam a presença de sinal de rede *Wireless*. A figura 2.5 ilustra o significado de cada figura:

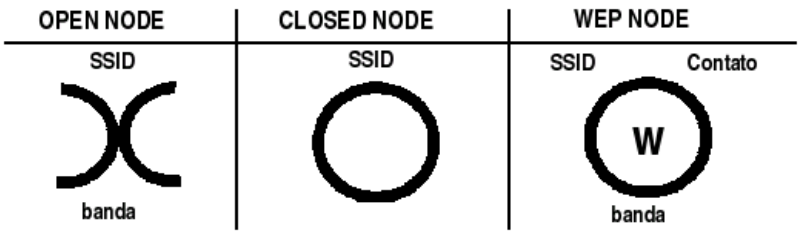

Figura 2.5: Símbolos de *Warkchalking* baseados na proposta de *Mat Jones*

O par de semi-círculos opostos em forma de X significa um *link* aberto. O círculo fechado significa *link* fechado e o círculo com um "W" no meio significa que a conexão está protegida por chave *WEP*, geralmente indicada no canto superior direito do símbolo. Abaixo do símbolo deve estar a velocidade do *link*. O SSID[15], nome do *hotspot*, fica no topo da figura. A segurança em redes *Wireless* pode ser implementada de três formas (MATOS, 2005):

1. Ocultação do *SSID* impedindo que ele seja mostrado por um *broadcast*, pois em redes *wi-fi* todos os dispositivos integrantes devem possuir o mesmo *SSID*. Caso um indivíduo não conheça o *SSID* da rede, ele não conseguirá conectar-se a ela por meios convencionais;

2. Autenticação das Placas *Wireless* por meio de seus endereços *MAC*. Em redes *wireless* no modo infra-estrutura, é possivel cadastrar os endereços *MAC* dos clientes que poderão conectar-se ao *Access Point* (MATOS, 2005);

3. Uso de protocolos de segurança (com a presença ou não de servidor de autenticação). Existem basicamente dois protocolos de segurança para redes *Wireless*: *WEP (Wireless Equivalence Privacy) e WPA (Wi-fi Protected Access)*.

[15]SSID - Service Set Identifier

É importante lembrar que este trabalho não se preocupa em explanar tais configurações, pois o objetivo é fornecer uma alternativa de autenticação baseada em *login* e *senha* e no conjunto de permissões previamente cadastradas em um banco de dados. Portanto, o foco do estudo é trabalhar na implementção de uma forma alternativa de promoção da segurança.

2.3 Protótipo de uma Aplicação de Autenticação Web para Redes *Wireless* para os Órgãos Públicos do Estado do Tocantins utilizando *FOSS*

A autenticação de usuários é uma das formas utilizadas para promover a segurança no acesso aos serviços. A identificação do usuário informa ao sistema de segurança quem você é, e fornece a informação necessária para comprovar a veracidade da sua identificação. A autenticação deve ser baseada em algumas características que apenas o usuário identificado possui ou conhece (TERADA, 2000).

Ela pode ser baseada em:

Algo que você sabe : a utilização de senhas de acesso;

Algo que você tem : uso de cartões magnéticos, *ships* e outros;

Alguma característica que você possui : impressão digital e retina dos olhos.

Tendo como ponto de partida um dos três princípios citados, será utilizada a técnica de autenticação de usuário baseado em usuário e senha, pois entende-se que esta técnica atende ao propósito deste trabalho, e também por tratar-se de um mecanismo sem custos adicionais com relação a adoção de cartões e leitores óticos que necessitam de investimentos financeiros.

A seguir far-se-á um detalhamento do modelo lógico e físico do Protótipo de uma aplicação de autenticação web para Redes *Wireless* utilizando *FOSS*, apresentando o cenário a ser aplicado e as tecnologias de *Software Livre* usadas.

A implementação da autenticação é construída através da configuração de serviços de redes, uso de técnicas de *firewall* e no desenvolvimento de uma aplicação responsável pelo gerenciamento das regras de *firewall* dos usuários. O uso em conjunto de ferramentas e serviços de redes, aliados com uma aplicação web objetiva construir uma “camada” proporcionando maior segurança em ambientes de rede e minimizando os riscos de invasão.

2.3.1 A Arquitetura do Protótipo de uma Aplicação de Autenticação Web para Redes *Wireless* para os Órgãos Públicos do Estado do Tocantins utilizando *FOSS*

O propósito do modelo proposto para a aplicação de autenticação web para Redes *Wireless* utilizando *FOSS* é visto como uma "camada" a mais para promover a segurança. Dessa forma pode ser usado em conjunto com os recursos de segurança inerentes aos equipamentos de redes como: ocultação do identificador do serviço, controle de endereços *MAC* e uso de criptografia.

A figura 2.6 representa a idéia lógica onde o serviço de autenticação pode ser usado.

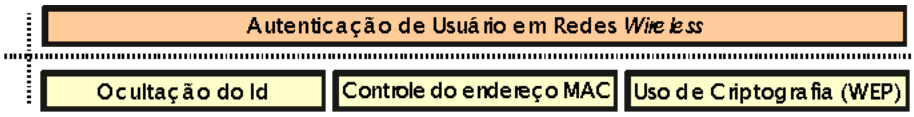

Figura 2.6: Modelo lógico da "camada" de Autenticação de Usuários

Não obstante ao fato da implementação do modelo lógico de autenticação de usuário ter a flexibilidade de ser usado em redes e sub-redes TCP/IP[16], observa-se, a aplicação deste modelo nas redes *Wireless* quando usado entre o ponto de acesso e outras redes.

Na figura 2.7 representa a idéia lógica do uso da proposta da aplicação de autenticação web para Redes *Wireless* utilizando *FOSS*.

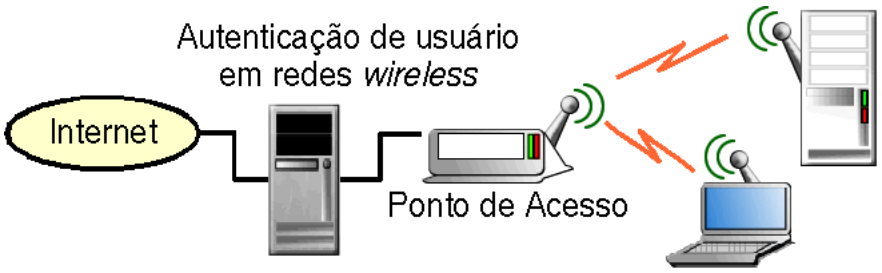

Figura 2.7: Aplicação voltada para redes *Wireless*

Observa-se que o servidor responsável pela autenticação atua entre o *Access Point* e a Internet, dessa forma ele pode agir de forma independente ou em conjunto

[16]TCP/IP - *Transmission Control Protcol / Internet Protocol* - Modelo e referência usando na internet

com as configurações de segurança existentes no ponto de acesso.

2.3.2 Visão dos Modelos Lógicos dos Módulos

A seguir será apresentada a descrição dos modelos lógicos dos módulos presentes no protótipo da aplicação de autenticação web para Redes *Wireless* utilizando *FOSS*.

Módulo de Administração : aplicação web responsável por gerenciar os usuários

- o administrador cadastra, exclui, altera e bloqueia o usuário no sistema;
- no cadastro são informados os dados do usuário, as portas de serviços que serão liberadas;

A figura 2.8 representa o módulo de gerenciamento dos usuários e o seu código fonte está ilustrado nas figuras A.3, A.4, A.5 e A.6 do Apêndice.

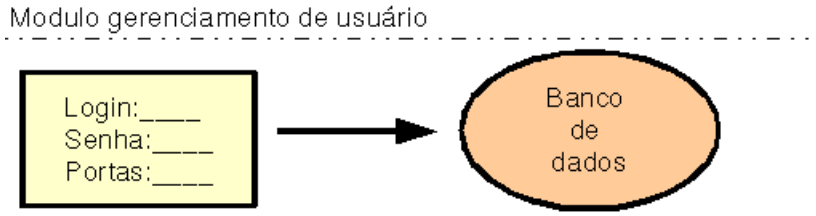

Figura 2.8: Administração dos usuários

Módulo de Autenticação de Usuário : trata-se de uma aplicação web responsável em autenticar o usuario

- o usuário loga-se no sistema;
- a aplicação web obtém o endereço ip do usuário;
- em seguida carrega as informações de acesso no banco de dados;
- gera um arquivo com o nome do usuário contendo as regras de *firewall* liberando as portas cadastradas para endereço ip do usuário logado;
- após isso a aplicação web executa um script *shell* que altera o arquivo de *firewall* e reinicializa o serviço;

- a partir desse ponto o usuário pode acessar os serviços disponibilizados para o perfil dele.

A figura 2.9 representa os passos que são executados no módulo de autenticação de usuário e o seu código fonte é ilustrado nas figuras A.1 e A.2 do Apêndice.

