UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ
Centro de Engenharia e Ciências Exatas

Autenticação única e segura utilizando Kerberos e LDAP

William Fernando Merlotto

Foz do Iguaçu-PR

2005

UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ
Centro de Engenharia e Ciências Exatas

William Fernando Merlotto

Autenticação única e segura utilizando Kerberos e LDAP

Monografia apresentada ao curso Ciência da Computação da Centro de Engenharia e Ciências Exatas da UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ, como requisito parcial para a obtenção do título de Bacharel sob a orientação do professor Renato Bobsin Machado.

Foz do Iguaçu-PR
2005

Software Livre é uma questão de liberdade, não de preço. Para entender o conceito, você deve pensar em ”liberdade de expressão”, não em ”cerveja grátis”.

Autor desconhecido

Resumo

Com a evolução tecnológica as redes de computadores se tornaram estratégicas para as mais diversas organizações. A partir deste contexto o estudo e aplicação de métodos para garantir a segurança das informações se tornaram fundamentais. O primeiro passo para a proteção dos ambientes computacionais consiste na aplicação de técnicas para a autenticação e autorização de usuários e serviços. Para o atendimento a essa demanda muitos métodos foram desenvolvidos, tais como a autenticação local, *Network Information System* (NIS), NIS+, X.500, entre outras. Neste trabalho propõe-se um método de autenticação híbrido integrando duas tecnologias amplamente utilizadas *Kerberos* e *Lightweight Directory Access Protocol* (LDAP). Por meio da aplicação desse modelo foi possível apreveitar as principais vantages das duas tecnologias, criando um ambiente de autenticação e autorização, garantindo-se as premissas de centralização, segurança e transparência.

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

O desenvolvimento tecnológico e a automação de processos tornaram as organizações cada vez mais dependentes dos sistemas de informação e das redes de computadores para a realização de suas atividades cotidianas. Dentro desse contexto, as informações possuem um grande valor e precisam ser adequadamente protegidas.

Em contrapartida, verifica-se uma crescente evolução de técnicas de ataques aos sistemas computacionais. Há diversos tipos de ameaças, entre as quais: *crackers*, *hackers*, vírus, cavalos de Tróia, negação de serviço, fraudes, sabotagem, fogo ou inundação, ataques realizados por pessoas autorizadas e exploração de informações por meio de engenharia social.

Por meio desse cenário enfatiza-se a carência por ferramentas e políticas capazes de evitar ataques e proteger informações e recursos. Uma atividade de importância fundamental para alcançar-se esse objetivo é o controle e a verificação permissões de acesso às informações e recursos. Muitas tecnologias e métodos foram desenvolvidos para suprir essa necessidade, algumas delas complexas e distribuídas, tais como *Network Information System* (NIS), NIS+, X.500 e outras mais simples como a própria autenticação local.

Nesse trabalho apresenta-se uma abordagem de autenticação e autorização híbrida, integrando as tecnologias *Kerberos* e *Lightweight Directory Access Protocol*(LDAP). O objetivo do trabalho consiste em utilizar e avaliar as características desse modelo em ambientes reais.

No Capítulo2 é apresentada a conceituação teórica sobre os sistemas de autenticação.

No Capítulo 3 aborda-se a conceituação teórica para o entendimento das tecnologias *Kerberos* e LDAP.

A descrição do problema, objetivos deste trabalho, contexto tecnológico e definição do método proposto são apresentados no Capítulo 4.

No Capítulo 5 apresenta-se a implementação da solução e uma discussão sobre os

resultados obtidos em um ambiente real.

No Capítulo 6 são realizadas considerações finais sobre a abordagem definida neste trabalho e apresentam-se propostas para trabalhos futuros.

2 Sistemas de autenticação

Neste capítulo serão descritas técnicas de violação de segurança, e suas conseqüências, utilizadas para obter acesso, sem autorização, a recursos computacionais e informações. Posteriormente, serão descritas premissas necessárias para um sistema seguro, com suas vantagens e desvantagens.

2.1 Histórico

O início da indústria da informática caracterizou-se por computadores de grande porte (*mainframes*), onde os recursos eram centralizados e com acesso restrito, mesmo aos funcionários da organização. Os usuários acessavam tais *mainframes* através de terminais burros conectados via cabo serial. Esse modelo multi-usuário com compartilhamento de tempo foi chamado de *time-sharing* (TANENBAUM, 2003b), conforme demonstrado na Figura 2.1.

Esse modelo tinha muitas vantagens administrativas, pois geralmente só havia um único grande computador para toda uma organização. A administração e manutenção de

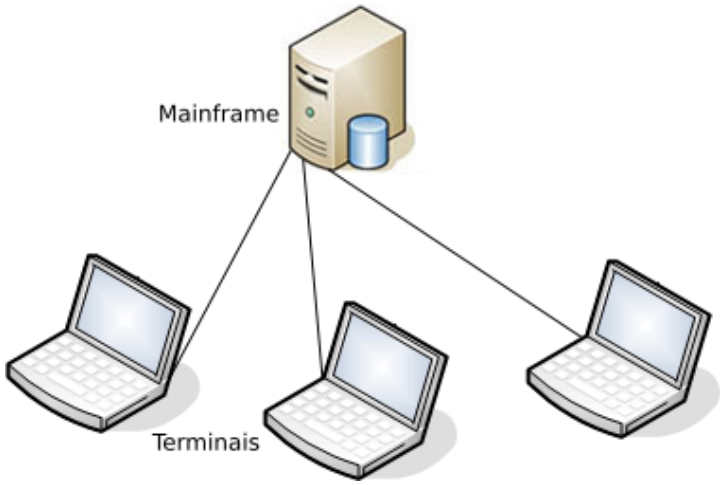

Figura 2.1: Sistemas de tempo compartilhado

políticas de segurança eram facilitadas pela centralização.

Dessa forma, não era necessário encriptar (STALLINGS, 2003) senhas durante a transmissão, pois os terminais dos usuários eram conectados com o *mainframe* por meio de cabos seriais dedicados, onde cada usuário possuía seu próprio terminal e cabo. Isso impossibilitava algumas formas de violação de informações, tal como a escuta.

Com o advento das redes de computadores, que compartilham os meios físicos de transmissão, e a crescente descentralização dos recursos computacionais, a interface entre computadores e usuários mudou significantemente (TANENBAUM, 2003a). Com a evolução da indústria de informática, a redução drástica de tamanho e custo dos computadores possibilitou que todos os usuários pudessem ter seus próprios computadores, conectados a outros através de uma rede. Nesse contexto, os servidores são componentes importantes, os quais são responsáveis por prover serviços de armazenamento, impressão, correio eletrônico, dentre outros.

Esse novo modelo computacional ficou conhecido como "cliente-servidor", conforme apresentado na Figura 2.2.

O maior problema desse modelo é que os usuários passaram a não ser confiáveis. No modelo tradicional de compartilhamento de tempo o administrador do sistema tinha controle total sobre os terminais. Atualmente qualquer usuário poder ter um computador, podendo ainda modificar qualquer parte dos softwares de suas máquinas sem que o administrador da rede saiba e/ou permita. Logo, o administrador não tem mais controle total sobre os terminais.

A partir deste momento surgiu a necessidade de garantir que usuários mal-intencionados não capturem ou, pior ainda, modifiquem secretamente informações enviadas a outros destinatários. Outro problema é certificar que somente pessoas autorizadas tenham acesso as informações e serviços remotos.

2.2 Técnicas de violação da segurança

Um ataque é o sucesso da violação à segurança de sistemas, derivado de uma ameaça inteligente. Um ato inteligente, uma tentativa deliberada (no sentido de um método ou de uma técnica) de evasão dos serviços de segurança e transgressão da política da segurança de um sistema.

Os seguintes ítens descrevem, de forma geral, quatro tipos de ataque (SHIREY, 2000):

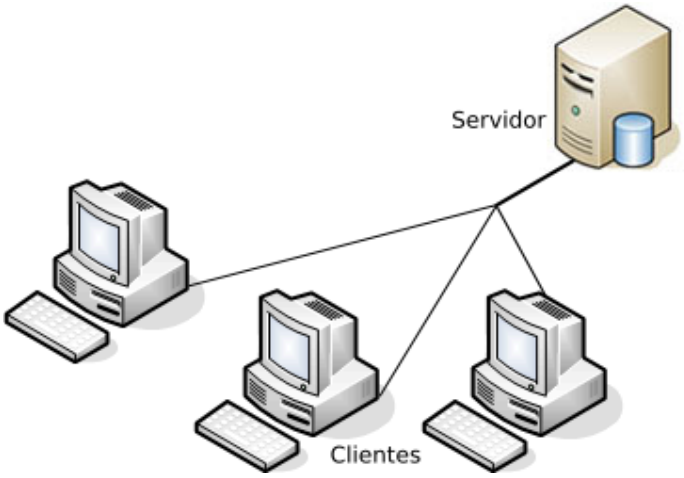

Figura 2.2: Cliente-Servidor

- Interceptação: Acesso não autorizado de um entidade às informações confidenciais. Consiste na captura dos dados que estão trafegando entre remetente e destinatários autorizados. Este ataque afeta a *confidencialidade* das informações;

- Modificação: Circunstância na qual o atacante consegue acesso não autorizado ao sistema e passa a responder como uma entidade autorizada, interceptando, alterando e reenviando mensagens ao destinatário. Este ataque afeta a *confidencialidade* e a *integridade* da mensagem;

- Interrupção: Circunstância na qual o sistema é interrompido ou impedido de funcionar corretamente, indisponibilizando seus serviços. Afeta a *disponibilidade* dos recursos e

- Usurpação: Circunstância ou um evento que resulte no controle de serviços ou de funções de sistema por uma entidade não autorizada, constituindo uma violação à *autenticidade* do sistema.

Segundo MACHADO (2005), a busca por uma proteção contra esses possíveis ataques constitui uma metodologia que contenha regras para utilização da rede e dos serviços da estrutura computacional de uma organização, incluindo-se aplicações encriptadas (STALLINGS, 2003), métodos de autenticação e validação, instalação de *patches* do sistema operacional e dos serviços, aplicação de *firewalls* e, como uma última barreira, a inclusão de sistema de detecção de intrusão. O conjunto de todas essas medidas determina como uma organização gerencia, protege e distribui suas informações e recursos, constituindo uma política de segurança.

2.3 Premissas para um sistema seguro

Segundo GARMAN (2003), autenticação, autorização e auditoria são parte crucial de um método de segurança de rede, cada qual com um propósito distinto.

2.3.1 Autenticação

Autenticação é uma verificação da identidade do usuário. Para ser autenticado, o usuário deve fornecer informações que provem sua identidade. Essas informações podem estar em uma ou mais dentre três categorias: senhas, dispositivos físicos ou biometria.

A primeira categoria, senhas, é a mais usada nas autenticações atualmente (GARMAN, 2003), sendo baseada em senhas para garantir o acesso ao sistema. Esta senha pode ser escolhida pelo próprio usuário ou gerada automaticamente por algum processo aleatório.

A segunda categoria, dispositivos físicos, é menos comum e mais segura. Como exemplo desse tipo de autenticação tem-se o *smart card*, método análogo a uma chave, de forma que somente quem possuir o cartão terá acesso.

A terceira categoria, biometria, refere-se as características biométricas do usuário. Partindo-se do princípio que todos os seres humanos possuem características distintas, podem ser aplicadas medidas biométricas como requisitos para a autenticação de usuários. Como exemplo cita-se: impressão digital, verificação da retina e reconhecimento de voz. Esses métodos são pouco utilizados, pois ainda não estão maduros o suficiente para uso intensivo, em função dos altos custos e pela falta de *softwares* que suportam os dispositivos necessários.

Um sistema de autenticação pode combinar esses fatores, tornando-o mais seguro que um sistema baseado em um único modelo de autenticação.

Uma vez que um servidor de autenticação recebe a informação do usuário que está tentando provar sua identidade, este verifica em sua base de dados se a informação enviada pelo usuário está correta. Caso afirmativo, o usuário é autenticado, em contrário a requisição é negada.

Ataques contra sistemas de autenticação são tipicamente de “força bruta”, onde o atacante gera diversas combinações até que a senha do usuário seja encontrada.

2.3.2 Autorização

É responsável pelo gerenciamento de acessos a um recurso que requer a identificação usuário. Esse processo é executado após a autenticação. A autorização é usualmente realizada por meio da associação da identificação do usuário, aplicando-se uma *access control list* (ACL) (listas de controle de acesso). Essas listas são compostas por informações como grupos aos quais o usuário pertence, regras de utilização, e outras informações que determinam o nível de acesso que o usuário possui em um computador ou recurso da rede.

Dessa forma, a única maneira de se ter uma autorização correta é se o usuário foi previamente autenticado. Caso a autenticação não seja confiável, a autorização perde seu sentido.

2.3.3 Auditoria

A auditoria é aplicada com o objetivo de gravar os resultados da autenticação e autorização em um *log* para futura avaliação do administrador. A autenticação e a autorização visam evitar acessos não autorizados ao sistema. No entanto, mesmo em situações de sucesso de uma intrusão, o administrador pode realizar uma auditoria em seus *logs* para descobrir as vulnerabilidades exploradas pelo atacante.

2.3.4 Privacidade

A privacidade visa ocultar o conteúdo de uma mensagem para evitar o acesso não autorizado de um usuário. Por meio de algoritmos criptográficos, que são funções matemáticas usadas para cifrar e decifrar dados, é possível tornar a informação incompreensível à entidades não envolvidas na comunicação.

A criptografia moderna baseia-se na utilização de chaves secretas, as quais são fundamentais nas operações de cifragem e decifragem. A variedade de valores possíveis para uma chave ("espaço-de-chave") é estritamente ligado ao tamanho da chave, constituindo um fator importante para a segurança, pois o custo de processamento e tempo necessários para desvendar o segredo é muito alto, visto um ataque por força bruta.

Existem dois tipos principais de algoritmos criptográficos utilizados, os algoritmos simétricos e assimétricos (STALLINGS, 2003). Os algoritmos simétricos baseiam-se na utilização de uma mesma chave tanto para o processo de cifragem quanto para o de decifragem de uma informação, conforme o ilustrado na Figura 2.3. Além disso, por se

basearem em processos de transposição e substituição dos dados, estes algoritmos são de baixa complexidade e rápida execução, fazendo com que o tamanho da entrada a ser computada não seja fator de preocupação em questões de performance. Como exemplos de algoritmos simétricos pode-se citar *Data Encryption Standard* (DES), *Triple DES* (3DES), *Advanced Encryption Standard* (AES) (STALLINGS, 2003).

