BeagleBone as Kerberos KDC for NFSv4 Authentication with the Synology Diskstation

Gregory Raven

August 9, 2017

Abstract

The BeagleBone series of single-board computers can be used as a Kerberos "Key Distribution Center" server for NFSv4 authentication. This document describes the configuration of the BeagleBone, a Synology DiskStation "Network Attached Storage" appliance, and a client workstation running Ubuntu Mate version 16.04.

Contents

1 BeagleBone Kerberos: The Viscious Beaglebone Guard Dog **3**
1.1 Outline of the Kerberos Configuration Process . . . 4

2 Preliminaries **5**
2.1 The "Realm" . . . 5

3 Kerberos and DNS versus /etc/hosts file **5**
3.1 DNS and Host Name Resolution with Kerberos . . . 5
3.2 Example Hostname and Host Files . . . 6
3.3 Example krb5.conf File . . . 6
3.4 Requirement for Time Synchronization . . . 6
3.5 Home Router and Network Configuration . . . 7

4 Configuration of the Beaglebone KDC **8**

5 Beagle Bone Console Image **8**
5.1 Configuration of Beaglebone as KDC . . . 9
5.2 Service and Client Principals for NFS . . . 11
5.3 Service Principals . . . 11
5.4 The Very Useful ktutil Command . . . 12
5.5 Authorizing a User: Client Principal . . . 12

6 Configuration Done on the Synology DiskStation **14**
6.1 Enable Kerberos Authentication on a Shared Folder . . . 14
6.2 Set up the NFSv4 File Service . . . 14
6.3 Configure the Synology Shares for Kerberized Client Access . . . 15
6.4 Mysteries of the Synology DiskStation . . . 16

7 Configuration Done on the Ubuntu Mate Client **17**
7.1 Install NFS Packages . . . 17
7.2 Install Kerberos Client Package . . . 17
7.3 Update the Client's Host File . . . 17
7.4 NFS and Kerberos: RPC.GSSD . . . 17
7.5 Set Mapping Domain in file /etc/idmapd.conf . . . 19
7.5.1 ID Mapping Configuration . . . 19

8 Mounting the NAS to the Ubuntu Client **20**
8.1 Mount the Server with Kerberos Authentication . . . 20
8.2 The Client Ticket . . . 20

9 Conclusion **21**

10 Web Sites with Good Information on NFS and Kerberos **22**
10.1 Links to Information on Kerberos . . . 22
10.2 Links to Information on NFS . . . 22

1 BeagleBone Kerberos: The Viscious Beaglebone Guard Dog

This is a project which uses a BeagleBone single-board computer to implement a Kerberized NFS interface to a "Network Attached Storage" device in a home network.

"Kerberos" is a network authentication protocol which originated at MIT in the 1980s:

https://en.wikipedia.org/wiki/Kerberos_(protocol)

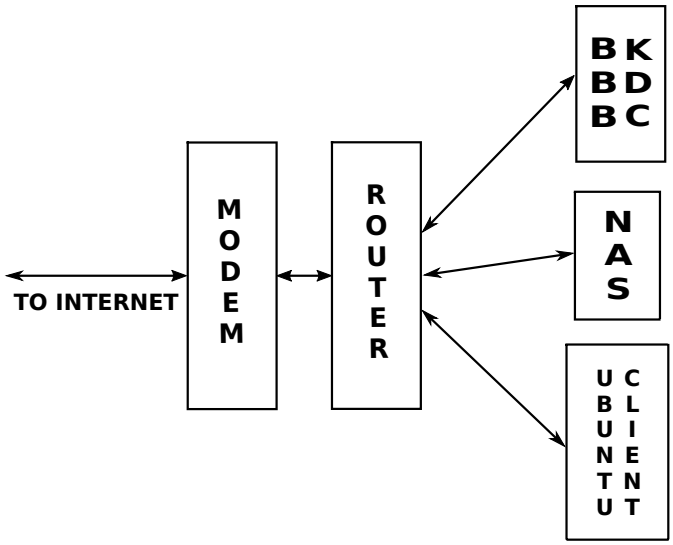

Figure 1: A Home Network with "Network Attached Storage" file share mounted to Ubuntu Client. A Beagle Bone Black acts as a Kerberos "Key Distribution Center".

This project adds Kerberos authentication to NFSv4 which is a "Network File System" commonly used to mount file servers to client workstations.

One of the goals was to get correct user name translation from client to mounted server files. In a traditional NFS system, the user ids (uid) and group ids (gid) must be perfectly synchronized in order for the mounted file system to work transparently. In practice this is a problem, as a client machine may assign a uid of 1000 to user name "smith", while the server machine could assign a uid of 500 to the same user name "smith". The user smith will probably see "nobody" and "nogroup" on his mounted files, even though the server sees them as belonging to user smith. Translation works by mapping names rather than id numbers. Synchronization of numeric ids is no longer required, and the end result is that the client sees his files as though they reside on his machine with correct user and group names.

