CSR Generation

Donnerstag, 28. April 2022 10:17

Plug&Trust MW Documentation:

AN13030

Also see file:///C:/Users/tbue/Desktop/smartCAN/compilation/simw-top/doc/pycli/doc/cli_commands_list.html

Preparation:

9.3.2 Communication interface (cmake SMCOM setting)

The communication interface to the secure element used by the `ssscli` tool is determined by the native shared library used. The communication interface supported by the shared library is controlled by the cmake `SMCOM` value.

We have the following options:
JRCP_V2, JRCP_V1, JRCP_V1_AM, VCOM, **SCI2C**, T1oI2C, PCSC, None

See

```
Usage: ssscli connect [OPTIONS] subsystem method port_name

  Open Session.

  subsystem = Security subsystem is selected to be used. Can be one of "se05x,
  auth, a71ch, mbedtls, openssl"

  method = Connection method to the system. Can be one of "none, sci2c, vcom,
  t1oi2c, jrcpv1, jrcpv2, pcsc"

  port_name = Subsystem specific connection parameters. Example: COM6,
  127.0.0.1:8050. Use "None" where not applicable. e.g. SCI2C/T1oI2C. Default
  i2c port (i2c-1) will be used for port name = "None".

Options:
  --auth_type [None|PlatformSCP|UserID|ECKey|AESKey|UserID_PlatformSCP|ECKey_PlatformSCP|A
                                  Authentication type. Default is "None". Can
                                  be one of "None, UserID, ECKey, AESKey,
                                  PlatformSCP, UserID_PlatformSCP,
                                  ECKey_PlatformSCP, AESKey_PlatformSCP"
  --scpkey TEXT                   File path of the platformscp keys for
                                  platformscp session
  --help                          Show this message and exit.
```

# You can connect to the SE050 like this:
$ ssscli connect se05x t1oi2c /dev/i2c-0:0x48

# We could reset the SE050 like this (optional):
$ ssscli se05x reset

# And disconnect like this:
$ ssscli disconnect

Also we need the following OpenSSL Config File in /etc/ssl/

The libsss_engine.so is in /usr/lib/ on Automation-One Device. We have to change this in the config file below

openssl_sss
_se050

Step 1: Key Generation on the SE050

See https://docs.aws.amazon.com/iot/latest/developerguide/transport-security.html for the supported TLS Cipher Suites of AWS IoT -> We can use NIST_P256 or RSA which is also supported by the SE050. NIST_P256 is recommended.

$ ssscli generate ecc 0x20181006 NIST_P256

Step 2: Get the Refpem File from the SE050

$ ssscli refpem ecc pair 0x20181006 privkey.pem

Step 3a: Creating the CSR

Use the OpenSSL Config File for the SE050 by setting the environment variable. This also has an impact on the systemd service of our awsclient:

$ export OPENSSL_CONF=/etc/ssl/openssl_sss_se050.cnf

# The OpenSSL Engine for the SE050 will use the values from the EX_SSS_BOOT_SSS_PORT environment variable. You can set it like this
$ export EX_SSS_BOOT_SSS_PORT=/dev/i2c-0:0x48

# Read the UID of the SE050
$ ssscli se05x uid
# Example: 04005001fbf14cf6a64270043b8212946680

# Generate the CSR like this (interactive):
$ openssl req -new -key privkey.pem -out csr.pem

# Or like this (non-interactive which is recommended for manual creation for the five Automation-One Test Devices):
$ openssl req -subj "/CN=IoT_Gateway_XYZ/O=automation-one" -new -key privkey.pem -out csr.pem

Or like this (non-interactive within a script which is recommended for productive use):
$ openssl req -subj "/CN=$device_name/O=$customer_name" -batch -new -key privkey.pem -out csr.pem

Step 4: Issue the Certificate (can only be done by OSB connagtive)

Send the csr.pem file(s) to tobias.buening@alten.com. Signed Email can be used if desired

Step 5: Copy the Certificate to the Device

Copy the received cert.pem file to /data/os/aws/certs/ on the IoT Gateway