NVIDIA DOCA Security Gateway

Application Guide

MLNX-15-060576 _v1.5.2 | July 2023

Table of Contents

Chapter 1. Introduction

DOCA Security Gateway leverages the DPU's hardware capability for secure network communication. The application demonstrates how to insert rules related to IPsec encryption and decryption based on the DOCA Flow and IPsec libraries.

The application gets a list of rules for IPsec encryption and decryption as an input which are then translated to the respective IPsec tunnel mode and ESP header definitions.

The application supports the following modes:

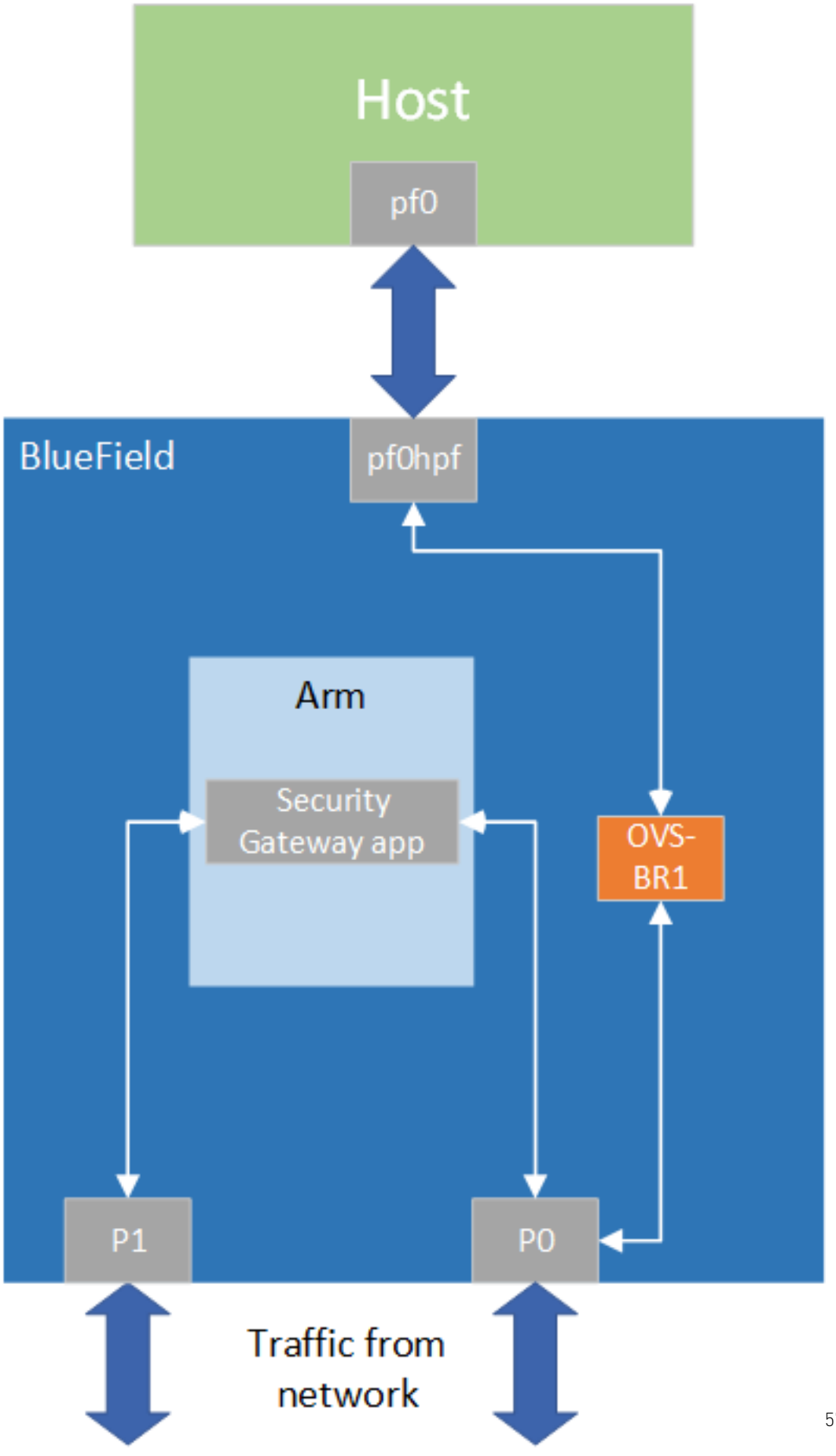
- Full offload – packets are processed in hardware and hairpinned to the next port
- Partial offload – packets are processed in hardware and passed onwards to the application before they are forwarded to the next port

Note: When creating the SA object, the application uses a fixed IPsec key to serve as an example. Make sure to replace it with your own implementation when using the application for more than testing.

