These are the ingredients to making AD Kerberos work with NFSv4. Note that puppet knows most of these things; I am documenting this since there are a lot of moving pieces and although they can mostly be discerned by reading the code in puppet sometimes the rationale behind certain elements of the configuration is not obvious.

You will want to read and understand Using AD Kerberos for authentication before proceeding with NFSv4. Please especially note that you must create principals (e.g. "join the domain") with the script mentioned in Using AD Kerberos for authentication#CreateKerberosprincipals.

- Prevent user tickets from expiring too quickly
- Enable winbind for group mapping
- Configure rpcidmapd
- Enable secure NFS
- Dealing with NFSv4 ACLs
- Using secure NFSv4

### Prevent user tickets from expiring too quickly

It is critical that users have kerberos tickets available, or they will be unable to access resources. This can be a problem when using AD krb; specifically, the default policy restricts ticket lifetimes to 10 hours. Without a current ticket, a user’s processes will all essentially become ‘nobody’ and will lose access to any NFSv4 exports that are mounted.

Luckily, users can renew their tickets for a maximum lifetime of 7 days. We do not want to require manual intervention for this, so we should automatically renew the user’s ticket when possible.

The bash::activedirectory module includes a script (/usr/local/sbin/renew_krb_tkts) which is used for this purpose. The script will automatically update renewable tickets from a cron job that runs every hour (the cron job itself comes from auth::ad).

This still has limitations, though; the renew lifetime of an AD TGT is 7 days, after which the script will no longer work. The user needs to get a new ticket granting ticket before those 7 days expire (and so requires a ‘kinit’ with password).

The script mentioned above will inform a user via ‘write’ when they have only 24 hours remaining on their ticket, and provide instructions for how to renew a ticket. Once the user has only three hours left, the script will send an email with instructions.

There are multiple ways to refresh a ticket-granting-ticket. Logging in via SSH with KRB credentials will do it, which means that in a lot of cases (where users would be logging into a system at least once per week) no special intervention should be required. However, if they are leaving jobs to run a very long time, or if they are logged in with a long running SSH session (e.g., using 'screen' or 'tmux' on a remote system), they may not SSH in frequently. In such a case a user can log out and back in, but they can also simply run ‘k5start -H 720 or ‘kinit' to be prompted for a password to update their ticket.

Documentation in the script mentioned above implies that running ‘k5start -H 360’ as your PROMPT_COMMAND is a good idea - this will execute after every command in the shell, and it will prompt for a password and retrieve a new ticket-granting ticket when their lifetimes are less than 6 hours (a situation that should only happen when a ticket is nearing the end of its renewable life span). I am undecided on this technique. This currently results in the user being prompted immediately to re-enter their krb password upon logging in with PAM, which is suboptimal. If I can eliminate that behavior this may be a good option.

There is another complication. For some reason, if you use GSSAPI to authenticate (i.e., ssh -K with KRB tickets from another system), you get a NEW ticket cache assigned to krb5ccname with those credentials for every login. This could become extremely complicated for somebody logging in multiple times via GSSAPI, as they would have different tickets for each login, and it would be very hard to renew tickets from a pre-established session. As of now, I think the only real workaround is to configure ssd to use a consistent CCACHE location, and to have users execute a login script that will overwrite the normal ssd cache with one forwarded by SSH (and update the krb5ccname variable to reflect that). Not really a fan, though...

### Enable winbind for group mapping

This is optional, and probably most useful if you are managing complex permission structures that closely map to AD organizational units (local groups are a simpler option for one-off scenarios).

This group mapping must come from samba which is documented more completely on the Samba CIFS server using AD page. If you enable the puppet 'samba::adclient' you'll get AD group mappings, but beware that actual samba servers will require additional configuration in order to support proper AD -> UNIX mappings above and beyond this. Due to relying on winbind, there are some substantial limitations to this approach which require some tricky workarounds.

