Linux authentication with Active Directory using Kerberos 5

TAMING THE DOGS OF HELL

Microsoft's Active Directory system provides centralized user management and single sign-on. If you're ready for a few manual steps, Linux can leverage this potential. **BY WALTER NEU**

In many enterprises, Linux and Windows now live together in peace. Very often, heterogeneous networks rely on Windows-dominated office software and traditional Unix-style servers. The Active Directory service, which Microsoft introduced with Windows 2000 Server, is often used for centralized user information management.

Linux typically uses the legacy */etc/passwd* system or a distributed solution such as NIS or LDAP, but if you are willing to configure a number of freely available tools and components, you can easily integrate your Linux systems into the Active Directory infrastructure.

In this article, I assume that you have an Active Directory server that manages a complete domain structure on Windows. With this, I'll show you how to configure your Linux clients to log in (authenticate), gain access (authorize), and leverage the domain infrastructure. The icing on the cake is single sign-on functionality, and the cherry on top is the ability to automatically create user directories on the client side.

The example in this article relies on the Samba project's Winbind service and Kerberos 5 for authentication. Of course, Kerberos was not invented by the software engineers in Redmond; Microsoft adopted this authentication method from the Unix world. Kerberos was originally developed at the Massachusetts Institute of Technology (MIT) in the 1980s. Both the free Heimdal [1] project and the MIT reference application [2] offer full Kerberos 5 support. Shishi [3] is another free implementation.

Well-Kept Secrets

Kerberos is a ticket-based network authentication service that relies on shared secrets. The system guards a logically separate area known as a realm, which can include a number of clients and services.

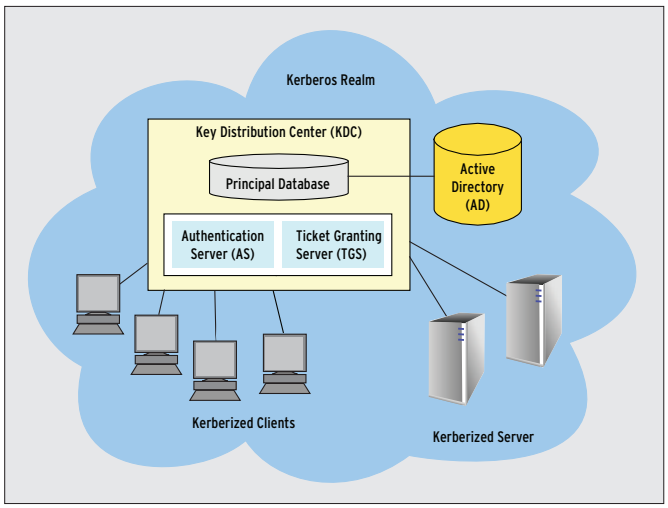

Figure 1: Kerberos is a ticket-based network authentication service that relies on shared secrets. In the Kerberos architecture, all the components belong to the Kerberos realm. The core of the realm is the Key Distribution Center (with its main components the AS and TGS) and the principal database.

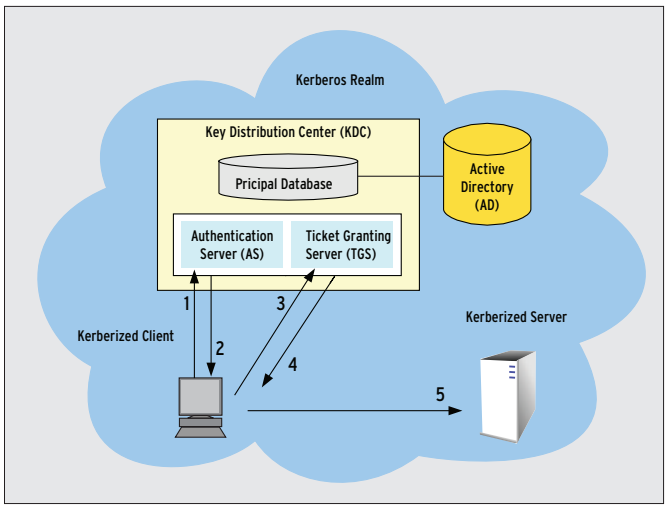

Figure 2: Authentication with Kerberos is sophisticated but flexible: a client sends a TGT request to the KDC (1) and receives a time-restricted ticket (2). The ticket authorizes the client to request (3) further tickets (4) for kerberized network services (5) without the need for additional password authentication.