It appears that Kerberos authentication is required for the id mapping feature of NFSv4 to function properly. It is difficult to confirm this, as NFSv4 is notoriously difficult to configure, and there is a great deal of confusion to be found when searching the internet on this topic. For sure, the id mapping works well when Kerberos security is in force. See note 5 under the sub-topic "NFSv4 Client" on this page:

https://help.ubuntu.com/community/NFSv4Howto

This BeagleBone project is somewhat unusual in that no additional hardware beyond the board itself is involved. The goal is not a traditional "embedded device", however, the BeagleBone series of devices seems well suited to this particular application. There are no detailed schematics or diagrams required, thus this project a compilation of notes which were scribbled down in the process of getting the system to work.

The machines on the home network are assumed to run the GNU/Linux operating system. No attempt was made to integrate Windows or other machines into the network, but it should be possible to do so.

This project requires minimal time to edit and configure files compared to the usual embedded project. However, documentation available on this subject of Kerberized NFS is sparse and not easily interpreted and applied to a particular situation. Be aware that minor differences in the machines or operating systems may not result in a working system. Many hours of research and troubleshooting may be required to achieve a satisfactory result. Be prepared to spend significant time on this project!

1.1 Outline of the Kerberos Configuration Process

This is an outline of the steps required to get Kerberos authentication working with NFSv4 on a home network. The rest of this document will cover each of these steps in detail. A list of links to reference material is included in the appendix.

1. Prepare the client. Install Kerberos and edit hostname and hosts file. Edit krb5.conf file.
2. Flash the Beaglebone with the "console" image. Install vim, git, and NTP. Update and run set-up scripts. Remove git.
3. Edit Beaglebone hostname and hosts file.
4. Install Kerberos on Beaglebone. Create master key. Create user principal.
5. Edit krb5.conf file on Beaglebone.
6. Create NFS keytab files for server and client. Copy keytabs files to client.
7. Configure NAS for Kerberos authentication using GUI. Install keytab file.
8. Mount the server to the client with Kerberos authentication option.
9. Authorize the user from the client machine.

2 Preliminaries

2.1 The "Realm"

Kerberos is built around the concept of a "Realm". There can be more than one Realm, but for the purposes of this project in a home network the discussion will be limited to one.

The client is a Linux workstation which is used for normal computing activities. The KDC is a dedicated machine which is responsible for issuing Kerberos "tickets". The "Network Attached Server" (NAS) is a server with a large redundant file system which is to be mounted via NFSv4 to the client workstation.

The boundaries of the Realm in this case are the client, KDC, and NAS. A router is connected to all three of these, however, the router has no role in Kerberos security and it will be assumed it silently does its job.

The example Realm for this project:

BADBEAGLE.COM

The Realm name is suggested to be the upper-case version of the domain assigned to the network. In the case of a home network, behind a router (and hopefully a firewall), the Realm name can be anything similar to typical domain name. The important point is to use the Realm name consistently in configuration of all machines involved in the system.

3 Kerberos and DNS versus /etc/hosts file

3.1 DNS and Host Name Resolution with Kerberos

In a lot of the Kerberos documentation, there are many references to and heavy dependence on a "Domain Name System". This is internet terminology for a server dedicated to translating textual "domain names" to IP addresses, and vice versa. There are references to "forward and reverse" translation.

This implies that DNS is a fundamental requirement, however, this is not the case and a simple configuration can achieve the forward and reverse translation well enough to allow the Kerberized NFS to function on a small home network.

In the case of a no-DNS network, the machine's "/etc/hostname" and "/etc/hosts" files are edited to mimic DNS forward and reverse translation.

From "The MIT Kerberos Administrators How-to Guide" by Jean-Yves Migeon:

> Kerberos relies heavily on DNS, either for contacting KDC, or resolving hostname for authentication (for service use – see later on). You do not necessarily need a DNS server for Kerberos to work. The /etc/hosts file is sufficient, but somewhat limited when network's size grows. Kerberos only needs proper direct- and reverse- resolution of hostname, which should point to their respective FQDN (Fully Qualified Domain Name). Installing and configuring a DNS does not belong to such an article. For our example, we will use a very simple /etc/hosts file:

```
/etc/hosts
127.0.0.1
localhost
192.168.1.101 kdc.foobar.com kdc
192.168.1.1 kdc-client.foobar.com kdc-client
...
```

Using a custom /etc/hosts file for each client and the NAS was used as described above. A DNS server was not required for this small-scale network.

