

# Writing tools for the SRB as a dumb FS.

T. O. H. White<sup>1</sup>

<sup>1</sup>Department of Earth Sciences,  
University of Cambridge

SRB Workshop, 10th January 2006



# Outline

- 1 Motivation
- 1 Example 1: Sgrep
- 2 Example 2: TobysSRB
- 2 SRB authentication
- 2 Security handling
- 3 Conclusion



- Want to interface our handwritten tools with SRB
- SRB tools are imperfect



## SDSC recommended options:

- Move or Replicate data to local space.
- Modify your Application to make SRB client calls. (C, Perl, Python, Java APIs available)
- Modify your Application to use the UnixIO or srbio libraries.
- Use Solaris (and maybe Linux) PRELOAD without recompiling.
- Mount SRB as file system.
- Move the app to the data.



## My recommended option:

- **Treat the SRB as just distributed filesystem**
- Semantics of a simple hierarchical FS:  
Sls, Scat, Sput  
optionally Smkdir, Scp, Smv, Srm, Smkdir, Srmdir

Quickly build application with (Python/Perl/sh/scripting language of your choice)



My recommended option:

- Treat the SRB as just distributed filesystem
- Semantics of a simple hierarchical FS:

Sls, Scat, Sput

optionally Smkdir, Scp, Smv, Srm, Smkdir, Srmdir

Quickly build application with (Python/Perl/sh/scripting language of your choice)



My recommended option:

- Treat the SRB as just distributed filesystem
- Semantics of a simple hierarchical FS:  
Sls, Scat, Sput  
optionally Smkdir, Scp, Smv, Srm, Smkdir, Srmdir

Quickly build application with (Python/Perl/sh/scripting language of your choice)



My recommended option:

- Treat the SRB as just distributed filesystem
- Semantics of a simple hierarchical FS:  
Sls, Scat, Sput  
optionally Smkdir, Scp, Smv, Srm, Smkdir, Srmdir

Quickly build application with (Python/Perl/sh/scripting language of your choice)



How to search through SRB repository for a file.

- cannot grep without retrieving collection manually.

Let us invent ...

- Sgrep



How to search through SRB repository for a file.

- cannot grep without retrieving collection manually.

Let us invent ...

- Sgrep

## Pseudocode:

```
def Sgrep(pwd):  
    Sls pwd > list  
    for item in list:  
        if item is file:  
            Scat file | grep pattern  
        if item is directory:  
            Sgrep(directory)
```



Need to:

- Parse `sls` output (easy)
- Write `grep` routine (use language with inbuilt RE)

21 lines of POSIX `sh`

- Error checking
- Prettify output

400 lines of Python

```
#!/bin/sh

Sgrep=$0
pattern=$1
pwd=$2

firstline=true
for filename in `Sls $pwd`
do
    if [ $firstline = true ]
    then
        firstline=false
    else
        if [ ${filename%%/*} = C- ]
        then
            $Sgrep $pattern ${filename#C-}
        else
            Scat $pwd/$filename | grep $pattern
        fi
    fi
done
```



I don't like MySRB.

Write new web interface to SRB.

Complication - now running as web-server user.



## Pseudocode:

```
def cgiscript(operation, filepath):
    authenticate to SRB
    if operation == get_file:
        Scat filepath
    if operation == get_directory:
        Sls directory > list
        for item in list:
            if item is directory:
                link to cgiscript(get_directory, item)
            if item is file:
                link to cgiscript(get_file, item)
    if operation = put_file:
        Sput filepath
```



## Need to:

- Authenticate to SRB (see below)
- Manage SRB session (see below)
- Parse SLS output (easy)
- Worry about security (important, but easy: see below)
- Error checking / Prettify output

400 lines of Python



## Need to:

- Authenticate to SRB (see below)
- Manage SRB session (see below)
- Parse `SLs` output (easy)
- Worry about security (important, but easy: see below)
- Error checking / Prettify output

400 lines of Python



Sc commands use `.srb` directory for

- authentication,
- configuration,
- session management.

