UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
ESPECIALIZAÇÃO EM REDES DE COMPUTADORES

AMILCAR MICHELIN

IMPLANTANDO CONTROLADOR DE DOMÍNIO COM AUTENTICAÇÃO LDAP OFFLINE COM USO DE SOFTWARE LIVRE

MONOGRAFIA DE ESPECIALIZAÇÃO

DOIS VIZINHOS

2015

AMILCAR MICHELIN

IMPLANTANDO CONTROLADOR DE DOMÍNIO COM AUTENTICAÇÃO LDAP OFFLINE COM USO DE SOFTWARE LIVRE

Trabalho de Conclusão de Curso, apresentado ao II Curso de Especialização em Redes de Computadores – Configuração e Gerenciamento de Servidores e Equipamentos de Redes, da Universidade Tecnológica Federal do Paraná, câmpus Pato Branco, como requisito parcial para obtenção do título de Especialista.

Orientador: M.e Marcos Talau

DOIS VIZINHOS

2015

TERMO DE APROVAÇÃO

Implantando Controlador de Domínio com autenticação LDAP Offline

com uso de Software Livre

por

Amilcar Michelin

Esta monografia foi apresentada às 14h00min do dia 26 de outubro de 2015, como requisito parcial para obtenção do título de ESPECIALISTA, no II Curso de Especialização em Redes de Computadores – Configuração e Gerenciamento de Servidores e Equipamentos de Redes, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho **aprovado**.

Banca Examinadora

Prof. M.Sc. Marcos Talau
Orientador / UTFPR-DV

Prof. M.Sc. Rober Mayer
UTFPR-PB

Prof. Dr. Éden Ricardo Dosciatti
UTFPR-PB

Prof. Dr. Fábio Favarim
Coordenador do II Curso de Especialização
em Redes de Computadores

Dedico essa nova conquista à minha esposa, pelo incentivo, e pela compreensão nas minhas ausências.

AGRADECIMENTOS

Agradeço a Instituição UTFPR e a coordenação do curso pela oportunidade, e o corpo docente e também ao meu orientador.

“O sucesso nasce do querer, da determinação e persistência em se chegar a um objetivo. Mesmo não atingindo o alvo, quem busca e vence obstáculos, no mínimo fará coisas admiráveis.” (José de Alencar)

RESUMO

MICHELIN, Amilcar. IMPLANTANDO CONTROLADOR DE DOMÍNIO COM AUTENTICAÇÃO LDAP OFFLINE COM USO DE SOFTWARE LIVRE. 47 f. Trabalho de Conclusão de Curso, – II Curso de Especialização em Redes de Computadores – Configuração e Gerenciamento de Servidores e Equipamentos de Redes,, Universidade Tecnológica Federal do Paraná, câmpus Pato Branco,. Dois Vizinhos, 2015.

Uma das maiores vantagens da utilização do serviço de controlador de domínio, com a utilização do protocolo LDAP, é a unificação do login dos usuários de diversos serviços da rede. O controlador de domínio pode ser implementado com a utilização de sistemas GNU/Linux; isto normalmente é feito com o uso do software SAMBA. Implementar um controlador de domínio com LDAP no GNU/Linux exige grande experiência do usuário, principalmente para a configuração do SAMBA e do LDAP, além de apresentar alguns problemas, como a necessidade de redigitação de senha e utilização de bind não anônimo. Este trabalho apresenta uma nova forma para se utilizar o SAMBA com o LDAP, onde o LDAP é usado de forma indireta. O método demonstrou ser uma alternativa viável e de fácil implementação, podendo ser utilizado em redes onde a segurança do login na estação Windows não é um fator de risco.

Palavras-chave: Autenticação em domínio, SAMBA, OpenLDAP.

ABSTRACT

MICHELIN, Amilcar. DOMAIN CONTROLLER WITH OFFLINE LDAP AUTHENTICATION USING FREE SOFTWARE. 47 f. Trabalho de Conclusão de Curso, – II Curso de Especialização em Redes de Computadores – Configuração e Gerenciamento de Servidores e Equipamentos de Redes,, Universidade Tecnológica Federal do Paraná, câmpus Pato Branco,. Dois Vizinhos, 2015.

One of the biggest advantages with the use of domain controller service, with LDAP, is the unification of user login of network services. The domain controller can be implemented using the GNU/Linux; this is normally done using the SAMBA software. Implement a domain controller with LDAP in GNU/Linux requires high user experience, especially for the configuration of SAMBA and LDAP, and present some problems, such as: the need for retyping the password, and use of non-anonymous bind. This paper presents a new way to use SAMBA with LDAP, where LDAP is used indirectly. The method proved to be a viable and easy to implement, it can be used in networks where the security of the Windows login season is not a risk factor.

Keywords: Domain authentication, SAMBA, OpenLDAP.

LISTA DE FIGURAS

LISTA DE SIGLAS

CIFS	Common Internet File System
DIT	Directory Information Tree
DNS	Domain name service
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
NetBEUI	acrônimo para NetBIOS Extended User Interface
NetBIOS	(Network Basic Input Output
NTLM	NT LAN Manager
PAM	Pluggable Authentication Module
PDB	Password database backends
PDC	Primary Domain Controller
RDN	Relative Distinguished Name
SAM	Security Accounts Manager
SMB	Server Message Block
TCP/IP	Transfer control protocol / Internet protocol
WINS	Windows Internet Name Service

SUMÁRIO

1 INTRODUÇÃO

A expansão das redes de computadores tem tomado um rumo um tanto quanto preocupante, quando se trata de complexidade e quantidade de serviços e dispositivos interligados, tornando-se cada dia mais difícil o seu gerenciamento.

Para auxiliar nesta manutenção exitem diversas soluções para uma melhor administração e eficiência da rede, é assim que surgiu o serviço de controlador de domínio, que de modo simples poderíamos fazer uma analogia com um grupo de trabalho, que nada mais é que a unificação por departamentos, centralizando configurações de segurança e necessidades específicas a cada tipo de grupo, como informações sobre todos os recursos da rede e dos usuários, de modo centralizado, disponibilizando-os para usuários e aplicações.

Uma das maiores vantagens na utilização do serviço de controlador de domínio, com a utilização do protocolo LDAP, é a unificação do login dos usuários de diversos serviços da rede, como, por exemplo, servidor de banco de dados, servidor de e-mail, servidor de arquivos, dentre outros. Para termos essa facilidade no Windows contamos com o *Active Directory*, serviço oferecido pelas versões da família *server* do Windows, a partir do Windows 2000.

O controlador de domínio também pode ser implementado com a utilização do sistema operacional GNU/Linux; o software SAMBA é capaz disto. O SAMBA surgiu com a principal premissa de fazer com que um computador com sistema Windows consiga através da rede manipular arquivos em um sistema Linux, o sistema foi evoluindo e aumentando as suas funcionalidades, dentre essas a de controlador de domínio. O *Active Directory* é uma implementação LDAP para Windows, no GNU/Linux a implementação LDAP mais conhecida é a OpenLDAP. O SAMBA com o OpenLDAP permitem a implementação de um controlador de domínio com LDAP em um sistema GNU/Linux; naturalmente, este sistema pode ser utilizado no lugar de um servidor Windows, fornecendo serviço a clientes Windows — uma vantagem imediata desta mudança é a redução de custos financeiros com licenças de softwares.

Implementar um controlador de domínio com LDAP no GNU/Linux exige grande experiência do usuário, principalmente para a configuração do controlador de domínio e para

a interligação entre o SAMBA e o OpenLDAP, que é feita, principalmente, pela edição e criação de arquivos. Outro complicador é a necessidade do SAMBA em utilizar LDAP com *bind* não anônimo, ou seja, é necessário ter e fornecer credenciais de acesso do LDAP ao SAMBA; a maioria das aplicações que fazem uso de autenticação via LDAP utilizam *bind* anônimo. Referências para implementar controlador de domínio com LDAP estão disponíveis na internet, como existem muitas formas de se fazer, devido a diversidade de distribuições GNU/Linux, assim como a versatilidade do SAMBA e LDAP, as referências podem ser confusas e incompletas.

Além disso, a utilização da autenticação LDAP no SAMBA pode apresentar os seguintes problemas:

- Necessidade da redigitação da senha do usuário. O Windows usa uma criptografia de senhas diferentes do LDAP e sistemas GNU/Linux.
- Utilização de *bind* não anônimo.
- Adição de campos extras na base LDAP específicos para o SAMBA.
- O servidor LDAP deve estar ativo para o login no sistema Windows. É possível usar *cache* de senha, mas mesmo assim é necessário o acesso, como, por exemplo, para verificar o nome da estação.

