Centro Universitário de Brasília – UniCEUB
Faculdade de Exatas e Tecnologia – FAET
Curso de Engenharia da Computação

TIAGO DE ASSIS OLIVEIRA CASTRO

ESTUDO DE CASO: MIGRAÇÃO DE ESTAÇÕES PARA SOFTWARE LIVRE EM DOMÍNIO MICROSOFT

Brasília – DF
2007

TIAGO DE ASSIS OLIVEIRA CASTRO

ESTUDO DE CASO: MIGRAÇÃO DE ESTAÇÕES PARA SOFTWARE LIVRE EM DOMÍNIO MICROSOFT

Trabalho apresentado ao
Centro Universitário de Brasília - UniCEUB
Como requisito parcial para obtenção do título de
Engenheiro da Computação

Orientador: Prof. MSc. Fabiano Mariath D`Oliveira

Brasília – DF
2007

DEDICATÓRIA

Ao Grande Arquiteto do Universo.
À minha família e amigos,
Pelo apoio durante essa caminhada.
À Graziele Mendes,
Pelo total apoio e compreensão.
Dedico-lhes essa vitória.

AGRADECIMENTOS

Aos meus pais, Francisco de Assis e Maria das Graças e aos meus irmãos, Robson e Carla, que me deram todo apoio necessário para que eu pudesse chegar até aqui, vocês são a base dessa conquista.

Agradeço intensamente a Graziele Mendes, pelo total apoio e compreensão durante essa etapa da minha vida. Essa vitória também é sua.

Ao meu orientador, Msc. Fabiano Mariath D'Oliveira, pelo intenso comprometimento e crédito dado a esse projeto.

Aos amigos e colegas de trabalho do Departamento de Produtos e Tecnologias (DPT) da Autotrac, pela ajuda e incentivo desde o princípio do curso.

Obrigado aos eternos amigos que fizeram parte direta ou indiretamente dessa conquista. Os integrantes da "CPO", do "TNSEA", dos "PINGÜINS" e todos que estiveram comigo em cada passo dessa caminhada.

RESUMO

Grande parte do custo com licenciamento de softwares na área de TI vem do sistema operacional e das ferramentas de escritório usados nas estações de trabalho (Pois necessitam de um número grande de licenças), o uso dos Softwares Livres nelas auxilia a reduzir o Custo Total de Propriedade (TCO) nas empresas.

Contudo, a migração de software proprietário para livre em estações de trabalho requer, entre outros esforços, a definição de onde os usuários dessas estações irão se autenticar e um conhecimento técnico mais avançado para alterar vários parâmetros em diversos arquivos de configuração.

Esse projeto visa apresentar um modelo de rede onde exista apenas uma base para autenticação de usuários que usam estações com softwares livres ou proprietários. Para isso, será criado um ambiente com um controlador de domínio Microsoft e estações com sistema operacional livre e proprietário; também será desenvolvido um software com uma interface simples, que poderá ser usado por pessoas sem conhecimentos técnicos avançados para adicionar as estações ao domínio de uma maneira fácil, sem que haja necessidade de abrir e alterar manualmente vários arquivos de configuração.

Serão utilizados o Windows Server 2003 e o Active Directory da Microsoft para criação de um Controlador de Domínio e também o compilador Free Pascal e a IDE Lazarus para o desenvolvimento do software das estações de trabalho com o sistema operacional GNU/Linux.

Palavras-chave: TCO, Kerberos, Winbind, Free Pascal, Lazarus, Active Directory, Windows e Linux.

ABSTRACT

Great part of the cost with licensing of softwares in the IT comes of the operational system and of the used tools clerical in the workstations (Therefore they need a great number of licenses), the use of Free Softwares in them assists to reduce the Total cost of ownership (TCO) in the companies.

However, the migration of proprietor software for free in workstations requires, among others efforts, the definition of where the users of these stations will go to legalize themselves and a knowledge more advanced technician to modify some parameters in diverse archives of configuration.

This project aims at to present a net model where only one base for authentication of users who use free or proprietors workstations. For this, it will be created an environment with a domain controller Microsoft and workstations with free operational system and proprietor; also a software with a simple interface will be developed, that could be used for people without advanced knowledge technician to add the stations to the domain in an easy way, without it has necessity to manually open and to modify some archives of configuration.

The Windows Server 2003 and Active Directory of the Microsoft for creation of a Domain Controller will be used and also the Free Pascal compiler and IDE Lazarus for the development of the software of the workstations of work with the operational system GNU/Linux.

Key Words: TCO, Kerberos, Winbind, Free Pascal, Lazarus, Active Directory, Windows and Linux.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1 – INTRODUÇÃO

1.1 MOTIVAÇÃO

No atual cenário Brasileiro (alta carga tributária, dificuldade na obtenção de crédito e juros altos); as empresas de pequeno e médio porte carecem de um modelo de administração de rede que possibilite o desenvolvimento de atividades básicas em TI, sem que isso comprometa um volume significativo de recursos para investimento, que estão mais escassos nos tempos atuais.

Na Autotrac, empresa onde será realizado o estudo de caso surgiu uma demanda por parte da Diretoria de Produtos e Tecnologias no sentido de reduzir os gastos com licenciamento de softwares em TI. Foi sugerido que fosse realizado um estudo completo para migrar sistemas que eram proprietários para software livre, abrangendo inclusive os custos dessa nova solução.

Essa necessidade gerou um plano de trabalho para resolver esse problema do alto valor gasto com licenças, principalmente os produtos da empresa Microsoft.

A rede da Autotrac, que dispõe de um site central na matriz, mais 16 sites remotos em diversas localidades do país e é composta basicamente de servidores e estações de trabalho Microsoft. Seu sistema interno feito em Java roda em uma Intranet e é acessado via navegador de internet.

O maior custo com licenciamento em TI acontece nas estações de trabalho (pois necessitam de um número grande de licenças), o uso dos Softwares Livres nelas faz reduzir significativamente o custo com aquisição de licenças. Já na parte servidora, o uso de Software Proprietário mostra-se vantajoso devido a facilidade na manutenção do ambiente sem onerar muito a solução, já que o número de licenças é reduzido.

Contudo, a migração das estações de trabalho requer, entre outros esforços, a definição de uma base para autenticação do sistema operacional livre e um esforço técnico para inclusão dessas máquinas no domínio proprietário existente, uma vez que para isso deverão ser alterados diversos parâmetros em alguns arquivos de configuração de softwares de integração.

1.2 OBJETIVO

Desenvolver e implementar um projeto de migração de estações de trabalho que permita a integração entre a plataforma de softwares livres e softwares proprietários. Especificamente uma plataforma composta por Windows Server 2003 na parte servidora e outra composta por GNU/LINUX nas estações de trabalho.

Os objetivos do projeto serão alcançados com:

1º) O êxito na integração das duas plataformas, ou seja, as estações que estarão rodando software livre irão se autenticar na rede Microsoft disponibilizada pelo Controlador de Domínio para acessar os recursos disponibilizados por ela;

2º) O desenvolvimento de um software com uma interface simples que irá permitir que usuários que possuam informações básicas do domínio consigam adicionar essas estações com softwares livres ao domínio Microsoft, fazendo com que elas se comuniquem com os servidores proprietários de acordo com os parâmetros passados para o mesmo.

3º) A apresentação dos valores economizados com a implantação do modelo proposto, bem como a curva da economia gerada com o projeto e seu ponto de saturação.

Espera-se com esse projeto de migração uma redução de custos resultado da não necessidade de adquirir licenças para as estações que irão utilizar softwares livres.

1.3 METODOLOGIA UTILIZADA

O método utilizado nesse projeto será o Estudo de Caso, esse método foi realizado na empresa Autotrac Comércio e Telecomunicações S/A.

O método de estudo de caso é um método específico de pesquisa de campo. Estudos de campo são investigações de fenômenos à medida que ocorrem, sem qualquer interferência significativa do pesquisador. Seu objetivo é compreender o evento em estudo e ao mesmo tempo desenvolver teorias mais genéricas a respeito dos aspectos característicos do fenômeno observado. [Fidel, 1992].

Esse método consiste em uma investigação detalhada de uma ou mais organizações, ou grupos dentro de uma organização, com vistas a prover uma análise do contexto e dos processos envolvidos no fenômeno em estudo. O fenômeno não está isolado de seu contexto (como nas pesquisas de laboratório), já que o interesse do pesquisador é justamente essa relação entre o fenômeno e seu contexto. [Hartley, 1994].

1.4 ESTRUTURA DA MONOGRAFIA

No capítulo 1 está descrito a motivação para a criação do projeto, bem como o objetivo esperado para o mesmo.

No capítulo 2 serão abordados os conceitos sobre os quais o projeto foi desenvolvido, apresentando os detalhes sobre as tecnologias e os softwares que serão usados para conceber a solução como um todo. Nesse capítulo estarão referenciados os locais onde foram pesquisados os temas.

No capítulo 3 será apresentado o ambiente proposto e sua implementação, detalhando todos os passos que foram realizados para a concepção do projeto. Também será apresentado o software Leasyconfig que foi desenvolvido.

Finalmente, no capítulo 4 serão feitas conclusões sobre o projeto e sugestões para futuros trabalhos que poderão surgir a partir da evolução do mesmo.

CAPÍTULO 2 – REFERENCIAL TEÓRICO

Nesse capítulo serão abordados os conceitos técnicos nos quais o projeto está embasado. Esses conceitos estarão divididos em: Segurança, Autenticação, e Serviços de Rede, Serviço de Diretório Microsoft e Sistema Operacional GNU/LINUX.

2.1 SEGURANÇA

No início da era da informática, a segurança das informações e dos sistemas computacionais não era uma questão tão importante, pois estes recursos tinham um acesso limitado, ficando confinados a uma mainframe protegido de qualquer acesso externo.

Com a evolução da informática, o surgimento dos PC's e de redes de computadores, tornou-se possível um compartilhamento de recursos e das informações. Este compartilhamento aumentou a facilidade do usuário acessar estas informações, e consequentemente cresceu também a probabilidade de intrusos compartilharem também deste acesso. Isto fez crescer muito a importância da segurança nos sistemas de informática.

Hoje, existe a necessidade de redes corporativas se comunicarem entre si através de redes abertas, mas garantindo a segurança dos seus sistemas. [FILHO, 1999]

O conceito de segurança abrange os seguintes aspectos:

-Autenticação: Assegura que a entidade que está fazendo o acesso (Máquina ou pessoa) é realmente quem declara ser.

-Confidencialidade: A confidencialidade numa comunicação é garantida se apenas os usuários autorizados tenham acesso as informações devidas e ninguém mais.

-Integridade: Permite que a informação que está sendo recebida é a mesma que foi enviada.

-Autorização: Permite que uma parte esteja autorizada a fazer o que está requisitando, geralmente ocorrendo após o processo de autenticação.

-Disponibilidade: A disponibilidade de Sistemas Computacionais está associada à continuidade dos seus serviços acessíveis aos usuários autorizados. Um Sistema pode ser vitimado por acidentes ou alvo de vários tipos de ataques interrompendo a disponibilidade dos seus serviços. [MOITINHO, 2001]

2.2 AUTENTICAÇÃO

A autenticação é a técnica através da qual um processo confirma que seu parceiro na comunicação é quem deve ser e não um impostor. Confirmar a identidade de um processo remoto, face à presença de um intruso ativo mal-intencionado, é surpreendentemente difícil e exige protocolos complexos baseados no uso da criptografia.

Por outro lado, algumas pessoas confundem autorização com autenticação. A autenticação lida com a questão de determinar se você está ou não se comunicando com um processo específico. A autorização se preocupa com o que essa entidade tem permissão para fazer. [TANENBAUM, 2003]

O modelo genérico que todos os protocolos de autenticação utilizam é descrito a seguir. O usuário A começa enviando uma mensagem para o usuário B ou para um KDC (Key Distribution Center) no qual confia e que sempre é honesto. Acontecem muitas outras trocas de mensagens em diferentes sentidos. À medida que essas mensagens são enviadas, um intruso mal-intencionado, o usuário T pode interceptar, modificar ou reproduzir essas mensagens a fim de enganar A e B, ou simplesmente para atrapalhar.

Todavia, quando a execução do protocolo tiver sido concluída, A terá certeza de que está se comunicando com B e vice-versa. Além disso, na maioria dos protocolos, os dois também terão estabelecido uma chave de sessão secreta que deverá ser usada durante a conversação. Na prática, por motivos de desempenho, todo o tráfego de dados é criptografado utilizando-se a criptografia de chave simétrica (em geral, AES ou DES triplo), embora a criptografia de chave pública seja extensamente usada nos próprios protocolos de autenticação e para estabelecer a chave de sessão.

O objetivo de se utilizar uma nova chave de sessão escolhida aleatoriamente para cada nova conexão é minimizar o volume de tráfego provocado pelo envio das chaves secretas ou públicas dos usuários, reduzir o volume de texto cifrado que um intruso pode obter e minimizar os danos causados, caso haja uma pane em um processo e seu dump de memória caia em mãos erradas. É muito provável que a única chave presente seja a chave de sessão. Todas as chaves permanentes deverão ser cuidadosamente zeradas depois que a sessão for estabelecida. [TANENBAUM, 2003]

2.3 KERBEROS

Kerberos é um serviço de autenticação distribuída desenvolvido no MIT (Massachusetts Institute of Technology) que permite um parceiro provar sua identidade perante um outro parceiro sem enviar dados confidenciais pela rede. Esse processo é realizado como um serviço de autenticação terceiro parceiro confiável utilizando criptografia convencional; opcionalmente ele também fornece integridade e confidencialidade das mensagens intercambiadas. [PINTO, 2006]

2.3.1 HISTORIA E COMPONENTES

A origem do nome desse serviço é proveniente da mitologia grega, onde KERBEROS era o nome do cachorro de três cabeças que vigiava os portões de Hades e sua principal missão era evitar a entrada de pessoas ou coisas

indesejáveis. Assim sendo foi o nome dado ao serviço de autenticação do projeto Athena por ele estar baseado em três servidores:

1) Servidor de Autenticação ou Authentication Server (AS): Confirma a identidade dos usuários durante o processo de login;

2) Servidor de Concessão de Ticket ou Ticket-Granting Server (TGS): Emite “bilhetes de comprovação de identidade”.

3) Servidor de Administração ou Kerberos Administration Server (KADM): Aplica políticas de segurança para formação de senhas.

O Kerberos utiliza Criptografia para provar a identidade do usuário manipulando dois tipos de chave:

• chave secreta do usuário - chave conhecida apenas pelo usuário e pelo Kerberos com a finalidade de autenticar o usuário ao Kerberos; deverá existir uma etapa anterior em que serão cadastrados os clientes e suas chaves secretas ficando armazenadas na Base de Dados do Kerberos;

• chave de sessão - chave gerada pelo Kerberos após ter autenticado o usuário e tem por objetivo autenticar o intercâmbio realizado por um determinado par de usuários que definem uma sessão; a chave é gerada atendendo a uma solicitação feita por um dos usuários, sendo válida por um tempo pré-determinado e conhecido apenas pelos dois parceiros para os quais ela foi originalmente gerada. [PINTO, 2006]

O AS é semelhante a um KDC, porque compartilha uma senha secreta com todos os usuários. O trabalho do TGS é emitir bilhetes que possam convencer os servidores reais de que o portador de um bilhete TGS realmente é quem afirma ser. [TANENBAUM, 2003]

2.3.2 FUNCIONAMENTO

Para dar início a uma sessão, o usuário A utiliza uma estação de trabalho arbitrária e digita seu nome. A estação de trabalho envia seu nome ao AS em texto simples (vide Figura 2.3.2.1). O que retorna é uma chave de sessão e um bilhete, KTGS (A, KS), destinado ao TGS. Esses itens são empacotados juntos e criptografados com a chave secreta de A, de modo que apenas A seja capaz de descriptografá-los. Somente quando a mensagem 2 chega, a estação de trabalho pede a senha de A. Em seguida, a senha é usada para gerar KA, a fim de descriptografar a mensagem 2 e obter a chave de sessão e o bilhete TGS que ela contém. Nesse momento, a estação de trabalho substitui a senha de A, para garantir que a senha só estará na estação de trabalho durante alguns milissegundos, no máximo. Se T tentar estabelecer um login como A, a senha que o usuário A digitar estará errada e a estação de trabalho detectará o problema, porque o trecho padrão da mensagem 2 estará incorreto.

Depois de estabelecer o login, A pode informar à estação de trabalho que deseja entrar em contato com B, o servidor de arquivos. Em seguida, a estação de trabalho envia a mensagem 3 ao TGS solicitando um bilhete para usar com B. O principal elemento nessa solicitação é KTGS(A, KS), que é criptografado com a chave secreta de TGS e é usado como prova de que o transmissor realmente é A. O TGS responde criando uma chave de sessão, KAB, para que A a utilize com B. Duas versões dessa chave são retornadas. A primeira é criptografada apenas com KS, para que Alice possa ler a mensagem. A segunda é criptografada com a chave de B, KB, de forma que B também possa ler a mensagem. [TANENBAUM, 2003]

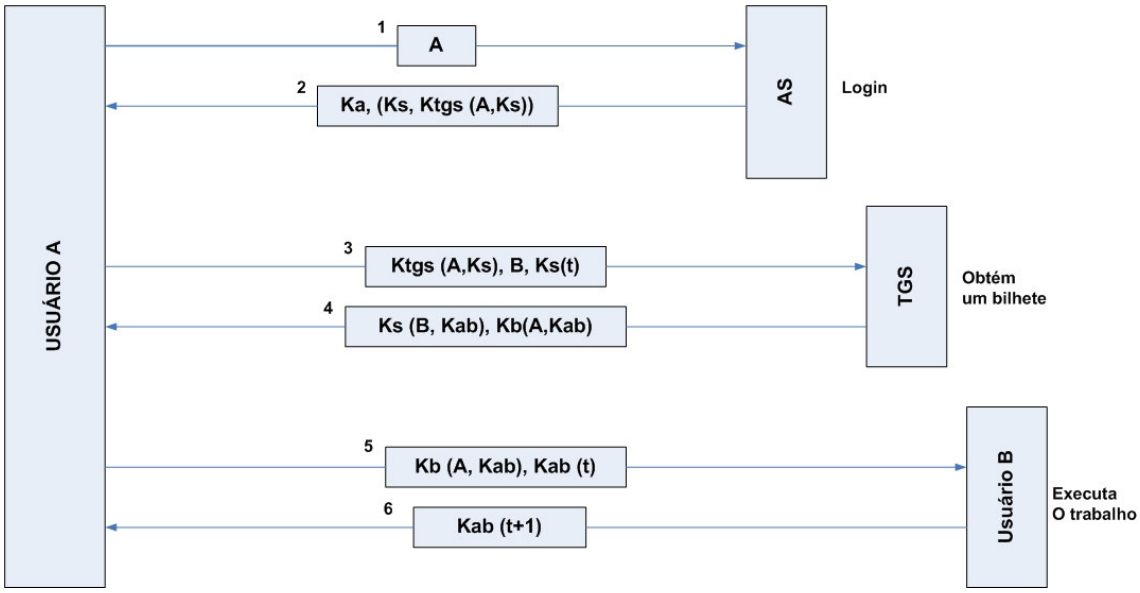

Figura 2.3.2.1 – Operação do Kerberos
(Fonte: TANENBAUM, Andrew S – Rede de Computadores 4ª Edição. Página 847)

2.3.3 SERVIDOR TGS (TICKET GARANTING SERVER)

No funcionamento do Kerberos acima mencionado surge um problema. Ele será usado toda vez que o usuário requisitar um serviço, e a cada vez que isto acontecer o usuário terá que entrar com sua senha. Uma primeira solução seria armazenar a chave do usuário gerada a partir da sua senha. Mas armazenar a chave do usuário é muito perigoso, pois com uma cópia desta um invasor se passaria pelo usuário até que sua senha fosse modificada.

O Kerberos resolve este problema introduzindo um novo agente chamado de ticket granting server (TGS). O TGS é logicamente distinto do AS, mas eles podem residir na mesma maquina. A função do TGS é a seguinte: antes de acessar qualquer serviço, o usuário requisita um ticket para contatar o TGS, como se ele fosse um serviço qualquer. Este ticket é chamado de ticket granting ticket (TGT).

Depois de receber o TGT, a qualquer momento que o usuário desejar requisitar um serviço, ele irá requerer um ticket não mais do AS, mais sim do TGS. Além disto, a resposta não será criptografada com a chave secreta do usuário, mas sim com a chave de sessão providenciada pelo AS para ser usada entre usuário e TGS. O conteúdo desta resposta é uma chave de sessão

que será usada com o serviço regular. O resto da comunicação continua como descrito anteriormente.

A vantagem é que enquanto senhas usualmente são válidas por meses, O TGT é válido somente por um curto período de tempo, tipicamente 8 horas. Depois deste tempo, o TGT não pode ser usado por ninguém, nem usuário nem invasor. Este TGT, como qualquer ticket que o usuário obtém é armazenado no credentials cache.[FILHO, 1999]

2.3.4 VERSÃO

A versão do Kerberos que será usada no projeto é a V5 que está presente no Active Directory do Windows Server 2003 e na estação de trabalho GNU/LINUX.

Nessa versão o tempo de duração de seus bilhetes é mais longo, ele permite que os bilhetes sejam renovados e emite bilhetes pós-datados. Além disso, pelo menos na teoria, ele não é dependente do DES e aceita vários domínios (realms), delegando a geração de bilhetes a diversos servidores. [TANENBAUM, 2003]

2.4 PAM (Plugglable Authentication Modules)

O serviço PAM (Módulos de Autenticação Conectáveis) é um conjunto de bibliotecas para a integração de múltiplos esquemas de baixo-nível (autenticação) com uma API de alto nível que permite que programas que necessitam de autenticações sejam escritos independentemente destes esquemas disponibilizados. [SAMAR e SCHEMERS, 1995]

O PAM funciona como um middleware entre as aplicações que efetuam a autenticação e a maneira como é feita essa autenticação. O objetivo do PAM é a separação entre as aplicações que fornecem determinados tipos de privilégios do desenvolvimento de métodos apropriados que autenticam o

utilizador. Para isso existe um conjunto de bibliotecas disponibilizadas que uma aplicação pode utilizar para que lhe seja feita uma determinada identificação de um utilizador. Assim sendo, as aplicações não têm de se preocupar com a autenticação dos utilizadores pois essa tarefa é transferida para o PAM.

