

# Securing the Nanohub Community Account

Aaron Shelmire

PSC Investigated the application of tools developed by Kevin Price of NCSA to secure the Nanohub “*community account*”. The original concept was to develop a set of restrictions that could be placed upon all “*community accounts*”. After developing an understanding of Stuart Martin’s Grid-Interop-Now! *community account*, it is apparent that this approach may not be suitable for all *community accounts* forthcoming.

Two tools were made available, the **chroot\_jail** and **commsh**. The nature of large scale computational runs renders the **chroot\_jail** environment far too cumbersome to implement for multiple accounts. The filesystems typically large enough to accommodate **chroot\_jail** environments for community accounts would be “scratch” space. “scratch” space is intended to be temporary. Utilizing this space would require exemptions to policies and would result in less and less usable space for computational results as community accounts are added. Further wasting disk space, each community account would require access to many of the same executable tools. This is in addition to the executables specific to the community, which typically reside in the area addressable via \$TG\_COMMUNITY. If only one or two *community accounts* would exist, the **chroot\_jail** may be an option, but current plans hoping for 30+ community style accounts will not allow for easy integration of such an environment.

The solution implemented uses only the **commsh** restricted shell environment. The most complicated issues encountered were non-technical in nature.

With the commsh in place, the *community account* only requires access to a shell script used to perform any actions while the job is running.

Technical issues encountered during the implementation are notably specific to PSC’s infrastructure. The Nanohub’s implementation assumes the ability to submit a job to a local jobmanager, and the ability to perform filestaging via Globus’s GridFTP functionality.

The PSC jobmanager included some customizations that required working around. Execution of system tools such as ‘echo’ were attempted via exec without their fully-qualified paths. In addition, these commands were attempted in a manner where multiple commands were included on a single line, delineated by the semi-colon character ‘;’. Similar issues were encountered with file-transfer utilities local to PSC’s environment.

The commsh was recently installed on the Xt3 system at PSC. We believe Community Accounts will be able to utilize this shell with ease. The entire installation and initial testing process should take a day or less. The supporting actions repeated for each Community Account added may take less than an hour. The process involves reviewing the executable pieces invoked by Globus, then copying those into place, configuring the Community Accounts .commsh.rc file to allow execution of those executable pieces, then finally allowing the community developers to test.

The developers of the community account are responsible for building a successful application, and testing the execution of the this application. The System support staff may wish to copy what the developers are trying to execute in an account they can run, so that they may observe the behavior, both inside the commsh and outside of the commsh.

## ***Securing a Community Account with commsh Step-by-Step:***

1. *Installation of the commsh utility.*
2. *Testing community shell (including Globus job submission to a community shell restricted account).*
3. *Set community account's shell to /path/to/commsh/bin/commsh*
4. *Edit /path/to/commsh/etc/commsh.conf to include*  
#Read the file 'commsh.rc' in the user's home directroy  
ReadConfig ~/.commsh.rc  
# Allow all users except for 'root' to run commsh  
AllowUser **communityUser**  
DenyUser root
5. *Retrieve **tested known-working** job script from technical staff of Community and place it in communityUser's home directory*
6. *Remove write access to the communityUser's home directory and all files included within.*
7. *Edit ~/.commsh.rc to include*  
DirectAccess /complete/path/to/home/**communityUser**/jobwrapper