Tentando resolver estes problemas, neste trabalho é proposto um esquema para o uso do SAMBA como o LDAP de forma indireta.

1.1 OBJETIVOS

1.1.1 OBJETIVO GERAL

Utilizar o serviço de controlador de domínio com autenticação LDAP de forma indireta. Será utilizado o SAMBA com o OpenLDAP, mas na configuração do SAMBA não haverá nenhuma configuração relacionada ao LDAP.

1.1.2 OBJETIVOS ESPECÍFICOS

- Instalar o SAMBA, e configurar para o modo controlador de domínio com autenticação simples (sem LDAP).
- Utilizar o proxy HTTP Squid para realizar autenticação LDAP.

- Elaborar um esquema para que o SAMBA possa utilizar dados de uma base LDAP na autenticação.

2 REFERENCIAL TEÓRICO

Uma rede local (LAN) pode ser definida como um conjunto de computadores de uma instituição com diversos prédios ou ambientes interconectados, localizados geograficamente numa mesma área. Atualmente, pode-se definir que uma LAN pertence ao mesmo domínio de *broadcast* e geralmente é administrada por uma única organização (FILIPETTI, 2009).

2.1 GRUPO DE TRABALHO E DOMÍNIO

Em uma rede local Windows podemos ter algumas configurações para que as estações de trabalho possam ser melhor gerenciadas, elas podem fazer parte de um grupo de trabalho ou de um domínio, a diferença em geral é a quantidade de hosts na rede, um grupo geralmente contém em torno de 20 máquinas, também no grupo de trabalho as máquinas ficam todas no mesmo nível, nenhuma tem controle sobre a outra, as contas dos usuários devem estar cadastradas em cada computador em um banco de dados local chamado SAM. Já em um Domínio pode haver um número de centenas ou milhares de hosts, centralizando a administração com as contas dos usuários em um ou mais servidores.

2.2 CONTROLADOR PRIMÁRIO DE DOMÍNIO

Como explicado anteriormente na utilização do domínio temos uma espécie de demarcação ou subdivisão da rede, então para gerenciar o controle e a centralização das informações necessitamos ter um controlador de domínio, que nada mais é do que um servidor e um administrador do mesmo contendo, no caso do Windows, o Active Directory, implementando o LDAP, com a base de dados de usuários da rede e as respectivas configurações para cada departamento. Juntamente com os usuários, essa base de dados contém informações sobre equipamentos e recursos da rede, conseguindo assim um gerenciamento mais eficiente de toda a rede, tanto no quesito humano como de hardware (SANTANA, 2015).

2.3 ACTIVE DIRECTORY

O Active Directory surgiu da necessidade de se ter uma centralização na administração dos recursos da rede de computadores tendo tudo em um único local, ou seja, ao invés do usuário ter um login para acessar cada serviço como; sistema principal da empresa; e-mails; logar no computador, dentre acesso a vários outros serviços, com a utilização do AD, os usuários poderão ter apenas uma conta de login para acessar todos os recursos disponíveis na rede. O Active Directory implementa um banco de dados com suporte ao protocolo LDAP, onde conterá além dos recursos da rede, as contas de login dos usuários. (SANTANA, 2015).

Além da centralização, também dispomos de diversas opões de políticas de acesso restritas por grupos de usuários, chamadas de *GPO (Group Polices Object)* que são configurações específicas por usuários ou grupos, como mapeamento de drivers impressoras, restrição de horário de acesso, scripts na inicialização após o login, dentre tantas outras.

2.4 LDAP

Muito similar a um banco dos dados, o LDAP consiste em um protocolo de acesso ao serviço de diretório, diferencia-se basicamente de um banco de dados por ter um foco em pesquisa e leitura e não em gravação, sendo que o banco de dados trata a leitura quanto a gravação como operações equivalentes, isso é, entende que haverá um número similar tanto de gravações como de leituras, no diretório as atualizações não são tão constantes.

Um serviço de diretório é uma forma de ordenação de informações em formato de árvore, facilitando a pesquisa, um dos exemplos mais utilizados para descrever um diretório, é uma lista telefônica ou um dicionário, então a grande eficiência do serviço de diretório é a pesquisa com rapidez em grande volume de dados, outra característica é a possibilidade de replicação, melhorando ainda mais a eficiência na pesquisa e segurança, um bom exemplo de utilização desse serviço é o DNS, serviço de resolução de nomes amplamente utilizados na internet.

O LDAP teve origem a partir do serviço de diretório X.500, que é um serviço que funciona sobre as sete camadas do modelo OSI, já o LDAP roda sobre o modelo TCP/IP, ficando assim bem mais leve que o primeiro, outra implementação para melhoria foi a retirada de operações pouco utilizadas no antigo X.500.

No LDAP as informações estão organizadas na forma de entradas, sendo que uma entrada é formada por uma coleção de atributos, esses atributos por sua vez são

caracterizados por um *Distinguished Name* (DN uid=am,ou=Pessoas,dc=ime,dc=usp,dc=br), ou Nome Distinto, único em toda a árvore.

Como exemplo de entrada, podemos equiparar a um caminho, utilizado para identificar o nome de um aluno por exemplo, em um cadastro escolar, ou o nome de um computador em um cadastro de equipamentos de rede de computadores, ou ainda uma cidade em um país, também temos o RDN que seria similar a apenas um campo do banco de dados, ex. uid=am (TUTTLE et al., 2004).

Cada DN contém um tipo para a entrada de dados, por exemplo, cn: commom name (nome comum) e mail: endereço de e-mail; esses campos são normalmente do tipo string (texto), mas também pode-se ter campos com formato de imagem (binário).

A estrutura da árvore, no LDAP é chamada de DIT, similar a Figura 1.

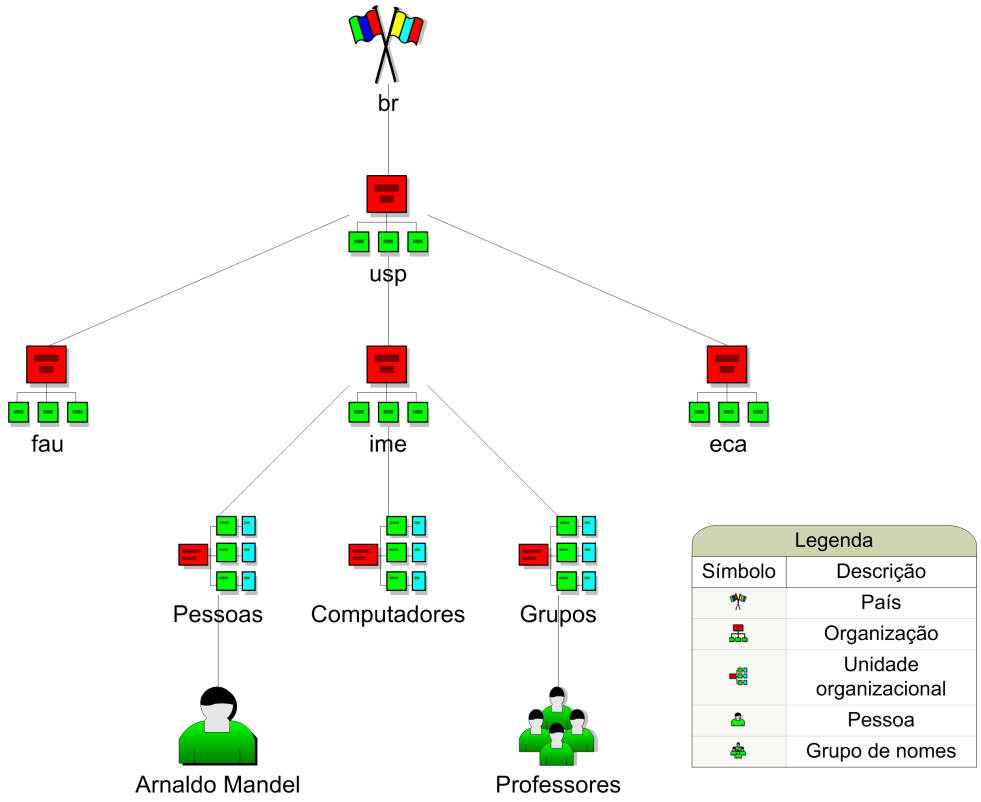

Figura 1: Exemplo de DIT.