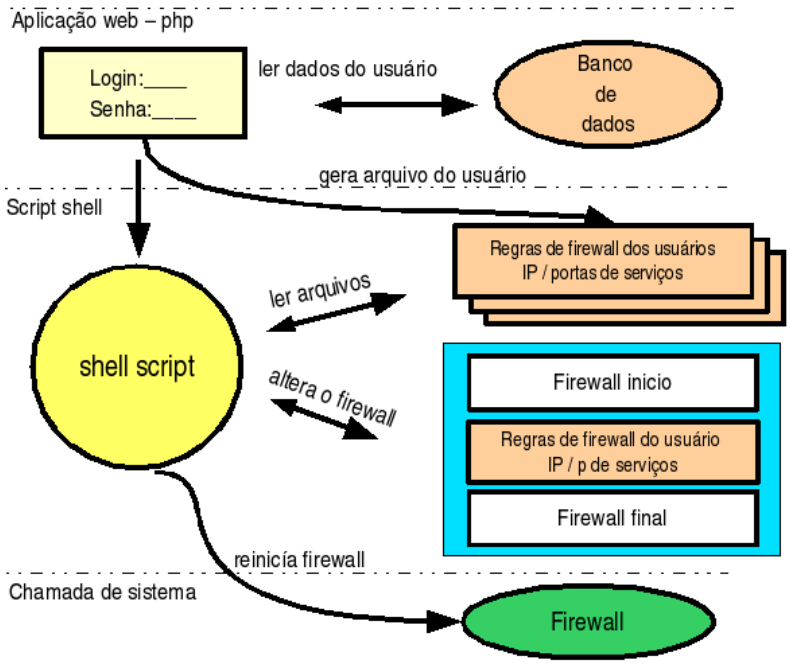

Figura 2.9: *Login* do usuário

Módulo de *Logout* do Sistema : ações executadas ao sair do sistema

- ao clicar no botão sair, o usuário faz o *logout*;
- a aplicação web deleta o arquivo do usuário;
- após isso a aplicação web executa um script *shell* alterando o arquivo de *firewall* e reinicializa o serviço;
- dito isso o *firewall* é atualizado sem as regras do usuário que fez *logout*;

A figura 2.10 representa os passos que são executados no módulo de *logout* do sistema.

Módulo de *Firewall* : arquivo de *firewall*

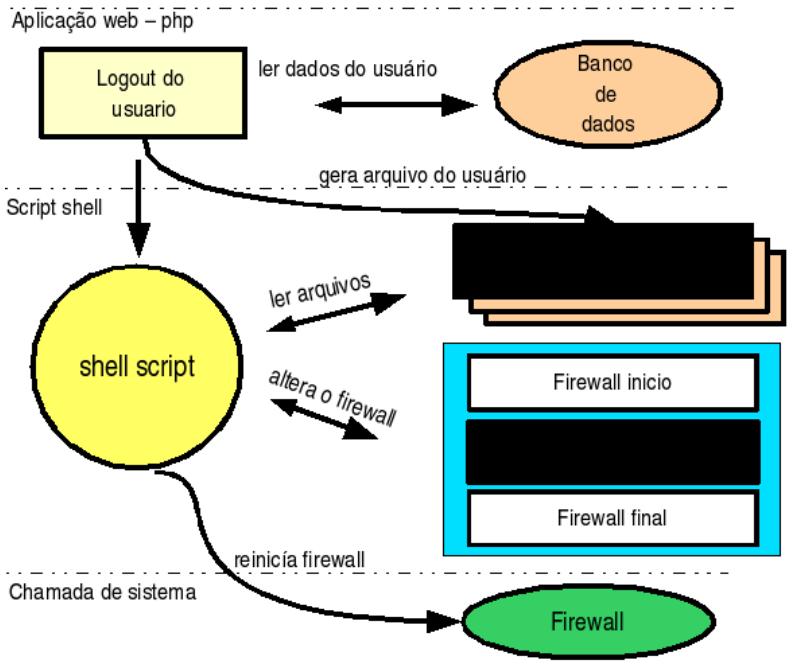

Figura 2.10: *Logout* do usuário

- possui um arquivo básico com o início das regras de *firewall* que é implementado inicialmente fechado, dando acesso apenas ao servidor de autenticação;
- a cada login de usuário, o *firewall* é alterado e reiniciado, dando acesso ao ip do usuário logado de acordo com o perfil dele;
- quando o usuario sair do sistema, o *firewall* é alterado e reiniciado sem as regras do usuário;
- a última regra fecha o *firewall* para os demais serviços que não estão disponíveis.

Módulo Script *Shell* : script responsável por fazer o gerenciamento do *firewall*

1. o script carrega o arquivo com opções iniciais do *firewall*;
2. faz uma leitura no diretório onde estão os arquivos dos usuários gerados durante o login;
3. concatena o conteúdo das opções iniciais com as regras dos arquivos dos usuários;

4. concatena o conteúdo anterior com o arquivo da regra de finalização *firewall*;
5. ao final, o script atualiza o novo arquivo de *firewall* com as regras de acessos dos usuários;
6. quando o usuário faz o *logout* no sistema, a aplicação web apaga o arquivo do usuário e executa o script *Shell* e faz novamente os itens 1, 2, 3, 4 e 5. Neste caso, ocorre uma atualização e reinicialização do *firewall* sem as regras do perfil do usuário, fechando, portanto, o acesso para o computador do usuário.

2.3.3 O Desenvolvimento do Protótipo de uma Aplicação de Autenticação Web para Redes *Wireless* para os Órgãos Públicos do Estado do Tocantins utilizando *FOSS*

Para conceber o modelo da aplicação de autenticação web para Redes *Wireless* utilizando *FOSS* estão sendo feitos estudos das tecnologias de *Free Software*, *Software de Código Aberto* e *Software Livre*.

O *Free Software* é uma forma de desenvolver e distribuir programas através do qual o usuário não precisa pagar os altos valores, normalmente cobrados pelos fornecedores de *software*, pela licença de uso e direitos autorais do produto. O *Free Software* é legal em todos os sentidos, porque não infringe qualquer lei de proteção autoral, por exemplo, não infringe a Lei 9.610[17], de 19 de fevereiro de 1998, que regula os direitos autorais, entendendo-se sob esta denominação os direitos do autor e os que lhes são conexos (UNESP, 2000).

Programa de Código Aberto ou *Open Source* é aquele em que se tem acesso ao código fonte e pode efetuar modificações no programa original. Os programas comerciais não costumam ter o código aberto. Os programas distribuídos através de *Free Software* podem ou não ter código aberto (UNESP, 2000).

No tocante ao *Software Livre*, "a sua principal filosofia é a questão de liberdade de expressão, dado que os usuários têm liberdade de executá-los, copiá-los, distribuí-los, estudá-los, modificá-los e aperfeiçoá-los" (UCHôA, 2003). Um *software* é considerado como livre quando atende aos quatro tipos de liberdade para os usuários do *software* definidas pela *FSF (Free Software Foundation)*:

- a liberdade de executar o programa, para qualquer propósito;
- a liberdade de estudar como o programa funciona, e adaptá-lo para as suas necessidades;

[17] Lei que regula os direitos autorais, entendendo-se sob esta denominação os direitos de autor e os que lhes são conexos. (BRASIL, 2001)

- a liberdade de redistribuir cópias de modo que você possa ajudar ao seu próximo;
- a liberdade de aperfeiçoar o programa, e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie, entre outos.

Uchôa (UCHôA, 2003) diz ainda que "a liberdade ainda inclui que não é necessário pedir ou pagar pela permissão de tomar todas essas atitudes".

Software Proprietários é o tipo de *software* onde a licença é a forma que os fornecedores encontraram para evitar que seus produtos sejam utilizados por um número indiscriminado de pessoas, sem o correspondente pagamento. Dessa forma, os programas, ainda que executados através de uma rede de computadores, somente poderão ser acessados por um número limitado de usuários, conforme a quantidade de licenças adquiridas.

A *GPL, General Public License*, é uma forma de distribuição em que o código fonte está disponível nos termos da *Free Software Foundation*. A licença *GPL* fornece liberdade para que se possa usar e alterar o programa original. A *GPL* não permite, entretanto, a apropriação indébita dos direitos autorais, sendo vedada sua comercialização. A maioria dos produtos gratuitos seguem esta filosofia.

O sistema operacional *GNU/Linux*, por ser um *software* não comercial, é o grande impulsionador do *Sofware Livre*. É nessa plataforma de *software* que é desenvolvida a grande maioria dos produtos de uso gratuito. O fato de ser um sistema operacional não proprietário tem um apelo especialmente interessante porque libera os aplicativos do domínio de um único fornecedor (UNESP, 2000).

Os sistemas e aplicativos utilizando o *FOSS* usados no desenvolvimento da Aplicação de Autenticação Web para Redes *Wireless* são:

- Sistema Operacional Unix FreeBSD;
- Firewall IPFW2;
- Linguagem de Programação PHP e Script *Shell*;
- Servidor Web Apache;
- Sistema de Gerenciamento de Banco de Dados Postgresql.

Sistema Operacional *FreeBSD* : O *FreeBSD*[18] é um sistema operacional *UNIX-like* para plataformas i386 e *Alpha/AXP*[19], baseado no *"4.4BSD-Lite"* da

[18] FreeBSD - http://www.freebsd.org/index.html

[19] *Alpha/AXP* é uma arquitetura de computadores RISC de 64 bit desenolvida pela *Digital Equipment Corporation* adquirida mais tarde pela *Compaq* que hoje é fundida com a *Hewlett-Packard*

Universidade da Califórnia em *Berkeley*, com alguns aprimoramentos adotados do *"4.4BSD-Lite2"*. O *FreeBSD* também é baseado, indiretamente, na conversão de *William Jolitz* conhecida como "386BSD"para a plataforma i386 do *"Net/2"* da Universidade da Califórnia, em *Berkeley* (FREEBSD, 2005).

O código do *FreeBSD* é distribuído sob a Licença Pública Geral (GPL)[20] ou sob a Licença Pública Geral de Bibliotecas GNU (LGPL)[21] que inclui algumas obrigações anexadas a ele. Contudo, tais restrições visam garantir o acesso livre a esse código. Devido à complexidade adicional que envolve a utilização comercial de *software* licenciado sob GPL, utiliza-se tal *software* sob a licença de direito autoral *FreeBSD*[22] (FREEBSD, 2005).