Figura 2.3: Algoritmos simétricos. Adaptado de STALLINGS (2003)

Os algoritmos assimétricos, ou de chave pública, utilizam duas chaves diferentes para os processos de encriptação e decriptação. Uma de conhecimento geral, chave pública, e outra secreta, chave privada, formando um par único, ou seja, para uma chave pública só existe uma chave privada correspondente, e são utilizadas de modo que um texto cifrado com uma chave só possa ser decifrado por outra, e vice-versa, como ilustrado na Figura 2.4.

A medida que aumenta-se o tamanho da chave, pode se tornar inviável, para um período considerável de tempo, a descoberta da chave privada a partir da chave pública. Essas chaves são geradas por funções matemáticas modulares e de exponenciação. Geralmente estes algoritmos possuem custo operacional elevado e não são muito empregados para a cifragem de dados em larga escala. Como exemplos de algoritmos assimétricos cita-se *Rivest-Shamir-Adelman* (RSA), *Diffie-Hellman*, dentre outros (STALLINGS, 2003).

Figura 2.4: Algoritmos assimétricos. Adaptado de STALLINGS (2003)

2.3.5 Integridade

Enquanto a criptografia proporciona a privacidade das informações, a integridade assegura que as mesmas não foram modificadas durante a transmissão. É comum referenciar

os algoritmos que verificam a integridade da mensagem de funções *hashes*.

O propósito das funções *hash* é produzir uma "impressão digital" de um arquivo, mensagem, ou qualquer outro bloco de dados. Em um processo de autenticação de mensagens uma função *hash* "H" deve possuir as seguintes características (STALLINGS, 2003):

- H pode de ser aplicada a um bloco de dados de qualquer tamanho;
- H deve produzir uma saída de tamanho fixo;
- H(x) é relativamente fácil de se computar para qualquer x;
- H(x) = h, para qualquer valor de h, é computacionalmente impossível encontrar x a partir de h. São funções matemáticas unidirecionais e
- Para qualquer valor do bloco de dados x, é computacionalmente impossível encontrar $y \neq x$ com H(y) = H(x).

2.4 Resumo

Neste capítulo foi apresentado um breve histórico sobre a evolução da informática e suas mudanças, no acesso aos recursos computacionais.

O compartilhamento de informações e recursos computacionais, o surgimento das redes de computadores interligando diversos computadores em uma mesma organização ou até mesmo na internet impossibilitaram que administradores tivessem controle total sobre o que acontece nesses ambientes.

Como conseqüência criou-se a necessidade de mecanismos e políticas de segurança que limitassem o acesso indevido de pessoas não autorizadas às informações e recursos privilegiados, garantindo ainda a integridade e autenticidade dos mesmos.

Em função desse contexto, no próximo capítulo serão apresentadas tecnologias direcionadas à segurança e gerenciamento de usuários.

3 *Tecnologias*

Neste capítulo serão abordados tecnologias, destacando-se suas características, que utilizem as premissas de segurança descritas no capítulo anterior, visando evitar formas de acessos não autorizados aos recursos computacionais.

3.1 *Kerberos*

Kerberos (GARMAN, 2003) é um protocolo de comunicação que prove um serviço seguro, de autenticação única, representando a terceira parte confiável de uma autenticação mútua. O projeto foi com o nome de *Athena*, em meados da década de 80, no *Massachusetts Institute of Technology* (MIT).

Uma autenticação única significa que o usuário somente necessita identificar-se uma única vez para acessar todos os recursos de rede suportados pelo *Kerberos*. Isso significa que o usuário foi autenticado pelo *Kerberos* no início de sua sessão e suas credenciais são passadas transparentemente para outros recursos, enquanto a sessão permanecer ativa.

A "terceira parte" refere-se ao fato que o *Kerberos* disponibiliza uma autenticação centralizada para todos os sistemas da rede que confiam de forma inerente em seu serviço. Todas as requisições de autenticação são direcionadas para um servidor *Kerberos* centralizado.

A autenticação mútua assegura que o usuário o qual está solicitando o acesso, é realmente quem ele diz ser, além de verificar se o servidor é realmente o esperado. A autenticação mútua garante a confidencialidade de informações, assegurando-se que a comunicação é genuína.

3.1.1 Terminologias e conceitos

O *Kerberos* envolve muitos conceitos e terminologias, cada qual com um objetivo específico e bem definido em sua arquitetura, que serão demonstrados nas próximas seções.

Principais, instâncias e *realms*

Toda entidade contida em um ambiente *Kerberos*, incluindo-se usuários, computadores e serviços são associados a um principal. Estes constituem nomes únicos organizados hierarquicamente, onde cada principal é associado a uma "chave", que pode ser, por exemplo, uma frase ou senha.

Todo principal inicia-se com um nome de usuário ou serviço, que podem ser seguidos por uma "instância", a qual é usada em duas situações: para principais dos serviços, e um principal especial para fins administrativos.

O principal e sua instância formam um identificador único dentro de um *realm* (domínio). Cada instância do *Kerberos* define um domínio administrativo que é distinto de qualquer outra instalação, isto é definido pelo *Kerberos* como *realm name*, que é *case-sensitive*. Por convenção, o *realm* do *Kerberos* é baseado no domínio *Domain Name System* (DNS) (TANENBAUM, 2003a), em caixa alta. Então, por exemplo, se o domínio DNS é prognus.com.br, o *realm* será PROGNUS.COM.BR.

Dessa forma, a assinatura do principal William, que trabalha na prognus, seria reconhecida da seguinte maneira pelo *Kerberos*:

```
william@PROGNUS.COM.BR
```

O *Key Distribution Center*

O *Kerberos Key Distribution Center* (KDC) é formado por três componentes lógicos: uma base de dados contendo todos os principais e suas respectivas chaves, um *Authentication Server* (AS) e o *Ticket Granting Server* (TGS). Usualmente esses componentes são implementados em um único programa.

Cada *realm* deve conter apenas um KDC, que mantém em sua base de dados informações sobre todos os principais e suas respectivas chaves. Muitas outras implementações do KDC incluem informações adicionais, tais como tempo de vida da senha, última alteração da senha, dentre outras.

É importante que o KDC esteja sempre funcional. Para tanto, utiliza-se replicação do KDC. Não há um padrão definido pelo *Kerberos* para realizar a sincronização das diversas bases de dados em um *realm*, ficando a critério de cada implementação do protocolo.

O *Authentication Server*

O *Authentication Server* (AS) cria um *Ticket Granting Ticket* (TGT), encriptado, para o cliente que acessou o *realm do Kerberos*. O cliente não necessita provar sua identidade ao KDC, pois o TGT enviado ao cliente é encriptado com sua senha de usuário. Quando o cliente desejar acessar o sistema será requisitada tal senha, caso seja a mesma conhecida pelo KDC, será possível decifrar o TGT, comprovando assim sua identidade. É importante que somente o cliente e o KDC conheçam tal senha, afim de garantir a segurança do sistema.

O *Ticket Granting Server*

O TGS cria um *ticket* individual para cada requisição de serviço efetuada pelo usuário. O TGS necessita de duas porções de dados do usuário: um *ticket* de requisição que inclui o nome do principal que representa o serviço requisitado, e um TGT criado pelo AS.

Tickets

Conceitualmente, um *ticket* é um estrutura de dados encriptada, emitida pelo KDC, que inclui uma chave compartilhada de cifra, única para cada sessão. *Tickets* possuem dois propósitos: confirmar a identidade dos participantes da extremidade e estabelecer uma chave de cifra *short-lived* ("vida-curta") para ambas as partes poderem se comunicar com segurança Essa chave é chamada *session key* (chave de sessão).

Os principais campos que o *Kerberos* inclui em todo *ticket* são:

- O nome do principal que está fazendo a requisição (nome de usuário);
- O nome do principal, referente ao serviço;
- Momento em que o *ticket* se tornará válido, e quando será expirado;
- Uma lista de endereços IP que o *ticket* pode ser utilizado e
- Uma chave de sessão compartilhada, utilizada para comunicação das partes envolvidas.

Muitos desses campos são preenchidos pelo KDC, como o tempo de vida do *ticket*. Os outros campos são preenchidos pelo cliente e passados para o KDC, durante a realização de uma requisição. O tempo de vida de um *ticket* é tipicamente curto, entre dez e vinte e quatro horas.

3.1.2 Estrutura funcional

Com o objetivo de demonstrar o funcionamento do *Kerberos*, apresenta-se a seguir analogias escritas por TUNG (2005) e DANTAS (2004), afim de descrever a seqüência de funcionamento e interação entre as entidades envolvidas em uma comunicação utilizando-se *Kerberos*.

Na vida real, um cidadão utiliza rotineiramente uma forma de autenticação, como por exemplo, ao adquirir um ingresso para um evento. Como comprovante de cidadania, a pessoa necessita obter uma Carteira de Identidade (RG), em uma entidade designada para tal finalidade, como o SSP-PR. Um pré-requisito para a obtenção do RG é o cadastro prévio de informações, como certidão de nascimento, nomes dos pais, impressões digitais, etc. Após este processo, o novo cidadão poderá utilizar o RG para diversas finalidades que exigem sua identificação.

Dessa forma, para comprar um ingresso para um determinado evento, é necessário que o comprador se dirija até o local de venda e apresente sua identidade (RG). Posteriormente as informações são validadas e se as mesmas estiverem de acordo, lhe é concedido o ingresso.

Geralmente um ingresso para um evento possui informações de controle, tais como uma data correspondente ao evento, e um número único de identificação. Esses dados servem exclusivamente para que nenhuma outra pessoa possa utilizar o mesmo ingresso.

O *Kerberos* trabalha de forma análoga. Um usuário autêntico de uma rede possui um cadastro em uma entidade chamada *Authentication Server* (AS). De acordo com a analogia, o mesmo corresponde a uma entidade que concede a carteira de identidade (RG) para o cidadão. Depois do cadastro inicial ser efetuado com sucesso, o "candidato a usuário"já é considerado um usuário da rede (da mesma forma que uma pessoa de posse do RG é um cidadão).

Da mesma forma, o local de venda do ingresso é substituído por um serviço chamado *Ticket Granting Service* (TGS), o RG por um *ticket* chamado *Ticket Granting Ticket* (TGT), o ingresso por um "ticket de autorização", e o evento em si por um serviço qualquer

da rede.

Logo, quando o portador do ingresso chega ao evento, o mesmo não necessita mais se identificar com o RG, e sim apenas apresentar o ingresso. Isto é possível porque a organização do evento pressupõe que o RG já foi apresentado no local de compra. Da mesma forma, um *ticket* de autorização em relação a um determinado serviço da rede, não é necessário mais mostrar o TGT, parte-se do princípio que se o usuário tem o *ticket* de autorização, o mesmo foi autenticado anteriormente.

Fluxo de autenticação

Um cenário comum, como a requisição de um determinado cliente a um servidor de *e-mails*, será utilizado para demonstrar o fluxo de autenticação do *Kerberos*.

Primeiramente o cliente envia uma requisição de utilização do determinado serviço ao AS, chamada "AS_REQ". O conteúdo desta requisição consiste, basicamente, no nome do usuário e do serviço requisitado, logo não necessita ser cifrado, por não conter informações sigilosas.

O AS então, provido do nome do usuário, obtém a senha deste em sua base de dados, respondendo para o cliente com dois pacotes de informações. O primeiro é cifrado utilizando sua senha, obtida da base de dados, como chave de encriptação e contém basicamente uma chave de sessão e o nome do serviço requisitado. O segundo pacote é cifrado utilizando uma chave do serviço e contém informações do usuário que o requisitou, além da chave de sessão.

Após a recepção do primeiro pacote, o cliente tenta decifrá-lo utilizando sua senha como chave. Se o processo falhar significa que a senha passada pelo usuário é incorreta, logo o mesmo não tem permissão para obter as informações de tal *ticket*. Nessa condição, o usuário não é autorizado a utilizar o serviço requisitado. O cliente não possui a chave, e nem precisa, para decifrar o segundo pacote, o TGT. O único procedimento que o mesmo deve realizar é encaminhá-lo para o TGS, conforme a Figura 3.1.

Posteriormente, o cliente cria um novo pacote chamado *authenticator*, que contém dados de controle como tempo de expiração (*timestamp*) e dados do usuário que requisitou o serviço. Este *ticket* é encriptado com a chave de sessão. O TGS só obtém às informações do *authenticator* se decifrar apropriadamente o TGT, garantindo assim a autenticidade do servidor.

A principal finalidade do *authenticator* é permitir a reutilização de *tickets*, prevenindo

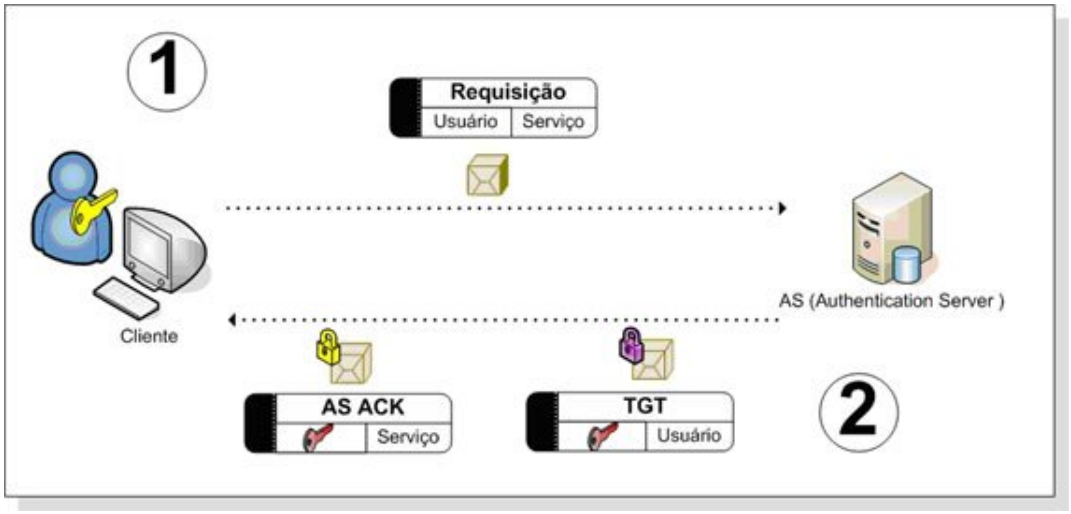

Figura 3.1: *Cliente se comunicando com Authentication Server. Fonte: (DANTAS, 2004)*

ataques como *message replay*. Caso um *ticket* seja obtido na rede por um atacante e reenviado para um servidor com intenções de utilizar um serviço fingindo ser o proprietário do *ticket* (*identity spoofing*), o servidor verifica o *authenticator*. Se o mesmo não existir, a requisição é invalidada. Nada impede que o atacante obtenha tanto o *ticket* quanto o *authenticator* na rede, e reenviá-los, no entanto o *authenticator* contém informações de controle como *timestamp* e ainda um identificador único (*nonce*). Dessa forma, quando um servidor recebe um *authenticator*, o mesmo verifica seu tempo de expiração, e ainda, se foi enviado recentemente por meio de um identificador único.

