UNIVERSIDADE FEDERAL DE ALFENAS
INSTITUTO DE CIÊNCIAS EXATAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Matheus Virtudes

FERRAMENTA PARA AUTOMATIZAR A CONFIGURAÇÃO DE SERVIDORES *PROXY*, *DNS* E *DHCP* NO LINUX

Alfenas, 27 de junho de 2011.

UNIVERSIDADE FEDERAL DE ALFENAS
INSTITUTO DE CIÊNCIAS EXATAS
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FERRAMENTA PARA AUTOMATIZAR A CONFIGURAÇÃO DE SERVIDORES *PROXY*, *DNS* E *DHCP* NO LINUX

Matheus Virtudes

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Alfenas como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação.

Orientador: Prof. M.Sc. Flavio Barbieri Gonzaga

Alfenas, 27 de junho de 2011.

Matheus Virtudes

FERRAMENTA PARA AUTOMATIZAR A CONFIGURAÇÃO DE SERVIDORES *PROXY*, *DNS* E *DHCP* NO LINUX

A Banca examinadora abaixo-assinada aprova a monografia apresentada como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação pela Universidade Federal de Alfenas.

Prof. Tomás Dias Sant' Ana

Universidade Federal de Alfenas

Prof. Ricardo Marques da Costa

Universidade Federal de Alfenas

Prof. Flavio Barbieri Gonzaga (Orientador)

Universidade Federal de Alfenas

Alfenas, 27 de junho de 2011.

A minha mãe Maria, aos meus irmãos Márcio, Marcos e Dayana, e em especial a minha namorada Letícia.

AGRADECIMENTO

Agradeço primeiramente a Universidade Federal de Alfenas pela constante busca na qualidade do ensino e pesquisa, e por ter me proporcionado um período agradável de estudo e convívio social.

A meu orientador Flávio, que mesmo atarefado se propôs a me auxiliar nesse projeto. Dedicando seu tempo e conhecimento em prol do meu objetivo.

Ao mestre Tomás pelos excelentes livros de administração de sistemas Linux.

Aos amigos que muito me ajudaram a chegar até aqui, em especial ao Guilherme Rodrigues, que mais que um amigo se tornou um irmão.

Aos meus familiares por compreenderem que muitas das minhas ausências eram por um bem maior.

E a minha namorada Letícia que sempre me fez querer o melhor.

"APRENDER É A ÚNICA COISA DE QUE A MENTE NUNCA SE CANSA, NUNCA TEM MEDO E NUNCA SE ARREPENDE."

(LEONARDO DA VINCI)

RESUMO

Atualmente os servidores são fundamentais para um bom desempenho da rede, e nesse contexto o Linux tem desempenhado um papel importante. Cada vez mais as organizações públicas e privadas tem optado por servidores Linux, e isso se deve ao fato do sistema ser robusto e seguro. Dessa forma, grande parte dos servidores que rodam algum tipo de serviço na Internet utilizam sistema operacional Linux. São diversos os serviços que um servidor pode executar, porém, em uma rede interna comumente são encontrados os serviços *Proxy*, *DNS* e *DHCP*. O *Proxy* controla o acesso a Internet, o DNS converte nomes em endereços IP, e o *DHCP* distribui endereços IP para as máquinas da rede. Esses serviços em conjunto controlam de maneira eficaz o fluxo e a carga de dados que trafegam na rede. Porém, a configuração desses serviços pode não ser trivial, uma vez que de acordo com a versão do *kernel* do sistema, a configuração a ser feita para colocar um serviço ativo pode mudar. Este trabalho teve como objetivo construir uma ferramenta em *Shell Script* para auxiliar administradores de redes, e até mesmo pessoas com conhecimentos básicos em configuração de servidores, automatizando as tarefas de configuração dos serviços citados anteriormente, sem a necessidade de interagir manualmente com os diversos arquivos de configuração que cada serviço possui. Nesse sentido a ferramenta se mostrou eficaz, pois reduziu a quantidade de configurações digitadas pelo usuário, alem disso fez com que os erros referentes aos detalhes de cada serviço fossem tratados e evitados.

Palavras-Chave: Servidor, Linux, *Proxy*, *DNS*, *DHCP*, *Shell Script*

ABSTRACT

Currently servers are fundamental for a good network performance, and in this context Linux has played an important role. More and more, public and private organizations have chosen Linux Servers, due to the fact that the system is robust and safe. For this reason, most of the servers running some kind of service on the Internet use Linux operational system. The services a server can run are various, however in an internal network Proxy, DHCP and DNS services are commonly found: Proxy controlling Internet access, DNS converting names into IP addresses, and DHCP distributing IP addresses to network machines. By these joined services a significant gain in performance is already obtained, for together they efficiently control the flow and load of data transiting in the network. However, the configuration of these services might not be trivial. For example, depending on the system's kernel version, the configuration to be created, in order to put an active service, can change. The aim of this work is to build a tool in Shell Script to help network administrators, or even people with basic knowledge in servers' configuration, to automate the task of configuration in Linux of each of the previously mentioned services without the necessity of manually interacting with the various configuration files that each service has.

Keywords: *Server, Linux, Proxy, DNS, DHCP, Shell Script*

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES

DNS - *Domain Name System* (Sistema de Nomes de Domínio)

DHCP - *Dynamic Host Configuration Protocol* (Protocolo de Configuração Dinâmica de Endereços de Rede)

IEEE – Instituto de Engenheiros Elétricos e Eletrônicos

SMTP - *Simple Mail Transfer Protocol* (Protocolo Simples de Transferência de Mensagens)

WWW - *World Wide Web* (Rede de Alcance Mundial)

DPN - Domínio de Primeiro Nível

MAC - *Media Access Control* (Mídia de Controle de Acesso)

HTTP - *HyperText Transfer Protocol* (Protocolo de Transferência de Hipertexto)

FTP - *File Transfer Protocol* (Protocolo de Transferência de Arquivo)

SUMÁRIO

1
Introdução

Desde o surgimento das primeiras redes de computadores que os pesquisadores buscam melhorar e agilizar a comunicação entre as máquinas que compõem a rede. Nesse processo de aperfeiçoamento descobriram a necessidade de designar máquinas ou até mesmo sistemas computacionais para desempenhar a tarefa de fornecer algum tipo de serviço aos demais componentes da rede.

Para melhor ilustrar esse fato imagina-se uma empresa onde existam dez máquinas e que cada uma acessa a mesma página da Internet varias vezes ao dia. Para tal cenário, seria mais fácil que apenas uma das máquinas fizesse o acesso a página e a armazenasse localmente para que as demais utilizassem o conteúdo. Sabendo que a velocidade de acesso na rede interna entre as máquinas é maior do que o acesso com a Internet, pode-se notar um ganho de velocidade significativo. Assim, um Servidor permite reduzir o fluxo na rede centralizando em si a obtenção de recursos e dados solicitados por outras maquinas.

Um servidor para ter um bom desempenho deve conter um Sistema Operacional robusto e seguro. Nos dias de hoje duas linhas de sistemas operacionais se destacam, os sistemas Windows e aqueles oriundos do UNIX (Linux).

A configuração de servidores Linux é o foco do presente trabalho. Comumente executada em modo texto por administradores de redes, a configuração de servidores não é uma tarefa trivial. Cada serviço a ser disponibilizado requer a configuração geralmente de vários arquivos, com diferentes localizações no sistema.

1.1 Justificativa e Motivação

Servidores Linux são amplamente utilizados, entre outros motivos, pela sua segurança e confiabilidade. Porém, configurar tais servidores nem sempre é uma tarefa simples. Existem detalhes específicos em cada servidor, como por exemplo, pode-se destacar a configuração do servidor *DNS*, onde para que o serviço funcione corretamente é necessário que o arquivo de configuração seja identado.

Outra questão importante a ser avaliada está relacionada ao fato da maioria das ferramentas existentes serem direcionadas para o gerenciamento dos servidores e não para a configuração em si. Além disso, essas ferramentas funcionam em modo gráfico o que torna o usuário muitas vezes dependente das mesmas. Como exemplos dessas ferramentas é possivel destacar o Webmin[1] que é uma interface baseada na web para administração de sistemas para Unix (Webmin, 2010), e o Kyapanel[2] que "é um painel para facilitar a administração de servidores GNU/Linux melhorando a administração da rede" (Kyapanel, 2010).

A partir do exposto, viu-se necessário o desenvolvimento, utilizando a linguagem *Shell Script,* de uma ferramenta sem interface gráfica para auxiliar o usuário na configuração dos servidores, funcionando como uma camada de controle entre o usuário e os arquivos de configuração de cada servidor.

---

1 http://www.webmin.com site da ferramenta Webmin.

2 http://www.kyapanel.com site da ferramenta Kyapanel

1.2 Problematização

Como promover no Linux a configuração de forma automatizada e rápida dos servidores *Proxy*, *DNS* e *DHCP*, dado que cada um destes possuem configurações específicas e atuam em arquivos diferentes no Sistema Operacional? Uma ferramenta que auxilie o administrador do sistema nessa tarefa será a solução?

1.3 Objetivos

1.3.1 Gerais

Visando simplificar o trabalho de configuração de servidores, é proposta uma ferramenta para automatizar a configuração dos servidores. Dentre os servidores existentes, para atendimento dos objetivos, foram abordados os seguintes serviços: *Proxy*, *DNS* e *DHCP*, por serem básicos em boa parte das redes de computadores.

1.3.2 Específicos

- Estudar a configuração de servidores Linux
- Estudar a linguagem *Shell Script*
- Pesquisar a respeito de ferramentas já existentes
- Implementar a ferramenta
- Configurar servidores através da ferramenta desenvolvida
- Testar as configurações realizadas através da ferramenta desenvolvida

2
Revisão Bibliográfica

2.1 Linux

O Linux é um sistema operacional baseado no padrão criado em 1990 pela IEEE (Instituto de Engenheiros Elétricos e Eletrônicos) o *POSIX* (*Portable Operating System Interface Unix*), sendo este padrão desenvolvido inicialmente com o objetivo de definir as características dos sistemas operacionais Unix (Ferreira, 2003).

No inicio o Unix era distribuído gratuitamente para universidades, porém após a sua expansão comercial a AT&T, sua desenvolvedora, passou a proibir a utilização do código-fonte do Unix para estudo em cursos. Andrew Tanenbaum (2000) percebendo a necessidade de ensinar sobre sistemas operacionais de modo prático desenvolveu o Minix, um pequeno sistema operacional compatível com o Unix, e que não utiliza nem uma linha de código desenvolvido pela AT&T, evitando com isso, as restrições de licenciamento impostas pela empresa (Ferreira, 2003).

Mesmo o Minix sendo de código aberto, sua utilização estava muito restrita para os fins didáticos, e por esse motivo, o finlandês Linus Torvalds da Universidade de Helsinque viu a necessidade de desenvolver "um Minix melhor que o Minix", como ele próprio definiu, iniciando em 1991 o desenvolvimento do Linux. Porém, após algum tempo trabalhando basicamente no *Kernel* do sistema, Linus percebeu que não conseguiria desenvolver o Linux sozinho, e então solicitou a ajuda de programadores do mundo todo através da lista de discussão comp.os.minix (Ferreira, 2003).

Em 5 de outubro de 1991, foi lançado o Linux 0.02 a primeira versão oficial do sistema, desde então o desenvolvimento e aperfeiçoamento do Linux vem sido

trabalhado por vários desenvolvedores, sejam eles empresas ou até mesmo programadores independentes.

Devido a sua portabilidade, estabilidade e segurança tem sido utilizado até os dias de hoje, tanto no meio acadêmico quanto em sistemas comerciais, tendo como principais distribuições o Slackware, Red Hat, Mandriva, Debian, Suse, Fedora e Ubuntu (Tibet, 2001).