In this example, the clients and a number of services, such as a file server, run on Linux. Windows handles directory services and authentication via the Key Distribution Center. The KDC is a central component in Kerberos (Figure 1) that includes the Authentication Server (AS) and the Ticket Granting Server (TGS).

At the start of a session, each member (or *principal*) of the realm demonstrates its authenticity once only. To do so, the principal requests an initial Ticket Granting Ticket (TGT) from the AS. It uses this ticket to apply to the TGS for further service tickets.

What Kerberos refers to as a ticket is an electronic credential. Once a principal has received a credential, it is granted access to "kerberized" applications that require proof of identity without the need to enter a password. Users just need to enter a password to receive the TGT.

Tickets Please!

The login program requests a TGT on behalf of the client (see Figure 2). Alternatively, the *kinit* can issue a request after the user logs on. The AS searches Active Directory for the requesting principal. Once the AS has found the principal, it issues a TGT.

The AS then encrypts the TGT with the principal's key and returns the hash to the requesting entity. If the requesting entity is a client, the KDC extracts the key from the user's password, encrypts it, and stores the hash in its principal database. The *login* program, or the *kinit*, calculates the secret key from the password entered by the user client-side and decrypts the TGT. The password is never transmitted in the clear.

When a user needs to access a kerberized service on a network, the user presents the TGT to the TGS and requests a service ticket for the service. The TGS issues the ticket in the background. Now that the client has the service ticket, it can automatically log the user in to the requested service without asking for a password.

Kerberos tickets have a limited lifetime. The time problem makes it essential to synchronize the system time on all the computers in the realm. The Kerberos server will refuse to issue an initial ticket to a machine that is out of sync by more than five minutes.

Listing 1: /etc/kr5b.conf

```
01 [libdefaults]
02   default_realm = KDC.EXAMPLE.ORG
03   dns_lookup_realm = false
04   dns_lookup_kdc   = false
05 [realms]
06   KDC.EXAMPLE.ORG = {
07    kdc = w2k.kdc.example.org
08    default_domain = KDC.EXAMPLE.
   ORG
09   }
10 [domain_realm]
11    .example.org = KDC.EXAMPLE.ORG
```

Although you can change the maximum clock skew through the Kerberos client or the Active Directory server, it makes more sense to set up a central timeserver to allow clients to synchronize.

The clients must also be capable of resolving the Kerberos server's DNS name. If necessary, you can add a record on the central nameserver or simply maintain the static */etc/hosts* file on all the systems involved in the exchange.

Installing Kerberos

After attending to the time and name resolution requirements, you can install Kerberos on your Linux clients from your distribution's packages. For the MIT variant with Ubuntu, you need the *krb5-user* and *krb5-config* packages from the Universe repository, or, if you use Fedora, *krb5-workstation* and *krb5-auth-dialog*. As an alternative, you might prefer to build the MIT sources.

To configure Kerberos, modify the */etc/kr5b.conf* file. Listing 1 shows a minimal but functional configuration with the MIT package; clients need this to set up a connection to the Kerberos server. The other Kerberos implementations use more or less the same syntax.

Building Realms

The *default_realm* line in the *[libdefaults]* section sets up a realm called *KDC.EXAMPLE.ORG* as the default for Kerberos applications. If you are using

multiple realms, you can add another expression to the *[realms]* section. The *[domain_realm]* section sets the domain name/realm link in the Kerberos library. If you want the Kerberos library to establish a connection to a remote host, the library needs to know the realm in which the host resides. Entries that start with a dot assign all hosts with the following suffix to the specified Kerberos realm. To ensure trouble-free communications with the Kerberos server, it is important to use uppercase characters for the realm name.

With the use of this configuration, you can test communications with the Kerberos server. The *kinit* command requests a TGT. If you do not specify any additional parameters, the program attempts to secure a TGT for the principal with the same name as the logged on user. To allow this to happen, the user needs to enter a password once only.

The *kinit* program now sends an unencrypted TGT request to the authentication server; the request includes the name of the principal (among other things). The response sent to the client includes the encrypted TGT, which *kinit* decrypts and stores locally.