Quando o TGS recebe o TGT, usando sua chave de serviço, o decifra, obtendo informações, tais como a chave de sessão, que será utilizada para decifrar o *authenticator* e validar suas informações. Caso estejam em conformidade, o cliente é realmente genuíno. Na Figura 3.2 ilustra-se esse processo.

Após isto, o TGS cria e envia o *ticket* necessário para utilização do serviço de *e-mail* da mesma forma que o AS enviou as informações necessárias para uso do TGS.

De posse do *ticket* de utilização do servidor de *e-mails*, o cliente cria um outro *authenticator* com uma chave de sessão passada pelo TGS e os envia para o servidor de *e-mail*. Este verifica a autenticidade sendo responsável pela permissão da utilização de seus serviços. Na Figura 3.3 é apresentado esse fluxo.

3.1.3 Características do ambiente *Kerberos*

Dentre as várias características do *Kerberos*, as que mais se destacam são:

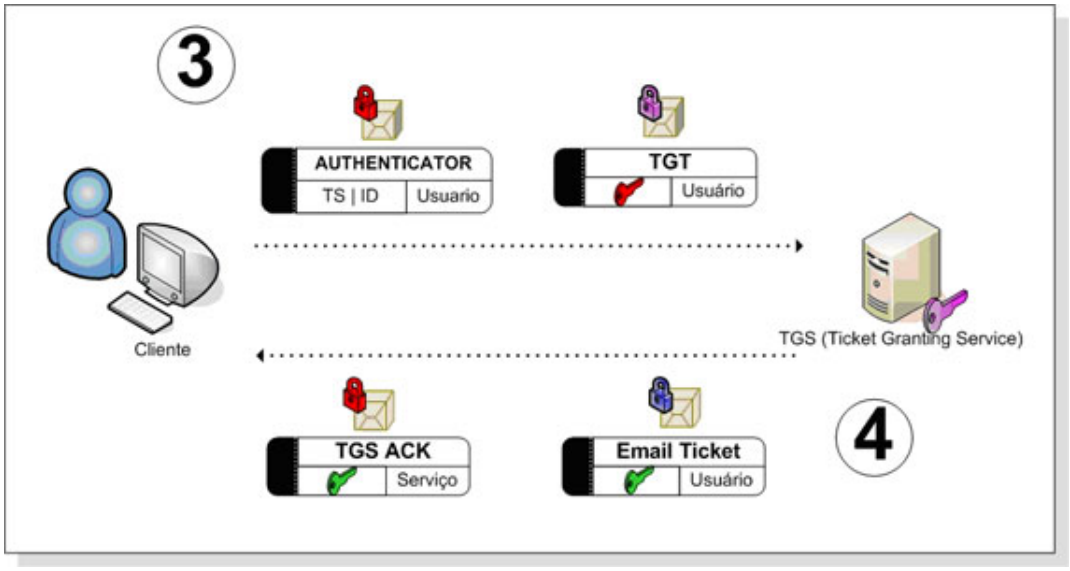

Figura 3.2: *Cliente se comunicando com Ticket Granting Service. Fonte: (DANTAS, 2004)*

- Proporciona maior segurança na rede, pois evita que a senha do usuário transite muitas vezes pela rede e seja "captada" por um *sniffer*;
- Proporciona uma verificação da autenticidade, não só do usuário, mas também do serviço, ou seja, uma autenticação mútua;
- Em sua versão 5 pode-se utilizar vários algoritmos de criptografia (STALLINGS, 2003), tais como DES, 3DES, AES, entre outras;
- Suporta grandes quantidades de clientes e serviços;
- Altamente escalável e replicável e
- Administração simples.

O *Kerberos* está em constante evolução, porém ainda possui pontos negativos (GARMAN, 2003), tais como:

- O sistema deve ser extremamente seguro, pois se um invasor conseguir acesso a base de dados das chaves de encriptação, todo o processo está comprometido;
- As aplicações que desejam utilizar o *Kerberos* devem ser reconstruídas para dar o suporte ao mesmo;
- Não está protegido contra senhas fracas, ou seja, senhas que podem facilmente ser quebradas com um ataque de dicionário;

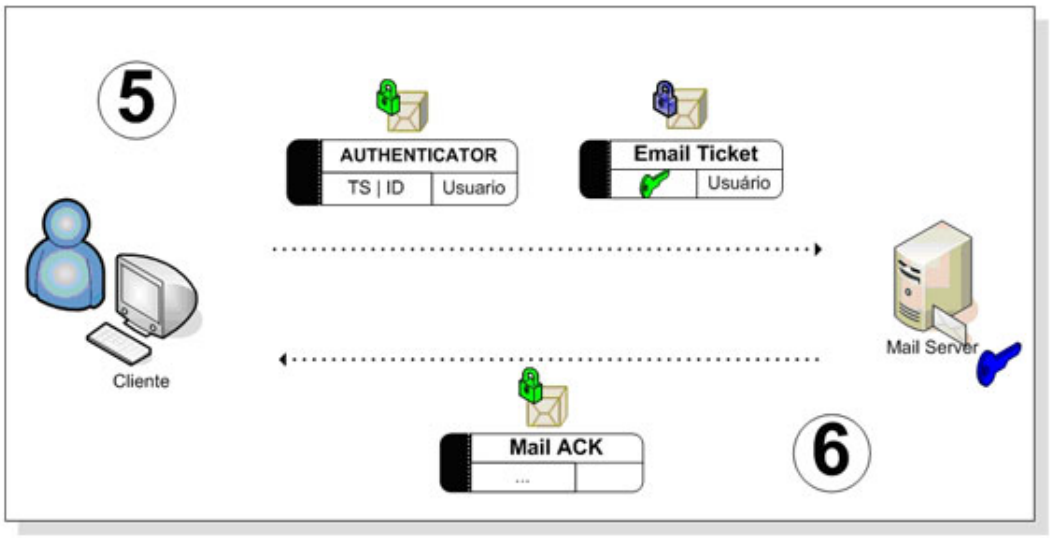

Figura 3.3: *Cliente se comunicando com servidor de e-mails. Fonte: (DANTAS, 2004)*

- Problemas com estações que possuem mais de um IP;
- As diferenças nos relógios dos envolvidos no processo de autenticação pode gerar problemas na validação do *timestamp*;

3.1.4 Considerações sobre *Kerberos*

A segurança é um fator crítico para sistemas computadorizados. Os métodos tradicionais de autenticação, autorização, privacidade e integridade, onde por exemplo, senhas não cifradas trafegam pela rede, não são apropriados para uso nas redes de computadores atuais, onde os atacantes podem monitorar o tráfego da rede para interceptar senhas e informações (STALLINGS, 2003). O *Kerberos* possui um forte esquema de segurança, utilizando criptografia em toda a comunicação, sendo uma ótima solução para tais ambientes (GOUVEIA, 2005).

3.2 *Lightweight Directory Access Protocol* (LDAP)

Quando a *International Organization Standardization* (ISO) e o *Consultative Committee for International Telegraphy and Telephony* (CCITT) se juntaram no início da década de 80 para criar um serviço de mensagens (a série X.400), houve a necessidade de desenvolver um protocolo que organizasse o gerenciamento de um serviço de nomes de forma hierárquico, capaz de suportar grandes quantidades de informação e com uma enorme capacidade de procura por informações. Esse serviço, criado pelas duas institu-

ições, foi apresentado em 1988, denominando-se X.500, juntamente com um conjunto de recomendações e da norma ISO 9594.

O X.500 especificava que a comunicação entre o cliente e o servidor do diretório utilizasse o *Directory Access Protocol* (DAP), que era executado sobre a pilha de protocolos do modelo *Open System Interconnection* (OSI). O fato de o X.500 ser muito complexo e de custo incompatível, levou os pesquisadores da Universidade de Michigan a criar um servidor LDAP, que atuava sobre o TCP/IP (TANENBAUM, 2003a). Em 1993 o LDAP foi então apresentado como alternativa ao protocolo DAP para acesso a Diretórios baseados no modelo X.500.

A disponibilização do código fonte do LDAP na internet possibilitou seu aperfeiçoamento e utilização. Logo o LDAP deixou de ser uma mera alternativa ao DAP do X.500, passando a competir diretamente com o X.500. Em Dezembro de 1997, o *Internet Engineering Task Force* (IETF) lançou a versão 3 do LDAP como proposta padrão para Serviços de Diretório.

Atualmente várias empresas oferecem produtos LDAP, incluindo a Microsoft, Netscape e Novell. A *OpenLDAP Foundation* mantém e disponibiliza uma implementação *Open Source* do Serviço de Diretório LDAP, baseada na Universidade de Michigan (GOUVEIA, 2005).

3.2.1 Terminologias e conceitos

O LDAP é derivado do protocolo X.500, que é um serviço de diretório universal, e possui como objetivo a definição de uma ligação entre serviços de diretório locais, permitindo gerar um diretório global distribuído. O LDAP é executado sobre a arquitetura TCP/IP, cliente-servidor, sendo formada por diversos conceitos, os quais serão demonstrados nas próximas seções.

Diretório

Um diretório (ISQUIERDO, 2001), é uma base de dados especializada, definida de forma hierárquica em uma estrutura de árvore chamada *Directory Information Tree* (DIT), conforme ilustra a Figura 3.4. É otimizada para leitura e suporta sofisticados métodos de pesquisa, onde são armazenadas informações estáticas de objetos, no intuito de proporcionar uma resposta rápida a um enorme volume de consultas. Não existem restrições quanto aos objetos que podem ser guardados em diretórios. Esses objetos podem ser

pessoas, organizações, endereços de *e-mail*, impressoras, dentre outros.

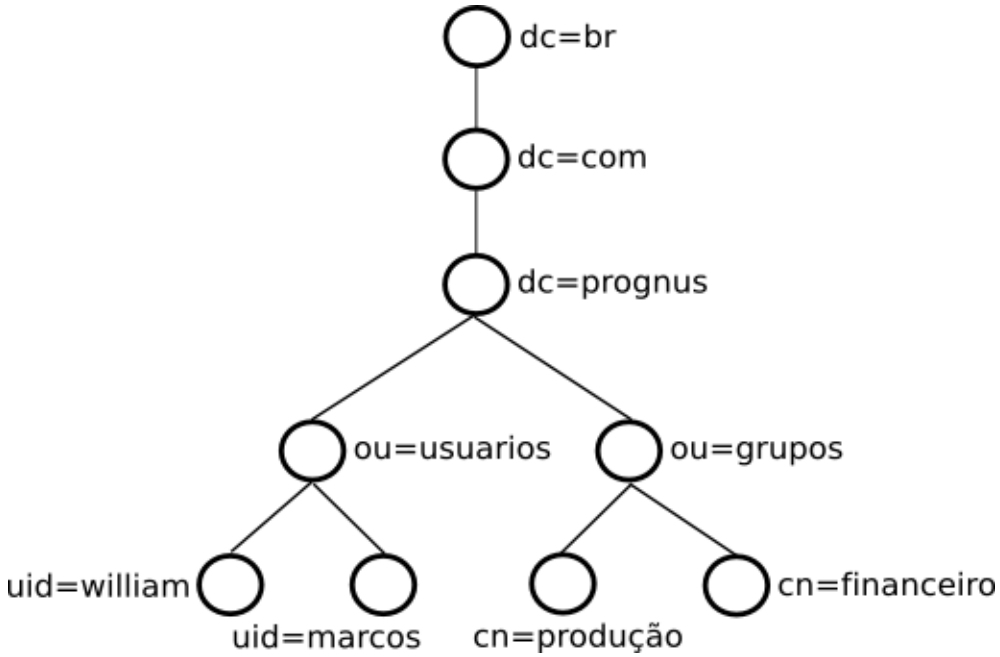

Figura 3.4: Árvore de diretório do LDAP. Adaptado de MARSHALL (2005)

Serviço de Diretório

Um serviço de diretório é uma aplicação projetada para gerir entradas e atributos em um diretório, controlar o acesso aos clientes ou a qualquer outra aplicação que o requisite. Possui mecanismos de pesquisa, remoção e atualização. Um serviço de diretórios possui as seguintes características:

- Flexibilidade: a informação pode possuir diversos tipos;
- Segurança: possui mecanismos de autenticação para o seu acesso interno e externo;
- Escalabilidade e Adaptabilidade: visando atender as necessidades atuais e futuras, tais como aumento de entradas, atributos, dentre outras.

Atributos

Os atributos são identificados por um nome ou acrônimo, possuem um tipo e um ou mais valores. O tipo do atributo está associado a uma sintaxe, a qual define qual tipo de valor pode ser armazenado no atributo. Exemplos de atributos são ilustrados na Tabela 3.1.

Atributo	Descrição
dn	distinguishedName
cn	commonName
sn	surname
gn	givenName
o	organizationName
ou	organizationalUnitName
st	stateOrProvinceName
postalCode	Código Posta
dc	domainComponent
uid	userID

Tabela 3.1: Exemplos de atributos. Fonte: (GOUVEIA, 2005)

Entrada

A unidade básica de informação armazenada em um diretório é denominada "entrada". As entradas são compostas por um conjunto de atributos referentes a um objeto, sendo organizadas em uma estrutura semelhante a uma árvore.

Object Class

Consiste em um conjunto de atributos referentes a uma entrada. Quando uma entrada é definida, são atribuídas um ou mais *object classes*, os quais possuem atributos que podem ser opcionais ou obrigatórios. Existem dois tipos de *object classes*: *structural* e *auxiliary*. Toda a entrada deve ter um *object class* do tipo *structural* e pode ter uma ou mais *object class auxiliary*. A seguir é apresentado um *object class "person"*, retirado do arquivo "core.schema" do OpenLDAP[1].

```
objectclass ( 2.5.6.6 NAME 'person'
DESC 'RFC2256: a person'
SUP top STRUCTURAL
MUST ( sn $ cn )
MAY ( userPassword $ telephoneNumber $ seeAlso $ description ) )
```

Schema

Os *schemas*, em LDAP, permitem manter a consistência dos dados do diretório. Uma característica importante é sua extensibilidade, possibilitando adicionar mais atributos ou

[1]Software de livre distribuição, disponível em www.openldap.org

classes dependendo das necessidades. Os *schemas* definem:

- Quais *object class* podem ser inseridas em um diretório;
- Quais atributos de uma determinada *object class* e
- Os possíveis valores dos atributos.

