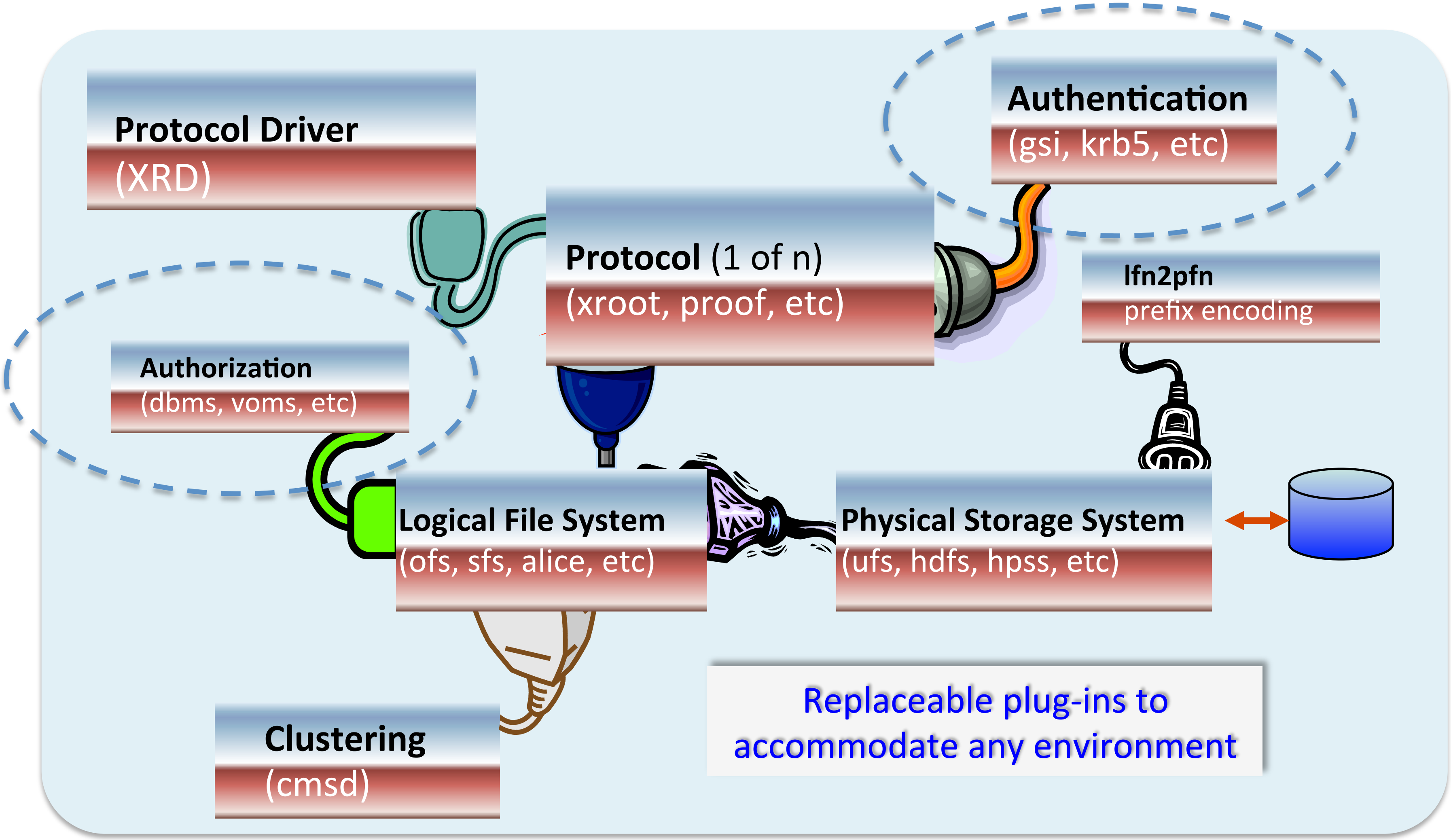
The Ins & Outs of XRootD Authentication & Authorization

G. Ganis, CERN, PH-SFT

XRootD Workshop @ UCSD

27-29 January 2015

Server Plug-In Architecture



Authentication

- Flexible architecture
  - Multiple protocol, easily expandable
  - Simultaneous heterogeneous protocols
    - Allow multiple administrate domains
- Simple administration
  - Server sets requirements
  - No or minimal client configuration