3.2 Example Hostname and Host Files

This /etc/hostname file is trivial and contains only the name of the host. For example, a client Linux box /etc/hostname:

```
ubuntu1
```

That is the entire contents of the hostname file!

Next, an example /etc/hosts file:

```
127.0.0.1 localhost
192.168.1.5 ubuntu1
192.168.1.10 kdc
192.168.1.7 diskstation
```

This file establishes a local forward and reverse mapping between the hostname and the IP address. This assumes the IP addresses are "static", and this is easily enforced with settings available in the home network router (note that this is not really "static" assignment, but DHCP assignment of a particular IP address to a particular device).

3.3 Example krb5.conf File

Installation of the Kerberos packages will create an example "krb5.conf" file in the /etc directory. This file includes Kerberos version 4 related set-up and other items which are not required for a home network.

The following example is the bare minimum required for a single client and the KDC. Note the special [logging] section which is used on the KDC only. The Synology NAS server creates its file automatically from inputs into the GUIs.

```
[libdefaults]
default_realm = BADBEAGLE.COM

kdc_timesync = 1
ccache_type = 4
forwardable = false
proxiable = false

[realms]
BADBEAGLE.COM = {
kdc = kdc
admin_server = kdc
}

#  The following is optional for KDC only.
[logging]
kdc = FILE:/home/debian/kerberos/kdc.log
admin_server = FILE:/home/debian/kerberos/kadmin.log
```

Please refer to this web page for explanation of the parameters in this file:

https://web.mit.edu/kerberos/krb5-latest/doc/admin/conf_files/krb5_conf.html

Note that in the example above, the values for "forwardable" and "proxiable" have been changed to false versus the true value provided by the example krb5.conf file inserted during the installation process.

3.4 Requirement for Time Synchronization

Kerberos expects the machines to maintain time synchronization. This is another aspect of security built into the Kerberos software.

The Beaglebones do not have their own “Real Time Clock” with battery back-up. The Beaglebones do have a hardware clock, however, it will not maintain absolute accuracy between boots.

The solution to this is to enable “Network Time Protocol” (NTP). The NAS and all clients connected to the home network should also be using NTP. This will resolve this concern.

It is important to understand that the “System Clock” is what the Beagle Bone actually uses for time keeping. The System Clock is synchronized at boot to either the Network Time, or to the non-backed-up hardware clock. This time will be grossly inaccurate if the Beagle Bone does not have a live internet connection with NTP active! Thus, it is important to boot the Beaglebone-KDC only with a live internet connection!

If the NAS fails to mount due to denied access, the Beaglebone-KDC clock is one of the first things to check. Simply login to a terminal and use the “date” command. Reboot as required.

3.5 Home Router and Network Configuration

The home network router does not have to do anything special to allow implementation of NFSv4 with Kerberos.

The only requirement is for it to enforce “static” IP addresses.

The test system was developed with a Netgear brand router. This router has a feature called “Address Reservation”. This works even when DHCP is active, and configuration of the clients for static IP addresses is not require.

In the Netgear GUI, Address Reservation is found in:

```
Advanced -> Setup -> LAN Setup
```

This works by associating the unique MAC address of a client with a manually chosen IP address. Once the fixed addresses are configured (being careful not to create conflicts), the IP addresses assigned at power-up to the various clients attached to the local network will always be the same.

4 Configuration of the Beaglebone KDC

Since this is the heart of the Kerberized NFS, the discussion of configuration will start with the "Key Distribution Center" (KDC).

It is recommended to use the "console" distribution of Debian for the Beaglebone. The sole purpose of the KDC should be to complete Kerberos transactions between the clients and services. It is a key element of security, and thus it should be well-secured from intrusion. Minimizing the "attack interface" is a good practice, and thus only the bare minimum software should be installed on the KDC.

It is important to be aware that Kerberized NFS will break down if the KDC goes off-line. It is possible to have "slave" KDCs which take over in the case of a failure of the "master". Reliability of the KDC is important, and a strategy to handle KDC failure should be in place.

In the case of a home network, multiple master/slave KDCs is probably overkill. However, a spare Beaglebone fully provisioned as a drop-in replacement is not unreasonable. The spare should be kept in a secure location.

5 Beagle Bone Console Image

The "console" image is a bare-bones Debian distribution for the Beaglebones. This minimalist Debian distribution is the correct choice for the Beaglebone-KDC. This applies a basic security principle of minimizing the "attack surface". There are a few things which must be installed for practical purposes.

The image is found on this web page:

http://elinux.org/Beagleboard:BeagleBoneBlack_Debian

Follow the instructions on the above page to download and flash the image to a micro-sd card.