Fonte: (IME, 2015)

2.5 SQUID

O Squid surgiu por volta de 1990, desenvolvido para sistemas tipo Unix, mas a partir da versão 2.6 também funciona para Windows. Sua principal utilização é a otimização da velocidade de navegação na internet, é um servidor proxy, funcionando da seguinte forma. Em uma empresa temos geralmente uma rede local com uma divisão de conexão com a internet externa, com essa rede interna, é exatamente ai nesse meio campo por assim dizer que trabalha o servidor proxy, ele armazena em um arquivo de cache todo o acesso externo que é feito na internet, então quando alguém tenta acessar um site, por exemplo, é verificado se este acesso já não foi feito anteriormente, se positivo, então o site é carregado a partir desse arquivo de cache sem a necessidade de acessar o site na internet. Com isso o acesso fica muitas vezes na própria rede interna que possui uma velocidade de acesso muito superior que a da internet.

Outra utilização é para aplicação de bloqueios de sites indevidos e também a armazenagem de arquivos de logs, contendo todo o acesso a internet da organização, como URL e acesso e IP, também muito utilizado pelos provedores de internet como forma de otimização do link de internet.

O Squid pode ser utilizado especificando-se no navegador ou sistema operacional o endereço do servidor e o número de porta, ou também de forma transparente, onde o usuário não precisa alterar nada em seu equipamento local, ficando imperceptível sua utilização por parte dos usuários (ROSA, 2015).

2.6 SAMBA

As redes de computadores surgiram com a necessidade de compartilhamento de arquivos, impressoras e outros recursos (SANTANA, 2015).

Para possibilitar esses tipos de compartilhamentos necessitamos de um protocolo, que nada mais é do que uma linguagem para os seres humanos, tornando a comunicação possível entre dois equipamentos, com isso em 1984 a IBM, criou o NetBIOS desenvolvido para troca de mensagens entre micros PC em uma rede local, já em 1985 o protocolo foi melhorado dando origem ao NetBEUI, muito utilizado em redes locais antes da ascensão do TCP/IP. Um pouco mais tarde com o Windows 3.11 surge o SMB, amplamente utilizado nas redes locais Microsoft para o compartilhamento de arquivos e impressoras e navegabilidade em rede, utilizando como pano de fundo o NetBIOS, fazendo a troca de mensagens entres hosts, rodando sobre o TCP/IP. Como o NetBIOS utiliza-se de um recurso de broadcast, o que executa muito tráfego na rede

com pacotes UDP nas portas (137, 138, e 139), surge então o CIFS, passando a utilizar apenas uma porta TCP (445).

Mas o CIFS, era somente para sistemas Windows, então enfrentava-se problemas de interoperabilidade entre diferentes sistemas operacionais, em se tratando de sistemas de arquivos. É exatamente nesse quesito que surgiu o SAMBA, da necessidade de compartilhamento de arquivos entre Linux e Windows através da rede, o nome deriva do protocolo utilizado pelo Windows para compartilhamento de arquivos e impressoras, o SMB.

O SAMBA surge como uma implementação do SMB para outras plataformas como Linux, BSD, Solaris, OS X, dentre outros, também acompanhando a evolução e implementando o CIFS.

A primeira versão do SAMBA foi desenvolvida em 1991, pelo australiano chamado Andrew Trigell, que tinha a necessidade de interligar um servidor Solaris com um computador operando MS-DOS, pensando em fazer isto, ele desenvolveu um *sniffer* (programa que intercepta dados em uma rede), e então fez a análise dos dados trafegados, desenvolvendo assim um protocolo similar ao SMB para que conseguisse fazer a comunicação entre esses dois sistemas. (ROSA, 2015).

A equipe SAMBA é um grupo de cerca de 40 pessoas de todo o mundo que contribuem regularmente, e quem têm acesso de gravação direta para o repositório SAMBA. Mas o número de pessoas diretamente envolvidas é de aproximadamente 15 pessoas.

2.6.1 PRINCIPAIS FUNCIONALIDADES DO SAMBA

Utilizamos a versão 4.0 do SAMBA, dentre suas principais funcionalidades estão, compartilhamento de arquivos entre máquinas Windows e Linux ou de máquinas Linux (sendo o servidor SAMBA) com outro SO que tenha um cliente NetBEUI (Macintosh, OS/2, LanManager, etc).

- Montar um servidor de compartilhamento de impressão no Linux que receberá a impressão de outras máquinas Windows da rede.
- Controle de acesso aos recursos compartilhados no servidor através de diversos métodos (compartilhamento, usuário, domínio, servidor), inclusive contas convidadas que possam se conectar sem senha.
- Possibilidade de uso do banco de dados de senha do sistema (/etc/passwd), autenticação usando o arquivo de dados criptografados do SAMBA, LDAP, PAM, etc.

- Permite ocultar o conteúdo de determinados diretórios que não quer que sejam exibidos ao usuário de forma fácil.
- Possui configuração bastante flexível com relação ao mapeamento de nomes DOS => UNIX e vice versa.
- Permite o uso de *aliases* na rede para identificar uma máquina com outro nome e simular uma rede NetBIOS virtual.
- Possui suporte completo a servidor WINS de rede.
- Faz auditoria tanto dos acessos a pesquisa de nomes na rede como acesso a compartilhamentos. Entre os detalhes salvos estão a data de acesso, IP de origem, etc.
- Suporte completo a controlador de domínio Windows PDC.
- Permite montar unidades mapeadas de sistemas Windows ou outros servidores Linux como um diretório no Linux (FOCALINUX, 2015).

2.6.2 FORMAS DE AUTENTICAÇÃO

2.6.2.1 NTLM

Em uma rede Windows, NTLM é um conjunto de protocolos de segurança da Microsoft que fornece autenticação, integridade e confidencialidade aos usuários. Este método de autenticação é utilizado pelo Windows NT e 2000 e atualmente não é utilizado com o *Active Directory*, sendo usado apenas quando um controlador de domínio não está disponível ou inacessível, ou ainda se o cliente não é compatível com o *kerberos*. A versão dois do NTLM, que foi introduzida pelo Windows NT 4.0 SP4 (e nativamente suportada no Windows 2000), aumentou a segurança NTLM através do amadurecimento do protocolo contra muitos ataques de spoofing e adicionando a capacidade de um servidor autenticar o cliente.

2.6.2.2 KERBEROS

Utilizado para autenticação cliente servidor, o Kerberos é um protocolo seguro que se utiliza de criptografia. No início da autenticação é verificado se ambas as pontas são verdadeiras, isto é, o cliente verifica se o servidor é verdadeiro, concluindo o processo de autenticação inicia-se uma sessão segura (ROSA, 2015).

Características positivas da utilização do kerberos:

- Autenticação mútua. O cliente pode validar a identidade do servidor e o servidor pode validar o cliente. Com essa documentação, as duas entidades são denominadas "cliente"e "servidor"mesmo que seja possível estabelecer conexões seguras de rede entre servidores.
- Permissões de autenticação segura. São utilizadas somente permissões criptografadas, e as senhas nunca são incluídas na permissão.
- Autenticação integrada. Uma vez que o usuário faça login, ele não precisa efetuar login novamente para acessar algum serviço que suporte a autenticação Kerberos, pois a permissão do cliente não expira. Todas as permissões têm um prazo de validade determinado pelas diretivas do Kerberos que geram a permissão.

Pontos Fracos na utilização do Kerberos:

- Apenas compatibilidade parcial com PAM.
- Seu principal método de trabalho é presumir que a rede é insegura, e os usuários são confiáveis, ele tem como premissa evitar que senhas não criptografadas trafeguem na rede, mas se um usuário mal intencionado acessar o servidor que emite os tickets, o servidor estará comprometido.
- Os aplicativos que fazem uso do kerberos devem ser alterados, para a utilização de suas bibliotecas, aplicativos que não tenham código aberto não tem suporte ao kerberos por padrão.

2.6.3 PAM.D

O surgimento do PAM veio para unificar a autenticação em um servidor, quando se utiliza uma base de clientes remota. Criado pela Sun Microsystems, são módulos que permitem a unificação de login de diversos aplicativos, incluindo a utilização de base de clientes LDAP.

Como citado anteriormente a principal utilização é a autenticação no sistema Linux, com a utilização dessa base de usuários não se tem a necessidade de cadastrar usuários locais, em um servidor Linux por exemplo, basta configurar o PAM com uma base de dados LDAP, e todos os usuários dessa base tem acesso ao sistema citado, com possibilidades de configurar acessos diferentes por grupos de usuários, e também por diversos outros critérios, como aplicativos, arquivos, etc (SRIVISTAVA, 2015).

O PAM é composto por quatro módulos principais, *Auth*, *Account*, *passwd* e *session*.

- Auth: faz a verificação com o usuário e senha, também tem a possibilidade de utilização de várias formas de autenticação como, retina, impressão digital, voz etc.
- Account: efetua a verificação das autorizações que o usuário possui, o que ele pode acessar e quando.
- Passwd: utilizado para fazer a alteração de senha.
- Session: cria o ambiente do usuário, montando dispositivos, como *cdroom*, *usb* e arquivos.