Abstract interface

```
class XrdSecProtocol {
public:
XrdSecEntity                   Entity;

virtual int                    Authenticate   (…) = 0;    // Server
virtual XrdSecCredentials *getCredentials(…) = 0;    // Client

virtual int                    Encrypt(…) = 0;
virtual int                    Decrypt(…) = 0;
virtual int                    Sign(…) = 0;
virtual int                    Verify(…) = 0;
}
```

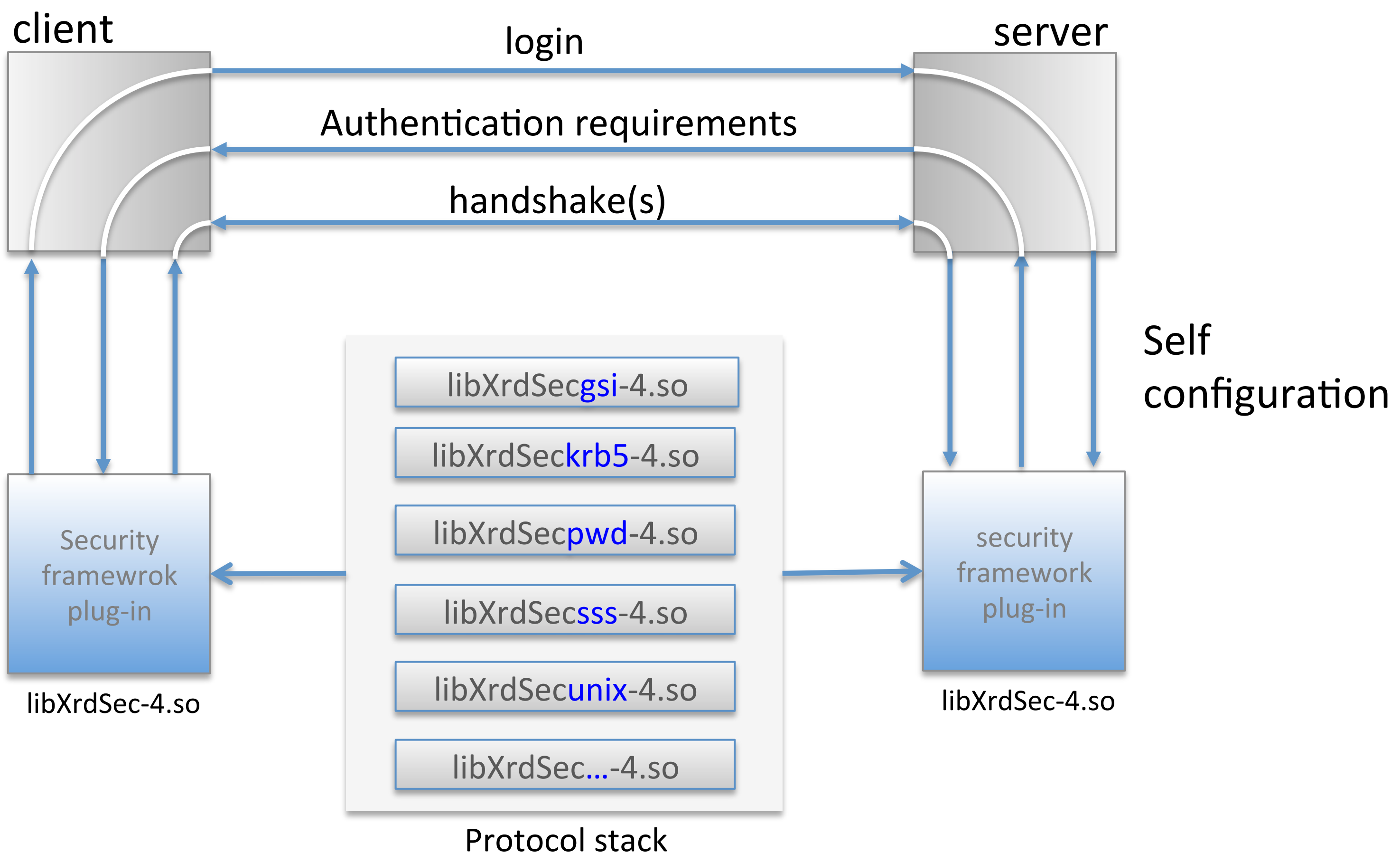
Passed to authz

Drive the handshake

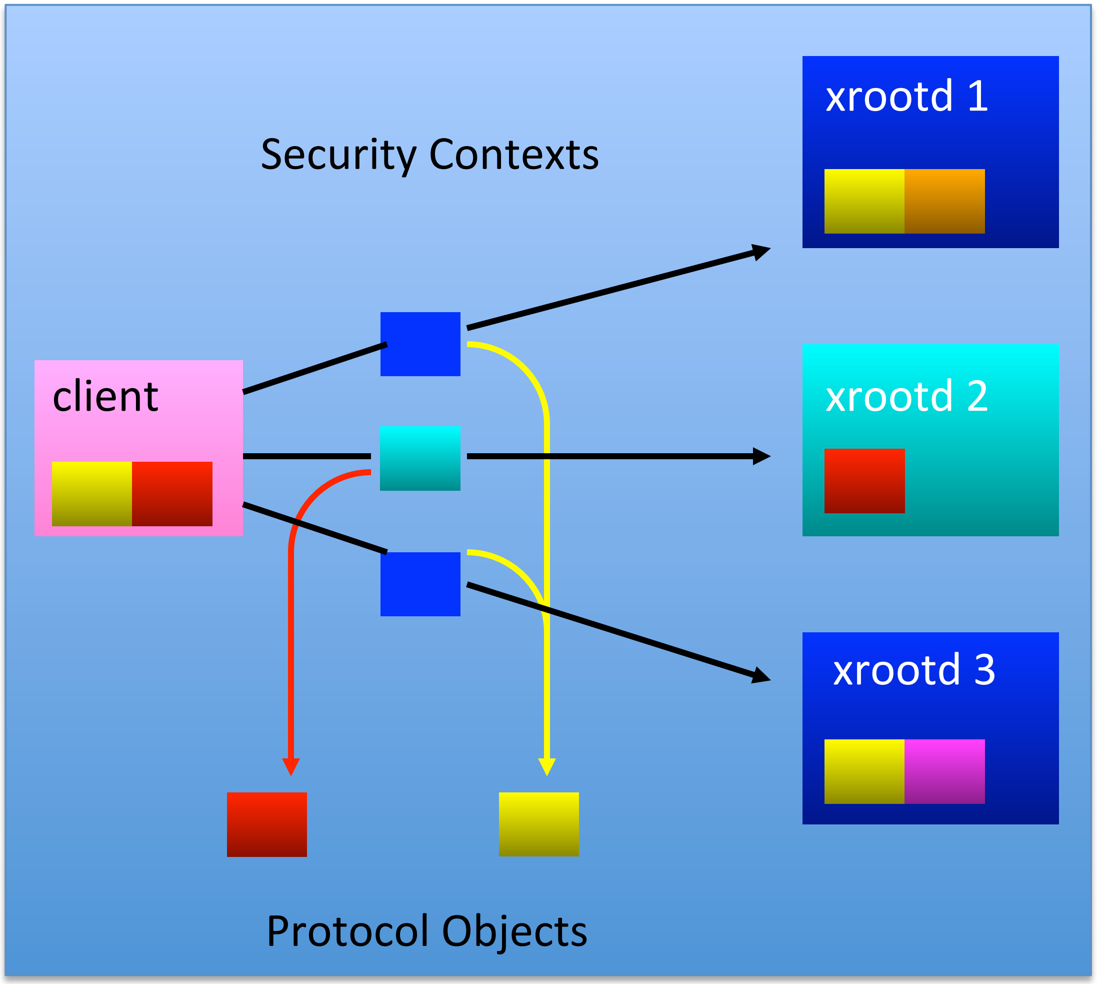
Based on the cipher session (not used)

The XrdSecProtocol object belongs to the XrdProtocol instance associated to the physical client connection

