Introduction

The purpose of this project is to demonstrate some of the One Platform Kit (onePK) abilities through an implementation a data path service set (dpss) functionality. In order to bring this project to life, both C programming skills on Unix like platforms and networking skills are required.

The program is able to track TCP packets and show some information out of them, for example, L4 protocol, source and destination addresses with ports, and other data like packet flow number. What is more, the program can track how many packets had already went through a device under test, which is router in our case, drop as many packets as configured in the program (drop rate) and saves the dropped packets information to a file called «dropped_packets_log.txt». The third implemented feature is an ability to change a port that is listened by the program on the router (those are, for example, fa0/1 or se0/0/0 interface). What is more, there is a counter that tracks a time difference between packets and shows mean, maximum and minimum times.

Program basic user guide

When the program starts, it shows some information regarding it's usage and prompts for credentials that are configured on the networking device.
There are three buttons available - I, D and E. I - for changing interface, D - for changing drop rate and E - for exiting.

Press, for example, I and then Enter button to be prompted with a change interface menu.

Program run requirements

In order to execute the program, DPSS should be installed to Linux PC. By default, It is installed when Cisco's OnePK SDK is installed. OnePK is available from the following link - https://developer.cisco.com/site/onepk/
But you can try the application with the virtual machine provided along with this document. See the topic in the end of this documentation.

The source code is available from this link:
https://drive.google.com/file/d/0B0yyqwXFTq7leHZjaWdZQU1lbFE/view?usp=sharing

Virtual machine is available from this link:
https://drive.google.com/file/d/0B0yyqwXFTq7lcWVCc0dkS0c2Ykk/view?usp=sharing

Installing the C SDK

If the SDK installer detects another instance of the SDK in the target installation directory, it gives you the option to install the new SDK as a sibling (leaving the old SDK untouched), or to install the SDK in a completely different directory.

1. Download either the 32- or 64-bit SDK. Use the uname command if you don't know which architecture your workstation is using.
2. uname -p
   x86_64
3. Unpack the .tar file.
4. # this example unpacks the 64-bit C SDK
5. tar -xvf onePK-sdk-c-rel-*SDK_version_num*-lnx-x86_64.tar
6. Change to the directory created by extracting the tar file and then run the installer file.
7. cd /home/cisco/onePK-sdk-c-*SDK_version_num*-lnx-x86_64
8. ./install.sh
9. Follow the prompts to accept the license agreement and install the SDK files.
   After you accept the license agreement, the following prompt appears:

   Install Cisco's onePK C SDK in (/opt/cisco/onep/c64)?
   Choose [n] to select a different directory (y/n) [y]
10. Respond to the prompt to specify the location at which to install the SDK:
    - **If you enter [y]**, the C SDK is installed in /opt/cisco/onep/c32 or (./c64). If a previous version of the SDK is present in this location, the installer places the new SDK next to the old one without overwriting the old SDK.
      You need root or sudo privileges to install in the /opt directory path. If you do not have these privileges, enter [n] and specify a different location when prompted to do so.
    - **If you enter [n]**, you are prompted to supply the path to a different location, and the installer places the new SDK at that location.
11. Optionally, you can delete the old SDK from /opt/cisco/onep/c32 or (./c64) if you don’t need it anymore.
12. Set LD_LIBRARY_PATH to point to the directory that contains the newly installed onePK libraries. (In the Virtual Machine, you can update /etc/bash.bashrc to set this value permanently.)
13. setenv LD_LIBRARY_PATH "${ONEP_SDK}/c/lib"
    or
    export LD_LIBRARY_PATH="${ONEP_SDK}/c/lib"
14. Set ONEP_SDK to point to the same directory as the previous step. (In the Virtual Machine, you can update /etc/bash.bashrc to set this value permanently.)
15. # for 32-bit SDK change occurrences of c64 to c32

16. setenv ONEP_SDK '/opt/cisco/onep/c64/*sdk-c64-VERSION*'
    or
    # for 32-bit SDK change occurrences of c64 to c32

export ONEP_SDK='/opt/cisco/onep/c64/*sdk-c64-VERSION*'