Desta maneira, um administrador de redes pode estipular que método de autenticação será usado sem ter que modificar as aplicações, pode preferir que os utilizadores sejam identificados pela sua impressão digital em vez do habitual par username/password, por exemplo. No entanto, é necessário que estas aplicações consigam suportar as funcionalidades disponibilizadas pelo PAM. [JACQUET, 2006]

2.5 SAMBA

O Samba é um servidor e conjunto de ferramentas que permite que máquinas com sistema operacional GNU/Linux e Windows se comuniquem entre si, compartilhando serviços (arquivos, diretório, impressão) através do protocolo SMB (Server Message Block)/CIFS (Common Internet File System), equivalentes a implementação NetBEUI no Windows. O SAMBA é uma das soluções em ambiente UNIX capaz de interligar redes heterogêneas.

Com o Samba, é possível construir domínios completos, fazer controle de acesso em nível de usuário, compartilhamento, montar um servidor WINS, servidor de domínio, impressão, etc. Na maioria dos casos o controle de acesso e exibição de diretórios no samba são mais minuciosos e personalizáveis que no próprio Windows. [SILVA, 2006]

2.5.1 HISTÓRIA

Desenvolvido inicialmente por Andrew Tridgell devido a uma necessidade de montar um volume Unix em sua máquina DOS. Inicialmente ele utilizava o NFS, mas um aplicativo precisava de suporte NetBIOS. Andrew então utilizou um método muito avançado usado por administradores para

detectar problemas: escreveu um sniffer de pacotes que atendesse aos requerimentos para ter uma única função: analisar e auxilia-lo a interpretar todo o tráfego NetBIOS da rede.

Ele escreveu o primeiro código que fez o servidor Unix aparecer como um servidor de arquivos Windows para sua máquina DOS que foi publicado mais ou menos em meados de 1992 quando também começou a receber patches. Satisfeito com o funcionamento de seu trabalho, deixou seu trabalho de lado por quase 2 anos. Um dia, ele resolveu testar a máquina Windows de sua esposa com sua máquina Linux, e ficou maravilhado com o funcionamento do programa que criou e veio a descobrir que o protocolo era documentado e resolveu levar este trabalho a fundo melhorando e implementando novas funções.

O Samba atualmente é um servidor fundamental para a migração de pequenos grupos de trabalho à grandes domínios com clientes mistos. Nenhum servidor de rede NetBEUI conhecido proporciona tanta flexibilidade de acesso a clientes como o Samba para compartilhamento de arquivos/impressão em rede. As funções de segurança que foram adicionadas ao SAMBA hoje garantem um controle mais rigoroso que a própria implementação usada no Windows NT, incluindo o serviços de diretórios, mapeamento entre IDs de usuários Windows com Linux, PDC, perfis móveis e uma coisa que inclusive apresenta problemas no Windows: compatibilidade total entre as diferentes implementações de versões do Windows.

Sua configuração pode receber ajustes finos pelo administrador nos soquetes TCP de transmissão, recepção, cache por compartilhamento, configurações físicas que afetam a performance de rede. Seu código vem sendo melhorado constantemente pela comunidade de software livre, obtendo excelente performance com hardwares mais obsoletos. [SILVA, 2006]

2.5.2 SMB/CIFS

Por volta de 1984 a IBM criou uma API (application programming interface) para conectar seus computadores em rede chamada Network Basic Input/Output System – NetBIOS. A API NetBIOS provê para a aplicação um design rudimentar para conexão e o compartilhamento de dados com outros computadores. É interessante pensar no NetBIOS como extensões de chamadas padrões de rede da API BIOS. Com a BIOS, cada chamada de baixo nível está confinada ao hardware da máquina local e não precisa de ajuda alguma para ir ao seu destino. Originalmente o NetBIOS tinha que trocar instruções com computadores através de IBM PC ou redes Token Ring. Era então requerido um protocolo de transporte de baixo nível para carregar as requisições de um computador para o outro.

No final de 1985 a IBM lançou tal protocolo, que integrava a API NetBIOS. Era o NetBIOS Enhanced User Interface (NetBEUI). O NetBEUI foi desenvolvido para pequenas LANs (com menos de 255 nós, o que era uma restrição prática em 1985) e permitia que cada máquina na rede clamasse um nome (de até 15 caracteres) que não estivesse já sendo utilizado por outra máquina. O protocolo NetBEUI era muito popular entre aplicativos de rede incluindo aqueles que rodam sobre Windows para Workgroups. Mais tarde implementações do NetBIOS sobre o protocolo IPX da Novell surgiram, competindo então com o NetBEUI. Apesar disto os protocolos utilizados pela comunidade da Internet eram o TCP/IP e o UDP/IP, e uma implementação da API NetBIOS sobre estes protocolos cedo se tornou necessária.

É importante lembrar que o TCP/IP utiliza números para representar o endereço dos computadores enquanto a NetBIOS utiliza apenas nomes. Isto representou um grande problema quando tentaram juntá-los. Em 1987 a IETF (Internet Engineering Task Force) publicou uma série de documento de padronização, entitulados RFC 1001 e 1002, que demonstravam como o NetBIOS deveria trabalhar sobre uma rede TCP/UDP. Desde então o padrão

que estes documentos regulamentam são conhecidos como NetBIOS sobre TCP/IP ou NBT. O padrão NBT estabelece o funcionamento de três serviços em uma rede : serviço de nomes, e dois serviços de comunicação; datagramas e sessões.

O serviço de nomes soluciona o problema de nome para endereço mencionado anteriormente, permitindo que cada computador declare um nome específico para si na rede que será traduzido em um nome “legível” – IP – pela máquina, de modo semelhante ao DNS (Domain Name Server) que existe na Internet. Os serviços de datagrama e sessão são protocolos secundários de comunicação usados para transmissão e recepção de dados de máquinas NetBIOS através da rede.

Em um mundo NetBIOS cada máquina, quando se conecta na rede, clama por um nome para si, o que é chamado de registro de nome (name registration). Entretanto, não pode haver mais de uma máquina reclamando o mesmo nome em um ambiente de trabalho, o que causaria tamanha confusão na comunicação entre as máquinas da rede. Existem então duas diferentes abordagens para evitar que isto ocorra:

1. usar o NetBIOS Name Server (NBNS) para manter um controle sobre qual cliente registrou cada nome NetBIOS;

2. permitir que cada máquina na rede reclame o seu nome quando uma outra máquina tentar registrar o nome já utilizado. [BARREIROS, 2001]

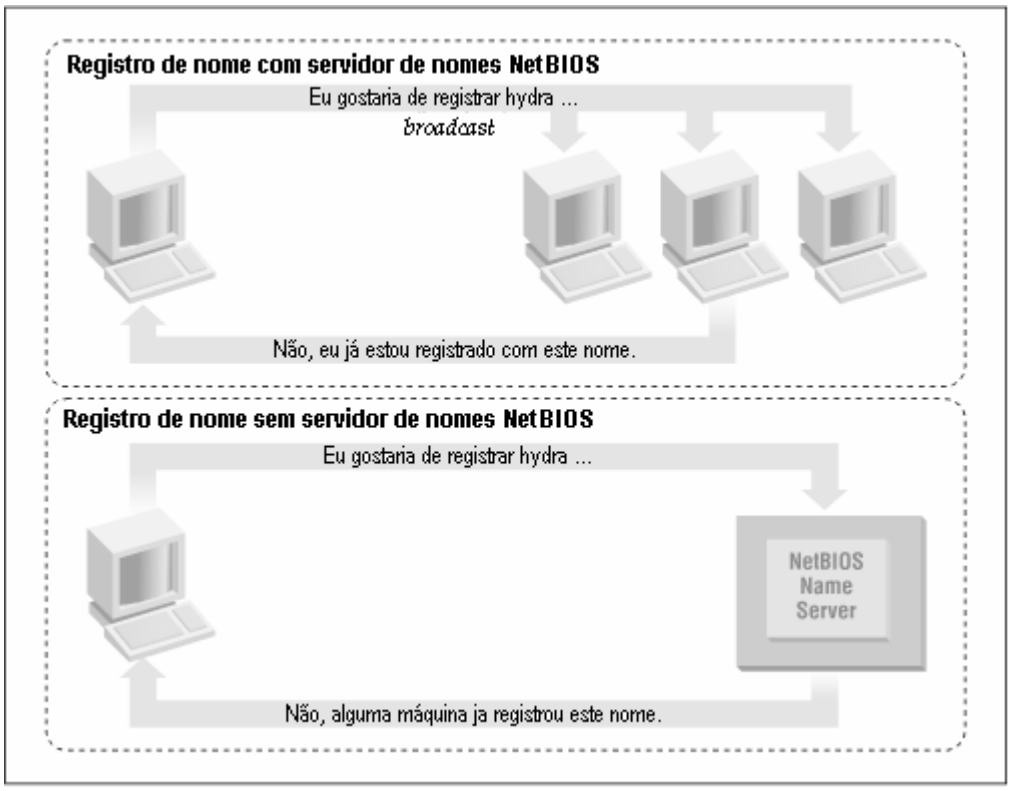

Figura 2.5.2.1 – Tentativa de registro de nome com e sem NetBIOS
(Fonte: http://www.gta.ufrj.br/grad/01_2/samba/images/netbios01.gif em 06/05/2007)

Além desta verificação do nome deve existir uma maneira de se traduzir o nome NetBIOS para um endereço IP, este processo é chamado de resolução de nome(name resolution). Na NBT esta resolução pode se dar de duas maneiras:

1. fazer com que cada máquina responda seu endereço IP quando "escuta" uma broadcast de requisição do seu nome NetBIOS;

2. usar o NBNS para ajudar na resolução do nomes NetBIOS para endereços IP.

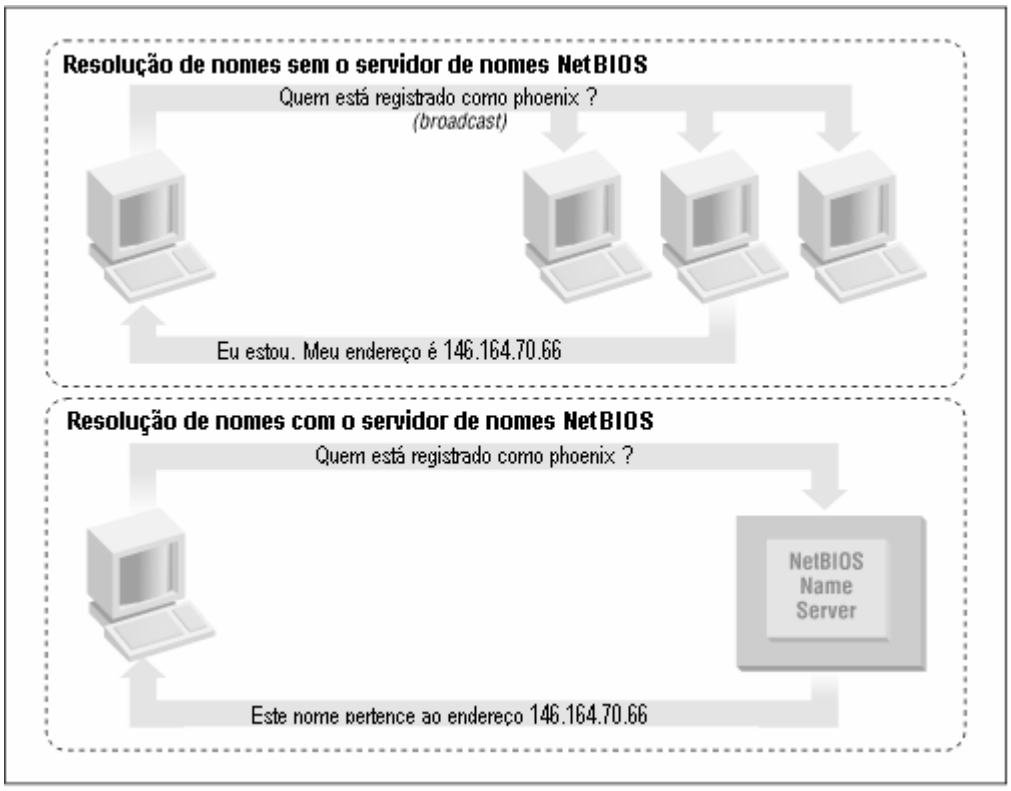

Figura 2.5.2.2 – Resolução de nomes com e sem NetBIOS
(Fonte: http://www.gta.ufrj.br/grad/01_2/samba/images/netbios02.gif em 06/05/2007)

Como podemos observar ter um NBNS na rede pode ser uma ferramenta extremamente poderosa. Veremos no caso do método sem o servidor de nomes. Quando uma máquina cliente se conecta à rede ele dá um broadcast declarando que quer registrar um nome NetBIOS específico. Se nenhuma máquina na rede reclamar do uso de tal nome o nome é registrado. Mas se alguma outra máquina no ambiente de trabalho já estiver utilizando este nome ela irá mandar uma mensagem a máquina que acabou de se conectar dizendo que tal nome já foi registrado. O que gera uma quantidade de tráfico muito grande para uma simples operação de registro de nome. Com um NBNS a mesma coisa ocorre com exceção de que a comunicação necessária para o registro ocorre somente entre a máquina que se conectou à rede e o NBNS. Não há então broadcast quando uma máquina tenta registrar seu nome na rede. A mensagem de registro é simplesmente enviada diretamente do cliente para o NBNS, e este responde se o nome já foi ou não registrado. Isto tipo de comunicação é conhecida como ponto a ponto, e é extremamente eficaz em redes com uma ou mais sub-redes.

O mesmo princípio pode ser aplicado à resolução de nomes. Sem um NBNS, a resolução de nomes NetBIOS seria feita através de um mecanismo de broadcast. Todos os pacotes de requisição seriam enviados para cada computador na rede com a esperança de que uma máquina responderia diretamente a máquina que fez a requisição. Fica claro então que a utilização de um NBNS resulta em entupimentos desnecessários na rede com broadcasts para cada requisição de registro de nome. [BARREIROS, 2001]

2.5.3 COMUNICAÇÃO SMB

Os elementos do protocolo request-response que o cliente e o servidor trocam são chamados de SMBs. Eles têm um formato específico que é muito similar entra o pedido e a resposta. Cada um consiste de um cabeçalho de tamanho fixo, seguido por uma VARIABLE SIZED PARAMETER e uma parte de dados.

Depois de conectar no nível do NetBIOS, também via NBF, NetBT e outros citados, o cliente está pronto para requisitar serviços do servidor. No entanto, o cliente e o servidor devem primeiro identificar qual variante do protocolo cada um deles entende.

O cliente envia um negprot SMB para o servidor, listando os dialetos de protocolo que ele entende. O servidor responde com o índice do dialeto que ele quer usar, ou 0xFFFF se nenhum dos dialetos do cliente é suportado pelo servidor conforme ilustra a figura abaixo.

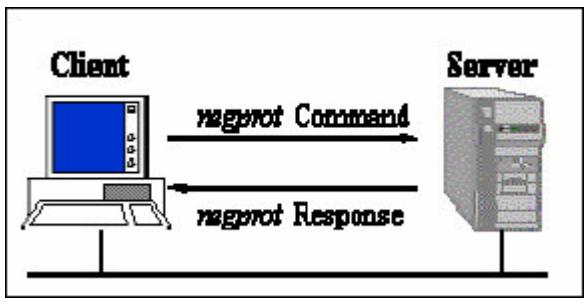

Figura 2.5.3.1 – Comunicação Negprot SMB

(Fonte: http://santarosa.rs.gov.br/~smb/samba.pdf)

Dialetos mais recentes que os protocolos Core e CorePlus fornecem informações na resposta do negprot para indicar suas capacidades (tamanha maior de buffer, nome de arquivos canônicos, etc).

Uma vez que o protocolo de comunicação foi definido, o cliente pode fazer o logon no servidor, se requerido. Eles fazem isso com um sesssetupX SMB (Figura 2.5.3.2).

A resposta diz se o cliente forneceu ou não o par de username e password validos e se, então, pode oferecer informações adicionais. Um dos mais importantes aspectos da resposta é o UID do usuário logado. Este UID deve ser submetido com todas as subseqüentes conexões via SMBs no servidor.

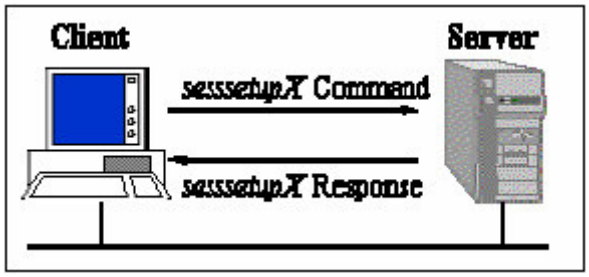

Figura 2.5.3.2 – Logon SessetupX SMB

(Fonte: http://santarosa.rs.gov.br/~smb/samba.pdf)

Uma vez que o cliente tenha se conectado (e em velhos protocolos – Core e CorePlus – não pode logar), este cliente pode executar a conexão com a (TREE) árvore. O cliente envia um tcon ou tconX SMB especificando o nome da rede do compartilhamento que quer se conectar, e se todos estão KOSHER, o servidor responde com um TID que o cliente usa em futuras SMBs relacionada com aquele compartilhamento (Vide Figura 2.5.3.3).

Conectado na arvore, o cliente pode agora abrir um arquivo com um open SMBs, seguido da leitura com read SMBs, escrita com write SMBs, e fechamento com close SMBs. [MAYER, FRIES e BAÚ, 2004]

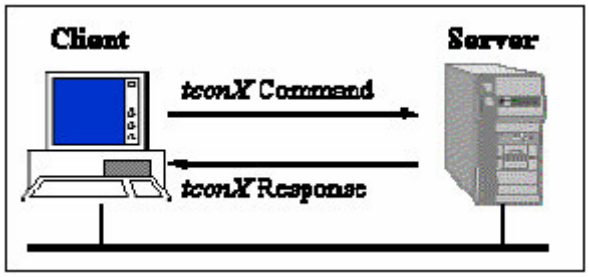

Figura 2.5.3.3 – Conexão TconX SMB

(Fonte: http://santarosa.rs.gov.br/~smb/samba.pdf)

2.5.4 CABEÇALHO SMB/CIFS

Todos as requisições e respostas SMB/CIFS possuem o seguinte template de cabeçalho apresentado na figura abaixo:

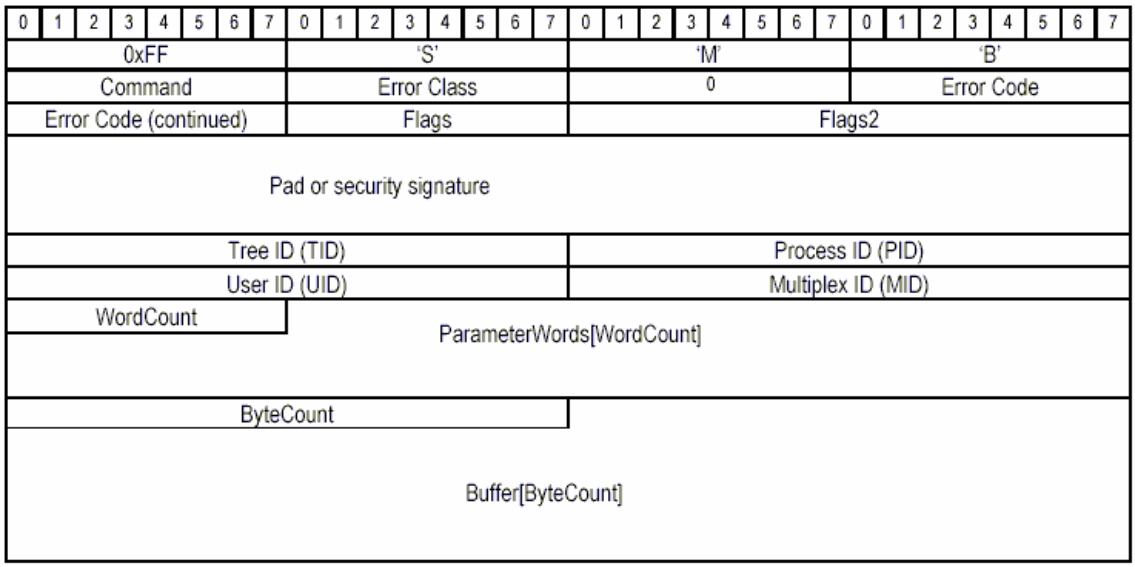

Figura 2.5.4.1 – Template de Cabeçalho SMB/CIFS

(Fonte: http://www.gta.ufrj.br/grad/01_2/samba/images/headersmb.gif)

Cabeçalho: O início de cada pacote SMB/CIFS contêm um cabeçalho de 4 octetos. O primeiro octeto é 0xFF seguido da representação ASCII das letras S, M e B.

Command: O campo de comando contêm um código de 1 octeto indicando o tipo de pacote. Alguns exemplos: SMB_COM_READ_ANDX7 (0x2e), SMB_COM_TREE_CONNECT (0x70) e SMB_COM_NEGOTIATE (0x72)

Error class: O servidor indica se uma requisição específica foi recebida com este campo. Tipicamente o campo é zero indicando o success. Se for um número diferente de zero o campo identifica a classe de erro, quando seta este campo pode ter os seguintes valores:

1. ERRDOS (0x01) – Erro é do núcleo do conjunto de instruções do sistema operacional DOS

2. ERRSRV (0x02) – Erro é gerado pelo gerenciador de arquivos de rede do servidor

3. ERRHRD (0x03) – Erro no hardware

4. ERRCMD (0xFF) – O comando não estava no formato 'SMB'

Error code: Este campo de 16 octetos indica o tipo de erro que ocorreu. Esta tipicamente setado em zero indicando que não houve erro. Se setado este número em conjunto com o error class define o erro ocorrido. Alguns destes erros são "bad password" ou "file does not exist". Assim como o campo error class este campo somente é setado pelos servidores nos pacotes de resposta às requisições.

Flags: A maioria dos oito octetos neste campo especificam opções particulares. A não ser o bit 3 que quando setado todos os caminhos devem ser tratados sem se preocupar se os caracteres são maísculos ou minúsculos

Flags2: Mais opções. Alguns bits úteis:

1. bit 0 - se setado o servidor poderá retornar arquivos com nomes longos

2. bit 6 - se setado indica que qualquer caminho na requisição pode ser um arquivo com nome longo

3. bit 16 - se setado indica que as strings no pacote estão codificadas com UNICODE.

Pad/security signature: Tipicamente setado em zero.