To flash the above images to a Micro-sd card:

```
xzcat bone-debian-8.8-console-armhf-2017-07-01-1gb.img.xz | sudo dd of=/dev/sdX
```

Where /dev/sdX is the target Micro-sd card. It is imperative to get the target path correct, as the file system there will be overwritten by the dd command.

An alternative to using xzcat expands the file first, and then writes to the Micro-sd card:

```
unxz bone-debian-8.8-console-armhf-2017-07-01-1gb.img.xz
```

This will create:

```
bone-debian-8.8-console-armhf-2017-07-01-1gb.img
```

Now use the dd command to flash the image:

```
sudo dd if=bone-debian-8.8-console-armhf-2017-07-01-1gb.img of=/dev/sdX bs=8M
```

Ubuntu 16.04 has a utility which will help determine the correct path to use in the above commands:

```
Applications -> Accessories -> Disks
```

For example, when a USB adapter is plugged-in, the device shown by the Disks utility is:

```
/dev/sdb
```

Note that the default of the Debian console image is to boot from the Micro-sd. This should only be done temporarily as the configuration process is completed and de-bugged. After the system is working, the image should be flashed to the Beaglebone eMMC and the Micro-sd removed. The eMMC will provide more reliable service than the Micro-sd. Follow the instructions on this page to flash the eMMC:

http://beagleboard.org/latest-images

Upon successful start-up of BBG with console image there are numerous steps to configure the Beaglebone as a KDC. Follow these steps carefully.

5.1 Configuration of Beaglebone as KDC

1. `sudo apt-get update`
2. Copy ssh public key from Ubuntu Mate workstation to Beaglebone:

   ```
   ssh-copy-id -i id_rsa.pub debian@192.168.1.10
   ```

3. Reserve IP address using router reservation.
4. Install Network Time Protocol:

   ```
   sudo apt-get install ntp
   ```

5. Git is temporarily installed in order to update and run some set-up scripts:

   ```
   sudo apt-get install git
   ```

6. Update the Beagle Bone scripts:

   ```
   cd /opt/scripts
   git pull
   cd tools
   sudo ./grow_partition.sh
   ```

7. Uninstall git:

   ```
   sudo apt-get remove git
   ```

8. `sudo apt-get install vim`
9. Use vim to change hostname beaglebone to kdc. This is the path to the file:

   ```
   /etc/hostname
   ```

   After editing, the file should contain a single line:

   ```
   kdc
   ```

10. Use vim to update hosts file. Update the kdc IP, and add the NAS and clients IP addresses. This is path to the file:

    ```
    /etc/hosts
    ```

    An example file looks like this:

    ```
    127.0.0.1       localhost
    127.0.1.1       kdc.localdomain kdc
    192.168.1.5 ubuntu1
    192.168.1.10 kdc
    192.168.1.6 ds216
    ```

    It is recommended to reboot the Beagle Bone before continuing. Hold the power button down for a few seconds and the Beaglebone will shutdown. A quick press of the power button will start boot.

11. Install kerberos:

    ```
    sudo apt-get install krb5-kdc krb5-admin-server
    ```

    A dialog will start which will ask for:

    - Default realm (example: BADBEAGLE.COM)
    - Hostnames of Kerberos servers (example: kdc).

- Administrative server hostname (example: kdc).

12. Run the command:

    ```
    sudo krb5_newrealm
    ```

    A KDC master key will be requested. Enter and confirm. This will complete setting up the realm. Note that this process may take a while as the software uses “entropy” to randomize.

13. Modify the file:

    ```
    /etc/krb5.conf
    ```

    The default file created by the above process will have lots of example code. Most of the lines in the file can be deleted. Here is an example after clean-up:

    ```
    [libdefaults]
        default_realm = BADBEAGLE.COM

    kdc_timesync = 1
    ccache_type = 4
    forwardable = false
    proxiable = false

    [realms]
        BADBEAGLE.COM = {
        kdc = kdc
        admin_server = kdc
        }

    [domain_realm]
    kdcspare = BADBEAGLE.COM
    ubuntu1 = BADBEAGLE.COM
    diskstation = BADBEAGLE.COM
    ```

The next section will cover the creation of the “principals” for Kerberized NFS.

5.2 Service and Client Principals for NFS

One of the most important terms in the Kerberos system is the concept of ”principal”. A “principal” is one of many users, servers, or services involved in a Kerberos Realm. The principals are the actors in the system who will request access and require authentication from one to another via the “ticket” granting authority of a third party, which in this case is the Beaglebone-KDC.

5.3 Service Principals

Kerberized NFS is unusual in that two “service principals” are required.

There is a service principal for the client and a service principal for the server:

```
nfs/diskstation@BADBEAGLE.COM
nfs/ubuntu1@BADBEAGLE.COM
```

The above two principals are used to form a “Machine principal” during the mounting of the file system.