A maioria dos aplicativos que necessitam de autenticação tem suporte ao PAM, e a sua configuração está no diretório /etc/pam.d, os aplicativos têm duas interfaces com essa biblioteca, uma de autenticação, responsável pela validação do usuário e outra de conversação, que tem como função passar informação para o usuário.

3 MATERIAIS E MÉTODO

Para utilizar o LDAP de forma indireta com o SAMBA, adotou-se o esquema apresentado na Figura 2.

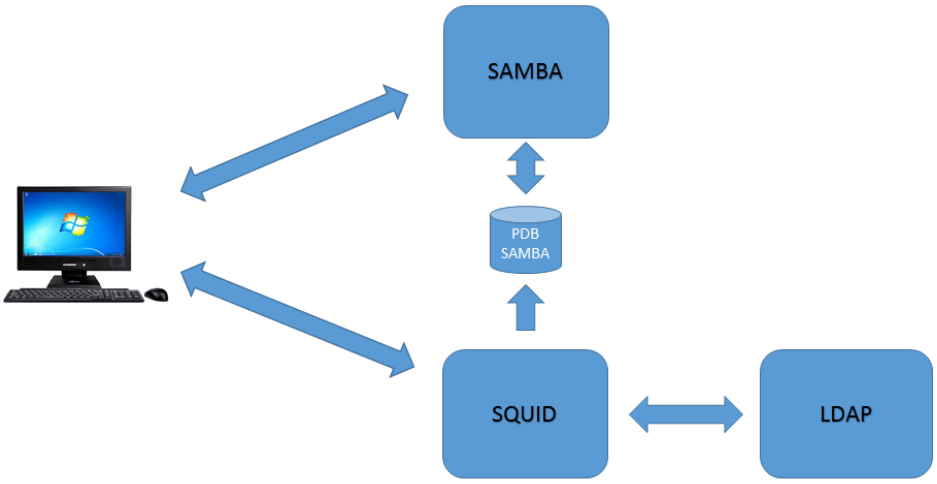

Figura 2: Esquema criado SAMBA com LDAP *offline*.

3.1 FUNCIONAMENTO DO ESQUEMA

O objetivo do esquema é permitir o uso da autenticação LDAP para o logon no Windows. Este esquema é diferente do modelo tradicional pois na configuração do SAMBA não foi utilizado configuração específica de LDAP.

O uso do SAMBA com LDAP, apresenta os seguintes problemas:

- Necessidade da redigitação da senha do usuário (o Windows usa uma criptografia de senha diferente do LDAP e sistemas GNU/Linux);
- Necessidade de *bind* não anônimo, ou seja, é necessário ter e fornecer credenciais de acesso do LDAP ao SAMBA – a maioria das aplicações que fazem uso de autenticação via LDAP utilizam *bind* anônimo;

- Adição de campos extras na base LDAP específicos para o SAMBA;
- O servidor LDAP deve estar ativo para o login no sistema Windows (é possível usar *cache* de senha, mas mesmo assim é necessário o acesso, como, por exemplo, para verificar o nome da estação).

O modelo proposto foi dividido em partes, para melhor descrevê-lo:

1. O SAMBA foi configurado para ser o controlador de domínio e fazer a autenticação de usuários de forma nativa, ou seja, utilizando uma base PDB.
2. A base PDB foi populada com informações de uma base LDAP, deixando os usuários sem senha.
3. Ao realizar o login no Windows um *script* é executado para ajustar o *web proxy* do usuário.
4. O Squid foi configurado para realizar autenticação de usuários, e o faz com o uso uma base LDAP.
5. Quando a autenticação do Squid é bem sucedida a senha do usuário da base PDB do SAMBA é atualizada. Desta forma o próximo login do usuário do Windows irá ser feito com sua senha do LDAP.

3.2 IMPLEMENTAÇÃO DAS PARTES

A seguir é fornecido alguns detalhes da implementação das partes, arquivos de configuração e códigos fontes são apresentados no Anexo A.

3.2.1 CONFIGURAÇÃO DO SAMBA

O SAMBA foi configurado para ser um controlador de domínio e utilizar autenticação em uma base PDB. Além de prover o serviço de domínio o SAMBA cria um compartilhamento chamado *netlogon*, de acesso público.

3.2.2 IMPORTAÇÃO LDAP PARA PDB

Como o SAMBA utiliza uma base PDB, quando um usuário tenta fazer logon no Windows esta base é consultada. Os usuários (nome e login) devem ser importados do LDAP

para a base PDB, isto é feito com o uso da ferramenta *LDAPsearch* e do script *ldap-pdb-register.py*.

O *LDAPsearch* é utilizado para realizar uma consulta na base LDAP e retornar os resultados no formato texto; estes resultados devem direcionados para um arquivo. O script *ldap-pdb-register.py* é responsável por fazer a leitura deste arquivo e adicionar usuários na base PDB. Isto é feito da seguinte forma:

- É localizado os campos login (uid) e nome do usuário (cn).
- É verificado se o login existe na base PDB. Se não existir:
  - O usuário é adicionado ao sistema com o comando *useradd*.
  - O usuário também é adicionado a base PDB com o comando *pdbedit*.
- Por final os usuários adicionados, ou seja, os usuários da base LDAP são sincronizados com os usuários locais.
  - Se um usuário foi deletado da base LDAP ele será deletado da base local.

3.2.3 SCRIPTS DE LOGON

O compartilhamento *netlogon* é utilizado para que após o login no controlador de domínio seja executado um script (BAT). Este script irá mapear uma unidade para o compartilhamento *netlogon* e na sequência irá executar um comando para adicionar a configuração de proxy no registro do Windows, isto é feito com o comando *regedit.exe* /S *proxy.reg*.

3.2.4 CONFIGURAÇÃO DO SQUID

A configuração do Squid foi feita para exigir autenticação dos usuários. O processo de autenticação foi feito pelo módulo *basic_pdbldap_auth* que é detalhado a seguir.

3.2.5 ATUALIZAÇÃO PDB PELO SQUID

O Squid utiliza como módulo de autenticação o *basic_pdbldap_auth*, este programa foi feito com base no programa de autenticação LDAP do Squid. O programa faz a autenticação LDAP das credenciais fornecidas, e caso elas sejam válidas é executada a alteração da senha do referido usuário na base PDB, isto é feito com o uso do comando *smbpasswd*.

4 RESULTADOS E DISCUSSÃO

O método proposto pode ser utilizado para substituir o esquema tradicional de uso do SAMBA + LDAP em redes onde o logon do usuário em uma estação não é considerado um fator de risco, pois quando os usuários são adicionados pelo script (*ldap-pdb-register.py*) a senha deles é nula, ou seja, em branco, logo, qualquer pessoa de posse do nome de login consegue ter acesso à estação; adicionou-se os usuários sem senha pois a criptografia utilizada pelo LDAP não é compatível com a do SAMBA. Devido a isso no primeiro acesso autenticado ao *Proxy Squid*, a senha foi automaticamente cadastrada na base PDB do SAMBA, com a devida criptografia. Para a implementação do novo método é necessário que os softwares SAMBA e Squid estejam no mesmo servidor, porém caso isto não ocorra, pode-se utilizar um sistema de arquivos via rede.

Em relação ao método tradicional SAMBA com LDAP, o novo esquema atuou como uma ponte entre estes serviços, obtendo dados de login do LDAP para criar/atualizar a base PDB, e atualizando a senha da base PDB após um login bem sucedido no Squid. Importante, utilizou-se o Squid para realizar a atualização de senha, mas qualquer outro serviço disponível ao usuário, que utilize LDAP, poderia ser utilizado para efetuar a atualização.

O uso do esquema também pode ajudar a melhorar o congestionamento da rede em horários de pico, pois requisições de login do Windows não serão encaminhadas ao LDAP, e sim ao PDB.

Conhecendo o login de algum usuário que nunca utilizou o sistema, um usuário poderia fazer uso deste login no Windows e utilizar o seu próprio login para autenticar no Squid, ou seja, ele usa um usuário para logar no Windows e outro para o Squid. Uma forma de se evitar isto é com a criação de um programa para ser executado em plano de fundo (*daemon*) na estação Windows, que fica aguardando uma mensagem via rede para verificar se o login contido na mensagem é o mesmo que está em uso no Windows, caso não o seja, uma ação de logoff poderia ser executada. Essa mensagem seria enviada por um outro programa acionado pelo Squid no momento da autenticação, isto seria possível com o uso do Squid na versão 3.5 ou

superior, graças ao recurso *key_extras*, que permite enviar parâmetros adicionais (como IP de origem) ao programa de autenticação. Este mesmo *daemon* poderia ser utilizado para notificar ao usuário mudanças na senha da base PDB, para que no próximo acesso ao Windows o usuário utilize a senha adequada.