### Configure rpcidmapd

RPCIDMAPD is the "Rosetta stone" which tells rpcbind which domain and KRB realms users and service principals should belong to. The daemon must be enabled for NFSv4 to really work at all. The correct configuration is available from nfs::v4ad. In short, /etc/idmapd.conf must contain:

1. a "Domain" entry, matching the domain. I use egr.duke.edu for this. Most of our machines technically do not require this, since they are generally named <hostname>.egr.duke.edu and rpcidmapd should use egr.duke.edu by default, but it never hurts to be explicit...
2. A Local-Realms setting which matches the AD KRB realm "WIN.DUKE.EDU." This instructs idmapd.conf to consider these KRB credentials as valid for resources on the machine.
Enable secure NFS

Again, this comes from nfs::v4ad. The pieces are:

1. Set "SECURE_NFS" to "yes" in /etc/sysconfig/nfs (RHEL init scripts check for this and disable krb if it is absent when starting)
2. The "nfs" meta-service must be restarted after this change, which will then start the gssapi service
3. For some reason (bug?) rpcgssd does not start from the nfs meta-service and should be enabled explicitly

RHEL > 7.1 will now automatically enable the appropriate services if you have a keytab installed (but beware, this happens at system boot only)

Dealing with NFSv4 ACLs

This is technically optional, but it's generally desirable (especially if you intend to use samba for accessing any of these resources). Keep in mind that ext4 does not support NFSv4 ACLs directly; it is limited to POSIX ACLs, which lack the semantics to express all elements of NFSv4 ACLs. So when you enable ACLs in Linux, the kernel will translate the NFSv4-style ACLs into POSIX equivalents as best as it is able. This is further complicated by the presence of Samba and NTFS ACLs, which are also different and are also translated to POSIX ACLs.

The long and short of it is that, in most cases, editing ACLs on a samba share with the NT ACL editor dialogue will probably map to the appropriate ACLs on the filesystem, which will probably map to NFSv4 ACLs - but not always exactly. So if you encounter strange issues, this is likely the cause.

On the server:

1. Ensure that any filesystems you will be exporting are mounted with the "acl" option in fstab. This enables POSIX ACLs.
2. For any filesystem exported with CIFS, you should also enable the "user_xattr" option in fstab. This is not required for NFSv4 but is a good idea anyway.
3. Ensure any NFS exports include the "acl" option
4. Inspect/edit POSIX ACLs as usual (with getfacl/setfacl)

On the client:

1. Mount the export with the "acl" option
2. Install the package "nfs4-acl-tools," which provides nfs4_getfacl and nfs4_setfacl (which are the nfs4 acl equivalents of the POSIX get/setfacl utilities) and go to town

Using secure NFSv4

There are some considerations when actually using NFSv4 + KRB. The magic strings are sec=(krb5|krb5i|krb5p) in /etc/exports or with 'mount' on the client. You may specify any or all of these options for the export, and you must select only one at mount time. Beware the following ramifications:

1. Any of the 'krb5' sec options imply that all users of the export must have valid kerberos credentials - if they do not, they will not be able to access the export at all. This means any local or machine accounts (e.g., apache) on the remote system are unable to access files without kerberos credentials on exports like this. In the case that you need a daemon-like user to have access to such nfs4 exports, you must create a keytab for that user and create krb credentials for that user in AD.
2. krb5 has a minimal performance penalty, as it simply authenticates the user. krb5i provides integrity in that it checksums traffic, preventing data corruption, but still allowing attackers to sniff data on the wire. krb5p provides privacy and encrypts the entire data stream. Which should be used? On modern systems, the overhead of krb5p is probably acceptable, but on slower machines it can be a substantial bottleneck.
3. You can't 'mount' a sec=krb5 export at all unless you have the nfs:<machinename> principal mentioned above, even if your actual users would otherwise be able to access the resource with their own tickets.
4. The net result of all of these conditions is that it should generally be safe to avoid host-based ACLs and export krb5-protected NFSv4 to *, allowing only on filesystem permissions to restrict access.
5. It is possible to allow for non-kerberized NFSv4 as well, if you add sec=sys to an export. But note that if you do this, you need to fall back on host-based export restrictions and/or the firewall, since sec=sys on an export basically means nfsv3 style security.