The above two service principals are created on the KDC. This is most easily done while remotely logged into the KDC via ssh.

This is a three step process for each of the required keytabs:

1. Add the new principal to the KDC’s Kerberos database.
2. Extract a keytab file.
3. Copy the respective keytab files to the client and server machines.

The following show the series of commands entered at the KDC. In this example, “diskstation” is all lower-case version of the server host name, and “ubuntu1” is all lower-case version of the client host name.

First, make a directory “kerberos” and cd into it.

The kadmin.local command will open a kadmin: prompt as shown.

```
sudo kadmin.local
Authenticating as principal root/admin@BADBEAGLE.COM with password.
kadmin.local:
```

Add the server principal:

```
kadmin: addprinc -randkey nfs/diskstation
```

Add the client principal:

```
kadmin: addprinc -randkey nfs/ubuntu1
```

The above commands will not ask for a password.

Keytab files must be extracted for both the server and client and written to the current directory:

```
kadmin: ktadd -k nfs-server.keytab nfs/diskstation
kadmin: ktadd -k krb5.keytab nfs/ubuntu1
kadmin: quit
```

Change the keytab files owners to debian so that they can be copied to the client.

```
debian@kdc:~/kerberos$ sudo chown debian krb5.keytab nfs-server.keytab
```

Use scp from the client to securely copy the keytab files from the KDC to the client machine.

```
scp debian@192.168.1.10:/home/debian/kerberos/nfs-server.keytab .
scp debian@192.168.1.10:/home/debian/kerberos/krb5.keytab .
```

where it is assumed that the kadmin tool was started in the /home/debian/kerberos directory.

The server keytab is added to the Synology NAS through the web interface. That process is described in another chapter.

The client keytab is copied to /etc/krb5.keytab on the client machine. Make sure the file permissions are changed to allow root read access only.

The local copies of the keytabs can be deleted after they are moved into the locations described above.

5.4 The Very Useful ktutil Command

The ktutil command is another Kerberos shell prompt which is very useful for examining keytab files. It can also be used to merge keytabs if required.

In the example below, a keytab file is read using the rkt command (read keytab). The keytab principal is listed using the list command.

If a second keytab file is read, the two keytabs can be combined into a single file using the write_kt command. This particular feature is not required for this project and is mentioned for reference only. However, the ability to read and list principals in a keytab file is very useful for debugging purposes.

```
root@kdc:/home/debian/kerberos# ktutil
ktutil:  rkt client.keytab
ktutil:  list
slot KVNO Principal
---- ---- ---------------------------------------------------------------------
   1    2             smith/ubuntu1@BADBEAGLE.COM
   2    2             smith/ubuntu1@BADBEAGLE.COM
   3    2             smith/ubuntu1@BADBEAGLE.COM
   4    2             smith/ubuntu1@BADBEAGLE.COM
ktutil:  rkt nfs-client.keytab
ktutil:  list
slot KVNO Principal
---- ---- ---------------------------------------------------------------------
   1    2             smith/ubuntu1@BADBEAGLE.COM
   2    2             smith/ubuntu1@BADBEAGLE.COM
   3    2             smith/ubuntu1@BADBEAGLE.COM
   4    2             smith/ubuntu1@BADBEAGLE.COM
   5    2               nfs/ubuntu1@BADBEAGLE.COM
   6    2               nfs/ubuntu1@BADBEAGLE.COM
   7    2               nfs/ubuntu1@BADBEAGLE.COM
   8    2               nfs/ubuntu1@BADBEAGLE.COM
ktutil:  write_kt krb5.keytab
ktutil:  quit
```

5.5 Authorizing a User: Client Principal

A non-privileged user must have a “principal” registered with the KDC. The easiest way to accomplish this is to ssh into the KDC from the Ubuntu client machine, and then use this series of commands:

```
sudo kadmin.local
Authenticating as principal root/admin@BADBEAGLE.COM with password.
kadmin.local:  addprinc smith
WARNING: no policy specified for smith@BADBEAGLE.COM; defaulting to no policy
Enter password for principal "smith@BADBEAGLE.COM":
Re-enter password for principal "smith@BADBEAGLE.COM":
```

```
Principal "smith@BADBEAGLE.COM" created.
kadmin.local:  quit
```

The normal user must obtain a ”ticket granting ticket” before authentication will happen. The command is:

kinit

and the user’s password is then entered. If successful, a TGT will be issued to the client, which allows transactions with the NAS to commence. The mounted and Kerberized file system will be accessible by the user. See the chapter “Mounting the NAS to the Ubuntu Client” for complete discussion of this process.

6 Configuration Done on the Synology DiskStation

Log into the Synology web server using a web browser.

