SIEMENS
Ingenuity for life

IEC/IEEE 60802, Joint Project Call, Aug. 30 2021

Bootstrapping IA-Stations Using NETCONF/YANG – The 60802 Recipe

K. Fischer, A. Furch, O. Pfaff

Outline of this Presentation

SIEMENS
Ingenuity for life

1. **Which obstacles have to be overcome by security for IEC/IEEE 60802?**
   - Considering IA-stations in factory default state and NETCONF/YANG security as specified by the IETF
2. **Are they real?**
   - Using Netopeer2 to provide evidence
3. **What is the rationale for specifying NETCONF/YANG according an '*security-always-on*' paradigm?**
   - Applying common wisdom
4. **Should IEC/IEEE 60802 adopt the IETF paradigm (NETCONF/YANG '*security-always-on*')?**
   - Considering alternatives and their price-tags

NETCONF/YANG Security Paradigm

SIEMENS
Ingenuity for life

NETCONF/YANG: *security-always-on*

- ~~NETCONF-plain or~~ NETCONF-over-TLS (or another secure transport)
- ~~No; unilateral or~~ mutual authentication
- ~~Without or~~ with rules/permissions (NACM)

Web*: security-as-an-option*

- HTTP-plain or HTTP-over-TLS
- No authentication (HTTP-plain); unilateral or mutual authentication (HTTP-over-TLS)
- Without or with rules/permissions (many flavors)

- Synopsis of employment options for cryptographic security:
  i. *No security*: cryptographic security is **not employed** ← *the OT legacy*
  ii. *Security-as-an-option*: cryptographic security is available; **opt-in possible** ← *IEC 61158 (some CPFs)*
  iii. *Security-by-default*: cryptographic security is enabled; **opt-out possible** ← *feasible with IEC 61158*
  iv. *Security-always-on*: cryptographic security is mandated; **opt-out impossible** ← *NETCONF/YANG*

NETCONF/YANG Building Blocks (focusing on TLS)

SIEMENS
Ingenuity for life

- Internet standards:
  - RFC 6241 Network Configuration Protocol (NETCONF) 🡐 *protocol for configuring a network component;* ***establishing "security-always-on" as the NETCONF/YANG paradigm***
  - RFC 7589 Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication 🡐 *message exchange protection for network configuration*
  - RFC 7950 The YANG 1.1 Data Modeling Language 🡐 *basic info model for network configuration*
  - RFC 8341 Network Configuration Access Control Model 🡐 *resource access authorization for network configuration*
  - RFC 8342 Network Management Datastore Architecture (NMDA) 🡐 configuration state handling
  - RFC 8808 A YANG Data Model for Factory Default Settings 🡐 *info model for factory reset*
- NETCONF WG draft documents (work-in-progress):
  - draft-ietf-netconf-trust-anchors-15 A YANG Data Model for a Truststore 🡐 *info model for trust anchors*
  - draft-ietf-netconf-keystore-22 A YANG Data Model for a Keystore 🡐 *info model for credentials*
  - draft-ietf-netconf-crypto-types-20 YANG Data Types and Groupings for Cryptography 🡐 *underpinnings*

Processing Pipelines in NETCONF/YANG (with TLS)

SIEMENS
Ingenuity for life

- NETCONF/YANG client:
  1. Establish TLS session with mutual authentication (RFC 7589, RFC 5246)
  2. Check expected vs. actual server identification (RFC 6125)
  3. Request configuration operation esp. imprinting (RFC 6241, RFC 7950, RFC 8342, RFC 8808)

- NETCONF/YANG server:
  1. Establish TLS session with mutual authentication (RFC 7589, RFC 5246)
  2. Map client certificate to NETCONF username (RFC 7589, section 7)
  3. Enforce client authorization (RFC 8341)
  4. Perform configuration operation esp. imprinting (RFC 6241, RFC 7950, RFC 8342, RFC 8808)

Quote from RFC 7589:
*The NETCONF protocol [RFC6241] requires that the transport protocol's authentication process results in an **authenticated NETCONF client identity** whose **permissions** are known to the server.*
The authenticated identity of a client is commonly referred to as the NETCONF username.

