## Kerberos-based NIS authentication

# Trusty Ticket Inspector

A combination of the Network Information Service (NIS) and Kerberos allows admins to manage large numbers of users while at the same time providing secure login facilities. Kerberos' cryptographic abilities, which provide a secure replacement for insecure IP services, are the key to this approach.

BY THORSTEN SCHERF


If you use the Network Information Service for central user management, you have a security problem. Especially so, if you allow NIS to handle the shadow password database. The idea of the shadow file is to prevent users from accessing the encrypted passwords. If NIS handles this file, the password hash crosses the wire each time a user logs on. An attacker may try to sniff out the data with a sniffer such as Ethereal, before trying a brute force or dictionary attack. There are any number of password crackers around, such as John or Crack.

Enter Kerberos [1]. Kerberos uses a mixture of cryptography and a transport paradigm to protect your passwords. In this article, we will be looking at NIS version 2, and using Kerberos 5 to authenticate. Although NIS version 3 has been released, there is only a client available for Linux at present. The server is still under development. NISv3, a.k.a NIS+, is also far more difficult to configure than the version we will be looking at. If you need more details on NIS+ check out the NIS how-to at [2].

### The Three-Headed Dog

Admins often use firewalls to solve their security problems. Not all threats are external. In fact, most unauthorized access stems from the local network, where an insider has an easier job of sniffing clear text passwords. An authentication software such as Kerberos can help close this hole. Programmers at the Massachusetts Institute of Technology (MIT) developed Kerberos to provide secure authentication for client-server applications. To do this, the software uses strong cryptography and a ticket-based authentication protocol.

Common network services like FTP or POP3 can endanger the security of a network, as user names and passwords travel between the client and the server in clear text. Kerberos removes this danger by preventing passwords from crossing the wire, and thus making sniffing impossible. Kerberos also makes spoofing and replay attacks more difficult, and provides single sign-on facilities. Clients authenticate with the network to gain access to the

**Listing 1: NIS Client Configuration**

```
01 # Valid entries are
02 #
03 # domain NISDOMAIN server HOSTNAME
04 # Use server HOSTNAME for the domain NISDOMAIN.
05 #
06 # domain NISDOMAIN broadcast
07 # Use broadcast on the local net for domain NISDOMAIN
08 #
09 # ypserver HOSTNAME
10 # Use server HOSTNAME for the local domain. The
11 # IP-address of server must be listed in /etc/hosts.
12 #
13 # broadcast
14 # If no server for the default domain is specified
15 # or none of them is reachable, try a broadcast call
16 # to find a server.
17 #
18 domain EXAMPLE server nis1.notexample.com
```

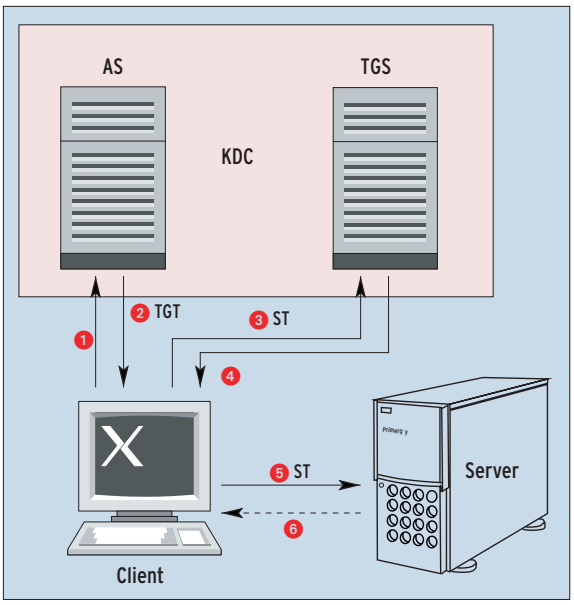


**Figure 1: In a Kerberos session the password will never cross the wire in the clear.**

whole range of services managed by Kerberos.

What Kerberos actually does is to handle the authentication process. It does not provide user facilities, such as user IDs, login shells, or home directories. Instead, these facilities are handled by a directory service such as NIS (see box Setting Up NIS).

## Distributed Mechanism

A distributed mechanism ensures a high level of security (see box Events in a Kerberos Session). The client opens up a connection to a key distribution center (KDC) (see Figure 1). This process is transparent to the user and is handled either by the *login* program or the *kinit* client. The KDC comprises two parts: the authentication server (AS) and the ticket granting server (TGS). The authentication server waits for client requests and checks whether the user principal is permitted to access services within the realm (namespace).