Tree ID (TID): O TID é um número de 2 octetos para identificar que recurso este pacote em particular esta se referindo. Quando os pacotes são trocados, o que não tem nada a ver com um recurso este campo não faz sentido e é ignorado. Se um cliente deseja receber acesso a um recurso o cliente envia um pacote com o campo de comando setado em SMB_COM_TREE_CONNECT_ANDX. Neste pacote o nome do compartilhamento ou a impressora é especificado (ex. \\server\dir). O servidor então verifica se o recurso existe e se o cliente possue acesso e então libera o compartilhamento enviando uma resposta de sucesso. Neste pacote o servidor irá setar qualquer número no campo TID. A partir daí o cliente quando quiser fazer requisições se referindo a esse compartilhamente utilizará o TID recebido anteriormente.

Process ID (PID): O PID é um número de 2 octetos que identifica que processo está fazendo a requisição no cliente. O servidor utiliza este número para checar problemas de concorrência (tipicamente para garantir que arquivos não serão corrompidos por processos concorrentes)

User ID (UID): O UID é um número de 2 octetos para identificar que usuário que está fazendo as requisições na máquina cliente. O cliente deve obter um UID do servidor enviando a este uma requisição de setup de sessão contendo o nome do usuário e sua senha. Passada a verificação de nome de usuário/senha o servidor responde a requisição incluíndo na resposta um UID gerado. O cliente então utiliza este UID em todas as futuras requisições. Se alguma das requisições dos clientes esbarrar em permissões de arquivos/ impressoras, o servido irá verificar se o UID da requisição possue as permissões necessárias para acessar tal recurso.

Multiplex ID (MID): O MID é um número de 2 octetos utilizados para gerenciar múltiplas requisições. Sempre que um cliente enviar um pacote o servidor checa o MID para ver se há alguma requisição pendente.

WordCount e parameter words: Os pacotes podem usar estes dois campos para armazenar dados específicos dos comandos. O template acima

não suporta todos os possíveis tipos de dados para um pacote SMB/CIFS. Para remediar este problema o campo parameter words foi criado com tamanho variável. O campo wordcaount especifica quantas palavras de 2 octectos o campo parameter words irá conter. Dessa forma cada pacote pode se ajustar ao tamanho necessário para transmitir dados de seu comando específico.

ByteCount e buffer: Estes campos são muito similares aos anteriores wordcount e parameter words. Eles irão armazenar uma quantidade de dados variável específica numa base por pacote. [BARREIROS, 2001]

2.5.5 WINBIND E O NAME SERVICE SWITCH

O PAM somente fornece parte da informação necessária para controlar os usuários em um sistema Linux. Além de permitir checar se um usuário entrou com a senha correta, um sistema Linux precisa outras informações, como o "ID" numérico do usuário, o diretório HOME, o shell padrão, etc. Esta informação, que normalmente é armazenada no arquivo /etc/passwd, pode ser determinada através de uma interface de sistema conhecida como NSS, ou Name Service Switch. [ALMEIDA, 2005]

A associação de UIDs de usuários de um domínio com usuários locais no Linux é feita pelo programa Winbind, que é um deamon que utiliza o mecanismo nsswitch (Name Service Switch) para obter outras fontes de dados de usuários e os associa nas ferramentas de gerenciamento de contas existentes no sistema. [SILVA, 2006]

O Winbind é composto por:

1) Servidor ("winbindd"): Se encarrega do contato com o servidor Windows por forma a obter informações sobre utilizadores e grupos lá existentes. Entre outras missões, este servidor mantém uma tabela de equivalências entre utilizadores do sistema MS-Windows e utilizadores Unix,

gerando os UIDs apropriados para cada SID correspondente a um utilizador Windows e mantendo as associações UID/SID entre sessões.

2) Modulo nss_winbind (/lib/libnss_winbind): Esta biblioteca NSS ("Name Service Switch") usa o servidor "winbind" e permite que os utilizadores do sistema Windows sejam válidos no sistema Unix. Para cada nome de utilizador obtém o respectivo UID/GID, "login shell", "home directory", etc.

3) Modulo pam_winbind (/lib/security/pam_winbind): Esta biblioteca PAM permite que os utilizadores do sistema Windows se autentiquem (Username/Password) perante o sistema Unix. Tal como o anterior usa o servidor "winbind" como intermediário.

O winbind funciona como um "gateway" de aplicação, recebendo pedidos do "nss_winbind" e do "pam_winbind" e usando a base de dados de utilizadores residente no sistema MS-Windows para proporcionar a informação que estes modulos necessitam.

Quando o winbind tem de resolver um nome, ele contata o servidor Windows e este lhe fornece o respectivo SID. Para obter o UID Unix equivalente o winbind verifica se o SID existe na sua base de dados, em caso afirmativo devolve o UID que lhe está associado.

Se o SID não está na base de dados, acrescenta-o, associando-lhe um novo UID que esteja livre no sistema Unix (o winbind permite que o administrador defina uma gama de UIDs para este efeito). Nesta altura o winbind pode, por opção do administrador, criar uma conta Unix local, nesse caso, para esse utilizador, o "pam_winbind" deixará de ser necessário, uma vez que o utilizador passa a ser local.

Seja ou não criada uma conta local para o utilizador, existe um conjunto de parâmetros Unix que o servidor "winbind" tem de fornecer conjuntamente com o UID recém definido ("primary group", "home directory", "login shell"). Estes parâmetros não existem na conta de utilizador Windows, por isso o

"winbind" utiliza "templates", basicamente trata-se de valores que serão iguais para todos os utilizadores, mas que eventualmente podem ter uma parte substituída por elementos variáveis, por exemplo, para a "home directory" poderá ser usado algo como:

/usr/homes/%U

Onde o elemento "%U" será depois substituído pelo nome do utilizador.

O winbind suporta não apenas nomes de utilizadores, mas também nomes de grupos, o princípio de funcionamento é semelhante como acontece para os utilizadores a possibilidade de criar grupos locais para corresponderem aos do sistema MS-Windows ou manter a utilização do "nss_winbind". Além de utilizadores e grupos, o winbind também resolve nomes de máquinas, utilizando o mecanismo NetBIOS, quer recorrendo a "queries broadcast", quer recorrendo a servidores WINS.

A grande vantagem do winbind é que o administrador apenas necessita de gerir utilizadores no sistema servidor Windows. Quando um novo utilizador Windows é criado, fica a ter automaticamente acesso ao sistema Unix, durante o primeiro acesso o winbind reserva um UID para esse novo utilizador e eventualmente cria mesmo uma conta local.

A forma como o winbind define os UIDs correspondentes aos SIDs dos novos utilizadores baseia-se num intervalo de valores estabelecido pelo administrador dentro do qual os UID vão sendo atribuídos à medida das necessidades. Isto significa que se existem vários sistemas Unix integrados num domínio Windows através do winbind, não há qualquer garantia de que um dado utilizador vai ter o mesmo UID em todos os sistemas. Sob o ponto de vista de administração esta situação não é nada desejável, especialmente se os UID's têm um caráter permanente (winbind configurado para criar contas locais).

A forma mais simples de resolver este problema consiste em optar pelo LDAP como base de dados para o Samba, nesse caso as equivalências SID/UID e SID/GID são armazenadas no servidor LDAP, se todos os servidores winbind utilizarem o mesmo servidor LDAP, então o problema desaparece. Quando um novo utilizador acede a um dos servidores Unix, será definido um UID e armazenado no servidor LDAP juntamente com o SID, ao efetuar o login em um segundo servidor, o SID já está na base de dados e por isso será usado o mesmo UID. [MOREIRA, 2002]

2.6 SERVIÇO DE DIRETÓRIO MICROSOFT

O serviço de diretório é um serviço de rede que identifica todos os recursos de uma rede e torna essa informação disponível para os usuários e aplicativos. Os serviços de diretório são importantes porque oferecem uma maneira consistente de nomear, descrever, localizar, acessar, gerenciar e tornar seguras as informação sobre esses recursos.

Quando um usuário pesquisa por uma pasta compartilhada na rede, é o serviço de diretório que identifica o recurso e fornece essa informação ao usuário.

Controladores de domínio armazenam dados e gerenciam a comunicação entre usuários e domínios, inclusive os processos de logon do usuário,autenticação e pesquisas de diretório. Quando o Active Directory em um computador que executa o Windows Server 2003, o computador se transforma em um controlador de domínio. [MICROSOFT, 2007]

O Active Directory é o serviço de diretórios do Windows Server 2003. Ele identifica todos os recursos disponíveis em uma rede, mantendo informações sobre estes dispositivos (contas de usuários, grupos, computadores, recursos, políticas de segurança, etc) em um banco de dados e torna estes recursos disponíveis para usuários e aplicações. [BATTISTI, 2003]

2.6.1 INTRODUÇÃO

O serviço de diretório do Active Directory pode ser instalado em servidores que executem o Microsoft Windows Server 2003, Standard Edition, Enterprise Edition e Datacenter Edition. Ele armazena informações sobre objetos na rede e facilita o acesso de administradores e usuários a essas informações. O Active Directory usa um armazenamento estruturado de dados como base para uma organização lógica e hierárquica das informações de diretório.

Esse armazenamento de dados, também conhecido como diretório, contém informações sobre os objetos do Active Directory. Geralmente, os objetos incluem recursos compartilhados como servidores, arquivos, impressoras e contas de usuário e de computador da rede.

A segurança é integrada ao Active Directory através da autenticação de logon e controle de acesso a objetos no diretório. Com um único logon na rede, os administradores podem gerenciar a organização e os dados de diretório em suas redes e os usuários de rede autorizados podem acessar recursos em qualquer lugar da rede. A administração com base em diretivas facilita o gerenciamento até mesmo das redes mais complexas.

O Active Directory também inclui, dentre outras funcionalidades, um conjunto de regras, o esquema, que define as classes de objetos e atributos contidos no diretório, as restrições e os limites das ocorrências desses objetos e o formato de seus nomes. [TECHNET, 2005]

2.6.2 SEGURANÇA NO AD

Os principais recursos do modelo de segurança da família Microsoft Windows Server 2003 são a autenticação do usuário e o controle de acesso. O serviço de diretório do Active Directory assegura que os administradores gerenciem esses recursos de maneira fácil e eficiente.

Na parte de autenticação, o Active Directory possui os seguintes recursos:

1) Logon interativo: Confirma a identificação do usuário para o computador local ou a conta do Active Directory.

2) Autenticação de Rede: Confirma a identificação do usuário para qualquer serviço de rede que o usuário esteja tentando acessar. Para fornecer esse tipo de autenticação, o sistema de segurança inclui, entre outros mecanismos, o Kerberos V5.

O Kerberos V5 é o principal protocolo de segurança para autenticação em um domínio. O protocolo Kerberos V5 verifica a identidade do usuário que solicita a autenticação assim como o servidor que fornece a autenticação solicitada. Essa verificação bidirecional é conhecida também como autenticação mútua.

O mecanismo de autenticação Kerberos V5 emite tíquetes para permitir o acesso aos serviços de rede. Esses tíquetes contêm dados criptografados, incluindo senha criptografada, que confirma a identidade do usuário para o serviço solicitado. Exceto pela inserção de uma senha, todo o processo de autenticação é invisível para o usuário.

Um serviço importante no Kerberos V5 é o Centro de Distribuição de Chaves (KDC). O KDC é executado em cada controlador de domínio como parte do serviço de diretório do Active Directory, que armazena todas as senhas de cliente e outras informações sobre contas.

O processo de autenticação Kerberos no Active Directory funciona da seguinte maneira:

1º) O usuário em um sistema cliente, com uma senha ou um cartão inteligente, fornece autenticação ao KDC.

2º) O KDC emite um tíquete de concessão de tíquete (TGT) especial para o cliente. O sistema cliente usa esse TGT para acessar o serviço de concessão de tíquete (TGS), que faz parte do mecanismo de autenticação Kerberos V5 no controlador de domínio.

3º) Em seguida, o TGS emite um tíquete de serviço para o cliente.

4º) O cliente apresentará esse tíquete de serviço ao serviço de rede que foi solicitado. O tíquete de serviço comprova a identidade do usuário para o serviço e a identidade do serviço para o usuário.

Todo controlador de domínio atua como um KDC. Um cliente usa uma pesquisa de sistema de nomes de domínios (DNS) para localizar o controlador de domínio disponível mais próximo. O controlador de domínio funcionará como o KDC preferencial para esse usuário durante a sessão de logon do usuário. Se o KDC preferencial ficar indisponível, o sistema localizará um KDC alternativo para fornecer autenticação. [TECHNET, 2005]

2.6.3 ESQUEMA

O esquema do Active Directory contém as definições para todos os objetos contidos no diretório. Todo novo objeto de diretório criado é validado de acordo com a definição de objeto apropriada no esquema antes de ser gravado no diretório. O esquema compõe-se de classes de objeto e atributos. O esquema base (ou padrão) contém um amplo conjunto de classes de objeto e atributos para atender às necessidades da maioria das organizações. Ele é modelado de acordo com o padrão X.500 da International Standards Organization (ISO) para serviços de diretório.

No esquema, uma classe de objeto representa uma categoria de objetos de diretório, como usuários, impressoras ou programas de aplicativo, que compartilham várias características comuns. A definição para cada classe de

objeto contém uma lista dos atributos do esquema que podem ser usados para descrever instâncias da classe.

Para melhorar o desempenho de operações do esquema (como a validação de novos objetos), cada controlador de domínio mantém uma cópia do esquema na memória (além da cópia que mantém em disco). A versão armazenada em cache é atualizada automaticamente (após um pequeno intervalo) toda vez que o esquema é atualizado. [TECHNET, 2005]

2.6.4 CLIENTES UNIX/POSIX *

A integração de estações de trabalho Unix/Posix com a plataforma de gerenciamento de identidades e acessos da Microsoft por meio do protocolo de autenticação Kerberos versão 5 envolve a transição de um ambiente NIS+ (Serviço de Informação de Rede) para outro baseado no serviço de diretório Microsoft® Active Directory®. Um ambiente baseado no Active Directory pode fornecer serviços de autenticação e autorização para as estações de trabalho Unix/Posix.

A figura 2.6.4.1 ilustra o diagrama da solução.

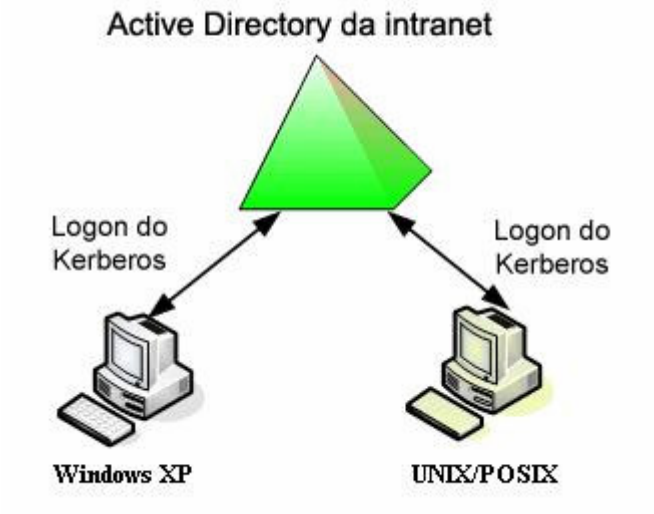

Figura 2.6.4.1 – Integração de Estação UNIX/POSIX com AD

(Fonte: http://www.microsoft.com/brasil/technet/security/topics/identitymanagement/idmanage/intr4-1.gif)

* **POSIX** (Portable Operating System Interface Unix), esse padrão foi desenvolvido pelo IEEE (Instituto de Engenheiros Elétrico Eletrônicos) para uniformizar as características dos sistemas baseados no Unix. O sistema GNU/LINUX usa o padrão POSIX.

Os usuários da estação de trabalho UNIX efetuam logon no shell da estação de trabalho UNIX usando o nome de usuário e as credenciais do Active Directory. A solicitação de logon passa do shell para o PAM (Pluggable Authentication Module) e, eventualmente, para a biblioteca Kerberos, que inicia uma troca de mensagens entre a estação de trabalho e o Windows KDC (vide figura 2.6.4.2). O resultado final dessa troca é um tíquete de serviço Kerberos, cuja biblioteca Kerberos da estação de trabalho UNIX pode descriptografar usando suas próprias chaves. [TECHNET, 2006]

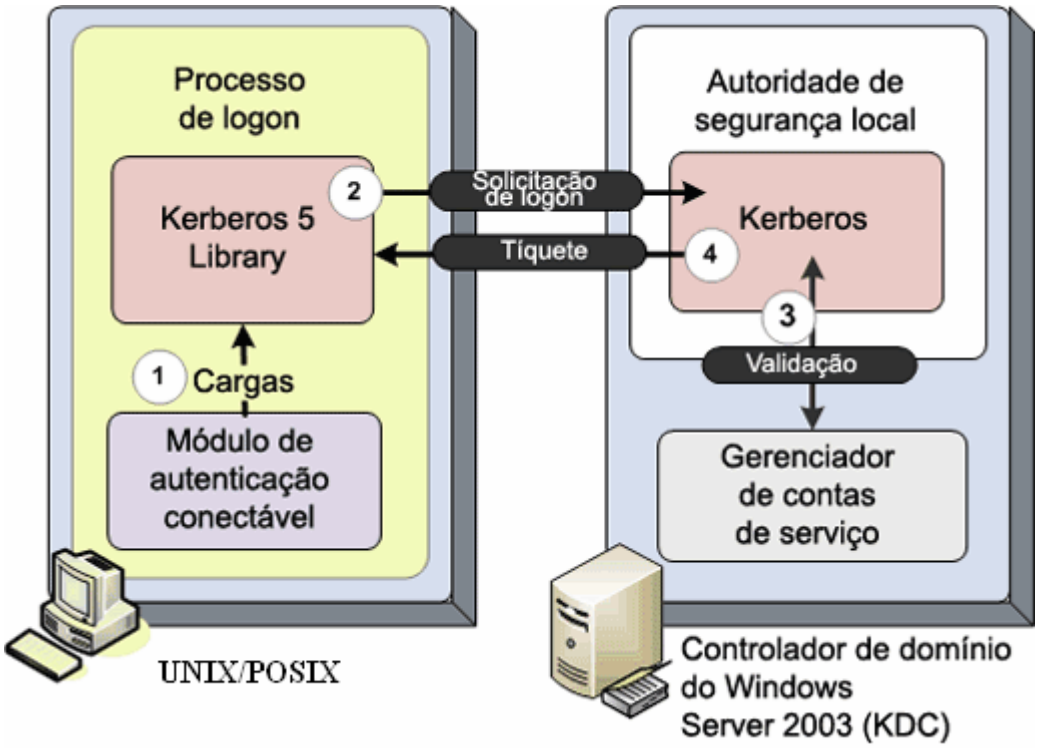

Figura 2.6.4.2 –Estação UNIX/POSIX usando Kerberos para autenticar no AD

(Fonte: http://www.microsoft.com/brasil/technet/security/topics/identitymanagement/idmanage/intr4-2.gif)

A biblioteca Kerberos do sistema UNIX descriptografa este o tíquete emitido pelo Domain Controller e extrai o UPN (nome principal do usuário) do usuário autenticado. Após a autenticação, o processo de logon usa as API (interfaces de programação de aplicativo) UNIX padrão para recuperar informações de autorização. Essas APIs são roteadas pelo NSS (name service switch), que depois usa o módulo nss_ldap para se conectar ao Active Directory e recuperar as informações solicitadas [como UID (Identificação de usuário) e GID (Identificação de grupo)]. Por fim, o processo de logon usa

essas informações para estabelecer o contexto de segurança do usuário e inicia o shell do logon usando tal contexto.

A biblioteca Kerberos também processará mensagens de erro e de diretiva do KDC que retransmitem requisitos de diretiva de conta, como aqueles que avisam aos usuários que eles devem alterar suas senhas. O processo de logon, pelo PAM, tomará as ações necessárias para atender as diretivas.

Como resultado da autenticação do KDC integrado ao Active Directory, a estação de trabalho (por meio do PAM e da biblioteca Kerberos) fará o cache dos tíquetes Kerberos para o usuário. O cache do tíquete é o que permite ao usuário acessar com segurança os aplicativos habilitados pelo Kerberos da rede e contar com a experiência SSO (logon único) ao mesmo tempo. [TECHNET, 2006]

2.7 O SISTEMA OPERACIONAL GNU/LINUX

O Linux é um sistema operacional que foi desenvolvido de acordo com a filosofia Unix e inspirado no Minix (Tanenbaum, 1987) e de acordo com as especificações do POSIX (Portable Operating System Interface Unix) da IEEE.

2.7.1 HISTÓRIA

O nome Linux surgiu da mistura de Linus + Unix. Linus é o nome do criador do Linux, Linus Torvalds. E Unix, é o nome de um sistema operacional de grande porte, no qual contaremos sua história agora, para que você entenda melhor a do Linux.

A origem do Unix tem ligação com o sistema operacional Multics, projetado na década de 1960. Esse projeto era realizado pelo Massachusets Institute of Technology (MIT), pela General Eletric (GE) e pelos laboratórios Bell (Bell Labs) e American Telephone na Telegraph (AT&T). A intenção era de que o Multics tivesse características de tempo compartilhado (vários usuários

compartilhando os recursos de um único computador), sendo assim o sistema mais arrojado da época. Em 1969, já existia uma versão do Multics rodando num computador GE645.

Ken Thompsom era um pesquisador do Multics e trabalhava na Bell Labs. No entanto, a empresa se retirou do projeto tempos depois, mas ele continuou seus estudos no sistema. Desde então, sua idéia não era continuar no Multics original e sim criar algo menor, mas que conservasse as idéias básicas do sistema. A partir daí, começa a saga do sistema Unix. Brian Kernighan, também pesquisador da Bell Labs, foi quem deu esse nome.

Em 1973, outro pesquisador da Bell Labs, Dennis Ritchie, rescreveu todo o sistema Unix numa linguagem de alto nível, chamada C, desenvolvida por ele mesmo. Por causa disso, o sistema passou a ter grande aceitação por usuários externos à Bell Labs.

Entre 1977 e 1981, a AT&T, alterou o Unix, fazendo algumas mudanças particulares e lançou o System III. Em 1983, após mais uma série de modificações, foi lançado o conhecido Unix System IV, que passou a ser vendido. Até hoje esse sistema é usado no mercado, tornando-se o padrão internacional do Unix. Esse sistema é comercializado por empresas como IBM, HP, Sun, etc. [ALECRIM, 2003]

O sistema operacional Linux teve o seu núcleo, mais conhecido como kernel, criado em 1991, pelo então pesquisador do Departamento de Ciências da Computação da Universidade de Helsinki (Finlândia) Linus Torvalds. O kernel é a mais baixa camada de interface entre o software e o hardware, responsável pelo gerenciamento de toda a operação de um computador.