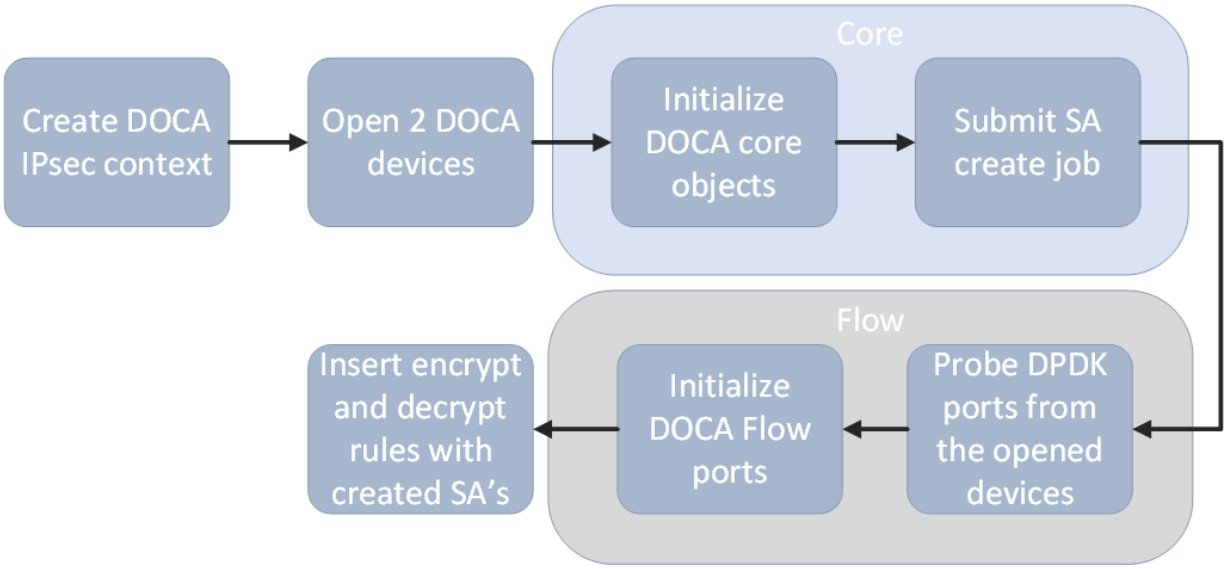
Chapter 2. System Design

DOCA Security Gateway is designed to run with 2 ports, secured and unsecured:

- Secured port – the application receives encrypted packets and, after decryption, they are sent through the unsecured port
- Unsecured port – the application receives regular (plain text) packets and, after encryption, they are sent through the secured port


Host
pf0
BlueField
pf0hpf
Arm
Security Gateway app
OVS-BR1
P1
P0
Traffic from network