The output from the *klist* command in Listing 2 includes the validity data for the TGT that was just issued. If the command output shows the ticket, you can assume that the Linux client configuration is complete. To destroy the test TGT, use *kdestroy*.

Membership

The next step is to add the Linux client as a member of the Active Directory domain. To allow this to happen, you need to install Samba version 3.0.14a or newer and the Winbind program package for centralized user management in Windows and Linux. Winbind uses a Unix implementation of Microsoft's RPC calls, the Pluggable Authentication Modules (PAM), and the Name Service Switch (NSS) to let users with Linux clients log in to the Windows domain and work as local users.

Samba is configured in the *smb.conf* file, which is typically found below */etc/samba/*. A complete sample configuration, which implements an Active Directory domain member server with the required Winbind configuration, is shown in Listing 3.

The *security = ads* parameter in line 5 tells Winbind not to look for the password in the local user database but to pass the request on to an Active Directory domain controller. The domain controller then decides whether the password is legitimate.

If you have a Windows 2003 AD domain controller, you need to set *client schannel = no* in the *[global]* section. Before the client becomes a domain member, the admin tells it (in line 6) which Kerberos realm the principal belongs to.

Listing 2: klist Showing Tickets

```
01 $ klist
02 Ticket cache: FILE:/tmp/
   krb5cc_1000
03 Default principal: user@KDC.
   EXAMPLE.ORG
04
05 Valid starting     Expires
   Service principal
06 03/17/08 11:10:27  03/17/08 21:10
   krbtgt/KDC.EXAMPLE.ORG@KDC.
   EXAMPLE.ORG
07        renew until 03/18/08 11:10
08
09 Kerberos 4 ticket cache: /tmp/
   tkt1000
10 klist: You have no tickets cached
```

Centralized User Management

Membership in a domain only removes the need for the Linux system to manage passwords; it does not remove the need to manage user entries. Domain users are still an unknown property on the system at this point. Unix-style operating systems need the *winbindd* daemon to ensure visibility. The Samba suite component program uses the Name Service Switch (NSS) to resolve domain user's identities and serve them up to Linux as if they were local credentials.

While Winbind is running, it temporarily transfers all the users and groups in Active Directory to the Linux system. This substantially reduces the administrative overhead for user management. Winbind is configured centrally in the *[global]* section of *smb.conf* (lines 15 through 20).

The *workgroup = kdc* instruction in line 3 is noteworthy: Samba uses *workgroup* to define both a workgroup and a domain. The Samba program decides what to configure later in the configuration process. The AD domain is stored in NT4 syntax here; in other words, if you have a Windows 2003 domain of *kdc.example.org*, Samba will expect *kdc*.

The Samba parameter in line 6 configures the realm; this is normally the domain controller's DNS name, but in upper case letters -- that is, *KDC.EXAMPLE.ORG* in this case.

Separation

The character separating the domain and user name in Windows is the backslash \, however, this character has a special meaning for the shell. To avoid conflict, admins should set *winbind separator* to avoid the use of a shell metacharacter and choose a plus sign (+), as shown in line 16 of Listing 3.

If you only have one domain, you don't need to separate the domain and user names. Winbind provides the setting *winbind use default domain = yes* in the global section of the configuration file. This parameter tells Linux to use Active Directory user names without the domain element. If you do not set this, you will need to add a domain name prefix to the domain users served up by Winbind to use them on Linux (see Fig-

Listing 3: smb.conf

```
01 [global]
02 ; Samba as a domain member
03    workgroup = kdc
04    password server = srv.kdc.
   example.org
05    security = ads
06    realm = KDC.EXAMPLE.ORG
07    encrypt passwords = yes
08
09 ; not the master browser for the
   Windows network
10    local master = no
11    os level = 20
12    domain master = no
13    preferred master = no
14
15 ; Winbind configuration
16    winbind separator = +
17    idmap gid = 10000-20000
18    idmap uid = 10000-20000
19    template shell = /bin/bash
20    template homedir = /home/%D/%U
21   winbind enum users = yes
22   winbind enum groups = yes
```

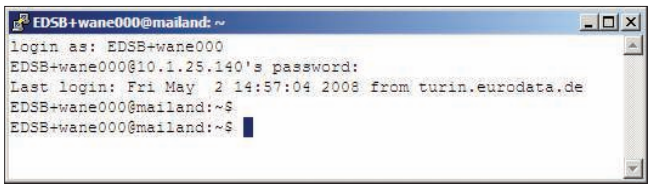

Figure 3: Because the SSH server identifies more than one realm, the user wane000 needs a domain prefix of EDSB. The + character separates the two names.

ure 3).