Atualmente, o *FreeBSD* encontra-se na versão 5.4 e é amplamente usado por empresas, órgãos governamentais, pesquisadores, profissionais de informática, estudantes e usuários domésticos. Trata-se de um sistema operacional *Unix* voltado para servidores e que também pode ser usado como *desktop*. O *FreeBSD* é considerado estável, confiável e possui uma excelente performance, com uma coleção de nove mil aplicativos, tendo como objetivo oferecer *software* de baixo custo, alta performance e segurança que possa ser utilizado para qualquer finalidade sem obrigações anexadas a esse código (FREEBSD, 2005).

O *Shell* : Um *Shell* é um interpretador de comandos, uma interface entre o usuário e o núcleo do sistema operacional, em outras palavras, uma camada entre o usuário e o *kernel*. Um programa em *Shell*, denominado de script, consite em uma ferramenta poderosa que utiliza os recursos dessa camada para promover ações no sistema seguindo a ordem descrita no script. Existem scripts para as mais variadas funções, desde fazer uma simples cópia de arquivos de um diretório para outro, até sistemas de *backup* remotos complexos de forma automatizada com geração de *logs* (NEVES, 2005).

O *Sh Shell* foi desenvolvido em *Berkeley University* e é um *shell* dos sistemas *Berkeley* BSD e XENIX. A sua apresentação para o sistema é *sh* (NEVES, 2005).

A linguagem de programação PHP : O PHP ou *Hipertext Processor* é uma linguagem de pré-processamento de hipertexto, uma poderosa linguagem de

[20] GPL - http://www.freebsd.org/copyright/COPYING
[21] LGPL - http://http://www.freebsd.org/copyright/COPYING.LIB
[22] http://www.freebsd.org/copyright/freebsd-license.html

programação web *open source*. O PHP é uma linguagem voltada para desenvolvimento web, apesar de existir o *PHP-GTK* para o desenvolvimento *Desktop*, o PHP destacou-se mundialmente por tratar-se de uma linguagem que possui a capacidade de misturar ao código *HTML*[23] e possibilitar o desenvolvimento de sites dinâmicos (WALACE, 2004).

O PHP possui a característica de ser usado em sistemas *Unix, GNU/Linux* bem como em ambientes *Windows* e tem seus binários e código fonte distribuídos sob a licença PHP[24]. O PHP em sistemas baseados em *Unix* converge em um poderoso recurso de desenvolvimento de sistemas, pois, existem recursos no PHP que vão desde uma simples chamada de sistemas até a possibilidade do gerenciamento do sistema operacional via web.

No processo de evolução do PHP, nota-se que na versão 5 foi feita uma mudança nos conceitos da linguagem no tocante a metodologia de desenvolvimento Orientado a Objetos. Desta forma é possível implementar sistemas usando os recursos de orientação a objetos como: classe, objeto, atributo e método (WALACE, 2004).

Neste aspecto o PHP fornece uma flexibilidade no desenvolvimento das aplicações, pois, deixa a cargo do desenvolvedor escolher a metodologia de desenvolvimento.

O SGBD *PostgreSQL* : O *PostgreSQL* é um sistema de gerenciamento e banco de dados distribuído sob a licença *BSD*[25], "o que torna o seu código fonte disponível e o seu uso livre para aplicações comerciais ou não" (POSGRESQL, 2005). O *PosgreSQL* implementa os padrão SQL[26] ANSI, 92, 96, 99 (NETO, 2003). Para Neto (NETO, 2003), o PostgreSQL pode ser classificado como um SGBDR (Sistema de Gerenciamento de Banco de Dados Relacional) de alta performance.

A construção do *PostgreSQL* iniciou-se em 1986 na Universidade de *Berkeley* nos Estados Unidos, e vem evoluindo aos longo dos anos. Hoje é um SGBDR que pode ser usado em sistemas *Unix*, *GNU/Linux* e *Windows*.

Dentre algumas das características do *PostgreSQL*, pode-se dizer que é um

[23]HTML - *Hyper Text Markup Language*, trata-se de uma linguagem de marcação utilizada para produzir páginas na Internet.

[24]A licença PHP é uma licença que possui o aviso Copyright e é mantida pelo PHP Group e atualmente está na versão 3.01 disponível em: http://www.php.net/license/3_01.txt

[25]Licença *BSD (Berkeley Software Distribution)* - Distribuição de Software de Berkeley. http://www.gnu.org/philosophy/bsd.pt.html

[26]SQL - Structured Query Langage - Linguagem estruturada de consulta a dados em um sistema de banco de dados

excelente SGBD que permite a utilização de SQL, *Triggers* e todos os demais recursos pertinentes aos mais comuns sistemas de gerenciamento de banco de dados comerciais como: *Oracle, InterBase, SQL Server, DB/2* e outros (NETO, 2003).

O servidor web *Apache* : O servidor web *HTTP - HyperText Transfer Protocol* Protocolo de Transferência de Hipertexto é um protocolo da camada de Aplicação do modelo OSI utilizado para transferência de dados na *World Wide Web.* Apache é um projeto baseado em *Software Livre* que tem como objetivo fornecer ao usuário um serviço seguro dentro dos padrões atuais do protoclo HTTP.

O servidor web Apache[27] obteve um sucesso e aceitação no meio comercial e acadêmico. O *Httpd* está na versão 2.x e pode ser usado em sistemas *Unix e GNU/Linux* e *Windows* (BENTES AMAURY, 2003).

O servidor web Apache possui licença própria e ainda possui incompatibilidades com a licença *GPL*. A *Apache Licence*[28] (Licença Apache) é uma licença para *Software Livre* (open source) de autoria da *Apache Software Foundation (ASF)* e está na versão 2.0 e é datada de janeiro de 2004. Todo software produzido pela *ASF* ou qualquer um dos seus projetos e subprojetos é licenciado de acordo com os termos da Licença Apache. Alguns projetos não pertencentes à ASF também utilizam esta licença (APACHE, 2006).

2.3.4 O Desenvolvimento da Aplicação Web

Um dos módulos da aplicação é desenvolvido em PHP. Como visto na figura 2.9, a aplicação via web possui a função de fazer o cadastro do usuário, *login* e *logout*. Ao logar no sistema é feita uma consulta na base de dados e verificadas quais as portas de serviços estão disponíveis no perfil do usuário. Dito isto, é criado um arquivo com o nome do usuário logado. Este arquivo contém o endereço ip e as portas de serviços seguindo o formato de regras de *firewall*. O arquivo é gerado no diretório de regras dos usuários e em seguida a aplicação web executa o script *shell*.

[27] Apache - www.apache.org

[28] Apache Licence - Licença Apache disponível em http://www.apache.org/licenses/LICENSE-2.0.html

2.3.5 O Desenvolvimento do Script *Shell*

Observando a figura 2.9, verifica-se que o projeto tem a característica de utilizar um módulo via web e outro via script *shell*. Esse arranjo visa diminuir a carga de processamento no tocante às mudanças ocorridas no *firewall* e sua atualização do sistema operacional. Considera-se que há ganho de desempenho usando codificação *shell* quando forem feitas operações no *firewall* e arquivos dos usuários, pois as chamadas de sistema são feitas diretamente usando a API[29] do Sistema Operacional.

Tendo em vista esta teoria, o script *shell* é responsável por fazer a manipulação dos arquivos como: leitura, gravação, remoção e alteração nos arquivos usuários e *firewall*. A cada alteração no arquivo de *firewall* ocorre uma reinicialização do serviço. A perspectiva é que essa estrutura funcione com um desempenho melhor caso deixasse todos os procedimentos por responsabilidade da aplicação em web pois, ao invés da aplicação web fazer várias chamadas de sistema, ela fará apenas uma chamada de sistema acionando o script *shell*.

A figura 2.11 ilustra a estrutura do script *shell*.

***firewall* - cabeçalho :** arquivo contendo as opções gerais no inicio de um *firewall* como limpeza das tabelas e regras, abertura somente para a máquina de autenticação;

***firewall* - rodapé:** arquivo contendo as opções finais do *firewall*, geralmente contendo regras de fechamento;

***firewall* dos usuários :** todos os arquivos dos usuários que estão logados pois, ao logar é gerado o arquivo do usuário contendo suas regras;

***firewall* geral :** script responsável por pegar o arquivo de cabeçalho colocar seu conteúdo no rc.*firewall*, fazer uma varredura no diretório dos arquivos dos usuários e em seguinda fazer a concatenação dos arquivos dos usuários no rc.*firewall*, em seguida concatenar com o arquivo de rodapé fazedo o fechamento do script. Por fim é reiniciado o *firewall* no sistema operacional.

Figura 2.11: Estrutura do Script *Shell*

[29] API - Applications Program Interface

Capítulo 3

Material e Método

3.1 Local e Período

O trabalho está sendo desenvolvido no Centro Operacional de Redes do Tribunal de Justiça com o apoio do Centro Operacional de Redes do Estado do Tocantins na capital Palmas, durante o período de setembro a julho de 2006. Sendo que o primeiro protótipo deste projeto será apresentado como trabalho de conclusão de curso de Pós-Graduação *"Lato Sensu"* em *"Administração em Redes Linux"* em dezembro de 2005 na Universidade Federal de Lavras no Estado de Minas Gerais.

3.2 Material

3.2.1 Hardware

Estão sendo utilizados computadores do Centro Operacional de Redes do Poder Judiciário, bem como equipamentos de rádio fornecidos pelo Centro Operacionl de Redes do Poder Executivo, são eles:

Estação de Trabalho : 1 - microcomputador Pentium IV com processador de 1.4 Mhz, 256Mb de memória RAM e HDD de 40Gb;

Servidor de autenticação: 1 - microcomputador AMD XP com processador de 1.8 Mhz, 256Mb de memória RAM e HDD de 40Gb;

Access Point : 1 - ponto de acesso configurado na modalidade de rede *infra-estrutura* - Air Point;

Notebook : 1 - notebook munido de placa *wi-fi*;

desktop : 1 placa de rede *wi-fi*.