A criação do sistema operacional veio no rastro do movimento do software livre, que ganhou impulso em 1984 com a fundação do Projeto GNU e a criação, em 1985, da primeira versão da licença GPL (General Public License), escrita por outro guru do mundo Linux, Richard Stallman. [ÂNGELO, 2007]

O nome Linux refere-se apenas ao kernel(núcleo) do sistema operacional, as distribuições que temos atualmente são variantes do projeto GNU (que incluem o kernel – Linux). Embora elas frequentemente sejam chamadas de “Linux”, elas seriam mais corretamente chamadas de sistemas GNU/Linux. [GNU, 1996]

O símbolo do sistema operacional GNU/LINUX é a figura de um pingüim chamado TUX, a sugestão desse símbolo foi dada por Linus Torvalds.

2.7.2 KERNEL

O Linux é um kernel monolítico com suporte a módulos carregáveis. Inclui capacidade de multitarefa, multiprocessamento, memória virtual por paginação, bibliotecas compartilhadas, copy-on-write.

Os drivers de dispositivos e extensões do núcleo correm tipicamente no espaço do kernel, juntamente com o restante do núcleo, com acesso total ao hardware. Diferentemente dos núcleos monolíticos tradicionais, os drivers de dispositivos podem ser configurados como módulos, o que permite que sejam carregados e descarregados enquanto o sistema corre. Também podem ser interrompidos (preempted) sob certas condições. Esta característica foi adicionada para lidar com interrupções de hardware corretamente e para melhorar o suporte ao multiprocessamento.

O facto do Linux não ser microkernel foi tema de uma famosa discussão entre Linus Torvalds e Andy Tanenbaum em um grupo de discussão. [BOVET e CESATI, 2005]

2.7.2.1 PROCESSOS

Um processo é uma instância de um programa em execução. Todo processo no Linux tem um pai (processo criador) e um número identificador (PID). O pai de todos os processos num ambiente Linux é o init, cujo PID é 1.

Este processo é criado pelo processo 0, que é um encadeamento (thread) do próprio kernel. O processo init irá permanecer em execução até o desligamento do sistema, e sua função é monitorar e criar os processos que implementam as camadas exteriores do sistema operacional.

Os processos são criados pelas chamadas de sistema fork() (processos tradicionais ou heavy weight) e clone() (processos leves ou light weight). Para otimizar a criação de processos tradicionais, o Linux usa o recurso de copy-on-write: quando um processo-filho é criado, ele compartilha as mesmas páginas de memória do pai. Quando um dos dois tenta escrever na memória, é gerada uma interrupção para o kernel, que então copia o conteúdo das páginas de memória para novas molduras de páginas, e estas são atribuídas ao processo que efetuou a escrita.

Para manter um ambiente multitarefa e possibilitar o multiprocessamento, o Linux mantém algumas estruturas importantes, das quais podemos citar duas: (i) o descritor do processo (task_struct), que contém todas as informações relativas ao processo; (ii) uma fila (runqueue) de processos por processador. Quando o sistema possui mais de um processador, o agendador do Linux faz o balanceamento de carga entre as filas. [BOVET e CESATI, 2005]

2.7.2.2 AGENDAMENTO

Para poder fazer um compartilhamento adequado de tempo do processador, o Linux usa duas classificações para avaliar qual a prioridade que um processo deve ter: (i) determina a responsividade do processo (tempo real, interativo, em segundo plano); (ii) verifica se o processo usa muito tempo de processador (CPU-bound) ou faz muitas operações de entrada e saída (I/O-bound).

Essas duas classes são razoavelmente independentes. Um processo pode executar em segundo plano (um daemon, por exemplo) e ser consumidor

de recursos de entrada e saída (um servidor de banco de dados) ou usar muito tempo de processador (um compilador). Um processo que executa em tempo real foi assim definido pelo seu programador, mas o agendador do Linux necessita fazer uma análise heurística para saber se um processo é interativo ou está executando em segundo plano.

O Linux utiliza um sistema de prioridades, onde um processo que possui prioridade maior tem precedência sobre um de prioridade menor para obter o processador. A identificação da prioridade de um processo pode ser estática ou dinâmica e varia de 1, a maior prioridade, a 139, a menor. Os números 1 a 99 são atribuídos a processos de tempo real e 100 a 139 são atribuídos a processos tradicionais (interativos e segundo plano).

Um processo em tempo real é classificado em FIFO (first-in, first-out) ou RR (Round-Robin) e somente será retirado do processador nos seguintes casos: (i) fim de execução; (ii) para ser substituído por um processo de maior prioridade; (iii) executar uma operação de bloqueio; (iv) espontaneamente; (v) é RR e esgotou seu quantum de processamento.

Um processo tradicional tem inicialmente atribuído uma prioridade estática (em geral 120) que determina o seu quantum de processamento, mas pode ter uma prioridade dinâmica, que é o valor analisado pelo agendador quando percorrer a lista de processos para determinar qual irá usar o processador. A prioridade dinâmica pode alterar o valor da prioridade estática em 5 pontos, para mais (penalidade) ou para menos (bônus), dependendo do passado do processo. O passado irá beneficiar o processo se o mesmo ficou muito tempo fora do processador (sleep time). Caso este tempo seja pequeno, o processo será penalizado. [BOVET e CESATI, 2005]

2.7.2.3 GERENCIAMENTO DE MEMÓRIA

O Linux utiliza memória virtual, que possui 3 funções básicas: (i) assegurar que cada aplicação (processo) tenha seu próprio espaço de

endereçamento, começando em zero (problema de realocação [1]); (ii) proteção de memória, para impedir que um processo utilize um endereço de memória que não lhe pertença; (iii) possibilitar que uma aplicação utilize mais memória do que a fisicamente existente.

Seu código é dividido em duas partes. Uma é dependente da arquitetura, onde são definidos o endereçamento – virtual e físico, o tamanho de página e o tratamento das tabelas de páginas. Na parte independente ficam o controle de alocação e liberação de memória e o esquema de substituição páginas.

O endereçamento virtual é dividido em espaço do usuário e espaço do kernel. O primeiro é privativo de cada processo, com início no endereço lógico zero e terminando no endereço determinado pela macro PAGE_OFFSET. O espaço do kernel é único e começa depois do espaço do usuário.

O código do kernel é carregado no início do espaço do kernel, sendo seguido pela área fisicamente mapeável (mem_map, estrutura que indexa as páginas físicas, e as páginas propriamente ditas).

Um endereço virtual no Linux, é dividido em 5 campos: diretório de páginas (PGD), diretório superior de páginas (PUD), diretório intermediário de páginas (PMD), tabela de páginas (PTE) e deslocamento (offset). A arquitetura x86 possui um espaço de endereçamento de 32 bits; quando são utilizadas páginas de 4 KiB (o padrão) o PUD e o PMD não são utilizados; o PGD e o PTE usam 10 bits cada, e o deslocamento usa 12 bits.

O esquema de substituição de páginas no Linux usa o algoritmo LRU (por aproximação) mantendo duas listas de envelhecimento (aging). A primeira, chamada active_list contém as páginas atualmente em uso e a segunda chamada inactive_list contém as candidatas a paginação (page out).

A paginação para disco pode ocorrer sob demanda, quando algum processo solicitar página e não houver alguma disponível. Neste caso, a página no final da lista inactive_list é liberada. Entretanto, existe um processo

chamado kswapd, inicializado pelo kernel, que verifica, periodicamente, o número de páginas livres. Caso este número seja menor que “pages_low”, “kswapd” é acordado para liberar páginas. Se o valor chegar a “pages_min”, “kswapd” entra num regime síncrono para agilizar a liberação. Quando o valor de páginas livres atingir “pages_high”, “kswapd” vai dormir. [BOVET e CESATI, 2005]

2.7.2.4 KERNEL PANIC (PÂNICO DO NÚCLEO)

Um pânico é um erro de sistema não-recuperável detectado pelo kernel, ao contrário dos erros de impressão e utilização por código do espaço de utilizador. É possível que o código do núcleo indique essa condição ao invocar a função panic localizada no cabeçalho sys/system.h. No entanto, grande parte dos pânicos são o resultado de exceções do processador não-lidadas no código do kernel, tal como referências a moradas de memória inválidas. [BOVET e CESATI, 2005]

2.7.3 X WINDOW

É um sistema gráfico de janelas que roda em uma grande faixa de computadores, máquinas gráficas e diferentes tipos de máquinas e plataformas Unix. Pode tanto ser executado em máquinas locais como remotas através de conexão em rede.

Em geral o ambiente gráfico X Window é dividido da seguinte forma:

1) Servidor X: É o programa que controla a exibição dos gráficos na tela, mouse e teclado. Ele se comunica com os programas cliente através de diversos métodos de comunicação.

O servidor X pode ser executado na mesma máquina que o programa cliente esta sendo executado de forma transparente ou através de uma máquina remota na rede.

2) O gerenciador de Janelas: É o programa que controla a aparência da aplicação. Os gerenciadores de janelas (window managers) são programas que atuam entre o servidor X e a aplicação. Você pode alternar de um gerenciador para outro sem fechar seus aplicativos.

Existem vários tipos de gerenciadores de janelas disponíveis no mercado entre os mais conhecidos estão o Gnome e KDE.

3) A aplicação cliente: É o programa sendo executado.

Esta organização do ambiente gráfico X traz grandes vantagens de gerenciamento e recursos no ambiente gráfico UNIX, uma vez que tem estes recursos você pode executar seus programas em computadores remotos, mudar totalmente a aparência de um programa sem ter que fechá-lo através da mudança do gerenciador de janelas. [SILVA, 2006]

Abaixo serão descritos as características dos gerenciadores de janela GNOME e KDE, uma vez que o software LeasyConfig foi compilado nos dois ambientes.

2.7.3.1 GNOME

O projeto nasceu em Agosto de 1997 por Miguel de Icaza com o esforço de criar um ambiente desktop inteiramente livre para sistemas livres.

No começo, o objetivo principal do GNOME foi fornecer uma suíte amigável ao usuário de aplicações e um desktop fácil de usar.

O GNOME foi escrito originalmente na linguagem de programação C. Logo depois um grande número de linguagens foram se incorporando ao GNOME e suas aplicações, por exemplo, linguagens como: C++, Ruby, Python, Perl e muitas outras. No lugar do Qt, o GTK foi escolhido como a base para desenvolvimento do GNOME. A licença é a GNU General Public License (GPL).

O Projeto GNOME provê duas coisas: o ambiente desktop GNOME, intuitivo e atraente para usuários finais, e a plataforma de desenvolvimento GNOME, um framework extenso para construção de aplicações que se integrem com todo o desktop.

O GNOME é Software Livre e parte do projeto GNU, que se dedica a dar a usuários e desenvolvedores controle sobre seus desktops, software e dados. Isso consiste, basicamente, em garantir a todos o direito a quatro liberdades básicas: a de usar o software para qualquer fim, estudar o seu código-fonte, modificá-lo para qualquer necessidade que você possa ter e redistribui-lo, modificado ou não.

Parte da filosofia por trás do Software Livre é garantir essa liberdade para todos, sem exceção, o que inclui usuários e desenvolvedores com deficiências. Graças a vários anos de esforço contínuo, o GNOME é, hoje, o ambiente desktop mais acessível em qualquer plataforma Unix.

O GNOME possui atualmente tradução para 43 línguas que, em número de falantes, correspondem a aproximadamente 70% da população mundial.

Além da comunidade internacional do GNOME, o projeto tem o apoio de empresas líderes em Unix e Linux, incluindo Hewlett-Packard, Mandriva, Novell, Red Hat e Sun. [GNOME, 2005]

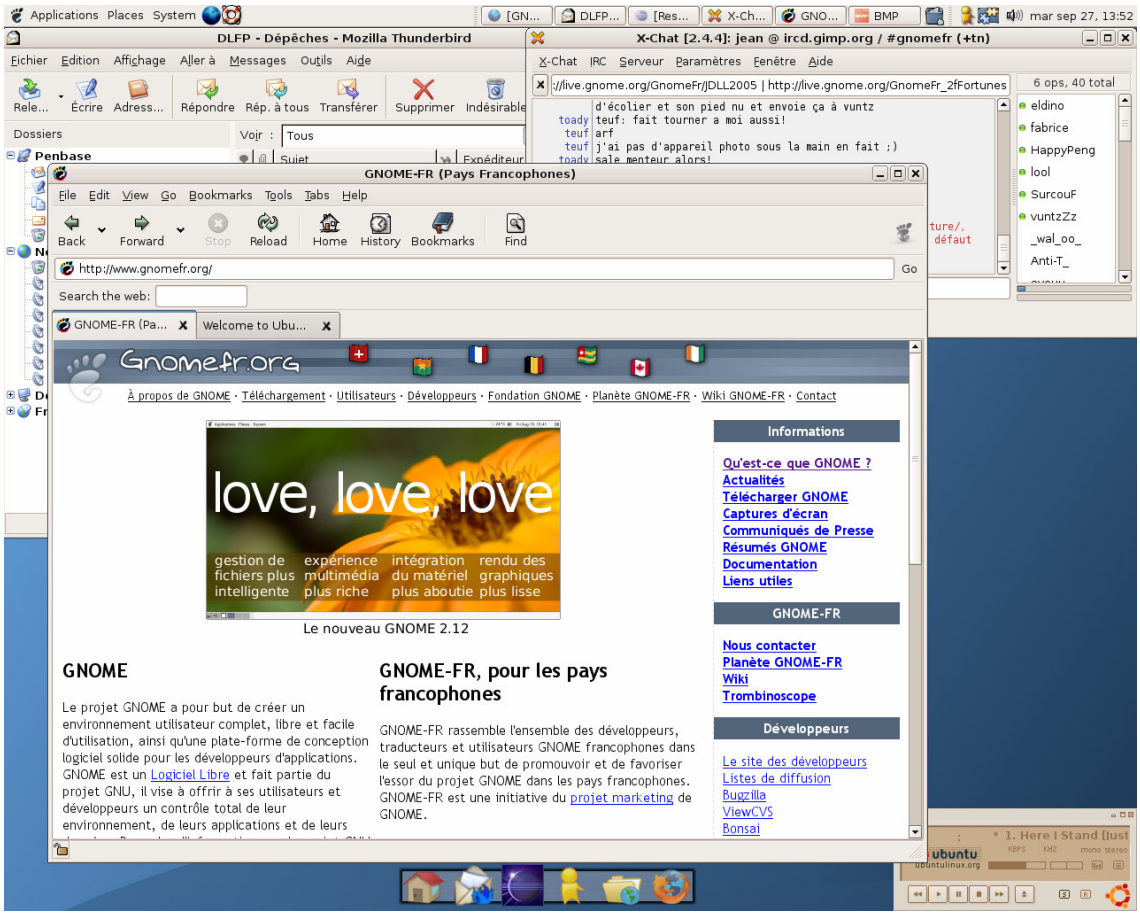

Figura 2.7.3.1.1 – Screenshot do GNOME

(Fonte: http://br.gnome.org/capturas/2.12/gnome-2.12-jean2.png em 12/05/2007)

2.7.3.2 KDE

Software de origem alemã, o KDE (sigla inglesa para K Desktop Environment) é, simultaneamente, um ambiente gráfico (que inclui um gerenciador de janelas) e uma plataforma de desenvolvimento livre e de código aberto, desenvolvido com base na biblioteca Qt escrita em C++. Voltado inicialmente aos utilizadores de platafomas Unix, funciona também no Mac OS X utilizando o seu servidor X11 e no Windows através do ambiente Cygwin.

O KDE é feito com base na biblioteca Qt, de propriedade da empresa Trolltech, enquanto que o GNOME, outro ambiente gráfico para Unix, é baseado na biblioteca GTK, do Projeto GNU. A biblioteca Qt é gratuita para aplicações Open Source, e as bibliotecas do KDE são licenciadas como LGPL

ou BSD-compatível. O licenciamento da Qt prevê que quem desenvolver aplicações de código fechado com Qt seja obrigado a adquirir uma licença.

O KDE procura preencher as necessidades por um uso facilitado do desktop em estações de trabalho UNIX, similar a ambientes de desktop encontrados em MacOS ou Microsoft Windows.

O Projeto KDE desenvolveu um framework de desenvolvimento de aplicações de primeiro nível, implementando os últimos avanços em frameworks e posicionando-se em competição direta a frameworks populares, como exemplo, as tecnologias Microsoft MFC/COM/ActiveX. A tecnologia KParts de componentes permite a desenvolvedores de rapidamente criarem aplicação de alto nível implementando tecnologia de ponta. [KDE, 2007]

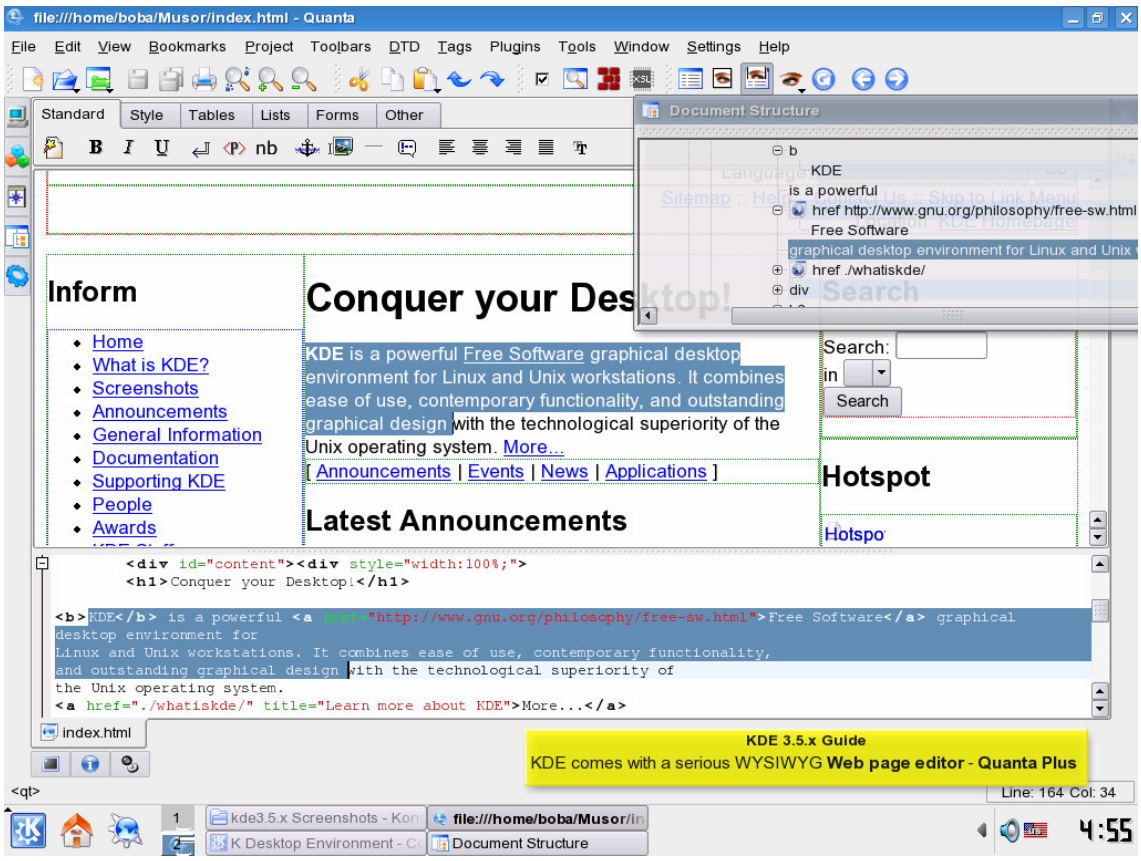

Figura 2.7.3.2.1 – Screenshot do KDE

(Fonte: http://www.kde.org/screenshots/images/3.5/32-htmleditor.png em 12/05/2007)

CAPÍTULO 3 - AMBIENTE PROPOSTO E IMPLEMENTAÇÃO

Nesse capítulo serão descritos os procedimentos realizados para obtenção da solução proposta, bem como a apresentação do software LeasyConfig que foi desenvolvido como parte da mesma.

3.1 TOPOLOGIA

Um dos grandes problemas em uma rede consiste na falta de um serviço de diretório que integre usuários e serviços independente da plataforma utilizada e sem a replicação de bases de informações, o que acabaria subutilizando algumas funções de segurança da solução.

Para a utilização de estações de trabalho com softwares livres, comumente são sugeridos modelos onde são migrados apenas os servidores que realizam a autenticação(Figura 3.1.1).

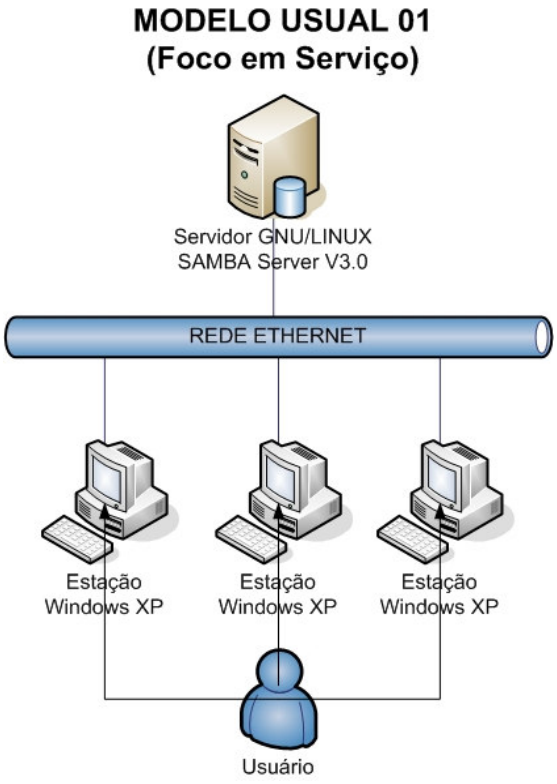

Figura 3.1.1 – Modelo Usual 01

Essa solução possui foco em serviço, uma vez que os usuários não serão afetados com a migração dos servidores, pois os serviços continuarão a ser disponibilizados de forma transparente.

Contudo, o custo dessa solução não é significativamente reduzido, haja vista o custo das estações de trabalho ser mantido após a implementação.

Também são sugeridos modelos onde há replicação da base de informações do domínio (Figura 3.1.2).