Left to its own devices, the Linux system is unable to convert domain user and group names to their numeric counterparts: User Identification (UID) and Group Identification (GID). However, this is necessary because Linux does not use names internally, relying on the UID and GID instead. For example, the *ls* command parses a file's inode to discover its owner's UID and translates this value to a name before displaying the information on screen.

Linux uses a universal API, NSS, for mapping names. NSS can search the */etc/passwd* file, or assuming you have the module loaded, query an Active Directory server. This capability lets you list the users and groups in an ADS realm as if they were local accounts. To allow this to happen, you need to add the *winbind* name service to the *passwd* and *group* databases in the central */etc/nsswitch.conf* configuration file:

```
passwd: files winbind
group:  files winbind
```

These lines tell the name service to start by searching local files such as */etc/passwd* before contacting *winbindd*. If you additionally run NIS, you can type *compat* instead of *files*.

One thing still standing in the way of successful cooperation between Linux and the Windows-based Active Directory Service is that the Linux computer needs to become a domain member to receive user and group information for the domain.

The *security = ads* parameter in line 5 adds Samba as an Active Directory domain member. The *net ads* command, which is part of the Samba distribution (see Figure 4), completes the transaction. The domain user, *Administrator* in this case, must be authorized to add the Linux computer to the domain. *net* prompts you to enter the password for the authorized user and, if the password is correct, creates the computer account on the domain controller. If this all works out, the Linux client is now a full member of the Active Directory environment.

To test whether the connection to the domain controller is working properly, run the *wbinfo* diagnostics tool. This tool is part of the Winbind package. The *-u* parameter tells the command to list all the domain users available in the domain:

```
KDC+wneu
KDC+mkreis [...]
```

The domain you are using here is called *KDC*. The domain name is followed by the separator configured as your *winbind separator*, + in this case, and the user name. The names re-

Figure 4: To add the local machine as a member of the standard domain, the user on the domain controller (DC) needs administrative privileges. After entering the password, the DC adds the new client to the domain.

trieved from Active Directory are now known to Linux and can be used to log in. The groups defined in Active Directory can be listed by calling *wbinfo -g*:

```
KDC+accounts
KDC+asp [...]
```

To output an overview of all known users and groups in the domain or local databases, use *getent passwd* or *getent group*. The output is similar to */etc/passwd* and */etc/group*.

Now test whether Linux can identify the user and group names in your Active Directory: If the Linux system administrator can assign the ownership and group ownership of a file stored on a Linux machine to a domain user and group in Active Directory, you're winning! Depending on the *winbind use default domain* parameter in your Samba configuration, root can specify the owner as *Domain + User* and the group as *Domain + Group* (Listing 4).

Wedding Kerberos to PAM

The next trick is to integrate Kerberos, the Active Directory domain users, and the Linux login mechanism. Formerly, each of these services expected users to authenticate, then each applied its own authentication and authorization mechanisms to grant users access to the services it provided. The Pluggable Authentication Modules (PAM) provide a unified interface for this integrated authentication [4].

Listing 4: Changing Ownership

```
01 # ls -l foo.txt
02 -rw-r--r-- 1 root root May 02 15:53
   foo.txt
03 # chown KDC+wnew foo.txt
04 # chgrp KDC+asp foo.txt
05 # ls -l foo.txt
06 -rw-r--r-- 1 KDC+wneu KDC+asp May
   02 15:53 foo.txt
```

Changing the authentication method in PAM means changing and serving up matching modules that all programs can then access. In other words, PAM adds an abstraction layer between authentication and the actual services but without needing to change applications. Applications such as FTP and Telnet servers connect to an authentication service by calling PAM library functions that are available as shared libraries.

