NFSRODS - Overview

- What
  - A new iRODS client
  - Presents iRODS as NFSv4.1
  - Allows an iRODS collection to be mountable
  - [https://github.com/irods/irods_client_nfsrods](https://github.com/irods/irods_client_nfsrods)

- Why
  - Provides a standard POSIX filesystem presentation to existing/legacy tools and applications
  - Provides full iRODS policy layer and enforcement

- How
  - A full nfs4j Virtual File System implementation
  - Implemented using Jargon
  - Deployed as a Docker container
NFSRODS v0.8 Release

Available today ...

Provides:

- Authentication: Trusted OS User
- Authorization: Traditional Unix Permissions
Initially built with a hard requirement on Kerberos. Why?

- We needed to distinguish users from each other.
- Kerberos provided access to the user's name which is what iRODS needed.
- NFS4J had built-in support for Kerberos.

The Good:

- It worked!
- It had built-in authentication.

The Bad:

- It was too complex to stand up quickly.
- It required knowledge of Kerberos and all of its tools.
- It couldn't be containerized because of Kerberos/Docker issues.
NFSRODS v0.8 - Current Authentication Model

- **Assumptions**
  - Authenticated access is via unix user with identically named iRODS user account.
  - Authenticated unix user is traversing the mount point (VM).
  - Entries in `/etc/passwd` and `/etc/shadow` are synced (uids/gids must match) on both the machine with the mount point (VM) and the machine running NFSRODS.

- **Note**
  - An authenticated user with sudo/root access on VM could appear to iRODS (and, therefore, all policy) as any user.
This model maps to traditional Unix permissions.
Permission masks change in real-time depending on who is accessing the mount point.
Groups are not (yet) supported.
Collections are always executable, while data objects are never executable.
iRODS users who have **own** permissions on a collection or data object are mapped into Unix-space as the owner.
iRODS users who have **read** or **write** permissions are mapped into Unix-space via world permissions.

<table>
<thead>
<tr>
<th>iRODS Permission</th>
<th>Collection as Directory</th>
<th>Data Object as File</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWN</td>
<td>drwx-----x</td>
<td>-rw------</td>
</tr>
<tr>
<td>WRITE</td>
<td>d--x---rwx</td>
<td>------rw-</td>
</tr>
<tr>
<td>READ</td>
<td>d--x---r-x</td>
<td>-------r--</td>
</tr>
<tr>
<td>NULL</td>
<td>d--x-----x</td>
<td>-----------</td>
</tr>
</tbody>
</table>
After early testing in an enterprise environment ...

The Good:
- Happy with deployment model (Docker)
- Happy with authentication model (Trusting the OS)
- Permissions mapping works for users

Other:
- Groups are missing
- Usage of world permissions was surprising/alarming to sysadmins

Suggestion:
- Can we have extended ACLs (getfacl, setfacl)?
NFSRODS - Deployment

1. Requirements:
   - iRODS 4.2.6
   - Update Collection MTime Rule Engine Plugin
   - Docker

2. Build the image (if desired):

   ```
   ubuntu$ git clone https://github.com/irods/irods_client_nfsrods
   ubuntu$ cd irods_client_nfsrods
   ubuntu$ docker build -t nfsrods .
   ```
3. NFSRODS Configuration:

```json
ubuntu$ cat /home/ubuntu/nfsrods_config/server.json
{
    "nfs_server": {
        "port": 2049,
        "irods_mount_point": "/tempZone",
        "user_information_refresh_time_in_minutes": 60,
        "file_information_refresh_time_in_milliseconds": 1000
    },
    "irods_client": {
        "zone": "tempZone",
        "host": "irods-server.ugm-2019",
        "port": 1247,
        "default_resource": "demoResc"
    },
    "irods_proxy_admin_account": {
        "username": "rods",
        "password": "rods"
    }
}
```
4. Launch the NFSRODS Docker container:

```bash
ubuntu$ docker run -d --name nfsrods \
    -p 3000:2049 \
    -v /home/ubuntu/nfsrods_config:/nfsrods_config:ro \
    -v /etc/passwd:/etc/passwd:ro \
    -v /etc/shadow:/etc/shadow:ro \
    nfsrods:latest
```

5. Create the mount point:

```bash
ubuntu$ sudo mkdir -p /mnt/the_nfsrods_mountpoint
ubuntu$ sudo mount -o sec=sys,port=3000 `hostname`:/ /mnt/the_nfsrods_mountpoint
```

6. Use the mount point:

```bash
bobby$ cd /mnt/the_nfsrods_mountpoint/home/bobby
bobby$ echo "science" > science.txt
bobby$ ls -l science.txt
-rw------- 1 bobby bobby 8 May 15 17:29 science.txt
bobby$ cat science.txt
science
```
NFSRODS v0.8 - Live Demo
GREAT!!!

Let's run all of our existing tools against NFSRODS, right?
Well ...
NFSRODS v0.8 - Caveats

• Speed
  ▪ NFSRODS slower than using direct clients (e.g. iCommands)

• Caching
  ▪ NFS caches file/directory information between all requests
  ▪ Possible information leakage
  ▪ Possible out-of-date information
  ▪ Increasing consistency decreases speed

Consider passing `lookupcache=none` as an additional option to `mount`. Although NFSRODS will be less responsive, the benefit to using this is that information will be more consistent and less likely to be leaked to users with more restrictive access.
NFSRODS - Future Work

- NFS 4.1 Access Control List (ACL) support
  - Standardized
  - Could enable support for groups
  - Removes the need for world permissions
  - Provides more granular control
- Parallel Transfer
- Unit Testing
- **NFStest** - POSIX Filesystem Level Access Testing
- Samba/CIFS - NFSRODS provides the reference implementation for making iRODS accessible to Microsoft Windows machines
Questions?

- Thank you!

- This version (NFSv4.1) of NFSRODS was built by:
  - Kory Draughn, iRODS Consortium
  - Alek Mieczkowski, iRODS Consortium
  - Mike Conway, NIH/NIEHS
  - Jason Coposky, iRODS Consortium
  - Terrell Russell, iRODS Consortium

- Inspired by work (NFSv3) presented at UGM2016 (slides, paper):
  - Danilo Oliveira, Center for Informatics UFPE, Brazil
  - I. Fé, Center for Informatics UFPE, Brazil
  - A. Lobo Jr., Center for Informatics UFPE, Brazil
  - F. Silva, Center for Informatics UFPE, Brazil
  - G. Callou, Center for Informatics UFPE, Brazil
  - V. Alves, Center for Informatics UFPE, Brazil
  - P. Maciel, Center for Informatics UFPE, Brazil
  - Stephen Worth, EMC Corporation

- Preliminary testing and feedback provided by:
  - Bristol-Myers Squibb Company