2.1.1 Principais Características do Linux

De acordo com Mazioli (2006) segue abaixo algumas das principais características do Sistema Operacional Linux são:

- Multitarefa real
- Multiusuário
- Multiprocesso
- Proteção entre processos executados na memória RAM
- Licenciado de acordo com os termos da GPL
- Suporte a nomes extensos de arquivos e diretórios (255 caracteres)
- Utiliza permissão de acesso a arquivos, diretórios e programas em execução na memória RAM

2.2 Shell Script

No Linux o *shell* é a interface entre o usuário e o *kernel* do sistema. Já o *kernel* é o responsável por acessar e acionar as funcionalidades do *hardware* da máquina, portanto o *shell* representa uma camada mais alto nível que abstrai detalhes específicos de implementação do *kernel* e de funcionamento do *hardware*.

O *shell* do Linux é bem mais poderoso do que se pode imaginar, ao invés de apenas interpretar comandos, ele disponibiliza toda a estrutura de uma linguagem de programação, oferecendo funções, laços de repetição, entre outros (Jargas, 2008).

Da combinação entre *script* e *shell* deriva a linguagem Shell Script, uma linguagem interpretada que fornece uma infinidade de opções para criação de *scripts*, seja para automatizar tarefas ou até mesmo para desenvolver programas complexos. A tabela 1 apresenta uma comparativo entre Shell Script e C.

Tabela 1 – Comparação entre trecho de código em *Shell Script* e C

Trecho de código em Shell Script	Trecho de código em C

1. i=1
2. while [$i -lt 10]
3. do
4. let resto=i%2
5. if [$resto -eq 0]; then
6. echo $i
7. fi
8. let i=i+1
9. done
	1. int i = 1;
	2. while (i < 10)
	3. {
	4. int resto = i % 2;
	5. if (resto == 0){
	6. printf("\n%d", i);
	7. }
	8. i++;
	9. }

Fonte: Dados da Pesquisa

O trecho de código presente na tabela 1 irá imprimir na tela os números pares contidos entre 0 e 10. Em *Shell Script* não há a necessidade de declarar o tipo da variável, e o tipo será identificado durante a execução do programa de acordo com o contexto em que ela se encontra. Esse fato pode ser observado na linha 1, onde em C a variável i foi declarada como inteira, e em *Shell Script* automaticamente foi atribuído o tipo inteiro, pois i recebe o valor 1.

Ao contrário da linguagem C, em *Shell Script* existe expressões especificas para trabalhar com inteiros, no código essas expressões são -lt, -eq e let. O -lt indica menor que, a linha 2 significa que o laço de repetição deve ser executado enquanto i for menor que 10. O let deve ser utilizado sempre que o resultado de uma equação for atribuído a uma variável inteira. E o -eq compara a igualdade entre valores inteiros, o que no

código pode ser observado na linha 5 que compara se o resto da divisão de i por 2 é zero, se (*if*) verdade então (*then*) executará a impressão do valor através do comando echo.

Mais algumas particularidades do *Shell Script* está na utilização dos valores das variáveis. Quando é necessário utilizar o valor contido dentro da variável, esta deve ser antecedida por um $, por exemplo $i. E para atribuir valores é necessário que não haja espaço entre a variável e o sinal de igual, senão a atribuição será interpretada como uma comparação.

Em C os blocos de repetição e condicional são delimitados por {} (abre e fecha chaves), já no Shell Script existem as palavras reservadas *do* e *done* para o laço de repetição *while*, e *then* e fi para o comando condicional *if*.

O objetivo desse tópico foi apenas apresentar algumas das inúmeras particularidades da linguagem Shell Script, detalhes mais específicos podem ser encontrados em *Shell Script* Profissional (Jargas, 2008).

2.3 Servidores

Servidores são sistemas computacionais capazes de oferecer algum tipo de serviço na rede (Torres, 2001). Os Servidores são muito utilizados em redes clientes/servidor, onde uma máquina (servidor) oferece um serviço e as demais máquinas (clientes) fazem uso do mesmo, tudo com o intuito de aumentar o desempenho, segurança e confiabilidade da rede.

Os servidores podem ser dedicados a executar apenas uma tarefa, aumentando assim o desempenho, pois podem responder mais rapidamente as requisições de outras máquinas. Porém é comum encontrarmos servidores que executem vários serviços ao mesmo tempo, e com o desenvolvimento do *hardware* essa característica se tornou ainda mais presente.

Existem vários tipos de servidores, tais como, *Proxy*, *DNS, DHCP, FTP, HTTP, SMTP, POP* entre outros. Nesse trabalho o foco será em servidores *Proxy*, *DNS* e *DHCP* os quais serão detalhados a seguir. A escolha se deu por esses serviços serem básicos, e normalmente disponíveis em redes de computadores de qualquer escala. Serviços como *HTTP*, *SMTP* e *POP* são igualmente importantes, mas já são específicos, podendo ser então uma extensão do presente trabalho.

2.3.1 Servidor Proxy

Em uma rede com servidor *Proxy* todas as requisições de conteúdos na Internet são redirecionadas para o servidor, este então fará uma análise se a solicitação deverá ser barrada ou poderá ser repassada para a Internet, mantendo ainda um *cache* que será usado em requisições futuras.

A principal funcionalidade do servidor *Proxy* é reduzir o trafego na rede. A Figura 1 ilustra o funcionamento desse tipo de servidor. Como pode-se notar todas as requisições da rede interna são enviadas ao *Proxy*, que no primeiro momento verificará se o que esta sendo solicitado é legitimo, ou seja, se o usuário tem permissão de acesso na rede, se a página solicitada não está bloqueada, entre outros. Caso não aconteça nenhum bloqueio, o servidor verificará em sua *cache* se contem uma versão atualizada do que está sendo requerido, em caso positivo retornará o conteúdo sem necessitar fazer uma nova consulta na Internet. O *Proxy* verificará em um tempo pré-estabelecido se o conteúdo armazenado no *cache* está atualizado.

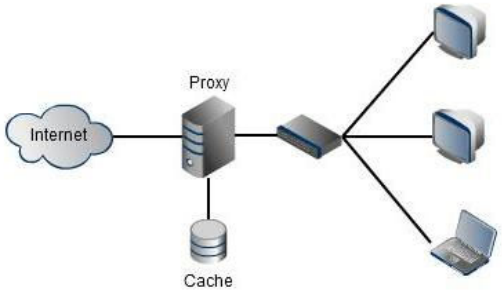

Figura 1 – Servidor *Proxy*

A verificação da atualização de páginas pelo *Proxy* é feita através do GET condicional. As páginas armazenadas no *cache* do servidor contém a data da última modificação, e é essa data que será utilizada para saber a respeito das atualizações. Quando o Proxy necessita saber se as páginas no *cache* estão atualizadas, ele envia um GET condicional ao servidor *Web*, informando a data da última atualização. Se não houve modificações, o servidor responderá a mensagem do *Proxy* apenas informando que não existe atualização, caso contrário, enviará a página com todo o conteúdo. O fato do servidor *Web* responder com uma mensagem informando apenas que não houve atualização, reduz consideravelmente o fluxo na rede (Kurose, 2006)

Existem duas maneiras de configurar um servidor Proxy no Linux através do Squid, uma delas é através do *Proxy* com autenticação, onde cada usuário possui *login* e senha para acessar conteúdos na Internet. Uma vantagem nesse tipo de abordagem está na segurança oferecida, uma vez que apenas usuários previamente cadastrados poderão fazer uso da Internet através da rede. A desvantagem está no fato de se ter que configurar individualmente cada máquina da rede para fazer uso do *Proxy* (ou fazer uma configuração extra no servidor, para já enviar a configuração para o navegador, mas que não funciona em todos os navegadores).

A segunda maneira é através do *Proxy* transparente, onde a desvantagem do *Proxy* com autenticação é suprida por uma regra de redirecionamento de portas feita no *firewall*, assim todas as requisições na porta 80 são repassadas para porta onde o servidor *Proxy* estiver rodando. A desvantagem está em não poder utilizar a autenticação de usuário, pois não pode existir *Proxy* transparente e com autenticação juntos (Morimoto, 2008).

O servidor *Proxy* é composto de regras que são definidas através das diretivas acl e http_access. A acl são listas de parâmetros onde o usuário definirá posteriormente a política de acesso, e são formadas da seguinte maneira **acl nome_da_acl tipo_da_acl string ou arquivo**, por exemplo **acl Safe_port port 80**, significa que foi criado uma acl

chamada Safe_port do tipo port que referencia a porta 80. Através da http_access as acl criadas terão o acesso permitido ou negado através das palavras reservadas *allow* e *deny* respectivamente (Morimoto, 2008).

Neste projeto todas as configurações deste servidor são feitas através do arquivo de configuração squid.conf. Uma configuração simples deste arquivo pode ser pela enumeração das etapas

1. http_port 3128
2. visible_hostname nomeServidor
3. acl all src 0.0.0.0/0.0.0.0
4. http_access allow all

Porém esta é uma configuração muito básica e não fornece nenhum tipo de segurança a rede. O que está definido nessa configuração é a porta onde o servidor *Proxy* irá atender as requisições (http_port 3128), o nome da máquina que será o servidor (visible_hostname nomeServidor), além de todos os endereços IP possíveis (acl all src 0.0.0.0/0.0.0.0), com acesso liberado (http_access allow all).

1. http_port 3128
2. visible_hostname nomeServidor
3. acl all src 0.0.0.0/0.0.0.0
4. acl bloqueado src 192.168.1.10
5. http_access deny bloqueado
6. http_access allow all

Um dos principais problemas em se configurar um servidor *Proxy* é que a ordem em que as regras aparecem pode interferir no funcionamento do mesmo, então é necessária uma atenção especial para que um erro na seqüência das regras não faça com o servidor funcione de maneira inadequada.

Para perceber melhor os efeitos que isso pode ocasionar é necessário observar o trecho de configuração anterior: caso a regra 6 apareça antes da regra 5, terá um http_access concedendo acesso sem restrições a todos os endereços IP possíveis, conseqüentemente a regra "http_access deny bloqueado" nem seria analisada, permitido acesso até mesmo ao IP 192.168.1.10 o qual queria bloquear. Esse é um dos problemas abordados no trabalho, e que será melhor discutido no Capítulo 3.

2.3.2 Servidor *DNS*

Servidores *DNS* (*Domain Name System*) oferecem um serviço de tradução de endereço IP em nomes e vice-versa, mantendo internamente uma tabela que relaciona o nome de cada máquina ao seu endereço IP.

Kurose (2006) destaca que os servidores *DNS* são divididos hierarquicamente em três classes, conforme Figura 2.

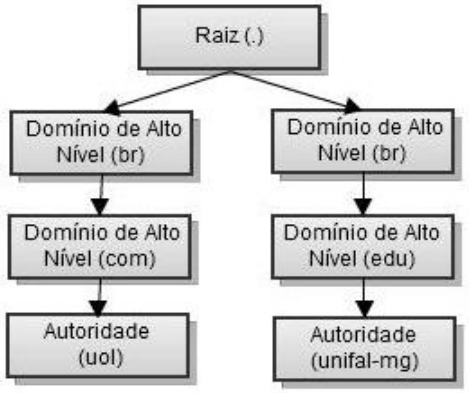

Figura 2 – Distribuição hierárquica dos servidores *DNS*

A primeira classe é composta por 13 servidores espalhados pelo mundo, formando os **servidores de nomes raiz**, cada um destes possui uma estrutura de replicação que faz com que cada servidor seja na verdade um conjunto de servidores. A segunda classe é formada pelos **servidores de nomes de Domínio de Alto Nível**, onde estão presentes os servidores que respondem pelos domínios com, org, net, entre outros

e os domínios dos países, que no caso do Brasil a listagem completa para os DPNs (Domínio de Primeiro Nível) podem ser encontradas no *site* da Fapesp[3]. Por último temos a classe dos **servidores de nomes com autoridade** que armazenam o registro das organizações, como por exemplo unifal-mg.