NETCONF/YANG Security (with TLS) Plain Vanilla

SIEMENS
Ingenuity for life

Configuration component
(NETCONF and TLS client)

Trust anchor for LDevID

LDevID-NETCONF* credential

IA component
(NETCONF and TLS server)

Trust anchor for LDevID

LDevID-NETCONF* credential

Cert-to-name mapping

Name-based permissions

TLS Handshake
(plain vanilla with LDevID-NETCONF credentials at both ends)

Server is authenticated (RFC 5246)
Check expected/actual server identity (RFC 7589)

Client is authenticated (RFC 5246)
Map client certificate to
NETCONF username (RFC 7589)

*NETCONF-over-TLS(*NetworkConfig*)*

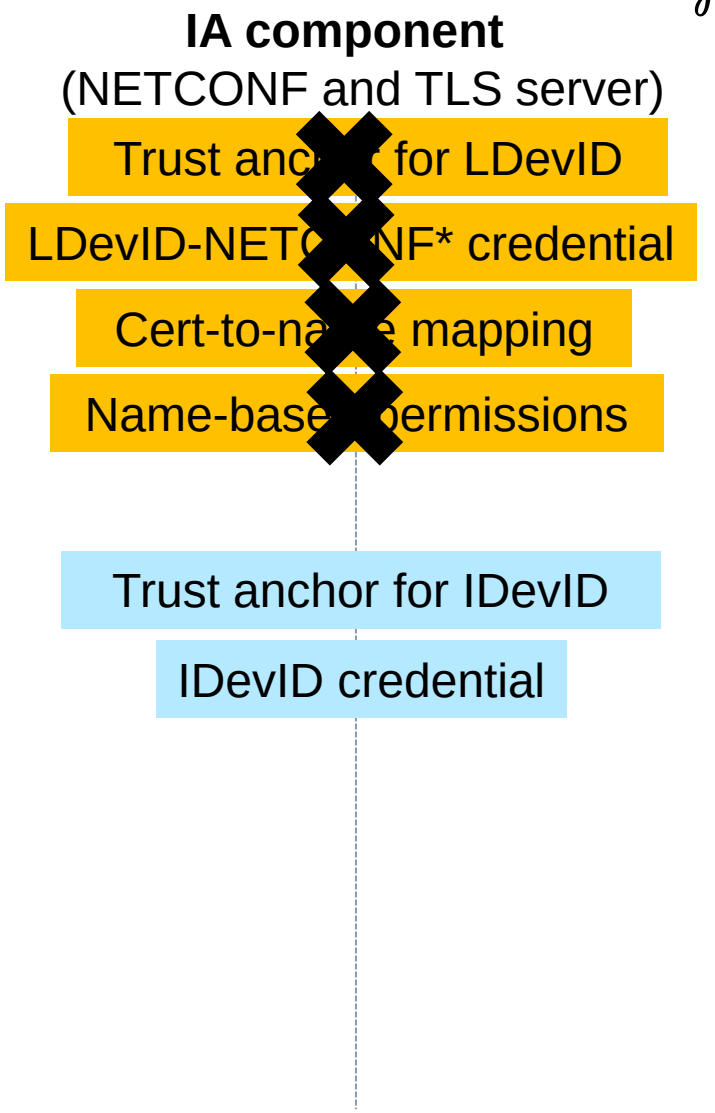
Check if client is authorized (RFC 8341)
Perform network configuration (RFCs 6241/7950/8342))

Network config

*: short-hand term for an 802.1AR LDevID that fulfills the prerequisites of RFC 7589 (NETCONF-over-TLS)