Architecture



Heterogeneous Security Support



- Servers have one or more protocol objects created at initialization time
- Client selects the protocol to use
- One security context per physical connection

XrdSecEntity

```
char    prot[XrdSecPROTOIDSIZE];  // Protocol used
char    *name;                    // Entity's name
char    *host;                    // Entity's host name dnr dependent
char    *vorg;                    // Entity's virtual organization
char    *role;                    // Entity's role
char    *grps;                    // Entity's group names
char    *endorsements;            // Protocol specific endorsements
char    *moninfo;                 // Additional info for monitoring
char    *creds;                   // Raw client creds or certificate
int      credslen;                // Length of the 'creds' field
int      rsvd;                    // Reserved field
XrdNetAddrInfo  *addrInfo;        // Connection details
const    char   *tident;          // Trace identifier always preset
void    *sessvar;                 // Plugin settable storage pointer
```

Server config directives

- Load the authentication framework
  - xrootd.seclib *so_path*

```
xrootd.seclib /opt/rooted/lib/libXrdSec-4.so
```

- Define protocols to load and its parameters
  - sec.protparm *protid parms*
  - sec.protocol [*libpath*] *protid* [*parms*]

```
Sec.protparm gsi –d:3
sec.protocol gsi  -dlgpxy:3
sec.protocol krb5
```

- Bind to a host
  - sec.protbind *hostpat* { **none** / [**only**] *protocols* }

```
sec.protbind * only gsi
sec.protbind *cern.ch krb5 gsi
sec.protbind lxplus*.cern.ch none
```

Available protocols

- Strong protocols
  - gsi
    - Globus Security Infrastructure
    - Used in all LHC data federations
  - krb5
    - Kerberos 5
  - pwd
    - Password-based
  - sss
    - Simple Shared secret
- *Identification* protocols
  - unix, providing {user, group}
  - host (built-in), providing host fqdn

Cryptography

- GSI and PWD use the cryptographic interface defined by XrdCrypto
- XrdCryptossl only concrete implementation based on OpenSSL
- SSS uses XrdCryptoLite cryptography
  - Also depends on OpenSSL (blowfish)
- Will need a replacement: libreSSL, …

Globus Security Infrastructure

- Mutual authentication, X509-based
- Basic configuration
  - Affecting the mutual C/S authentication
  - Uses Globus defaults for file locations
  - Can be tuned with parameter switches (server side) or environment variables (client side)
- Proxy delegation
  - Generation of a delegate proxy for downstream authentication
- DN-to-name mapping, extension interpretation
  - Alternative Grid map file functionality; special attribute extraction (VOMS …)

GSI basic server configuration

- Location of certificate, private key, CA dir

```
-cert:/cert/file    (/etc/grid-security/xrd/xrdcert.pem)
-key:/key/file      (/etc/grid-security/xrd/xrdkey.pem)
-certdir:/ca/dir    (/etc/grid-security/certificates)
```

- CRL handling

```
-crl:option              (use-if-available)
-crldir:/crl/dir         (same as CAdir)
-crlrefresh:frequency    (1 day)
```

- Debug

```
-d:debug_level           (none)
```

GSI basic client configuration

- Location of certificate, private key, CA dir

```
X509_USER_CERT       ($HOME/.globus/usercert.pem)
X509_USER_KEY        ($HOME/.globus/userkey.pem)
X509_USER_PROXY      (/tmp/x509up_u<uid>)
X509_CERT_DIR        (/etc/grid-security/certificates)
```

  – Can be passed in the URL

```
root://host:port/path?xrd.gsiusrpxy=/tmp/u_mine
```

- CRL handling

```
XrdSecGSICRLCHECK       (use-if-available)
XrdSecGSICRLDIR         (same as CAdir)
```

