dCache & xroot
TLS, Tokens, Login Protection

Albert L. Rossi (FNAL)

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

What's new in 2020-2021

1. TLS in xroot doors and pools (since 6.2)
2. SciToken support (since 6.2)

- NB: not guaranteed to work for third-party copy (TPC). Pending discussion with the SLAC xrootd development team.

3. Login protection ['ZTN'] (since 7.1)

- NB: not guaranteed to work for third-party copy (TPC). Pending discussion with the SLAC xrootd development team.

dCache xroot: TLS

1. Should be preferred to signed hash verification (more secure, standard).
2. Can be enabled for both doors and pools.
3. CAVEAT: ***must*** be enabled to protect tokens (pools included!)

dCache xroot: TLS

- **TWO PARTY: dCache does not require token to log in to the pool (uses UUID);**

— BUT—

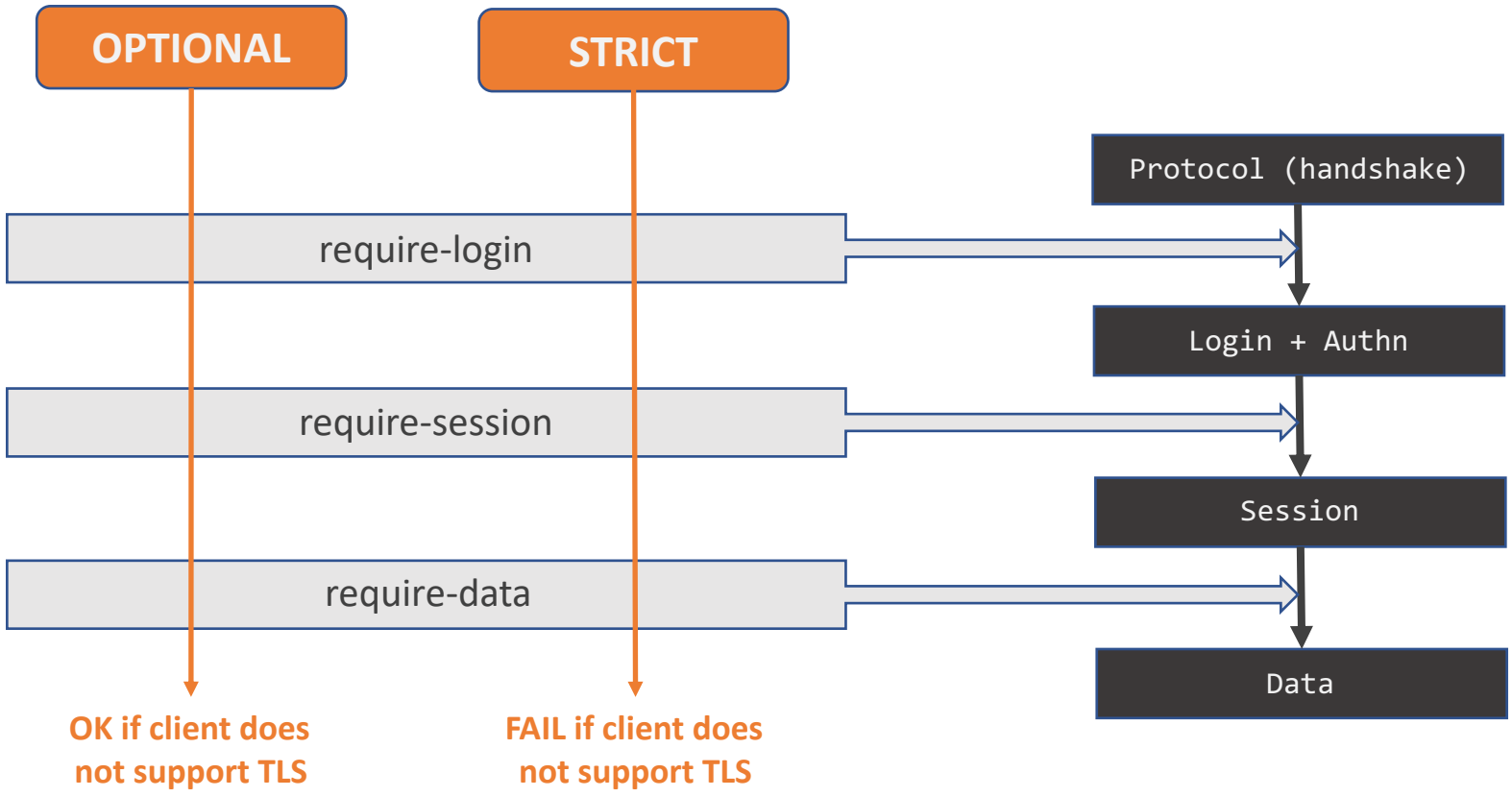
- **Unfortunately, the token is currently exchanged as part of the URL query included on the transfer path, and the xrootd client sends this to the pool on redirect;**

— SO —

- **If you are using Scitokens, TLS must be activated on both doors and pools.**

(dCache has never implemented splitting control from data channels in xroot; we are currently investigating whether the amount of work necessary to do so would be justified.)

xrootd.security.tls and the xrootd protocol



TLS Best Practice (doors)

