CERN

Michal Simon

XRootD: the importance of R5

Outline

- A lot about TLS
- What's else in R5
- What's on the horizon

The Next BIG ONE: R5

- Introduces (and sets ground) for many new features, most notably **encryption**
  - Breaks plug-in ABI in some cases
    - Some external plug-ins will need to recompile (e.g. EOS!), no source changes required

- First release candidate cut in November 2019, second coming soon, final release **planned for 2020 Q1**

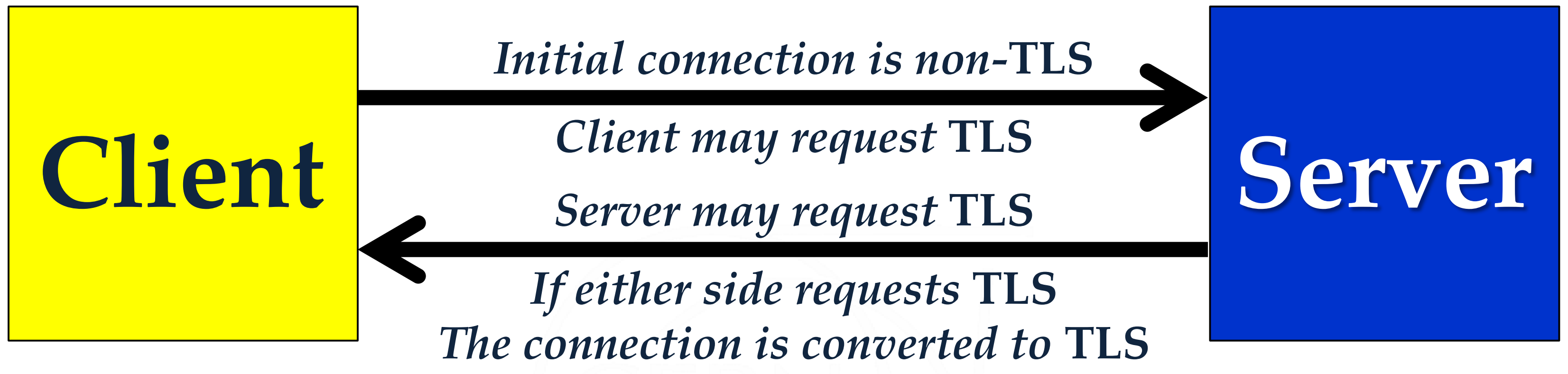
TLS: Transport Layer Security

- Why to do it?
  - Allow for **authorization token** handling (e.g. SciTokens)
  - Further **evolution of TPC** mechanism
  - Improves **security and data integrity**
  - Transfer confidential data with root/xroot protocol
- Biggest challenges:
  - Backward compatibility and forward migration path

Flexibility first!

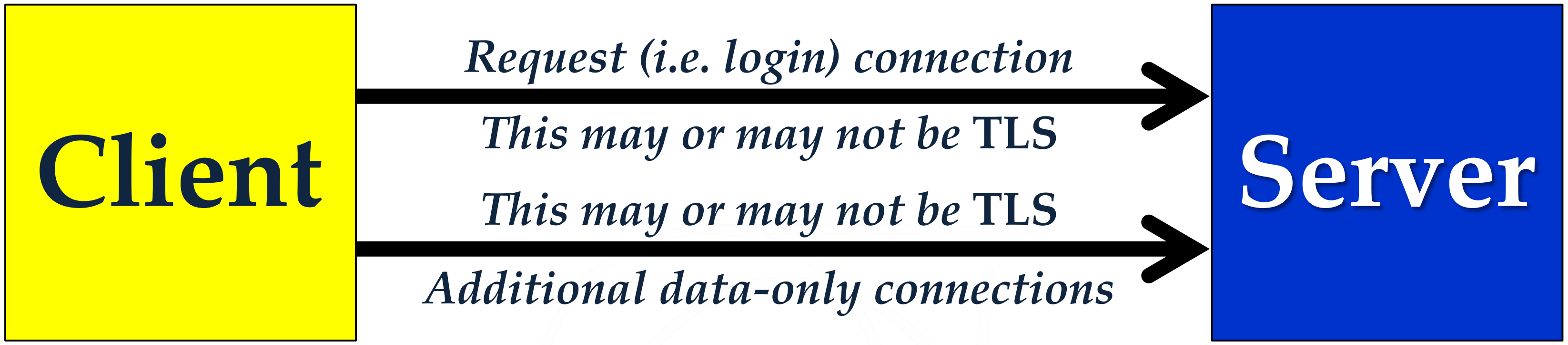
- Not every client supports TLS
  - We need to supply backwards compatibility
- Not everything needs TLS
  - we need to account for operational context
- A connection may or may not require TLS
  - at the discretion of the client, or
  - at the insistence of the server

Flexibility first!