3.2.2 Software

A estação de trabalho está configurada com dois sistemas operacionais: Windows XP e o GNU/Linux Debian Sarge. Pretende-se testar a conexão nos dois sistemas operacionais para que se possa promover um serviço utilizável em ambas as plataformas.

A configuração do servidor será em cima do Sistema Opercional *FreeBSD* versão 5.4, foi instalado e configurado o servidor web *Apache* e o kernel foi recompilado para funcionar o *firewall IPFW2*. As linguagens de programação usadas para desenvolver as aplicações são escritas em script *PHP* e em *shell sh*[1] e o SGBD[2] foi o *Postgresql*.

Software utilizados no servidor de autenticação:

- Sistema Operacional FreeBSD;
- Servidor Web Apache versão 2.x;
- PHP Versão 5.x;
- Shell SH;
- Postgresql 8;
- IPFW2;
- Kile 1.7.1 para editoração em LaTeX

3.3 Método

A metodologia para o desenvolvimento do protótipo de uma aplicação de autenticação web para Redes *Wireless* utilizando *FOSS* consiste na instalação e configuração de serviços no sistema operacional bem como no desenvolvimento de aplicações via web.

[1] shell - Shell padrão em sistemas *BSD*
[2] SGBD - Sistema de Gerenciamento de Banco de Dados

3.3.1 Instalação do *Software*

A seguir serão elencados os pontos para a criação da infra-estrutura básica de *software* que foi usada para a implantação inicial do modelo de autenticação proposto. Nesta, são relacionadas o sistema operacional e serviços usados.

- Instalação e configuração do *FreeBSD*;
- Instalação e configuração do servidor web *Apache*;
- Instalação e configuração do PHP;
- Instalação, configuração do Postgresql e criação do banco de dados;
- Instalação do PGAdminIII
- Criação das tabelas e visualizações usando o PGAdminIII;
- Edição das páginas em PHP no phpedit;

3.3.2 Implementação da Autenticação

A implementação do protótipo da aplicação de autenticação web para Redes *Wireless* utilizando *FOSS* é feita através do desenvolvimento de uma aplicação web escrita em PHP e do uso de scrips em *shell* que fazem manipulação dos arquivos e interação com o sistema operacional. Para o desenvolvimento foram feitos os seguintes passos:

- desenvolvimento do protótipo executável da aplicação via web escrita em PHP que será responsável pela gerência dos usuários;
- o desenvolvimento web seguiu a metodologia de Programação Orientada a Objetos;
- alimentação do banco de dados com cadastro de serviços, regras, usuários e administradores;
- desenvolvimento do *shell* script responsável pela automatização das tarefas relacionadas ao *firewall* e interação com o sistema operacional;
- o desenvolvimento do script *shell* seguiu a metodologia de Programação Estruturada e sequencial;
- interação da aplicação web com o script *shell* através de chamada de sistemas;

- testes de usabilidade do protótipo executável observando sua funcionalidade e resultado;

3.4 Atividades

A seguir é apresentado um resumo das atividades realizadas durante o período de desenvolvimento do Trabalho de Conclusão de Curso.

Agosto: Estabelecimento do contato com o orientador

- No mês de agosto feito contato com o orientador e determinação da forma de trabalho;
- Envio do pré-projeto;
- Início do processo de escrita do trabalho.

Setembro: Determinação da forma de trabalho e pesquisa bibliográfica

- Definição do trabalho estabelecendo as tecnologias a serem utilizadas. Aquisição de conhecimento a partir de coleta de dados na Internet e estudo de trabalhos acerca da área de *wireless*;
- Definição da ferramenta para edição do relatório, na qual foi eleito o *kile* para editar documentos LaTeX;
- Definição das tecnologias de código livre tais como PHP, Apache e FreeBSD;
- Definição da Arquitetura para desenvolvimento da autenticação de usuários

Outubro: Pesquisa bibliográfica e coleta de material para estudo

- Estudo das tecnologias envolvidas;
- Criação do prmeiro protótipo em script *shell* para alteração do arquivo de *firewall* fazendo a inserção e a remoção das supostas regras do usuário;
- automatização da atualização do arquivo de *firewall* a cada alteração das regras dos usuários no *firewall* em tempo de execução promovendo o *stop* | *start* de forma automática;
- Confecção do trabalho escrito

Novembro: Estudo e programação em PHP e confecção do trabalho escrito

- Desenvolvimento do primeiro protótipo em PHP utilizando as funções: para cadastrar o usuário e estabelecendo seu perfil de acesso no banco de dados, geração dos arquivos com regras dos usuários e execução do *script shell* de automação das tarefa do *firewall*;
- execução de testes dos módulos separadamente;
- relização dos testes no módulo de gerenciamento via web e no módulo de script;
- Confecção do trabalho escrito.

Dezembro: Apresentação do Trabalho

- “Defesa” do Trabalho na UFLA - Lavras - MG.

Capítulo 4

Resultados e Discussão

A seguir far-se-á um detalhamento dos resultados obtidos até o presente momento, ilustrando as telas do protótipo da aplicação de autenticação web para Redes *Wireless* utilizando *FOSS*. A implementação da autenticação é construída através da configuração de um *firewall* e do desenvolvimento da aplicação responsável pelo gerenciamento dos usuários.

Os resultados alcançados até o presente momento contemplam apenas a utilização básica do serviço, onde é possível efetuar as seguintes operações:

- cadastrar e gerenciar os usuários e respectivas portas de serviços disponíveis;
- logar no sistema e gerar as regras de *firewall* por usuário;
- executar o script de construção do novo arquivo de regras de *firewall*, alterar e reinicializar o *firewall* em tempo de execução;
- exclusão das regras de *firewall* dos usuários ao sair do sistema.

4.1 A aplicação web

A figura 4.1 ilustra o momento em que o administrador do sistema faz o *login*. Este possui recursos de cadastro, consulta alteração e exclusão de usuários. Para facilitar o gerenciamento, foi criado uma campo no banco de dados para informar se o usuário está ativo ou inativo.

Já a figura 4.2 ilustra o cadastro de um usuário, onde é informado o *login*, a senha e as portas de acesso para o seu perfil. Na figura A.3 do Apêndice é ilustrado o código fonte de cadastro de usuário.

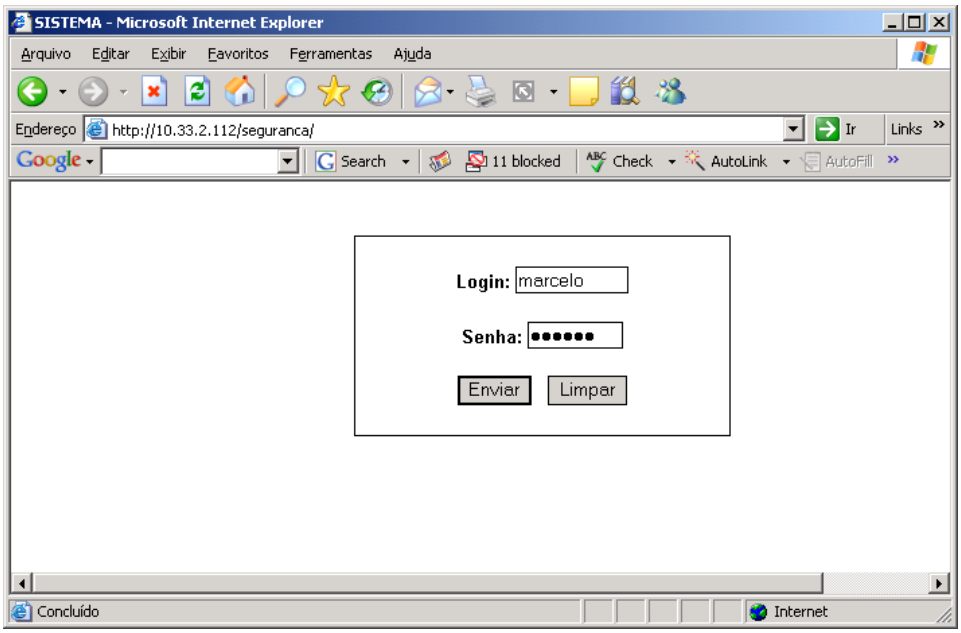

Figura 4.1: Tela de *Login*

Na figura 4.3 mostra o momento em que o usuário loga no sitema, apenas ilustrando os serviços disponíves. Pois, neste momento, no servidor, o script foi acionado e liberou as referidas portas para o endereço ip do computador do usuário. Desta forma, o script *shell*, faz a reorganização do conteúdo do *firewall* e o reinicializa dando acesso ao usuário logado. A figura A.1 do Apêndice ilustra o código fonte em PHP responsável em fazer o *login* do usuário.

A figura 4.4 ilustra o exemplo de um arquivo de regras gerado no *login* do usuário.

O arquivo do usuário logado contém as regras de acesso aos serviços para o endereço IP do computador ao qual o usuário está usando naquele momento. Após a criação do arquivo é executado o *script* que manipula as regras de *firewall*.

