OSDP Testing with libosdp-conformance

An open source implementation of IEC 60839-11-5 (Open Supervised Device Protocol - OSDP.)

Table of Contents

Quick Start

From user opsadmin1 from a Ubuntu 20 or equivalent Linux platform...

Get it:

```
github clone https://github.com/Security-Industry-Association/libosdp-conformance
```

Set up to build:

```
apt get install build-essential gdb libjansson-dev libgnutls28-dev
git clone https://github.com/kokke/tiny-AES-c
```

Build:

```
cd tiny-AES-c
make
cd ../libosdp-conformance
sudo mkdir -p /opt/osdp-conformance; chown opsadmin1:opsadmin1 /opt/osdp-conformance
./install-aes
make 2>stderr osdp-tls
tar czvf ~/libosdp-conformance.tgz opt
cd /
tar xvf ~/libosdp-conformance.tgz
```

Configure:

```
cd /opt/osdp-conformance/run/ACU
cp open-osdp-params-ACU.json open-osdp-params.json
```

Run (as an ACU):

```
cd /opt/osdp-conformance/run/ACU
sudo /opt/osdp-conformance/bin/open-osdp
```

Similarly for the ACU or the Monitor. See the install details to set up a platform with a web UI.

Introduction

DRAFT documentation. Web UI is being updated to support ***OSDP Verified***.

The libosdp-conformance package implements the OSDP protocol as an ACU, a PD, or in a monitoring configuration. It is written in C for use in a generic Linux/Posix environment such as Ubuntu or Devuan or Raspbian. It was originally built to operate from the linux command line, a simplified HTML/CGI interface was later added. This document describes how to build and operate the package. The main purpose of this package is to provide a conformance test platform for protocol interoperability testing. The monitor is useful by itself. ACU's or PD's can be exercised with the package. It is assumed that a proper Linux serial device driver is available to access a 2-wire RS-485 interface. TCP and TLS are also available in the package.

what's in this document

quick-start for build and run
build instructions

what's in the documentation
- this document
- example osdp configuration files
- example platform set-up fils
- osdp pcap format documentation

whats in the repo
Contents of the package
- src-lib
- include
- src-ui
- test
- doc
  - doc-src
  - doc-pdf
  - spec - iec errata
- src-485
- src-tls
- src-tools

Building from source

This assumes a Linux platform. The description is written for user "opsadmin1", which has sudo. It is assumed there is a 2-wire RS-485 device attached as /dev/ttyUSB0.

Steps to set up libosdp-conformance

A. install these packages to build:

```
      apt install build-essential gdb libjansson-dev libgnutls28-dev apache2
```

B. get the AES library from github

```
      git clone https://github.com/kokke/tiny-AES-c
```

C. get libosdp-conformance from github

```
git clone https://github.com/Security-Industry-Association/libosdp-conformance
```

D. set up environment

```
      sudo mkdir /opt/osdp-conformance
      sudo chown opsadmin1:opsadmin1 /opt/osdp-conformance
```

E. build AES

```
      cd tiny-AES-c
      make
```

F. install AES for use by OSDP

```
      cd libosdp-conformance
      ./install-aes
```

G. build the OSDP code

```
      make 2>stderr osdp-tls
```

H. create the distribution tarball

```
      tar czvf ~/libosdp-conformance.tgz opt
```

I. install the package

```
      cd /
      tar xvf ~/libosdp-conformance.tgz
```

Platform Set-Up

The user interface uses HTML and CGI programs. It is built on apache2. It also uses shell scripts which rely on "sudo -n" and so www-data has been configured with a shell and sudo.

A. install apache2. place the apache config file (in doc/linux-sample/...) in /etc/apache2/sites-enabled. This causes /opt/osdp-conformance/www to be the web server content directory and /opt/osdp-conformance/cgi-bin to be the web applications directory.

- remove the default configuration, replace it with the "osdp.conf" apache config file (see doc/linux-sample.)
- enable CGI processing (add symlinks in /etc/apache2/mods-enabled)

B. Set up user www-data

- change /etc/passwd so that www-data's default shell is /bin/bash
- add a sudo entry to /etc/sudoers.d (see file in doc/linux-sample/.)