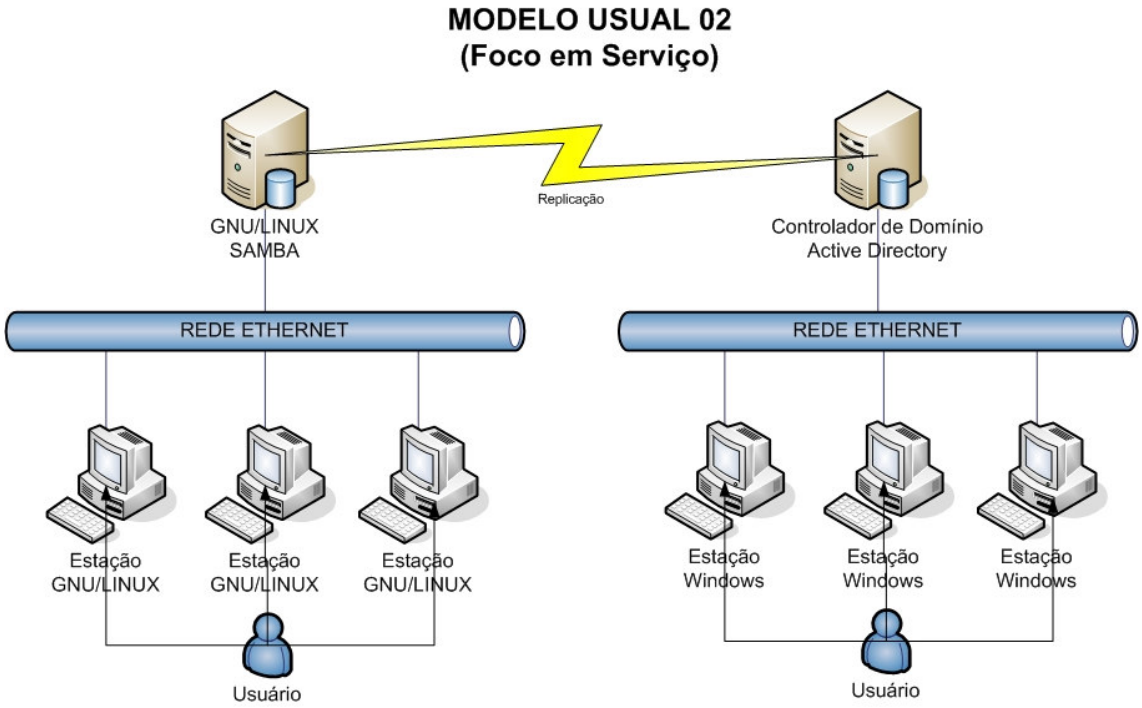

Figura 3.1.2 – Modelo Usual 02

Essa solução também possui foco em serviço, contudo, já propicia uma redução maior no custo da solução uma vez que poderão ser utilizadas estações de trabalho com software livre.

O problema é que nessa solução a base de usuários é replicada para outro servidor, limitando algumas funções de segurança e descentralizando a administração da rede.

O modelo sugerido nesse projeto possui foco em custo, tendo em vista que as estações de trabalho serão migradas para software livre e continuarão se autenticando no controlador de domínio existente, sem necessidade de replicação da base de informações (Figura 3.1.3).

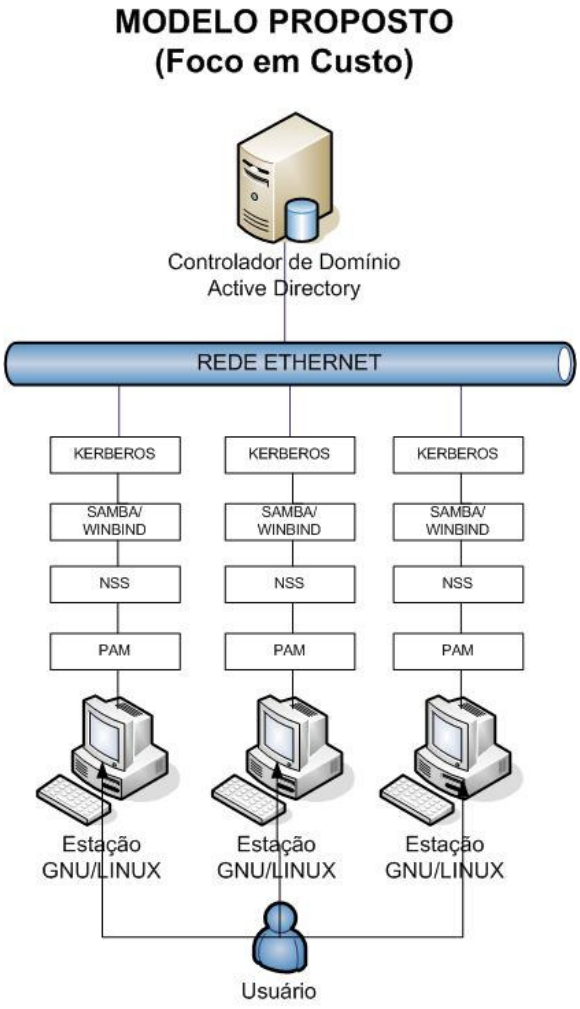

Figura 3.1.3 – Modelo Proposto

No cenário estudado, apesar do modelo proposto possuir o foco em custo, o mesmo também trouxe a transparência dos sistemas internos, uma vez que os mesmos são disponibilizados através de uma intranet via navegador de internet. Quanto aos serviços de correio e arquivos, a nova suíte de escritório das estações foi totalmente compatível.

Um quarto modelo, com servidores e estações rodando o Sistema Operacional GNU/LINUX, não será abordado uma vez que o projeto trata de

um processo de migração em um ambiente onde já existam softwares proprietários. Ou seja, um modelo com essas características foge do escopo do projeto definido anteriormente.

A empresa onde foi realizado o estudo possui quadro de pessoal técnico já treinado para gerenciar o ambiente servidor através da plataforma Microsoft, bem como possui compradas as licenças desses servidores, sendo assim, um modelo 100% em software livre não usufrui do investimento já realizado e ainda gera um custo extra com treinamento de pessoal. Esse modelo seria recomendado em caso de uma implantação nova e não em uma migração.

3.2 O LEASYCONFIG

Para viabilizar a adoção do modelo proposto é necessária a inclusão das estações de trabalho com software livre ao domínio Microsoft existente, essa tarefa não é trivial haja vista a necessidade de alterar vários parâmetros em diversos arquivos de configuração. Esse fato acaba dificultando a adoção da solução uma vez que grande parte dos técnicos não estão habituados a realizar esse tipo de tarefa.

Para solucionar o problema, foi desenvolvido um software de configuração com interface amigável para que esses técnicos possam, de maneira rápida e fácil, adicionar as estações com software livre ao domínio Microsoft.

O software desenvolvido foi denominado LeasyConfig – Linux Easy Config. Como o LeasyConfig faz parte desse projeto acadêmico, todos os códigos-fonte serão disponibilizados nos apêndices desta monografia e o uso, modificação/aperfeiçoamento e distribuição poderão ser feitos sem necessidade de licenciamento e/ou compra no intuito de disseminar o uso do modelo proposto no ambiente corporativo.

Para o criação do LeasyConfig foi utilizado um ambiente de desenvolvimento chamado Lazarus (Figura 3.2.1) que utiliza um compilador Free Pascal.

Quando surgiu a necessidade de desenvolvimento do LeasyConfig, foram testadas algumas linguagens e IDE's para serem usadas no projeto.

A linguagem C era a mais indicada, contudo, das IDE's testadas com essa linguagem no ambiente GNU/LINUX, nenhuma ofereceu os recursos que o Lazarus com linguagem Free Pascal possuía.

Sendo assim, apesar da linguagem Free Pascal não ser a mais utilizada nesse ambiente, ela atendia aos requisitos necessários para o desenvolvimento da ferramenta e por isso foi escolhida.

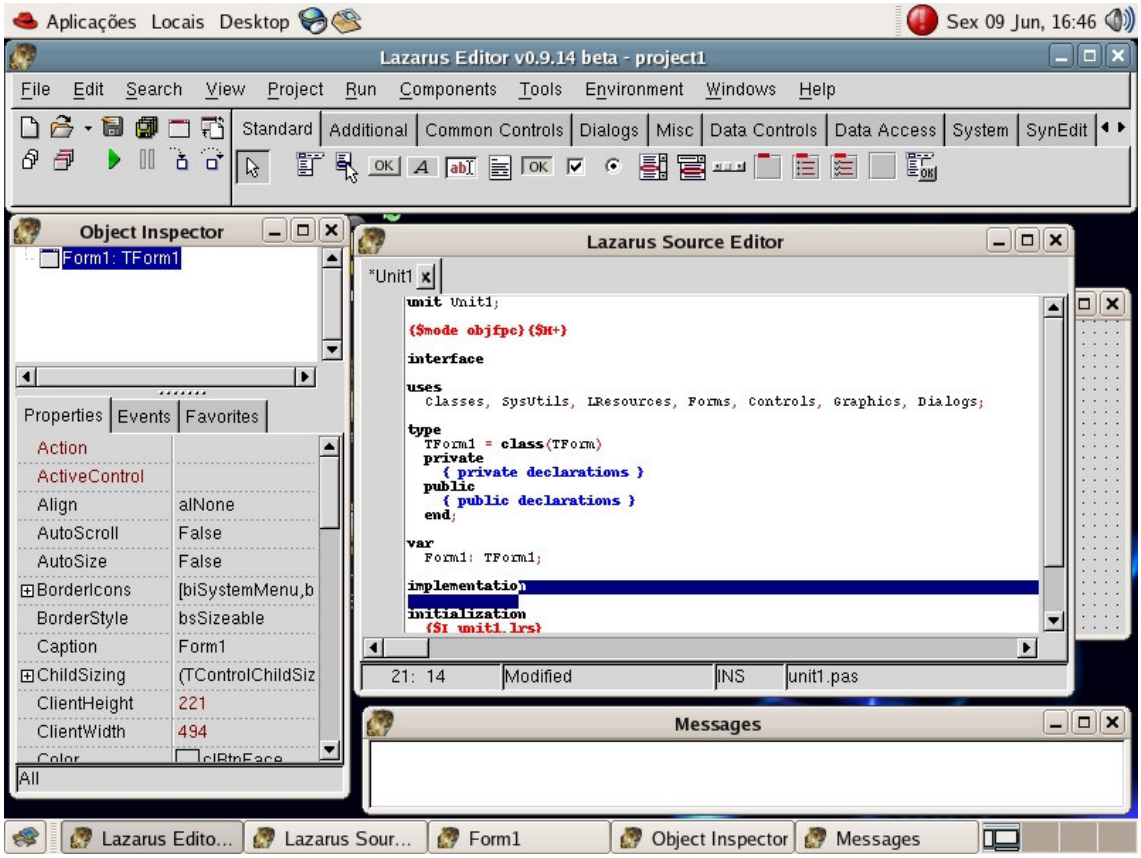

Figura 3.2.1 – IDE Lazarus
(www.lazarus.freepascal.org)

O Lazarus é um projeto licenciado sob a GPL que visa criar um ambiente de desenvolvimento clone do Delphi com o Free Pascal como compilador. Ele procura manter compatibilidade com o Delphi quando possível, mas roda em Linux e Windows e está num estágio bastante avançado em seu desenvolvimento. O lazarus é baseado na sua própria biblioteca de componentes visuais e não-visuais: a LCL (Lazarus Component Library). [LAZARUS, 2007]

3.2.1 INTERFACE E FUNCIONALIDADES

O LeasyConfig possui interface amigável, sendo que isso era uma premissa para seu desenvolvimento; ele possui 2 guias, básico e avançado.

A guia “Básico” coleta algumas informações que são necessárias para ingressar a estação ao domínio Microsoft e o software não executa a configuração caso os campos obrigatórios presentes nessa guia não sejam preenchidos, contudo, o campo “Descrição do Computador” poderá deixar de ser preenchido.

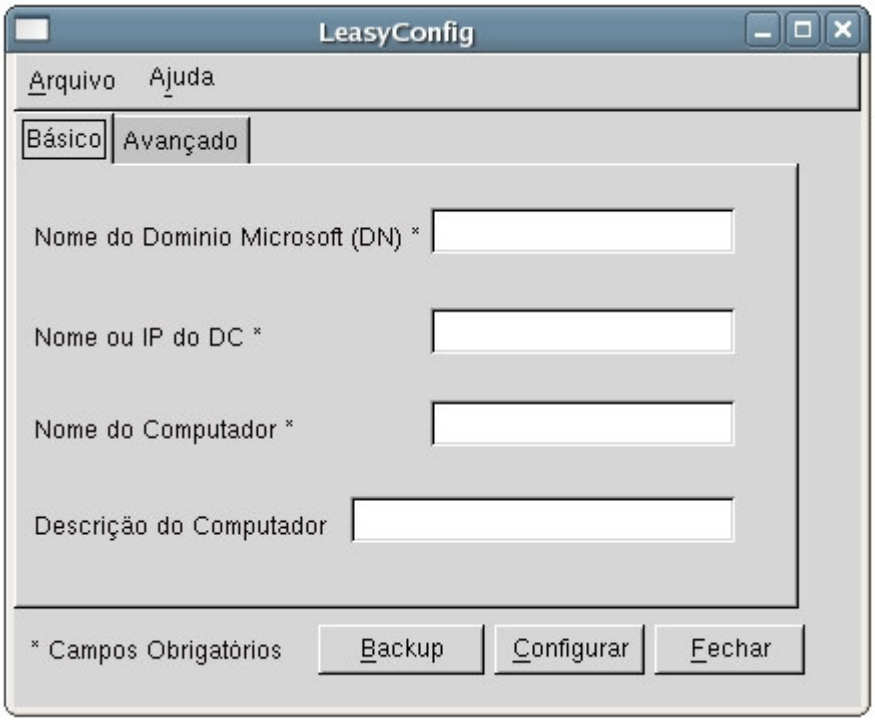

Figura 3.2.1.1 – Guia Básico do LeasyConfig

A guia “Avançado” traz a funcionalidade de personalizar itens de segurança (Kerberos Ticket Lifetime e Template Shell) e personalização (Winbind Separator).

Os campos da guia “Avançado” não têm a obrigatoriedade de serem preenchidos, uma vez que na ausência de informações para esses campos, o Leasyconfig efetua a configuração da estação GNU/LINUX com valores padrões.

A Figura 3.2.1.2 traz um screenshot da guia “Avançado”.

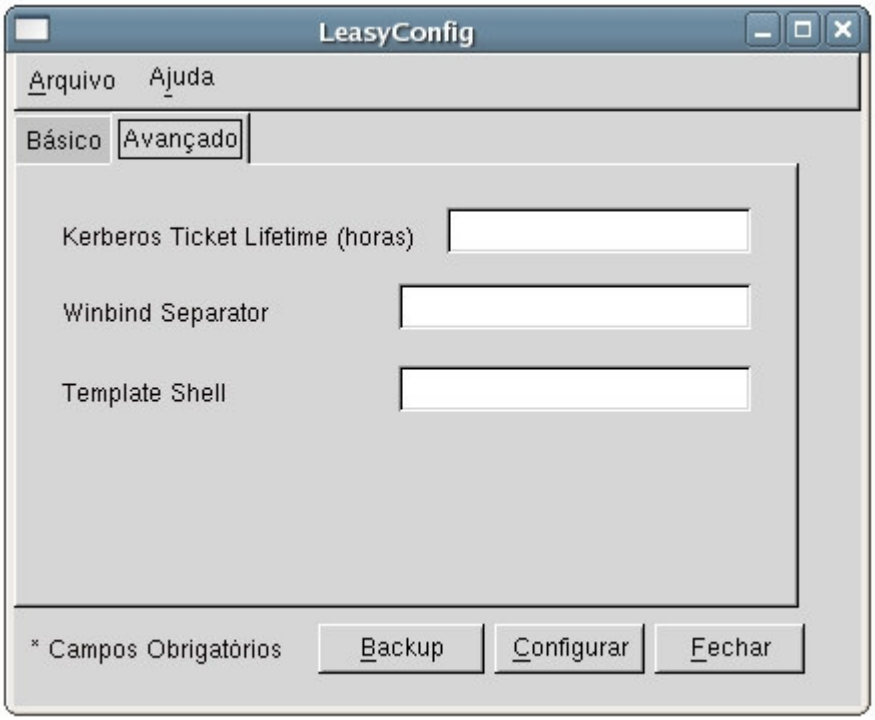

Figura 3.2.1.2 – Guia Avançado do LeasyConfig

Além de efetuar a configuração da estação GNU/LINUX com as informações coletadas nas guias “Básico” e “Avançado”, o Leasyconfig pode efetuar um backup da configuração atual da estação para uma futura necessidade de rollback.

Após o backup, o Leasyconfig informa através de uma janela de informações simples.

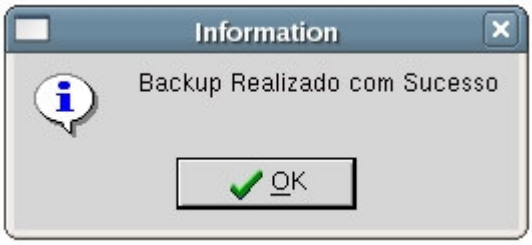

Figura 3.2.1.3 – Informação sobre o Backup do LeasyConfig

Após a configuração, também é informado que a estação está configurada.

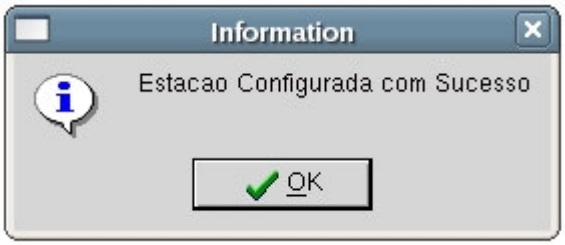

Figura 3.2.1.4 – Informação sobre a Configuração

3.3 O AMBIENTE DE ESTUDO

O ambiente utilizado para implementação do projeto possui 2 máquinas físicas e 2 virtuais.

Para uso das máquinas virtuais foi utilizado um software de virtualização chamado VmWare Server (Figura 3.3.1). A versão Server 1.0.2 do VmWare é gratuita e pode ser baixada do site do fabricante.

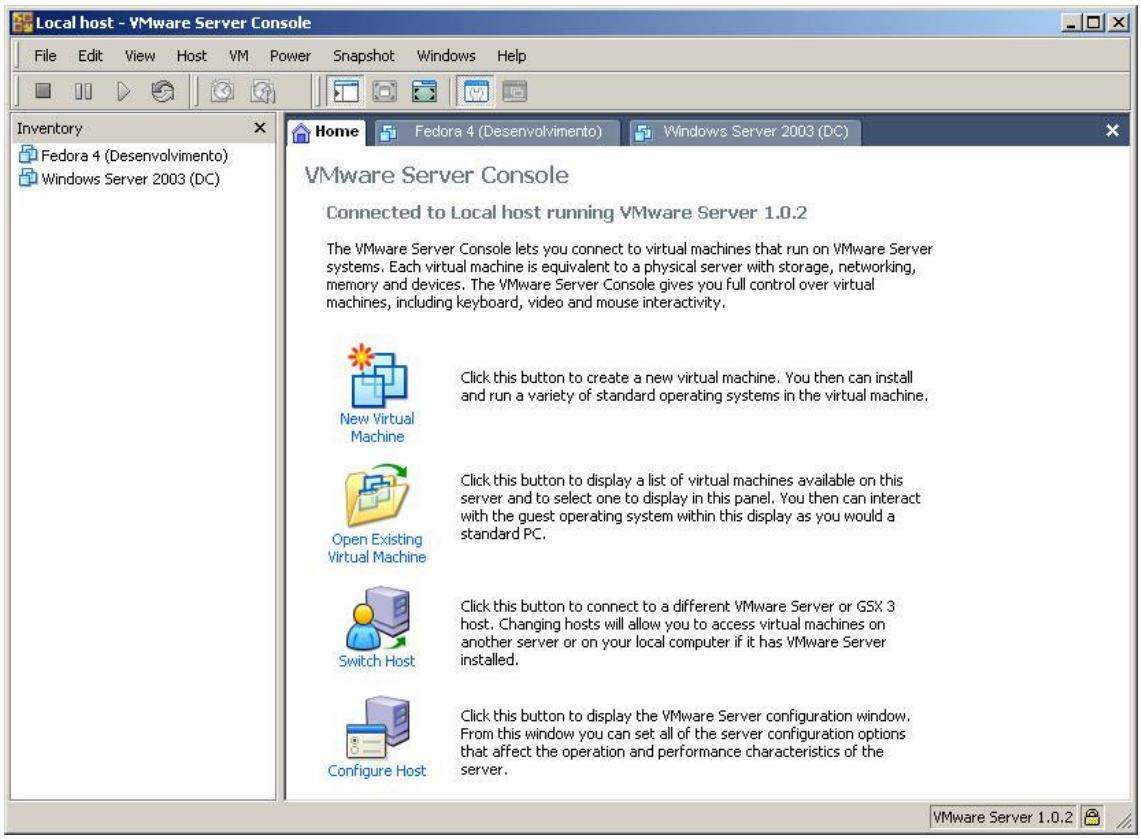

Figura 3.3.1 – Console do software VmWare Server.

Através da console do VmWare foram criadas 1 máquina virtual com sistema operacional Microsoft Windows Server 2003 Standard Edition para ser usado como controlador de domínio, bem como 1 máquina virtual com o sistema operacional Fedora Core 4 para desenvolvimento do software LeasyConfig.

As máquinas físicas utilizadas possuem a seguinte especificação:

1) Desktop com processador Intel Pentium IV 2.4GHz, 1Gb de memória DDR-333, 2 HD's de 80Gb, interface de rede 10/100Mbps com sistema operacional Microsoft Windows XP

2) Notebook com processador Intel Pentium III 650MHz, 192Mb de memória RAM, 1 HD de 12Gb, interface de rede 10/100Mbps com sistema operacional GNU/Linux Fedora Core 4.