Assuming that the principal exists within the Kerberos database, the AS generates a random session key and a so-called ticket granting ticket (TGT). The TGT includes various details, such as the client's hostname and IP address, the ticket validity date, a timestamp and the session key.

Kerberos uses a key, which is known only to the authentication server and the ticket granting server to encrypt the TGT. The server sends the ticket to the client along with the session key for the current session. To prevent sniffing, Kerberos encrypts the ticket with a key consisting of a client password hash.

## Enter the Password

After receiving the response from the authentication server, in the form of an encrypted TGT and session key, the local system displays a login password prompt. The client converts the password to a DES key, which it uses to decrypt the TGT it has received. The client then stores the TGT in its credential cache, and deletes the password from memory. A user can prove his identity by producing the TGT as long as the ticket is valid. When the ticket expires, the user will be required to log on once more, and the whole procedure is repeated.

The password and the TGT ensure the authenticity of the users on the local workstation. If the user then needs to access a network service like FTP, he needs to request a service ticket from the key distribution center. To do so, the client contacts the ticket granting server (TGS). The service ticket (ST) handles one specific service, the one the client is requesting, in this case FTP.

## Requesting a Service Ticket

Requesting a service ticket is far more complex than obtaining a TGT. The client sends a request to the TGS. The request is made up of the name of the service the client wants to access (authenticator), and the stored TGT. The authenticator includes the client name, its IP address, and the time client-side. The authenticator is encrypted with the TGT session key. The client sends the encrypted TGT and the authenticator to the ticket granting server. The TGT decrypts the authenticator and the TGT, and then compares the content, the IP address, and the time. If the credentials are identical, the server generates a new session key, which will allow the client to access the requested service (the FTP server in this case) in future. The new session key is included in the service ticket which the ticket granting server now issues. The server encrypts the whole caboodle with the TGT session key and bundles it off to the client.

Then the ticket touting process continues. The client receives the service ticket and hands it to the required server (FTP)

## Setting up a NIS Client

The NIS client configuration is quite simple on Red Hat Linux, as the *authconfig* tool facilitates the setup procedure (see Figure 2). You need to enter the NIS domain and the NIS server. The tool then stores this information in */etc/yp.conf* (see Listing 1). Authconfig then goes on to modify your */etc/sysconfig/network* file, again adding the NIS domain name, which will be parsed when you reboot. Authconfig's third step is to configure the nameserver switch file, */etc/nsswitch.conf*. The file specifies where, and in what order, the client will search for information such as password or host files. The following entries are required:

```
passwd: files nis
shadow: files
group: files nis
```

In this example, the client will first inspect its local */etc/passwd* and */etc/group* files, and then look at the NIS maps, *passwd* and *group*. You can omit the *files* entry, if you do not need local authentication. However, this will prevent a local root login. Authconfig then modifies the PAM file, */etc/pam.d/system-auth*, ensuring that a user password on the NIS server is changed if it belongs to a NIS account.

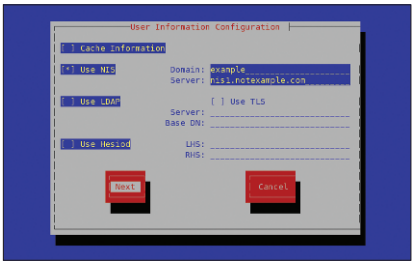


**Figure 2: Admins can use Authconfig to enter the name of the NIS domain, and the NIS server in the corresponding files.**

Users can then use *yppasswd* to change their passwords. Finally, the tool launches the NIS client service, *ypbind*, in the background to open the connection to the NIS server.

in order to prove its (the client's) identity. To accompany the service ticket, the client again creates an authenticator and sends it to the server. If the details of the ST and the authenticator check out, the server can validate the client. The client is authenticated and will not need to produce its user name and password to identify itself to the server again.

```
[root@kermit root]# kinit thorsten
Password for thorsten@EXAMPLE.COM:
[root@kermit root]# klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: thorsten@EXAMPLE.COM

Valid starting     Expires            Service principal
01/15/04 20:53:43  01/16/04 06:53:43  krbtgt/EXAMPLE.COM@EXAMPLE.COM
        renew until 01/15/04 20:53:43


Kerberos 4 ticket cache: /tmp/tkt0
klist: You have no tickets cached
[root@kermit root]#
```

**Figure 3: The Authconfig configuration program provides a form where you can enter the realm and Kerberos domain controller for a server.**

## No Protection Without NTP

The authenticator provides effective protection against attackers sniffing the network traffic and capturing a service ticket, which they could then replay to the server to gain access. To allow client authentication to work, it is absolutely essential to synchronize the time across all the machines on the local network. A NTP (Network Time Protocol) server can be used to do so [3]. However, NTP is an insecure IP service itself, and needs encryption to protect it from prying eyes. Again check out [3] for approaches to encrypting NTP.