Operation

This section describes how to use the package. It can act as a PD or an ACU or a protocol monitor. This assumes you set up using the build procedure so the package is in /opt/osdp-conformance and you have the configuration parameters set (the default is 9600/address 0.)

Using the OSDP Monitor

The osdp server can be used to monitor an OSDP session. The 2-wire RS-485 data connection is capable of supporting an additional device. This is used in monitor mode where it simply listens for connections. No web user interface is used, you need to connect one or more console sessions and "tail" the log.

Set-up

Create a parameter file in the MON directory

```
cd /opt/osdp-conformance/run/MON
cp open-osdp-params-MON.json open-osdp-params.json
```

Starting the Monitor

To start the monitor, start two shells.

In the first shell, strat the monitor:

```
cd /opt/osdp-conformance/run/MON
sudo /opt/osdp-conformance/bin/open-osdp
```

In the second shell, watch the log:

```
cd /opt/osdp-conformance/run/MON
tail -f osdp.log
```

Using the OSDP PD

This is a PD implementation used to exercise ACU's. A log is created, file /opt/osdp-conformance/run/PD/osdp.log. Set up the settings file open-osdp-params.json once and then you can start it from the command line or the web UI.

Set-up

One-time set-up: In the PD directory copy the sample parameter file to open-osdp-params.json. This is set up for 9600, address 0. Edit the JSON file with a text editor if you want to change the start-up settings.

```
cd /opt/osdp-conformance/run/PD
cp open-osdp-params-PD.json open-osdp-params.json
```

Running the PD

From the web UI navigate to http://tester/osdp-conformance-PD.html. Start the PD, then use the PD status to conform there are messages being exchanged.

PD (Web) Interface

The keywords in bold below appear as HTML links.

- **Start** to start the PD
- **Stop** to stop the PD
- Display PD **status**. This displays the JSON status file. acu-polls and pd-acks should be changing.
- Set the log verbosity to normal (**moderate**) or loud (**verbose**.) These correspond to verbosity 3 and verbosity 9 respectively if you want to change the start-up settings.
- You can generate an old-style conformance report ("Generate **report**"), and display it ("Display **report**".) Generating the report creates the new-style test result files for the automatic entries.
- The most recent messages are displayed with "Recent PD **log**".
- Errors in operation appear in the **error** log.
- you can stop the platform with the **stop** command (it delays serveral seconds before poweroff.)

Various tests can be activated. Some tests are listed as "automatic" which means the test results are reported either when the message exchange happens or when a report is generated.

Command Line

From the command line

```
cd /opt/osdp-conformance/run/PD
sudo /opt/osdp-conformance/bin/open-osdp
```

Using the OSDP ACU

This is an ACU implementation used to exercise PD's. A log is created, file /opt/osdp-conformance/run/ACU/osdp.log. Set up the settings file open-osdp-params.json once and then you can start it from the command line or the web UI.

Set-up

One-time set-up: In the ACU directory copy the sample parameter file to open-osdp-params.json. This is set up for 9600, address 0. Edit the JSON file with a text editor if you want to change the start-up settings.

```
cd /opt/osdp-conformance/run/ACU
cp open-osdp-params-ACU.json open-osdp-params.json
```

Running the ACU

From the web UI navigate to http://tester/Test-ACU.html. Start the ACU, then use the ACU status to conform there are messages being exchanged.

ACU (Web) Interface

The keywords in bold below appear as HTML links.

- **Start** to start the ACU
- **Stop** to stop the ACU
- Display ACU **status**. This displays the JSON status file. acu-polls and pd-acks should be changing.
- Set the log verbosity to normal (**moderate**) or loud (**verbose**.) These correspond to verbosity 3 and verbosity 9 respectively if you want to change the start-up settings.
- You can generate an old-style conformance report ("Generate **report**"), and display it ("Display **report**".) Generating the report creates the new-style test result files for the automatic entries.
- The most recent messages are displayed with "Recent ACU **log**".
- Errors in operation appear in the **error** log.

Various tests can be activated. Some tests are listed as "automatic" which means the test results are reported either when the message exchange happens or when a report is generated.