Se um objeto (entrada) não obedecer às regras do *schema*, o mesmo não poderá ser inserido na base de dados (GOUVEIA, 2005).

Distinguished Names

O *Distinguished Name* (DN) é utilizado para identificar uma entrada de forma não ambígua em um serviço de diretório. Os DN´s são compostos por uma seqüência de *Relative Distinguished Name* (RDN), onde cada RDN corresponde a um ramo na DIT, desde a raiz até a entrada a qual o DN faz referência. Um DN é formado por uma série de RDN´s separados por vírgulas, como demonstrado a seguir:

```
dn: uid=a9767,ou=exactas,dc=ubi,dc=pt
```

Object Identifier

Cada *object class* ou tipo de atributo possui uma sintaxe que identifica o tipo de objeto, isto é, um *Object Identifier* (OID) globalmente único. Os OID´s são representados como strings decimais separados por pontos, representando uma árvore hierárquica, conforme demonstrado na Tabela 3.2.1. A *Internet Assigned Authority* (IANA) é a entidade responsável pelo registro de "sub-árvores" de OID´s.

OID	Utilização
1.1	Organizações OID
1.1.1	Elementos SNMP
1.1.2	Elementos LDAP
1.1.2.1	Tipos de Atributos
1.1.2.1.1	Meus Atributos
1.1.2.2	Object Classes
1.1.2.2.1	Minhas Object Classes

Tabela 3.2: Exemplos de OID's. Fonte: (GOUVEIA, 2005)

***LDAP Data Interchange Format* (LDIF)**

É um formato texto de intercâmbio de informações com o objetivo de:

- Importar dados para o diretório;
- Alterar objetos existentes;
- Efetuar uma cópia do diretório e
- Replicação.

3.2.2 Estrutura funcional

O LDAP define operações para consultar e atualizar o diretório. Operações são fornecidas para adição e remoção de uma entrada do Diretório, modificação de uma entrada existente e modificação do nome de uma entrada. A operação LDAP de busca pode abranger a árvore toda ou apenas um ramo, sem deslocar-se para os demais. Além de especificar, com filtros, quais entradas se deseja encontrar, também é possível especificar quais atributos desta entrada estão sendo procurados. Se os atributos não forem especificados, todos serão retornados.

3.2.3 Características do LDAP

Dentre as características (GOUVEIA, 2005) mais interessantes do protocolo LDAP, destacam-se:

- É um padrão aberto;
- Altamente otimizado para realizar pesquisas de informações, suportando grandes quantidades de requisições;
- Centraliza todas as informações possibilitando muitos benefícios, tais como: único ponto de administração, menor duplicidade de dados, dentre outras;
- Mecanismo de replicação incluído;
- Possui mecanismos de segurança tanto para a autenticação (SASL), como para o troca de dados (SSL/TLS) e
- Atualmente várias aplicações suportam LDAP.

Como características negativas deste protocolo, cita-se:

- O LDAP, em alguns casos, não substitui as bases de dados relacionais;
- Raramente são efetuadas atualizações;
- É conveniente que se guarde somente dados estáticos;
- Não é possível relacionar dois atributos, visto que não se trata de uma base de dados relacional, e sim de uma base de dados estruturada hierarquicamente.

É importante destacar características e potencialidades (KELLERMANN; SILVELLO, 2005) relevantes com relação a implementação livre do protocolo LDAP, o *OpenLDAP*, tais como:

- Suporte a IPV6;
- Autenticação e Segurança: suporta serviços de forte autenticação por meio do uso do *Simple Authentication and Security Layer* (SASL);
- Segurança da camada de transporte: fornece proteção quanto a privacidade, integridade e autenticação, utilizando-se *Transport Layer Security* (TLS) ou de *Secure Sockets Layer* (SSL);
- Flexibilidade da base de dados e
- Altamente configurável.

3.2.4 Considerações sobre o LDAP

O LDAP vem sendo utilizado cada vez mais por administradores de rede, pois suas características e vantagens, em muitas situações compensam as desvantagens. A prova disso é que cada vez mais aplicações e sistemas operacionais possuem suporte para LDAP (GOUVEIA, 2005).

3.3 Resumo

Nesse capítulo abordou-se conceitos, terminologias, funcionamento e características dos protocolos *Kerberos* e LDAP. Esse tema, associado aos conceitos definidos no Capítulo 2 constituem o subsídio para a definição do modelo computacional o qual será apresentado no próximo capítulo.

4 Modelo

Neste capítulo apresenta-se uma descrição do problema da segurança das redes de computadores e seu impacto no tráfego de informações, apresentando ainda algumas tecnologias utilizadas ao logo da história para solucionar, ou ao menos minimizar tais problemas.

Posteriormente apresenta-se a proposta deste trabalho, constituindo uma abordagem envolvendo a utilização das tecnologias *Kerberos* e LDAP.

4.1 Descrição do problema

Conforme TANENBAUM (2003a) muitas empresas possuem um número significativo de computadores distribuídos nos mais distintos departamentos organizacionais. As redes de computadores permitiram a interligação desses computadores e subsidiaram o desenvolvimento de sistemas, assim como o compartilhamento de recursos e informações.

Essa descentralização constitui uma arquitetura Cliente/Servidor e gera a necessidade de mecanismos de segurança aos usuários e aplicações, tais como sistemas de autenticação. As máquinas clientes e servidores são conectadas entre si por uma rede, como ilustra a Figura 4.1.

A arquitetura Cliente/Servidor é amplamente utilizada e constitui a base da grande utilização da rede. Em detalhes, há dois processos envolvidos, um na máquina cliente e um na máquina servidora. A comunicação toma o processo cliente enviando uma mensagem pela rede ao processo servidor. Então, o processo cliente espera por uma mensagem em resposta. Quando o processo servidor recebe a solicitação, então executa o trabalho ou procura pelos dados solicitados enviando de volta uma resposta. Essas mensagens são mostradas na Figura 4.2.

As redes de computadores possibilitaram uma grande revolução no modo como as pessoas e empresas se comunicam e trocam informações, transpondo as limitações físicas

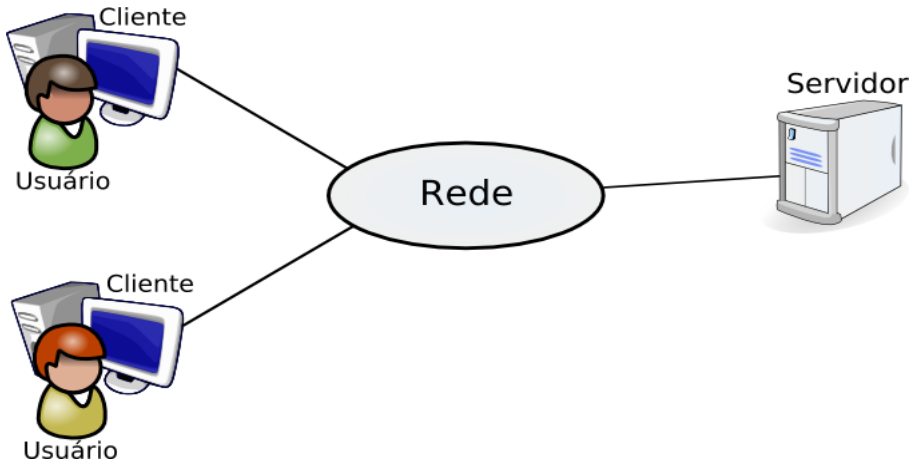

Figura 4.1: *Uma rede com dois clientes e um servidor. Fonte: (TANENBAUM, 2003a)*

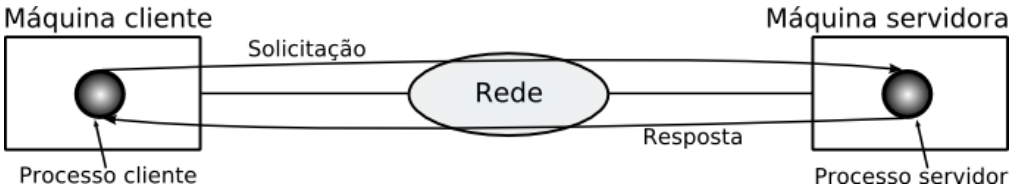

Figura 4.2: *O modelo cliente/servidor envolve solicitações e respostas. Fonte: (TANENBAUM, 2003a)*

das distâncias.

Porém, muitas empresas têm grande quantidade de informações confidenciais *on-line*, tais como segredos comerciais, planos de desenvolvimento de produtos, estratégias de *marketing*, análises financeiras, dentre outras. A revelação dessas informações para um concorrente pode acarretar em terríveis conseqüências. A rápida expansão da internet e das redes de computadores possibilitou ainda que milhões de cidadãos comuns utilizem as redes para executar operações bancárias, fazer compras, acessar dados pessoais. A partir deste ponto, a segurança das redes passa a ser um potencial problema (TANENBAUM, 2003a).

A segurança é um assunto abrangente e inclui inúmeros tipos de problemas, como demonstrados na Seção 2.2 (Página 3) . De maneira geral preocupa-se em garantir que pessoas mal-intencionadas não leiam ou, pior ainda, modifiquem secretamente mensagens enviadas a outros destinatários. Outra preocupação é verificar o acesso a serviços remotos. Também lida com meios para saber se uma mensagem supostamente verdadeira é um trote. A segurança trata de situações em que mensagens legítimas são capturadas e reproduzidas, além de lidar com usuários que tentam negar o fato de terem enviado determinadas mensagens.

A maior parte dos problemas de segurança é causada intencionalmente por pessoas maliciosas que tentam obter algum benefício, chamar a atenção ou prejudicar alguém. Alguns dos invasores mais comuns são listados na Tabela 4.1. Para tornar uma rede segura, com freqüência é necessário lidar com adversários inteligentes, dedicados e, às vezes, bem subsidiados. Segundo TANENBAUM (2003a), os registros policiais mostram que a maioria dos ataques são executados por pessoas ressentidas com a organização a que pertencem.

Tabela 4.1: Algumas pessoas que podem causar problemas de segurança e os motivos para fazê-lo. Fonte: (TANENBAUM, 2003a).

Adversário	**Objetivo**
Estudante	Divertir-se bisbilhotando as mensagens de correio eletrônico de outras pessoas
Cracker	Testar o sistema de segurança de alguém; roubar dados
Representante de vendas	Tentar representar toda a Europa e não apenas Andorra
Executivo	Descobrir a estratégia de marketing do concorrente
Ex-funcionário	Vingar-se por ter sido demitido
Contador	Desviar dinheiro de uma empresa
Corretor de valores	Negar uma promessa feita a um cliente através de uma mensagem de correio eletrônico
Vigarista	Roubar números de cartão de crédito e vendê-los
Espião	Descobrir segredos militares ou industriais de um inimigo
Terrorista	Roubar segredos de armas bacteriológicas

Segundo TANENBAUM (2003a), os problemas de segurança das redes podem ser divididos, como descrito na Seção 2.3 (Página 5), nas seguintes áreas: autenticação, autorização, privacidade e integridade.

É importante ressaltar que a autenticação trata de verificar a identidade do usuário. Os seres humanos possuem capacidade de distinguir uns aos outros, por meio dos sentidos, analisando muitas características específicas de cada indivíduo. É muito fácil reconhecer uma pessoa, por meio da visão, que já vimos anteriormente. Porém, segundo GARMAN (2003), a capacidade atual dos computadores de reconhecer pessoas por meio de padrões biométricos está longe de ser utilizada no mundo real, ainda é uma tecnologia cara em constante desenvolvimento. Enquanto essa tecnologia não atinge um grau de maturidade aceitável, usuários são autenticados por meio de senhas, chaves secretas, que em geral são seqüências de caracteres.

Existem dois problemas com as senhas utilizadas atualmente para autenticação. O primeiro se trata de um problema humano, que não gostam de memorizar senhas com-

plexas, composta por letras e números, as quais são muito mais seguras. Para facilitar, as pessoas geralmente utilizam senhas simples, como seu próprio nome, sua data de aniversário. Conseqüêntemente, a senha pode ser facilmente identificada pelo atacante, o qual se utiliza do usuário de uma pessoa para personificá-la e utilizar os recursos disponíveis na rede.

O segundo é um problema técnico, relacionado a privacidade e integridade. Mesmo que o usuário escolha uma senha realmente secreta, composta por vários caracteres e números, a comunicação entre o computador utilizado pelo usuário e a outra ponta a qual deseja se conectar pode estar em texto puro. Isso se torna um grande problema quando muitos computadores estão utilizando um meio compartilhado de conexão, pois as mensagens trocadas na rede podem ser visualizadas e/ou modificadas por qualquer um que faça parte da rede.

Por fim, a autorização trata validar a um recurso específico da rede, baseado na identidade do usuário. A autorização é executada posteriormente ao processo de autenticação. Seu funcionamento consiste em utilizar listas de controles de acesso (ACL), onde a identidade dos usuários estão associadas a seus direitos de acesso. A autorização inclui informações como grupos a que o usuário pertence e outras informações que determinam qual o nível de acesso o usuário pode ter ao computador ou recurso da rede.

É importante destacar que a habilidade em efetuar uma correta decisão de autorização depende exclusivamente de um sólido mecanismo de autenticação.

Poucas ferramentas/métodos proporcionam todas essas características, sendo necessário a implementação e utilização de várias ferramentas/métodos para propiciar um ambiente que atenda a esses requisitos.

Atualmente, quase todas as ferramentas e métodos de segurança das redes baseiam-se em princípios criptográficos (STALLINGS, 2003). Seja criptografando senhas de usuários ou conexões de rede, como a utilização de *Secure Sockets Layer* (SSL)/*Transport Layer Security* (TLS). Outros métodos, como *firewalls*, são utilizados para filtrar pacotes por meio de reconhecimento de padrões permitindo-se inspecionar pacotes que trafegam entre redes. Os pacotes que atenderem a algum critério são remetidos normalmente, mas os que falharem são descartados. Há ainda a possibilidade de se utilizar *Virtual Private Networks* (VPNs) ou *Redes Privadas Virtuais*, as quais utilizam redes públicas de dados para proporcionar um canal de comunicação isolado virtualmente (TANENBAUM, 2003a). As tecnologias descritas acima não são as únicas, outras podem ser encontradas na bibliografia e muitas mais no mercado.

