

Guide to use of libosdp

Project Summary

Purpose:

Implement OSDP protocol

Platform:

Linux (Debian, should be portable)

GCC or CLANG

GnuTLS, OpenSSL, Nettle, libtasn

Jansson

Project Home:

<https://github.com/smithee-us/libosdp>

Table of Contents

Intro.....	3
what it is.....	3
Components.....	3
using the libosdp tools.....	4
Building the Software.....	4
Directory Structure.....	4
Required Libraries.....	4
How to use.....	6
General program set-up.....	6
How to use it – 485.....	7
CP (RS-485).....	7
PD (RS-485).....	7
how to use it – tls.....	7
Certificates.....	7
CP (TLS).....	8
Configuration File Set-up.....	8
Configuring an RS-485 PD.....	8
Configuring an RS-485 CP.....	8
Configuring a TLS PD (Server).....	8
Configuring a TLS CP (Client).....	9
Configuration Parameters.....	9
libosdp internals.....	10
osdp protocol implementation.....	10
libosdp Control API.....	10
appendix.....	11
a. colophon.....	11
b. basic osdp end to end test.....	12
c. osdp interop cable.....	13
D. OSDP Testing.....	14
D.1 present card data.....	14
D.2 LED Testing.....	14
punchlist.....	15

Intro

t.b.d.

what it is

t.b.d. (use wiki page)

Components

The package consists of RS-485 and TLS variants of a program that runs on Linux and implements the OSDP protocol. CP and PD roles are both implemented. Also, for RS-485, there is a “monitor” mode that simply displays the messages read from the RS-485 interface.

RS-485: The program open-osdp implements OSDP using a RS-485 interface that supports the Linux (/dev/tty) interface.

TLS: the programs osdp-net-client and osdp-net-server implement OSDP using TLS over a network connection.

The OSDP programs run as background processes. You send JSON commands to control the program. A simple web-based user interface has been implemented.

The whole thing fits together in a small Linux system (like a Raspberry Pi) using Debian Linux and an Apache web server.

using the libosdp tools

Building the Software

Get the latest release from the github repo. Load it on a Debian linux system, preferably with CLANG but GCC will work. Do “make clean” and “make build” in the top level directory to build the software. Do “make release” in the top level directory to create a tarball meant to be expanded from the root (as root) so as to create a /opt/open-osdp/... file structure. There isn't an install package, it's assumed you create the tarball and load it yourself.

Directory Structure

The built directory structure is

```
/opt/open-osdp/bin
/opt/open-osdp/etc
/opt/open-osdp/run/CP
/opt/open-osdp/run/PD
```

bin - programs osdp-net-client, osdp-net-server

etc - certificate files ca_keys.pem (for root key), key.pem (private key for server), cert.pem (certificate for server), client_key.pem (private key for client), client_cert.pem (cert for client.)

run/CP - directory where CP is run

run/PD - directory where PD is run

run/MON - directory where Monitoring is run

Required Libraries

Update Debian and add build-essential and some other packages.

```
apt-get update
apt-get upgrade
apt-get install build-essential
```

```
apt-get install clang lzip pkg-config libgmp3-dev libgmpxx4ldbl screen
apt-get install tcpdump gdb apache2 git
```

download jansson, libtasn, nettle, and gnutls.

```
build jansson
./configure --prefix=/opt
make
sudo make install
```

```
build tasn
./configure --prefix=/opt
make
sudo make install
```

```
build nettle
PKG_CONFIG_PATH=/opt/lib/pkgconfig ./configure --prefix=/opt
make
sudo make install
```

build gnutls

```
PKG_CONFIG_PATH=/opt/lib/pkgconfig/ ./configure --prefix=/opt --without-p11-kit  
make  
sudo make install
```

building libosdp

```
cd setup/libosdp  
make clean  
make CFLAGS="-I/opt/include" LDFLAGS="-L/opt/lib" release  
cd /  
sudo tar xzvf /home/osdp/setup/libosdp/release-libosdp.tgz
```

How to use

General program set-up

it writes to `open_ospd.log` in the current directory. it reads its configuration from `open-osdp-params.json` in the current directory. The build process is set up to prepare `/opt/open-osdp/run/CP` (and `PD` and `MONITOR`) as the directory to run in. The tools read configuration and command files from the current directory and write the log to the current directory.

How to use it – 485

CP (RS-485)

Check your RS-485 adapter and cable hardware.

cd to /opt/open-osdp/run/CP

set up a configuration file for CP use. Use the sample config files in the doc directoy as a guide. Be sure to set the PD address to match your PD.

Run /opt/open-osdp/bin/open-osdp

An example configuration is in doc/config-samples/ open-osdp-params-CP-485.json

Example. This runs the CP from /opt/open-ospd/run/CP. The parameter file has been set up there (open-ospd-params.json) After you start the program you can go to another command shell and "tail" the log file.

```
cd /opt/open-osdp/run/CP
ls -l open-osdp-params.json
sudo /opt/open-osdp/bin/open-osdp
```

(in a separate window)

```
cd /opt/open-ospd/run/CP
sudo tail -f ./open_osdp.log
```

PD (RS-485)

Check your RS-485 adapter and cable hardware.

cd to /opt/open-osdp/run/PD

set up a configuration file for PD use. Use the sample config files in the doc directoy as a guide. Be sure to set the PD address to match your CP.