- Debug

```
XrdSecDEBUG             (none)
```

GSI configuration summary

```
•    150128 09:01:47 13069 secgsi_InitOpts: ***
     ------------------------------------------------------------ ***
•    150128 09:01:47 13069 secgsi_InitOpts:  Mode: server
•    150128 09:01:47 13069 secgsi_InitOpts:  Debug: 0
•    150128 09:01:47 13069 secgsi_InitOpts:  CA dir: /etc/grid-security/certificates/
•    150128 09:01:47 13069 secgsi_InitOpts:  CA verification level: 1
•    150128 09:01:47 13069 secgsi_InitOpts:  CRL dir: /etc/grid-security/certificates/
•    150128 09:01:47 13069 secgsi_InitOpts:  CRL extension: .r0
•    150128 09:01:47 13069 secgsi_InitOpts:  CRL check level: 1
•    150128 09:01:47 13069 secgsi_InitOpts:  CRL refresh time: 86400
•    150128 09:01:47 13069 secgsi_InitOpts:  Certificate: /etc/grid-security/hostcert.pem
•    150128 09:01:47 13069 secgsi_InitOpts:  Key: /etc/grid-security/hostkey.pem
•    150128 09:01:47 13069 secgsi_InitOpts:  Proxy delegation option: 0
•    150128 09:01:47 13069 secgsi_InitOpts:  GRIDmap file: /etc/grid-security/grid-mapfile
•    150128 09:01:47 13069 secgsi_InitOpts:  GRIDmap option: 1
•    150128 09:01:47 13069 secgsi_InitOpts:  GRIDmap cache entries expiration (secs): 600
•    150128 09:01:47 13069 secgsi_InitOpts:  Client proxy availability in XrdSecEntity.endorsement: 0
•    150128 09:01:47 13069 secgsi_InitOpts:  VOMS option: 1
•    150128 09:01:47 13069 secgsi_InitOpts:  MonInfo option: 0
•    150128 09:01:47 13069 secgsi_InitOpts:  Crypto modules: ssl
•    150128 09:01:47 13069 secgsi_InitOpts:  Ciphers: aes-128-cbc:bf-cbc:des-ede3-cbc
•    150128 09:01:47 13069 secgsi_InitOpts:  MDigests: sha1:md5
•    150128 09:01:47 13069 secgsi_InitOpts: ***
     ------------------------------------------------------------ ***
```

- Default on server
- Set XrdSecDEBUG to get it on client

Typical issues

- Most of the problems come from
  - Invalid or expired certificates
  - Missing or non-default located CAs
    - Should install using RPMs provided by OSG, EGI
  - Server name mismatch due to use of aliases

```
export XrdSecGSISRVNAMES="*/lx*.cern.ch"
```

  - …

GSI plug-in extensions

- Three hooks applied (in order of call) after a successful handshake:
  - GMAPFun
    - Alternative (to grid map file) DN-to-user mapping
  - VOMSFun
    - VOMS attributes extraction
  - AuthzFun
    - Generic 'authorization' function, allows redefinition of key for caching
- All update XrdSecEntity
  - Main goal: prepare it for the authorization step
    - May fail, providing 'authorization-like' filtering
- Examples how to write these kind of plug-ins are provided under src/XrdSecgsi

vomsxrd: VOMS extractor

- VOMS extractor plug-in based on VOMS libraries
  - Depends on libvomsapi.so
- Can be configured to extract information only for a given group and/or VO
  - Pre-selector functionality
- Distributed as RPM in the WLCG repository
  - E.g. vomsxrd-0.3.0 , vomsxrd-compat-0.3.0
  - Used by ATLAS and CMS

vomsxrd config options

```
certfmt=raw|pem|x509    Certificate format:  [raw]
                                   raw    to be used with XrdCrypto tools
                                   pem    PEM base64 format (as in cert files)
                                   x509   As a STACK_OF(X509)
grpopt=opt              What to do with the group names:  [1]
                                   opt = sel * 10 + which
                                   with 'sel'
                                     0     consider all those present
                                     1     select among those specified by
                                           'grps' (see below)
                                   and 'which'
                                     0     take the first one
                                     1     take the last
grps=grp1[,grp2,...]    Group(s) for which the information is extracted;
                        if specified the grpopt 'sel' is set to 1 regardless
                        of the setting.
vos=vo1[,vo2,...]       VOs to be considered; the first match is taken
dbg                     To force verbose mode
```

Authorization

- Acts at logical file system level
- Based on XrdSecEntity
- Using capabilities provides required scalability
  - Built-in mechanism à la NT
- Framework defined by XrdAccAuthorize
  - Can provide own implementation, for example to use VOMS