Um ponto que chama a atenção está relacionado à sintaxe do IPFW2, pois um dos parâmetros utilizados é um número no formato incremental e crescente que serve para identificar cada regra. Contudo, o modelo lógico de construção das regras criado para esse projeto não obriga a indicação desse número para cada regra, pois, a sequência numérica é feita automaticamente pelo IPFW2. A partir de experimentos observou-se que o IPFW2 cria e gerencia a sequência numérica das regras, facilitando a criação dinâmica das regras de firewall. Contudo, em

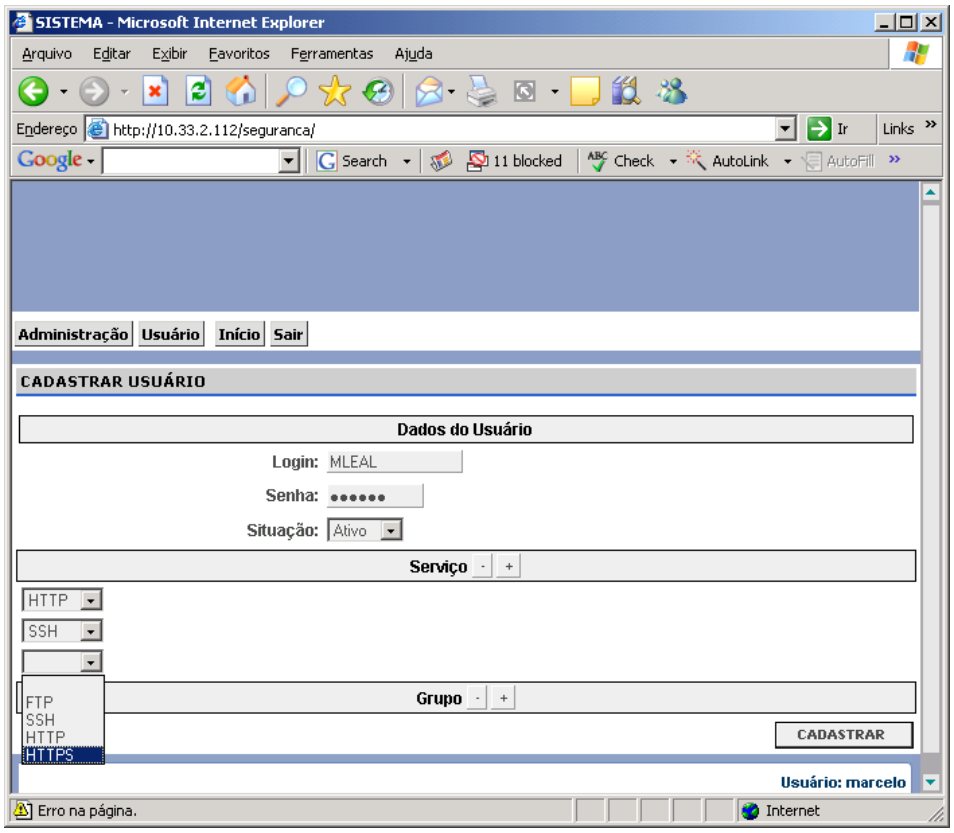

Figura 4.2: Tela de Cadastro de Usuários

versões futuras será desenvolvido um mecanismo para gerenciar os números que identificam as regras dos usuários.

O IPFW2 consegue fazer o gerenciamento das regras, que envolve inclusão e remoção, de forma dinâmica e em tempo de execução sem a necessidade de reinicialização do sistema operacional. Este fato converge num fator crítico de sucesso para a sua adoção, pois este modelo de controle é baseado na alteração constante das regras.

4.2 O Script *Shell*

O *Scrip Shell* foi escrito inicialmente em linguagem *Bash* testado em um sistema operacional *GNU/Linux Debian* e em seguida adaptado ao servidor de autenticação

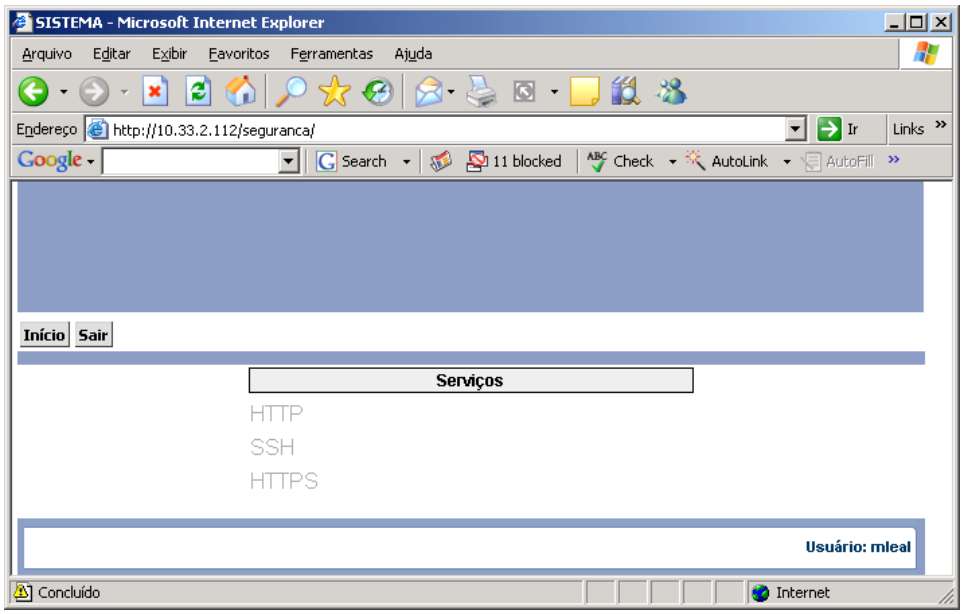

Figura 4.3: Tela de de Acesso de um Usuário

```
ipfw add permit tcp from 10.33.2.81 to any 22 via vr0
ipfw add permit tcp from 10.33.2.81 to any 443 via vr0
```

Figura 4.4: Arquivo gerado durante o *login* do usuário

com sistema operacional *Unix FreeBSD*. O código possui rotinas simples e funcionais como ilustrado na figura 4.5. A figura A.8 do Apêndice traz a ilustração do arquivo gerado pelo script *shell*.

- Na linha 1 é criado o arquivo *rc.firewall* vazio;
- Nas linhas de 2 e 3 sao dadas as permissões de acesso total ao aquivo para qualquer usuário;
- A linha 4 faz a concatenação do cabeçalho inicial *fw-cabecalho* com o arquivo *rc.firewall*. O arquivo *fw-cabecalho* possui regras de limpeza de tabelas do IPFW2 e acesso somente ao servidor de autenticação;
- Em seguida na linha 5 é executado o comando para entrar no diretório onde estão os arquivos dos usuários logados;

```
#!/bin/sh
1 - touch /usr/local/www/seguranca/cw/rc.firewall
2 - chown root.root /usr/local/www/seguranca/cw/rc.firewall
3 - chmod 777 /usr/local/www/seguranca/cw/rc.firewall

4 - cat /usr/local/www/seguranca/cw/fw-geral/fw-cabecalho > /usr/local/www/seguranca/cw/rc.firewall

5 - cd /usr/local/www/seguranca/cw/fw-user
6 - for i in *; do
7 -     cat $i >> /usr/local/www/seguranca/cw/rc.firewall;
8 - done

9 - cat /usr/local/www/seguranca/cw/fw-geral/fw-rodape >> /usr/local/www/seguranca/cw/rc.firewall

10 - chown root.root /usr/local/www/seguranca/cw/rc.firewall
11 - chmod 777 /usr/local/www/seguranca/cw/rc.firewall

12 - /usr/local/www/seguranca/cw/rc.firewall
13 - echo "Firewall␣atualizado!!!!!!"
14 - echo ".....␣ok"
15 - sleep 1
```

Figura 4.5: *Script* de criação e manipulação do arquivo de *firewall*

- Da linha 6 até 8 é feita uma varredura no diretório concatenando os arquivos com as regras dos usuários logados ao arquivo rc.firewall;
- A linha 9 faz a concatenação do rodapé *fw-rodape* com o arquivo *rc.firewall*. O arquivo *fw-rodape* possui a regra de fechamento do *firewall*;
- As linhas 10 e 11 novamente dão permissão de leitura, gravação e execução no arquivo *rc.firewall*;
- A linha 12 possui o comando de execução do *firewall* atualizado.

A função deste *script* é construir um arquivo de *firewall* de acordo com os arquivos existentes no diretório dos usuários. A cada *login* ou *logout* de um usuário ocorre a alteração no *firewall*, automatizando a construção, alteração e remoção das regras de *firewall* em tempo de execução. Apesar de ser um mecanismo simples, é notável a eficácia do seu funcionamento baseado na automação de tarefas.

4.3 Considerações

Acerca do primeiro protótipo executável é necessário realçar as conquistas e os problemas que deverão ser resolvidos nos protótipos futuros. O resultado alcançado até o presente momento, serve de base para um produto de *software* que possui abertura para agregar novas funcionalidades e fornecer uma solução refinada no tocante a prover não apenas um mecanismo de segurança, mas uma solução que consiga autenticar um usuário, informar quais os serviços disponíves, qual o tamanho da banda e proporcione registros para auditoria.

4.3.1 O Problema Encontrado no Modelo Atual

O atual modelo é composto de uma aplicação via web onde o usuário faz o *login* do usuário, em seguida o seu arquivo de regra é criado, após isso o *firewall* é alterado em tempo de execução. Este processo de alteração das regras em tempo de execução consiste no fator crítico de sucesso, pois, no modelo atual, as regras dos usuários não possuem identificação fixa, ficando por responsabilidade do próprio IPFW2 criar o identificador para as regras.

Diante deste fato, não obstante o funcionamento do modelo atual atender em parte à finalidade do projeto, existem casos específicos em que é possível que uma conexão criada por um usuário possa ser prejudicada em virtude de outro usuário efetuar o *login*, pois ocorre a reinicialização do *firewall*, risco este assumido e reconhecido durante o processo de prototipação a ser corrigido em versões futuras. Nos casos onde a conexão é um acesso a sítio web, o prejuízo não é perceptível. Entretanto, nos casos onde a conexão é um *download*, ele pode ser "quebrado". Contudo, caso o cliente utilize ferramentas que fazem *download* cumulativo e com re-conexão posterior, esse problema deixa de existir, pois caso a conexão seja interrompida, o usuário pode acionar a opção para atualizar e continuar com o *download*.

Uma solução ao problema citado está na construção de um controle de identificação de regras. A partir da identificação é possível desabilitar a regra sem a necessidade de reiniciar todo o *firewall*, limpando todas as regras. Para implantar esse mecanismo de controle, existe a necessidade de um estudo aprofundado nos recursos do IPFW2 para que se possa viabilizar o controle de identificação das regras em projetos futuros.