## The Kerberos Server

The Kerberos server manages a database in which the principals are stored. The Kerberos machine needs special protection, and should be set up in a room, or at least a rack, of its own. You should avoid running any other services on the Kerberos machine to rule out potential security holes.

Principals exist both for uses and for Kerberos-based services and hosts. A principal has the following structure: *Primary/instance@REALM*. The instance is optional, and serves to group primaries. A user principal might look like the following example: *thorsten/admin@NOTEXAMPLE.COM*. A FTP server prin-

## Setting Up NIS

A NIS server manages user and host information, making it available to clients in the same NIS domain. Administrators can use NIS to centralize user and host management. Users can access the database with any NIS client to authenticate (see Box Setting Up a NIS Client). To allow this, admins simply allow NIS to manage the */etc/passwd* and */etc/group* files. The NIS server can also handle shadow passwords, providing glibc is available. This will not work with older libraries such as libc5.

### Configuring the NIS Server

Before starting to configure the server, the admin first needs to assign a name to the new NIS domain. This should not be the DNS domain name, as intruders will often try this name out first. In this article, the NIS domain is called *example*. We created a *NISDOMAIN=example* entry in */etc/sysconfig/network*. Now for the configuration steps proper, starting with configuring the Makefile, which is located below */var/yp*.

Most default entries should be fine. *MINUID* specifies the number of the lowest user ID that NIS should manage. *MINGID* does the same thing for the user groups. In our example, we need to set *MERGEPASSWD* to *false*, as NIS will not be validating passwords. The last step is to tell NIS to manage our user accounts and groups by setting *all: passwd group*. NIS can do more than this, but the other settings are not relevant to our sample configuration.

NIS does not actually use the */etc/passwd* and */etc/group* ASCII files, but instead uses the *makedbm* tool to create GDBM files, which are known as maps in NIS-speak. Makedbm creates two NIS maps from each file. Each map is sorted by different criteria. The passwd map is sorted by login name (*passwd.byname*) and by user ID (*passwd.byuid*). After entering the following command

```
/usr/lib/yp/ypinit  m
```

to initialize the server, you will find the NIS maps in a folder below */var/yp*. The folder uses the same name as your NIS domain. The NIS database has entries for the user accounts that exist at this point. If you add users or groups later, you will need to add them to your NIS maps. Admins can do this using the *make -C /var/yp* command.

The NIS server starts listening when you launch the *portmap* and *ypserv* services. To launch these services, modify the entries for the appropriate runlevel directories below */etc/init.d/*.

### Redundancy and Security

If you have a centralized directory service such as NIS, it makes sense to add a backup server. This allows users to continue to access the network if your primary NIS server goes down. The secondary server is easy to set up. Simply supply the IP address of the primary when issuing the *ypinit* command:

```
/usr/lib/yp/ypinit -s ➥
Master-Server-IP
```

From now on, the second machine will have the same maps as your master server. There are two steps to complete beforehand: the secondary server must be set up as a NIS client, and the */var/yp/ypservers* file on the master must point to the secondary.

Admins should also put some thought into securing their NIS servers. Any NIS client can display any file that NIS manages. Calling *ypcap passwd* is all it takes to display the passwd map. Passwd is an alias for *passwd.byname* and *passwd.byuid* in this case. The */var/yp/nicknames* file has more aliases.

Although the client will not be able to access the encrypted user passwords, as the *MERGEPASSWD=false* statement in the Makefile prevents this, the server will provides a mass of information about user names. Admins should restrict access to the NIS maps to prevent this. This is exactly what */var/yp/securenets* (see Listing 2) does. The file is a list of permitted networks. At IP level, you can use packet filter rules to restrict access to the portmapper. A netfilter firewall rule might be as follows, for example:

```
iptables -t filter -A FORWARD ➥
-p udp -dport 111 -s ➥
192.168.0.0/24 -d 192.168.0.100➥
 -j ACCEPT
```

In this example, 192.168.0.0/24 is the network with access permissions, and the portmapper resides at 192.168.0.100. This assumes connection tracking [7] and some modifications to the standard policy for this chain. However, you should note that these restrictions will also apply to any other services that rely on the portmapper. The same thing applies if you use TCP Wrappers to protect the portmapper.

cipal might look like this: *ftp/station1.notexample.com@ NOTEXAMPLE.COM*.