AuthZ: Access Envelopes

- Used by ALICE
  - Proposed and implemented for XRootD by A. Peters, D. Feitchinger
- Envelope passed as opaque data in file URL
  - Created by VO's file catalogue
  - Contains TURLs and access permissions
    - Encrypted and protected from modification
- Could be adapted to VOMS

Abstract interface

```
class XrdAccAuthorize {
virtual XrdAccPrivs Access(const XrdSecEntity *Entity,
                           const char             *path,
                           const Access_Operation oper,
                                 XrdOucEnv        *Env=0) = 0;
virtual int         Audit(const int accok,
                          const XrdSecEntity *Entity,
                          const char *path,
                          const Access_Operation oper,
                          XrdOucEnv *Env=0) = 0;
…
}
```

Main AuthZ config directives

- Enable

```
ofs.authorize
```

- Database file for built-in

```
acc.authdb path
```

- Load as a plug-in

```
ofs.authlib path [ parms ]
```

Built-in Authorization Model

- Capability based model
  - Each entity has a list of capabilities
- A capability is a path prefix-privilege pair
  - Any number of such pairs may be specified
  - More scalable when number of objects greatly exceeds number of entities
- Can mimic an access control model

Entities can be:
Hosts
NIS Netgroups
Unix Groups
Users

```
u  hab  rw  /fnal/files/usr/hab
        r   /cern/files
```

Builtin Authorization Entities

- *idtype id* { *path privs* | *tempid* } [...] [\ ]
  - **u** - user's name (can be DN)
    - Applied for specific user, as identified by authentication protocol
  - **g** - Unix group name
    - Applied when user is a member of the group
  - **h** - Host name
    - Applied when request originates from this host
      - Always fully qualify the host name and specify in lower case
  - **n** - NIS netgroup name
    - Applied when the triplet (hostname, username, domainname) is a member of the specified netgroup
  - **t** - template name
    - Specification substituted in future authorization records for *tempid*

Special Entities

Fungible

- **u =** { *path privs* | *tempid* } [• • •] [****]
  - User’s name replaces the first occurrence of **@=** in *path*
  - Allows specializing privileges by user’s name without listing all users
    - Only one such entry may exist
  - Example: **u = /usr/@=/files a**
    User hab has all privileges for /usr/hab/files

Default

- **u *** { *path privs* | *tempid* } [• • •] [****]
  - The entry applies to all users regardless of the originating host
  - Essentially default privileges
    - Only one such entry may exist
  - Example: **u * /files rws**

Builtin Authorization Privileges

- *idtype id* { *path privs* | *tempid* } [• • •] [****]

a - all privileges	**i** - insert (create)	**l** - lookup	**r** - read
d - delete	**k** - lock (unused)	**n** - rename	**w** - write

  - Positive and negative privileges allowed
    - Negative privileges always override positive privileges
  - Examples
    - **u aaa /foo rw**
      - User aaa has read/write privileges in /foo
    - **u abh /foo a-n**
      - User abh has all privileges except rename in /foo
    - **u xyz /foo –wind**
      - User xyz is denied write/insert/rename/delete privileges in /foo

Example of real config file

```
# X509 configuration

# Load security framework
xrootd.seclib /usr/lib64/libXrdSec.so

# VOMS extractor loaction and configuration
sec.protparm gsi -vomsfun:/usr/lib64/libXrdSecgsiVOMS.so
sec.protparm gsi -vomsfunparms:certfmt=raw|vos=atlas|grps=/atlas

# Load GSI security plugin
sec.protocol /usr/lib64 gsi -ca:1 -crl:3

# Enable authorization
acc.authdb /etc/xrootd/auth_file
acc.authrefresh 60
ofs.authorize
```

Summary

- XRootD AuthN & AuthZ
  - Fully configurable, extendable, even replaceable
- Standards-based authentication
  - GSI, Kerberos 5, password, shared secret
- Builtin capability-based authorization
  - Extensive privilege support, auditing
- Good model for application level security
  - Addresses well current needs

Documentation

- Configuration Reference guide
  http://xrootd.org/doc/prod/sec_config.htm
- For building plug-ins examples in the relevant source code directories and cmake files

Questions ?