A implementação de autenticação e a autorização, conjuntamente, criam mais um grande problema, o gerenciamento de informações (como identificação de usuários, senhas, ACLs, dentre outras), necessárias para a correta utilização desses dois processos.

Em empresas pequenas, que utilizam poucos computadores, esse problema é relativamente simples, pois poucos computadores geralmente resultam em poucos usuários, logo poucas informações para serem administradas. Porém, em grandes empresas, esse problema se torna muito mais complexo e difícil de administrar, pois gerenciar informações de milhares de usuários espalhados por diversos setores, filiais, cidades e até países pode se tornar uma tarefa extremamente custosa.

Essa complexidade pode ser multiplicada pela quantidade de aplicações/recursos disponíveis na rede, pois muitas aplicações possuem uma base dados própria com as entradas dos usuários que podem utilizá-la, juntamente com suas respectivas informações de autorização, conforme ilustrado na Figura 4.3.

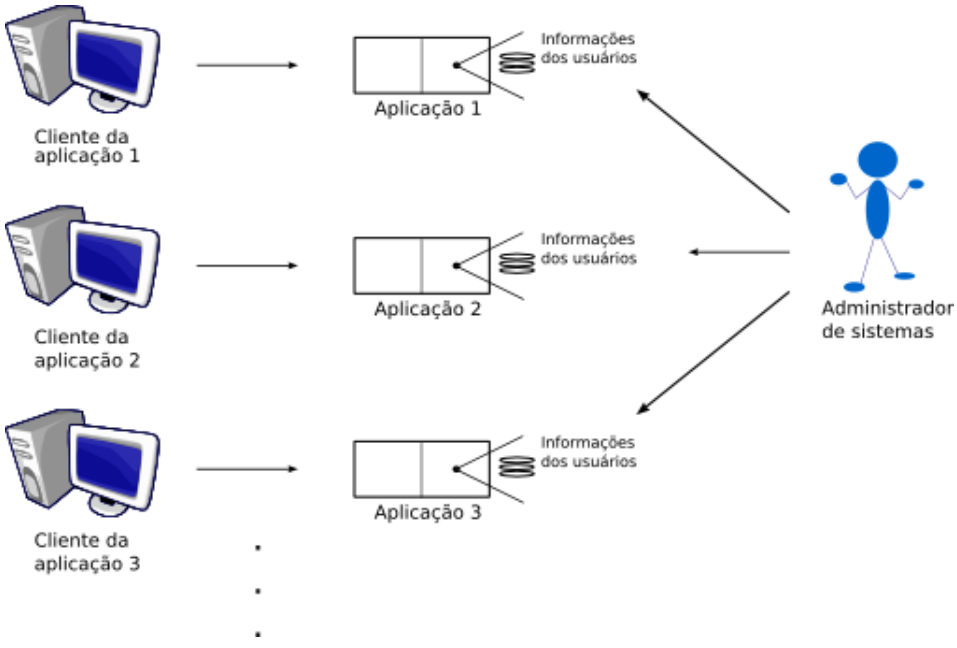

Figura 4.3: *Cada aplicação gerencia os usuários, causando duplicação das informações e muitos problemas ao administrador. Fonte: (HOWES; SMITH; GOOD, 2003).*

Isso causa replicação das informações dos usuários, dificultando posteriormente o seu gerenciamento. Outra consequência é o desconforto ao usuário, pois este terá que memorizar os nomes de usuários e senhas das várias aplicações que utiliza e ainda deverá autenticar-se cada vez que desejar utilizar tais aplicações.

Para de solucionar o problema de replicação dos dados, muitas tecnologias foram

desenvolvidas, como o *Network Information Service*[1] (NIS) (STRUBE, 2003), X.500, LDAP, dentre outras. Essas tecnologias visam centralizar as informações que antes estavam distribuídas pelas várias aplicações e computadores da rede, facilitando o gerenciamento das informações.

Atualmente, as tecnologias de autenticação e autorização estão caminhando para a centralização das informações (GARMAN, 2003), pois assim a manutenção, gerenciamento e segurança das informações são garantidas.

4.2 Objetivos gerais

Existem diversas tecnologias que visam solucionar, ou ao menos minimizar, os problemas de segurança em redes de computadores. *Kerberos* e LDAP são tecnologias presentes há bastante tempo no mercado, mostrando-se muito sólidas e promissoras (GARMAN, 2003). Muitos estudos de caso focam a utilização de alternativas tecnológicas, realçando suas principais características. Algumas dessas propriedades não contribuem diretamente para a segurança das redes, porém proporcionam recursos que podem ser utilizados com outras tecnologias que podem contribuir para uma melhor segurança das redes de computadores.

Desse modo, este trabalho tem como objetivo unir as principais características das tecnologias *Kerberos* e LDAP, afim de proporcionar um ambiente seguro, facilidade e flexibilidade na administração de usuários. Dessa maneira, maior comodidade e segurança para os usuários da rede.

4.3 Solução proposta

A solução proposta tem como objetivo utilizar as tecnologias *Kerberos* e LDAP para proporcionar um ambiente de rede onde as informações sobre usuários e serviços de rede estejam centralizados. Dessa maneira a manutenção, gerenciamento e o processo de autenticação também possam ser centralizados.

Com a utilização do *Kerberos*, não há transporte de senhas (em texto puro ou criptografadas) pela rede, impossibilitando que um atacante obtenha a senha de um usuário, personifique-o e acesse os recursos disponíveis na rede. Em virtude das chaves de sessões

[1]Essa tecnologia está praticamente extinta, poucos fornecedores de sistemas operacionais ainda a suportam.

geradas pelo *Kerberos*, é possível criar uma comunicação segura entre cliente e recurso/serviço disponível na rede, pois todas as mensagens trocadas entre as duas pontas são criptografadas utilizando essa chave, onde somente os dois envolvidos conhecem. Ainda é possível proporcionar uma autenticação mútua, tanto do cliente quanto do recurso/serviço, garantindo a genuinidade dos envolvidos na comunicação. Outra característica provida pelo *Kerberos* é um tempo de vida dos *tickets*, sendo necessária uma única requisição de senha ao usuário. Enquanto o usuário tiver uma sessão válida, este poderá acessar qualquer recurso/serviço disponível na rede, de maneira transparente.

Em complemento, o LDAP proporcionará grandes flexibilidade no gerenciamento e manutenção das informações dos usuários e ainda respostas rápidas às buscas por informações. A arquitetura e os detalhes da solução são descritos nas seções seguintes.

4.4 Arquitetura

A arquitetura da solução consiste basicamente em três componentes chaves, cliente, servidor e autenticador. Os clientes são as aplicações utilizadas para acessar os recursos/serviços disponíveis na rede. O servidores são as aplicações que proporcionam algum recurso/serviço na rede, como um servidor `ftp, ssh, e-mail`, dentre outros. E por fim o autenticador é o centro de tudo, onde todas as informações dos clientes e serviços residem. Todos esses componentes estão interconectados por meio de uma rede de computadores, como mostrado na Figura 4.4.

O autenticador é um elemento fundamental da solução, pois este é responsável pela autenticação entre clientes e serviços. A centralização das informações e da autenticação, possui todas as vantagens discutidas na Seção 4.3. Por outro lado, o autenticador deve atender solicitações corretamente, de modo ininterrupto. É difícil garantir 100% de disponibilidade em um computador, mesmo que se tenha um *hardware* de qualidade e *softwares* configurados corretamente. Para tanto é possível criar uma estrutura redundante, por meio de servidores *master* e *slaves*, onde as informações são replicadas e sincronizadas do *master* para todos os *slaves*. Assim é possível ter vários servidores de autenticação respondendo a muitas requisições diferentes, dividindo-se a carga e possibilitando uma alta disponibilidade.

De maneira geral, quando um usuário deseja acessar um serviço disponível na rede, este deve primeiramente obter as credenciais necessárias com o autenticador, em específico o *Kerberos*. Assim o cliente obedecerá o fluxo de autenticação do *Kerberos*, como descrito

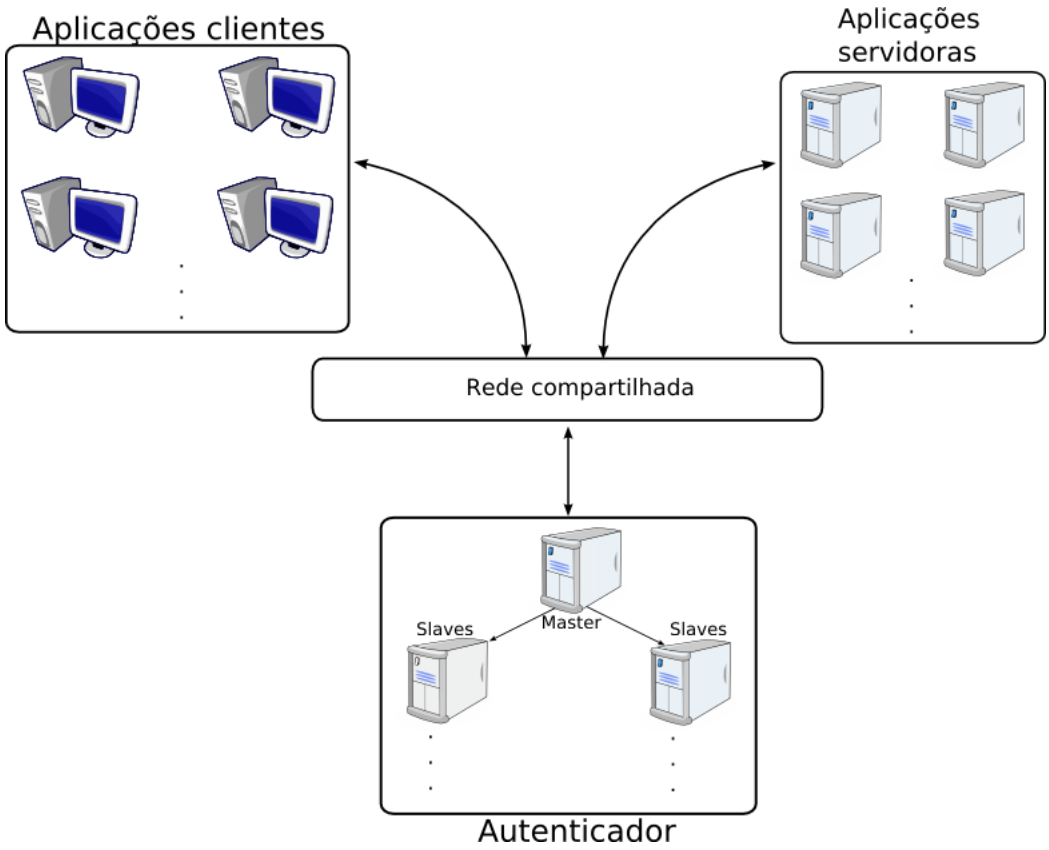

Figura 4.4: Arquitetura da solução.

na Seção 3.1.2 (Página 13).

4.5 Descrição dos componentes da arquitetura

Nesta seção são apresentados os detalhes de cada componente da arquitetura ilustrada na Figura 4.4.

A partir do modelo descrito, utilizou-se uma arquitetura simplificada contendo cada um dos três componentes da solução enfatizando-se os pontos chaves de comunicação entre cada um. A Figura 4.5 exibe, de uma forma geral, a comunicação entre os componentes, que serão detalhadas nas próximas seções.

4.5.1 Aplicação cliente

A aplicação cliente corresponde ao componente que será utilizado por um usuário para acessar algum serviço disponível na rede. Como exemplo será utilizado o serviço de *File Transfer Protocol* (ftp), onde se tem uma aplicação servidora, chamada `ftpd` (ftp *daemon*) que aguarda conexões de clientes `ftp`. Porém, antes que o cliente possa utilizar

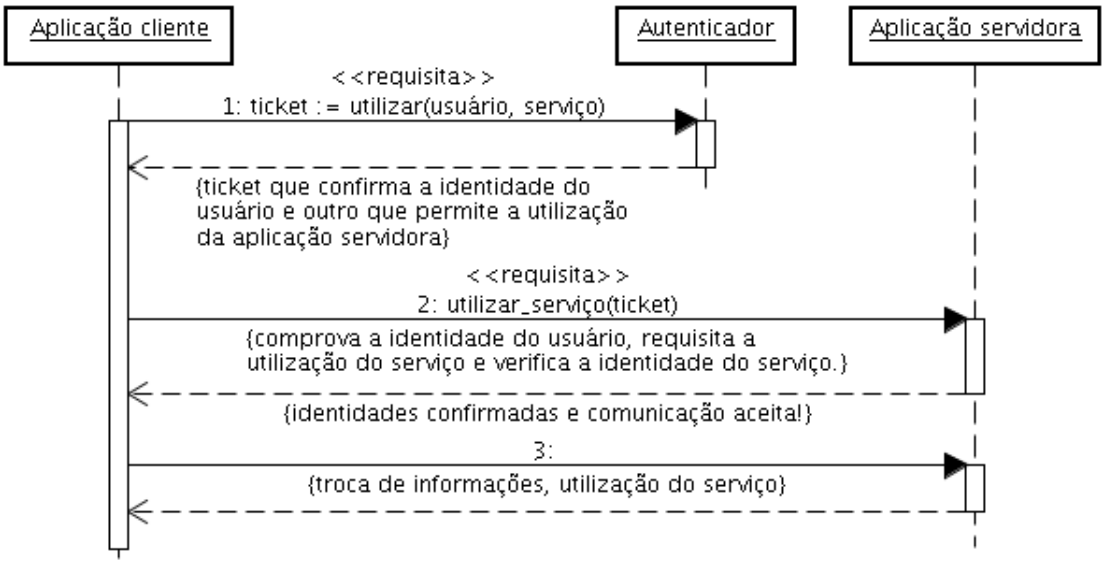

Figura 4.5: Comunicação entre os componentes.

o serviço ele deve ser autenticado.

Para que se possa utilizar o *Kerberos* como autenticador, tanto a aplicação cliente quanto a servidora devem suportar o processo de autenticação via *Kerberos* . Existem muitas maneiras de se fazer isso. O modo mais básico é incluindo-se no código das duas aplicações o suporte a esse processo por meio da *Kerberos 5 API*. É possível também utilizar a *Generic Security Service Application Program Interface* (GSSAPI) (LINN, 2000), a qual proporciona uma interface genérica e um formato de mensagens que encapsula qualquer método de autenticação compatível com a biblioteca GSSAPI, como as implementações do *Kerberos* MIT[2] e *Heimdal*[3]. Implementações do *Kerberos* para o Microsoft *Windows* não suportam a biblioteca GSS-API, sendo utilizada uma biblioteca própria chamada *Security Support Provider Interface* (SSPI).