A special module library is available to change the user login authentication method to Kerberos via the Pluggable Authentication Modules. Packages for the library are available for most popular distributions. The module itself is called *pam_krb5.so*, and it typically resides in */lib/security* [5].

Individual Configuration

The module not only handles the Kerberos-based login, but transparently requests a TGT from the Authentication Server on behalf of the user. Getting this to work involves changing a number of configuration settings in the */etc/pam.d/* directory.

Each application that requires authentication and uses PAM requires an individual file in */etc/pam.d/*. Distributions tend to organize the configuration in slightly different ways, and some of them import shared files. Each line in these files includes the type, a control flag, a path to the module in question, and optional arguments, all of which are separated by blanks (see Listing 5). Fedora uses the *authconfig* tool, OpenSUSE relies on YaST for manipulating the PAM configuration, and Debian users need to fire up their favorite editor to manually modify the files.

Section by Section

The configuration files are divided into sections for the four PAM module types: *auth*, *account*, *password*, and *session*. The *auth* section defines two alternative authentication methods in order to determine whether the users are who they claim to be (lines 2 and 3). PAM prompts the user for a password once only, and Kerberos checks the credentials (line 2). If this step is successful, *pam_krb5.so* requests a TGT with a set forward-capable bit to use the ticket on a remote system.

If successful, the authentication process tags *pam_krb5.so sufficient* and terminates without processing any additional modules.

Second Chance

If authentication fails, PAM calls the second module, *pam_unix.so*, to check that the user account exists locally (line 3). If PAM fails to reach the authentication server, root can still log in. The module argument *use_first_pass*, means the second authentication method reuses the password entered by the user instead of prompting again. Thanks to *nullok_secure*, there is no need to set a password in the local password file. Although you can log in with a blank password, you can only do so on terminals listed in */etc/securetty*.

Listing 5: PAM Configurations

```
01 # /etc/pam.d/common-auth
02 auth     sufficient pam_krb5.so forwardable
03 auth     required   pam_unix.so nullok_secure use_first_pass
04 auth     required   pam_deny.so
05
06 # /etc/pam.d/common-account
07 account  sufficient pam_krb5.so forwardable
08 account  required   pam_unix.so
09
10 # /etc/pam.d/common-session
11 session  sufficient pam_krb5.so
12 session  required   pam_unix.so
13
14 # /etc/pam.d/common-password
15 password sufficient pam_krb5.so nullok obscure md5
16 password required   pam_unix.so nullok obscure md5
```

IT University of Gothenburg
October 24-26 2008

Feel Free

Free Society Conference and Nordic Summit
October 2008
Gothenburg, Sweden

Agile, Arduino, ccMixter, Coreboot, Debian,
Google, Gtk, Jabber, KDE, Magnatune,
Midgard, Open Street Map, Oscar Swartz,
PostgreSQL, QT, Skolelinux, Tango,
Ubuntu and many more