17. Set LBITS to represent the architecture of the SDK you just installed. (In the Virtual Machine, you can update /etc/bash.bashrc to set this value permanently.)
18. # for 32-bit SDK use the value 32 instead of 64
19. setenv LBITS 64
or
# for 32-bit SDK use the value 32 instead of 64
export LBITS=64
20. Locate the default dpss.conf file:
    - 64-bit SDK
      /opt/cisco/onep/c64/sdk-c64-<*SDK_version_num*>/c/bin
    - 32-bit SDK
      /opt/cisco/onep/c32/sdk-c32-<*SDK_version_num*>/c/bin
21. Update fields in the dpss.conf file to reflect the values that your network element uses, and then save the file in place.

FIeld Name	**Value**	**Description**
LOCAL_IP	10.10.10.1	Local ipv4 address on which the DPSS application listens for packets.
GROUP_NAME	cisco	Linux group to which the onePK application developer belongs.
USER_NAME	onepk	After starting as root, the dpss_main process runs as this user for lower-privileged tasks.
TRANSPORT	raw	Always use this value.

DPSS

The Data Path service set (DPSS) allows an application developer to hook in to the packet flow through a Cisco switch or router and extract packets from that flow of packets. With this service set, your application can perform the following operation on packets:

• **Punt**--The packet is sent to the application and does not continue on its way until the packet is returned to the data path by the application. While the application is examining the packet and deciding what to do, no other application, including the host platform itself, can do anything with the packet. The application can modify the packet in any way it sees fit, including dropping it.

• **Copy**-- A copy of the packet is sent to the application. The packet is also sent to its original destination. The application can examine the packet's contents, but it cannot affect the packet.

• **Sample**--This is the same as a copy. The difference is that packets are sent only occasionally, according to the sampling rules specified by the application.

You should run DPSS like this, otherwise the program will not run -
sudo /opt/cisco/onep/c64/sdk-c64-1.3.0.181/c/bin/dpss_mp -c /opt/cisco/onep/c64/sdk-c64-1.3.0.181/c/bin/dpss.conf --fg &

where dk-c64-1.3.0.181 is the version of the sdk.

DPSS requires some preconfigurations that are stored to dpss.conf file.
LOCAL_IP: Set this value to the IP address of Ethernet interface that is the endpoint to the router.
GROUP_NAME and USER_NAME : Set this to the username and group as appropriate on your Linux installation. If you run into permission failure starting DPSS, some operations in your NOS configuration may require root privilege.
TRANSPORT : IOS/IOS-XE support GRE transport only. Transport must be set to "raw".
NOTE: Ensure that the user and group names are correct for your environment, else the dpss_mp will fail to load due to permissions issues. Also ensure you have access to a /tmp directory.

Router configuration.

1) Physically connect a router and a host-PC
2) Assign an IP addresses and verify connectivity with a ping command
3) Generate new certificate to access our router via TLS

It can be done like this -
./.simpleCA/createNEp12.sh -cn Router -ip 192.168.20.2 -out Router.p12 -pass cisco1
where *192.168.20.2* is the address of router and *cisco1* is your password
./.simpleCA is located in your home directory (cd ~).

4) Install tftp to copy the certificate inside the router. And then configure tftp:

nano /etc/default/tftpd-hpa

```
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/tftpboot"
TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="--secure --create"
```

Probably you must change TFTP_ADDRESS to 0.0.0.0
Restart tftp -
sudo service tftpd-hpa restart
copy certificate into TFTP_DIRECTORY, in my case /tftpboot
sudo cp Router.p12 /tftpboot/

5) Configure router

On router:

1. en
2. conf t
3. crypto pki import myTP pkcs12 tftp://192.168.20.1/Router.p12 password cisco1

#192.168.20.1 is IP of host-PC interface, password cisco1 is same as in process of generation new certification.