A resolução do nome em endereço IP é feita da direita para esquerda, percorrendo a Figura 2 do topo para a base. Vale salientar que todo endereço contém o caractere "." no final, mesmo que ele não seja digitado explicitamente o navegador se encarregará de adicioná-lo. Portanto no endereço **unifal-mg.edu.br** o servidor raiz lê o caractere "." e direciona a requisição para o servidor de nome de Domínio de Alto Nível que responde por br, esse por sua vez indica o próximo servidor que será o edu, que por último informa o endereço do servidor de nome com autoridade da unifal-mg.

A consulta anterior pode ser feita de forma iterativa e/ou recursiva. Além disso, o servidor possui um *cache* contendo as consultas realizadas anteriormente, o que agiliza consultas futuras.

Nas consultas iterativas como mostra a Figura 3 cada servidor consultado responde diretamente ao cliente informando o endereço de um servidor que seja hierarquicamente inferior a ele, até que o cliente consiga a resposta do endereço desejado (Kurose, 2006).

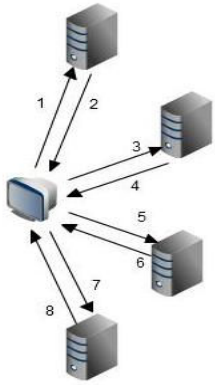

Figura 3 – Consulta iterativa

[3] https://registro.br/dominio/dpn.html site da Fapesp que é responsável pelos domínios no Brasil.

Quando a consulta é recursiva o cliente solicita a resolução do nome e fica aguardando apenas a resposta final, que na Figura 4 será no passo 8. Portanto cada servidor consultará o próximo servidor abaixo da sua hierarquia, até que todo o nome seja resolvido e as consultas comecem a retornar recursivamente (Kurose, 2006).

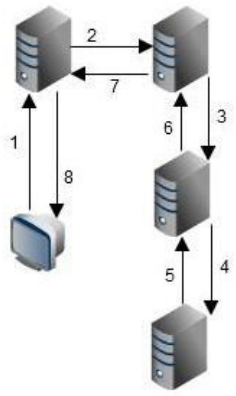

Figura 4 – Consulta recursiva

Além das duas consultas citadas existem as consultas inversas, que podem ser tanto iterativas quanto recursivas, o que as diferem é o modo como a pesquisa é realizada, pois o que se deseja encontrar é um nome através do endereço IP (Tibet, 2001).

Ferreira (2003) classificou os servidores *DNS* em três tipos básicos:

- Servidor *Cache*: apenas armazena o resultado de consultas feitas a ele.
- Servidor Mestre: contém todas as informações de um domínio especifico armazenadas em arquivos locais.
- Servidor Escravo: contêm uma cópia do banco de dados do Servidor Mestre de determinado domínio, pode responder tanto a requisições em caso de indisponibilidade do Servidor Mestre quanto para oferecer melhor distribuição de carga entre os servidores.

A configuração ideal para um bom desempenho do servidor *DNS*, é o conjunto de *DNS* primário e secundário. O *DNS* primário contém os arquivos originais de zona e

responderá por todas as resoluções de determinado domínio, qualquer alteração no serviço será feita nesse servidor. Embora o serviço de resolução de nomes possa funcionar apenas com o *DNS* primário, é essencial configurar o *DNS* secundário, pois ele armazenará uma cópia sempre atualizada da zona do DNS primário, e responderá pelo domínio em caso de falha do primário (Tibet, 2001).

Para garantir que outros servidores *DNS* validem o endereço do servidor de determinado domínio é necessário que se tenha o *DNS* reverso configurado. Assim quando um servidor receber uma requisição verificará se existe uma zona reversa que responda por aquele *IP* (Morimoto, 2008).

2.3.3 Servidor *DHCP*

Servidor *DHCP* (*Dynamic Host Configuration Protocol*) basicamente fornece endereços IP dinamicamente na rede, verificando qual endereço está disponível no momento da requisição, repassando-o a estação solicitante, automatizando assim o processo de configuração de endereços *IP*.

O funcionamento do servidor *DHCP* é bastante simples. Quando uma nova máquina se conecta a rede esta envia uma mensagem em *broadcast,* chamada DHCPDISCOVERY, contendo seu endereço *MAC*, o servidor ao receber esta mensagem verifica em uma tabela previamente cadastrada qual o endereço IP disponível para aquele momento, e responde uma mensagem chamada DHCPPOFFER fornecendo o endereço IP e os demais dados da rede para o endereço *MAC* da máquina solicitante, e mantém o endereço reservado para esta, a troca dessas mensagens podem ser observadas na Figura 5 (Tibet, 2001).

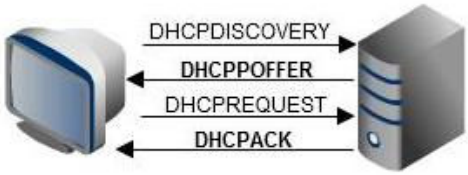

Figura 5 – Troca de mensagens entre servidor *DHCP* e cliente

Se houver vários servidores *DHCP* na rede e todos responderem a requisição, apenas a primeira mensagem DHCPPOFFER que chegar ao cliente é que será respondida, assim o cliente envia uma mensagem DHCPREQUEST, contendo a requisição oficial dos dados, direcionada ao servidor de quem recebeu a primeira resposta. A Figura 5 mostra que o servidor após receber a mensagem DHCPREQUEST do cliente, envia uma mensagem de confirmação chamada DHCPPACK, fornecendo o empréstimo do endereço IP (Tibet, 2001).

O servidor *DHCP* ficará periodicamente verificando se o cliente para o qual determinado endereço foi emprestado ainda está ativo, exigindo que o cliente reenvie uma mensagem DHCPREQUEST solicitando a renovação do empréstimo do endereço, com isso consegue-se garantir que os endereços *IP* sejam gastos somente com quem realmente está *online*, evitando esgotamento de endereços disponíveis (Morimoto, 2008).

3
Configuração Manual e *Scripts* Desenvolvidos

3.1 Introdução

Neste capítulo são apresentadas as configurações comumente feitas nos servidores escolhidos como foco nesse trabalho (*Proxy*, *DNS* e *DHCP*), e em seguida a cada explicação, são apresentadas as formas de implementação das funções em *Shell Script* desenvolvidas.

É proposta uma topologia exemplo para a qual as configurações foram realizadas. Ao se analisar as seções seguintes nesse capítulo, deve-se ter em mente então a rede hipotética exibida na Figura 6, que apresenta um servidor que oferecerá os serviços *Proxy*, *DNS* e *DHCP*, como uma placa de rede eth1 ligada na Internet e uma placa de rede eth0 ligada em um switch para distribuir a conexão para dez maquinas. .

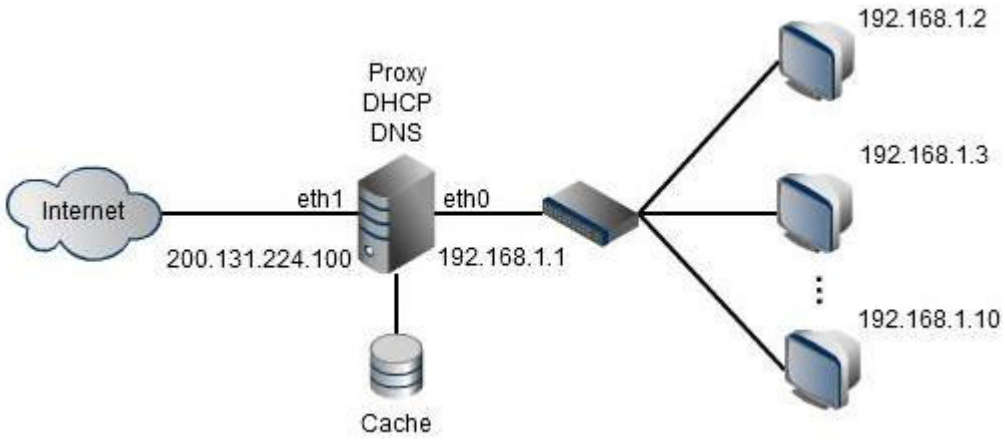

Figura 6 – Topologia da Rede Proposta

3.2 Squid

3.2.1 Manual

O servidor *Proxy* foi configurado através do Squid, onde todas as configurações são feitas em um único arquivo, o squid.conf. Como foi exposto na seção 2.3.1 deste trabalho, o *Proxy* pode funcionar tanto como um *cache* de páginas, como também oferecer uma infinidade de configurações para controle de acesso a Internet.

Armazenar páginas acessadas é muito básico e não traz segurança a rede, por esse motivo, este tipo de configuração não foi tratado pela ferramenta. Para tanto foi preferível trabalhar com o Proxy transparente e com autenticação, que apresentam o *cache* embutido nas suas configurações.

No Proxy transparente existe duas configurações distintas, uma para versões anteriores ao Squid 2.6, que devem conter no final do arquivo squid.conf as seguintes configurações:

1. http_accel_host virtual
2. http_accel_port 80
3. http_accel_with_proxy on
4. http_accel_use_host_header on

E outra para versões iguais ou superiores a 2.6 que tiveram as configurações anteriores trocadas por um simples "http_port 3128 transparent" adicionado no inicio do arquivo squid.conf . Em ambos os casos, o redirecionamento da porta 80 para a porta do *Proxy* deve ser feito através do comando **iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 80 -j REDIRECT --to-port 3128**, onde o 3128 deve ser trocado pela porta onde o *Proxy* responderá e o eth0 pela interface da placa de rede.

Para configurar o *Proxy* com autenticação é necessário criar um arquivo onde serão inseridos os *logins* e senhas dos usuários. Após criar o arquivo deve executar o

comando “htpasswd” que será responsável por criar a conta de acesso do usuário e transferi-la para o arquivo. Esse comando verificará se o *login* criado existe no arquivo, em caso afirmativo terá que ser definido uma nova senha para o usuário. Já no arquivo squid.conf deve constar as seguintes informações.

1. auth_param basic program /usr/lib/squid/ncsa_auth /etc/squid/squid_passwd
2. acl autenticados proxy_auth REQUIRED
3. http_access allow autenticados

A primeira linha indica a pasta que contém a biblioteca de autenticação, e o /etc/squid/squid_passwd indica o endereço e nome (squid_passwd) do arquivo que armazena os *logins* e senhas criados anteriormente. A segunda linha cria a acl autenticados do tipo proxy_auth indicando que os usuários que possuírem autenticação pertencem a acl. E por fim o http_access concede acesso a acl criada.

O trecho de configuração a seguir é responsável pelo *cache* do *Proxy* e terá o mesmo formato independente se o *Proxy* for transparente ou com autenticação.

1. cache_mem 64 MB
2. maximum_object_size_in_memory 64 KB
3. maximum_object_size 512 MB
4. minimum_object_size 0 KB
5. cache_swap_low 90
6. cache_swap_high 95
7. cache_dir ufs /var/spool/squid 2048 16 256
8. cache_access_log /var/log/squid/access.log
9. refresh_pattern ^ftp: 15 20% 2280
10. refresh_pattern ^gopher: 15 20% 2280
11. refresh_pattern . 15 20% 2280

As linhas 1 e 2 estão relacionadas a memória RAM, indicando a quantidade de memória que será reservada ao *cache* e o tamanho máximo dos objetos que serão armazenados na mesma. As 3 e 4 configura o tamanho máximo e mínimo dos arquivos armazenados no disco.

Para que arquivos antigos cedam espaço em disco, é necessário definir a porcentagem máxima de esgotamento que um *cache* em disco deve atingir, por exemplo, a linha 6 indica que quando o *cache* atingir 95% de sua capacidade, arquivos antigos começarão a ser descartados, e a linha 5 indica que esse descarte se dará até que a capacidade de armazenamento volte para 90% (Morimoto, 2008).