Run /opt/open-osdp/bin/open-osdp

An example configuration is in doc/config-samples/ open-osdp-params-PD-485.json

how to use it – tls

osdp-net-server and osdp-net-client are the TLS server and client respectively. Each one runs the open-osdp code. Configuration details are read from open-osdp-params.json.

Certificates go in /opt/open-osdp/etc. For the server side it uses ca_keys.pem, key.pem, cert.pem. the key is not encrypted. For the client side it uses ca_keys.pem, client_key.pem, clietn_cert.pem. The key is not encrypted.

Certificates

Both sides need certificates and keys. Each side has to be configured to trust the root that issued the other end's cert. For the test program, the Common Name field of the Subject Name must be set in the

config file as GnuTLS checks this.

set up test certificates

```
cd setup/test-ca
./0-init-ca
./1-create-certs

copy the appropriate certs to etc
cp root.pem osdp-pd-1_cert.pem osdp-pd-1_key.pem /opt/open-osdp/etc
cd /opt/open-osdp/etc
cp root.pem ca_certs.pem
cp osdp-pd-1_cert.pem cert.pem
cp osdp-pd-1_key.pem key.pem
```

CP (TLS)

Check your certificates, ip addresses, and port settings to match the PD.

cd to /opt/open-osdp/run/CP

set up a configuration file for CP use. Use the sample config files in the doc directoy as a guide. Be sure to set the PD address to match your CP.

To connect to a PD listening for an incoming connection (CP is the client, PD is the server.) There has to be something listening at the destination address or else the program fails.

```
run /opt/open-osdp/bin/osdp-net-client
```

To listen for TLS connections (PD is the client, CP is the server)

```
run /opt/open-osdp/bin/osdp-net-server
```

An example configuration is in doc/config-samples/ open-osdp-params-CP-TLS-Client.json

Configuration File Set-up

You set the OSDP role in the configuration file (open-osdp-params.json.) This is independent of whether it's RS-485 or which end of a TLS connection you configure. In the networking cases either the PD or the CP can be listening for a network connection from the other end.

Configuring an RS-485 PD

Configuring an RS-485 CP

"role" is "CP"

"serial_device" is "/dev/tty/USB0" (most likely, check this.)

"addr" is "02" (the address of the PD, must match PD address.)

Configuring a TLS PD (Server)

set it to be the PD

set the verbosity

set the fqdn to the CN field of the other end's certificate.

set the test card data

Configuring a TLS CP (Client)

set it to be the CP

set the verbosity

set the fqdn to the CN field of the other end's certificate.

set the ip address of the server

Configuration Parameters

These are specified in open-osdp-param.json.

address - PD address to use or to talk to. must be valid OSDP address value. Value is in hex.

bits - number of bits in RAW response. Value is in decimal.

disable_checking - disable certificate checking ("1") or not ("0")

fqdn - DN field of peer's certificate if cert checking is turned on.

init_command - command to initialize serial device.

network_address - ipv4 address to use to connect (for TLS or TCP client.)

poll -- poll frequency (CP polling the PD.)

raw_value = value is hex, this is the card data. Note for 26 bit it is left justified (bottom 6 bits of last octet are not used.)

role - CP PD or MON

serial_device -- name of serial device for RS-485. Typically /dev/ttyUSB0

slow_timer -- in TLS (or TCP) causes the CP to wake up if PD traffic arrives before the poll interval.

timeout

verbosity -- logging verbosity. 1-3 are normal, 9 is loud, >9 is very loud.

libosdp internals

osdp protocol implementation

t.b.d.

libosdp Control API

The main loop listens on a Unix socket (“open-osdp-control” in the current directory) for a “kick” (in the style of the POSIX “HUP” signal.) When it gets kicked it reads open_osdp_command.json and processes that command.

The format is

```
{  
  “command” : “<action>”  
}
```

Where <action> is one of

- capabilities
- dump_status
- identify
- led
- output
- present_card
- reset_power
- send_poll

Note some of these have additional arguments.

command led
value (led color)

appendix

a. colophon

Revised for x.x build x (post isc west 2016 interopfest)

Written in OpenOffice and LibreOffice. The HACK font is available from <http://sourcefoundry.org/hack/>.

©2014-2016 Smithee, Spelvin, Agnew & Plinge, Inc.

Work funded in part by securityindustry.org

b. basic osdp end to end test

t.b.d.

c. osdp interop cable

t.b.d.

D. OSDP Testing

D.1 present card data

open-osdp as the PD

DUT as the CP

Create an appropriate command file. Copy it to /opt/open-osdp/run/PD. From the PD directory do

```
/opt/open-osdp/bin/write-osdp-PD-command present_card  
/opt/open-osdp/bin/HUP-PD
```

D.2 LED Testing

open-osdp as the PD

DUT as the PD

Set the card data value in the osdp parameter file (open-osdp-params.json) and start open-osdp. Use the "102-led" command to set the LED color. "102-led" uses the "led" command, which sets some LED parameters to constant values (timers) and sets the LED color. doc/command-samples/example-led_open_osdp_command.json is a sample of the command file.

Example: Red LED

```
cd /opt/open-osdp/run/CP  
/opt/open-osdp/bin/102-led 1
```

Example: Green LED

```
cd /opt/open-osdp/run/CP  
/opt/open-osdp/bin/102-led 1
```

punchlist

fill this in and publish
pull wiki
pull from other sources

config samples fix ref make separate samples
cp 485
pd 485
cp tls client
pd tls server