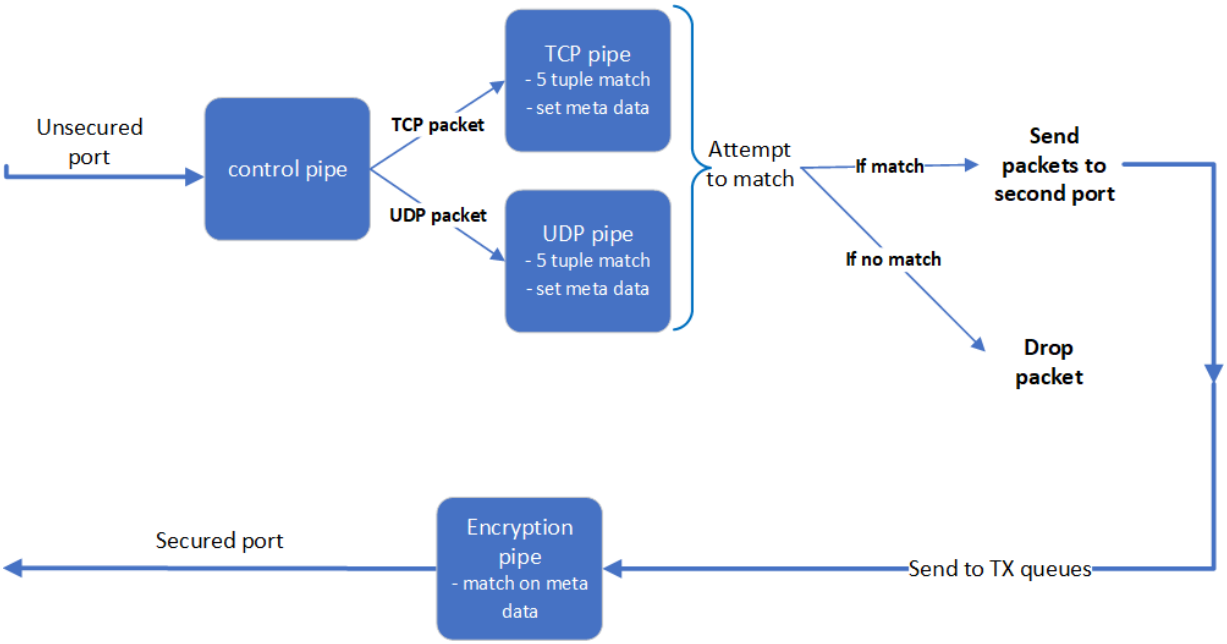
Chapter 3. Application Architecture



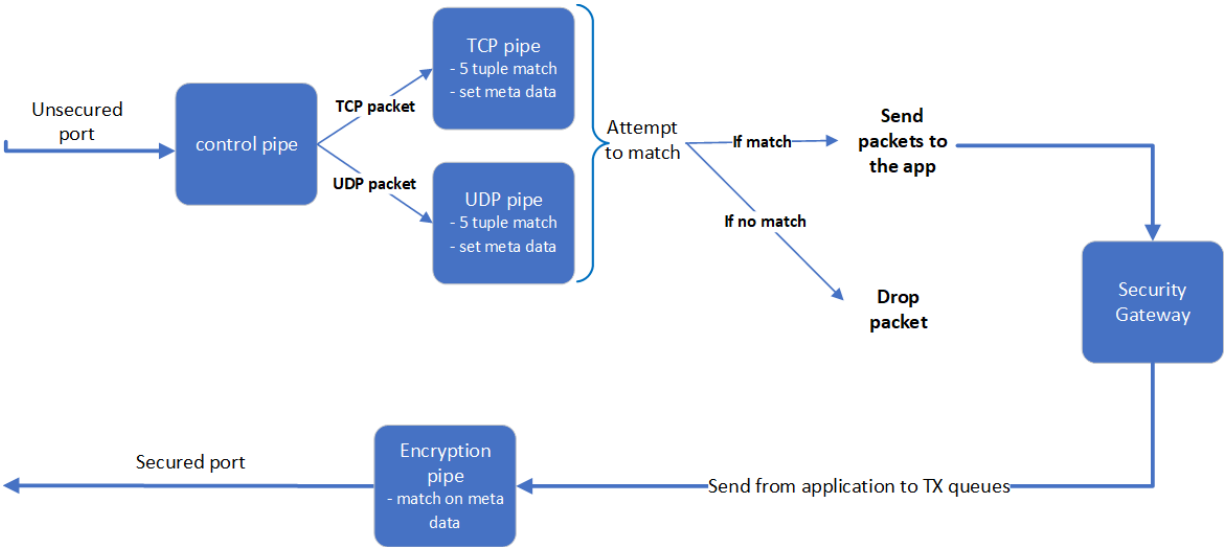
1. Crete IPsec library context.
2. Open two DOCA devices, one for the secured port and another for the unsecured port.
3. Initialize the DOCA work queue.
4. Submit "Create SA" jobs, one job for encryption and another for decryption.
5. With the open DOCA devices, the application probes DPDK ports and initializes DOCA Flow and DOCA Flow ports accordingly.
6. On the created ports, you build and insert DOCA Flow pipes and entries according to the input JSON rules and the created SA objects.