Factory Default State

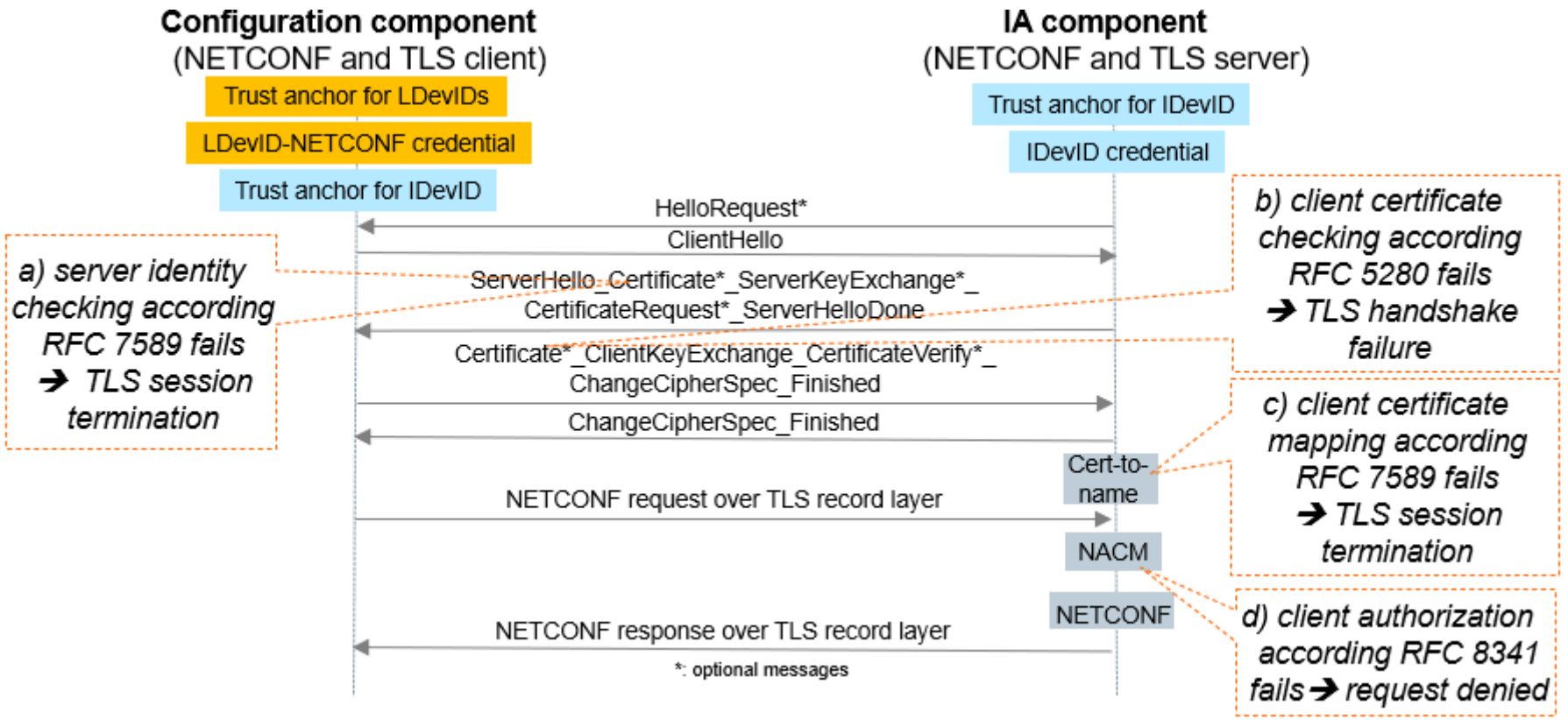
- Objects containing deployment-specific information (not known at IA-station manufacturing time) can be part of the factory default state:
  - Trust anchor for LDevID: key and subject/issuer name are deployment-specific
  - LDevID-NETCONF: hostname or IP address are deployment-specific
  - Cert-to-name mapping: fingerprints of CA (and/or EE) certificates in LDevID-NETCONF credentials are deployment-specific
  - Name-based permissions: NETCONF usernames (mapped from EE certificates in LDevID-NETCONF credentials of configuration clients) are deployment-specific



NETCONF/YANG Security (with TLS) Blocking Points for Factory Default State

SIEMENS
Ingenuity for life

- Blocking points: the *security-always-on* model presents challenges when components are in **factory default state**:



Reality Check

SIEMENS
Ingenuity for life

- Done with the NETCONF/YANG source-code package: **Netopeer2**
- Using following components:
  i. **Netopeer2 server**, representing the IA-system:
    - Setup represents the ‘factory default state’ (trust anchor for IDevID, IDevID credential)
    - Runs in Ubuntu (version 20.04.2 LTS)
  ii. **Netopeer2 remote client** using NETCONF-over-TLS, representing the configuration client:
    - Setup represents the ‘commissioning default’ state (trust anchor for LDevID, LDevID credential, IDevID credential)
    - Runs in Ubuntu (version 20.04.2 LTS), is hosted on another system as the Netopeer2 server
  iii. **Netopeer2 local client** using unix-socket mode, a Netopeer2 package component that does not map to IA-systems (no network communications ➔ no actual NETCONF client):
    - No setup needed in terms of trust anchor for LDevID, LDevID credential, IDevID credential
    - Runs in Ubuntu (version: 20.04.2 LTS), is hosted on the same system as the Netopeer2 server, needs to be executed with ‘root’ privilege

Blocking-Point a) Reality Check

SIEMENS
Ingenuity for life

- Blocking-point a) was not witnessed with the native **Netopeer2 remote client** and the **Netopeer2 server** in ‘factory default state’. But this is a ‘false positive’:
  - The native Netopeer2 remote client apparently violates RFC 7589 and RFC 6125, section 6.6.
  - This violation is not infrequent, see *The Most Dangerous Code in the World: Validating SSL Certificates in Non-Browser Software*
- The blocking-point a) was witnessed with following clients:
  - **Modified Netopeer2 remote client**

```
> connect --tls --san test.com
There is SAN intput from client. The input is test.com
[LOG] SubjectAlternativeName (actual): jiye-test-server.com
[LOG] SubjectAlternativeName (expect): test.com
[ERR] Compare vs. Expect SubjectAlternative Name does NOT match
[ERR] Terminate TLS session
```