Após configurar os espaços necessários para se manter o *cache*, é preciso informar onde os arquivos serão salvos, para isso deve-se seguir o padrão da linha 7. Nesse exemplo indica que a pasta de *cache* estará no caminho /var/spool/squid, terá o tamanho de 2048 MB e será divida em 16 pastas, cada uma contendo 256 subpastas Morimoto, 2008).

Se o administrador do sistema desejar manter um log de acesso do *Proxy*, basta executar a configuração conforme indica a linha 8, que mostra que as informações de log serão armazenadas no arquivo access.log dentro da pasta indicada pelo caminho /var/log/squid/ (Morimoto, 2008).

O *Proxy* apresenta um sistema para manter os arquivos atualizados no *cache*, conforme foi discutido na seção 2.3.1, e as configurações quanto ao tempo de verificação de atualizações são feitas através das linhas 9,10 e 11. Essas três linhas devem estar juntas no arquivo squid.conf e indicam que a cada acesso se houver um arquivo com mais de 15 minutos no *cache* a atualização deste deverá ser verificada, e mesmo se um arquivo não for acessado todos deverão ser verificados em um tempo pré-estabelecido, que no caso será de 2280 minutos (Morimoto, 2008)..

Para evitar problemas com acesso de portas não permitidas, o Squid oferece a configuração do bloqueio ou acesso a portas, isso pode ser feito utilizando mais uma vez as diretivas acl, http_access,

As acl's podem ser definas da seguinte maneira "acl Safe_ports port" seguida das portas que se deseja controlar o acesso. O ideal é que portas que rodem serviços ligados a Internet sejam liberadas e as demais bloqueadas, nesse exemplo assume-se que as portas dos seguintes serviços precisem ser liberadas (Morimoto,2008) : 80 - http, 21, 22 – ftp, 443, 563 – https, 70 – gopher, 210 – wais, 280 – http-mgmt, 488 – gss-http, 591 – filemaker, 777 – multiling http, 901 – swat, 1025-65535 – portas altas. Após criar as acl's dessas portas, cria-se um htp_access negando acesso as demais, para isso basta utilizar o caractere de negação '!', assim a configuração ficará http_access deny !Safe_ports

1. acl Safe_ports port 80
2. acl Safe_ports port 21
3. acl Safe_ports port 443 563
4. acl Safe_ports port 70
5. acl Safe_ports port 210
6. acl Safe_ports port 1025-65535
7. acl Safe_ports port 280
8. acl Safe_ports port 488
9. acl Safe_ports port 591
10. acl Safe_ports port 777
11. acl Safe_ports port 901
12. http_access deny !Safe_ports

Para encerrar deve-se criar as regras de acesso para a rede local e *localhost*, ambas são no formato acl NOMEACL src IP, onde o IP deve ser formado pelo endereço e a respectiva máscara. As linhas de 1 a 5 permitem acesso ao *localhost* (próprio servidor) e aos computadores da rede local (192.168.1.0/24). A linha 6 nega o acesso a todos os demais computadores que tentem acesso, mas não se enquadrem nos padrões estabelecidos nas linhas anteriores.

1. acl all src 0.0.0.0/0.0.0.0

2. acl localhost src 127.0.0.1/255.255.255.255
3. acl redelocal src 192.168.1.0/24
4. http_access allow localhost
5. http_access allow redelocal
6. http_access deny all

3.2.2 Script

Nessa seção são apresentados os *scripts* desenvolvidos para a ferramenta e em que parte da configuração eles agem. O *script* do *Proxy* chama-se configurarSquid. Com o objetivo de melhorar a visualização do código, as funções tem as primeiras letras que identificam sua funcionalidade em maiúsculo, as variáveis todas escritas em maiúsculo, e as palavras reservadas e funções do próprio *Shell Script* em minúsculo.

Na configuração do *Proxy* transparente o primeiro passo é verificar a versão do Squid, isso é feito pela função VerificarVersoes que captura a versão atual do Squid através da função squid –v, e trabalha o resultado para retornar apenas o valor numérico.

```
1. VerificarVersoes(){
2.         COMANDO=$(squid -v)
3.         APENASVERSAO=$(echo $COMANDO | cut -d " " -f 4)
4.         VERSAO=$(echo $APENASVERSAO | cut -d S -f 1)
5.         TAMSTRING=$(echo $VERSAO | wc -c)
6.         TAMSTRING=$(($TAMSTRING-2))
7.         TESTVERSAO=$(expr substr $VERSAO 1 $TAMSTRING)
8. }
```

Existem duas funções no Proxy transparente para determinar a porta onde o *Proxy* responderá. PortaLocalTransparente para versões do Squid iguais ou superiores à 2.6, e PortaLocal para as demais. Inclusive a função PortaLocal é a mesma utilizada

pelo *Proxy* com autenticação. O conteúdo destacado em vermelho representa o que mudará de uma função para outra.

```
1.  PortaLocalTransparente() {
2.          echo
3.          echo "Defina a porta onde o servidor Squid irá rodar"
4.          echo "Para porta padrao 3128 tecle Enter"
5.          read -p ">> " PORT
6.          if [ "$PORT" = " ]; then
7.                  PORT=3128
8.          else until grep -E '^[0-9]+$' <<< $PORT
9.              do
10.                     read -p "Informe apenas numeros: " PORT
11.             done
12.         fi
13.         echo http_port $PORT transparent > /etc/squid/squid.conf
14.         NOMESERV=$(hostname)
15.         echo visible_hostname $NOMESERV >> /etc/squid/squid.conf
16. }
```

Por uma questão de ordem nas regras do *Proxy* com autenticação, onde os sites e palavras bloqueadas devem aparecer antes da regra de autenticação, as funções BloquearSite e BloquearPalavras devem ser executadas logo após a execução da função que define a porta do *Proxy*. Caso essa ordem não seja obedecida, o bloqueio não funcionará. Esse é um erro que possivelmente seria cometido ao se fazer a configuração manual.

```
1.  BloquearSite() {
2.  echo
3.  echo "Defina os sites que terão o acesso bloqueado"
4.  echo "Para parar de incluir sites digite ."
5.  read -p ">> " SITE
6.  while [ $SITE != "." ];
```

```
7.  do
8.  echo $SITE >> /etc/squid/sitesbloqueados
9.  read -p ">> " SITE
10. done
11. echo acl bloqueados url_regex -i \"/etc/squid/sitesbloqueados\" >> /etc/squid/squid.conf
12. echo http_access deny bloqueados >> /etc/squid/squid.conf
13. }
1.  BloquearSite() {
2.          echo
3.          echo "Defina os sites que terão o acesso bloqueado"
4.          echo "Para parar de incluir sites digite ."
5.          read -p ">> " SITE
6.          while [ $SITE != "." ];
7.          do
8.                  echo $SITE >> /etc/squid/sitesbloqueados
9.                  read -p ">> " SITE
10.         done
11.         echo acl bloqueados url_regex -i \"/etc/squid/sitesbloqueados\" >> /etc/squid/squid.conf
12.         echo http_access deny bloqueados >> /etc/squid/squid.conf
13. }
```

```
1.  BloquearPalavras() {
2.          echo "Defina as palavras que terão o acesso bloqueado"
3.          echo "Para parar de incluir sites digite ."
4.          read -p ">> " PALAVRAS
5.          while [ $PALAVRAS != "." ];
6.          do
7.                  echo $PALAVRAS >> /etc/squid/palavrasbloqueadas
8.                  read -p ">> " PALAVRAS
9.          done
10.         echo acl nomesproibidos dstdom_regex \"/etc/squid/palavrasbloqueadas\" >>
    /etc/squid/squid.conf
11.         echo http_access deny nomesproibidos >> /etc/squid/squid.conf
12. }
```

No *Proxy* com autenticação, após cadastrar os sites e palavras bloqueadas é preciso executar a função CadastrarLogin, que criará um arquivo que conterá o *login* e senha de cada usuário, e adicionará as regras no arquivo squid.conf. Nessa função é feita uma verificação se o usuário digitado está cadastrado, oferecendo a opção de redefinir a senha caso o usuário já conste no arquivo squid_passwd.

```
1.  CadastrarLogin() {
2.          echo
3.          echo "--------------------CADASTRO DE LOGINS--------------------"
4.          touch /etc/squid/squid_passwd
5.          echo
6.          echo "Defina o login do usuario"
7.          echo "Para sair digite ."
8.          read -p ">> " USUARIO
9.          while [ $USUARIO != "." ];
10.         do
11.                 RESP="s"
12.                 while read pal
13.                 do
14.                         CADASTRADO=$(echo $pal | cut -d : -f 1)
15.                         if [ "$USUARIO" = "$CADASTRADO" ]; then
16.                                 IGUAL=1
17.                         fi
18.                 done < /etc/squid/squid_passwd
19.                 if [ $IGUAL = 1 ] ; then
20.                         echo
21.                         echo "Usuario $USUARIO ja cadastrado, deseja redefinir senha s/n?"
22.                         read -p ">> " RESP
23.                 fi
24.                 if [ "$RESP" = "s" ]; then
25.                          htpasswd /etc/squid/squid_passwd $USUARIO
26.                 fi
27.                 echo
28.                 echo "Defina outro usuario ou para sair tecle ."
29.                 read -p ">> " USUARIO
```

```
30.         done
31.         echo auth_param basic program /usr/lib/squid/ncsa_auth /etc/squid/squid_passwd >>
    /etc/squid/squid.conf
32.         echo acl autenticados proxy_auth REQUIRED >> /etc/squid/squid.conf
33.         echo http_access allow autenticados >> /etc/squid/squid.conf
34. }
```

Após a configuração do tipo de serviço *Proxy* vem as regras de objetivo geral, que são realizadas pelas funções ConfigurarCache e LiberarPortas. ConfigurarCache apresenta as opções que guiará o usuário na definição dos parâmetros descritos na seção 3.2.1 sobre o *cache* de arquivos. E a função LiberarPortas define as regras de acesso de portas, rede local, e *localhost*.

3.3 Bind

3.3.1 Manual

As configurações das zonas do servidor *DNS* no *Bind* são feitas no arquivo named.conf.local, e para as configurações do domínio deve-se criar um arquivo separado, em geral esse arquivo é iniciado com db seguido do nome do site, por exemplo db.lares.unifal-mg.edu.br.

A configuração da zona será feita utilizando apenas um servidor, embora esse tipo de configuração não seja aconselhável, uma vez que em caso de queda do servidor o domínio ao qual ele responde ficará fora do ar. Para a configuração com apenas o servidor primário o arquivo named.conf.local terá uma zona como mostrado a seguir:

```
1. zone "lares.unifal-mg.edu.br" {
2. type master;
3. file "/etc/bind/db.lares.unifal-mg.edu.br";
4. };
```

O bloco é iniciado com a palavra reservada zone seguida do nome do site, todas as instruções da zona devem estar contidas dentro de {}. O type master quer dizer que esse é um servidor do tipo mestre (primário), e *file* "/etc/bind/db.lares.unifal-mg.edu.br", define o caminho onde está contido o arquivo de configuração db.lares.unifal-mg.

No arquivo named.conf.local é feita a configuração apenas da zona, para as configurações do domínio deve-se criar um arquivo, com o nome e no local indicado no campo file da zona, pois será este campo que fará a ligação entre as configurações. A seqüência de configuração enumerada a seguir apresenta um arquivo criado para o domínio lares.unifal-mg.edu.br.

1. @ IN SOA lares.unifal-mg.edu.br. root.lares.unifal-mg.edu.br. (
2. 2011060501 ; Serial
3. 8h; Refresh
4. 2h; Retry
5. 7d; Expire
6. 1d) ; Negative Cache TTL
7. @ IN NS lares.unifal-mg.edu.br
8. ns IN A 192.168.1.1
9. www IN A 192.168.1.1
10. ftp IN A 192.168.1.1

A configuração desse arquivo requer uma atenção especial, pois deve existir uma identação para que o *Bind* possa executá-lo sem erros. No exemplo apresentado essa identação foi feita com espaços simples, mas pode ser utilizado tabs.

