PRESENTED BY RED HAT

LEARN. NETWORK.
EXPERIENCE OPEN SOURCE.

www.theredhatsummit.com

Interoperability with Windows using CIFS File Sharing with Kerberos Authentication

Jeff Layton
Senior Software Engineer, Red Hat
June 23, 2010

Who is this person?

- Around a decade of work as Unix sysadmin
- Member of file system engineering team at Red Hat since 2006
- Joined worldwide Samba team in 2008
- Primarily work on NFS and CIFS, but also dabble in generic VFS layer (and other places)
- Maintain the cifs-utils package upstream, and in Fedora and RHEL

Overview

- Basic Kerberos Concepts
- CIFS and Kerberos 5
- Problems with current implementation
- Deployment scenarios and recommendations
- Future directions

Introduction to Kerberos

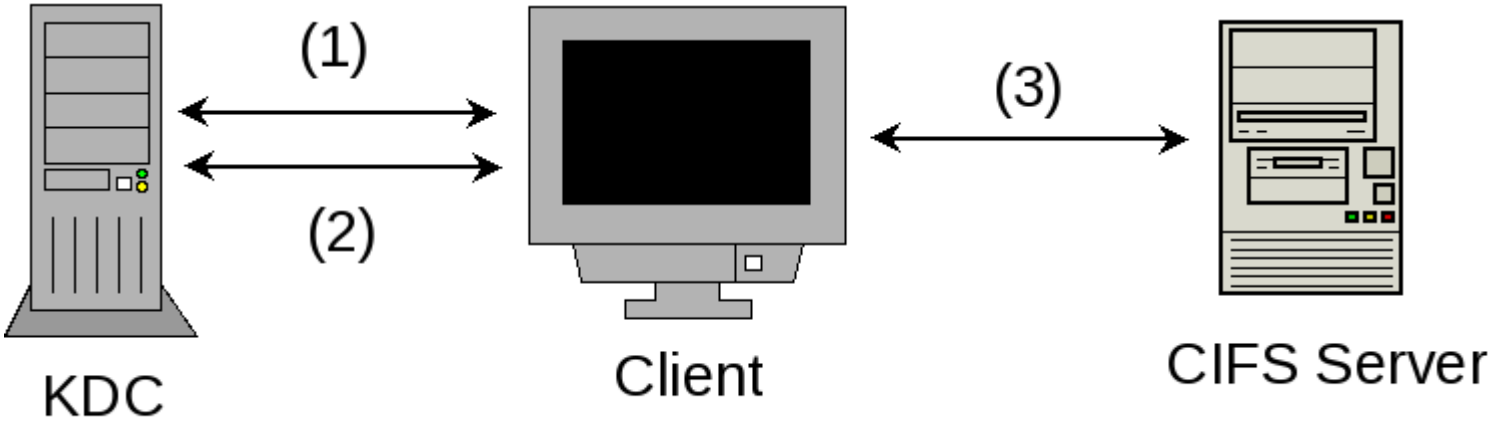
What is Kerberos?

- Secure authentication over insecure networks:
  - Verify identity without exposing passwords to network
  - Relies on a trusted 3rd party – the Key Distribution Center (KDC)
  - All entities (users and services) are considered “principals” to the KDC
  - Authentication is mutual

A Brief History of Kerberos

- Invented at MIT, first publication of v4 in the 1980's
- v5 published in 1993
- Most Unix-like OS's have had it for many years
- Microsoft adopted it as the basis of its authentication model with Windows 2000

Overview of Kerberos Authentication



1) Get Ticket Granting Ticket (TGT)
2) Use TGT to get service ticket
3) Use service ticket to establish server session