5 CONCLUSÃO

Uma das maiores vantagens da utilização do serviço de controlador de domínio, com a utilização do protocolo LDAP, é a unificação do login dos usuários de diversos serviços da rede. O controlador de domínio pode ser implementado com a utilização de sistemas GNU/Linux; isto normalmente é feito com o uso do software SAMBA. Implementar um controlador de domínio com LDAP no GNU/Linux exige grande experiência do usuário, principalmente para a configuração do SAMBA e do LDAP, além de apresentar alguns problemas, como: necessidade de redigitação de senha e utilização de bind não anônimo. Este trabalho apresentou uma nova forma para se utilizar o SAMBA com o LDAP, onde o LDAP é usado de forma indireta.

O novo esquema fez uso do SAMBA, LDAP e Squid, além de novos utilitários desenvolvidos. Com a aplicação do novo método foi possível realizar a autenticação do login do Windows com o uso do LDAP de forma indireta ou *offline*. O esquema permitiu o uso do SAMBA com o LDAP sem a necessidade de se fornecer credenciais de acesso ao SAMBA, e o usuário não precisou redigitar a senha para ter acesso ao logon do Windows. Se o login na estação for considerado um fator de risco, o método não é recomendado, pois a senha só é importada após a autenticação no Squid. Como trabalho futuro pode se pensar em uma integração mais genérica entre serviços de troca de senha, para se evitar problemas de login.

REFERÊNCIAS

FILIPETTI, M. A. **CCNA4.1 Guia de Estudo Completo**. Florianópolis: Visual Books, 2009.

FOCALINUX. **A origem do Samba**. 2015. Disponível em: <http://www.guiafoca.org/cgs/guia/avancado/ch-s-samba.html>. Acesso em: 9 de junho de 2015.

IME. **Exemplo de dit**. 2015. Disponível em: <http:/linux.ime.usp.br>. Acesso em: 11 de junho de 2015.

ROSA, R. da S. **Instalação do Squid com autenticação NTLM e Kerberos**. 2015. Disponível em: <http://www.hardware.com.br/dicas/samba-windows-vista.html>. Acesso em: 26 de julho de 2015.

SANTANA, F. de. **WINDOWS 2000 AD – Active Directory**. 2015. Disponível em: <http://juliobattisti.com.br/fabiano/artigos/activedirectory.asp>. Acesso em: 11 de junho de 2015.

SRIVISTAVA, V. **Entendendo e Configurando o PAM**. 2015. Disponível em: <http://www.ibm.com/developerworks/br/library/l-pam/>. Acesso em: 29 de junho de 2015.

TUTTLE, S. et al. **Understanding LDAP Design and Implementation**. Estados Unidos da América: IBM, 2004.

ANEXO A – CONFIGURAÇÕES E CÓDIGOS FONTES

A.1 CONFIGURAÇÃO DO SAMBA

```
[global]
    workgroup = GRUPODETRABALHO
    netbios name = nomedoservidor
####### Autenticacao #######
  passdb backend = tdbsam
  security = user
  bind interfaces only = yes
  encrypt passwords = yes
########## Dominio ###########
  domain logons = Yes
  preferred master = yes
  domain master = yes
  username map = /etc/samba/smbusers
  wins support = yes
  passwd program = /usr/bin/passwd %u
  add machine script = /usr/sbin/useradd -s /bin/false/ -d /var/lib/nobody %u
  logon script = netlogon.bat
#======================= Compartilhamentos ======================
[netlogon]
    comment = Network Logon Service
    path = /home/samba/netlogon
    guest ok = yes
    read only = yes
    share modes = no
```

A.2 IMPORTAÇÃO LDAP PARA PDB

```
#!/usr/bin/python
#
# Copyright (C) Amilcar Michelin 2015, Marcos Talau 2015
# Author Amilcar Michelin (amilcarm@utfpr.edu.br)
# Author Marcos Talau (talau@users.sourceforge.net)
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
```

```
9  # the Free Software Foundation, either version 3 of the License, or
10 # (at your option) any later version.
11 #
12 # This program is distributed in the hope that it will be useful,
13 # but WITHOUT ANY WARRANTY; without even the implied warranty of
14 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
15 # GNU General Public License for more details.
16 #
17 # You should have received a copy of the GNU General Public License
18 # along with this program. If not, see <http://www.gnu.org/licenses/>.
19 #
20
21
22 import sys
23 import commands
24 import os
25 import time
26
27 DEFAULT_GROUP = "smb"
28 LOG_ADD = "/var/log/ldap-pdb-register.log"
29 ldap_users = []
30
31 def log(msg):
32     t = time.strftime("%Y/%m/%d %H:%M")
33     f = open(LOG_ADD, "a")
34     f.write("%s: %s\n" % (t, msg))
35     f.close()
36
37 def login_exists(login):
38     out = commands.getoutput("pdbedit -L")
39
40     for l in out.splitlines():
41         if login == l.split(":")[0]:
42             return 1
43
44     return 0
45
46 def add_user(username, userlogin):
47     print("Adding user " + userlogin)
48     log("Added %s; %s" % (username, userlogin))
49
50     # first add a system user
51     cmd = "useradd -M -N -g %s -s /usr/sbin/nologin -c \"%s\" %s" %
           (DEFAULT_GROUP, username+",,,", userlogin)
52     os.system(cmd)
53
54     # now samba
55     os.system("printf \"\\n\\n\" | pdbedit -a -u %s -f \"%s\" -t > /dev/null" % (userlogin,
           username))
56
57 def remove_user(userlogin):
```

```
58      print("Removing user " + userlogin)
59      log("Removed %s; %s" % userlogin)
60
61      os.system("userdel −f −r %s" % userlogin)
62      os.system("pdbedit −x −u %s" % userlogin)
63
64  def get_local_users():
65      f = open("/etc/group")
66      lines = f.readlines()
67      f.close()
68      for l in lines:
69          if l.find("%s:" % DEFAULT_GROUP) != −1:
70              return l.split(":")[3].split(",")
71
72  def is_remote_user(user):
73      for u in ldap_users:
74          if u == user:
75              return True
76      return False
77
78  def sync_ldap_pdb():
79      local_users = get_local_users()
80
81      for lu in local_users:
82          if not is_remote_user(lu):
83              remove_user(lu)
84
85  def start():
86      if commands.getoutput("whoami") != "root":
87          print "[ERROR] You must be a root user"
88          sys.exit(−1)
89
90      if os.system("which pdbedit > /dev/null"):
91          print "[ERROR] command pdbedit not found. samba package installed?"
92          sys.exit(−1)
93
94      r = os.system("groupmod %s" % DEFAULT_GROUP)
95
96      if r:
97          os.system("groupadd %s" % DEFAULT_GROUP)
98
99  def check_add(userlogin, username):
100     ldap_users.append(userlogin)
101     if not login_exists(userlogin):
102         add_user(username, userlogin)
103
104 # main
105 ldapdata = sys.argv[1]
106
107 start()
108
```

```
f = open(ldapdata)
lines = f.readlines()
f.close()
newentry = False
uid = None
cn = None
for l in lines:
    if l[0] == "#":
        continue
    if l.startswith("dn:") and l.find("uid=") != -1:
        if l.find("uid=root") != -1 or l.find("$") != -1:
            continue
        if uid and cn:
            check_add(uid, cn)
        newentry = True
        uid = l.split("uid=")[1].split(",")[0]
        cn = None
    if newentry and l.startswith("cn: "):
        cn = l[4:len(l)-1]
        newentry = False
# get the last too
if uid and cn:
    check_add(uid, cn)

sync_ldap_pdb()
```

A.3 SCRIPTS DE LOGON

netlogon.bat

```
@echo off
net use l: \\IP_SERVIDOR_SAMBA\netlogon
regedit.exe /S l:\proxy.reg
```

proxy.reg

```
Windows Registry Editor Version 5.00

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Internet Settings]
"MigrateProxy"=dword:00000001
"ProxyEnable"=dword:00000001
"ProxyHttp1.1"=dword:00000000
"ProxyServer"="IP_SQUID:3128"
"ProxyOverride"="<local>"
```

A.4 CONFIGURAÇÃO DO SQUID

```
http_port *:3128

# Autenticacao LDAP
auth_param basic realm Digite seu login e senha para o acesso a Internet.
auth_param basic program /usr/lib/squid3/basic_pdbldap_auth -b
    "dc=exemplo,dc=com,dc=br" -f "uid=%s" -H IP_SERVIDOR
auth_param basic children 5
auth_param basic credentialsttl 4 hour
acl autenticados proxy_auth REQUIRED

http_access allow autenticados
http_access deny all
```