The following assumes the users and shared folders have already been set up and are fundamentally working. Synology documentation is quite good for this basic set-up. All of the documentation is available from inside the web-based GUI, and it can be referenced right along side the actual configuration dialogs in the browser.

6.1 Enable Kerberos Authentication on a Shared Folder

Select “Control Panel”.

Select “Shared Folder”.

Highlight the exported directory (in this example ds214). Click “edit” at the top of the GUI. Notice the GUI which pops up is “tabbed”. Select the tab at the far right, “NFS Permissions”. This allows the option to create multiple clients which can access the exported folder. Click “Create” to add additional clients. Select a client. Click “edit” at the top of the GUI. Under “Security”, switch to “Kerberos Authentication”.

The GUI will now show “Krb5”. Apply and close out the GUIs.

6.2 Set up the NFSv4 File Service

Select “Control Panel”. Select “File Services”. Select “SMB/AFP/NFS”. Click “Advanced Settings”. Click “Kerberos Settings”.

This GUI will allow the import of the nfs-server.keytab file which was generated on the KDC, and then secure-copied to the client workstation.

Also in this GUI, the mapping of users (file owners) can be completed. This illustrates how the translation of client names to server names is done.

Note the “Advanced Settings” button at the bottom of the Synology GUI. This is used to import the server’s keytab file:

Click “Advanced Settings” then click “Kerberos Settings” then click “Import” and navigate to the nfs-server.keytab file which was copied from the KDC to the Ubuntu client.

If the keytab is successfully imported, the GUI should display a list of principals and encryption types. Dropping back to the “Advanced Settings” GUI, there should be a line at the bottom like:

```
Principal:  nfs/diskstation@BADBEAGLE.COM
```

There is more configuration to be done. Return to the “Kerberos Settings” GUI. Note that are two tabs at the top, and the one on the left has already been used to import the keytab file. Now click the “ID Mapping” tab on the right. There is a “Create” switch/button. Switch to ”Suggested Mapping List”. This will populate a list of useful mappings. These mappings appear to based on obvious choices like root and admin plus user names which have already been set up for the file shares. Use the mouse to highlight all of them and OK to add them to the list. These auto-generated mappings seem to be sufficient in the test system so far.

This should complete the configuration of the Synology NAS with regards to NFS and the Kerberos system. The other aspect of configuration is security level of the shared folders. Going back to the Synology GUI:

Control Panel and then Shared Folder Click and highlight a shared folder in the list, and then click “Edit”. This opens another tabbed GUI with numerous settings for the client which has been granted access. The tab on the far right is “NFS Permissions”. Click this and numerous additional settings are revealed. The setting of interest is “Security:”. Uncheck the box for the default “AUTH_SYS”, and check the box for “Kerberos Authentication”. Note there are additional levels of Kerberos Authentication, however, only this level will be used in this project.

OK and close the GUIs.

6.3 Configure the Synology Shares for Kerberized Client Access

The Synology NAS will require configuration to allow client access to its exported NFS file shares. The following two screenshots show how this is accomplished.

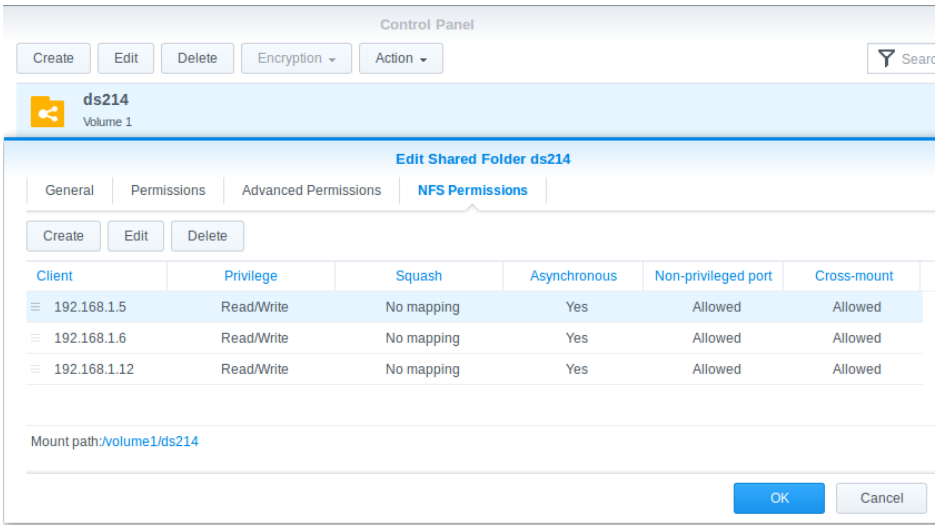

Figure 2: The Synology GUI " Control Panel → Shared Folder → Edit Shared Filter (folder name) "