The realm is a kind of catch basin for the principals in a specific area, and looks like the capitalized DNS domain name. The Kerberos database stores the user passwords and services along with the principals.

The server is quite simple to configure. Enter the name for your Kerberos realm in */etc/krb5.conf* (see Figure 3). Use the *kdb5_util create* command to create the database in */var/kerberos/krb5kdc*. You can use the *kadmin.local* tool for local database management or *kadmin* for remote management. This assumes that the *kadmin* service is running on the KDC, and that a valid admin principal has been added to */var/kerberos/krb5kdc/kadm5.acl*.

To add a new principal to the database, launch the management tool and then call *add_principal*, for example:

```
add_principal -pw ➥
Password thorsten
```

Adding a service or workstation principal follows the same steps:

```
add_principal -pw ftp ftp/ftp.➥
notexample.com
add_principal -pw host ➥
host/station1.notexample.com
```

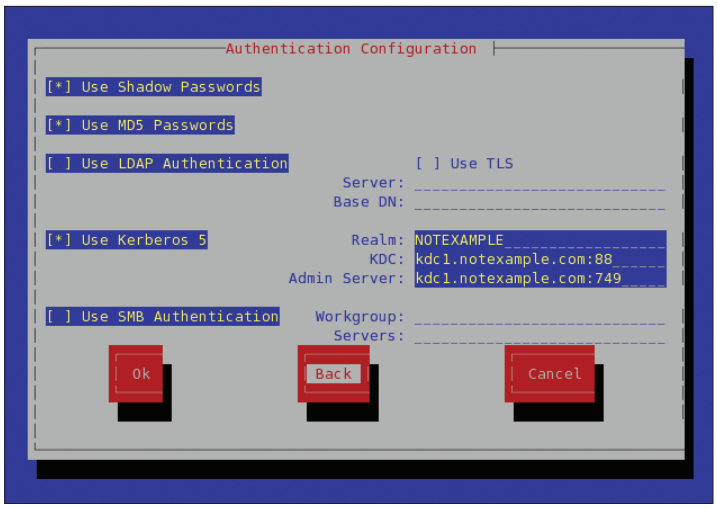


**Figure 4: A user ticket after *kinit*.**

Service principal passwords must be known to the servers themselves. This is achieved by extracting the service password from the Kerberos database:

```
ktadd -k /etc/krb5.keytab ➥
host/station1.notexample.com
```

After performing this step, securely copy the */etc/krb5.keytab* file to the machine running the service, using *scp*, for example. Then launch KDC, by typing *service krb5kdc start* (for Red Hat), to make the Kerberos server available on the network.

## Using Kerberos Services

To use Kerberos-based services on a network, you first need to install these services on the computers that require access. Kerberos can handle multiple services. As an example, */usr/kerberos/sbin* includes a Kerberos variant of the FTP daemon, *ftpd*. The matching configuration file, *gssftp*, is located in */etc/xinetd.d*. You can use the file to stipulate whether Xinetd should launch the service, among other things.

Kerberos can secure any service that provides a GSS API (Generic Security Service, [4]). Admins should note that each TCP/UDP port can be bound to a single service only. If you decide to deploy the Kerberos version of *ftpd*, you will thus need to disable the FTP daemon used previously, or preferably remove this daemon completely. You will also need to configure */etc/krb5.conf* (see Listing 3) on the servers.

## Kerberos Client Configuration

Admins with Red Hat can use the Authconfig tool (see Figure 3) to configure user workstations, just like for NIS (see Box Setting up NIS). Enter the Kerberos realm, the KDC, and the admin server, which are running on the same machine in our example. The Authconfig tool adds these parameters to */etc /krb5.conf* (see Listing 3) and */etc/pam.d/system-auth*.

The Kerberos client calls the PAM module, *pam_krb5.so*, in *system-auth* so as to pass the username to the KDC (or to the authentication server to be more precise), thus requesting a TGT. As an alternative, you can replace the login program, */bin/login*, with the Kerberos version, which is located in the */usr/kerberos/sbin* directory.