Enable onepk on router side (onepk 1.2.0)

1. en
2. conf t
3. onep
4. transport type tls localcert myTP disable-remotecert-validationt

Create user on router

1. username onepk passwork cisco
2. username onepk privilege 15

Next create a certificate to authenticate yourself when connecting to the router

1. en
2. conf t
3. crypto pki myTP export pem terminal

You will get the following information

```
% CA certificate:
-----BEGIN CERTIFICATE-----
MIIC7zCCAdegAwIBAgIBATANBgkqhkiG9w0BAQUFADAZMRcwFQYDVQQDEw5vbmVQ
SyBzaW1wbGVDQTAeFw0xMjEyMTAwMDAwMDBaFw0zMDEyMTAwMDAwMDBaMFzA
      V
BgNVBAMTDm9uZVBLIHNpbXBsZUNBMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIB
CgKCAQEAtn1BH3e69SClUwAeflMBVlTB6ZbUXlU/YugDPVictxs+w5BB/TGQEcd7
hR602SnTTIRYVLCkGPrFkE4gAkWanObEZT9UlIdSoZzxudP9n97a8KnY15rcYQ9p
sqWNQ/FwwA2RvG2BxNcaAgX3oiR+AXggV0Lswi/iV0+jwfo24sqG54bv5223YSXd
d4cOgTkHCjbaxEsjCCyURvfozRinIm8hlOtXCGxOLyoRVqJg+kGCJaia5b+f0j+Z
qjhtH+1LVLyjj3P5RX0Zx8cEF4DkThP8Xp+a/dwNW64u80r/waCX3czg22O3eu+s
C2f27tq++VtNM536lF60GBGYw/iO+QIDAQABo0IwQDAPBgNVHRMBAf8EBTADAQH/
MB0GA1UdDgQWBBQZ6QnJkIzpWa6fSX0o2WAJCBleUDAOBgNVHQ8BAf8EBAMCAw
DQYJKoZIhvcNAQEFBQADggEBAFe2CIc23aHbzO4Awq47pNz5orekh7Ww1U7wtFHi
bOayk0gEh+llfBmeyWcR2v1Q2uqL5XWWZdYwsEAZDNMqZdsN8hOe5xurvEqfplLI
iYt81QhxF2Heb+WcjdVYVE7hQ43IepG3zB6ZH3gLOcg5CXVex0VJLIQjMMKCdBpT
PUpG5rl7Pj1NsRo0GgjoNMEk2EO7A3fEdE7iS7qKaK4OlLJbPiuoCkYpI4TfnWd1
mnACtF/jKyAaIRPivRIRiVHES3uXW70JGkB4fYf0pa14qkpc2RqtIFAVhtA6D8L9
1mw51jzCysRHuZtFQ1LhEzFOl+w5Z71mRYu84p7cVEJudAM=
-----END CERTIFICATE——
```

Copy the key including both ———BEGIN CERTIFICATE—— and ——END CERTIFICATE—— strings.
Create a file called CA.pem in your home directoryand paste
the key copied to that file and save

6) Now you can connect to the router from application

cd ~/InnovationProject
sudo ./bin/InnovationProject -a 192.168.20.2 -i gi0/2 -R ~/CA.pem

where CA.pem is the certificate that we made using the key inside the router.
and gi0/2 is the interface that we want to listen to.

Next you will be prompted for the username and password that we created in the router.

Now the program is ready to be used.

What is more, it is possible to use the program with GNS3.
Virtual machine with installed SDK, GNS3 and other tools is also available along with this document. All the configuration steps for real devices are applicable for the virtual ones as well.