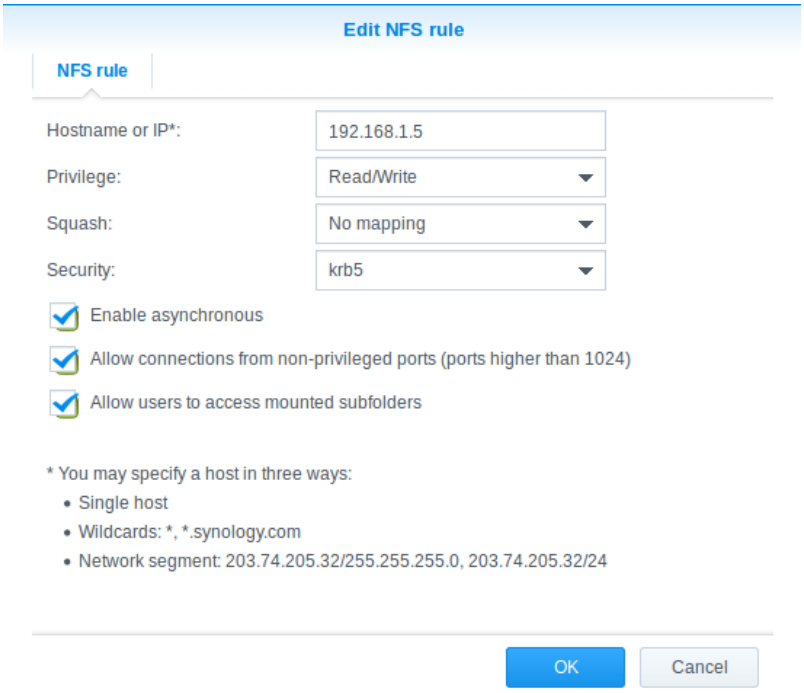

Figure 3: From "Edit Shared Folder (folder name) → NFS Permissions (tab)", highlight and Edit client. This GUI also allows new clients to be added to the list.

Note the switch is set to "Security: krb5". This enables Kerberos authentication on the NFS file share. OK the GUIs and this completes the client-specific configuration of the NAS.

6.4 Mysteries of the Synology DiskStation

Though the DiskStation is based on Linux, it does not have the Kerberos configuration files in the usual location. Instead, they are located in /etc/nfs. This can be seen by enabling ssh and logging into the NAS at the command line.

There is an additional file syno_nfs_conf which appears to correlate to the settings in the GUI.

Manually editing the Kerberos configuration files may be required in some systems. The test project described in this paper did not require additional configuration beyond what was described in the web GUI above.

7 Configuration Done on the Ubuntu Mate Client

The following describes the configuration of Ubuntu Mate 16.04 which must be completed before using NFSv4 and Kerberos.

7.1 Install NFS Packages

The Ubuntu distributions do not have the NFS client installed. However, NFS is easy to install using apt:

```
sudo apt install nfs-common
```

This is all that is required to enable NFSv4.

7.2 Install Kerberos Client Package

Install the Kerberos client package:

```
sudo apt install krb5-user
```

The installation process will bring up a GUI which will ask for three items (example inputs shown):

- Host name of kdc: kdc
- Host name of admin server: kdc
- Default realm: BADBEAGLE.COM

Note that in the above example the kdc and admin server are the same, which is a typical configuration.

The above install will create a file /etc/krb5.conf. This is an example template which will can be further customized. The example file shown in the KDC configuration section can be duplicated for the client.

7.3 Update the Client's Host File

All of the examples in this document use manual configuration of the hosts. It is assumed this is practical on a small home network. This eliminates the requirement of deploying DNS.

Here is an example /etc/hosts file which includes two hosts, a KDC, and an NAS:

```
127.0.0.1 localhost
192.168.1.5 ubuntu1
192.168.1.6 ubuntu2
192.168.1.7 diskstation
192.168.1.10 kdc
```

7.4 NFS and Kerberos: RPC.GSSD

There is another "daemon" which must be running for Kerberos to work with NFS. This is called "GSSD".

After installing NFS, the GSSD service should be running. Try this command:

```
systemctl list-units | grep gssd
```

The command should return two active services:

```
rpc-gssd.service
loaded active running   RPC security service for NFS client and server
rpc-svcgssd.service
loaded active running   RPC security service for NFS server
```

It is highly recommended to activate a high level of verbosity from GSSD. This is done starting with this command:

```
systemctl edit --full gssd
```

This will open a systemd unit file. The last line of the file needs the -vvv option added:

```
ExecStart=/usr/sbin/rpc.gssd -vvv
```

Upon system restart, the GSSD service will begin verbose logging to the /var/log/syslog file.

The best way to use this is to open a terminal use the tail command:

```
tail -f /var/log/syslog
```

This log was most useful on the client. The log can be monitored while attempting to mount an NFS file system with Kerberos security enabled.