A linha 1 apresenta o nome do *site* e o *e-mail* do administrador do domínio. Tudo o que está contido entre os parentes refere-se as configurações necessárias para manter o servidor secundário atualizado, o arquivo deve conter essas configurações mesmo que não exista um servidor secundário.

É importante destacar cada parte do arquivo responsável por manter a atualização no servidor secundário. Primeiramente temos o Serial, este número por padrão tem o formato AAAAMMDDSS e representa a data da última modificação feita

na configuração do servidor, por exemplo na linha 2 do arquivo o serial 2011060501 indica que o arquivo foi alterado pela ultima vez no dia 31 de maio de 2011, e a ultima seqüência SS que no exemplo está 01, significa que essa foi a primeira modificação naquele dia.

Em seguida temos o *Refresh*, nesse campo será definido o tempo que o servidor secundário aguardará para verificar uma nova atualização, caso não consiga contato com o servidor primário o secundário entenderá que o serviço está fora do ar e tentará responder pelo domínio, ou seja a cada 8 horas um nova verificação será feita. Se o servidor secundário não conseguir assumir o lugar do primário, novas tentativas serão feitas a cada 2 horas conforme definido na linha 3 em *Retry.*

O servidor secundário não poderá ficar respondendo indefinidamente pelo domínio, por isso na linha 5 (*Expire*) é definido o tempo máximo que isso pode ser feito, o ideal para configurar esse campo é estabelecer um tempo suficiente para o reparo do servidor primário, por esse motivo na linha 5 o valor 7d, que representa sete dias. A linha 6 (*Negative Cache TTL)* define o tempo mínimo que o servidor secundário ficará respondendo pelo domínio.

As linhas 7 e 8 indicam os dados do servidor responsável pelo domínio. E as linhas 8 e 9 definem os serviços pelo qual o domínio responderá. Na topologia proposta não foi indicado a utilização do servidor secundário, porém para análise é importante mostrar como ficaria essa configuração, o que será apresentado na sequência.

```
1. zone "lares.unifal-mg.edu.br" {
2. type master;
3. file "/etc/bind/db.lares.unifal-mg.edu.br";
4.  allow-transfer { 192.168.1.12; };
5. };
```

A configuração do *DNS* com servidor secundário inclui algumas configurações adicionais, a primeira alteração a ser feita será na configuração da zona que terá “allow-

transfer { 192.168.1.12; };” adicionado, isso indica o endereço *IP* do servidor secundário que manterá uma cópia atualizada das zonas do servidor primário.

```
1. zone "lares.unifal-mg.edu.br" {
2. type slave;
3. file "/etc/bind/db.lares.unifal-mg.edu.br";
4. master { 192.168.1.1; };
5. };
```

Além disso deverá ser criada uma zona específica para o servidor secundário, conforme mostrado na configuração anterior. Nessa zona o tipo do servidor será *slave* e deve ser indicado o endereço *IP* do servidor primário, essas configurações estão destacadas em vermelho.

Existe ainda a possibilidade de configurar o *DNS* reverso, e nessa configuração independe se existe ou não um servidor secundário. Do mesmo modo das configurações anteriores deverá ser criado uma configuração para a zona e outra especifica para tratar do domínio reverso.

```
1. zone "1.168.192.in-addr.arpa" {
2. type master;
3. file "/etc/bind/db.lares.unifal-mg.edu.br.rev";
4. };
```

Analisando o conteúdo anterior nota-se que as únicas coisas que modificaram foram as linhas que indicam o nome da zona e o nome do arquivo de configuração do domínio. O nome dessa zona será formado pelo endereço IP do servidor primário escrito de trás para frente, desconsiderando o último octeto do IP. No exemplo o IP do servidor primário era 192.168.1.1, e o nome da zona ficou 1.168.192.in-addr-arpa, onde “in-addr-arpa” é uma palavra reservada do Bind. Se existir servidor secundário acrescentar o allow-transfer, assim como foi descrito anteriormente para a zona do servidor primário.

```
1. @ IN SOA lares.unifal-mg.edu.br. root.lares.unifal-mg.edu.br. (
```

2. 2011060501 ; Serial
3. 8h ; Refresh
4. 2h ; Retry
5. 7d ; Expire
6. 1d) ; Negative Cache TTL
7. IN NS ns1.lares.unifal-mg.edu.br

Do mesmo modo de um servidor primário, deverá ser criado um arquivo de configuração do domínio para o *DNS* reverso, e seu conteúdo será quase idêntico ao descrito anteriormente, a principal modificação como se pode observar será na ultima linha. O 1 representa o último octeto do *IP* que foi omitido durante a configuração da zona. Se existir um servidor secundário basta repetir a linha descrita colocando no lugar do 1 a última seqüência do *IP* do servidor secundário.

3.3.2 Script

Para configuração do *Bind*, foi criado o script configurarBind, e segue os mesmo padrões descritos para a configuração do *Squid*.

Na configuração da zona constando apenas o servidor primário utiliza-se a função ConfigurarZona, o único parâmetro que o usuário terá que informar é o nome do *site*, a função configurará o restante das informações.

```
1.  ConfigurarZona Primario (){
2.          echo
3.          echo ".....CONFIGURANDO A ZONA....."
4.          echo
5.          echo "Defina o nome do seu site para configurar a nova zona (exemplo: dominio.com)"
6.          read NOMESITE
7.          echo " " >> /etc/bind/named.conf.local
8.          echo "zone "\"$NOMESITE"\" {" >> /etc/bind/named.conf.local
9.          echo "     type master;" >> /etc/bind/named.conf.local
```

```
10.         echo "     file "\"/etc/bind/db.$NOMESITE"\";" >> /etc/bind/named.conf.local
11.         echo "};" >> /etc/bind/named.conf.local
12.         echo "Defina o IP do servidor secundario"
13.         read IPSECUNDARIO
14.         echo "     allow-transfer { $IPSECUNDARIO; };" >> /etc/bind/named.conf.local
15.         echo "};" >> /etc/bind/named.conf.local
16.  }
```

Quando o *DNS* possuir um servidor secundário as funções ConfigurarZonaPrimario e ConfigurarZonaSecundario deverão ser executadas. A função ConfigurarZonaPrimario terá a mesma estrutura da ConfigurarZona, o que a difere são as linhas de 12 a 14, pois o usuário terá que informar o IP do servidor secundário.

A função ConfigurarZonaSecundario terá o formato apresentado a seguir, seus diferenciais são que a configuração que define o tipo da zona passa a ser *slave*, e o *IP* do servidor primário deverá ser informado, garantindo assim o *link* entre as zonas do primário e secundário.

```
1.  ConfigurarZonaSecundario(){
2.          echo
3.          echo ".....CONFIGURANDO A ZONA NO SERVIDOR SECUNDARIO....."
4.          echo
5.          echo "Digite o nome do site definido no servidor primario para configurar a nova zona"
6.          read NOMESITE
7.          echo " " >> /etc/bind/named.conf.local
8.          echo "zone "\"$NOMESITE"\" {" >> /etc/bind/named.conf.local
9.          echo "     type slave;" >> /etc/bind/named.conf.local
10.         echo "     file "\"/etc/bind/db.$NOMESITE"\";" >> /etc/bind/named.conf.local
11.         echo
12.         echo "Defina o IP do servidor primario"
13.         read IPPRIMARIO
14.         echo "     allow-transfer { $IPPRIMARIO; };" >> /etc/bind/named.conf.local
15.         echo "};" >> /etc/bind/named.conf.local
```

```
16. }
```

Após configurar as zonas é necessário configurar os arquivo de domínio, independente de possuir ou não o *DNS* secundário, toda a configuração será feita pela função configurarServidor.

```
1.  ConfigurarServidor(){
2.      echo
3.      echo ".....CONFIGURANDO O SERVIDOR....."
4.      echo
5.      echo "Defina o email do administrator do site, no lugar de @ utilize ponto"
6.      read ADMINISTRADOR
7.      TESTA=$(echo $ADMINISTRADOR | grep "@")
8.      while [ "$TESTA" != " ];
9.      do
10.         echo
11.         echo "O email nao pode conter @, digite novamente o email trocando o @ por ."
12.         echo "------------------------------------------------------------------------"
13.         echo
14.         echo "Defina o email do administrator do site, no lugar de @ utilize ponto"
15.         read ADMINISTRADOR
16.         TESTA=$(echo $ADMINISTRADOR | grep "@")
17.     done
18.     echo "@ IN SOA $NOMESITE. $ADMINISTRADOR. (" > /etc/bind/db.$NOMESITE
19.     echo
20.     echo "Defina um numero para ser usado como serial e manter o servidor secundario
    atualizado"
21.     echo "Para mais informacoes sobre o formato do serial digite a"
22.     read SERIAL
23.     if [ "$SERIAL" = "a" ]; then
24.         echo "Geralmente e utilizado como serial a data de modificacao deste arquivo no
    formato AAAAMMDDSS"
25.         echo "Exemplo: 2011011701 seria um servidor configurado em 17 de janeiro de
```

```
2011."
26.             echo "SS eh um numero sequencial que voce deve incrementar a cada
configuracao feita no dia."
27.             echo "Assim, no exemplo essa teria sido a primeira configuracao feita no dia."
28.             read SERIAL
29.         fi
30.         echo "$SERIAL ; Serial" >> /etc/bind/db.$NOMESITE
31.         echo
32.         echo "Defina em minutos o tempo de atualizacao do servidor DNS"
33.         read REFRESH
34.         echo "$REFRESH ; Refresh" >> /etc/bind/db.$NOMESITE
35.         echo
36.         echo "Defina em minutos o tempo "
37.         read RETRY
38.         echo "$RETRY ; Retry" >> /etc/bind/db.$NOMESITE
39.         echo
40.         echo "Defina em minutos o tempo para expirar"
41.         read EXPIRE
42.         echo "$EXPIRE ; Expire" >> /etc/bind/db.$NOMESITE
43.         echo
44.         echo "Defina em minutos o tempo de TTL"
45.         read NEGATIVECACHE
46.         echo "$NEGATIVECACHE) ; Negative Cache TTL" >> /etc/bind/db.$NOMESITE
47.         echo "@ IN NS ns1.$NOMESITE" >> /etc/bind/db.$NOMESITE
48.         if [ $SECUNDARIO -eq 1 ]; then
49.             echo "@ IN NS ns2$NOMESITE" >> /etc/bind/db.$NOMESITE
50.         fi
51.         echo "@ IN MX 10 smtp.$NOMESITE." >> /etc/bind/db.$NOMESITE
52.         echo
53.         echo "Defina o endereco IP do servidor primario"
54.         read IPPRIMARIO
55.         echo "ns IN A $IPPRIMARIO" >> /etc/bind/db.$NOMESITE
56.         if [ $SECUNDARIO -eq 1 ]; then
57.             echo
58.             echo "Defina o endereco IP do servidor secundario"
```

```
59.                 read IPSECUNDARIO
60.                 echo "ns IN A $IPSECUNDARIO" >> /etc/bind/db.$NOMESITE
61.         fi
62.         echo
63.         echo "Defina os servicos que o servidor ira rodar. Exemplo www, ftp..."
64.         echo "Para parar de inserir servicos digite ."
65.         read SERVICO
66.         while [ $SERVICO != "." ];
67.         do
68.                 echo "$SERVICO IN A $IPPRIMARIO" >> /etc/bind/db.$NOMESITE
69.                 read SERVICO
70.         done
71. }
```

Na função acima serão informados os dados que manterão o *DNS* secundário atualizado, o *e-mail* do administrador do domínio e os serviços que o servidor irá executar. Como na topologia em questão não existe o *DNS* secundário as linhas que correspondem a sua configuração serão omitidas do arquivo.