Krb5 Principal Format

primary/instance@REALM

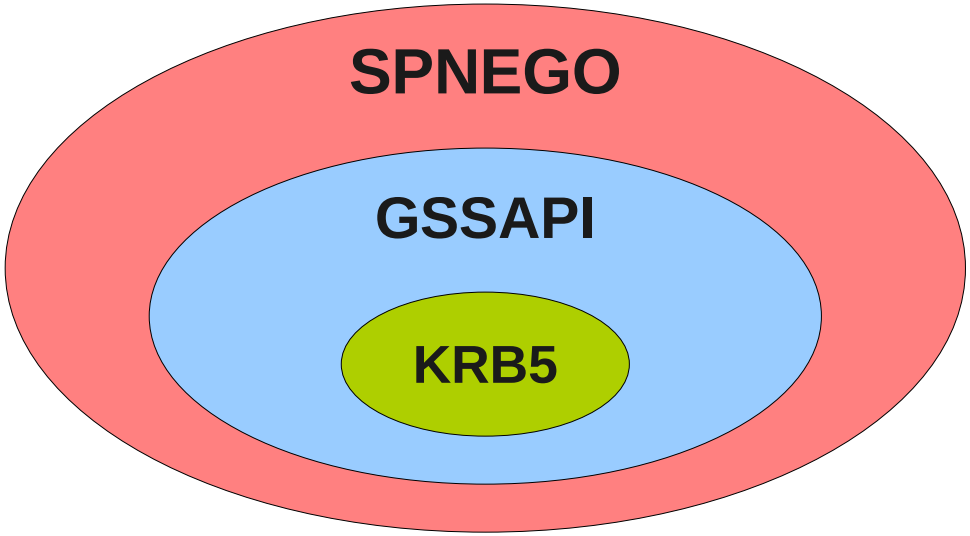
- **primary:** user or service name (e.g. “nfs”, “cifs”, or “host”)
- **instance:** optional qualifier. Usually FQDN for service principals. Sometimes “/admin” for user principals.
- **realm:** all principals are unique within a realm. Convention is to use DNS domain name in uppercase.

Examples of Principals

- User Principals:
  - `jlayton@EXAMPLE.COM`
  - `jlayton/admin@EXAMPLE.COM`

- Service Principals:
  - `host/server.example.com@EXAMPLE.COM`
  - `cifs/server.example.com@EXAMPLE.COM`

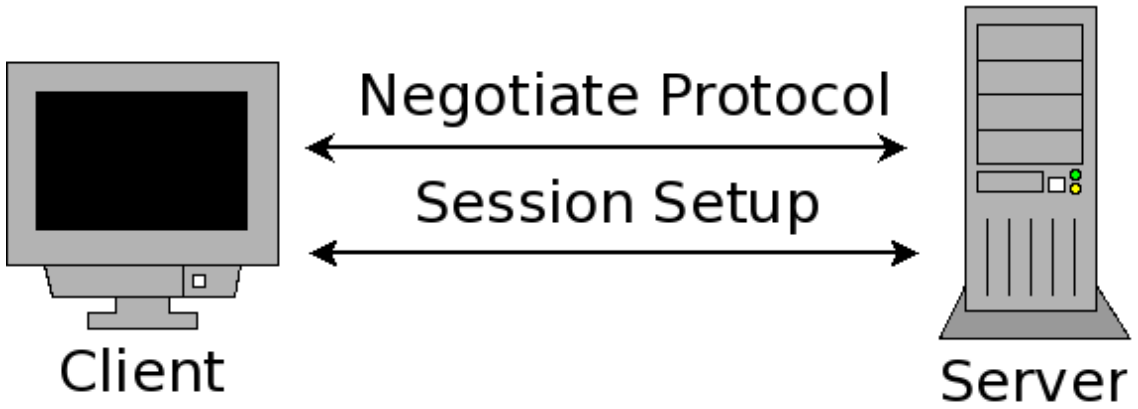
Authentication Layers



- **GSSAPI:** Generic Security Services Application Programming Interface. A standard plugin interface for authentication schemes.
- **SPNEGO:** Simple Protected GSSAPI Negotiation Mechanism. A way for client and server to agree on an authentication method to use.