www.fscons.org

Free Software | **Free Culture** | **Free Content**

fsfe

creative commons.se

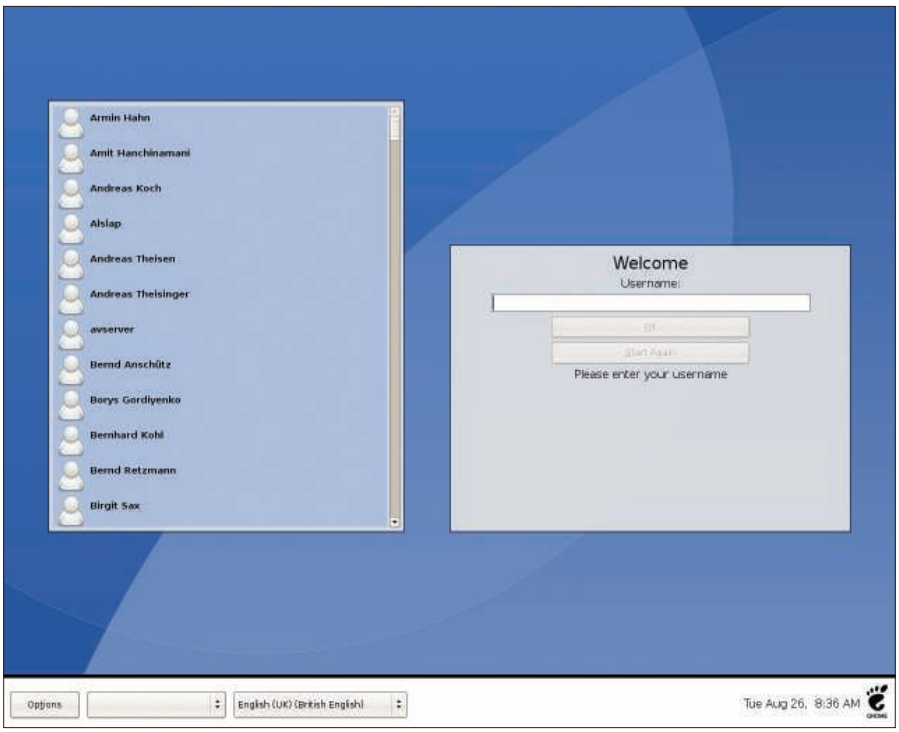

Figure 5: The Gnu Display Manager (GDM) displays both local and Active Directory users as login candidates.

After PAM has finished processing the *auth* type modules, it goes on to execute the next *include*. The *common-account* file again contains two modules, *pam_krb5.so* and *pam_unix*, and is responsible for handing system access.

The PAM *account* service checks the user's password to see if it is still valid or if user access to the system is restricted with respect to time, resource usage, or location.

If the user exists and is allowed to log in, PAM turns to the next stack of modules, which is described by the *common-password* file. These modules give users the ability to change their passwords. In contrast to the normal procedure on Linux, which only allows users to change their own passwords, the Kerberos module gives any user the ability to change any other user's password. However, to do so, the user needs to know the current password for the account they are changing.

Session Management

Finally, PAM calls the modules configured in *common-session*. The *session* type is responsible for all the additional authentication chores. The remaining steps can include setting variables or mounting directories. Within the scope of this PAM service, *pam_krb5* fulfills a very important task: it deletes a user's tickets when the user logs off.

After completing the configuration, any PAM-capable program can use PAM to access Active Directory. For example, the GNU Display Manager (GDM) offers both local and domain user accounts as login candidates and launches the required session after authentication with the Kerberos server (Figure 5).

As a final step, Active Directory users will need a home directory. If the NSS does not have any details on the directory, or if the directory does not exist, Linux will either send the user off to the root directory or it will not let the user access the system – despite successful authentication – because a desktop environment such as KDE needs to read and write to certain files that just do not exist.

/home, Sweet /home

The home directories are configured in the line 20 of the *smb.conf* file shown in Listing 3: *template homedir = /home/%D/%U*. Samba will replace *%D* with the short domain name and *%U* with the domain user. The administrator can either create the directories individually for each user, or automate the process by calling the *pam_mkhomedir* module, which is part of the PAM distribution and is configured in the *session* section:

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

This configuration tells the module to dynamically create missing home directories. The *silent* argument suppresses messages caused by copying from the skeleton directory. The last argument tells PAM to set the *umask* as the default for file and directory permissions to *0022*. The setting allows programs running in the session to create directories with *rwxr-xr-x* and files with *rw-r--r--* permissions.

As an alternative to local directories on kerberized clients, you could use home directories on a central file server. The PAM *pam_mount.so* module helps you do this. Any generic commands you want to run after the login procedure are added to the start scripts in */etc/profile*.

Fully Integrated

Serveral steps are required to support Active Directory automated log in and home directories on a Linux client, but with Kerberos, NSS, PAM, and Samba, this integration project will help you stay friends with your neighbors in Redmond. ■

INFO

[1] Heimdal Kerberos: *http://www.h5l.org*

[2] MIT Kerberos: *http://web.mit.edu/kerberos/*

[3] Shishi Kerberos: *http://josefsson.org/shishi/*

[4] Linux PAM: *http://ftp.kernel.org/pub/linux/libs/pam/*

[5] Pam-krb5: *http://www.eyrie.org/~eagle/software/pam-krb5/*

THE AUTHOR

Walter Neu works as a system administrator for eurodata GmbH & Co. KG, Germany. He is a lecturer at the ASW – Berufsakademie Saarland University of cooperaive education, where he introduces computer science and economics students to Linux, Windows networking, and web server technologies.