3.1. Encryption

Full offload flow:

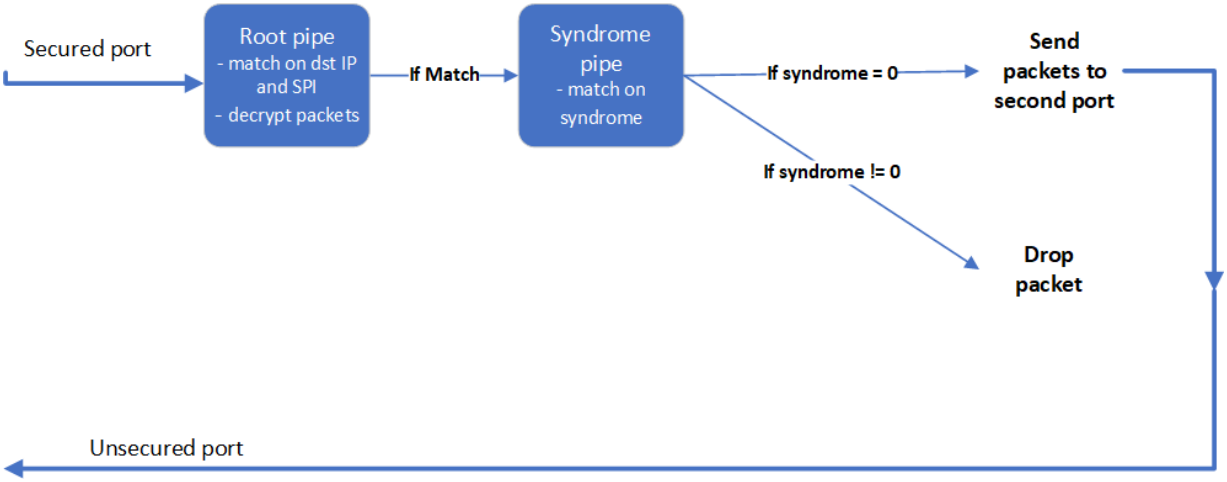


Partial offload flow:

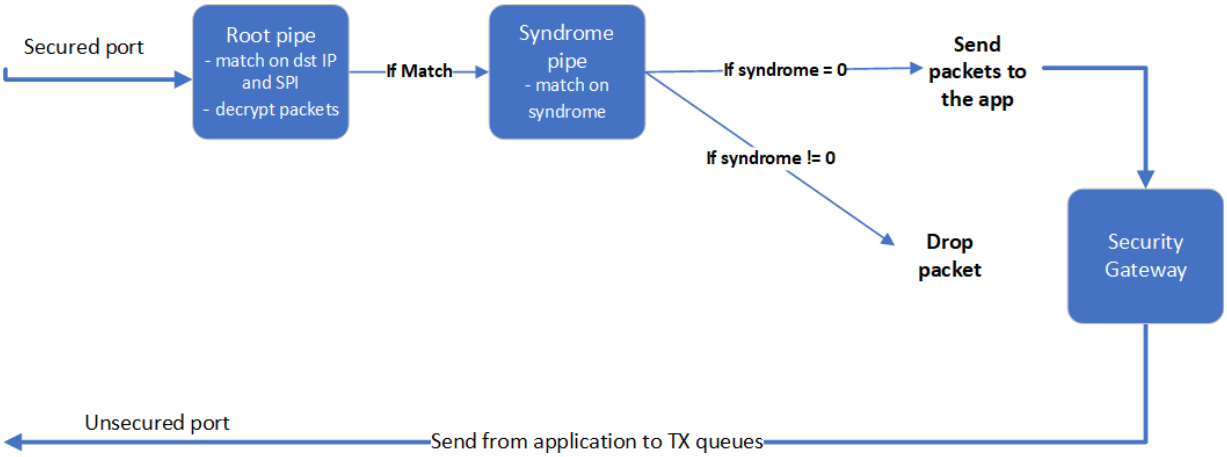


3.2. Decryption

Full offload flow:



Partial offload flow:



Chapter 4. DOCA Libraries

This application leverages the following DOCA libraries:

- DOCA Flow library
- DOCA IPsec library

Chapter 5. Configuration Flow

1. Parse application argument.
   a). Initialize the arg parser resources and register DOCA general parameters.
   ```
   doca_argp_init();
   ```
   b). Register application parameters.
   ```
   register_security_gateway_params();
   ```
   c). Parse application flags.
   ```
   doca_argp_start();
   ```
      i. Parse app parameters.
2. Parse rules file.
   ```
   security_gateway_parse_rules();
   ```
3. DPDK initialization.
   ```
   rte_eal_init();
   ```
   Call `rte_eal_init()` to initialize EAL resources with the provided EAL flags for not probing the ports.
4. Initialize devices and ports.
   ```
   security_gateway_init_devices();
   ```
   a). Open DOCA devices with input PCIe addresses.
   b). Probe DPDK port from each opened device.
5. Initialize and start DPDK ports.
   ```
   dpdk_queues_and_ports_init();
   ```
   a). Initialize DPDK ports, including mempool allocation.
   b). Initialize hairpin queues if needed.
   c). Binds hairpin queues of each port to its peer port.
6. Create SA object for encryption and decryption.
   ```
   security_gateway_create_ipsec_sa();
   ```
   a). Create IPsec library context.
   b). Create DOCA Work queue.
   c). Submit create SA job.
7. Initialize DOCA Flow.
   ```
   security_gateway_init_doca_flow();
   ```
   a). Initialize DOCA Flow library.

b). Find the indices of the DPDK-probed ports and start DOCA Flow ports with them.

8. Insert rules.

   a). Insert encryption rules.
   ```
   security_gateway_insert_encrypt_rules();
   ```
   b). Insert decryption rules.
   ```
   security_gateway_insert_decrypt_rules();
   ```

9. Wait for traffic.
   ```
   security_gateway_wait_for_traffic();
   ```
   a). In full offload mode, wait in a loop until the user terminates the program.

   b). In partial offload, receive packets on all cores and send the received packets to second port.

10. Security gateway cleanup:

    a). DOCA Flow cleanup; destroy initialized ports.
    ```
    doca_flow_cleanup();
    ```
    b). Destroy DPDK ports and queues.
    ```
    dpdk_queues_and_ports_fini();
    ```
    c). DPDK finish.
    ```
    dpdk_fini();
    ```
    Calls `rte_eal_destroy()` to destroy initialized EAL resources.

    d). Arg parser destroy.
    ```
    doca_argp_destroy();
    ```

Chapter 6. Running the Application

1. Refer to the following documents:
   - NVIDIA DOCA Installation Guide for Linux for details on how to install BlueField-related software.
   - NVIDIA DOCA Troubleshooting Guide for any issue you may encounter with the installation, compilation, or execution of DOCA applications.
   - NVIDIA DOCA Applications Overview for additional compilation instructions and development tips for the DOCA applications.
2. DOCA Security Gateway binary is located under `/opt/mellanox/doca/applications/security_gateway/bin/doca_security_gateway`. To build all the applications together, run:

```
cd /opt/mellanox/doca/applications/
meson build
ninja -C build
```

3. To build only the security gateway application:

   a). Edit the following flags in `/opt/mellanox/doca/applications/meson_option.txt`:
   - Set `enable_all_applications` to `false`
   - Set `enable_security_gateway` to `true`

   b). Run the commands in step 2.

> **Note:** `doca_security_gateway` will be created under `./build/switch/src/`.