CIFS and Kerberos 5

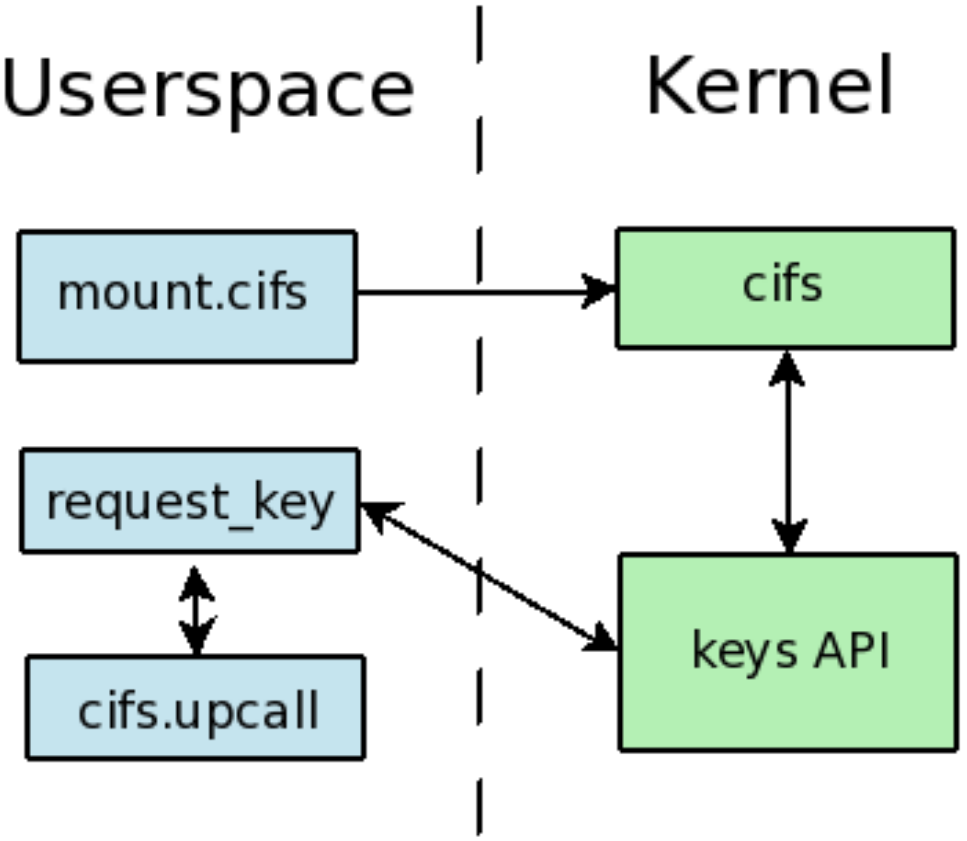
CIFS Authentication with KRB5



- Client sends NegProt req with extended security bit set
- Server replies with list of auth methods that it supports (via SPNEGO)
- Client sends Session Setup request with SPNEGO blob that contains KRB5 ticket wrapped in GSSAPI

CIFS+KRB5 Upcall Process

- mount.cifs reqests krb5 auth
- cifs calls into keys API for SPNEGO blob
- keys api calls out to /sbin/request_key
- request_key calls cifs.upcall which builds SPNEGO blob



Requirements for CIFS + krb5

- Linux kernel that supports SPNEGO upcalls
  - Support first went into mainline in 2.6.24
  - Also backported to RHEL5.3
- Client Configured for krb5 **(`/etc/krb5.conf`)**
- **`/sbin/request-key`**
  - part of the “keyutils” package
- **`/usr/sbin/cifs.upcall`**
  - RHEL5 & Fedora (pre F13): samba-client package
  - RHEL6 & F13+: cifs-utils package

Basic krb5.conf configuration

- Easiest to use **system-config-authentication**
- Basic config follows:

```
[realms]
 EXAMPLE.COM = {
   kdc = ad.example.com:88
   admin_server = ad.example.com:749
 }

[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM
```


