Proof of Concept of device attestation over OPC-UA protocol

Arto Juhola

16. February 2022

General

This is an collection of example programs and auxiliary files, like certificates and keys, for demonstrating the use of FreeOpcUa python realisation of OPC-UA protocol for making queries of TPM data.

The FreeOpcUa web-site: https://github.com/FreeOpcUa

Our proof-of-concept has benefited from the examples given in the FreeOpcUa web-site, and we wish to thank their authors as well as the creators of the FreeOpcUa.

Installation

We use the FreeOpcUa's asyncua version since it is the one under active development. The installation is tested with Debian Linux, but this advice should apply to Ubuntu and other Debian derived distributions, and to other Linuxes with approriate modifications.

First, make sure that you have python3 >= 3.7, IPython3 and pip3. We have used the software packaged by Debian (for IPython the package is python3-ipython and for pip3 python3-pip).

Then, use pip3 to install FreePpcUa opcua-asyncua:

```
$pip3 install asyncua
$pip3 install nest_asyncio
```

The files that are specific to our example are in a .zip file together with this README file.

When unzipped, OPCUA.zip will create a directory "OPCUA" together with subdirectories.

Usage

Launching the server/client: Open a terminal window and Go to the directory where server resides, and issue:

```
$python3 server_with_attest_method.py
```

Then open another terminal window, go to the location of the client and give the command:

```
$python3 client_with_encryption_IPython.py
```

When launched, the client offers an IPython prompt. Using it commands can be given, or loaded from a file with `"%load <filename>"`. See the file `"attest.py"`, intended to be loaded into the above client for requesting attestation from server `"server_with_attest_method.py"`.

The session can be terminated by giving IPython the command "exit". When the client subsequently exits you will get this error message: "TypeError: object NoneType can't be used in 'await' expression" It just means that the client's loop noticed that the IPython disappeared, so this is nothing to worry about.

Keys and Certificates

The required certificates and keys reside in respective client/server (sub)directories. They have been generated by us and are specific to our example. This was done due to some difficulties with the example

certificates provided: a case of expiration and "application_uri mismatches (see https://github.com/FreeOpcUa/opcua-asyncio/issues/430).

If you want to make your own, you need to make sure that the certificate/key and “application_uri” definitions in client & server files match the ones in the certificates.

More exactly, the current appliation_uri definitions are as follows:

In server: “urn:freeopcua:python:server” and in client “urn:freeopcua:client”

The respective field in the certificates is: “subjectAltName”

In principle the standard requires IP-address or DNS-name as well. e.g. like this:

```
"subjectAltName = URI:urn:opcua:python:server,IP: 127.0.0.1"
```

(According to the opcua-asyncio example file “examples/generate_certificate.sh”).

We have provided the commands used for generating our keys/certificates in the file “usefull_commands.txt”.

TODO

“server-with-encryption.py” didn’t allow the writing of a variable with regular user rights, although the code seems to suppose so since it had this line:

```
"await myvar.set_writable()  # Set MyVariable to be writable by clients"
```

For this reason the client was given admin rights in the server.