A.5 ATUALIZAÇÃO PDB PELO SQUID

```
/*
 * squid_pdbldap_auth: authentication via ldap, build SMB PW HASH and stores.
 *
 * Authors:
 * Amilcar Michelin
 * Marcos Talau
 *
 * This file is based in basic_ldap_auth.cc
 *
 * Usage: squid_pdbldap_auth -b basedn [-s searchscope]
 * [-f searchfilter] [-D binddn -w bindpasswd]
 * [-u attr] [-h host] [-p port] [-P] [-R] [ldap_server_name[:port]] ...
 *
 * LDAP codes from: basic_ldap_auth.cc
 *
 * Dependencies: You need to get the OpenLDAP libraries
 * from http://www.openldap.org or another compatible LDAP C-API
 * implementation.
 *
 * If you want to make a TLS enabled connection you will also need the
 * OpenSSL libraries linked into openldap. See http://www.openssl.org/
 *
 * License: squid_pdbldap_auth is free software; you can redistribute it
 * and/or modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2,
 * or (at your option) any later version.
 *
 */

#include "squid.h"
#include "helpers/defines.h"

#define LDAP_DEPRECATED 1
```

```
34
35 #include "rfc1738.h"
36 #include "util.h"
37
38 #include <stdio.h>
39 #include <stdlib.h>
40 #include <string.h>
41 #include <ctype.h>
42
43
44 #if _SQUID_WINDOWS_ && !_SQUID_CYGWIN_
45 #error This auth module is useful only in UNIX/Linux
46
47 #else
48
49 #include <lber.h>
50 #include <ldap.h>
51
52 #endif
53
54
55 #define PROGRAM_NAME "basic_pdbldap_auth"
56
57 /* Global options */
58 static const char *basedn;
59 static const char *searchfilter = NULL;
60 static const char *binddn = NULL;
61 static const char *bindpasswd = NULL;
62 static const char *userattr = "uid";
63 static const char *passwdattr = NULL;
64 static int searchscope = LDAP_SCOPE_SUBTREE;
65 static int persistent = 0;
66 static int bind_once = 0;
67 static int noreferrals = 0;
68 static int aliasderef = LDAP_DEREF_NEVER;
69 #if defined(NETSCAPE_SSL)
70 static const char *sslpath = NULL;
71 static int sslinit = 0;
72 #endif
73 static int connect_timeout = 0;
74 static int timelimit = LDAP_NO_LIMIT;
75
76 /* Added for TLS support and version 3 */
77 static int use_tls = 0;
78 static int version = -1;
79
80 static int checkLDAP(LDAP * ld, const char *userid, const char *password, const char
       *server, int port);
81 static int readSecret(const char *filename);
82
83 /* Yuck.. we need to glue to different versions of the API */
```

```
84
85  #ifndef LDAP_NO_ATTRS
86  #define LDAP_NO_ATTRS "1.1"
87  #endif
88
89  #if defined(LDAP_API_VERSION) && LDAP_API_VERSION > 1823
90  static int
91  squid_ldap_errno(LDAP * ld)
92  {
93      int err = 0;
94      ldap_get_option(ld, LDAP_OPT_ERROR_NUMBER, &err);
95      return err;
96  }
97  static void
98  squid_ldap_set_aliasderef(LDAP * ld, int deref)
99  {
100     ldap_set_option(ld, LDAP_OPT_DEREF, &deref);
101 }
102 static void
103 squid_ldap_set_referrals(LDAP * ld, int referrals)
104 {
105     int *value = static_cast<int*>(referrals ? LDAP_OPT_ON :LDAP_OPT_OFF);
106     ldap_set_option(ld, LDAP_OPT_REFERRALS, value);
107 }
108 static void
109 squid_ldap_set_timelimit(LDAP * ld, int aTimeLimit)
110 {
111     ldap_set_option(ld, LDAP_OPT_TIMELIMIT, &aTimeLimit);
112 }
113 static void
114 squid_ldap_set_connect_timeout(LDAP * ld, int aTimeLimit)
115 {
116 #if defined(LDAP_OPT_NETWORK_TIMEOUT)
117     struct timeval tv;
118     tv.tv_sec = aTimeLimit;
119     tv.tv_usec = 0;
120     ldap_set_option(ld, LDAP_OPT_NETWORK_TIMEOUT, &tv);
121 #elif defined(LDAP_X_OPT_CONNECT_TIMEOUT)
122     aTimeLimit *= 1000;
123     ldap_set_option(ld, LDAP_X_OPT_CONNECT_TIMEOUT, &aTimeLimit);
124 #endif
125 }
126 static void
127 squid_ldap_memfree(char *p)
128 {
129     ldap_memfree(p);
130 }
131
132 #else
133 static int
134 squid_ldap_errno(LDAP * ld)
```

```
135 {
136     return ld->ld_errno;
137 }
138 static void
139 squid_ldap_set_aliasderef(LDAP * ld, int deref)
140 {
141     ld->ld_deref = deref;
142 }
143 static void
144 squid_ldap_set_referrals(LDAP * ld, int referrals)
145 {
146     if (referrals)
147         ld->ld_options |= ~LDAP_OPT_REFERRALS;
148     else
149         ld->ld_options &= ~LDAP_OPT_REFERRALS;
150 }
151 static void
152 squid_ldap_set_timelimit(LDAP * ld, int timelimit)
153 {
154     ld->ld_timelimit = timelimit;
155 }
156 static void
157 squid_ldap_set_connect_timeout(LDAP * ld, int timelimit)
158 {
159     fprintf(stderr, "Connect timeouts not supported in your LDAP library\n");
160 }
161 static void
162 squid_ldap_memfree(char *p)
163 {
164     free(p);
165 }
166
167 #endif
168
169 #ifdef LDAP_API_FEATURE_X_OPENLDAP
170 #if LDAP_VENDOR_VERSION > 194
171 #define HAS_URI_SUPPORT 1
172 #endif
173 #endif
174
175 static LDAP *
176 open_ldap_connection(const char *ldapServer, int port)
177 {
178     LDAP *ld = NULL;
179 #if HAS_URI_SUPPORT
180     if (strstr(ldapServer, "://") != NULL) {
181         int rc = ldap_initialize(&ld, ldapServer);
182         if (rc != LDAP_SUCCESS) {
183             fprintf(stderr, "\nUnable to connect to LDAPURI:%s\n", ldapServer);
184             exit(1);
185         }
```

```
186     } else
187 #endif
188 #if NETSCAPE_SSL
189         if (sslpath) {
190             if (!sslinit && (ldapssl_client_init(sslpath, NULL) != LDAP_SUCCESS)) {
191                 fprintf(stderr, "\nUnable to initialise SSL with cert path %s\n",
192                         sslpath);
193                 exit(1);
194             } else {
195                 ++sslinit;
196             }
197             if ((ld = ldapssl_init(ldapServer, port, 1)) == NULL) {
198                 fprintf(stderr, "\nUnable to connect to SSL LDAP server: %s port:%d\n",
199                         ldapServer, port);
200                 exit(1);
201             }
202         } else
203 #endif
204             if ((ld = ldap_init(ldapServer, port)) == NULL) {
205                 fprintf(stderr, "\nUnable to connect to LDAP server:%s port:%d\n",
206                         ldapServer, port);
207                 exit(1);
208             }
209     if (connect_timeout)
210         squid_ldap_set_connect_timeout(ld, connect_timeout);
211
212 #ifdef LDAP_VERSION3
213     if (version == -1) {
214         version = LDAP_VERSION3;
215     }
216     if (ldap_set_option(ld, LDAP_OPT_PROTOCOL_VERSION, &version) !=
        LDAP_SUCCESS) {
217         fprintf(stderr, "Could not set LDAP_OPT_PROTOCOL_VERSION %d\n",
218                 version);
219         exit(1);
220     }
221     if (use_tls) {
222 #ifdef LDAP_OPT_X_TLS
223         if (version != LDAP_VERSION3) {
224             fprintf(stderr, "TLS requires LDAP version 3\n");
225             exit(1);
226         } else if (ldap_start_tls_s(ld, NULL, NULL) != LDAP_SUCCESS) {
227             fprintf(stderr, "Could not Activate TLS connection\n");
228             exit(1);
229         }
230 #else
231         fprintf(stderr, "TLS not supported with your LDAP library\n");
232         exit(1);
233 #endif
234     }
235 #endif
```

```
236     squid_ldap_set_timelimit(ld, timelimit);
237     squid_ldap_set_referrals(ld, !noreferrals);
238     squid_ldap_set_aliasderef(ld, aliasderef);
239     return ld;
240 }
241
242 /* Make a sanity check on the username to reject oddly typed names */
243 static int
244 validUsername(const char *user)
245 {
246     const unsigned char *p = (const unsigned char *) user;
247
248     /* Leading whitespace? */
249     if (xisspace(p[0]))
250         return 0;
251     while (p[0] && p[1]) {
252         if (xisspace(p[0])) {
253             /* More than one consequitive space? */
254             if (xisspace(p[1]))
255                 return 0;
256             /* or odd space type character used? */
257             if (p[0] != ' ')
258                 return 0;
259         }
260         ++p;
261     }
262     /* Trailing whitespace? */
263     if (xisspace(p[0]))
264         return 0;
265     return 1;
266 }
267
268 int
269 main(int argc, char **argv)
270 {
271     char buf[1024];
272     char *user, *passwd;
273     char *ldapServer = NULL;
274     LDAP *ld = NULL;
275     int tryagain;
276     int port = LDAP_PORT;
277     char cmd[300];
278
279     setbuf(stdout, NULL);
280
281     while (argc > 1 && argv[1][0] == '-') {
282         const char *value = "";
283         char option = argv[1][1];
284         switch (option) {
285         case 'P':
286         case 'R':
```

```
287         case 'z':
288         case 'Z':
289         case 'd':
290         case 'O':