EXAMPLE FOR DOOR using token authorization

```
[xrootd-1096-${host.name}Domain]
[xrootd-1096-${host.name}Domain/xrootd]
xrootd.cell.name=xrootd-1096-${host.name}
xrootd.net.port=1096
xrootd.authz.write-paths=/
xrootd.authz.read-paths=/
xrootd.security.tls.mode=STRICT
xrootd.security.tls.require-login=true
xrootd.plugins=gplazma:none,authz:scitokens
xrootd.plugin!scitokens.strict=true
```

THESE PATHS DEPEND ON HOW YOU SET UP SCITOKENS

DO NOT ALLOW CLIENTS WHICH DON'T SUPPORT TLS

BEGIN THE HANDSHAKE AT LOGIN (TOKEN IS EXCHANGED LATER)

NO AUTHN MODULE IS NECESSARY; SCITOKENS IS AUTHZ

DO NO AUTHORIZE WITHOUT A TOKEN (NO FALLBACK TO ANONYMOUS)

IT IS NOT STRICTLY NECESSARY TO START TLS AT LOGIN, SINCE THE TOKEN IS EXCHANGED DURING THE FILE OPEN REQUEST; HOWEVER, WITH LOGIN PROTECTION (SEE LATER) THIS IS NECESSARY.

IF YOU ARE USING TLS JUST TO PROTECT THE SESSION/DATA, BUT THE AUTHORIZATION PROTOCOL IS, E.G., GSI, YOU HAVE THE OPTION OF SETTING TLS TO BEGIN AFTER THE AUTHN HANDSHAKE ('session') (see next slide).

```
xrootd.security.tls.require-session=true
xrootd.plugins=gplazma:gsi,authz:none
```

TLS Rule of thumb

- **GSI**
  - mode = OPTIONAL
  - required-session=true
- **SciTokens**
  - mode = STRICT (doors)
  - required-login=true

TLS Best Practice (pools)

EXAMPLE FOR POOL using token authorization

I recommend setting these globally in the ***dcache.conf*** *file, to avoid a misconfiguration allowing an xroot door to redirect to a pool that does not support TLS.*

```
pool.mover.xrootd.security.tls.mode=OPTIONAL
pool.mover.xrootd.security.tls.require-login=true
```

OR = STRICT, IF *ALL* CLIENTS ARE GUARANTEED TO BE TLS CAPABLE
AGAIN, 'LOGIN' NOT STRICTLY NECESSARY EXCEPT WHEN USING ZTN

There are other properties having to do with TLS in xroot, but these are set to reasonable defaults and can usually be left alone.

NOTE: we have refactored 7.2 to use a somewhat more efficient SSL encryption (native pass-through rather than the Java implementation) which should give us a little better performance.

SciTokens configuration

GPLAZMA

This is the same for both xroot and https. In `gplazma.conf`, add

```
auth    sufficient  scitoken
```

In addition, you will need to define properties for issuers and audience-targets.
Be sure that the root paths on the issuers correspond to the read/write paths set your doors.

gplazma.scitoken.issuer!TEST = https://demo.scitokens.org / org.dcache.auth.UserNamePrincipal:arossi uid:8773
gplazma.scitoken.audience-targets = https://demo.scitokens.org

EXAMPLE XRDCP READ WITH SCITOKEN

```
xrdcp -f xroot://fndcatemp2.fnal.gov:1096//pnfs/fs/usr/test/data_100mb?authz=Bearer%20<encoded-token>  /dev/null
```

Login protection (ZTN)

- This is a login protocol assuring the server that the client is capable of obtaining a legitimate token.
- Though it was not strictly necessary for dCache to implement this, we did so simply to conform to the xroot protocol specification.
- In essence, a minimal token specifying `issuer` and `audience` (`sub` and `scope` are not required here) is passed in at login and if the server recognizes these as valid, the user is allowed to continue.
- This token is meant as a single use throw-away, and is not treated as fallback for the authorization token to be passed on the request URL.

Login protection (ZTN)

For the xroot client, this login token can be made available by setting the env variable for file location:

```
BEARER_TOKEN_FILE=/tmp/bt_u8773
```

ZTN can be enabled on the doors.

xrootd.plugins=gplazma:ztn,authz:scitokens ON THE DOORS, IT IS A GPLAZMA AUTHN MODULE

There is also a module available for the embedded third-party client on the pools, enabled as usual using:

pool.mover.xrootd.tpc-authn-plugins=gsi,unix,ztn

BUT AS MENTIONED EARLIER, HOW THIS WILL WORK WITH TPC IS YET TO BE ESTABLISHED.

Login protection (ZTN)

Since ZTN is a kind of sub-protocol based on the SciToken specification, to enable it, you must prepare gPlazma as you would to use Scitoken authorization. It would not make much sense, in fact, to use ZTN without enabling SciTokens.

Once again, there are several other properties available for configuration (which reflect attributes or settings specified by the xroot protocol), but these can usually be left alone.

Further Information

dCache, *The Book*, Chapter on Xrootd, gPlazma

https://dcache.org/old/manuals/Book-7.2/config-xrootd.shtml
https://dcache.org/old/manuals/Book-7.2/config-gplazma.shtml

Documentation in defaults:

```
pools.properties
xrootd.properties
xrootd-scitoken.properties
xrootd-ztn.properties
```