7.5 Set Mapping Domain in file /etc/idmapd.conf

7.5.1 ID Mapping Configuration

The so-called ”domain” which is set in the idmapd.conf file is apparently not related to any actual domain. It is only important that the setting is exactly the same in both the NAS and the client.

In the Ubuntu client, the domain is set in the /etc/idmapd.conf file:

```
[General]

Verbosity = 0
Pipefs-Directory = /run/rpc_pipefs
# set your own domain here, if id differs from FQDN minus hostname
Domain = BADBEAGLE.COM

[Mapping]

Nobody-User = nobody
Nobody-Group = nogroup
```

In the NAS, the Domain setting is done using a GUI. This same GUI enables NFS and also NFS version 4:

```
Control Panel and then
File Services
Enable NFS (check box)
Enable NFSv4 support (check box)
NFSv4 domain: BADBEAGLE.COM  (typed into form box)
```

8 Mounting the NAS to the Ubuntu Client

8.1 Mount the Server with Kerberos Authentication

Mounting a file system must be done as "superuser". An example command to mount with Kerberos authentication:

```
sudo mount -t nfs4 -o sec=krb5 192.168.1.7:/volume1/ds214 /mnt/ds214
```

This could also be done using the server's host name:

```
sudo mount -t nfs4 -o sec=krb5 diskstation:/volume1/ds214 /mnt/ds214
```

The command is interpreted as follows:

- -t nfs4. File system type (-t) option is NFSv4.
- -o sec=krb5. The -o option indicates a comma separated list of options. In this case there is only a single option, so no commas are used. This is the Kerberos authentication option.
- diskstation:/volume1/ds214. This is the export from the NAS.
- /mnt/ds214. This is the mount point on the client. This directory should be created with root read/write permissions.

It is convenient to put the mount command in a one-liner bash shell script like this:

```
#!/bin/bash
sudo mount -t nfs4 -o sec=krb5 192.168.1.7:/volume1/ds216 /mnt/ds216
```

And the unmount script:

```
#!/bin/bash
sudo umount /mnt/ds216
```

Note that the mounting command triggers an event which causes Kerberos credentials to be issued to the client. This creates the "machine" credentials which allow the client to access the exported file system from the NAS. However, this is only superuser (root) access! A normal user must be authenticated to be granted access.

8.2 The Client Ticket

A normal non-priveged user on the machine which now has a Kerberized server mounted to it must be granted a ticket in order to access the server. This is done with the kinit command:

```
smith@ubuntu1:/mnt/ds216$ kinit
Password for smith@BADBEAGLE.COM: (enter password here)
```

If all is well, the user will be granted a ticket which allows the user to access the mounted server's file system. The user should see normal file ownership and permissions assuming the server has been configured correctly.

9 Conclusion

All of the configuration steps required for simplest possible home network deployment of Kerberos authentication for NFSv4 were covered in this article. The Beagle Bone series of Linux development boards was chosen as this design is believed to be a solid choice for long-term reliability. The Kerberos software components are easily installed thanks to the Beagle Bone's Debian-based distribution of Linux. The availability of a "console image" is another favorable aspect of Beagle Bone as this implements a basic principle of software security by minimizing "attack surface".

The lowest level of Kerberos security (authentication) was used throughout. The user can choose to take this to the next level if desired. NFS encryption is possible.

The configuration described does not use DNS, and thus limits the described system to a small home network. Perhaps the next step is to deply DNS and make the system more easily scalable.

10 Web Sites with Good Information on NFS and Kerberos

There are a few good sites with information on Kerberos and NFS. The following lists the sites that were found to be most useful.

10.1 Links to Information on Kerberos

http://www.kerberos.org/

https://web.mit.edu/kerberos/

https://wiki.debian.org/NFS/Kerberos

https://wiki.debian.org/LDAP/Kerberos

"Kerberos V5 Tutorial" by Ken Hornstein and Jeffrey Altman:

https://www.secure-endpoints.com/talks/Kerberos_Tutorial_BPW2007.pdf

https://www.kerberos.org/software/tutorial.html

This has some outdated material (July 2008), however, most is still relevant and this is one of the best resources for configuring Kerberos:

http://www.kerberos.org/software/adminkerberos.pdf

10.2 Links to Information on NFS

https://en.wikipedia.org/wiki/Network_File_System

Kerberos NFS authentication is dependent on "RPC":

https://en.wikipedia.org/wiki/Remote_procedure_call

https://help.ubuntu.com/community/SettingUpNFSHowTo

https://debian-handbook.info/browse/stable/sect.nfs-file-server.html

In Ubuntu 16.04:

/usr/share/doc/nfs-common/README.Debian.nfsv4