Client config (YANG modules*):
ietf-truststore: **LDevID** and **IDevID** trust anchor
ietf-keystore: **LDevID-NETCONF** credential

Server config (YANG modules):
ietf-truststore: **IDevID** trust anchor
ietf-keystore: **IDevID** credential
ietf-netconf-server: **IDevID** trust anchor fingerprint

*: the Netopeer2 client does not use YANG objects, terms are used to ease comparison

  - **Web browsers** (can not talk NETCONF-over-TLS, talk HTTP-over-TLS; sufficient for blocking-point a)

Blocking-Point b) Reality Check

SIEMENS
Ingenuity for life

- Blocking-point b) was witnessed with the native/modified **Netopeer2 remote client** and the **Netopeer2 server** in ‘factory default state’ (snapshot shows output of Netopeer2 server):

```
[INF]: LN: Accepted a connection on 0.0.0.0:6513.
[INF]: SR: Session 44 (user "root", CID 1) created.
[INF]: SR: Session 45 (user "root", CID 1) created.
[ERR]: LN: Cert verify: fail (self signed certificate in certificate chain).
[ERR]: LN: SSL_accept failed (certificate verify failed).
```

Client config (YANG modules*):
ietf-truststore: **LDevID** and **IDevID** trust anchor
ietf-keystore: **LDevID-NETCONF** credential

Server config (YANG modules):
ietf-truststore: **IDevID** trust anchor
ietf-keystore: **IDevID** credential
ietf-netconf-server: **IDevID** trust anchor fingerprint

*: the Netopeer2 client does not use YANG objects, terms are used to ease comparison

Blocking-Point c) Reality Check

SIEMENS
Ingenuity for life

- Blocking-point c) was witnessed with the native/modified **Netopeer2 remote client** and the **Netopeer2 server** in ‘factory default state’ plus blocking-point b) resolution (snapshot shows output of Netopeer2 server):

```
[INF]: LN: Cert verify: depth 1.
[INF]: LN: Cert verify: subject: /CN=Jiye CA/C=DE/ST=Bavaria/L=Munich.
[INF]: LN: Cert verify: issuer:  /CN=Jiye CA/C=DE/ST=Bavaria/L=Munich.
[INF]: LN: Cert verify CTN: cert fail, cert-to-name will continue on the next cert in chain.
[INF]: LN: Cert verify: depth 0.
[INF]: LN: Cert verify: subject: /CN=Jiye client/C=DE/ST=Bavaria/L=Munich.
[INF]: LN: Cert verify: issuer:  /CN=Jiye CA/C=DE/ST=Bavaria/L=Munich.
[INF]: LN: Cert-to-name unsuccessful, dropping the new client.
[ERR]: LN: SSL_accept failed (certificate verify failed).
```

Client config (YANG modules*):
ietf-truststore: **LDevID** and **IDevID** trust anchor
ietf-keystore: **LDevID-NETCONF** credential

Server config (YANG modules):
ietf-truststore: **IDevID** and **LDevID**** trust anchor
ietf-keystore: **IDevID** credential
ietf-netconf-server: **IDevID** trust anchor fingerprint

*: the Netopeer2 client does not use YANG objects, terms are used to ease comparison
**: added in order to overcome blocking-point b), using the Netopeer2 local client

Blocking-Point d) Reality Check

SIEMENS
Ingenuity for life

- Blocking-point d) was checked in a happy-day-scenario using the native/modified **Netopeer2 remote client** and the **Netopeer2 server** in ‘factory default state’ plus blocking-points b) and c) resolutions plus 60802 permissions (role-based, no snapshot taken as all is okay now):

Client config (YANG modules*):
ietf-truststore: **LDevID** and IDevID trust anchor
ietf-keystore: **LDevID-NETCONF** credential (with 60802 role information in the EE certificate)

Server config (YANG modules):
ietf-truststore: IDevID and **LDevID**** trust anchor
ietf-keystore: IDevID credential
ietf-netconf-server: IDevID and **LDevID**** trust anchor fingerprints
ietf-netconf-acm: 60802 permissions*** (role-based)

*: the Netopeer2 client does not use YANG objects, terms are used to ease comparison
**: added in order to overcome blocking-points b), and c) using the Netopeer2 local client
***: added to check d)

- Note: authorization fails in many cases including:
  - Client EE certificate was authenticated, maps to name (role), access to data node is not permitted
  - Client EE certificate was authenticated, maps to name (role), requested protocol operation is not permitted
  - Client EE certificate was authenticated, does not map to name (role)