Configuring /etc/request-key.conf

Tells request-key program what program it should run and how. Note that cifs also uses this to handle DNS resolution for DFS (see cifs.upcall(8)):

/etc/request-key.conf:

```
#OPERATION  TYPE          D C PROGRAM ARG1 ARG2...
#=========  ===========   = = ========================
create      cifs.spnego   * * /usr/sbin/cifs.upcall %k
create      dns_resolver  * * /usr/sbin/cifs.upcall %k
```


Simple Mount with krb5

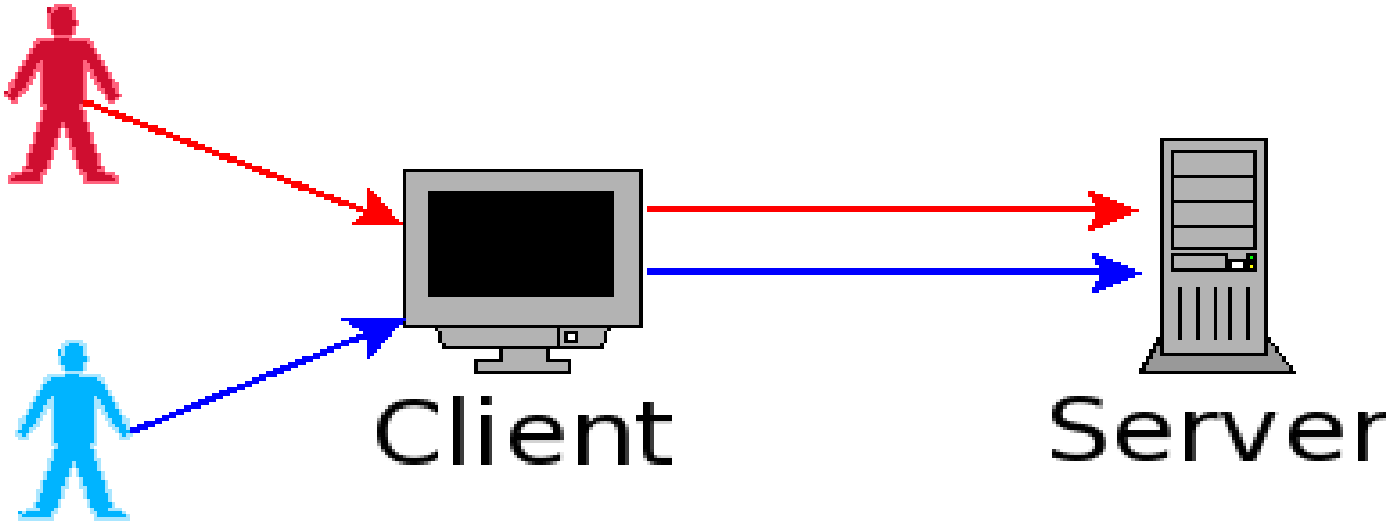
- Get a krb5 ticket for the user as whom you'll be authenticating.
- Then mount the share with the sec=krb5 option
  - **Hostname in UNC much match service principal!**

```
# kinit testuser@EXAMPLE.COM
Password for testuser@EXAMPLE.COM:

# mount -t cifs -o sec=krb5 \
//server.example.com/export /mnt/cifs
```