Command Line

From the command line

```
cd /opt/osdp-conformance/run/ACU
sudo /opt/osdp-conformance/bin/open-osdp
```

Conformance Testing

Conformance Instrumentation

There is code to exercise the tests listed in [the test list]. It outputs test results, test by test, to JSON files in /opt/osdp-conformance/results. There are commands to invoke many of the exercises. some are automatic. Not all tests are covered, this is a work in progress. There is an HTML interface (on port 80, unencrypted, no password.) The interface uses HTML pages and CGI programs to run shell scripts that inject actions into the OSDP process to exercise the protocol.

Reporting

output:

osdp.log

osdpcap trace file

other output and collected information.

<json test results file format goes here or in appendix C>

Conformance Exercises

use the tool to run the exercises. html is set up to align with the conformance test list.

OSDP Process

A single process executes the protocol. It acts in one of the three roles. Due to file name use you can only one run per machine at this time. There is a settings file read on startup to configure it. A log file is created, with various levels of detail available. Optionally a raw trace file can be created.

Start-up Settings

The program takes one argument, the name of the settings file. If no name is given it reads from open-osdp-params.json in the current directory.

settings and saved configuration

- Saved (and restored) parameters are in osdp-saved-parameters.json.

osdp-saved-parameters.json is written/read in the current working directory. It is used to load a specified secure channel key. (yes it should save/restore speed and address that's on the to-do list.)

start-up parameters are in appendix b.

Controlling the OSDP Process

Control directives can be passed to the OSDP process. A simplified Unix socket mechanism is used. A parameter file is used to specify a directive and it's parameters.

These are the values allowed for the “command” field in a command json file. Some commands take sub-options (some are mandatory, some are optional.)

acurxsize

[ACU]
Sends osdp_ACURXSIZE to PD, using value from the code (approx 1k.)

bio_read

busy

busy
Causes the PD to respond with BUSY to next incoming command.

buzz

[ACU]
buzz [off_time=xx] [on_time=xx] [repeat=xx]
default
15 15 3

capabilities

[ACU]
capabilities - sends an osdp_CAP to the PD.

comset

[ACU]
comset [new_address=xx] [new_speed=xxx]
Sends a command to request to set the speed and address of the PD. New_address must be 00-7E (hex.) new_speed must be a valid sped (9600,115200,etc.)

conform_x_x_x

[ACU PD]

These are used to induce various conditions for conformance testing. The numbering comes from the old profile documents section headings.

conform_2_2_1
conform_2_2_2
conform_2_2_3
conform_2_2_4
conform_2_6_1

conform_2_11_3
conform_2_14_3
conform_3_14_2
conform_3_20_1

dump_status

factory-default

[ACU PD]
factory-default
removes saved parameters settings (i.e. the preshared key.)

Commands are JSON files. You create a command, copy it to

/opt/osdp-conformance/run/CP/open_osdp_command.json

then “HUP” the process (see do-HUP-CP.)

Example command. There's always a “command”, the other items depend on the specific command.

```
{
  “command” : “xwrite”,
  “action” : “set-mode”,
  “mode” : “1”,
  “#” : “_end”
}
```

genauth

[ACU]
genauth [template=<witness | challenge>] [algoref=<algo>] [payload=zzz]

Template is the challenge operation type (witness crypto or challenge cryto)
algoref is the algorithm used.
payload is the value to be sent as the input payload.

<algo> must be the character strings 07 or 11 or 14 (rsa, ECC P-256, ECC P-384)

identify

[ACU]
identify - sends an osdp_ID to the PD.

induce-NAK

[PD]
induce-NAK - causes the PD to NAK the next incoming message.

initiate-secure-channel

[ACU]

This command to the ACU initates a secure channel session. It sends an osdp_CHLNG. It has one parameter, "key slot", which specifies "0" for SCBK-D or "1" for the selected key. The selected key has to either be defined in the start-up parameter file (parameter "key") or the saved settings file (parameter blah)

options:

    "key-slot" - “0" (for default - SCBK-D) or "1" (for specified key)

example:

```
{ “command”:“initiate-secure-channel”, “key-slot”:“1”}
```

input_status

[ACU]
local_status - send osdp_LSTAT request.

keep-active

keep-active [milliseconds=xx]
Sends osdp_KEEPACTIVE. Default value is 7000 (7 milliseconds.)

keypad

keypad [digits=zzz]
Sends a key input or the (1-9) digits specified. Value for the digits option is 1-9 OSDP keypad values (0-9,*,#)

keyset

[ACU]
keyset [psk-hex=zzzz]
sends an osdp_KEYSET. Default is to send the key specified in the settings. the psk-hex option provides a hex value for the key. Key must be 16 octets (so 32 hexits.)