O método mais comum de autenticação, em sistemas baseados no *UNIX*, se dá por meio do *Pluggable Authentication Modules* (PAM), descrito na Seção 4.5.1 (Página 33).

Após a autenticação correta do usuário, esse já pode utilizar o serviço de `ftp`. Para a correta utilização são necessárias informações sobre o usuário em questão, como seu *username*, os grupos a que pertence, seu identificar único (uid), dentre outras. Tais informações estão centralizadas em uma árvore LDAP, como será descrito na Seção 4.5.3 (Página 37) deste capítulo.

[2]Mais informações em: http://www.mit.edu/ kerberos/
[3]Mais informações em: http://www.pdc.kth.se/heimdal/

Como na autenticação, na autorização é necessário que tanto a aplicação cliente quanto servidora suportem buscas de informações em uma base LDAP. Há diversas maneiras para se fazer isso, sendo a mais básica a inclusão no código das duas aplicações o suporte ao LDAP por meio das bibliotecas LDAP SDKs que, segundo Howes, Smith e Good (2003), proporcionam um acesso completo e de alta-performance a todas as características do LDAP. Até a finalização deste trabalho encontravam-se disponíveis SDKs para C, C++, Perl, Java, Python e Ruby.

Em sistemas baseados no *UNIX* é possível que as aplicações utilizem funções da biblioteca "C"(também chamada de `libc`) para obter informações sobre o usuário. A `libc` por sua vez utiliza o *Name Service Switch* (NSS), descrito na Seção 4.5.1 (Página 35), para especificar como as aplicações clientes, como o `ftp`, obtém informações.

Por fim, utilizando a aplicação de exemplo especificada no começo desta seção, o `ftp`, a qual utiliza o PAM para autenticar usuários e o NSS para obter suas informações, tem-se a seqüência de execução mostrada na Figura 4.6.

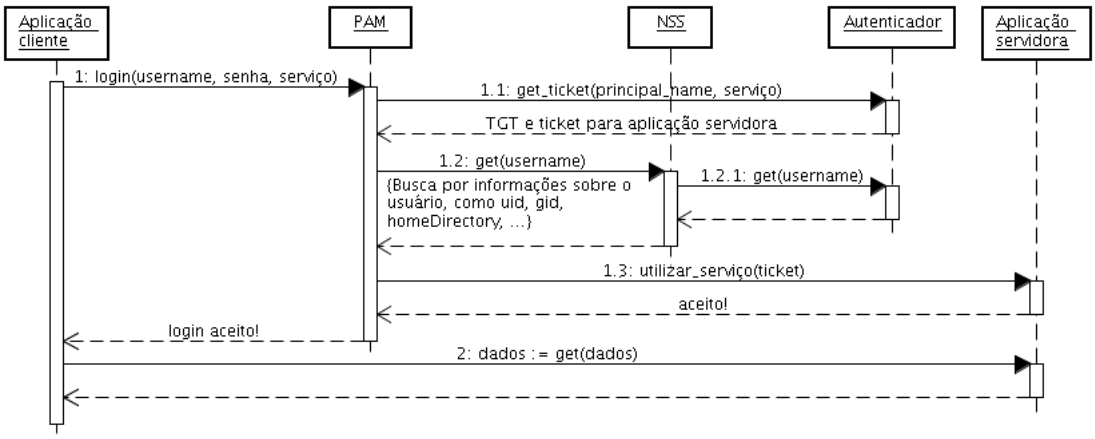

Figura 4.6: Mensagens trocadas pela aplicação cliente até a utilização do serviço.

O processo é iniciado quando o usuário utilizar a aplicação cliente, indicando em qual aplicação servidora deseja se conectar. A aplicação cliente por sua vez requisita ao usuário suas informações de acesso, *username* e senha, que são repassadas ao PAM (descrito na próxima subseção) para que seja confirmada a identidade do usuário. O PAM então envia ao autenticador da aplicação servidora requisitada e o nome do "principal *Kerberos* ", composto pelo ***username*`@realm`** (como mostrado na Seção 3.1.1, página 10).

No autenticador, o *Kerberos* é encarregado de verificar a identidade do usuário, por meio de muitas trocas de mensagens as quais serã explicadas na Seção 4.5.3 (Página 37). Caso a autenticação tenha sucesso, é retornado ao cliente o *ticket* para utilizar

a aplicação requisitada anteriormente e um *Ticket Granting Ticket* , o qual pode ser utilizado posteriormente para utilizar outros serviços sem que seja requisitado novamente a senha ao usuário.

Caso seja necessário informações sobre o usuário em questão é requisitado ao NSS que as encontre. Informações como *User IDentification* (uid) (número inteiro utilizado pelo sistema operacional para identificar o usuário), *Group IDentification* (gid) (utilizado para identificar grupos que o usário pertence), dentre outras. Muitas outras informações podem ser requisitas, dependendo da necessidade de cada aplicação.

O NSS por sua vez, repassa a requisição das informações ao autenticador, especificamente ao LDAP. Essa comunicação é mostrada em detalhes na Seção 4.5.3 (Página 37).

Para que seja feita a comunicação com a aplicação servidora, é necessário enviar o *ticket* (retornado anteriormente pelo autenticador) que comprova a identidade do usuário e ainda verifica a identidade do servidor, garantindo a genuinidade da comunicação. Caso a autenticação da aplicação servidora tenha sucesso, a comunicação é aceita.

Por fim, a aplicação cliente e servidora estão conectadas, permitindo ao usuário a sua utilização.

Pluggable Authentication Modules (PAM)

O PAM (SAMAR; III, 1995) foi desenvolvido pela Sun®[4] com o objetivo de prover uma interface padrão tanto para aplicações quanto para métodos de autenticação. Caso haja a necessidade de se utilizar diversos algoritmos de criptografia para senhas ou um novo método de autenticação, não é necessário alterar cada aplicação que necessite de autenticação.

A partir de então, o aplicativo delega a responsabilidade da autenticação para o PAM. Para que a autenticação seja efetuada por um novo método, basta criar um módulo PAM para que as aplicações automaticamente utilizem, de forma transparente, esse novo recurso. Muitos sistemas operacionais como *GNU/Linux, FreeBSD, Solaris, HP-UX*, dentre outros suportam o PAM (GARMAN, 2003).

O PAM é composto por vários módulos, que podem ser divididos em quatro classes (CONECTIVA, 2004), descritas a seguir:

[4]Mais informações em: http://docs.sun.com

- *auth*: Responsável pelo processo de autenticação do usuário. Módulos dessa classe utilizam o *username* e a senha inseridas pelo usuário, efetuam a verificação por meio de algum método de autenticação e então retornam um código que determina se a senha é válida para o usuário em questão. Sua configuração geralmente se encontra no arquivo `/etc/pam.d/common-auth`[5];

- *account*: Responsável pela manutenção da conta do usuário. Muitas de suas funções se referem a autorização, que determiam se o usuário em questão possui acesso ao serviço que deseja acessar. Sua configuração geralmente se encontra no arquivo `/etc/pam.d/common-account`[5];

- *passwd*: Utilizado para alteração de senha. Pode-se ainda inserir módulos que verifiquem a "qualidade"da senha. Sua configuração geralmente se encontra no arquivo `/etc/pam.d/common-passwd`[5] e

- *session*: Classe encarregada por criar o ambiente do usuário. Sua configuração geralmente se encontra no arquivo `/etc/pam.d/common-session`[5].

Um exemplo de configuração do PAM pode ser visto na Tabela 4.2. A primeira coluna indica a classe que o módulo pertence, a segunda coluna é chamada de campo de controle, descrita após a tabela e por fim o módulo em si.

Tabela 4.2: Exemplos de configurações do PAM.

```
#Arquivo /etc/pam.d/common-auth
auth sufficient pam_krb5.so
auth required pam_unix.so use_first_pass nullok

#Arquivo /etc/pam.d/common-password
password sufficient pam_krb5.so
password required pam_unix.so use_authtok nullok md5 shadow

#Arquivo /etc/pam.d/common-session
session required pam_mkhomedir.so skel=/etc/skel umask=0022
session sufficient pam_krb5.so
session required pam_unix.so
```

O campo de controle define como resultado de cada módulo pode influenciar no resultado do processo de autenticação. Samar e III (1995) define as seguintes diretivas para esse campo:

[5]No S.O. Debian GNU/Linux, em outros esse caminho pode ser diferente.

- *required*: Caso um módulo com essa diretiva falhe outros módulos são executados, no entanto o processo de autenticação irá falhar.
- *requisite*: Uma falha em um módulo, com essa diretiva, irá terminar imediatamente o processo de autenticação;
- *sufficient*: A falha de um módulo não implica a falha do processo de autenticação. Se o módulo falhar, o próximo da classe é executado. Se não houver próximo, então a classe retorna com sucesso. Se, por outro lado, o módulo terminar com sucesso, então os módulos seguintes dessa classe não serão executados e
- *optional*: Os módulos marcados como *optional* praticamente não influenciam no resultado da autenticação. Terão apenas alguma influência caso os módulos anteriores da mesma classe não apresentem um resultado definitivo.

Assim, para que o PAM suporte autenticação via *Kerberos* é necessário instalar o módulo que o suporte, como o `libpam-krb5`[2] e configurar as classes do PAM para utilizar tal módulo, exibido na Tabela 4.2.

Name Service Switch (NSS)

O NSS (SUN, 2002) é um arquivo chamado `nsswitch.conf`, o qual controla como uma máquina ou aplicação cliente obtém informações sobre os usuários.

Cada máquina possui esse arquivo em seu diretório de configurações, `/etc`. Cada linha desse arquivo identifica um tipo particular de informações, como grupos, usuários e suas senhas, seguidos por uma ou mais fontes onde o cliente encontrará tais informações.

A Tabela 4.3 exibe um exemplo de configuração do NSS. A primeira coluna indica o tipo de informação e as colunas seguintes indicam a ordem e onde as informações podem ser encontradas.

Tabela 4.3: Exemplo de configuração do NSS.

```
password  files ldap
group     files ldap
shadow    files ldap
```

Conforme a tabela anterior, o NSS buscará informações, sobre determinado usuário, primeiramente nos arquivos locais, indicados pela diretiva `files`, `/etc/passwd, /etc/group`

[2]Esse é o nome do pacote fornecido pela distribuição Debian GNU/Linux. Para outras distribuições esse nome pode mudar.

`e /etc/shadow`. Caso as informações não sejam encontradas nos arquivos locais, o NSS segue a seqüência e faz a busca na próxima base de dados indicada, `ldap`.

4.5.2 Aplicação servidora

A aplicação servidora corresponde ao componente que proporciona algum serviço à aplicação cliente, como dito na seção anterior. Pouco tem-se a dizer sobre a aplicação servidora, já que seu principal objetivo é oferecer algum recurso ao cliente, o que está fora do escopo deste trabalho. No entanto seguindo-se o exemplo de aplicação definido na seção anterior, tem-se a seguinte seqüência de execução, como exibida na Figura 4.7.

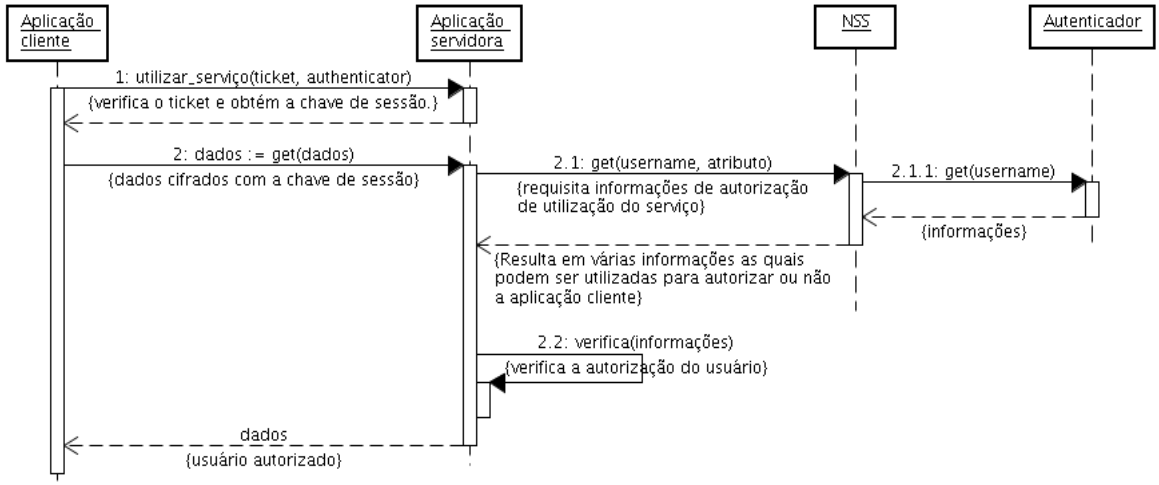

Figura 4.7: Mensagens trocadas pela aplicação servidora.

Para que o cliente estabeleça uma comunicação com a aplicação servidora, deve primeiramente ser autenticado. Esse processo foi discutido anteriormente, na Seção 4.5.1, e o seu funcionamento completo pode ser visto na Seção 3.1.2 (Página 12).

De posse do *ticket* recebido do autenticador, o qual verificou a identidade do usuário, a aplicação cliente o envia à aplicação servidora requisitando o estabelecimento de uma comunicação. Ao recebê-lo, a aplicação servidora utiliza sua chave secreta para decifrar e verificar suas informações. O *ticket* contém informações sobre o cliente e o mais importante, a chave de sessão, que será utilizada para criptografar a comunicação entre cliente e servidor.

Com a comunicação estabelecida, o usuário inicia a utilização dos recursos oferecidos pela aplicação servidora. Levando em consideração o exemplo do serviço de `ftp`, o usuário poderia requisitar algum arquivo do servidor. Antes de processar a requisição do usuário, a aplicação servidora deve verificar se o mesmo possui autorização para fazê-lo.

As informações de autorização, como grupos a que o usuário pertence, dentre outras, se encontram no autenticador, mais especificamente na base LDAP. Para obter tais informações, a aplicação servidora faz uma requisição ao NSS, que por sua vez requisita ao autenticador, como mostrado na Figura 4.7. De posse das informações, a aplicação servidora decide se autoriza ou não a requisição do usuário. Caso o a autorização seja concedida, o aplicação cliente receberá o arquivo solicitado.