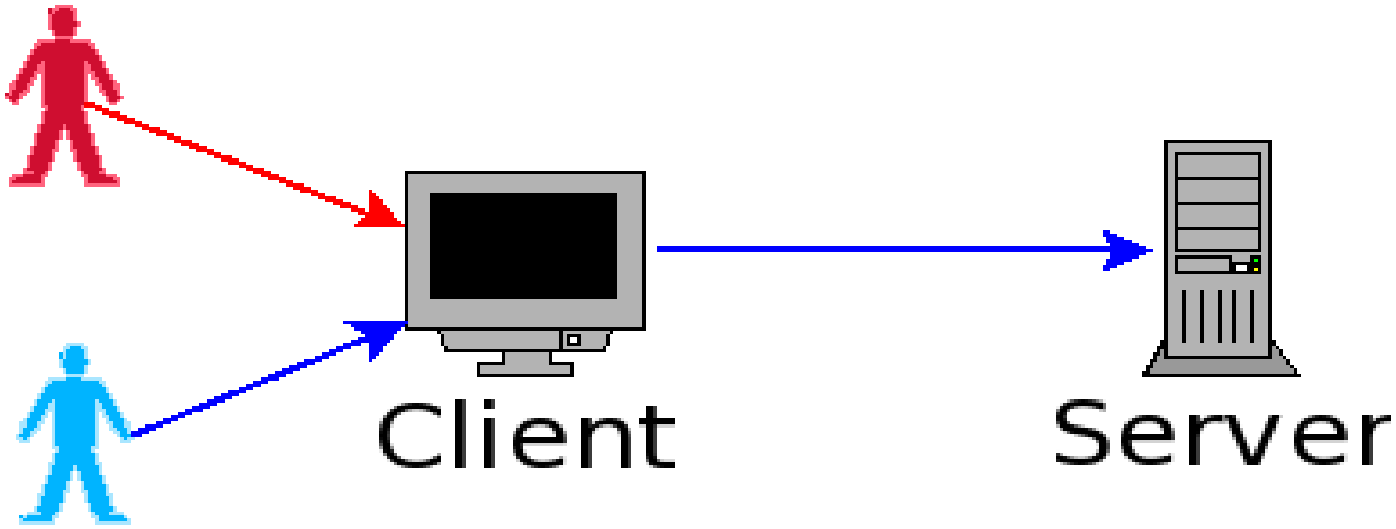

Problems with Current Implementation

The NFS Mount Model



- “Traditional” network filesystem for unix is NFS
- User creds are sent to the server in each call
- No one “owns” the connection to the server

The CIFS Mount Model Today



- Only one CIFS session per mount
- One set of credentials per CIFS session
- Other users who use the mount are using the same credentials as the user who “owns” the mount

POSIX Extensions

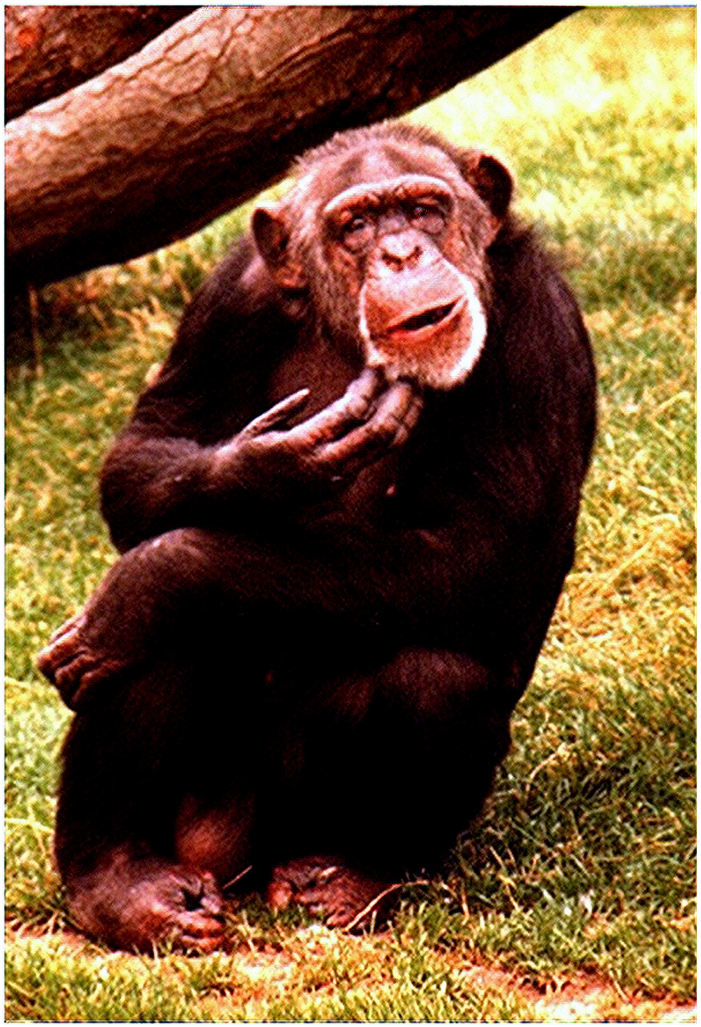
- CIFS enables POSIX extensions by default
- Problems with modes and ownership:
  - uid/gid may not match on client and server
  - uid=/gid= mount options override ownership but not mode. The result is permissions that have no basis in reality.
  - all ops on the server are done using mount creds, but VFS enforces these permissions locally. The client's VFS may limit a user from doing ops that the server would allow