O Ambiente estará disposto como demonstrado na figura 3.2.2 abaixo:

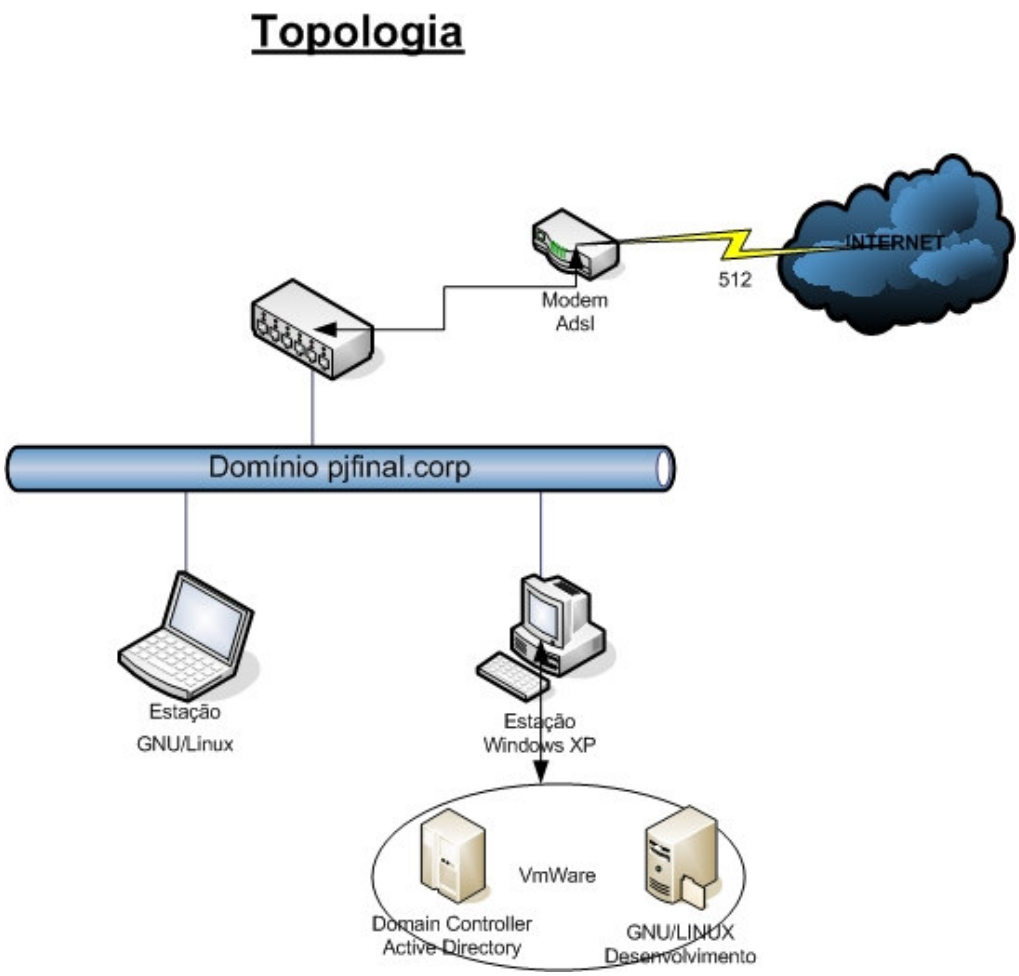

Figura 3.3.2 – Topologia do ambiente de testes.

3.3.1 CONFIGURANDO O CONTROLADOR DE DOMÍNIO MICROSOFT

Para configurar o Windows Server 2003 Standard Edition de modo que pudesse simular o controlador de domínio real da empresa, seguimos os seguintes procedimentos:

Após a instalação do sistema operacional com as opções padrões sugeridas, executamos o comando *dcpromo*, que promove o servidor a controlador do domínio (DC) com a instalação do Active Directory.

Na tela de configuração do tipo de domínio selecionado (Domain Controller Type) há a opção Domain Controller for a new Domain que tem a função de criar um novo controlador de domínio, esta opção é utilizada quando

não há nenhuma máquina executando esta função pois essa máquina também será responsável pela administração e por permitir ou não a adição de um controlador dentro do domínio.

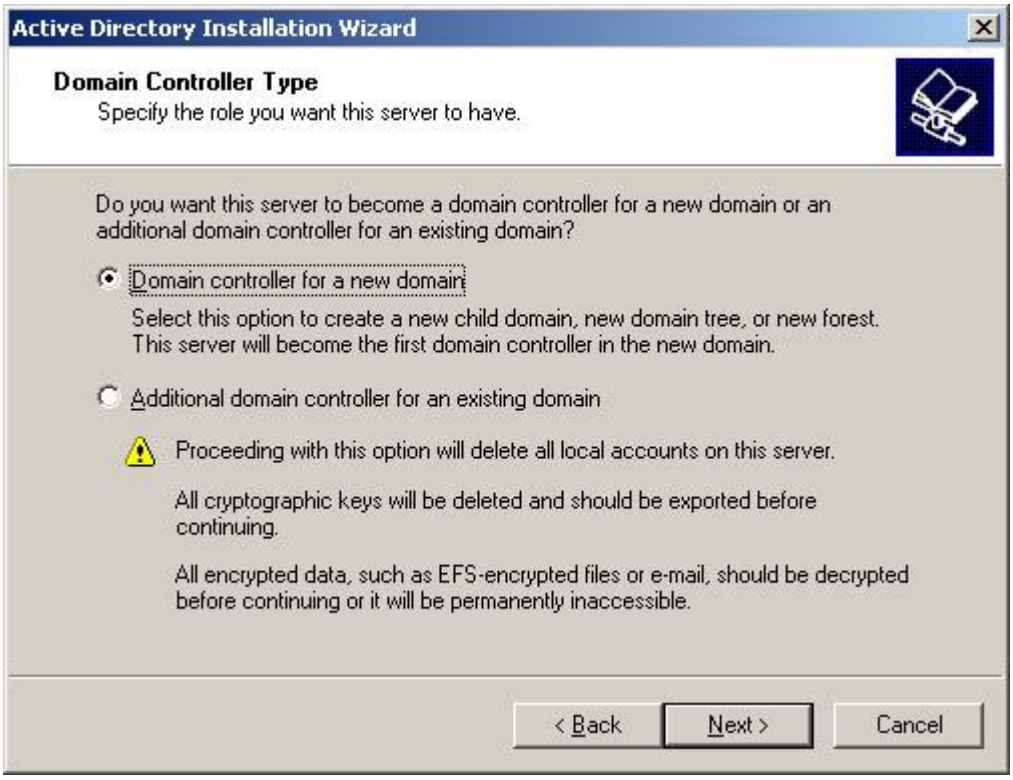

Figura 3.3.1.1 – Instalando um novo domínio.

A próxima opção trata-se da criação de uma floresta, que seria o conjunto de árvores do domínio, com essa configuração a máquina se torna responsável pela floresta permitindo ou não a criação de novas árvores dentro do domínio. Esse servidor passa então a ser o controlador da floresta e gerenciador de domínios.

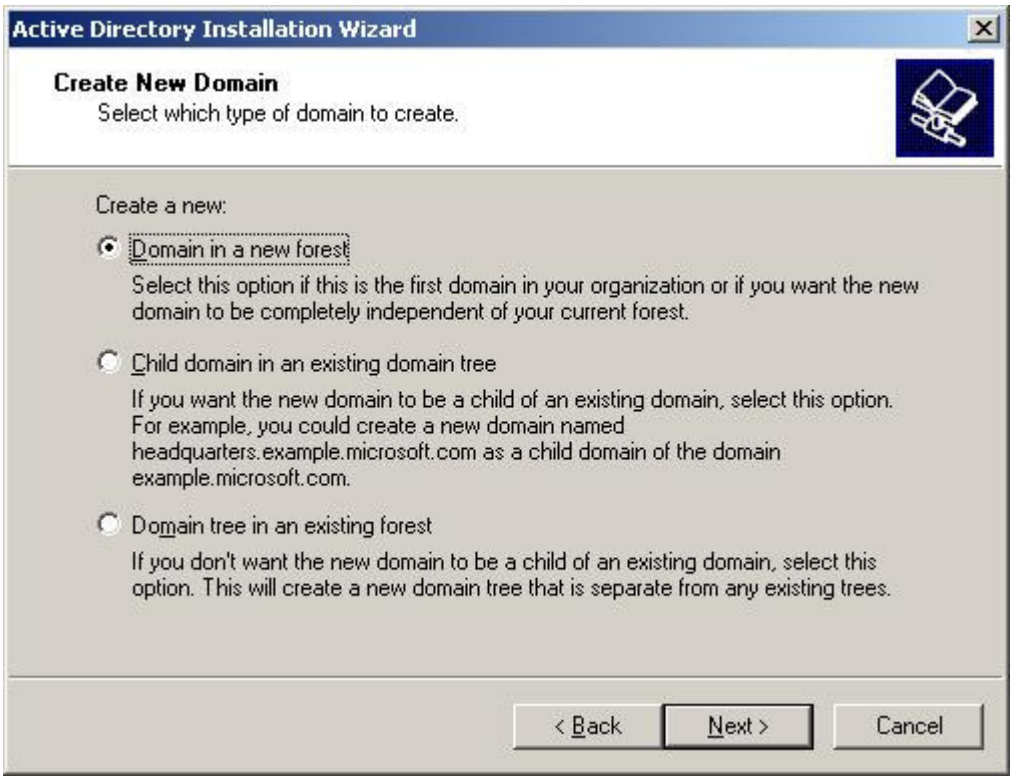

Figura 3.3.1.2 – Criando uma nova floresta.

Em seguida, temos que dar um nome para o domínio que está sendo criado, o Full DNS name, nesse caso o nome dado foi **pjfinal.corp** conforme ilustrado na figura abaixo.

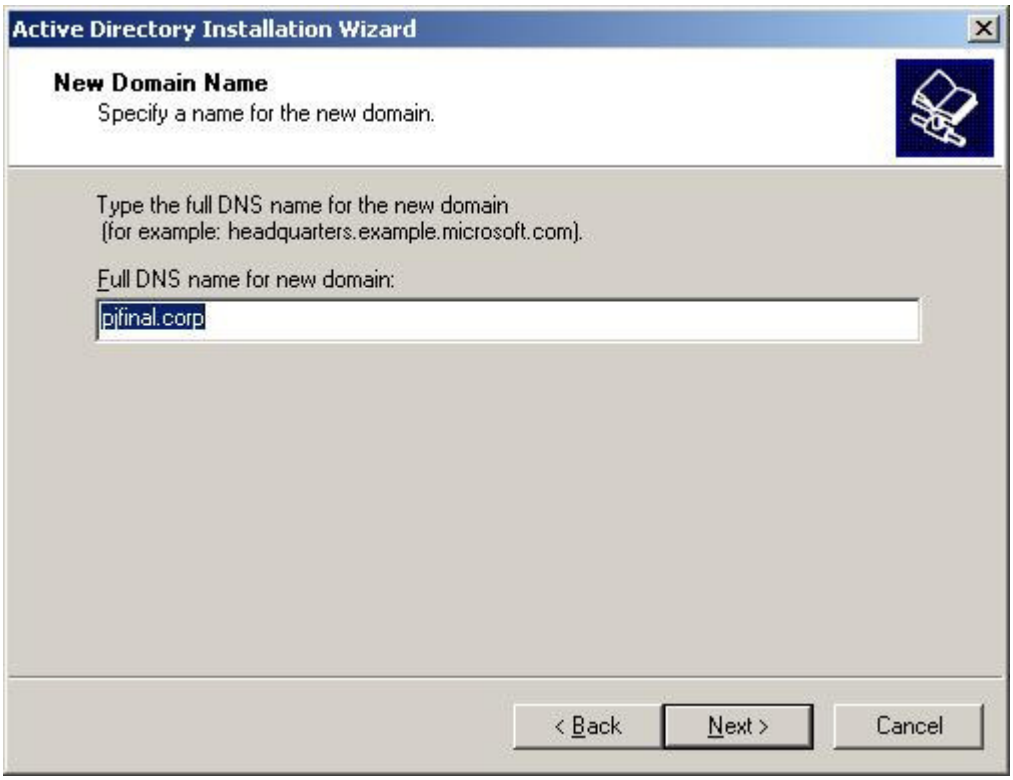

Figura 3.3.1.3 – Nome do Domínio.

Por conseguinte, informamos o nome NetBIOS para o domínio, esse nome poderá ser usado em sistemas operacionais Microsoft anteriores à versão 2000. O nome escolhido foi **pjfinal**.

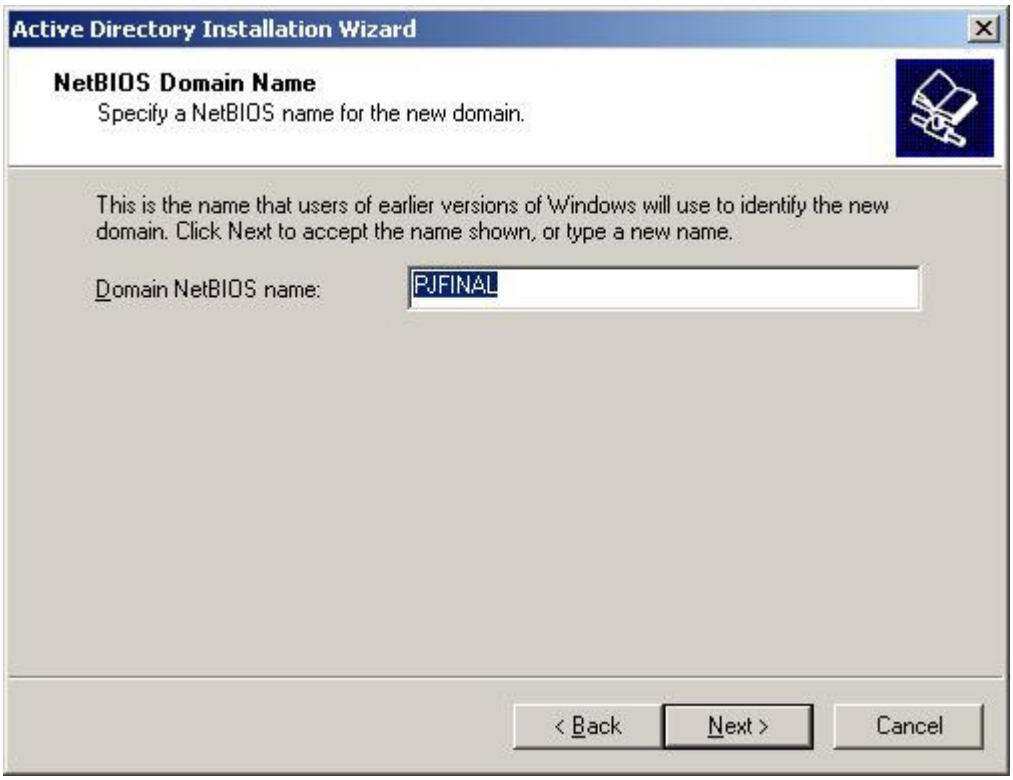

Figura 3.3.1.4 – Nome NetBIOS.

Após a escolha do nome NetBIOS, informamos em que local ficará a base do Active Directory e os arquivos de log. A opção escolhida foi a padrão.

Também escolhemos onde será instalada a pasta SYSVOL, que é uma pasta muito importante para a replicação do Active Directory e para a distribuição de políticas de segurança (Group Policies). Deve-se colocar a pasta SYSVOL em um volume formatado com NTFS.

Depois é informado que esse servidor terá um DNS para resolver os nomes internos do domínio conforme ilustrado na figura 3.3.1.5.

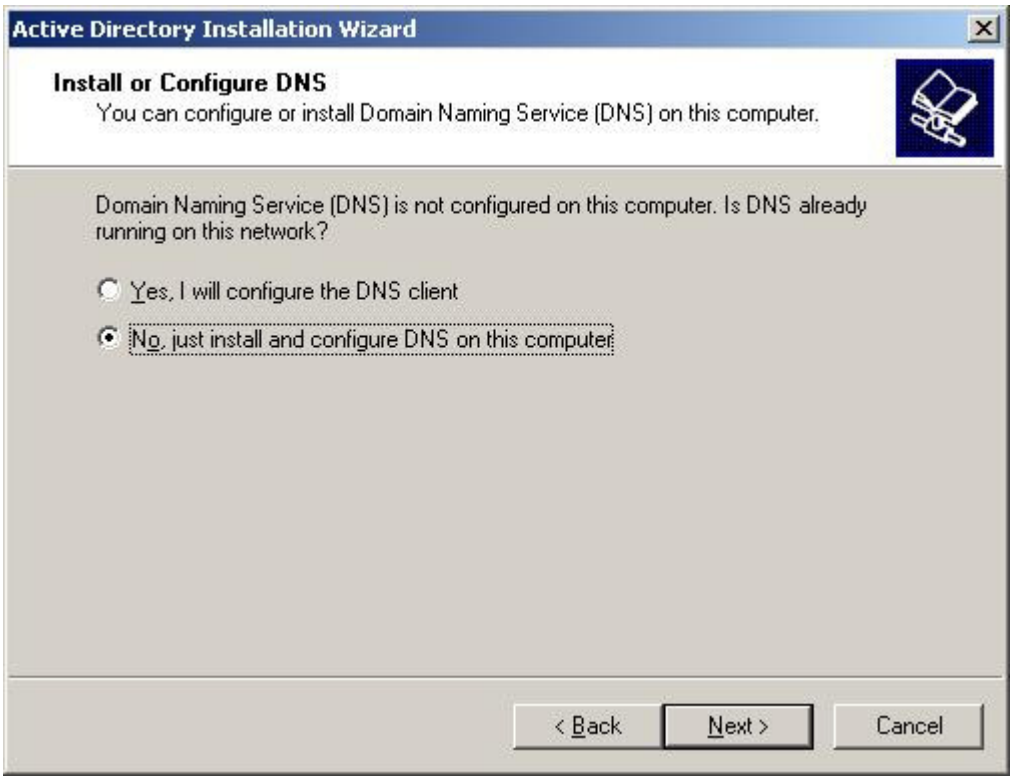

Figura 3.3.1.5 – Instalação do DNS interno.

Após coletadas as informações necessárias, é mostrado um relatório (summary) onde pode ser conferidas as opções marcadas. Confirmando essas informações o Active Directory será instalado no servidor.

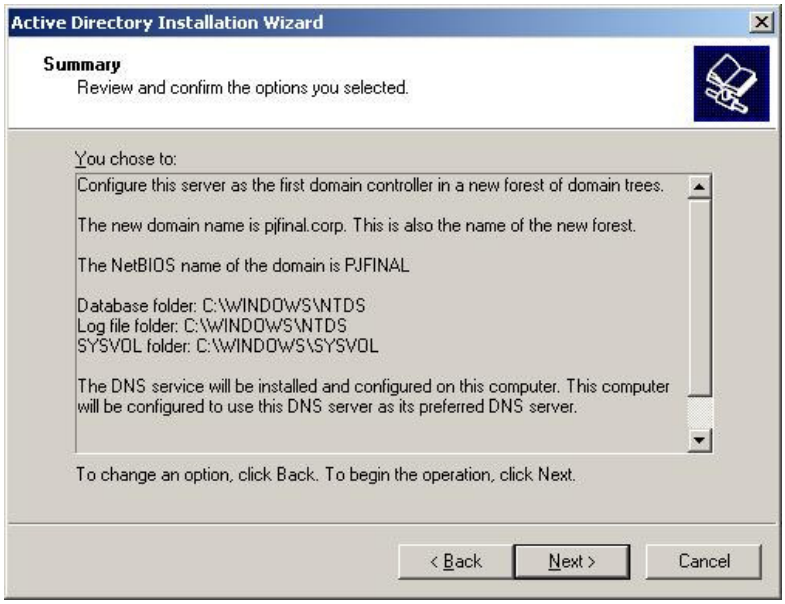

Figura 3.3.1.6 – Relatório da Configuração

Após a instalação do AD, é solicitado um reboot no servidor. Quando ele é reiniciado, a base de usuários do Windows Server 2003 deixa de ser a SAM e passa a ser o próprio Active Directory.

Logando com o usuário administrator, que agora é um Administrador do Domínio, abrimos a console de usuários e computadores do AD para criação dos usuários que serão usados nos testes de autenticação.

Foram criados 2 usuários com os seguintes alias:

- tiago.aluno
- fabiano.orientador

Ambas as contas de usuários pertencentes ao domínio pjfinal.corp.

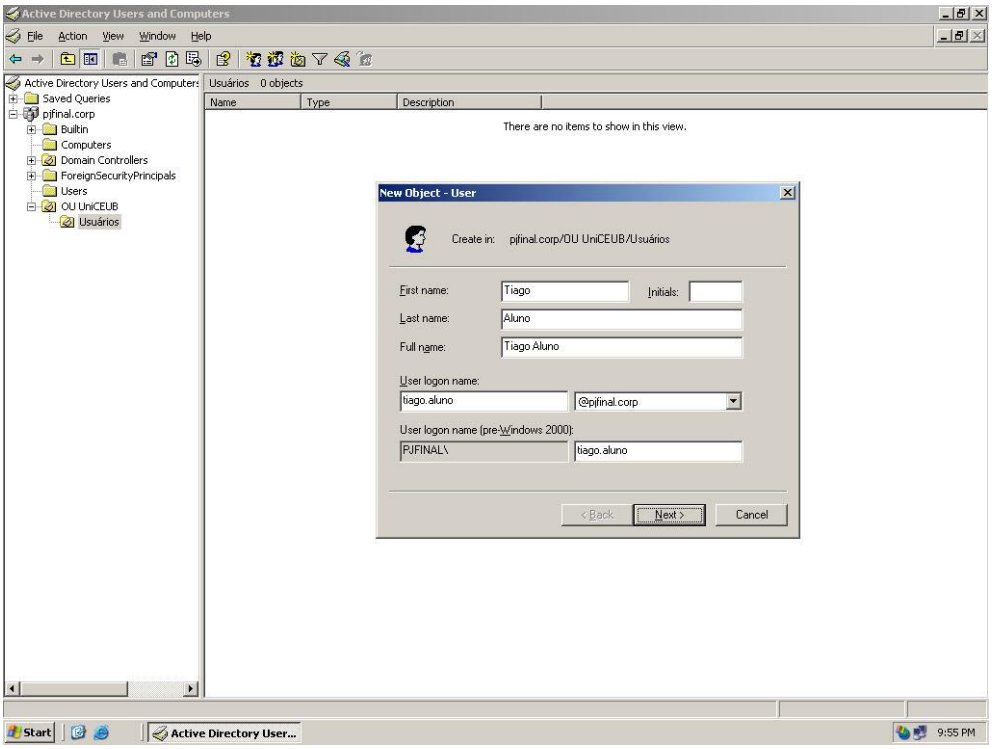

Figura 3.3.1.7 – Criando usuários na console do AD.

Por derradeiro, foram configuradas algumas políticas de segurança para o domínio usando o Group Policy Editor (GPO), dentre elas as opções que determinam as políticas sobre o kerberos. Foram reduzidos os tempos de vida

do ticket de serviço e usuário para 6 horas, o tempo padrão do Windows Server 2003 era de 10 horas. Isso ajuda a elevar o nível de segurança do domínio.