4.5.3 Autenticador

O autenticador consiste no núcleo de toda a solução. Seu principal objetivo é fornecer por meio de uma implementação do *Kerberos* um sistema de autenticação de usuários e serviços, e ainda por meio de uma implementação do LDAP, centralizar as informações sobre os usuários e serviços de rede.

MIT *Kerberos* , *Heimdal* são implementações, do *Kerberos* , de código aberto muito conhecidas. Já o *OpenLDAP* [5] é uma implementação de código aberto da tecnologia LDAP. Existem muitas outras implementações, porem a maioria são proprietárias e específicas para alguns ambientes. Cada implementação possui vantagens e desvantagens, dependendo do ambiente que será utilizada.

Assim, para este trabalho foi necessário a utilização do *Heimdal Kerberos* , pois este possibilita a utilização de uma base LDAP, em específico do *OpenLDAP* . A integração entre essas duas aplicações pode ser visualizada na Figura 4.8.

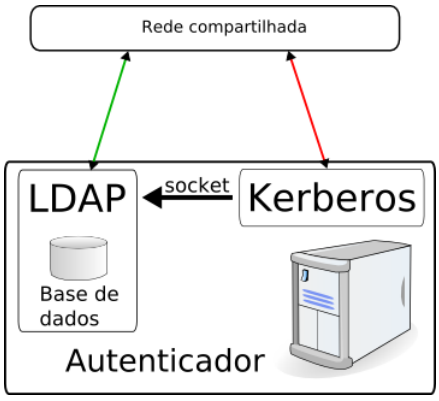

Figura 4.8: Integração *Kerberos* e LDAP.

É importante ressaltar que a comunicação entre o *Kerberos* e o LDAP se dá via *Unix Socket*, o qual pode ser visto detalhadamente em (TANENBAUM, 2003a) e (TANENBAUM,

[5]Mais informações em: www.openldap.org

2003b). As trocas de mensagens ocorrem da mesma forma que em uma conexão TCP, como exibido na Figura 4.9.

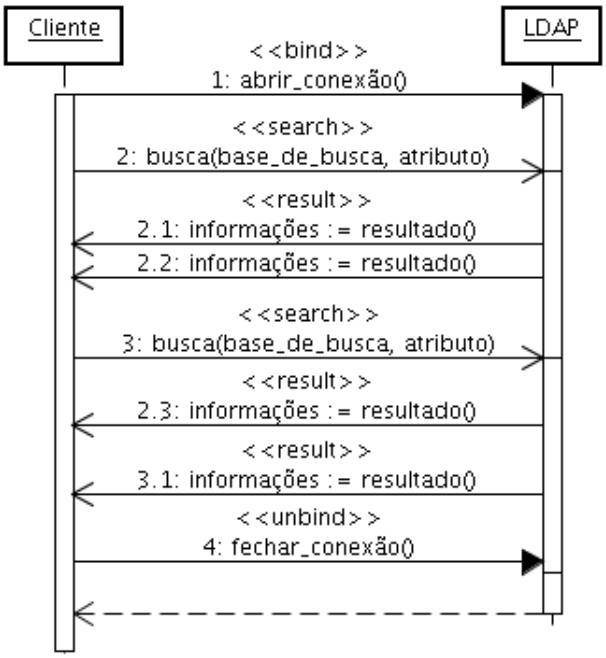

Figura 4.9: Mensagens entre *Kerberos* e LDAP.

Antes que se possa obter qualquer informação da base LDAP, é necessário que o cliente se abra uma conexão no servidor LDAP e posteriormente faça uma busca por informações, indicando a base de busca e por quais atributos deve-se procurar. Podem ser efetuadas várias buscas, onde cada uma pode retornar uma ou mais informações. Após concluídas as buscas, é importante fechar a conexão. A sequência apresentada na Figura 4.9 é válida para o processo de conexão e busca realizada pelo *Kerberos* e por outras aplicações que desejam acessar informações da base LDAP.

Conforme apresentado na Seção 3.1.1 (Página 10), o *Kerberos* é formado por um *Authentication Service* (AS), um *Ticket Granting Server* (TGS) e uma base de dados para armazenar os *principals*, a qual será substituída pelo LDAP, constituindo o *Key Distribution Center* (KDC). O funcionamento em conjunto do *Kerberos* e LDAP pode ser visualizado na Figura 4.10, a qual representa, de maneira geral, o processo de autenticação proporcionado pelo *Kerberos* , como descrito na Seção 3.1.2 (Página 12).

Uma base LDAP pode conter muitas entradas com diversas informações, e nem todas são importantes ou necessárias para o *Kerberos* , mas podem ser para outras aplicações. As *Object Class*, mais informações na Seção 3.2.1 (Página 19), `krb5KDCEntry` e `krb5Principal` possibilitam ao *Kerberos* identificar as entradas da base LDAP que são

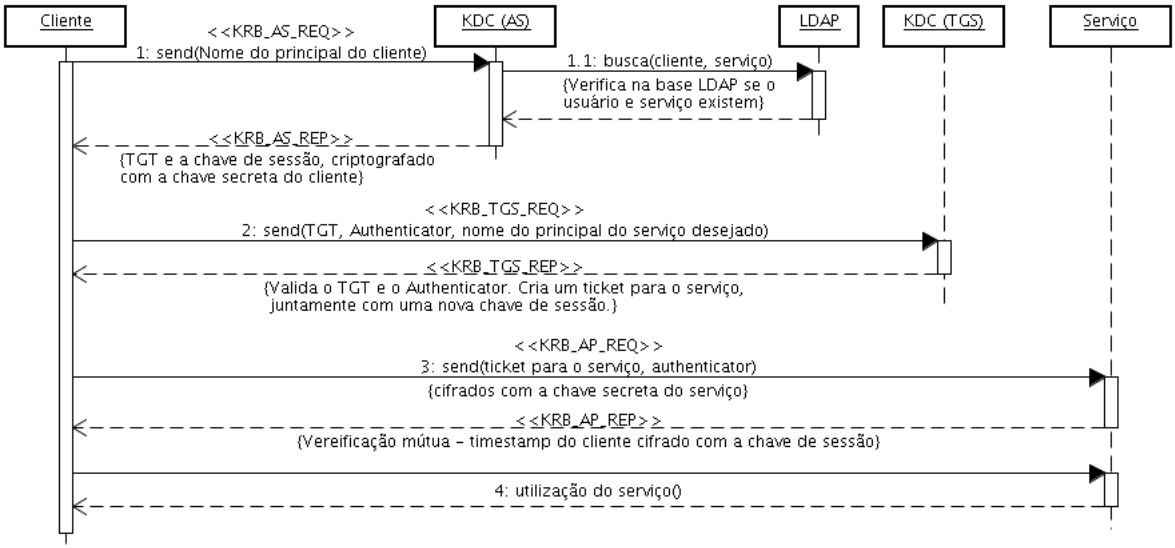

Figura 4.10: Troca de mensagens entre cliente, *Kerberos* , LDAP e serviço.

importantes para o seu funcionamento, como a identificação dos `principals`.

4.6 Resumo

Neste capítulo foi apresentado a problemática da segurança das redes de computadores, e algumas tecnologias utilizadas para solucionar ou ao menos minimizar os impactos causados pela não preocupação com a segurança.

Posteriormente foi apresentada uma solução utilizando as tecnologias *Kerberos* e LDAP afim de proporcionar um serviço de autenticação centralizado, e de fácil gerenciamento, tanto para usuários quanto para serviços.

O próximo capítulo ilustrará um estudo de caso da utilização do modelo em uma empresa, afim de verificar as vantagens e desvantagens do modelo proposto.

5 *Estudo de caso*

Neste capítulo é apresentada uma implementação do modelo proposto no Capítulo 4, afim de analisar e discutir as características positivas e negativas dessa solução.

Nas próximas seções serão apresentadas as configurações do ambiente, características de *hardware*, *software*, rede, dentre outros. Posteriormente serão discutidas as vantagens, desvantagens identificadas.

5.1 Objetivo

O principal objetivo desta implementação utilizar os conceitos apresentados no Capítulo 3 (Página 9) por meio da solução proposta no Capítulo 4.

Desse modo, a implementação visa autenticar usuários, por meio de *login* em uma máquina cliente, criando-se uma sessão que possibilite a utilização transparente do serviço `ssh`, disponível no servidor.

5.2 Ambiente

Conforme descrito no capítulo anterior, o modelo é constituído por três componentes, aplicação cliente, aplicação servidora e autenticador, implementadas em diversas máquinas. É possível concentrar servidor e autenticador na mesma máquina, o que geralmente ocorre em ambientes computacionais pequenos.

Desse modo, a implementação realizada neste trabalho utilizou cinco clientes em máquinas diferentes e outra representando o autenticador e o servidor. Os clientes possuem a seguinte configuração de *hardware*:

- Processador: Intel Pentium 4 2.80 Ghz;
- Disco rígido: IDE/ATA 100 de 30 GBytes;

- Memória RAM: 512 MBytes, DDR 333 e
- Interface de rede: Broadcom NetXtreme Gigabit Ethernet.

O servidor possui a seguinte configuração de *hardware*:

- Processador: Intel Xeon 3.20 Ghz *Hyper-Threading*;
- Disco rígido: Ultra 320 SCSI de 73.4 GBytes;
- Memória RAM: 1024 MBytes, DDR2 e
- Interface de rede: Intel Gigabit Ethernet.

As máquinas estão interconectadas por meio de um *switch ethernet* de 100 Mbits. Os clientes utilizam o sistema operacional *Ubuntu Linux*[1], em sua versão *Breezy* (5.10). A partir da instalação padrão foram necessários alguns *softwares* adicionais para a implementação da solução, como mostrado na Tabela 5.2.

Tabela 5.1: Principais *softwares* utilizados e suas respectivas versões.

Software	**Versão**	**Nome do pacote**
Kernel Linux	2.6.12-9-686	linux-image-2.6.12-9-686
Cliente ssh	3.8.1p1-8	ssh-krb5
Módulo *Kerberos* para PAM	1.0-12	libpam-krb5
LDAP para NSS	238-1ubuntu1	libnss-ldap
Aplicativos clientes para o Heimdal	0.6.3-11ubuntu1	heimdal-clients
Arquivos de configuração do *Kerberos*	1.6	krb5-config

O servidor utiliza o sistema operacional *Debian GNU/Linux*[2], em sua versão estável (*sarge*, 3.1). Instalou-se pacotes adicionais, conforme apresentado na Tabela 5.2.

Tabela 5.2: Principais *softwares* utilizados e suas respectivas versões.

Software	**Versão**	**Nome do pacote**
Kernel Linux	2.6.8-2-686-smp	kernel-image-2.6.8-2-686-smp
Heimdal KDC	0.6.3-10sarge1	heimdal-kdc
Arquivos de configuração do *Kerberos*	1.6	krb5-config
Servidor OpenLDAP	2.2.23-8	slapd

[1]Maiores informações em: www.ubuntulinux.org
[2]Maiores informações em: www.debian.org

5.3 Implementação do cliente

Para que o cliente possa se autenticar no servidor deve-se alterar algumas configurações no sistema. Primeiramente é necessário alterar as configurações do NSS, para que este procure as informações sobre os usuários na base LDAP do servidor. As configurações devem ser alteradas conforme o Anexo A.1.

O processo de *login*, no sistema da máquina cliente utiliza o PAM para autenticação de usuários, portanto é necessário alterar as configurações do PAM, para que este utilize o módulo de autenticação via *Kerberos* . Os arquivos de configuração se encontram no diretório `/etc/pam.d/`. A descrição do PAM foi apresentada na Seção 4.5.1 (Página 33). As configurações devem ser alteradas conforme o Anexo A.2.

Após a configuração do PAM para utilizar o *Kerberos* é necessário indicar a localização do autenticador, o qual possui uma instalação do *Kerberos* , e a qual `realm` o cliente pertence. Essas configurações são utilizadas pelo módulo *Kerberos* do PAM, conforme apresentadas no Anexo A.3.

5.4 Implementação do servidor/autenticador

O autenticador, como descrito no Capítulo 4, possui uma instalação do *Kerberos* e LDAP. Exitem várias implementações dessas duas tecnologias, o *Heimdal Kerberos* e *OpenLDAP* foram escolhidos por duas características importantes:

- São implementações de código aberto, disponíveis para *download* e
- O *Heimdal Kerberos* possibilita utilizar a base de dados do LDAP, em específico do *OpenLDAP* , ao invés da sua própria.

Os principais programas do *OpenLDAP* (OPENLDAP, 2004) são:

- stand-alone LDAP daemon (slapd): Um servidor LDAP que roda em muitas plataformas compatíveis com UNIX, como o Linux.
- *stand-alone LDAP update replication daemon* (slurpd): Juntamente com o slapd proporciona um serviço de replicação da base de dados. É responsável por distribuir as mudanças na principal (*master*) base de dados do slapd para as várias réplicas (*slave*).

Os principais arquivos e diretórios de configuração são:

- `/etc/ldap/schema/`: Diretório contendo os diferentes *schemas*, como descrito na seção 3.2.1 (Página 19), suportados pelo slapd;
- `/etc/default/slapd`: Arquivo de configuração do processo slapd e
- `/etc/ldap/slapd.conf`: Arquivo de configuração do servidor slapd;

Antes de utilizar o *OpenLDAP* é necessário configurá-lo e definir a estrutura da sua árvore, como mostrada na Figura 5.1.

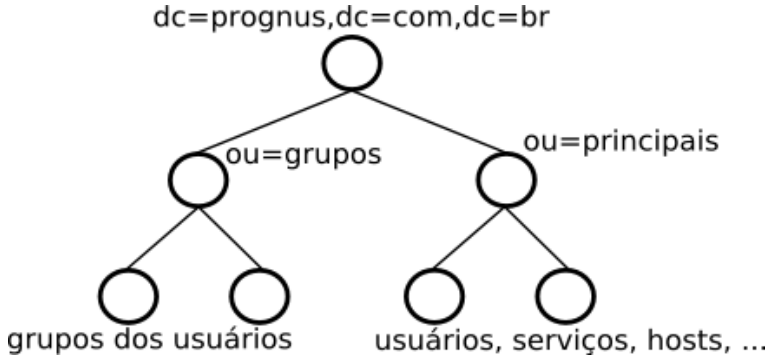

Figura 5.1: Modelo da árvore LDAP da solução.