A configuração da zona reversa exige uma transformação no endereço *IP* do servidor primário para obter o *IP* reverso. Isso é feito com o comando IP1=$(echo $IPPRIMARIO | cut -d . -f 1), onde o número destacado em vermelho é substituído pela posição do octeto. Por exemplo, para o IP 192.168.1.1, ficaria IP1 = 192, IP2 = 168 e IP3 = 1, depois esses valores seriam atribuídos a variável IPREVERSE na ordem inversa com o caractere '.' separando-os. Mais uma vez será verificado a existência do *DNS* secundário para fazer as configurações referentes ao mesmo.

```
1.  ConfigurarZonaReversa(){
2.          echo
3.          echo ".....CONFIGURANDO A ZONA REVERSA....."
4.          IP1=$(echo $IPPRIMARIO | cut -d . -f 1)
5.          IP2=$(echo $ IPPRIMARIO | cut -d . -f 2)
6.          IP3=$(echo $ IPPRIMARIO | cut -d . -f 3)
```

```
7.          IPREVERSE=$IP3.$IP2.$IP1
8.          echo $IPREVERSE
9.          echo  " " >> /etc/bind/named.conf.local
10.         echo "zone "\"$IPREVERSE.in-addr.arpa"\" {" >> /etc/bind/named.conf.local
11.         echo "     type master;" >> /etc/bind/named.conf.local
12.         echo "      file "\"/etc/bind/db.$NOMESITE.rev"\";" >> /etc/bind/named.conf.local
13.         if [ $SECUNDARIO -eq 1 ]; then
14.                 echo "     allow-transfer { $IPSECUNDARIO; };" >> /etc/bind/named.conf.local
15.         fi
16.         echo "};" >> /etc/bind/named.conf.local
17. }
```

A configuração do domínio do servidor Reverso segue os mesmos princípios utilizados na configuração dos demais domínios. Sua principal diferença está em "$IP4 IN PTR www.$NOMESITE", onde a variável IP4 receberá o último octeto do *IP* primário.

```
1.  ConfigurarServidorReverso(){
2.          echo
3.          echo ".....CONFIGURANDO O SERVIDOR REVERSO....."
4.          echo "@ IN SOA $NOMESITE. $ADMINISTRADOR. (" >
    /etc/bind/db.$NOMESITE.rev
5.          echo "$SERIAL ; Serial" >> /etc/bind/db.$NOMESITE.rev
6.          echo "$REFRESH ; Refresh" >> /etc/bind/db.$NOMESITE.rev
7.          echo "$RETRY ; Retry" >> /etc/bind/db.$NOMESITE.rev
8.          echo "$EXPIRE ; Expire" >> /etc/bind/db.$NOMESITE.rev
9.          echo "$NEGATIVECACHE) ; Negative Cache TTL" >> /etc/bind/db.$NOMESITE.rev
10.         echo "  IN NS ns1.$NOMESITE" >> /etc/bind/db.$NOMESITE.rev
11.         if [ $SECUNDARIO -eq 1 ]; then
12.                 echo "  IN NS ns2.$NOMESITE" >> /etc/bind/db.$NOMESITE.rev
13.         fi
14.         IP4=$(echo $IPPRIMARIO | cut -d . -f 4)
15.         echo "$IP4 IN PTR www.$NOMESITE" >> /etc/bind/db.$NOMESITE.rev
16.         if [ $SECUNDARIO -eq 1 ]; then
```

```
17.                 IP4=$(echo $IPSECUNDARIO | cut -d . -f 4)
18.                 echo "$IP4 IN PTR www.$NOMESITE" >> /etc/bind/db.$NOMESITE.rev
19.         fi
20. }
```

3.4 DHCP3

3.4.1 Manual

Um servidor *DHCP* pode oferecer, tanto *IP* dinamicamente quanto *IP* fixo para a rede, para isso basta configurá-lo de acordo com as necessidades. As configurações do *DHCP* são feitas no arquivo dhcpd.conf.

A primeira configuração feita no *DHCP* será quanto ao tempo de uso dos *IP* fornecidos. Nesse ponto será definido com que freqüência o servidor verificará se as estações estão ativas e o tempo máximo que cada uma pode utilizar determinado endereço.

Para exemplificar é necessário analisar as duas linhas onde estas configurações são feitas:

1. default-lease-time 600
2. max-lease-time 7200

A primeira linha indica que a cada 600 segundos uma nova verificação será feita e se alguma estação estiver inativa seu endereço *IP* será disponibilizado para outra máquina se necessário. Na segunda linha independente da estação estar ou não ativa, ela terá seu endereço revogado depois de transcorridos 7200 segundos. Após configurar os tempos de permanência dos endereços IP's, se faz a configuração para a distribuição de IP dinâmico.

1. subnet 192.168.1.0 netmask 255.255.255.0 {
2. range 192.168.1.2 192.168.1.10;
3. option routers 192.168.1.1;
4. option domain-name-servers 192.168.1.1;
5. option broadcast-address 192.168.1.255 ;

Na linha 1 foi definida a sub-rede e a máscara da sub-rede. A sub-rede é formada por quatro octetos de bits, em que os bits mais significativos indicam o endereço da rede e os bits menos significativos indicam o endereço dos hosts, na configuração anterior isso significa que o endereço da rede será 192.168.0. Para evitar possíveis conflitos com a internet as rede locais existem faixas de *IP* reservados para a rede, esses endereços obedecem aos seguinte formatos: 10.x.x.x, 192.168.x.x, ou 172.16.x.x a 172.31.x.x.

Na máscara da sub-rede os octetos referente ao endereço da rede são substituídos por 255 e os demais que representaram os endereços dos hosts podem ser representados por valores entre 0 e 254 de acordo com a faixa de endereços que a sub-rede terá.

É necessário informar ao *DHCP* qual a faixa de endereço será destinado aos IP dinâmicos, e isso é feito na linha 2 utilizando a opção range, que no exemplo delimita os endereços entre 192.168.1.2 e 192.168.1.10.

O endereço da máquina que compartilha a conexão na rede deve ser informado na linha 3, que na configuração anterior é o mesmo endereço do servidor *DCHP*, mas isso não é uma regra.

O *DHCP* informa aos demais micros da rede os endereços dos servidores *DNS* que lhes fará a resolução de nomes, por isso a configuração option domain-name-servers armazena o endereço dos servidores *DNS*. Podem ser adicionados vários servidores *DNS*, bastando para isso que se insira todos os endereços separados por virgula e sem espaço.

Por último na configuração dos IP's dinâmicos deve ser informado o endereço de *broadcast* da rede, no caso da sub-rede configurada será 192.168.1.255.

De acordo com o que foi dito no inicio do tópico o *DHCP* pode oferecer endereços fixos para determinadas máquinas, para isso deve-se incluir no arquivo dhcpd.conf a configuração a seguir.

```
1. host maquina1 {
2. hardware ethernet 08:00:27:64:1f:83;
3. fixed-address 192.168.1.11;
4. }
```

Nessa configuração cada máquina deverá ter um nome associado, e isso será feito pela opção *host*, conforme mostrado na linha 1. Em seguida deve ser adicionado o endereço *MAC* da máquina, esse endereço pode ser visualizado através do comando ifconfig. Na linha 3 será atribuído realmente o endereço *IP* da máquina. É importante informar que os IP's fixos devem estar fora da faixa de *IP* definida para IP's dinâmicos, para evitar conflito entre IP's.

Em resumo, a configuração anterior informa que a máquina de endereço *MAC* 0B:11:4A:2C:20:45, será chamada de maquina1 e terá o endereço IP 192.168.1.11.

O usuário pode optar por definir *IP* fixo para todas as máquinas, deixar a distribuição dos *IP* por conta do *DHCP* através da configuração de IP's dinâmicos, ou trabalhar com as duas configurações em conjunto, qualquer uma dessas opções são validas.

3.4.2 Script

Os tempos de renovação e permanência de um *host* com determinado *IP* será definido pela função DefinirTempo, essa configuração é extremamente básica e será feita em apenas dois passos. Primeiro será definido em minutos o tempo de renovação, em seguida o usuário determinará, também em minutos, o tempo máximo que um *host* pode permanecer com determinado endereço.

```
1. DefinirTempo(){
```

```
2.      echo ddns-update-style none; > /etc/dhcp3/dhcpd.conf
3.      echo "Determine em segundos o tempo de renovacao dos enderecos IP"
4.      echo "Exemplo 600, assim a cada 600 minutos o servidor verificara se as estacoes estao
    ativas"
5.      read RENOVACAO
6.      echo "default-lease-time $RENOVACAO;" >> /etc/dhcp3/dhcpd.conf
7.      echo "Determine em segundos o tempo maximo que uma estacao pode usar determinado
    IP"
8.      echo "Exemplo 7200, significa que a estacao so pode usar o endereco IP por 7200
    segundos"
9.      read USOMAXIMO
10.     echo "max-lease-time $USOMAXIMO;" >> /etc/dhcp3/dhcpd.conf
11.     echo "authoritative;" >> /etc/dhcp3/dhcpd.conf
12. }
```

Na função ConfigurarRede será feito a configuração da distribuição de endereços *IP* dinamicamente. Será neste momento que o usuário informará os dados relativos a sub-rede, a faixa de *IP* que serão disponibilizados, o *IP* do servidor ou servidores *DNS*, *IP* da máquina que compartilha a conexão, e o endereço de *broadcast*.

```
1.  ConfigurarRede(){
2.      echo
3.      echo "Determine o endereço da subrede. Exemplo 192.168.1.0"
4.      read SUBREDE
5.      echo
6.      echo "Determine a mascara da subrede. Exemplo 255.255.255.0"
7.       read MASCARA
8.      echo "subnet $SUBREDE netmask $MASCARA {" >> /etc/dhcp3/dhcpd.conf
9.      echo
10.     echo "Determine o inicio da faixa de enderecos IP que sera usado pelo servidor"
11.     echo "Exemplo 192.168.1.1"
12.     read FAIXAINICIO
```

```
13.         echo
14.         echo "Determine o final da faixa de enderecos IP que sera usado pelo servidor"
15.         echo "Exemplo 192.168.1.100"
16.         read FAIXAFINAL
17.         echo "range $FAIXAINICIO $FAIXAFINAL;" >> /etc/dhcp3/dhcpd.conf
18.         echo
19.         echo "Determine o endereco IP do servidor que esta compartilhando a conexao"
20.         read SERVCONEXAO
21.         echo "option routers $SERVCONEXAO;" >> /etc/dhcp3/dhcpd.conf
22.         echo
23.         echo "Determine o endereco IP do servidor DNS que sera usado pelas estacoes"
24.         echo "Se existir mais de um servidor DNS, digitar os IP separados por virgula e sem
    espaco"
25.         echo "Exemplo 200.177.250.10,200.204.0.10"
26.         read SERVDNS
27.         echo "option domain-name-servers $SERVDNS;" >> /etc/dhcp3/dhcpd.conf
28.         echo
29.         echo "Determine o endereco IP de broadcast"
30.         read BROAD
31.         echo "option broadcast-address $BROAD;" >> /etc/dhcp3/dhcpd.conf
32.         echo "}" >> /etc/dhcp3/dhcpd.conf
33. }
```

Ao optar por definir hosts com *IP* fixo a função ConfigurarHost será executada. Como já foi explicado na seção Manual deste tópico, para se ter o *IP* fixo o usuário terá que fornecer um nome para a máquina, o endereço *MAC*, e o endereço *IP* que deseja que aquela estação receba.

```
1.  ConfigurarHost(){
2.          echo ".....CONFIGURANDO IP FIXOS....."
3.          echo "Determine o nome da maquina"
4.          read NOME
```

```
5.      echo "host $NOME {" >> /etc/dhcp3/dhcpd.conf
6.      echo
7.      echo "Determine o endereco Mac da maquina. Exemplo 0B:11:4A:2C:20:45"
8.      read MAC
9.      echo "hardware ethernet $MAC;" >> /etc/dhcp3/dhcpd.conf
10.     echo "Determine o IP fixo para maquina"
11.     read IPFIXO
12.     echo fixed-address $IPFIXO; >> /etc/dhcp3/dhcpd.conf
13.     echo "}" >> /etc/dhcp3/dhcp.conf
14. }
```

4
Análise de Resultados

4.1 Introdução

Neste capítulo são exibidas as execuções dos principais passos para cada script criado, bem como os resultados finais mostrando que os serviços foram configurados corretamente após a execução dos mesmos. A Topologia proposta a ser configurada é a mesma já exibida na Figura 6 no Capítulo 3. As seções 4.1, 4.2 e 4.3 apresentam respectivamente as configurações do *Proxy*, *DNS* e *DHCP*. A Figura 7 apresenta o menu inicial da ferramenta.

