
CERN

Michal Simon

Client side implementation of secure xroot/root protocol

Outline

- Secure root/xroot protocol
- Client enforced encryption
- Server enforced encryption
- Implementation

Secure xroot/root protocol

- Secure xroot/root protocol using TLS for encryption
- Prerequisite for
  - **token** based authentication/authorization
  - further **evolution of TPC** mechanism
  - transferring **confidential data** with xroot/root protocol

Highlights

- **Based on OpenSSL**
  - For performance reasons OpenSSL asynchronous API has been employed
- Uses **same port for encrypted and unencrypted traffic** to simplify service operations
- **Level of encryption can be enforced by server** (potential replacement for request signing)

Client enforced encryption

- By choosing **roots/xroots** protocol, the user enforces encryption
- The client notifies server by setting `kXR_wantTLS` flag in `kXR_protocol` request

Server enforced encryption

- Server **may force the client to convert to TLS** by setting `kXR_gotoTLS` flag in `kXR_protocol` response

- Server may ask the client to:
  - `kXR_tlsLogin`: to convert to TLS right away
  - `kXR_tlsSess`: to convert to TLS after logging in
  - `kXR_tlsData`: to encrypt read and write requests
  - `kXR_tlsTPC`: to encrypt requests issued due to TPC

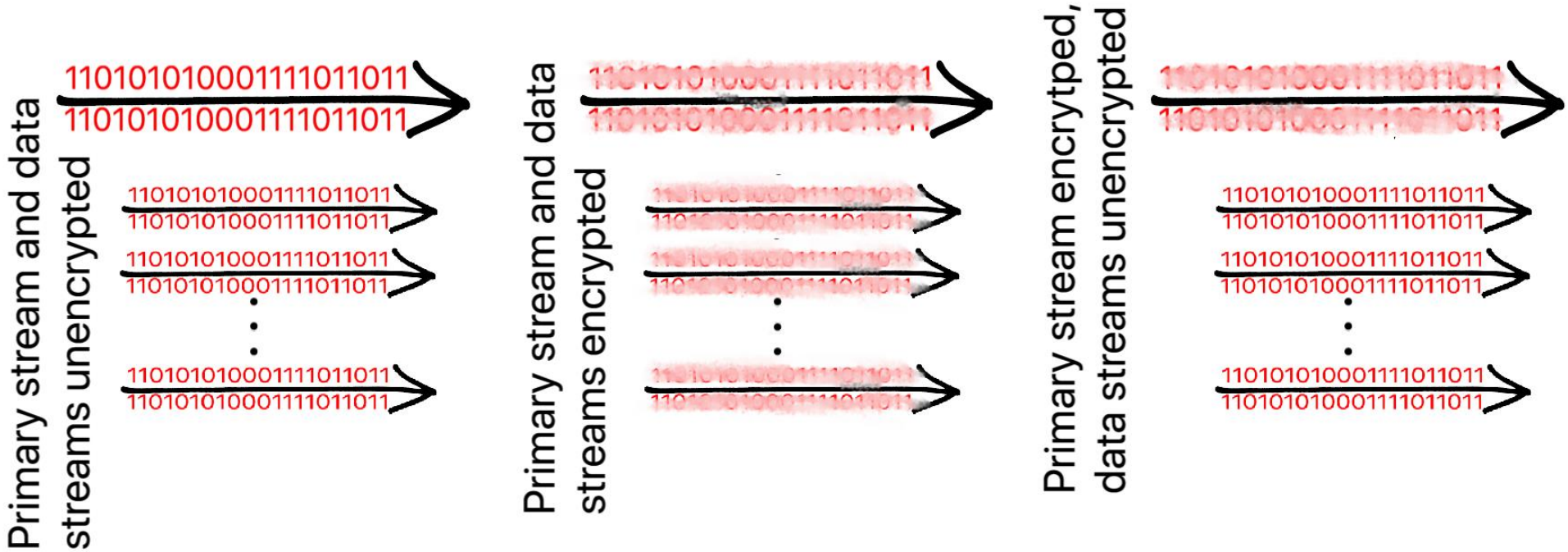
Expected

- The server knows in which cases to convert to TLS (e.g. `kXR_tlsData`), but doesn't know what will happen after `kXR_protocol`
- The client knows what will follow (e.g. `kXR_bind`), but doesn't know the configuration of the server
- To avoid additional RTTs, the client has to let the server now in advance what is the next request to come (e.g. `kXR_ExpBind` + `kXR_tlsData` = upgrade to TLS)

Encrypt only control stream

- The server might instruct the client to encrypt only the control stream (0) but not the data (reads and writes)
- **Useful for the HEP use case**
- In this case client has to open and bind a second unencrypted connection right away after the main encrypted connection has been established

Encrypted / unencrypted / ctrl encrypted only



TLS upgradable connection

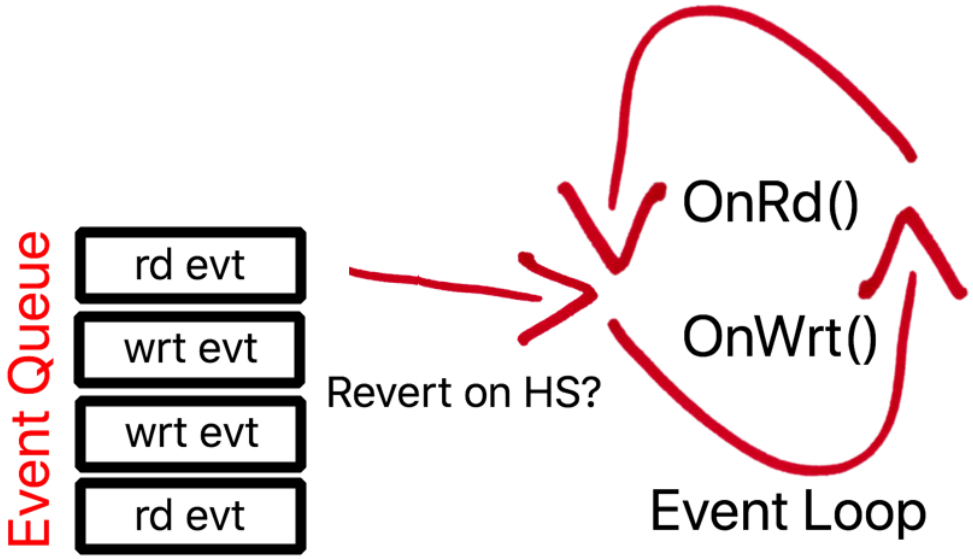
- Every connection by default is unencrypted
- During kXR_protocol request/response exchange it is decided if the connection should be encrypted
- Hence, **the connection abstraction** (socket, socket handler) **is upgradable to TLS**
- This design allows XRootD to use **same port for unencrypted and encrypted traffic**

Implementation

- For performance reasons we have chosen to use:
  - **OpenSSL asynchronous API** with an event loop
  - **OpenSSL socket BIO** (avoids unnecessary in-memory copies of data)

Implementation

- Since server may decide to redo TLS handshake at any time any SSL_read might require a write event and vice versa SSL_write might require a read event
- In this case we have to revert the event loop and call read callbacks on write events and write callbacks on read events



Questions?