A raiz da árvore, `dc=prognus,dc=com,dc=br`, por convenção é baseado no domínio de rede, que é `prognus.com.br`. No entanto pode-se utilizar outra palavra. Ramificando-se a árvore, tem-se a unidade organizacional `ou=grupos`, onde serão armazenados os grupos de usuários para melhor organizar e definir autorizações. Na outra ramificação da árvore tem-se a unidade organizacional `ou=principais`, onde encontram-se todas as entidades do sistema, incluindo-se usuários, servidores e entidades específicas do *Heimdal Kerberos* .

Para que o *OpenLDAP* possa ser utilizado com o *Heimdal* são necessárias alterações nos arquivos de configuração. Primeiramente é necessário obter o *schema*[3] que suporte o *Kerberos*, o qual deve ser copiado para o diretório dos *schemas*, como dito anteriormente.

Então, para permitir que o *Heimdal* acesse a base LDAP é necessário ativar no *OpenLDAP* conexões via *Unix Socket*, conforme os Anexos B.1 e B.2. Por fim, deve-se alterar as configurações do *Heimdal Kerberos* conforme o Anexo B.3.

[3]Podendo ser obtido por meio do *link*: http://www.prognus.com.br/ william/downloads/krb5-kdc.schema

5.5 Análise dos resultados

A utilização das tecnologias *Kerberos* e LDAP podem contribuir de maneira satisfatória para um serviço de autenticação de usuários e serviços, conforme demonstrado em vários estudos realizados por empresas e universidades, como Danielsson (2000), Marshall (2005), Heiss (2000), entre outros.

A integração do *Kerberos* com o LDAP resulta em diversas propriedades importantes, as quais podem se tornar vantagens ou desvantagens, de acordo com o ambiente que é implementado. A centralização das informações, por exemplo, proporciona uma redução do custo de manutenção, pois concentra todo esforço dos administradores em uma única base de informações. Outra característica a se destacar é a flexibilidade de customização da base LDAP, possibilitando uma integração em qualquer ambiente e ainda permitindo adaptações para ”acomodar”as informações à evolução dos sistemas utilizados.

A segurança proporcionada pelo *Kerberos* é um dos fatores mais importantes dessa solução, pois seu esquema de troca de mensagens com clientes e servidores impede muitas formas de ataques, que visam obter informações de maneira indevida.

Porém, o *Kerberos* não está protegido contra senhas pobres, escolhidas pelos usuários, e que são facilmente quebradas, por meio de ataques de força bruta ou mesmo engenharia social (GARMAN, 2003), por atacantes. Uma vez que o atacante obtém a senha secreta do usuário, este pode acessar as informações e recursos disponíveis ao usuário atacado. No entanto, este tipo de problema pode ser solucionado por meio de políticas de segurança que impedem que usuários utilizem senhas fracas, e ainda obriga a troca das senhas em um período curto de tempo, como a cada mês.

Segundo Danielsson (2000), que conduziu um estudo sobre a utilização do *OpenLDAP* como base de dados para *Heimdal Kerberos* , há uma queda de desempenho considerável na resposta emitida pelo *Kerberos* da ordem de 300 vezes mais lenta (de aproximadamente 7.9 ms para 2.4 s). Porém o poder de processamento dos computadores e a grande taxa de transferência de dos discos autuais e as vantagens proporcionadas pelo LDAP suprimem essa deficiência.

5.6 Resumo

Neste capítulo foi abordada a implementação da solução proposta no Capítulo 4, utilizando os conceitos discutidos ao longo deste trabalho. Apresenta-se o ambiente da

implementação com as modificações necessárias para que fosse possível utilizar os recursos provindos da integração do *Kerberos* e LDAP.

Posteriormente foi discutido suas principais vantagens e desvantagens. Esta solução pode apresentar mais vantagens ou desvantagens de acordo com o ambiente em que se deseja utilizá-la.

6 Conclusões e Projetos Futuros

Conforme apresentado nesse trabalho, as redes de computadores e os sistemas de informação estão presentes nas mais diversas organizações e são cada vez mais vitais nos processos organizacionais.

Por meio deste cenário enfatiza-se a importância de assegurar a segurança e a integridade das informações e recursos. A aplicação de técnicas para a autenticação e autorização de usuários e serviços é uma atividade essencial para a proteção desses ambientes.

Nesse trabalho apresentou-se uma abordagem de autenticação e autorização híbrida aplicando as tecnologias *Kerberos* e LDAP. Por meio da aplicação do modelo proposto foi possível agregar-se as principais vantagens apresentadas pelo *Kerberos* e LDAP, tais como centralização, flexibilização das informações dos usuários e serviços, proporcionando um serviço de autenticação seguro e transparente ao usuário.

A partir dos resultados obtidos, propõe-se os seguintes trabalhos futuros:

- Realização de um estudo de performance da solução para analisar o impacto sobre o tráfego de dados na rede;
- Realizar um comparativo, considerando os atributos performance e segurança, com outros métodos de autenticação;
- Realizar estudos de caso em outros ambientes, visando analisar o comportamento do modelo em distintos ambientes computacionais e
- Utilizar o modelo de forma distribuída, tendo-se a necessidade de segmetar e replicar o serviço de autenticação, visando aumentar sua disponibilidade no ambiente.

Referências

BRANCO, R. R. Segurança da informação - ldap. 2005. Disponível em: <http://www.firewalls.com.br>. Acesso em: 10 jul 2005.

CONECTIVA. *Guia do Servidor Conectiva Linux*. [S.l.], 2004. Disponível em: <http://www.conectiva.com/doc/livros/online/>. Acesso em: 14 out 2005.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. *Distributed Systems: Concepts and Design*. 3. ed. São Paulo, Brasil: Addison-Wesley, 2001.

DANIELSSON, J. Different database methods in heimdal. 2000. Disponível em: <http://www.chips.chalmers.se/danielsson.et.al.pdf>. Acesso em: 15 out 2005.

DANTAS, B. S. Desenvolvendo web services seguros com kerberos. 2004. Disponível em: <http://www.linhadecodigo.com.br>. Acesso em: 20 maio 2005.

GARMAN, J. *Kerberos: The Definitive Guide*. Sebastopol, EUA: O'Relly, 2003.

GOUVEIA, B. Ldap para iniciantes. 2005. Disponível em: <http://www.ldap.org.br/>. Acesso em: 10 jun 2005.

HEISS, J. Replacing nis with kerberos and ldap. 2000. Disponível em: <http://www.ofb.net/ jheiss/krbldap/kerberos_and_ldap.html>. Acesso em: 15 out 2005.

HOWES, T. A.; SMITH, M. C.; GOOD, G. S. *Understanding and Deploying LDAP Directory Services*. 2. ed. Boston, EUA: Addison Wesley, 2003.

ISQUIERDO, G. S. *Integração do Serviço de Diretório LDAP com o Serviço de Nomes CORBA*. Dissertação (Mestrado em Ciência da Computação) — Universiade de São Paulo, São Paulo, 2001.

KANIES, L. A. An introducion to ldap. 2001. Disponível em: <http://www.onlamp.com/pub/a/onlamp/2001/08/16/ldap.html>. Acesso em: 10 jun 2005.

KELLERMANN, G. A.; SILVELLO, J. C. *Manual OpenLDAP*. [S.l.], 2005. Disponível em: <http://paginas.terra.com.br/informatica/silvello/openldap/>. Acesso em: 11 jun 2005.

LINN, J. *Generic Security Service Application Program Interface*. [S.l.], 2000. Disponível em: <http://www.opengroup.org/tech/rfc/rfc86.0.html>. Acesso em: 25 out 2005.

MACHADO, R. B. *UMA ABORDAGEM DE DETECÇÃO DE INTRUSÃO BASEADA EM SISTEMAS IMUNOLÓGICOS ARTIFICIAIS E AGENTES MÓVEIS*. Dissertação (Mestrado em Ciência da Computação) — Universiade Federal de Santa Catarina, Florianópolis, 2005.

MARSHALL, B. *Introduction to LDAP*. [S.l.], 2005. Disponível em: <http://quark.humbug.org.au/publications/ldap/ldap-tut.html>. Acesso em: 11 jun 2005.

MARSHALL, B. System authentication using ldap. 2005. Disponível em: <http://www.padl.com/Contents/Documentation.html>. Acesso em: 15 out 2005.

NEUMAN, C. et al. *RFC 4120 - The Kerberos Network Authentication Service (V5)*. [S.l.], 2005. Disponível em: <http://www.rfc-archive.org/getrfc.php?rfc=4120>. Acesso em: 20 set 2005.

NUNES, R.; ANTÔNIO, R.; BERNARDO, R. Estudo do protocolo ldap e desenvolvimento de uma pequena api. 2003. Disponível em: <http://www.deis.isec.pt/RSD/AULAS/0203/RD2/trabalhos.htm>. Acesso em: 10 jun 2005.

OPENLDAP, F. *OpenLDAP 2.2 Administrator's Guide*. [S.l.], 2004. Disponível em: <http://www.openldap.org/doc/admin22/>. Acesso em: 14 out 2005.

SAMAR, V.; III, R. J. S. *PLUGGABLE AUTHENTICATION MODULES (PAM)*. [S.l.], 1995. Disponível em: <http://www.opengroup.org/tech/rfc/rfc86.0.html>. Acesso em: 14 out 2005.

SHIREY, R. W. *RFC 2828, Internet Security Glossary*. [S.l.], 2000. Disponível em: <http://www.rfc-archive.org/getrfc.php?rfc=2828>. Acesso em: 10 jun 2005.

STALLINGS, W. *Cryptography and Network Security: Principies and Praticies*. 3. ed. New Jersey, EUA: Prentice Hall, 2003.

STRUBE, A. O. *O HOWTO do NIS(YP)/NYS/NIS+*. [S.l.], 2003. Disponível em: <http://www.surak.eti.br/nis-HOWTO/>. Acesso em: 25 out 2005.

SUN, M. *The Name Service Switch*. [S.l.], 2002. Disponível em: <http://docsun.cites.uiuc.edu/>. Acesso em: 14 out 2005.

TANENBAUM, A. S. *Redes de Computadores*. 4. ed. Rio de Janeiro, Brasil: Ed. Campus, 2003.

TANENBAUM, A. S. *Sistemas Operacionais Modernos*. 2. ed. São Paulo, Brasil: Prentice Hall, 2003.

TUNG, B. *The Moron's Guide to Kerberos*. [S.l.], 2005. Disponível em: <http://gost.isi.edu/brian/security/kerberos.html>. Acesso em: 5 maio 2005.

ANEXO A – Arquivos de configuração do Cliente

A.1 Alterações no NSS

Tabela 1.1: Alterações no arquivo `/etc/nsswitch.conf`.

```
password        files ldap
group           files ldap
shadow          files ldap
```

A.2 Configurações do PAM

Tabela 1.2: Configurações do PAM.

```
#Arquivo /etc/pam.d/common-auth
auth sufficient pam_krb5.so
auth required pam_unix.so use_first_pass nullok

#Arquivo /etc/pam.d/common-password
password        sufficient      pam_krb5.so
password required        pam_unix.so use_authtok nullok md5 shadow

#Arquivo /etc/pam.d/common-session
session required pam_mkhomedir.so skel=/etc/skel umask=0022
session sufficient      pam_krb5.so
session required        pam_unix.so
```

A.3 Configurações do *Kerberos*

Tabela 1.3: Alterações no arquivo `/etc/krb5.conf`.

```
#Valores padrões para a biblioteca do Kerberos
[libdefaults]
        #Identifica o realm padrão que o cliente usará
        default_realm = PROGNUS

#Descreve onde estão os servidores de cada realm
[realms]
        #Nome do realm
        PROGNUS = {
                #Nome do computador que possui uma instalação do KDC
                kdc = vader.prognus
                #Nome do computador que roda o servidor administração
                admin_server = vader.prognus
        }

#Proporciona a tradução de um domínio para um realm, onde o que
#pertencer aodomínio "prognus" e "prognus.com.br" pertencerá também
#ao realm PROGNUS
[domain_realm]
        prognus = PROGNUS
        .prognus = PROGNUS
        prognus.com.br = PROGNUS
        .prognus.com.br = PROGNUS
```

É importante que o DNS (TANENBAUM, 2003a) utilizado pelo cliente, tenha a entrada `vader.prognus` que aponta para o autenticador.

ANEXO B – Arquivos de configuração do Servidor

B.1 Arquivo de configuração do *OpenLDAP*

Tabela 2.4: Alterações no arquivo `/etc/default/slapd`.

```
SLAPD_SERVICES="ldap:/// ldapi:///"}
```

B.2 Arquivo de configuração do *OpenLDAP*

Tabela 2.5: Alterações no arquivo `/etc/ldap/slapd.conf`.

```
#Indica ao slapd qual schema carregar
include         /etc/ldap/schema/krb5-kdc.schema

#Raiz da árvore LDAP
suffix          "dc=prognus,dc=com,dc=br"

#ACL para acesso a base de dados
access to *
        by sockurl.regex="^ldapi:///$" write
        by self write
        by * read \hline
```

B.3 Aterações nas configurações no *Heimdal Kerberos*

Tabela 2.6: Alterações no arquivo `/etc/krb5.conf`.

```
#Valores padrões para a biblioteca do Kerberos
[libdefaults]
        #Identifica o realm padrão que o cliente usará
        default_realm = PROGNUS

#Ativa/Desativa algumas propriedades dos tickets
[appdefaults]
        encrypt = true
        forwardable = true
        proxiable = true
        renewable = true

#Descreve onde estão os servidores de cada realm
[realms]
        #Nome do realm
        PROGNUS = {
                #Nome do computador que possui uma instalação do KDC
                kdc = vader.prognus
                #Nome do computador que roda o servidor administração
                admin_server = vader.prognus
        }

#Proporciona a tradução de um domínio para um realm, onde o que
#pertencer ao domínio "prognus" e "prognus.com.br pertencerá também
#ao realm PROGNUS
[domain_realm]
        prognus = PROGNUS
        .prognus = PROGNUS
        prognus.com.br = PROGNUS
        .prognus.com.br = PROGNUS
[kdc]
        #Indica onde será gravado o arquivo de "log" das ações
        #do KDC
        logging = FILE:/var/log/heimdal-kdc.log
        #Configura a base de dados do KDC, é aqui onde Kerberos
        #e LDAP se unem!
        database = {
                #Indica em qual ramificação da árvore LDAP o Heimdal
                #adicionará e buscará por informações.
                dbname = ldap:ou=principais,dc=prognus,dc=com,dc=br
                #Arquivo que contém a chave secreta que criptografa
                #as informações da base de dados
                mkey_file = /var/lib/heimdal-kdc/m-key
                #Indica qual realm terá suas informações na base
                realm = PROGNUS
        }
```