Client
Initial connection is non-TLS
Client may request TLS
Server may request TLS
If either side requests TLS
The connection is converted to TLS
Server

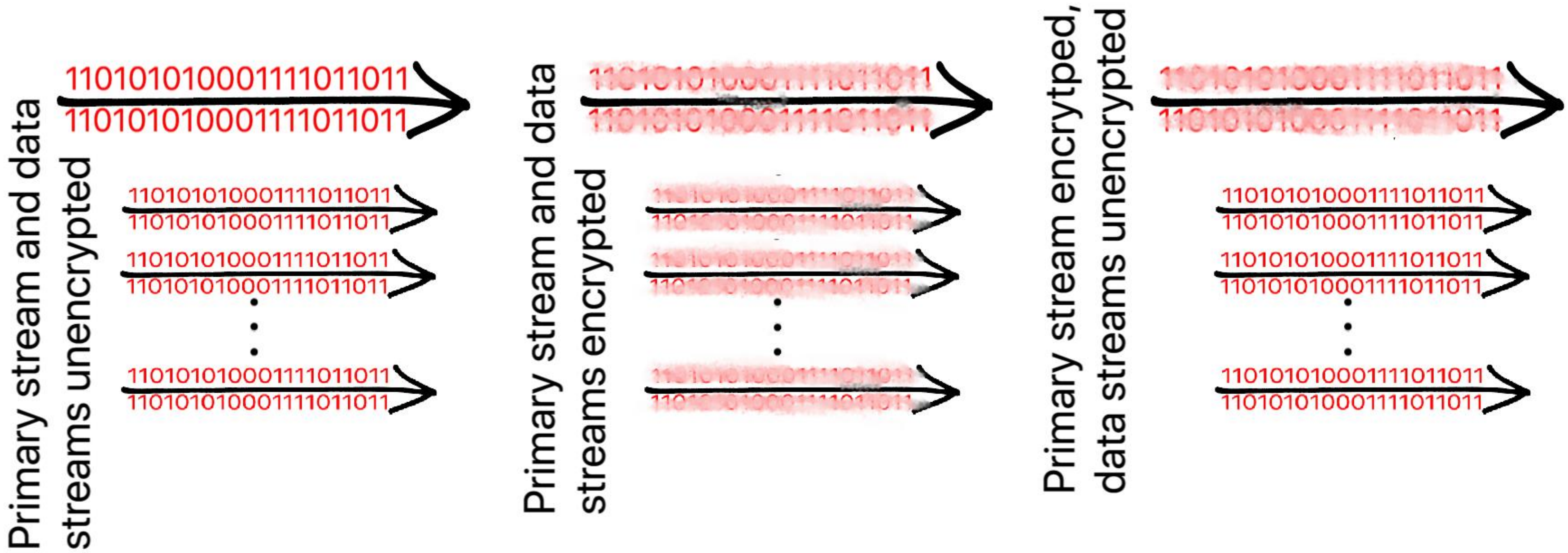
- The heart of flexible TLS is negotiations
  - Ability to **go from non-TLS to TLS at any time**
  - Provides backward compatibility and eases migration
  - **No special ports are needed!**

Flexibility first: super flexibility


Client
Request (i.e. login) connection
This may or may not be TLS
This may or may not be TLS
Additional data-only connections
Server

- The server may instruct the client to **encrypt only the control stream but not the data** (reads and writes payload)
  - Useful for HEP use case (similar to gridFTP)

Flexibility first: at a glance


Primary stream and data streams unencrypted
Primary stream and data streams encrypted
Primary stream encrytped, data streams unencrypted
11010101000111011011

What triggers TLS?

- Client URL that uses **roots** or **xroots**
  - E.g.: xrdcp roots://server//mydata /tmp
- **Server configuration**
  - TLS may be required for certain contexts
    - Third Party Copy
    - All TLS-capable clients
    - Control channel only
    - For all data

XRootD TLS implementation

- Based on OpenSSL, All typical deployment versions are supported:
  - Version 1.1.0 and above
  - Version 1.0.x with custom hostname verification
  - Should also work with 0.9

- All TLS actions are logged
  - What version of TLS is being used
  - When connection switches to TLS

- OpenSSL asynchronous API with an event-loop (including TLS handshake, occasionally a read operation may require a write event and *vice-versa*)

R5: more than TLS …

- Plug-in stacking
- New general monitoring stream
- Better containerization coexistence
- XCache improvements
- Extended stat
- Client channel-level plug-ins (allow for redirections between protocols)

R5: Extended Attributes

- Allow adding metadata to a file
  - Server exposes only user namespace

- Done via C++ or Python API or xrdfs command
  - Also, xrdcp has an option to preserve xattr

- Requires underlying file system support (most file systems have it)

Post R5

- RDMA support for better HPC integration
  - Maybe use cases in modern DAQ (reassembles more and more HPC)?
- Appending data to ZIP archives (with server side support)
- Request bundling, file descriptor and memory splicing
- TPC put/get requests encapsulating whole process
  - multiprotocol, access tokens
- uid/gid tracking for files/directories
- Streaming with end-to-end data verification

Summary

- Significantly extends usability
  - Important as XRootD is embedded in many HEP storage systems (EOS!, DPM, CTA, dCache (Java implementation), QServ)
  - New experiments are also relying on XRootD
    - E.g. Dune, LSST, LCLS II

- Addresses new use cases, e.g. access tokens, growing importance of XCache

Questions?