291             break;
292         default:
293             if (strlen(argv[1]) > 2) {
294                 value = argv[1] + 2;
295             } else if (argc > 2) {
296                 value = argv[2];
297                 ++argv;
298                 --argc;
299             } else
300                 value = "";
301             break;
302         }
303         ++argv;
304         --argc;
305         switch (option) {
306         case 'H':
307 #if !HAS_URI_SUPPORT
308             fprintf(stderr, "ERROR: Your LDAP library does not have URI support\n");
309             exit(1);
310 #endif
311             /* Fall thru to -h */
312         case 'h':
313             if (ldapServer) {
314                 int len = strlen(ldapServer) + 1 + strlen(value) + 1;
315                 char *newhost = static_cast<char*>(malloc(len));
316                 snprintf(newhost, len, "%s %s", ldapServer, value);
317                 free(ldapServer);
318                 ldapServer = newhost;
319             } else {
320                 ldapServer = xstrdup(value);
321             }
322             break;
323         case 'b':
324             basedn = value;
325             break;
326         case 'f':
327             searchfilter = value;
328             break;
329         case 'u':
330             userattr = value;
331             break;
332         case 'U':
333             passwdattr = value;
334             break;
335         case 's':
336             if (strcmp(value, "base") == 0)
337                 searchscope = LDAP_SCOPE_BASE;
```

```
338                 else if (strcmp(value, "one") == 0)
339                     searchscope = LDAP_SCOPE_ONELEVEL;
340                 else if (strcmp(value, "sub") == 0)
341                     searchscope = LDAP_SCOPE_SUBTREE;
342                 else {
343                     fprintf(stderr, PROGRAM_NAME ": ERROR: Unknown search scope '%s'\n",
                            value);
344                     exit(1);
345                 }
346                 break;
347             case 'E':
348 #if defined(NETSCAPE_SSL)
349                 sslpath = value;
350                 if (port == LDAP_PORT)
351                     port = LDAPS_PORT;
352 #else
353                 fprintf(stderr, PROGRAM_NAME " ERROR: -E unsupported with this LDAP
                        library\n");
354                 exit(1);
355 #endif
356                 break;
357             case 'c':
358                 connect_timeout = atoi(value);
359                 break;
360             case 't':
361                 timelimit = atoi(value);
362                 break;
363             case 'a':
364                 if (strcmp(value, "never") == 0)
365                     aliasderef = LDAP_DEREF_NEVER;
366                 else if (strcmp(value, "always") == 0)
367                     aliasderef = LDAP_DEREF_ALWAYS;
368                 else if (strcmp(value, "search") == 0)
369                     aliasderef = LDAP_DEREF_SEARCHING;
370                 else if (strcmp(value, "find") == 0)
371                     aliasderef = LDAP_DEREF_FINDING;
372                 else {
373                     fprintf(stderr, PROGRAM_NAME ": ERROR: Unknown alias dereference
                            method '%s'\n", value);
374                     exit(1);
375                 }
376                 break;
377             case 'D':
378                 binddn = value;
379                 break;
380             case 'w':
381                 bindpasswd = value;
382                 break;
383             case 'W':
384                 readSecret(value);
385                 break;
```

```
386             case 'P':
387                 persistent = !persistent;
388                 break;
389             case 'O':
390                 bind_once = !bind_once;
391                 break;
392             case 'p':
393                 port = atoi(value);
394                 break;
395             case 'R':
396                 noreferrals = !noreferrals;
397                 break;
398     #ifdef LDAP_VERSION3
399             case 'v':
400                 switch (atoi(value)) {
401                 case 2:
402                     version = LDAP_VERSION2;
403                     break;
404                 case 3:
405                     version = LDAP_VERSION3;
406                     break;
407                 default:
408                     fprintf(stderr, "Protocol version should be 2 or 3\n");
409                     exit(1);
410                 }
411                 break;
412             case 'Z':
413                 if (version == LDAP_VERSION2) {
414                     fprintf(stderr, "TLS (-Z) is incompatible with version %d\n",
415                             version);
416                     exit(1);
417                 }
418                 version = LDAP_VERSION3;
419                 use_tls = 1;
420                 break;
421     #endif
422             case 'd':
423                 debug_enabled = 1;
424                 break;
425             default:
426                 fprintf(stderr, PROGRAM_NAME ": ERROR: Unknown command line option
                        '%c'\n", option);
427                 exit(1);
428             }
429         }
430
431         while (argc > 1) {
432             char *value = argv[1];
433             if (ldapServer) {
434                 int len = strlen(ldapServer) + 1 + strlen(value) + 1;
435                 char *newhost = static_cast<char*>(malloc(len));
```

```
436                 snprintf(newhost, len, "%s %s", ldapServer, value);
437                 free(ldapServer);
438                 ldapServer = newhost;
439             } else {
440                 ldapServer = xstrdup(value);
441             }
442             --argc;
443             ++argv;
444         }
445         if (!ldapServer)
446             ldapServer = xstrdup("localhost");
447 
448         if (!basedn) {
449             fprintf(stderr, "Usage: " PROGRAM_NAME " -b basedn [options]
                    [ldap_server_name[:port]]...\n\n");
450             fprintf(stderr, "\t-b basedn (REQUIRED)\tbase dn under which to search\n");
451             fprintf(stderr, "\t-f filter\t\tsearch filter to locate user DN\n");
452             fprintf(stderr, "\t-u userattr\t\tusername DN attribute\n");
453             fprintf(stderr, "\t-s base|one|sub\t\tsearch scope\n");
454             fprintf(stderr, "\t-D binddn\t\tDN to bind as to perform searches\n");
455             fprintf(stderr, "\t-w bindpasswd\t\tpassword for binddn\n");
456             fprintf(stderr, "\t-W secretfile\t\tread password for binddn from file secretfile\n");
457 #if HAS_URI_SUPPORT
458             fprintf(stderr, "\t-H URI\t\t\tLDAPURI (defaults to ldap://localhost)\n");
459 #endif
460             fprintf(stderr, "\t-h server\t\tLDAP server (defaults to localhost)\n");
461             fprintf(stderr, "\t-p port\t\t\tLDAP server port\n");
462             fprintf(stderr, "\t-P\t\t\tpersistent LDAP connection\n");
463 #if defined(NETSCAPE_SSL)
464             fprintf(stderr, "\t-E sslcertpath\t\tenable LDAP over SSL\n");
465 #endif
466             fprintf(stderr, "\t-c timeout\t\tconnect timeout\n");
467             fprintf(stderr, "\t-t timelimit\t\tsearch time limit\n");
468             fprintf(stderr, "\t-R\t\t\tdo not follow referrals\n");
469             fprintf(stderr, "\t-a never|always|search|find\n\t\t\t\twhen to dereference aliases\n");
470 #ifdef LDAP_VERSION3
471             fprintf(stderr, "\t-v 2|3\t\t\tLDAP version\n");
472             fprintf(stderr, "\t-Z\t\t\tTLS encrypt the LDAP connection, requires LDAP version
                    3\n");
473 #endif
474             fprintf(stderr, "\t-d\t\t\tenable debug mode\n");
475             fprintf(stderr, "\n");
476             fprintf(stderr, "\tIf no search filter is specified, then the dn
                    <userattr>=user,basedn\n\twill be used (same as specifying a search filter of
                    '<userattr>=',\n\tbut quicker as as there is no need to search for the user
                    DN)\n\n");
477             fprintf(stderr, "\tIf you need to bind as a user to perform searches then use the\n\t-D
                    binddn -w bindpasswd or -D binddn -W secretfile options\n\n");
478             exit(1);
479         }
480 
```

```
481     while (fgets(buf, sizeof(buf), stdin) != NULL) {
482         user = strtok(buf, " \r\n");