- `.srb/.MdasAuth` - contains password
- `.srb/.MdasEnv`

```
mdasCollectionHome '/home/tow.eminerals'  
mdasDomainHome    'eminerals'  
srbUser           'tow'  
srbHost           'forth.dl.ac.uk'  
srbPort           '5544'  
defaultResource   'CambsLake'  
AUTH_SCHEME       'ENCRYPT1'
```

- `.srb/.MdasEnv.$$` - as above plus current directory.

Sinit, Sexit, Scd manipulate these files.  
All unnecessary.

- `.srb/.MdasAuth` - contains password
- `.srb/.MdasEnv`

```
mdasCollectionHome '/home/tow.eminerals'  
mdasDomainHome    'eminerals'  
srbUser            'tow'  
srbHost            'forth.dl.ac.uk'  
srbPort            '5544'  
defaultResource    'CambsLake'  
AUTH_SCHEME        'ENCRYPT1'
```

- `.srb/.MdasEnv.$$` - as above plus current directory.

`Sinit`, `Sexit`, `Scd` manipulate these files.  
All unnecessary.



- `.srb/.MdasAuth` - contains password
- `.srb/.MdasEnv`

```
mdasCollectionHome '/home/tow.eminerals'  
mdasDomainHome    'eminerals'  
srbUser           'tow'  
srbHost           'forth.dl.ac.uk'  
srbPort           '5544'  
defaultResource   'CambsLake'  
AUTH_SCHEME       'ENCRYPT1'
```

- `.srb/.MdasEnv.$$` - as above plus current directory.

`Sinit`, `Sexit`, `Scd` manipulate these files.  
All unnecessary.



- `.srb/.MdasAuth` - contains password
- `.srb/.MdasEnv`

```
mdasCollectionHome '/home/tow.eminerals'  
mdasDomainHome    'eminerals'  
srbUser           'tow'  
srbHost           'forth.dl.ac.uk'  
srbPort           '5544'  
defaultResource   'CambsLake'  
AUTH_SCHEME       'ENCRYPT1'
```

- `.srb/.MdasEnv.$$` - as above plus current directory.

Sinit, Sexit, Scd manipulate these files.

All unnecessary.



## Create:

- temporary file containing password
- temporary file containing environment

Always specify full SRB path.

```
mdasAuthFile=/tmp/firstfile \  
mdasEnvFile=/tmp/secondfile \  
Scat /home/tow.eminerals/testfile
```



TobysSRB needs to (in order of security importance)

- 1 pass username & password from client to server.
- 2 pass data/files to and from client and server.
- 3 pass filenames and directory paths from client to server.



TobysSRB needs to (in order of security importance)

- 1 pass username & password from client to server.
- 2 pass data/files to and from client and server.
- 3 pass filenames and directory paths from client to server.



TobysSRB needs to (in order of security importance)

- 1 pass username & password from client to server.
- 2 pass data/files to and from client and server.
- 3 pass filenames and directory paths from client to server.

## Solutions:

- 1 is passed as name-value CGI arguments;  
2 is passed as the HTTP payload.

Through an SSL connection both of these are completely encrypted.

- 3 is stored in the URL;  
makes the UI friendlier, but leaves path info sniffable.

!! And massage all user input before execution !!



## Solutions:

- 1 is passed as name-value CGI arguments;  
2 is passed as the HTTP payload.

Through an SSL connection both of these are completely encrypted.

- 3 is stored in the URL;  
makes the UI friendlier, but leaves path info sniffable.

**!! And massage all user input before execution !!**



- Easy to interface with SRB as dumb FS using only three (or eight) Scommands
- Authentication/session management easy (once you know the trick)
- Secure sessions easy with SSL

No need for complicated SDSC-supplied APIs.



- Easy to interface with SRB as dumb FS using only three (or eight) Scommands
- Authentication/session management easy (once you know the trick)
- Secure sessions easy with SSL

No need for complicated SDSC-supplied APIs.



- Easy to interface with SRB as dumb FS using only three (or eight) Scommands
- Authentication/session management easy (once you know the trick)
- Secure sessions easy with SSL

No need for complicated SDSC-supplied APIs.



- Easy to interface with SRB as dumb FS using only three (or eight) Scommands
- Authentication/session management easy (once you know the trick)
- Secure sessions easy with SSL

No need for complicated SDSC-supplied APIs.