Rationale for the *Security-Always-On* Paradigm

SIEMENS
Ingenuity for life

- Mandating **resource access authorization** for network configuration makes sense from IETF perspective:
  - Network configuration resources can be regarded instantiations of the class "*private resources exposed at public-facing (or internal network-facing) endpoints*"
  - AllowAll, denyAll, askUser, throwACoin do not make much sense ➔ requiring the to-be-configured entity to be able to deterministically deciding about granting/denying configuration requests makes more sense
  - The requires to be able of authenticating the entity that requests configuration changes (enforcing e.g. "*only JaneDoe can read CompanySecrets*" does not make much sense if anybody can impersonate Jane)
- Mandating **message exchange protection** becomes a corollary:
  - "No security" is not an option – at least requestor authentication is a need
  - "Security-as-an-option" would be inconsequent and can easily go wrong – forget to enable protected exchanges and resource accesses
  - "Security-by-default" would be inconsequent too and can go wrong too – accidentally disable protected exchanges and resource accesses
  - In addition protection of configuration exchanges against manipulation is needed

Adopt the *Security-Always-On* Paradigm or Deviate

SIEMENS
Ingenuity for life

- Key question: how to move IA-stations from the ‘factory default state’ to the ‘’NETCONF/YANG security plain vanilla’ prerequisite or another functional state (without security)?
  - Approach A: use NETCONF/YANG, resolve blocking points a)-d)
  - Approach B: use or invent another means of communication and state management for the deployment-specific security setup
  - Approach C: do not use cryptographic security; invent “NETCONF/YANG-plain/unprotected”
- IEC/IEEE 60802 security uses approach A in D1.3:
  - Approach A: no functional loss at modest specification and implementation costs (see *Time-Sensitive Networking Profile for Industrial Automation,* D1.3, section 6.3.3.4 for details)
  - Approach B: no functional loss at large specification and implementation costs
  - Approach C: deviation from IETF specs with functional losses (no cryptographic security ➔ isolation, isolation blocks new uses cases in I4.0) at modest specification and implementation costs

Resolution Strategy for Blocking-Points a)-d)

SIEMENS
Ingenuity for life

a) Check the server identification based on **product master data** contained in IDevID EE certificates

b) Perform a "**provisional accept of client certificate**" that tolerates an actual error situation as an interim with the obligation for the caller to immediately provide additional information that allows the callee to resolve the error situation

c) Skip certificate-to-name mapping when IA-stations are in factory default state

d) Use NACM according to a **role-based model** with IEC/IEEE 60802-defined role names. With this approach it is possible to relief owner/operator organizations from the obligation to coin NACM objects for IEC/IEEE 60802 resources by themselves. Assignment of role names is a means to describe a cohort of client instances.

Note: see *Time-Sensitive Networking Profile for Industrial Automation,* D1.3, section 6.3.3.4 for details

Summary

SIEMENS
Ingenuity for life

- NETCONF/YANG security according the IETF belongs to the “security-always-on” paradigm
- IEC/IEEE 60802, D1.3 security adopts the IETF paradigm and closes the gap (blocking points a-d) between the IETF-defined NETCONF/YANG security as well as IA-stations that are in factory default state
- In particular, the IEC/IEEE 60802 security recipe covers following phases:
  - **Bootstrapping phase**: use NETCONF/YANG security to set-up IA-stations that are in factory default state for plain vanilla NETCONF/YANG security – contained in D1.3, see 6.3.3.4
  - **Plain vanilla phase I**: use plain vanilla NETCONF/YANG security to configure IA-stations for TSN – contained in D1.3, see 6.3.3.5
  - **Provisioning phase I**: use plain vanilla NETCONF/YANG security to set-up IA-stations for security in further IEC/IEEE 60802-defined exchanges – contained in D1.3, see 6.3.3.6
  - **Plain vanilla phase II**: use security in further IEC/IEEE 60802-defined exchanges – not part of D1.3
- It allows middleware/applications to utilize IEC/IEEE 60802 means for their security set-up:
  - **Provisioning phase II**: use plain vanilla NETCONF/YANG security to set-up IA-stations for security in middleware/application exchanges – offer by IEC/IEEE 60802; accept by 3rd parties

Authors

Kai Fischer, Siemens AG, T RDA CST SES-DE,
kai.fischer@siemens.com

Andreas Furch, Siemens AG, T RDA CST SES-DE,
andreas.furch@siemens.com

Oliver Pfaff, Siemens AG, T RDA CST,
oliver.pfaff@siemens.com