4.3.2 Os Objetivos Alcançados

Em todo projeto que se encontra em desenvolvimento, tolera-se a existência de problemas, de soluções aos problemas e também de objetivos alcançados. A primeira versão do protótipo da aplicação de autenticação web para Redes *Wireless* utilizando *FOSS* conseguiu atingir seu objetivo básico inicial, pois foi realizado um estudo nas ferramentas de *Software Livre*, *Software Aberto*, determinado o sistema operacional, o modelo lógico, o modelo físico e o *software* para a construção da aplicação. Em seguida foi feita a configuração dos serviços, o desenvolvimento dos programas para web e a automação de tarefas. Foi realizado o primeiro teste tendo alcançado o sucesso na materialização da idéia. A teoria se tornou realidade quando o usuário de teste efetuou o *login*, suas regras foram criadas em tempo de execução e ele pôde acessar os recursos de rede disponíveis no laboratório de teste no Centro Operacional de Redes do Poder Judiciário.

A partir da “materialização” da idéia base é possível aprimorar o projeto para que nos protótipos futuros, o produto de *software* possa atender não só ao propósito inicial como também às novas funcionalidades que podem ser agregadas ao modelo.

Capítulo 5

Conclusão

Os estudos realizados para o desenvolvimento do Protótipo de uma Aplicação Web para Autenticação de Usuários em Redes *Wireless* para os Órgãos Públicos do Estado do Tocantins utilizando *FOSS* resultaram na aquisição de novos conhecimentos que contribuíram para a formação profissional e acadêmica. A partir da utilização de técnicas e práticas obtidas no curso de Pós Graduação *"Latu Sensu"* em *"Administração em Redes Linux"*, foi possível concluir o primeiro protótipo do projeto.

5.1 Considerações Finais

O protótipo da aplicação de autenticação web para Redes *Wireless* utilizando *FOSS* consiste numa alternativa de baixo custo no intuito de construir uma "camada" em busca de um ambiente seguro. O uso das ferramentas baseadas em *Software Livre* e *Software Aberto* fornecem um leque de possibilidades para a construção de mecanismos de segurança. Não obstante o resultado obtido no primeiro protótipo do projeto ser usado em redes *Wireless*, ele pode ser usado para isolar sub-redes *Ethernet*, revelando a flexibilidade nas aplicações deste projeto em outros cenários.

O primeiro protótipo da aplicação de autenticação web para Redes *Wireless* nos Órgãos Públicos do Estado do Tocantins utilizando *FOSS* foi construída dentro de um ambiente de testes do Centro Operacional de Redes do Poder Judiciário, foram feitos testes *in loco* alcançado os objetivos iniciais de forma satisfatória. Contudo, para ser usado no ambiente de produção para o qual foi projetado, são necessários investimentos em pesquisa e desenvolvimento com objetivo de promover melhorias no projeto.

A combinação das diferentes tecnologias usadas no projeto como: o desenvolvimento da aplicação *Web* escrita em PHP, o desenvolvimento de *scripts shell*

e a configuração dos serviços de rede, mostrava-se a princípio, um produto de *software* bizarro. Entretanto, ao conseguir fazer essas tecnologias trabalharem de forma harmônica em prol da construção do projeto, verificou-se não só sua viabilidade, como a abertura de um leque de funcionalidades que podem ser agregadas.

O projeto possui importância para o meio acadêmico e comercial, pois, trata-se de um mecanismo de autenticação de usuário baseado em tecnologia de *Software Livre*, que busca ter o controle em tempo real dos usuários. A automatização de construção e remoção das regras dos usuários em tempo de execução é o principal objetivo alcançado neste primeiro protótipo. Contudo, o projeto encontra-se em desenvolvimento e ainda existem várias funcionalidades a serem adicionadas.

Até o presente momento o conjunto de tecnologias empregadas no projeto contemplam as funções básicas da autenticação e o gerenciamento das regras de *firewall* em tempo de execução. Todavia, há a necessidade de criação de novos recursos como a criação de *log* de usuário, controle de banda e geração de relatórios.

5.2 Projetos Futuros

Um passo importante foi dado no tocante ao desenvolvimento do primeiro protótipo da aplicação de autenticação web para Redes *Wireless* utilizando *FOSS*. Através desse protótipo, conseguiu-se obter o objetivo base do projeto que é fazer a autenticação dos usuários e o gerenciamento das regras do *firewall* em tempo de execução. Porém existe a necessidade de acrescentar novas funcionalidades como:

Criação do módulo de *Log*: criação de *logs* de acessos dos usuários para fins de auditoria;

Interface: promover melhorias na interface *web* e adicionar funcionalidades como informar o tempo de acesso e a velocidade;

Banco de Dados: acrescentar novos campos com informações adicionais do usuário, campo para o cadastro de velocidade de acesso por usuário, informações de data e hora;

Função Sair: automatizar a função sair, pois na versão atual é necessária a ação do usuário fazer o *logout*;

Proxy: criação do módulo de cadastro de regras de *firewall* baseadas em controle de conteúdo através da manipulação do arquivo do *squid*, análogo a manipulação do arquivo de *firewall*, verifica-se a possibilidade manipular o arquivo do *squid* para inserir e remover as regras em tempo de execução;

Manipulação das regras de *firewall*: promover melhorias no processo de inserção e remoção das regras com o acréscimo do controle da numeração de cada regra do usuário;

Segurança: Uso de Apache com *SSL*[1] para acrescentar segurança no acesso do navegador dos computadores com o servidor de autenticação.

O protótipo implementado até o presente momento é válido pois dá subsídio para implementações de versões futuras. Para este projeto, verifica-se a necessidade de um estudo mais aprofundado das tecnologias de *Sofware Livre* na busca das melhorias necessárias para dar continuidade a aplicação de autenticação web para Redes *Wireless* utilizando *FOSS*.

[1] SSL - Socket Secure Layer

Referências Bibliográficas

APACHE. The apache software foundation. *Apache*, Abr 2006. `http://www.apache.org/licenses/` [Online; accessado 23-Abril-2006].

BENTES AMAURY, C. S. E. F. *Guia oficial para administradores Red Hat Linux*. 3. ed. Rio de Janeiro: Campus, 2003.

BRASIL, C. F. . *Constituição da República Federativa do Brasil (1988): Edição contendo as Emendas constitucionais de n.1 de 1992, a 32 de 2001*. 17.. ed. Imprensa Nacional: Editora Brasilia, 2001.

CORREIA LUIZ HENRIQUE ANDRADE, R. M. d. A. S. *Redes de Computadores*. 1. ed. Lavras - MG: UFLA/FAEPE, 2002.

FREEBSD. Perguntas mais freqüentes sobre freebsd 2.x, 3.x e 4.x. *FreeBSD*, Nov 2005. `http://www.freebsd.org/doc/pt_BR.ISO8859-1/books/faq/introduction.html#WHAT-IS-FREEBSD` [Online; accessado 22-Abril-2006].

GOMES SOARES, G. L. S. C. L. F. *Rede de Computadores: das MANs LANs e WANs às Redes ATM*. 2. ed. Rio de Janeiro, RJ: Editora Campus, 1995.

MARCELO, A. *Firewall em Linux pra pequenas corporações*. 1. ed. Rio de Janeiro: Bradport, 2003.

MATOS, L. *Guia Profissional de Redes Wireless*. 1. ed. São Paulo, SP: Digerati Comunicação e Tecnologia Ltda, 2005.

NETO, Á. P. *PostgreSQL: técnicas avançadas:versões open source: soluções para desenvolvedores e administradores de banco de dados*. Série: Banco de dados. São Paulo: Érica, 2003.

NEVES, J. C. *Programação Shell Linux*. 5. ed. Rio de Janeiro: Brasport, 2005.

PFSENSE. The pfsense web site. *pfSense*, Abr 2006. http://www.pfsense.com [Online; accessado 23-Abril-2006].

POSGRESQL. Posgresql: Perguntas frequentes (faq) sobre postgresql. *PosgreSQL*, Jan 2005. http://www.postgresql.org/docs/faqs.FAQ_brazilian.html#1.2 [Online; accessado 20-Abril-2006].

ROSS, j. *WI-FI - Instale, configure, e use redes wireless, do original The Book of Wi-Fi*. 1. ed. Rio de Janeiro, RJ: Alta Books Ltda, 2003.

TERADA, R. *Segurança de Dados: Criptografia em Redes de Computadores*. 1. ed. São Paulo: Edgard Blücher, 2000.

UCHôA, K. C. A. *Introdução a Cibercultura*. 3. ed. Lavras - MG: UFLA/FAEPE, 2003.

UNESP. Unesp: Faq sobre o free software. *UNESP free*, Jan 2000. http://w3.unesp.br/unespfree/unespfree_faq.htm [Online; accessado 12-Novembro-2005].

WALACE, W. S. *Php5: Conceitos, Programação e integração com Banco de Dados*. 1. ed. São Paulo: Érica, 2004.

Apêndice A

O Código Fonte do Protótipo de uma Aplicação de Autenticação Web para Redes *Wireless* para os Órgãos Públicos do Estado do Tocantins utilizando *FOSS*

Os capítulos 2 e 4 tratam dos modelos lógicos e dos resultados obtidos acerca da implementação do primeiro protótipo da aplicação via web. Estes capítulos descrevem que o processo de autenticação do usuário é feito através de dois módulos principais, um via aplicação web e um segundo módulo implementado através de *script shell*.

A.1 Módulo Via Web

A aplicação via web é responsável por fazer a gerência e autenticação dos usuários. Este módulo se subdivide em área de administração e área do usuário. Ná área de Administração é possível cadastrar, excluir, consultar e alterar os dados dos usuários. Na área de usuário é feito apenas o *login* e em seguida liberado o acesso de acordo com as regras de *firewall* cadastradas no banco de dados.