483         passwd = strtok(NULL, "\r\n");
484 
485         if (!user) {
486             printf("ERR Missing username\n");
487             continue;
488         }
489         if (!passwd || !passwd[0]) {
490             printf("ERR Missing password '%s'\n", user);
491             continue;
492         }
493         rfc1738_unescape(user);
494         rfc1738_unescape(passwd);
495         if (!validUsername(user)) {
496             printf("ERR No such user '%s':'%s'\n",user, passwd);
497             continue;
498         }
499         tryagain = (ld != NULL);
500 recover:
501         if (ld == NULL && persistent)
502             ld = open_ldap_connection(ldapServer, port);
503         if (checkLDAP(ld, user, passwd, ldapServer, port) != 0) {
504             if (tryagain && squid_ldap_errno(ld) != LDAP_INVALID_CREDENTIALS) {
505                 tryagain = 0;
506                 ldap_unbind(ld);
507                 ld = NULL;
508                 goto recover;
509             }
510             SEND_ERR(ldap_err2string(squid_ldap_errno(ld)));
511         } else {
512                 sprintf(cmd, "printf \"%s\n%s\n\" | smbpasswd -s -U %s", passwd, passwd,
                        user);
513                 system(cmd);
514 
515                 SEND_OK("");
516         }
517         if (ld && (squid_ldap_errno(ld) != LDAP_SUCCESS && squid_ldap_errno(ld) !=
                LDAP_INVALID_CREDENTIALS)) {
518             ldap_unbind(ld);
519             ld = NULL;
520         }
521     }
522     if (ld)
523         ldap_unbind(ld);
524     return 0;
525 }
526 
527 static int
528 ldap_escape_value(char *escaped, int size, const char *src)
529 {
```

```
530     int n = 0;
531     while (size > 4 && *src) {
532         switch (*src) {
533         case '*':
534         case '(':
535         case ')':
536         case '\\':
537             n += 3;
538             size -= 3;
539             if (size > 0) {
540                 *escaped = '\\';
541                 ++escaped;
542                 snprintf(escaped, 3, "%02x", (unsigned char) *src);
543                 ++src;
544                 escaped += 2;
545             }
546             break;
547         default:
548             *escaped = *src;
549             ++escaped;
550             ++src;
551             ++n;
552             --size;
553         }
554     }
555     *escaped = '\0';
556     return n;
557 }
558
559 /* Check the userid & password.
560  * Return 0 on success, 1 on failure
561  */
562 static int
563 checkLDAP(LDAP * persistent_ld, const char *userid, const char *password, const char
        *ldapServer, int port)
564 {
565     char dn[1024];
566     int ret = 0;
567     LDAP *bind_ld = NULL;
568
569     if (!*password) {
570         /* LDAP can't bind with a blank password. Seen as "anonymous"
571          * and always granted access
572          */
573         debug("Blank password given\n");
574         return 1;
575     }
576     if (searchfilter) {
577         char filter[16384];
578         char escaped_login[1024];
579         LDAPMessage *res = NULL;
```

```
580         LDAPMessage *entry;
581         char *searchattr[] = {(char *)LDAP_NO_ATTRS, NULL};
582         char *userdn;
583         int rc;
584         LDAP *search_ld = persistent_ld;
585
586         if (!search_ld)
587             search_ld = open_ldap_connection(ldapServer, port);
588
589         ldap_escape_value(escaped_login, sizeof(escaped_login), userid);
590         if (binddn) {
591             rc = ldap_simple_bind_s(search_ld, binddn, bindpasswd);
592             if (rc != LDAP_SUCCESS) {
593                 fprintf(stderr, PROGRAM_NAME ": WARNING, could not bind to binddn
                        '%s'\n", ldap_err2string(rc));
594                 ret = 1;
595                 goto search_done;
596             }
597         }
598         snprintf(filter, sizeof(filter), searchfilter, escaped_login, escaped_login,
                escaped_login, escaped_login, escaped_login, escaped_login, escaped_login,
                escaped_login, escaped_login, escaped_login, escaped_login, escaped_login,
                escaped_login, escaped_login, escaped_login);
599         debug("user filter '%s', searchbase '%s'\n", filter, basedn);
600         rc = ldap_search_s(search_ld, basedn, searchscope, filter, searchattr, 1, &res);
601         if (rc != LDAP_SUCCESS) {
602             if (noreferrals && rc == LDAP_PARTIAL_RESULTS) {
603                 /* Everything is fine. This is expected when referrals
604                  * are disabled.
605                  */
606                 debug("noreferrals && rc == LDAP_PARTIAL_RESULTS\n");
607             } else {
608                 fprintf(stderr, PROGRAM_NAME ": WARNING, LDAP search error '%s'\n",
                        ldap_err2string(rc));
609 #if defined(NETSCAPE_SSL)
610                 if (sslpath && ((rc == LDAP_SERVER_DOWN) || (rc ==
                        LDAP_CONNECT_ERROR))) {
611                     int sslerr = PORT_GetError();
612                     fprintf(stderr, PROGRAM_NAME ": WARNING, SSL error %d (%s)\n",
                            sslerr, ldapssl_err2string(sslerr));
613                 }
614 #endif
615                 ret = 1;
616                 goto search_done;
617             }
618         }
619         entry = ldap_first_entry(search_ld, res);
620         if (!entry) {
621             debug("Ldap search returned nothing\n");
622             ret = 1;
623             goto search_done;
```

```
624             }
625             userdn = ldap_get_dn(search_ld, entry);
626             if (!userdn) {
627                 fprintf(stderr, PROGRAM_NAME ": ERROR, could not get user DN for '%s'\n",
                        userid);
628                 ret = 1;
629                 goto search_done;
630             }
631             snprintf(dn, sizeof(dn), "%s", userdn);
632             squid_ldap_memfree(userdn);
633
634             if (ret == 0 && (!binddn || !bind_once || passwdattr)) {
635                 /* Reuse the search connection for comparing the user password attribute */
636                 bind_ld = search_ld;
637                 search_ld = NULL;
638             }
639     search_done:
640             if (res) {
641                 ldap_msgfree(res);
642                 res = NULL;
643             }
644             if (search_ld && search_ld != persistent_ld) {
645                 ldap_unbind(search_ld);
646                 search_ld = NULL;
647             }
648             if (ret != 0)
649                 return ret;
650         } else {
651             snprintf(dn, sizeof(dn), "%s=%s,%s", userattr, userid, basedn);
652         }
653
654         debug("attempting to authenticate user '%s'\n", dn);
655         if (!bind_ld && !bind_once)
656             bind_ld = persistent_ld;
657         if (!bind_ld)
658             bind_ld = open_ldap_connection(ldapServer, port);
659         if (passwdattr) {
660             if (ldap_compare_s(bind_ld, dn, passwdattr, password) != LDAP_COMPARE_TRUE)
                    {
661                 ret = 1;
662             }
663         } else if (ldap_simple_bind_s(bind_ld, dn, password) != LDAP_SUCCESS)
664             ret = 1;
665         if (bind_ld != persistent_ld) {
666             ldap_unbind(bind_ld);
667             bind_ld = NULL;
668         }
669         return ret;
670     }
671
672     // used for binddn -W opt
```

```
673 int
674 readSecret(const char *filename)
675 {
676     char buf[BUFSIZ];
677     char *e = NULL;
678     FILE *f;
679     char *passwd = NULL;
680
681     if (!(f = fopen(filename, "r"))) {
682         fprintf(stderr, PROGRAM_NAME " ERROR: Can not read secret file %s\n", filename);
683         return 1;
684     }
685     if (!fgets(buf, sizeof(buf) - 1, f)) {
686         fprintf(stderr, PROGRAM_NAME " ERROR: Secret file %s is empty\n", filename);
687         fclose(f);
688         return 1;
689     }
690     /* strip whitespaces on end */
691     if ((e = strrchr(buf, '\n')))
692         *e = 0;
693     if ((e = strrchr(buf, '\r')))
694         *e = 0;
695
696     passwd = (char *) calloc(sizeof(char), strlen(buf) + 1);
697     if (!passwd) {
698         fprintf(stderr, PROGRAM_NAME " ERROR: can not allocate memory\n");
699         exit(1);
700     }
701     strcpy(passwd, buf);
702     bindpasswd = passwd;
703
704     fclose(f);
705
706     return 0;
707 }
```