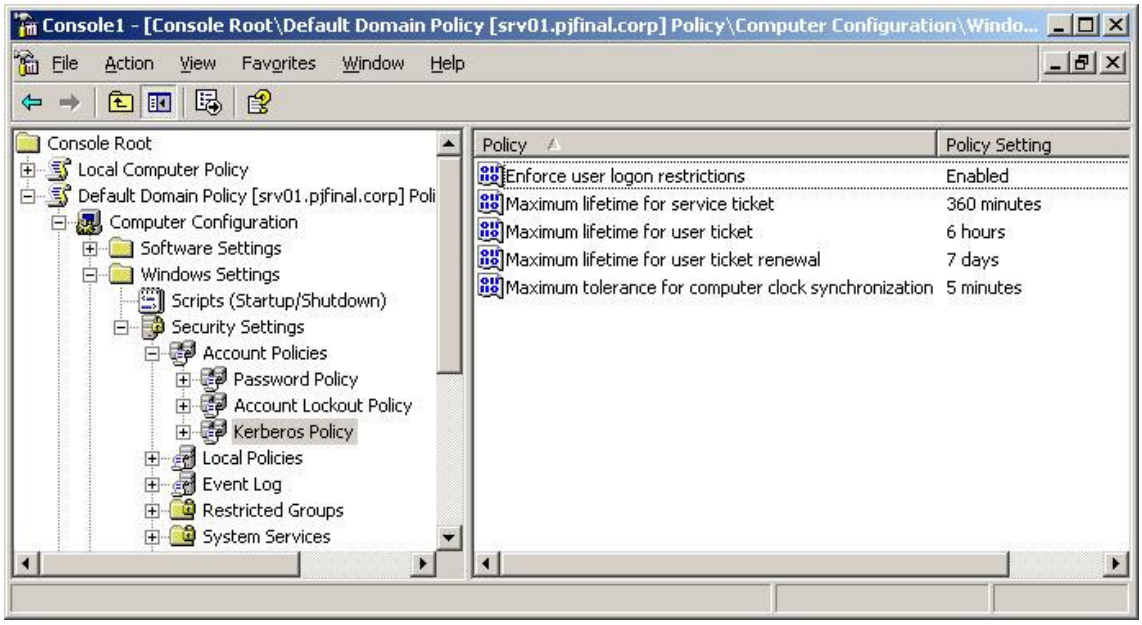

Figura 3.3.1.8 – Definindo as políticas de segurança no GPO.

3.3.2 CONFIGURANDO A ESTAÇÃO GNU/LINUX

Para configurar a estação GNU/LINUX de modo que fizesse parte do domínio PJFINAL.CORP utilizamos o seguinte procedimento:

Primeiramente executamos o software LeasyConfig, é apenas um executável e não há necessidade de instalação.

Antes de começar colocar as informações, realizamos um backup da configuração existente clicando no botão Backup, o LeasyConfig informou que esse procedimento foi realizado (vide Figura 3.2.2.3).

Inserimos as informações nas guias Básico e Avançado conforme ilustram as figuras abaixo:

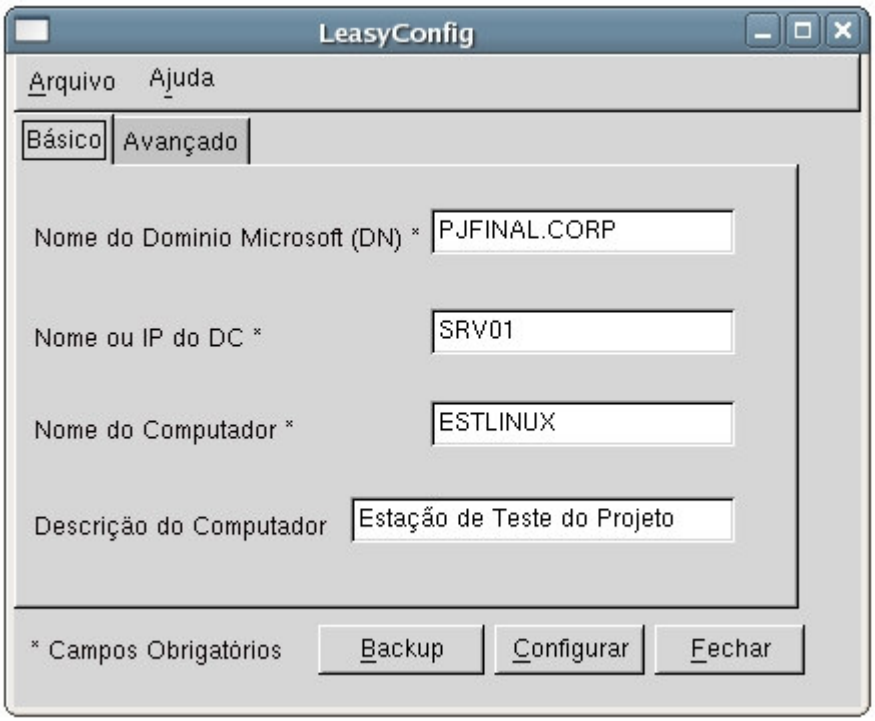

Figura 3.3.2.1 – Inserindo informações na guia Básico do LeasyConfig.

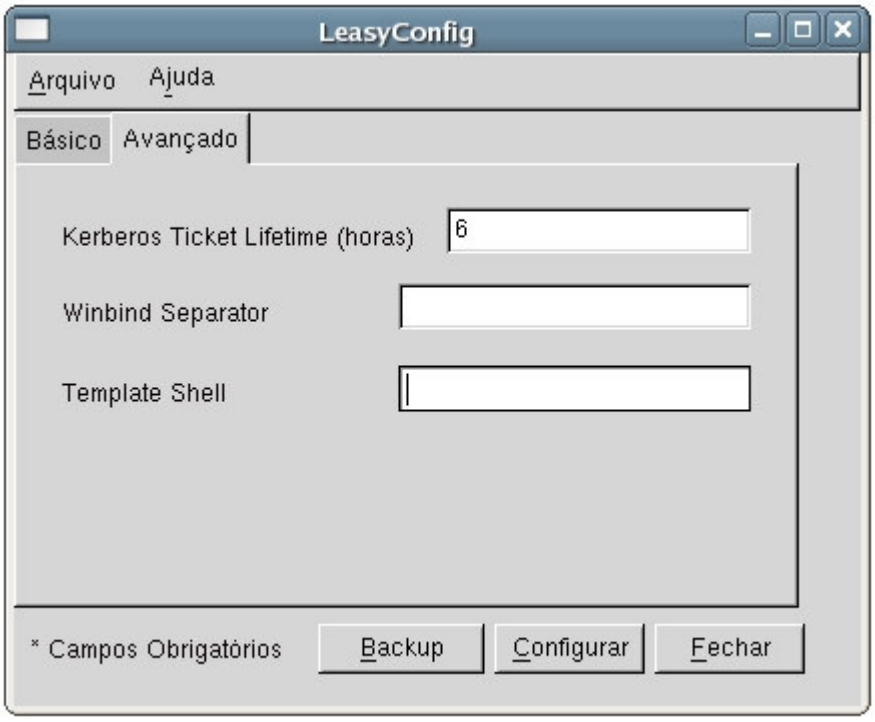

Figura 3.3.2.2 – Inserindo informações na guia Avançado do LeasyConfig.

Nota-se que na guia Avançado personalizamos o tempo de vida do ticket kerberos para 6 horas e não preenchemos as demais guias. Assim sendo, o

software irá usar os valores padrões para os campos não preenchidos, esses valores são:

- Winbind Separator = +
- Template Shell = /bin/bash

Após clicar no botão Configurar, o LeasyConfig informou que a configuração foi realizada com sucesso conforme ilustrado na Figura 3.2.2.4.

Os arquivos de configuração do Kerberos, Samba, PAM e NSS que são gerados pelo LeasyConfig serão anexados na sessão APÊNDICE desta monografia.

Após da configuração através do LeasyConfig, precisamos reiniciar a estação e executar o seguinte comando:

- **net ads join -U <administrador>**

Após a inserção da senha de administrador do domínio pjfinal.corp esse comando irá criar uma conta para o computador no Active Directory com base nos arquivos gerados pelo LeasyConfig.

3.3.3 ETAPAS PARA IMPLANTAÇÃO

Após o projeto ser validado no ambiente de testes, o início da migração ocorreu em setores estratégicos, como o de desenvolvimento de softwares e de engenharia de hardware.

As estações de desenvolvimento e de engenharia foram escolhidas devido ao fato de seus usuários, geralmente engenheiros e analistas de sistemas, possuírem conhecimentos técnicos avançados em informática. Assim sendo, os problemas iniciais foram detectados por pessoas conhecedoras do novo sistema operacional, isso auxiliou e deu agilidade para que as devidas correções fossem realizadas em tempo hábil.

Somente após a correção desses problemas pontuais, que serão descritos no próximo tópico, é que a migração foi liberada para usuários de outros departamentos da empresa.

3.3.4 DIFICULDADES COM A IMPLANTAÇÃO

Após a implantação do modelo proposto na empresa estudada, foram encontrados alguns problemas conforme descrito na tabela abaixo:

Tabela 3.3.4.1 – Problemas com a Implantação.

PROBLEMA	SOLUÇÃO
Estação GNU/Linux não permite o login na interface gráfica, mas permitia em modo texto.	Inclusão da configuração do arquivo PAM referente ao GDM (gestor de Login) no LeasyConfig.
Estação não ingressava no domínio após configuração via LeasyConfig.	Configuração do DNS da rede na estação (pois estava com IP FIXO).
Algum tempo após o login, estação não acessava recursos da rede.	Checar se o tíquete kerberos estavam configurados com um lifetime que não expirasse em 1 turno de 4 horas.
Shell retornando erro "kerberos file is invalid".	Alteração no Leasy config para inserir o caracter " } " de fim da função em uma linha a parte.
Usuário não conseguir logar.	Instruir usuários a colocar o nome do domínio antes do username.

Esses problemas foram solucionados basicamente com alterações no código do Leasyconfig e após a realização de instruções aos usuários das estações de trabalho.

3.4 CUSTOS DO PROJETO

Com o funcionamento das duas plataformas na mesma rede, proprietária na parte servidora e livre nas estações; As empresas, principalmente pequenas e médias, poderão usufruir-se do melhor de cada uma delas.

Tabela 3.4.1 - Comparativo do custo de licenciamento.

(Valores de uma cotação feita em Fevereiro de 2007 por um Revendedor Microsoft)

Valor Unitário	Software	Modelo Proprietário		Modelo Proposto	
		Qtd	Total	Qtd	Total
$829,00	Windows Server 2003 Standard Edition OLP	3	$2.487,00	3	$2.487,00
$806,00	Exchange Server 2005 Standard OLP	1	$806,00	1	$806,00
$1.021,00	SQL Server Standard Edition OLP	1	$1.021,00	1	$1.021,00
$33,00	CAL Windows Server	70	$2.310,00	70	$2.310,00
$77,00	CAL Exchange Server	70	$5.390,00	70	$5.390,00
$186,00	CAL SQL	70	$13.020,00	70	$13.020,00
$215,00	Windows Vista Business Enterprise OLP	70	$15.050,00	0	$0,00
$551,00	Office Professional OLP	70	$38.570,00	0	$0,00
TOTAL (U$ Dólar)		$78.654,00		$25.034,00	
TOTAL em Reais R$ (Cotação de Fev 2007)		**R$ 169.892,64**		**R$ 54.073,44**	
Percentual do Custo frente ao Modelo Proprietário		**31,83%**			

Na tabela 3.4.1 notamos que adotando um modelo onde as estações usam software livre, podemos reduzir o valor a ser gasto com licenciamento de software em uma rede de médio porte (tamanho aproximado de uma unidade da Autotrac) com:

-1 Servidor de Aplicação/Banco de Dados;

-1 Servidor de Correio Eletrônico;

-1 Controlador de Domínio/Arquivos;

-70 Estações de Trabalho.

Essa redução é de aproximadamente 31,83% da previsão de um modelo 100% proprietário (ou uma economia de 68,17%).

A redução dos gastos com licenciamento de software pode variar de acordo com o número de estações de trabalho migradas, sendo que o aumento na quantidade de estações é diretamente proporcional à redução dos custos com licenciamento.

Na tabela 3.4.2 podemos notar que a adoção do modelo sugerido por esse projeto pode propiciar uma economia que varia entre 40% e 70% dependendo do número de estações migradas.

Tabela 3.4.2 – Evolução da economia do modelo proposto.

(Valores de uma cotação feita em Fevereiro de 2007 por um Revendedor Microsoft)

Quantidade de Estações	Custo Modelo Proprietário	Custo Modelo Proposto	Economia Frente ao Modelo Proprietário
5	R$ 20.787,84	R$ 12.515,04	39,80%
10	R$ 32.257,44	R$ 15.711,84	51,29%
20	R$ 55.196,64	R$ 22.105,44	59,95%
40	R$ 101.075,04	R$ 34.892,64	65,48%
80	R$ 192.831,84	R$ 60.467,04	68,64%
160	R$ 376.345,44	R$ 111.615,84	70,34%
320	R$ 743.372,64	R$ 213.913,44	71,22%
640	R$ 1.477.427,04	R$ 418.508,64	71,67%

Nesse modelo, o aumento da economia sofre uma saturação quando o número de estações chega a 100 aproximadamente (conforme ilustra o gráfico da Figura 3.4.1).

Com base nessas informações, esse projeto poderá ser aplicado em qualquer empresa, uma vez que o mesmo traz uma economia inicial de 40% aproximadamente, mesmo para um número muito pequeno de estações de trabalho a serem migradas. Essa economia poderá chegar ao patamar de 70%, onde irá ocorrer a saturação.

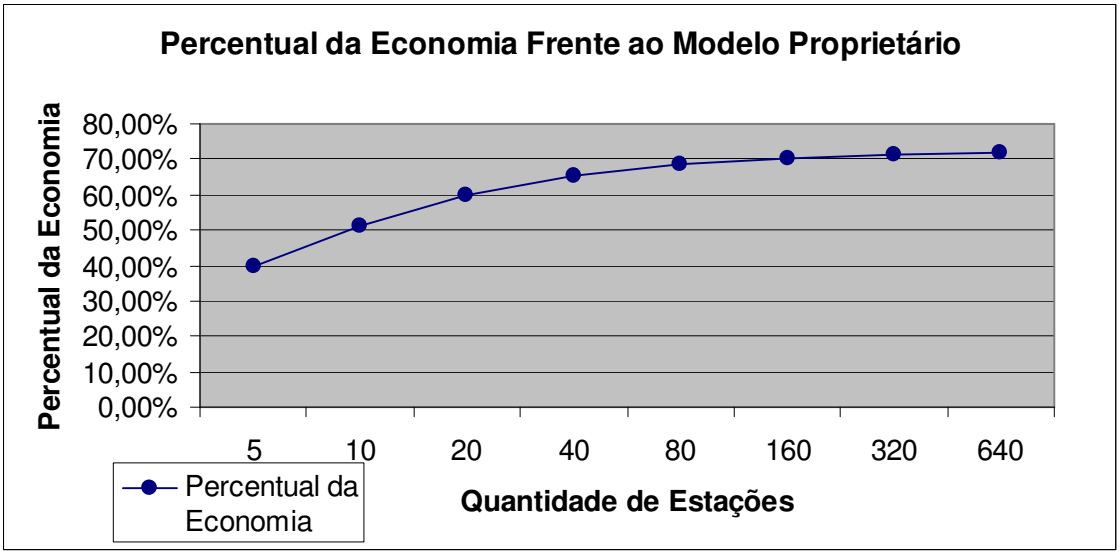

Figura 3.4.1 – Gráfico do Percentual de Economia
(Dados extraídos da Tabela 3.4.2)

A Microsoft possui um complexo modelo de licenciamento, assim sendo, foi preciso consumir muito tempo com a pesquisa dos modelos existentes e quais seriam usados para alimentar as planilhas do projeto.

Após a pesquisa, foram utilizados os valores da modalidade OPEN LICENSE, pois era o modelo usado pela empresa estudada e também era um modelo de licenciamento indicado para empresas com até 500 estações, o que se encaixa no escopo desse projeto.

CAPÍTULO 4 – CONCLUSÃO

O projeto surgiu em virtude de uma demanda no mercado, esse modelo de rede proposto aliado ao software LeasyConfig que foi desenvolvido irá auxiliar na adoção dessa solução no mercado corporativo.

Os objetivos foram alcançados com o sucesso na autenticação da estação GNU/LINUX de teste no controlador de domínio. Os usuários criados no Active Directory e respectivas senhas de acesso foram usados para efetuar login nas estações.

Também foram gerados com sucesso todos os arquivos de configuração através do software LeasyConfig que foi desenvolvido, bem como os arquivos antigos foram salvos através da opção de backup.

Com o término do projeto, vimos que sua aplicação ajuda a reduzir o custo com licenciamento de softwares nas empresas, sendo que essa economia cresce exponencialmente até um patamar de 70%, onde ocorre a saturação desse crescimento. Esses dados foram validados na empresa onde foi realizado o estudo de caso, sendo que esta emitiu uma declaração que está em anexo na seção Apêndice.

O projeto se baseou em diversos conteúdos ministrados no curso de engenharia da computação como conceitos de redes, segurança, linguagens de programação, projeto de sistemas, arquitetura de computadores, entre outros assuntos.

4.1 PROJETOS FUTUROS

Algumas perspectivas de evolução do projeto estão listadas abaixo:

- Desenvolver funcionalidade de configuração em rede para o software LeasyConfig;
- Desenvolver uma interface Web para o LeasyConfig;
- Desenvolver funcionalidade de configuração de serviços e comandos externos do shell para o LeasyConfig;
- Validar o projeto para outras distribuições e interfaces gráficas.

REFERÊNCIAS BIBLIOGRÁFICAS

ALECRIM, E.: O Sistema Operacional GNU, 2003. (http://www.infowester.com/linux5.php acessado em 07/05/2007)

ALMEIDA, C. M.: Compilando o Squid com autenticação PAM, 2005. (http://www.vivaolinux.com.br/artigos/verArtigo.php?codigo=2329 acessado em 07/05/2007)

ÂNGELO, F.: Conheça um pouco da história e do funcionamento do Linux, 2007. (http://tecnologia.uol.com.br/ultnot/2007/04/13/ult4213u66.jhtm acessado em 07/05/2007)

BARREIROS, C. C.: O Samba, 2001. (http://www.gta.ufrj.br/grad/01_2/samba/samba.htm acessado em 06/02/2007)

BATTISTI, J.: Livro Windows Server 2003 Curso Completo Editora Axcel Books. Capítulo 5 – Pág 180 e 181, 2003.

BOVET, D. P.; **CESATI**, M.: Understanding the Linux kernel 3 Ed. Sebastopol: O'Reilly, 2005.

FIDEL, Raya. The case study method: a case study. In: GLAZIER, Jack D. & POWELL, Ronald R. Qualitative research in information management. Englewood, CO: Libraries Unlimited, 1992. 238p. p.37-50.

FILHO, M. M. C.: Necessidade de Segurança, 1999 (http://www.gta.ufrj.br/grad/99_2/marcos/seguranca.htm acessado em 05/02/2007)

FILHO, M. M. C.: Kerberos, 1999 (http://www.gta.ufrj.br/grad/99_2/marcos/kerberos.htm acessado em 05/02/2007)

FSF, 2000: Free Software Foundation – Free software is a matter of liberty not price. (www.fsf.org acessado em 07/05/2007)

GNOME, 2005: Sobre o Gnome (http://br.gnome.org/sobre/ acessado em 12/05/2007)

GNU, 1996: The GNU Operating System (www.gnu.org acessado em 07/05/2007)

HARTLEY, Jean F. Case studies in organizational research. In: CASSELL, Catherine & SYMON, Gillian (Ed.). Qualitative methods in organizational research: a practical guide. London: Sage, 1994. 253p. p. 208-229.

JACQUET, B.: Pluggable Authentication Modules – PAM, 2006.(http://web.ist.utl.pt/ist151604/files/tfc/**pam**.pdf acessado em 06/02/2007)

KDE, 2007: K Desktop Environment Brasil (http://twiki.softwarelivre.org/bin/view/KdeBR/WebHome acessado em 12/05/2007)

KOHL, J.; **NEUMAN**, B. C.: The Kerberos Network Authentication Service (Version 5). Internet Request for Comments RFC-1510. Project Athena – MIT , 1993 (ftp://ftp.isi.edu/in-notes/rfc1510.txt acessado em 05/02/2007)

LAZARUS, 2007: About the Lazarus Project (http://www.lazarus.freepascal.org/ acessado em 12/05/2007).

MAYER, E. V. D.; **FRIES**, F. D.; **BAÚ**, G.: Samba Servidor de Domínio. DeTec – UNIJUI, 2004. (http://santarosa.rs.gov.br/~smb/samba.pdf acessado em 05/02/2007)

MICROSOFT: 2147A Gerenciando um Ambiente Microsoft Windows Server 2003. Microsoft Official Course. Módulo 1 Pág 08, 2007

MOITINHO, S. D.: Segurança em Sistemas Distribuídos. Pós-Graduação em Sistemas Distribuídos. Universidade Federal da Bahia, 2001. (http://twiki.im.ufba.br/pub/GESI/WebHome/SeguranaemSistemasDistribudos.pdf acessado em 10/12/2006).

MOREIRA, A.: Integração Unix / MS-Windows – ISEP, 2002. (http://www.dei.isep.ipp.pt/~andre/documentos/unix-windows.html acessado em 07/05/2007)

PINTO, A. C.: Mecanismos para Gerência de Segurança em Redes CPGCC da UFRGS – Departamento de Informática, 2006 (http://penta.ufrgs.br/gereseg/kerberos.html acessado em 10/09/2006)

SAMAR, V.; **LAI**, C.: Making Login Services Independent of Authentication Technologies. SunSoft Inc – Sun Microsystems, 1996.