File Creation and Permissions

- First test:
  - Mount cifs share with one user's credentials and with unix extensions enabled
  - Share is world-writable
  - “touch” file in share as another user

```
$ touch testfile
touch: cannot touch `testfile': Permission denied
```


What happened?



- File was created on server using mount credentials
- CIFS attempts to enforce permissions on client
- That can't fix ownership
- File is created but later operations fail!

Permissions Enforcement

- Second test:
  - Mount share with one user's credentials and without unix permissions
  - As another user, access a file that should be accessible by only that user.
- You can't enforce permissions correctly if you don't know what they should be
- Even if you do, checking on the client is racy – permissions can change after you check them but before they are enforced

So there are problems...

- Summary:
  - POSIX extensions aren't terribly useful as implemented by CIFS VFS
  - “shared” mounts doesn't work as expected
- Recommendations:
  - Limit permissions to the user who owns the creds
  - Maybe disable unix extensions altogether?

Deploying CIFS with Kerberos 5

User mounts in /etc/fstab

- mount(8) allows unprivileged users to mount filesystems if:
  - /bin/mount and mount helper are setuid root (not recommended with the version that ships in RHEL4/5)
  - the user owns the mountpoint
  - mount is in /etc/fstab with “user” option (distinct from user= option that CIFS uses)

User mounts in /etc/fstab

/etc/fstab:

```
//server/share /home/testuser/cifs \
sec=krb5,user,nounix,file_mode=0700, \
dir_mode=0700,noauto 0 0
```

Then, as unprivileged user:

```
testuser@client$ kinit
Password for testuser@EXAMPLE.COM:
testuser@client$ mount /home/testuser/cifs
```


Using autofs

- Another possibility is to use autofs:

```
jlayton \
-fstype=cifs,sec=krb5,uid=$UID,gid=$GID \
\\\\server.example.com\\jlayton
```

- How do we ensure that the “right” user gets the mount?

Using pam_mount

- Linux PAM module that can mount filesystems on login
  - PAM == Pluggable Authentication Modules
- Users can configure their own set of mounts (within limits set by admin)
- Most useful when combined with pam_krb5
- see pam_mount(8) and pam_mount.conf(5)

What about MultiuserMount?

- Can be enabled via:

  `/proc/fs/cifs/MultiuserMount`

- When there are multiple sessions to the same server, use one that's owned by my UID
- Problems with this approach:
  - Requires a separate mount for each user
  - Users w/o a mount use “default” creds
  - Permissions and file ownership, writeback...

MultiuserMount Redux

- Patchset in progress to do multiuser mounts the “right way” (renamed “multisession mounts” to avoid confusion)
- Have multiple sessions per mount
- Sessions are established on an as-needed basis
- Server handles permissions
- **Goal:** as easy as Kerberized NFS (or easier)

Questions?

FOLLOW US ON TWITTER

www.twitter.com/redhatsummit

TWEET ABOUT IT

#summitjbw

READ THE BLOG

http://summitblog.redhat.com/