Program compilation and source code

In order to compile the program, you should invoke **make** command in the main folder of the program. In our case it is in **cd ~/InnovationProject**
The program has the following functions:

```
/*
 * Extract the IP version from a packet.
 */
onep_status_t get_ip_version(struct onep_dpss_paktype_ *pakp, char *ip_version);

/*
 * Method for extracting IP address and
 * Getting port information from the packet.
 */
onep_status_t get_ip_port_info(struct onep_dpss_paktype_ *pakp, char **src_ip, char **dest_ip,
uint16_t *src_port, uint16_t *dest_port, char *prot, char ip_version );

/*
 * Packet callback method that shows packet information
 * And drops packets according to packet drop rate.
 */
void dpss_display_pak_info_callback(onep_dpss_traffic_reg_t *reg,
    struct onep_dpss_paktype_ *pak, void *client_context, bool *return_packet);

/*
 * Get interfaces available on device
 */
void get_interface_filter(onep_interface_filter_t* intf_filter);

/*
 * Calculate time difference between packets
 */
void timespec_diff(struct timespec *start, struct timespec *stop,
                   struct timespec *result);

/*
 * Display a list of interfaces.
 */
int display_intf_list(FILE *op, int get_name, int intfs_num, onep_if_name* func_name);

/*
 *   Get traffic action table
 */
onep_status_t find_datapath_table(onep_network_element_t *elem, onep_policy_table_cap_t
**table_cap);
```

```
/*
 * Example function to create a simple ACL and Policy Map
 */
onep_status_t create_ip_pmap (
    onep_network_element_t *elem,
    onep_dpss_pak_callback_t callback,
    onep_dpss_pkt_action_type_e action,
    onep_policy_pmap_handle_t *pmap_handle,
    onep_policy_pmap_op_t *pmap_op,
    onep_policy_op_list_t *pmap_op_list,
    onep_policy_op_list_t *cmap_op_list,
    onep_policy_cmap_handle_t *cmap_handle,
    onep_policy_cmap_op_t *cmap_op,
    onep_acl_t ** acl);

/*
 * Thread function that runs the main logic of the program in while loop
 */
void *thread_program_run(void *p);

/*
 * Create thread method
 */
void start_thread(pthread_t *pthr, onep_if_name *intf_name);

/*
 * Main application method that tracks for buttons press to configure the app in
 * Runtime
 */
int main (int argc, char* argv[]);
```

First main method executes a function to prompt a user for a username and password.
Then it creates a thread that starts executing the main logic - it creates an ACL rule to accept all the packets, initializes DPSS to copy all the packets from the router to an application and sets callback method that processes packets (it tracks the drop rate and drops as many packets as configured). What is more, the callback method has a logic to count timing between packets and it shows packets' information like IP src and dst address, IP version, dropped or not packets, counting and etc. Furthermore, the callback method saves information of dropped packets to a file.
While executing all the mentioned things, the thread is running in forever loop until it receives a command that some user pushed some menu buttons (I, D or E). Then it closes the connection, deletes ACL and other configs on the router and the program will open the desired menu. When the menu is closed, the program will run from the very beginning, but with other configs from the menu.

More comments on the source code are available from the source file.

Virtual machine run

There is a ready-made virtual machine with preinstalled tools available for usage.
It has .ova extension and can be imported to a *VMware Fusion/Workstation, VirtualBox* or whatever. After successful import, just run the virtual machine called ***OnePK Ubuntu*** and wait for a boot. When it prompts for the password, type **onepk** (*username and password are both **onepk***).

You can start working after login. SDK is installed and all the environmental paths are set.

Just open GNS3, which can be found on the left side launcher, then open ready-made **Innovation_project** GNS3 project. You will see 3 routers called OnePK-130-*, one switch and a cloud that connects our virtual PC that hosts an application to a router. The cloud uses tap0 interface for communication between the middle router and the PC.

You can create own interface using the following commands if needed –

```
sudo tunctl -u cisco -t tap0
sudo ifconfig tap0 inet 192.168.20.1/24 promisc up
```

Then you should start all the three routers and wait until they boot up and establish a connection between each other. Check the IP addresses of the interfaces of the most left and right routers and try to ping them. If succeeded, you can try to run the onePK application.

Firstly, start DPSS

1. Execute **sudo** and then enter password (onepk)
2. Execute **sudo /opt/cisco/onep/c64/sdk-c64-1.3.0.181/c/bin/dpss_mp -c /opt/cisco/onep/c64/sdk-c64-1.3.0.181/c/bin/dpss.conf --fg &**

Secondly, go to the main folder of the application and run the program

1. **cd ~/InnovationProject**
2. **sudo ./bin/InnovationProject -a 192.168.20.2 -i gi0/2 -R ~/CA.pem**
3. Enter sudo password if prompted: **onepk**
4. Enter username: **onepk**
5. Enter password: **onepk**
6. You will see the message that gi0/2 interface is being listened to
7. Try to ping from the left most router to the right most router and see packets going on in the application
8. Try to play with the menu buttons (I – change interface, D – change drop rate, E – exit). **Menu buttons are case sensitive!**