(http://www.sun.com/software/solaris/pam/pam.external.pdf acessado em 06/02/2007)

SAMAR, V.; **SCHEMERS**, R.: Unified Login With Pluggable Authentication Modules (PAM) 1995. (http://www.opengroup.org/tech/rfc/rfc86.0.html acessado em 06/02/2007)

SILVA, G. M.: Guia FOCA LINUX – Módulo Avançado Samba Versão 6.40 de 30 de Outubro de 2006 (http://focalinux.cipsga.org.br/ acessado em 06/02/2007)

SILVA, G. M.: Guia FOCA LINUX – Módulo Intermediário Versão 5.45 de 30 de Outubro de 2006 (http://focalinux.cipsga.org.br/ acessado em 06/02/2007)

TANENBAUM, A. S.: Rede de Computadores 4ª Edição Editora Campus. Capítulo 8 – Pág 834 à 848, 2003

TANENBAUM, A.: Sistemas Operacionais Modernos. Rio de Janeiro: LTC, 1999.

TECHNET, 2005: Microsoft Windows Server 2003 Tech Center (http://www.microsoft.com/technet/prodtechnol/windowsserver2003/pt-br/library/ServerHelp/2e0186ba-1a09-42b5-81c8-3ecca4ddde5e.mspx?mfr=true acessado em 07/05/2007)

TECHNET, 2006:Gerenciamento de acesso à Intranet. Capítulo 4 (http://www.microsoft.com/brasil/technet/security/topics/identitymanagement/idmanage/P3Intran_3.mspx acessado em 07/05/2007)

APÊNDICES

A seguir será descrito o código fonte do LeasyConfig, como esse é um projeto acadêmico, qualquer pessoa poderá usar/alterar/distribuir esse código. Essa decisão visa ajudar a disseminar o modelo proposto nesse projeto.

Os arquivos de configuração gerados pelo software LeasyConfig também serão anexados.

Finalmente, será apresentada uma declaração emitida pela empresa Autotrac Comércio e Telecomunicações S/A, local onde foi realizado o projeto.

APÊNDICE A

Código Fonte do LeasyConfig

```
unit Unit1;

{$mode objfpc}{$H+}

interface

uses
  Inifiles, Classes, SysUtils, LResources, Forms, Controls, Graphics, Dialogs,
Menus,
  ExtCtrls, Buttons, StdCtrls;

type

  { TForm1 }

  TForm1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit4: TEdit;
    Edit5: TEdit;
    Edit6: TEdit;
    Edit7: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    MainMenu1: TMainMenu;
    MenuItem1: TMenuItem;
    MenuItem2: TMenuItem;
    MenuItem3: TMenuItem;
    MenuItem4: TMenuItem;
    MenuItem5: TMenuItem;
    MenuItem6: TMenuItem;
    Notebook1: TNotebook;
    Page1: TPage;
    Page2: TPage;
```

```
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
    procedure Edit6Change(Sender: TObject);
    procedure Label1Click(Sender: TObject);
    procedure Label2Click(Sender: TObject);
    procedure Label4Click(Sender: TObject);
    procedure MenuItem3Click(Sender: TObject);
    procedure MenuItem4Click(Sender: TObject);
    procedure MenuItem5Click(Sender: TObject);
    procedure MenuItem6Click(Sender: TObject);
    procedure Page1ContextPopup(Sender: TObject; MousePos: TPoint;
      var Handled: Boolean);
    procedure Page2ContextPopup(Sender: TObject; MousePos: TPoint;
      var Handled: Boolean);
    procedure ToggleBox1Change(Sender: TObject);
  private
    { private declarations }
  public
    { public declarations }
  end;

var
  Form1: TForm1;
  strtmp: string;
  Krb_file,smb_file: TIniFile;
  pam_login,pam_samba,nsswitch_file: TextFile;

implementation

{ TForm1 }

procedure TForm1.Edit6Change(Sender: TObject);
begin

end;

procedure TForm1.Button2Click(Sender: TObject);
begin
//Sair do LeasyConfig
close;
//
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
//
//Bloco para Backup da Configuracao...
//
```

```
RenameFile('/etc/krb5.conf','/etc/krb5.conf_old');
RenameFile('/etc/samba/smb.conf','/etc/samba/smb.conf_old');
RenameFile('/etc/nsswitch','/etc/nsswitch_old');
MessageDlg ('Backup Realizado com Sucesso',mtinformation,[mbok], 0);
Button3.Enabled:=false;
//
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
//
//Bloco para configuracao do Samba
//
smb_file:=TIniFile.create('/etc/samba/smb.conf'); //Cria o Arquivo de
configuracao
//
smb_file.WriteString('global','workgroup',edit1.Text); //Secao Global
smb_file.WriteString('global','realm',edit1.Text); //Secao Global
smb_file.WriteString('global','server string',Edit5.Text); //Secao Global
smb_file.WriteString('global','printcap name','/etc/printcap'); //Secao Global
smb_file.WriteString('global','load printers','yes'); //Secao Global
smb_file.WriteString('global','printing','lprng'); //Secao Global
smb_file.WriteString('global','log file','/var/log/samba/%m.log'); //Secao Global
smb_file.WriteString('global','max log size','50'); //Secao Global
smb_file.WriteString('global','security','ADS'); //Secao Global
smb_file.WriteString('global','password server',edit2.Text); //Secao Global
smb_file.WriteString('global','unix password sync','Yes'); //Secao Global
smb_file.WriteString('global','socket options','TCP_NODELAY
SO_RCVBUF=8192 SO_SNDBUF=8192'); //Secao Global
smb_file.WriteString('global','local master','yes'); //Secao Global
smb_file.WriteString('global','domain master','no'); //Secao Global
smb_file.WriteString('global','preferred master','yes'); //Secao Global
smb_file.WriteString('global','name resolve order','bcast wins hosts lmhosts');
//Secao Global
smb_file.WriteString('global','wins support','no'); //Secao Global
smb_file.WriteString('global','dns proxy','yes'); //Secao Global
smb_file.WriteString('global','idmap uid','16777216-33554431'); //Secao Global
smb_file.WriteString('global','idmap gid','16777216-33554431'); //Secao Global

if edit7.Text='' then
   smb_file.WriteString('global','template shell','/bin/bash') //Secao Global
   else
   smb_file.WriteString('global','template shell',Edit7.Text); //Secao Global

smb_file.WriteString('global','winbind use default domain','yes'); //Secao Global
smb_file.WriteString('global','max connections','0'); //Secao Global
smb_file.WriteString('global','update encrypted','yes'); //Secao Global
smb_file.WriteString('global','allow trusted domains','yes'); //Secao Global
smb_file.WriteString('global','winbind cache time','10'); //Secao Global
smb_file.WriteString('global','winbind enum users','yes'); //Secao Global
```

```
smb_file.WriteString('global','winbind enum groups','yes'); //Secao Global
smb_file.WriteString('global','template homedir','/home/%D/%U'); //Secao Global
smb_file.WriteString('global','template shell','/bin/bash'); //Secao Global
smb_file.WriteString('global','netbios name',edit3.Text); //Secao Global

if edit6.Text='' then
   smb_file.WriteString('global','winbind separator','+') //Secao Global
   else
   smb_file.WriteString('global','winbind separator',edit6.Text); //Secao Global
//
smb_file.WriteString('homes','comment','Pastas Pessoais'); //Secao Homes
smb_file.WriteString('homes','browseable','yes'); //Secao Homes
smb_file.WriteString('homes','writeable','yes'); //Secao Homes
//
smb_file.WriteString('share','comment','Pasta Publica'); //Secao Share
smb_file.WriteString('share','path','/tmp'); //Secao Share
smb_file.WriteString('share','writeable','yes'); //Secao Share
smb_file.WriteString('share','guest ok','yes'); //Secao Share
//
smb_file.WriteString('printers','comment','Impressoras Compartilhadas');
//Secao Printers
smb_file.WriteString('printers','path','/var/spool/samba'); //Secao Printers
smb_file.WriteString('printers','browseable','yes'); //Secao Printers
smb_file.WriteString('printers','writeable','yes'); //Secao Printers
smb_file.WriteString('printers','printable','yes'); //Secao Printers
//
smb_file.Destroy;
//
//
//Bloco para configuracao do Kerberos v5
//
Krb_file:=TIniFile.create('/etc/krb5.conf'); //Cria o Arquivo de configuracao
//
Krb_file.WriteString('Logging','default','FILE:/var/log/krb5libs.log'); //Secao
Logging
Krb_file.WriteString('Logging','kdc','FILE:/var/log/krb5kdc.log'); //Secao Logging
Krb_file.WriteString('Logging','admin_server','FILE:/var/log/kadmind.log');
//Secao Logging
//
Krb_file.WriteString('libdefaults','default_realm',Edit1.Text); //Secao Libdefaults
Krb_file.WriteString('libdefaults','dns_lookup_realm','false'); //Secao Libdefaults
Krb_file.WriteString('libdefaults','dns_lookup_kdc','false'); //Secao Libdefaults

if edit4.Text='' then
   Krb_file.WriteString('libdefaults','ticket_lifetime','24h') //Secao Libdefaults
   else
   Krb_file.WriteString('libdefaults','ticket_lifetime',edit4.Text); //Secao
Libdefaults

Krb_file.WriteString('libdefaults','forwardable','yes'); //Secao Libdefaults
```

```
//
Krb_file.WriteString('realms',Edit1.Text,'  {'); //Secao Realms
strtmp:=(' '+Edit2.Text+'.'+Edit1.text+':88');
Krb_file.WriteString('realms','kdc',strtmp); //Secao Realms
strtmp:=(' '+Edit2.Text+'.'+Edit1.text+':749');
Krb_file.WriteString('realms','admin_server',strtmp); //Secao Realms
strtmp:=(Edit1.text+'  }');
Krb_file.WriteString('realms','default_domain',strtmp); //Secao Realms
//
strtmp:=('.'+Edit1.text+' ');
Krb_file.WriteString('domain_realm',strtmp,edit1.Text); //Secao Domain_Realm
Krb_file.WriteString('domain_realm',edit1.Text,edit1.Text); //Secao
Domain_Realm
//
Krb_file.WriteString('kdc','profile','/var/kerberos/krb5kdc/kdc.conf'); //Secao KDC
//
Krb_file.WriteString('appdefaults','pam',' {'); //Secao Appdefaults
Krb_file.WriteString('appdefaults','debug','false'); //Secao Appdefaults
Krb_file.WriteString('appdefaults','ticket_lifetime','36000'); //Secao Appdefaults
Krb_file.WriteString('appdefaults','renew_lifetime','36000'); //Secao Appdefaults
Krb_file.WriteString('appdefaults','forwardable','true'); //Secao Appdefaults
Krb_file.WriteString('appdefaults','krb4_convert','false   }'); //Secao Appdefaults
//
Krb_file.Destroy;
//
//
// Geracao dos arquivos da pasta PAM (Login e Samba)
//
AssignFile(pam_login,'/etc/pam.d/login');
Rewrite(pam_login);
Writeln(pam_login,'#Arquivo de Configuracao do PAM-LOGIN');
Writeln(pam_login,'#Gerado pelo LEASYCONFIG');
Writeln(pam_login,'#Orientador: Fabiano Mariath D´Oliveira');
Writeln(pam_login,'#Aluno: Tiago de Assis O Castro');
Writeln(pam_login,'#UniCEUB - Projeto Final');
Writeln(pam_login,'#');
Writeln(pam_login,'#');
Writeln(pam_login,'auth     requisite      pam_securetty.so');
Writeln(pam_login,'auth     requisite      pam_nologin.so');
Writeln(pam_login,'auth     requisite      pam_env.so');
Writeln(pam_login,'auth     sufficient     pam_winbind.so');
Writeln(pam_login,'auth     sufficient     pam_unix.so nullok use_first_pass');
Writeln(pam_login,'account  sufficient     pam_winbind.so');
Writeln(pam_login,'account  sufficient     pam_unix.so');
Writeln(pam_login,'account  required       pam_stack.so service=system-auth');
Writeln(pam_login,'session  required       pam_mkhomedir.so skel=/etc/skel
umask=0022');
Writeln(pam_login,'session  required       pam_unix.so');
Writeln(pam_login,'session  optional       pam_lastlog.so');
Writeln(pam_login,'session  optional       pam_motd.so');
```

```
Writeln(pam_login,'session  optional      pam_mail.so standard noenv');
Writeln(pam_login,'password sufficient  pam_unix.so nullok obscure min=4');
Writeln(pam_login,'session  required      pam_selinux.so multiple');
Writeln(pam_login,'session  required      pam_stack.so service=system-auth');
Writeln(pam_login,'session  optional      pam_console.so');
Writeln(pam_login,'#');
Writeln(pam_login,'#LeasyConfig - Fim do Arquivo');
Writeln(pam_login,'#');
CloseFile(pam_login);
//
AssignFile(pam_samba,'/etc/pam.d/samba');
Rewrite(pam_samba);
Writeln(pam_samba,'#Arquivo de Configuracao do PAM-SAMBA');
Writeln(pam_samba,'#Gerado pelo LEASYCONFIG');
Writeln(pam_samba,'#Orientador: Fabiano Mariath D´Oliveira');
Writeln(pam_samba,'#Aluno: Tiago de Assis O Castro');
Writeln(pam_samba,'#UniCEUB - Projeto Final');
Writeln(pam_samba,'#');
Writeln(pam_samba,'#');
Writeln(pam_samba,'auth     required     /lib/security/pam_securetty.so');
Writeln(pam_samba,'auth     required     /lib/security/pam_nologin.so');
Writeln(pam_samba,'auth     sufficient   /lib/security/pam_winbind.so');
Writeln(pam_samba,'auth     required     /lib/security/pam_pwdb.so
use_first_pass shadow nullok');
Writeln(pam_samba,'account  required     /lib/security/pam_winbind.so');
Writeln(pam_samba,'#');
Writeln(pam_samba,'#LeasyConfig - Fim do Arquivo');
Writeln(pam_samba,'#');
CloseFile(pam_samba);
//
//
// Geracao do arquivo NSSWITCH
//
AssignFile(nsswitch_file,'/etc/nsswitch.conf');
Rewrite(nsswitch_file);
Writeln(nsswitch_file,'#Arquivo de Configuracao NSSWITCH');
Writeln(nsswitch_file,'#Gerado pelo LEASYCONFIG');
Writeln(nsswitch_file,'#Orientador: Fabiano Mariath D´Oliveira');
Writeln(nsswitch_file,'#Aluno: Tiago de Assis O Castro');
Writeln(nsswitch_file,'#UniCEUB - Projeto Final');
Writeln(nsswitch_file,'#');
Writeln(nsswitch_file,'#');
Writeln(nsswitch_file,'passwd:     files winbind');
Writeln(nsswitch_file,'shadow:     files winbind');
Writeln(nsswitch_file,'group:      files winbind');
Writeln(nsswitch_file,'hosts:      files dns');
Writeln(nsswitch_file,'bootparams: nisplus [NOTFOUND=return] files');
Writeln(nsswitch_file,'ethers:     files');
Writeln(nsswitch_file,'netmasks:   files');
Writeln(nsswitch_file,'networks:   files');
```

```
Writeln(nsswitch_file,'protocols:  files winbind');
Writeln(nsswitch_file,'rpc:        files');
Writeln(nsswitch_file,'services:   files winbind');
Writeln(nsswitch_file,'netgroup:   files winbind');
Writeln(nsswitch_file,'publickey:  nisplus');
Writeln(nsswitch_file,'automount:  files winbind');
Writeln(nsswitch_file,'aliases:    files nisplus');
Writeln(nsswitch_file,'#');
Writeln(nsswitch_file,'#LeasyConfig - Fim do Arquivo');
Writeln(nsswitch_file,'#');
CloseFile(nsswitch_file);
//
//Informacao do termino da Operacao
MessageDlg ('Computador Configurado com Sucesso. Favor
Reiniciar...',mtinformation,[mbok], 0);
Button1.Enabled:=false;
//
end;

procedure TForm1.Edit1Change(Sender: TObject);
begin

end;

procedure TForm1.Label1Click(Sender: TObject);
begin

end;

procedure TForm1.Label2Click(Sender: TObject);
begin

end;

procedure TForm1.Label4Click(Sender: TObject);
begin

end;

procedure TForm1.MenuItem3Click(Sender: TObject);
begin
//Executa o Backup da Configuracao
Button3.Click;
//
end;

procedure TForm1.MenuItem4Click(Sender: TObject);
begin
//Executa a Configuracao
Button1.Click;
```

```
//
end;

procedure TForm1.MenuItem5Click(Sender: TObject);
begin
//Sair do LeasyConfig...
close;
//
end;

procedure TForm1.MenuItem6Click(Sender: TObject);
begin
//Sobre o LeasyConfig
MessageDlg ('LeasyConfig v1.0 (Software Livre)
tiago.acastro@gmail.com',mtinformation,[mbok], 0);
//
end;

procedure TForm1.Page1ContextPopup(Sender: TObject; MousePos: TPoint;
  var Handled: Boolean);
begin
   close;
end;

procedure TForm1.Page2ContextPopup(Sender: TObject; MousePos: TPoint;
  var Handled: Boolean);
begin

end;

procedure TForm1.ToggleBox1Change(Sender: TObject);
begin

end;

initialization
  {$I unit1.lrs}

end.
```

APÊNDICE B

Arquivo de configuração do Kerberos.

Krb5.conf

[Logging]
default=FILE:/var/log/krb5libs.log
kdc=FILE:/var/log/krb5kdc.log
admin_server=FILE:/var/log/kadmind.log

[libdefaults]
default_realm=PJFINAL.CORP
dns_lookup_realm=false
dns_lookup_kdc=false
ticket_lifetime=6h
forwardable=yes

[realms]
PJFINAL.CORP={
kdc=SRV01.PJFINAL.CORP:88
admin_server=SRV01.PJFINAL.CORP:749
default_domain=PJFINAL.CORP
}

[domain_realm]
.PJFINAL.CORP=PJFINAL.CORP
PJFINAL.CORP=PJFINAL.CORP

[kdc]
profile=/var/kerberos/krb5kdc/kdc.conf

[appdefaults]
pam={
debug=false
ticket_lifetime=36000
renew_lifetime=36000
forwardable=true
krb4_convert=false
}

APÊNDICE C

Arquivo de configuração do Samba.

smb.conf

[global]
workgroup=PJFINAL.CORP
realm=PJFINAL.CORP
server string=Estação de Teste do Projeto
printcap name=/etc/printcap
load printers=yes
printing=lprng
log file=/var/log/samba/%m.log
max log size=50
security=ADS
password server=SRV01
unix password sync=Yes
socket options=TCP_NODELAY SO_RCVBUF=8192 SO_SNDBUF=8192
local master=yes
domain master=no
preferred master=yes
name resolve order=bcast wins hosts lmhosts
wins support=no
dns proxy=yes
idmap uid=16777216-33554431
idmap gid=16777216-33554431
template shell=/bin/bash
winbind use default domain=yes
max connections=0
update encrypted=yes
allow trusted domains=yes
winbind cache time=10
winbind enum users=yes
winbind enum groups=yes
template homedir=/home/%D/%U
netbios name=ESTLINUX
winbind separator=+

[homes]
comment=Pastas Pessoais
browseable=yes
writeable=yes

[share]
comment=Pasta Publica
path=/tmp
writeable=yes
guest ok=yes

[printers]
comment=Impressoras Compartilhadas
path=/var/spool/samba
browseable=yes
writeable=yes
printable=yes

APÊNDICE D

Arquivo de configuração do NSS.

nsswitch.conf

```
#Arquivo de Configuracao NSSWITCH
#Gerado pelo LEASYCONFIG
#Orientador: Fabiano Mariath D´Oliveira
#Aluno: Tiago de Assis O Castro
#UniCEUB - Projeto Final
#
#
passwd:     files winbind
shadow:     files winbind
group:      files winbind
hosts:      files dns
bootparams: nisplus [NOTFOUND=return] files
ethers:     files
netmasks:   files
networks:   files
protocols:  files winbind
rpc:        files
services:   files winbind
netgroup:   files winbind
publickey:  nisplus
automount:  files winbind
aliases:    files nisplus
#
#LeasyConfig - Fim do Arquivo
#
```

APÊNDICE E

Arquivos de configuração do PAM.

Login

```
#Arquivo de Configuracao do PAM-LOGIN
#Gerado pelo LEASYCONFIG
#Orientador: Fabiano Mariath D´Oliveira
#Aluno: Tiago de Assis O Castro
#UniCEUB - Projeto Final
#
auth     requisite      pam_securetty.so
auth     requisite      pam_nologin.so
auth     requisite      pam_env.so
auth     sufficient     pam_winbind.so
auth     sufficient     pam_unix.so nullok use_first_pass
account  sufficient   pam_winbind.so
account  sufficient   pam_unix.so
account  required     pam_stack.so service=system-auth
session  required     pam_mkhomedir.so skel=/etc/skel umask=0022
session  required     pam_unix.so
session  optional     pam_lastlog.so
session  optional     pam_motd.so
session  optional     pam_mail.so standard noenv
password sufficient pam_unix.so nullok obscure min=4
session  required     pam_selinux.so multiple
session  required     pam_stack.so service=system-auth
session  optional     pam_console.so
#
#LeasyConfig - Fim do Arquivo
#
```

Samba

```
#Arquivo de Configuracao do PAM-SAMBA
#Gerado pelo LEASYCONFIG
#Orientador: Fabiano Mariath D´Oliveira
#Aluno: Tiago de Assis O Castro
#UniCEUB - Projeto Final
#
auth     required     /lib/security/pam_securetty.so
auth     required     /lib/security/pam_nologin.so
auth     sufficient   /lib/security/pam_winbind.so
auth     required     /lib/security/pam_pwdb.so use_first_pass shadow nullok
account  required     /lib/security/pam_winbind.so
#
#LeasyConfig - Fim do Arquivo
#
```

APÊNDICE F

Declaração da empresa Autotrac Comércio e Telecomunicações S/A.

AUTOTRAC

DECLARAÇÃO

Declaramos para os devidos fins, que o funcionário TIAGO DE ASSIS OLIVEIRA CASTRO, cuja função na Autotrac é ADMINISTRADOR DE REDES desenvolveu um plano de trabalho com objetivo na redução de custo com licenciamento de softwares na área de TI utilizando softwares livres.
Os resultados desse plano de trabalho auxiliaram a Autotrac no sentido de iniciar a utilização de softwares livres em suas estações de trabalho.

Sendo só, colocamo-nos a disposição para maiores esclarecimentos.

Brasília, 14 de junho de 2007.

Eng. Eduardo de Oliveira Paes
Gerente de TI
CREA-DF 9726/D

INSCRIÇÃO NO CNPJ
40 281 347/0001-74
AUTOTRAC COMÉRCIO E TELECOMUNICAÇÕES S. A.
Campus Universitário Darcy Ribeiro
Globo "A" - Autotrac - CEP 70910-901
BRASÍLIA - DF

OmniSAT