Figura 7 – Menu inicial da ferramenta

Fonte: Dados da Pesquisa

A ferramenta não possui interface gráfica, assim toda a execução do programa deve ser feita via linha de comando no *Shell* do Linux. Antes de iniciar a utilização do programa é necessário que o usuário esteja com permissão de *root*, para que os *scripts* criados consigam alterar os arquivos de configuração em cada servidor.

Para o perfeito funcionamento durante a configuração do serviço é importante que os scripts configurarSquid, configurarBind, configurarDHCP e virtserver estejam na mesma pasta.

O virtserver é o script responsável por fazer a chamada das demais funções, que redirecionará o usuário para o serviço que se deseja configurar. Após garantir que todos os arquivos estejam juntos no mesmo diretório, basta executar no *Shell* do Linux o comando **"./virtserver"**, lembrando-se que este comando precisa ser dado dentro da pasta onde estão salvos os *scripts*.

Toda a configuração com a ferramenta é feita através de perguntas, que oferecem uma breve explicação do que está sendo configurado, dando ainda opções padrão para agilizar o trabalho do administrador.

4.2 Squid

Para a topologia proposta foi configurado um *Proxy* com autenticação, visando facilitar a análise da autenticação por *login* e senha, juntamente com o bloqueio por *sites* e palavras.

A Figura 8 apresenta o menu inicial da configuração do *Proxy*. Neste, foi selecionado a opção 2 para prosseguir com a configuração do *Proxy* com autenticação.

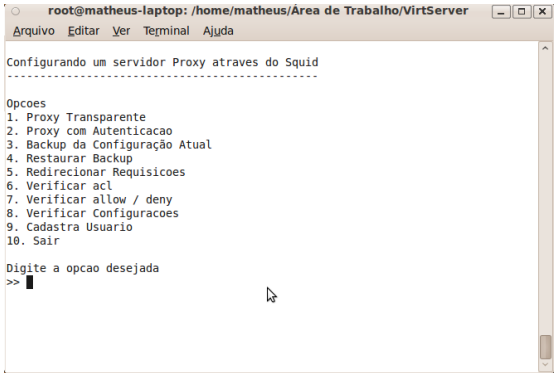

Figura 8 – Menu inicial de configuração do *Proxy*

Fonte: Dados da Pesquisa

Assim como aconteceria com a configuração do *Proxy* transparente a primeira fase de configuração foi da porta do *Squid*, e o cadastro dos sites e palavras que se deseja bloquear. Para análise foi bloqueado o site exemplo1.com.br e qualquer URL que tenha em seu conteúdo a palavra exemplo2. As opções de bloqueio podem ser observadas na Figura 9.

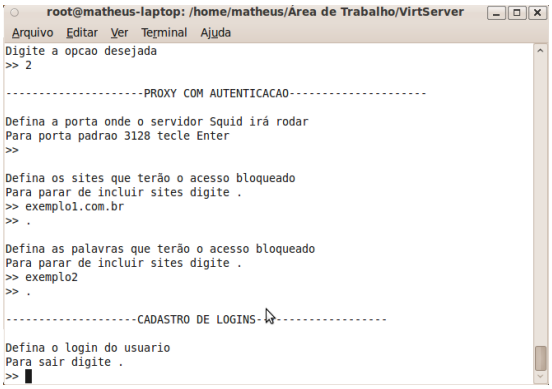

Figura 9 – Definição de porta e bloqueio de sites e palavras

Fonte: Dados da Pesquisa

Na Figura 10 pode-se observar o bloqueio foi feito pelo *Proxy*, pois a mensagem exibida é de acesso negado. Além disso, no rodapé da mensagem aparece o nome do servidor matheus-laptop e a versão 2.7 do Squid.

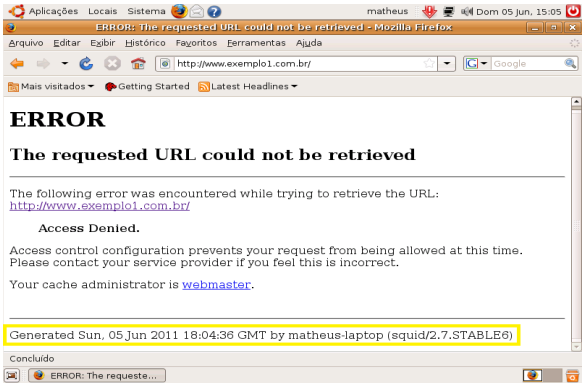

Figura 10 – Tela do *browser* informando o bloqueio do *site*

Fonte: Dados da Pesquisa

O cadastro de usuários pode ser observado na Figura 11. Nesse exemplo foi feita uma tentativa de se adicionar um usuário já cadastrado. Nesse caso, o sistema detectou se tratar de um usuário cadastrado, e perguntou pela possibilidade de redefinir a senha.

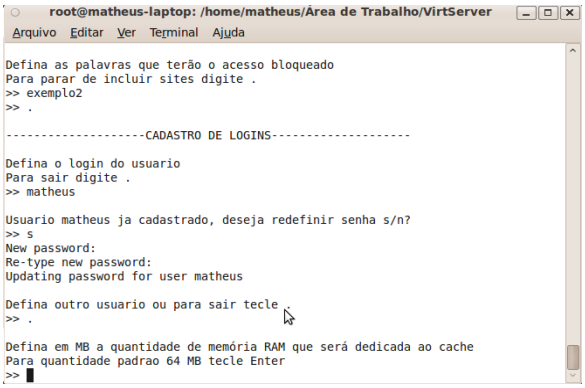

Figura 11 – Cadastro de usuário
Fonte: Dados da Pesquisa

A Figura 12 apresenta a solicitação do nome e senha do usuário cadastrado no *Proxy*. Observa-se que a tela de autenticação mostra as informações de qual servidor são solicitadas e em qual porta esse serviço está sendo executado.

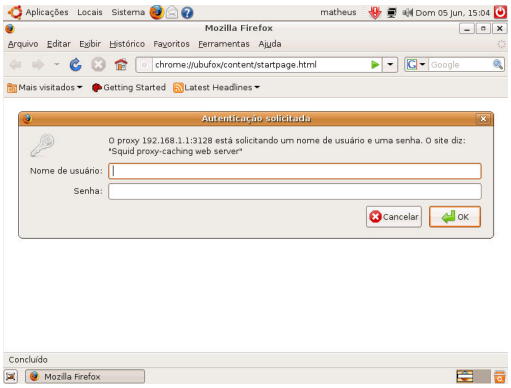

Figura 12 – Tela do *browser* solicitando *login* e senha
Fonte: Dados da Pesquisa

A partir do cadastro de usuários a ferramenta guiará o usuário pelas demais configurações, tais como de *cache*, liberação e bloqueio de portas, definição da faixa de endereços IP e a máscara da sub-rede, que no caso da topologia em questão são respectivamente 192.168.1.0 e 255.255.255.0.

A Figura 13 apresenta o relatório *allow/deny*, esse relatório foi criado para facilitar a visualização de acessos negados e permitidos.

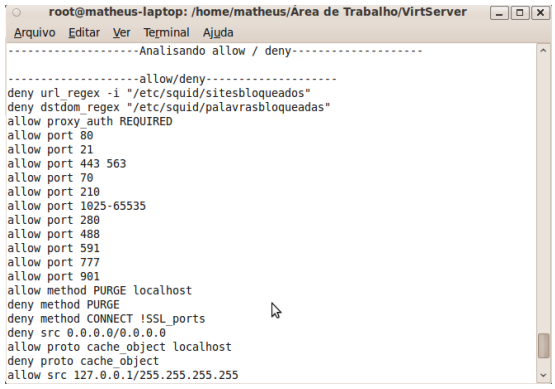

Figura 13 – Relatório de permissões (*allow/deny*)
Fonte: Dados da Pesquisa

Além do relatório de permissões o usuário pode ainda visualizar os relatórios de acl's e o relatório de configurações do *Proxy*.

4.3 Bind

Para o *Bind* foi configurado um servidor de nomes contendo apenas um servidor primário e um *DNS* reverso. Na Figura 14, pode ser observada que essa configuração corresponde a opção 3 do menu.

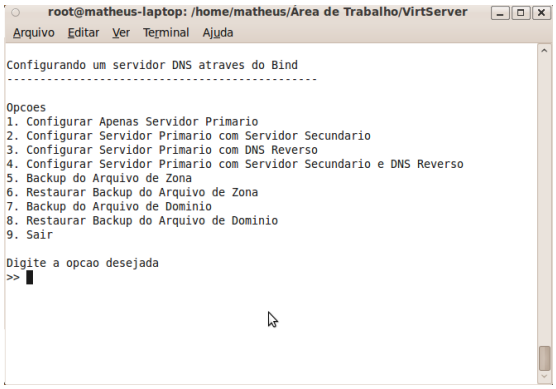

Figura 14 – Menu inicial de configuração do *DNS*
Fonte: Dados da Pesquisa

A configuração da zona é bastante simples como mostra a Figura 15. Nesse passo o usuário deve digitar apenas o nome do *site*, para que a ferramenta se encarregue

de executar o restante da configuração. Ainda na Figura 15 existe uma parte da configuração do domínio, onde foi informado o *e-mail* do administrador do domínio. O *e-mail* não pode conter @, por isso a ferramenta faz essa verificação não permitindo que esse erro aconteça.

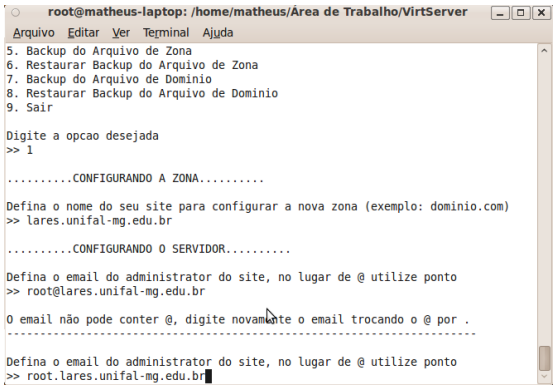

Figura 15 – Tela de configuração da zona lares.unifal-mg.edu.br
Fonte: Dados da Pesquisa

A Figura 16 mostra a configuração dos dados responsáveis por manter o servidor secundário atualizado.Se o usuário optar pelas configurações padrão, essa etapa poderá ser realizada rapidamente, pois basta pressionar a tecla *enter* para que isso seja feito.

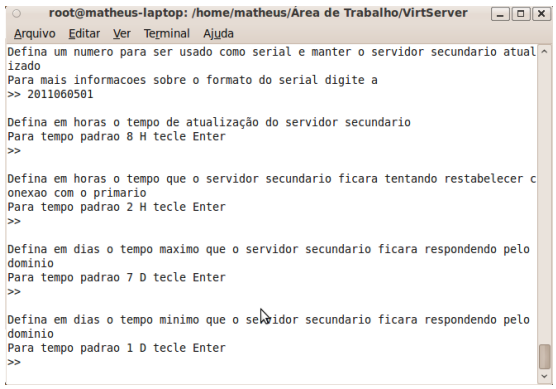

Figura 16 – Tela de configuração dos dados de atualização do *DNS* secundário
Fonte: Dados da Pesquisa

Para a configuração em questão foi definido para o servidor primário o *IP* 192.168.1.1 e os serviços www, e *FTP*, como mostra a Figura 17.