led

[ACU]
led [led-number=x] [perm-control=x] [perm-off-time=x] [perm-off-color=x] [perm-on-time=x] [perm-on-color=x] [temp-off-color=x] [temp-off=x] [temp-on-color=x] [temp-timer=x] [temp-control=x]

default

    LED 0
    control=set
    off-time=0
    off-color black
    on-color green
    on timer 30
    reader 0
    temp - no operation

local_status

[ACU]
local_status - send osdp_LSTAT request.

mfg

[ACU]
mfg [command-id=xx] [command-specific-data=aaaa] [oui=aabbcc]

command-id is in decimal.
command-specific-data is the payload.
oui is the organizational unit identifier.

operator_confirm

[ACU PD]
operator_confirm test=xxx
confirms test xxx ran successfully.

output

[ACU]
output - outputs via osdp_OUT. The default is output 0, permanent on immediate, forever.

output [output-number=x] [control-code=x] [timer=x]

output_status

[PD]
send osdp_OSTAT to the PD.

polling

[ACU]
polling enables or disables sending poll commands.
"polling" toggles the setting.

"polling action=reset" resumes polling and resets the sequence number to 0.

"polling action=resume" just resumes polling.

present_card

[PD]
present_card [raw=xxx] [bits=n] [format=p-data-p]
sends an osdp_RAW to the ACU.

Options:

- "bits" sets the bits in the message to the specified value.
- "format" sets the format to raw, "format=p-data-p" sets the data to P-data-P.

- "raw" sets the card value to the specified hex string. string must contain enough hex bytes to contain the specified number of bits.

reader_status

[ACU]
reader_status - send osdp_RSTAT to PD

reset

[ACU]
reset - resets sequence number to 0

reset_power

[PD]
reset_power - induces a power reset condition (reset after nex time LSTATR is sent.)

send_poll

[ACU]
send_poll - directs the ACU to send a poll.

stop

[ACU MON PD]
stop - directs the program to stop.

tamper

[PD]
tamper - induces a tamper condition (reset after next time LSTATR is sent.)

text

[ACU]
text message=zzz sends message in osdp_TEXT.

trace

[ACU PD MON]
toggles tracing of OSDP data to ./current.osdpcap.

transfer

[ACU]
transfer [file=fff] - initiates a file transfer. the file zzz is used if no file is specified.

verbosity

[ACU MON PD]
verbosity [level=x] - set message verbosity. By convention this is 0 (none), 3 (normal), or 9 (debug).

Command xwrite

[ACU]

issues an osdp_XWRITE. The action is get mode, scan, set mode, set zero, or done. an optional apdu hex payload may be provided. Experimental.
options:
action : get-mode | scan | set-mode | set-zero | done] [apdu : <hex value>]
action - “set-mode” [mode - “1” or “0”]

example:
```
{
  “command” : “xwrite”,
  “action” : “set-mode”,
  “mode” : “1”,
  “#” : “_end”
}
```

Tools

OSDP Capture - OSDPCAP

Packet Decoder

A cgi for packet decode is provided. opt/osdp-conformance/cgi-bin/osdp-decode takes a single field which is a hex string dump (spaces ok.) A command line tool (osdp-dump) is also available.

```
A command line tool for calculating secure channel values is provided (osdp-sc-
calc.)
```

Appendix

A. Colophon

part of libosdp-conformance, see github.com/security-industry-association/libosd-conformance.

B. Parameter files

Start-up settings - open-osdp-params.json

"address" : "0"

"bits" : "26"

"check" : "CHECKSUM" or "CRC"

"disable_checking" : "1" - nonzero causes it to disable certificate checking with osdp over tls.

`"enable-install" : "1"`

`"enable-secure-channel" : "DEFAULT" or (whatever)`

`"fqdn"`

`"init_command" : "<filename>" - this is the shell script to run at initialization time, should perform STTY...`

`"key" : "<32-hexit string>" or "DEFAULT"`

Saved parameters

parameters in osdp-saved-parameters.json:

`key - value is a 16 byte AES key in hex e.g. "aabbccdd11223344eeff001199887766"`