A página inicial *login.php* é composta por vários arquivos cada um com uma função. Estes arquivos são responsáveis por fazer o controle de sessão, validação dos campos e construir a estrutura de apresentação da página.

A.1.1 *Login* do Administrador

Ao realizar o *login* informando usuário e senha, a aplicação redireciona o usuário para a página *adm_geral.php*. A figura A.1 ilustra o código fonte do arquivo *login.php*.

```
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/sessao.php");
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/funcoes.php");
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/estrutura.php");

sessao();
$_SESSION = array();
@session_destroy();
// cabecalho(0);
 $file =  "autenticacao.php";
 tela_login($file);
// fecha_estrutura();

function tela_login($file)
{
 echo "<table␣WIDTH='770'><tr><td␣height=20></td></tr><tr><td␣align=center>";
 echo "<table␣border=0␣␣cellspacing=0␣cellpadding=0␣bordercolor='black'␣width=280>␣";
 echo "<tr>\n";
 echo "<td␣width=100%␣align=center␣valign=middle␣style='border:1␣solid␣black'>\n";
 echo "<table␣border=0>\n";
 echo "␣<tr>\n";
 echo "␣␣<td␣colspan=2␣height=10\n";
 echo "␣␣</td>\n";
 echo "␣</tr>\n";
 echo "␣<tr>\n";
 echo "␣␣<td␣colspan=2␣align='center'>\n";
 echo "␣␣</td>\n";
 echo "␣</tr>\n";
 echo "␣<tr>\n";
 echo "␣␣<td␣colspan=2␣align='center'>\n";
 echo "␣␣␣<font␣face='Arial'␣size=4␣color=green><b>".$nome_sistema."</b></font><p>\n";
 echo "␣␣</td>\n";
 echo "␣</tr>\n";
 echo "␣<tr>\n";
 echo "␣␣<td␣align=center>\n";
 echo "␣␣␣<form␣method='post'␣action='$file'>\n";
 echo "␣␣␣<font␣face='Arial'␣color=black␣size=2><b>Login:␣</b></font>\n";
 echo "␣␣␣<input␣type='text'␣size=10␣name='login_form'␣value='$login_form'␣maxlength=12␣style='border:1␣solid␣black'><p>\n";
 echo "␣␣␣<font␣face='Arial'␣color=black␣size=2><b>Senha:␣</b></font>\n<input␣type='password'␣size=8␣name='senha_form'␣␣style='border:1
 echo "␣␣␣<input␣type='submit'␣name='enviar'␣value='Enviar'␣style='border:1␣solid␣black'>  \n";
 echo "␣␣␣<input␣type='reset'␣name='limpar'␣value='Limpar'␣style='border:1␣solid␣black'>\n";
 echo "␣␣␣</form>\n";
 echo "␣␣</td>\n";
 echo "␣</tr>\n";
 echo "</table>\n";
 echo "</td>\n";
 echo "</tr>\n";
 echo "</table>\n";
 echo "</td></tr></table>";

}
?>
```

Figura A.1: Arquivo *login.php*

A figura A.2 ilustra o código da página *adm_geral.php* que tem a função de verificar se o usuário logado é administrador ou não. Sendo Administrador, são carregados os módulos de gerenciamento.

Confome visto na figura A.2, a página *adm_geral.php* carrega os módulos de

sessão, conexão com o banco de dados e as classes de usuários e serviços de redes. Nesta página, estão disponíveis as funções de cadastro, exclusão, consulta e alteração de usuários. Respectivamente as figuras A.3, A.4, A.5 e A.6 correspondem aos códigos de cadastro, consulta, exclusão e alteração dos usuários, formando assim o módulo de gerenciamento do sistema.

A.1.2 *Login* do Usuário

O procedimento de *login* do usuário é feito análogo ao de um administrador. O usuário ao logar no sistema é redirecionado para a página *adm_geral.php*, porém esta ao verificar que não se trata de um adminsitrador não carrega os módulos de gerenciamento. A figura A.2 ilustra o código que faz a geração do arquivo com o nome do usuário contendo as regras de *firewall* de acordo com seu perfil cadastrado no banco de dados. Em seguida executa uma chamada de sistema acionando o *shell script*.

A.2 Módulo de *Script*

O módulo de *script* é construído a partir do *script* de geração do arquivo de *firewall* usando arquivos numa estrutura de diretórios pré-estabelecida.

A.2.1 Estrutura de diretórios

A estrutura básica de diretórios dos arquivos é descrito na figura A.7.

Dentro do diretório seguranca/cw/ estão os scritps de criação do *firewall*. No diretório "seguranca/cw/fw-geral/" estão os arquivos contendo as partes incial e final das regras do *firewall*. Em "seguranca/cw/fw-user/" é gerado o arquivo com as regras do *firewall* do usuário pela aplicação web.

A.2.2 Arquivo: cria-script.sh

```
#!/bin/sh
touch /usr/local/www/seguranca/cw/rc.firewall
chown root.root /usr/local/www/seguranca/cw/rc.firewall
chmod 777 /usr/local/www/seguranca/cw/rc.firewall

cat /usr/local/www/seguranca/cw/fw-geral/fw-cabecalho > /usr/local/www/seguranca/cw/rc.firewall

cd /usr/local/www/seguranca/cw/fw-user
for i in *; do
        cat $i >> /usr/local/www/seguranca/cw/rc.firewall;
done

cat /usr/local/www/seguranca/cw/fw-geral/fw-rodape >> /usr/local/www/seguranca/cw/rc.firewall

chown root.root /usr/local/www/seguranca/cw/rc.firewall
chmod 777 /usr/local/www/seguranca/cw/rc.firewall
```

```
/usr/local/www/seguranca/cw/rc.firewall
                        echo "Firewall␣atualizado!!!!!!"
echo "..............................................␣ok"
sleep 2
```

A.2.3 Arquivo: rc.firewall

Durante a execução do script *cria-script.sh* é gerado o arquivo *rc.firewall* contendo as configurações iniciais do cabeçalho do *firewall* e em seguida, a concatenação das regras dos usuários, e por fim é feita a concatenação do arquivo de rodapé que contém o fechamento das regras. A figura A.8 ilustra o arquivo *firewall* gerado após o *login* dos usuários.

Nota: O *firewall* gerado na ilustração deste exemplo serve apenas de demonstração pois, o arquivo de *firewall* que será usado para o ambiente de produção será construído com opções adicionais de acordo com a realidade do ambiente computacional.

```
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/sessao.php");
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/conexao.php");

sessao();
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/estrutura.php");
abre_estrutura(1);

include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/usuario/class_usuario.php");
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/usuario/class_servico_usuario.php");
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/usuario/class_servico.php");

$servico_usuario = new servico_usuario($conexao);

$servico_usuario->sql->tabela = "vw_usuario_servico";
$servico_usuario->busca("id_usuario",$_SESSION["session_id_usuario"]);

echo "<table␣width=50%␣align=center>";
echo "<tr><th>çServios</th></tr>";
$conteudo="";
$ip="";
$ifacesaida="␣via␣vr0";
$ip_proxy="";

if (getenv(HTTP_X_FORWARDED_FOR))
{
        if (getenv(HTTP_CLIENT_IP))
        {
                $ip=getenv(HTTP_CLIENT_IP);
        }
        else
        {
                $ip=getenv(HTTP_X_FORWARDED_FOR);
        }
        $ip_proxy=getenv(REMOTE_ADDR);
}
else
{
    $ip="ipfw␣add␣permit␣tcp␣from␣".getenv(REMOTE_ADDR);
}


for ($i=0;$i<$servico_usuario->query->nlinha;$i++)
{
        $login = $servico_usuario->query->dados["login"];
    echo "<tr><td>".$servico_usuario->query->dados["nome"]."</td></tr>";
    echo "<tr><td>".$servico_usuario->query->dados["descricao"]."</td></tr>";

   # $conteudo = $conteudo.$ip.";".$servico_usuario->query->dados["login"]."␣any␣to␣any␣to␣".$servico_usuario->query->dados["porta"].";
    $conteudo = $conteudo.$ip."␣to␣any␣".$servico_usuario->query->dados["porta"].$ifacesaida."\r\n";
    $servico_usuario->query->proximo();
}
echo "</table>";

        $filename = $_SERVER["DOCUMENT_ROOT"]."/seguranca/cw/fw-user/$login.txt";

    if (!$handle = fopen($filename, 'w')) {
         print "Erro␣abrindo␣arquivo␣($filename)";
         exit;
    }

    if (!fwrite($handle, $conteudo)) {
        print "Erro␣escrevendo␣no␣arquivo␣($filename)";
        exit;
    }

//    print "Sucesso:␣escrito␣($somecontent)␣no␣arquivo␣($filename)";
//    fclose($filename);

// ÇÃEXECUO DA CHAMADA DE SISTEMA PARA ACIONAR O SCRIPT SHELL
shell_exec(./cria-script.sh);

fecha_estrutura();
?>
```