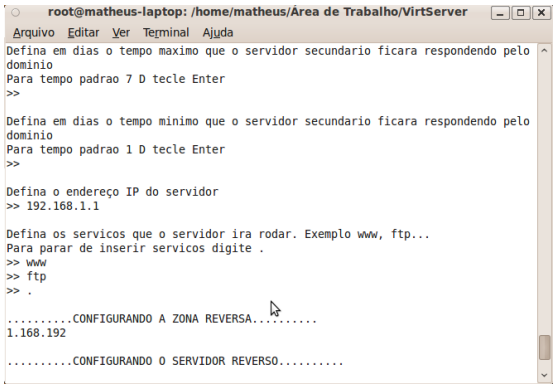

Figura 17 – Tela de configuração do *IP* do servidor primário e serviços
Fonte: Dados da Pesquisa

Para verificar se a ferramenta executou corretamente as configurações do servidor *DNS*, foi dado o comando ping www.lares.unifal-mg.edu.br. A Figura18 mostra uma perfeita comunição entre o cliente e o servidor *DNS*, inclusive no que se refere ao *DNS* reverso, pois em cada linha são apresentados os dados 1.168.192.in-addr.arpa, que corresponde a zona do *DNS* reverso do domínio lares.unifal-mg.edu.br.

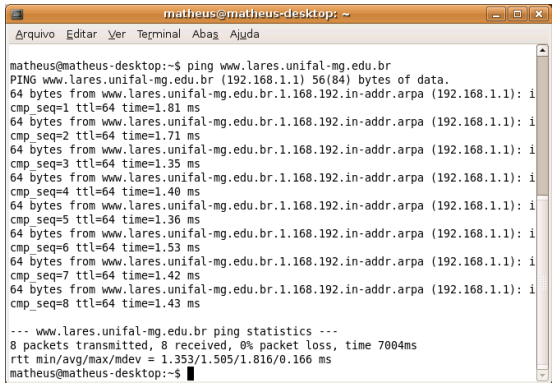

Figura 18 – Comando ping para verificar as configurações do *DNS*
Fonte: Dados da Pesquisa

Um outro comando utilizado para analisar a configuração do *DNS* foi o dig, que tem o formato dig <domínio> @<endereço IP>. Assim obteve-se os dados presentes na Figura 19.

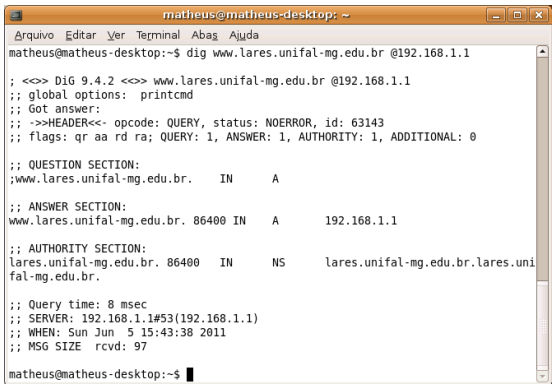

Figura 19 – Comando dig para verificar as configurações do *DNS*
Fonte: Dados da Pesquisa

Com o comando dig pode-se verificar que o domínio lares.unifal-mg.edu.br realmente está executando no endereço IP 192.168.1.1.

4.4 DHCP3

Antes de iniciar a configuração do servidor *DHCP*, para cada cliente, deve-se fazer uma alteração no arquivo interfaces, que é encontrado dentro da pasta /etc/network, incluindo o seguinte conteúdo iface <interface de rede > init dhcp. Na configuração a seguir a interface de rede foi a eth0.

A Figura 20 apresenta o menu inicial de configuração do *DHCP*, a opção escolhida foi a 2, pois o objetivo era atribuir a rede *IP* dinamicamente e deixar uma máquina com *IP* fixo.

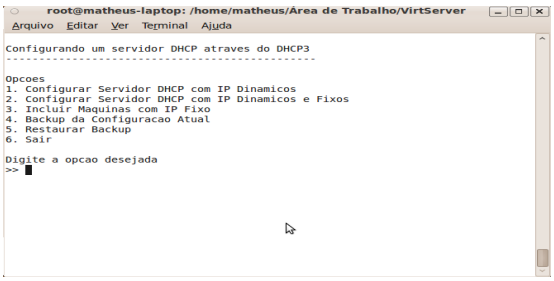

Figura 20 – Menu inicial de configuração do *DHCP*
Fonte: Dados da Pesquisa

O tempo escolhido para que o servidor verifique se as estações estão ativas foi de 600 minutos. E o tempo que uma estação pode permanecer com determinado endereço foi de 7200 minutos.

O endereço definido foi o 192.168.1.0 conseqüentemente a mascará teve que ser 255.255.255.0. Como a topologia proposta apresenta dez maquinas onde uma terá *IP* fixo, a faixa de endereços dinâmicos ficou entre 192.168.1.2, pois o IP com final 1 pertence ao servidor, e 192.168.1.10. As informações citadas anteriormente são observadas na Figura 21.

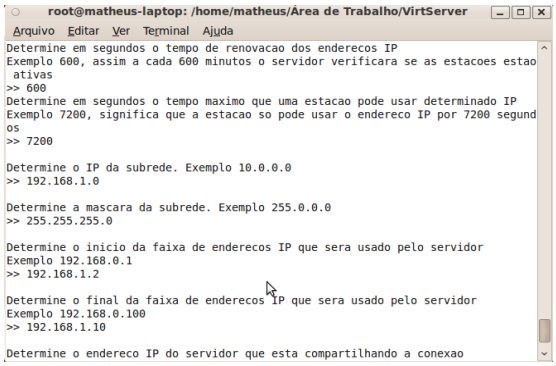

Figura 21 – Tela de configuração de *IP* dinâmico
Fonte: Dados da Pesquisa

O endereço do servidor que compartilha a conexão foi o mesmo endereço do servidor que rodará os serviços de *Proxy*, *DNS* e *DHCP*. Uma atribuição de um endereço *IP* dinâmico está presente na Figura 22, onde foi executado o comando ifconfig no cliente.

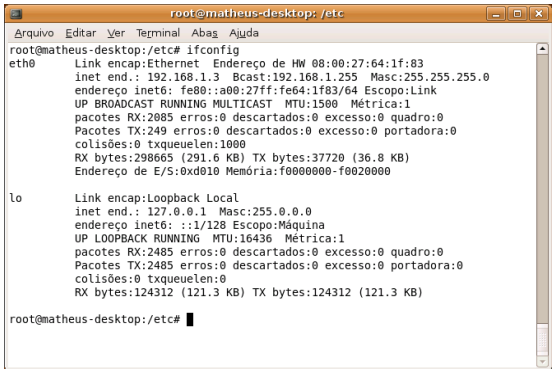

Figura 22 – Comando ifconfig mostrando *IP* dinâmico
Fonte: Dados da Pesquisa

Com exposto anteriormente, uma máquina ficou com IP fixo. Essa maquina foi chamada de maquina1 e teve seu endereço MAC 08:00:27:64:1f:83 atrelado ao *IP* 192.168.1.11, essa configuração é simples e feita em apenas três passos, como mostra a Figura 23.

Figura 23 – Menu inicial de configuração do *DHCP*
Fonte: Dados da Pesquisa

A Figura 24 é apenas mostra o resultado do comando ifconfig após a configuração do *IP* fixo.

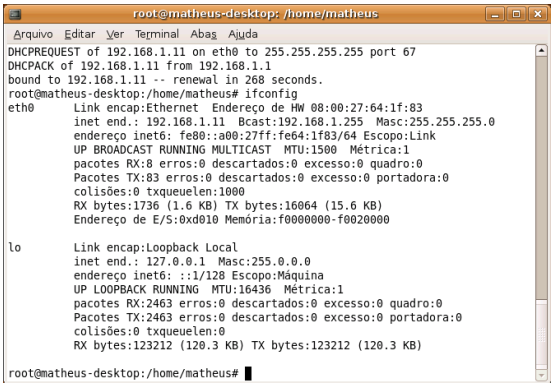

Figura 24 – Comando ifconfig mostrando *IP* fixo
Fonte: Dados da Pesquisa

Com a configuração do *DHCP* encerra-se a análise dos resultados e os testes com as configurações realizadas com a ferramenta.

5
Conclusões e Trabalhos Futuros

A principal dificuldade encontrada durante o desenvolvimento desse trabalho, e em especial da ferramenta, foi quanto às configurações dos arquivos de cada serviço. Pois antes de implementar uma funcionalidade na ferramenta, era necessário localizar o arquivo de configuração do serviço desejado e fazer a configuração manualmente.

Assim, a cada modulo tinha-se que partir para o objetivo específico de estudar a configuração do servidor, verificar suas particularidades e fazer testes em suas configurações, para que possíveis erros não fossem repassados para a ferramenta. Erros esses muitas vezes referentes ao grau de detalhe do serviço, tal como a ordem em que as permissões devem aparecer na configuração do *Proxy.*

Por esse motivo é possível concluir que embora eficiente, as configurações de servidores de fato envolvem a edição de diversos arquivos, e detalhes minuciosos, que podem resultar na falha no funcionamento dos mesmos. Com os scripts desenvolvidos observou-se não apenas uma redução no tempo necessário para configurar um servidor, como também na probabilidade de que ocorra algum erro, uma vez que a quantidade de configurações digitadas diminui consideravelmente.

Além disso, a ferramenta apresenta alguns controles que impedem que o usuário digite algum conteúdo inadequado, como por exemplo o @ no *e-mail* do administrador do domínio no *DNS*, e ainda garante a identação do arquivo de domínio. detalhes que podem passar despercebidos durante a configuração manual.

Como trabalhos futuros são propostos o desenvolvimento de mais alguns módulos para ferramenta, tais como configuração de servidores de *FTP*, *Web* e *SMTP*. Uma prioridade na continuidade do presente trabalho é a automatização da

configurações do Iptables. Serviço esse que é essencial mesmo em uma rede interna e que não foi contemplado nessa etapa de desenvolvimento.

6
Referências Bibliográficas

FERREIRA, Rubem E. Linux Guia do Administrador do Sistema. São Paulo: Novatec Editora Ltda, 2003. 510 p.

JARGAS, Aurélio Marinho. *Shell Script* Profissional. São Paulo: Novatec Editora, 2008. 480 p.

MAZIOLI, Gleydson da Silva. Guia Foca GNU/Linux. GNU Free Documentation License , 2006. 427 p.

MORIMOTO, Carlos E. Linux Servidores: Guia Pratico.2.ed. São Paulo: GDH Press e Sul Editores, 2008. 736 p.

TANEMBAUM, Andrew S., WHOODHULL, Albert S. Sistemas Operacionais: Projeto e Implementação. 2.ed.Porto Alegre, Bookman Companhia Editora, 2000. 343 p.

TIBET, Chuck V. Linux Administração e Suporte. São Paulo: Novatec Editora Ltda, 2001. 379 p.

TORRES, Gabriel. Redes de Computadores: Curso Completo. Rio de Janeiro: Axcel Books do Brasil Editora, 2001. 664 p.

KUROSE, James F.,ROSS, Keith W., Redes de Computadores e a Internet: Uma Abordagem Top-Down. 3.ed. São Paulo: Pearson Addison weasley, 2006. 634 p.

WEBMIN. *What is Webmin*? Disponível em: < http://www.webmin.com>. Acesso em: 02 de outubro de 2010.

KYAPANEL. Disponível em: <http://www.kyapanel.com>. Acesso em: 02 de outubro de 2010.