After providing user credentials in a virtual console to authenticate to a workstation with this configuration, the user is assigned a valid TGT. *klist* or *klist -5* (that is, Kerberos version 5 tickets only)

### Events in a Kerberos Session

- The user logs on to the local machine and sends a login request to the authentication server (AS).
- The authentication server responds with a ticket granting ticket (TGT).
- The client needs to establish a connection to a Kerberos service, and requests a service ticket (ST).
- The ticket granting server (TGS) issues the requested service ticket.
- The client passes the service ticket to the Kerberos service.
- The user is authenticated, the connection established, and the user does not need to log on again, while the ST is valid.

### Software

We used Red Hat 9 for this article. The examples will work with most other distributions, or versions such as RHEL 3 and Fedora Core 1. The following RPM packages [6] must be installed:

- krb5-workstation-1.3.1-6
- krb5-libs-1.3.1-6
- krb5-server-1.3.1-6
- pam_krb5-2.0.4-1

### Listing 2: Securenets

```
01 # Explicitly permit access for
   localhost:
02 host   127.0.0.1
03
04 # Permit access for host on
   192.168.0.0/24 network:
05 255.255.255.0   192.168.0.0
06
07 # Permit global access
   (insecure!):
08 #0.0.0.0   0.0.0.0
```

allows the user to validate the ticket. The klist program not only displays the received tickets, but additional information, such as the name of the service principal and the validity details. A ticket is normally valid for ten hours, but you can modify this default in */etc/krb5.conf*. Users can change their passwords on the Kerberos server, using the *kpasswd* tool to do so.

## What About Windows?

Microsoft clients can also use the Linux KDC for authentication purposes [5]. After all, a Windows domain controller is simply a combination of a Kerberos server and a LDAP (Lightweight Directory Access Protocol) server. To make this work, you need to define host principals for your Windows machines in your Kerberos database. Windows clients also need to know which KDC to use, the realm, and the host password.

It is quite easy to do this using Ksetup, which is included with the Windows 2000 Resource Kit. However, for this to work, your Windows clients will need to obtain user information from a directory service (such as LDAP). Unfortunately, NIS is not suitable as it requires a Unix-based server.

### Listing 3: Kerberos Client Configuration

```
01 #/etc/krb5.conf
02 [logging]
03 default =
   FILE:/var/log/krb5libs.log
04 kdc =
   FILE:/var/log/krb5kdc.log
05 admin_server =
   FILE:/var/log/kadmind.log
06
07 [libdefaults]
08 ticket_lifetime = 24000
09 default_realm = NOTEXAMPLE.COM
10 dns_lookup_realm = false
11 dns_lookup_kdc = false
12
13 [realms]
14 NOTEXAMPLE.COM = {
15 kdc = kdc1.notexample.com:88
16 admin_server =
   kdc1.notexample.com:749
17 default_domain =
   notexample.com
18 }
19
20 [domain_realm]
21 .example.com = NOTEXAMPLE.COM
22 example.com = NOTEXAMPLE.COM
23
24 [kdc]
25 profile =
   /var/kerberos/krb5kdc/kdc.conf
26
27 [appdefaults]
28 pam = {
29 debug = false
30 ticket_lifetime = 36000
31 renew_lifetime = 36000
32 forwardable = true
33 krb4_convert = false
34 }
```

## No Exceptions

As we have seen, Kerberos can considerably increase the security of a network by preventing clear text passwords from crossing the wire. Sadly, Kerberos is no help at all if the users continue to authenticate against services that do not belong to the Kerberos realm. If Kerberos cannot handle a service that you are running, it might be preferable to discontinue the service. If you need secure authentication, there can be no exceptions. ■

### INFO


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

[2] NIS HOWTO: *http://www.linux-nis.org/nis-howto*

[3] NTP: *http://www.eecis.udel.edu/~mills/ntp/servers.html*

[4] GSS API RFC: *http://www.faqs.org/rfcs/rfc1964.html*

[5] Kerberos 5 Interoperability: *http://www.microsoft.com/windows2000/techinfo/planning/security/kerbsteps.asp*

[6] Red Hat FTP-Server: *ftp://ftp.redhat.com/redhat/linux/9/en/os/i386/RedHat/RPMS*

[7] Connection Tracking: *http://www.sns.ias.edu/~jns/security/iptables/iptables_conntrack.html*



THE AUTHOR

*Thorsten Scherf a network security specialist who works for Red Hat Linux, where he is involved in project deployment and training.*