XRootD Server-Side TLS

XRootD Workshop

IN2P3, Lyon

June 11 - 12, 2019

Andrew Hanushevsky, SLAC

http://xrootd.org

Introduction

- The **XRootD** server will have TLS in R5
  - Improves security and data integrity
  - A requirement for token-based authorization
    - SciToken, Macaroons, etc.
  - Introduces backward compatibility challenges
    - Only R5 clients are TLS capable
  - This talk discusses
    - TLS configuration
    - Backward compatibility options

Basic TLS Directive

xrd.tls *cpath* [*kpath*] [[**no**]**dump** [**hsto** *sec*]

- *cpath*
  - Path to host certificate
- *kpath*
  - Path to private key
    - If not specified the cert must contain the private key.
- **dump**
  - Dumps full ssl error stack
    - Default is **nodump** as top-level error message is sufficient.
- **hsto** *sec*
  - SSL handshake timeout.
    - Default is 10 seconds after which the connection is closed.

What xrd.tls gives you

- Enables TLS on port 1094 (default port)
  - It's merely enabled, not forcibly TLS'd
  - Client can request its socket to change to TLS
    - Server may require the client change it
      - If the client doesn't it will get an error response
  - Change normally happens during handshake
    - Client is always free to change it at any time
  - Once socket in TLS mode it can never go back
- This provides backward compatibility!

Specifying a TLS-only port

xrd.port tls *number*
xrd.network tls *tls_port_socket_options*

- Port *number* is permanently in TLS mode
  - Client connections to port must be in TLS mode
    - Can never be downgraded to non-TLS mode

Certificate Verification

xrd.tlsca noverify | {**cadir**|**cafile**} *path* [*options*]

options: [**log** {**failure** | **off**}] [**verdepth** *vdn*]

- Disables or enables x509 cert verification
  - This is a required directive because of security
    - You must make a choice here
  - To enable you need to specify one or both:
    - Directory containing the trusted CA certificates
    - File containing all of the trusted CA certificates
  - Other options are esoteric
    - Specify then only if you really need them

XRootD Protocol and TLS

- You can specify when TLS is required
  - Not all operations need TLS
- Flexible backward compatibility
  - You can slowly phase-in TLS
    - Except for certain new types of requests
- Specified using the xrootd.tls directive

The xrootd.tls Directive

```
xrootd.tls [capable] reqs
reqs: [all] [[-]data] [[-]gpfile]
      [[-]login] [none] [[-]session]
      [-][tpc]
```

- capable - TLS reqs only apply to R5+ clients
  - data - Parallel data sockets
  - gpfile - Getfile and Putfile requests (default)
  - login - Login request
  - none - TLS not required
  - session - Any request after login
  - tpc - Old-style third party copy

Why such specificity?

- XRootD can split data & control
  - Not everything necessarily needs TLS (like data)
    - Similar to the gridFTP approach
  - Many say that TLS is cheap
    - You will max out I/O before CPU
      - True for streaming copies but not much else
  - TLS introduces latency (server & client)
    - More memory-to-memory copies plus CPU
  - This can be significant at scale (100's of clients)
    - Most visible in relatively small-read workloads
      - Performance loss is inevitable without lots of tuning

Conclusion

- With advent of TLS new possibilities open
  - Novel authentication & authorization schemes
    - It may be possible to eliminate x509 user certs
- TLS essentially replaces request signing
  - An **XRootD** feature to detect MIM attacks
    - We don't know of anyone who's enabled it
  - Considering deprecating signing in the future
    - Then again dCache **XRootD** recently implemented it
      - So who knows