Figura A.2: Arquivo *adm_geral.php*

```
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/seguranca/autorizacao.php");
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/sessao.php");
sessao();
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/estrutura.php");
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/conexao.php");

$cod_modulo = $_SESSION['session_cod_modulo'];
abre_estrutura($cod_modulo);
$cad_usuario= new usuario($conexao);
if (!isset($_POST["ins_usuario"]))
{
        $cad_usuario->tela_cadastro("Cadastrar␣");
}
else
{
        if ($_SESSION['flag_root']=='1')
                $id_empresa = $_POST["id_empresa"];
        else
                $id_empresa = $_SESSION['session_id_empresa'];


        $cad_usuario->login=$_POST["login_usu"];
        if($cad_usuario->busca("login",$cad_usuario->login))
        {
            echo "<script>alert('Esse␣login␣áj␣existe');</script>";
            $cad_usuario->login=$_POST["login_usu"];
            $cad_usuario->senha=md5($_POST["senha_usu"]);
        $cad_usuario->id_pessoa = intval($_POST["id_pessoa"]);

            $cad_usuario->id_empresa = intval($id_empresa);

            $cad_usuario->tela_cadastro("Cadastrar␣");
        }
        else
        {
            $cad_usuario->login=$_POST["login_usu"];
            $cad_usuario->senha=md5($_POST["senha_usu"]);
            $cad_usuario->id_pessoa = intval($_POST["id_pessoa"]);
            $cad_usuario->id_empresa = intval($id_empresa);
            $cad_usuario->ativo = $_POST["ativo"];

            if($cad_usuario->incluir())
            {
                $cad_grupo_usuario= new grupo_usuario($conexao);
                $cad_grupo_usuario->id_usuario=$cad_usuario->ultimo_id("id_usuario");
                for($c=0;$c<$_POST["numGrupo"];$c++)
                {
                    if($_POST["grupo$c"]!="")
                    {
                        $cad_grupo_usuario->id_grupo=$_POST["grupo$c"];
                        $cad_grupo_usuario->incluir();
                        }
                }
                $cad_servico_usuario= new servico_usuario($conexao);
                $cad_servico_usuario->id_usuario=$cad_usuario->ultimo_id("id_usuario");
                for($c=0;$c<$_POST["numServico"];$c++)
                {
                    if($_POST["servico$c"]!="")
                    {
                        $cad_servico_usuario->id_servico=$_POST["servico$c"];
                        $cad_servico_usuario->incluir();
                        }
                }

                echo "<script>alert('áUsurio␣Cadastrado␣Com␣Sucesso');</script>";
                echo "<script␣language='JavaScript'>";
//              echo "location.href='cad_usuario.php'";
                echo "</script>";
            }
            else
            {
                //echo $cad_usuario->sql->sql;
                echo "<script>alert('Erro␣na␣ãIncluso␣do␣áUsurio');</script>";
            }
        }
}
fecha_estrutura();
?>
```


Figura A.3: Arquivo *cad_usuario.php* - Função Cadatrar Usuário

```
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/seguranca/autorizacao.php");
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/sessao.php");
sessao();
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/estrutura.php");
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/conexao.php");

$cod_modulo = $_SESSION['session_cod_modulo'];
abre_estrutura($cod_modulo);
$cons_usuario = new usuario($conexao);

limpa_sessao_busca();
lista_titulo("Consultar áUsurio");
   echo "<form name=\"form\"  method='post' action='res_consulta.php'>";
   echo "<TABLE cellSpacing=0 cellPadding=2 width='100%' border=0>";
   echo "<TR>";
   echo "<TD class=lbForm width=100>Login:</TD>
          <TD class=lbFormLeft><INPUT type='text' size=10 name=login_usr></TD></TR>";

        echo "</table>";
    echo "<input type='hidden' name=busca value='usuario'>";
    echo "<input type='hidden' name=id_modulo value='".$_GET["cod_modulo"]."'>";

    echo "<TABLE cellSpacing=0 cellPadding=2 width='100%' border=0>
          <TBODY>
          <TR>
          <TD align=right colSpan=2><INPUT class=submit name='bt_busca' type=submit value='Consultar'>
          </TD></TR></TBODY></TABLE>";

     echo "</form>";
fecha_estrutura();
?>
```

Figura A.4: Arquivo *cons_usuario.php* - Função Consultar Dados do Usuário

```
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/seguranca/autorizacao.php");
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/sessao.php");
sessao();
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/estrutura.php");
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/conexao.php");

$cod_modulo = $_SESSION['session_cod_modulo'];

abre_estrutura($cod_modulo);

  $exclusao= new usuario($conexao);

if($_POST["exclu"]!="sim")
{
 echo "<form name=\"exc_form\" method='post' action='exc_usuario.php'>";
 echo "<input type='hidden' name='exclu' value=''>";
 echo "<input type='hidden' name='id_exc' value='".$_GET["id_exc"]."'></form>";
 echo "<script>if (confirm('Tem certeza que deseja excluir o áusurio $desc ?'))
            {document.exc_form.exclu.value='sim';
            document.exc_form.submit(); }
            else
            location.href='/seguranca/controle/usuario/cons_usuario.php';";
 echo "</script>";

}
else
 {
   $exclusao->id_usuario=intval($_POST["id_exc"]);
   if ($exclusao->excluir())
    {
     echo "<script>alert('áUsurio íExcludo com Sucesso!');</script>";
     echo "<script language='JavaScript'>";
     echo "location.href='/seguranca/controle/usuario/cons_usuario.php';";
     echo "</script>";

    }
 }
fecha_estrutura();
?>
```

Figura A.5: Arquivo *exc_usuario.php* - Função Excluir Usuário

```
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/seguranca/autorizacao.php");
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/sessao.php");
sessao();
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/estrutura.php");
include($_SERVER["DOCUMENT_ROOT"]."/seguranca/controle/conexao.php");

$cod_modulo = $_SESSION['session_cod_modulo'];

abre_estrutura($cod_modulo);

        $cad_usuario= new usuario($conexao);

        if (!isset($_POST["ins_usuario"]))
        {
            $id_usuario = intval($_GET["id_altera"]);
            $cad_usuario->busca("id_usuario",$id_usuario);
            $cad_usuario->tela_cadastro("Alterar ");
        }
        else
        {
            $id_usuario = intval($_POST["id_altera"]);
            $cad_usuario->id_usuario = $id_usuario;

            $cad_usuario->id_empresa =$_POST["id_empresa"];
            $cad_usuario->login=$_POST["login_usu"];
        $cad_usuario->id_pessoa=$_POST["id_pessoa"];
        $cad_usuario->ativo=$_POST["ativo"];

        if($_POST["senha_usu"]=="")
                $desc = "sem_senha";
        else
        {
                $desc = "com_senha";
            $cad_usuario->senha=md5($_POST["senha_usu"]);
        }



            if($cad_usuario->alterar($desc))
            {
            $cad_grupo_usuario= new grupo_usuario($conexao);
            $cad_grupo_usuario->id_usuario = $id_usuario;
            if($cad_grupo_usuario->excluir())
            {
                    for($c=0;$c<$_POST["numGrupo"];$c++)
                    {
                        if($_POST["grupo$c"]!="")
                        {
                        $cad_grupo_usuario->$id_usuario = $id_usuario;
                            $cad_grupo_usuario->id_grupo=$_POST["grupo$c"];
                            $cad_grupo_usuario->incluir();
                        }
                    }
            }


            $cad_servico_usuario= new servico_usuario($conexao);
                $cad_servico_usuario->id_usuario=$id_usuario;
            if($cad_servico_usuario->excluir(2))
                for($c=0;$c<$_POST["numServico"];$c++)
                {
                    if($_POST["servico$c"]!="")
                    {
                                $cad_servico_usuario->id_usuario=$id_usuario;
                        $cad_servico_usuario->id_servico=$_POST["servico$c"];
                        $cad_servico_usuario->incluir();
                        }
                }
           //   echo $cad_usuario->sql->sql;
                echo "<script>alert('áUsurio Alterado Com Sucesso');</script>";
                echo "<script language='JavaScript'>";
//              echo "location.href='cons_usuario.php'";
                echo "</script>";

                }
            else
            {
                //echo $cad_usuario->sql->sql;
                echo "<script>alert('Erro na çãAlterao do áUsurio');</script>";
            }
        }
fecha_estrutura();
?>
```


Figura A.6: Arquivo *alt_usuario.php* - Função Alterar Dados do Usuário

```
\lstset{language=VBScript,basicstyle=\tiny\ttfamily,stepnumber=5}
\begin{lstlisting}
tjsrv02# ls -l
total 9
-rwxrwxrwx  1 root  wheel  548 Nov 21 17:39 cria-script.sh
drwxr-xr-x  2 root  wheel  512 Nov 21 17:35 fw-geral
drwxr-xr-x  2 root  wheel  512 Nov 21 17:38 fw-user
-rwxrwxrwx  1 root  wheel  664 Nov 21 18:13 rc.firewall
tjsrv02# cd fw-geral/
tjsrv02# ls -l
total 4
-rw-r--r--  1 root  wheel  228 Nov 21 19:34 fw-cabecalho
-rw-r--r--  1 root  wheel  110 Nov 21 19:32 fw-rodape
tjsrv02# cd ../fw-user/
tjsrv02# ls -l
total 4
-rw-r--r--  1 root  wheel  217 Nov 21 20:10 fw-marcelo
-rw-r--r--  1 root  wheel  109 Nov 21 20:10 fw-marciley
tjsrv02#
```

Figura A.7: Estrutura de Diretórios

```
#------------------------ çcabealho -------------------
#!/bin/sh
ipfw flush

ipfw="/sbin/ipfw␣-q"

# ------ libera acesso na porta 80 para ápgina de çãautenticao
ipfw add permit tcp any to 10.33.2.112 80 via vr0

# ----- regras dos áusurios -----

ipfw add permit tcp from 10.33.2.81 to any 80 via vr0
ipfw add permit tcp from 10.33.2.81 to any 443 via vr0
ipfw add permit tcp from 10.33.2.81 to any 21 via vr0
ipfw add permit tcp from 10.33.2.81 to any 22 via vr0
ipfw add permit tcp from 10.33.2.21 to any 80 via vr0
ipfw add permit tcp from 10.33.2.21 to any 443 via vr0

ipfw add 65000 deny ip from any to any
# ----------------------Fechamento do Firewall -----------------------
```

Figura A.8: Arquivo de *firewall* gerado após o *login* dos usuários