Application usage:

```
Usage: doca_security_gateway [DOCA Flags] [Program Flags]
DOCA Flags:
  -h, --help                        Print a help synopsis
  -v, --version                     Print program version information
  -l, --log-level                   Set the log level for the program
 <CRITICAL=20, ERROR=30, WARNING=40, INFO=50, DEBUG=60>
 Program Flags:
  -s, --secured                     secured port pci-address
  -u, --unsecured                   unsecured port pci-address
  -r, --rules                       Path to the JSON file with 5-tuple rules
  -o, --offload                     offload mode - {partial/full}
```

> **Note:** For additional information on the app, use `-h`:
>
> ```
> /opt/mellanox/doca/applications/<application name>/bin/doca_<application
>  name> -h
> ```

4. Running the application on BlueField:
   - Pre-run setup:
     a). The security gateway example is based on DPDK libraries. Therefore, the user is required to allocate huge pages.
     ```
     echo 2048 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
     ```
     b). The security gateway example requires disabling some of the hardware tables which can be done using the following commands:
     ```
     echo none > /sys/class/net/p0/compat/devlink/encap
     echo none > /sys/class/net/p1/compat/devlink/encap
     ```
   - CLI example for running the application:
     ```
     /opt/mellanox/doca/applications/security_gateway/bin/doca_security_gateway
      -s 03:00.0 -u 03:00.1 -r applications/security_gateway/
     security_gateway_rules.json -o full
     ```
5. Running the application on the host, CLI example:
   ```
   /opt/mellanox/doca/applications/security_gateway/bin/doca_security_gateway -s
    08:00.0 -u 08:00.1 -r applications/security_gateway/security_gateway_rules.json
    -o full
   ```

   Note: Refer to section "Running DOCA Application on Host" in NVIDIA DOCA Virtual Functions User Guide.

6. To run `doca_security_gateway` using a JSON file:
   ```
   doca_security_gateway --json [json_file]
   ```
   For example:
   ```
   cd /opt/mellanox/doca/applications/security_gateway/bin
   ./doca_security_gateway --json ./security_gateway_params.json
   ```

Chapter 7. Arg Parser DOCA Flags

Refer to NVIDIA DOCA Arg Parser User Guide for more information.

Flag Type	Short Flag	Long Flag/JSON Key	Description	JSON Content
General flags	`l`	`log-level`	Sets the log level for the application: ▶ CRITICAL=20 ▶ ERROR=30 ▶ WARNING=40 ▶ INFO=50 ▶ DEBUG=60	`"log-level": 60`
	`v`	`version`	Print program version information	N/A
	`h`	`help`	Print a help synopsis	N/A
Program flags	`r`	`rules`	Path to JSON file with rules for encrypt and decrypt	`"rules": security_gateway_rules.json`
	`u`	`unsecured`	PCIe address for the unsecured port	`"unsecured": 03:00.1`
	`s`	`secured`	PCIe address for the secured port	`"secured": 03:00.0`
	`o`	`offload`	Offload mode	`"offload": "full"`

Chapter 8. References

- `/opt/mellanox/doca/applications/security_gateway/src/security_gateway.c`

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation nor any of its direct or indirect subsidiaries and affiliates (collectively: "NVIDIA") make no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assume no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

NVIDIA products are not designed, authorized, or warranted to be suitable for use in medical, military, aircraft, space, or life support equipment, nor in applications where failure or malfunction of the NVIDIA product can reasonably be expected to result in personal injury, death, or property or environmental damage. NVIDIA accepts no liability for inclusion and/or use of NVIDIA products in such equipment or applications and therefore such inclusion and/or use is at customer's own risk.

NVIDIA makes no representation or warranty that products based on this document will be suitable for any specified use. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to evaluate and determine the applicability of any information contained in this document, ensure the product is suitable and fit for the application planned by customer, and perform the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this document. NVIDIA accepts no liability related to any default, damage, costs, or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this document or (ii) customer product designs.

No license, either expressed or implied, is granted under any NVIDIA patent right, copyright, or other NVIDIA intellectual property right under this document. Information published by NVIDIA regarding third-party products or services does not constitute a license from NVIDIA to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property rights of the third party, or a license from NVIDIA under the patents or other intellectual property rights of NVIDIA.

Reproduction of information in this document is permissible only if approved in advance by NVIDIA in writing, reproduced without alteration and in full compliance with all applicable export laws and regulations, and accompanied by all associated conditions, limitations, and notices.

THIS DOCUMENT AND ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL NVIDIA BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF NVIDIA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms of Sale for the product.

Trademarks

NVIDIA, the NVIDIA logo, and Mellanox are trademarks and/or registered trademarks of Mellanox Technologies Ltd. and/or NVIDIA Corporation in the U.S. and in other countries. The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a world¬wide basis. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com