APPLICATION RECOGNITION

Reference Guide

MLNX-15-060459 _v1.2 | January 2022

Table of Contents

Chapter 1. Introduction

Application Recognition (AR) allows identifying applications that are in use on a monitored networking node.

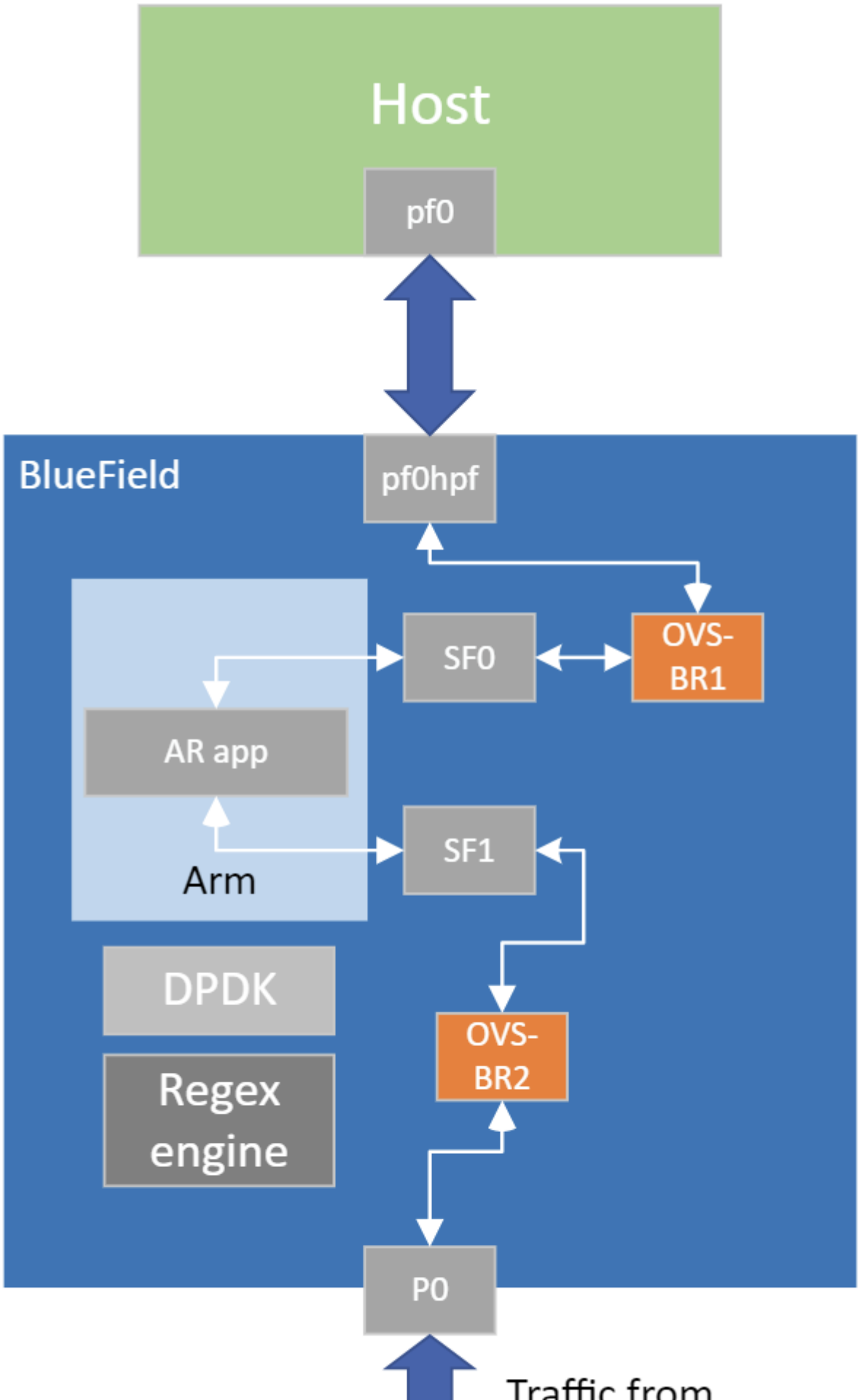
AR enables the security administrator to generate consolidated reports that show usage patterns from the application perspective. AR is also used as a corner stone of many security applications such as L7-based firewalls.

Due to the massive growth in the number of applications that communicate over Layer 7 (HTTP), effective monitoring of network activity requires looking deeper into Layer 7 traffic so individual applications can be identified. Different applications may require different levels of security and service.

This document describes how to build AR using the deep packet inspection (DPI) engine, which leverages NVIDIA® BlueField®-2 DPU capabilities such as regular expression (RXP) acceleration engine, hardware-based connection tracking, and more.

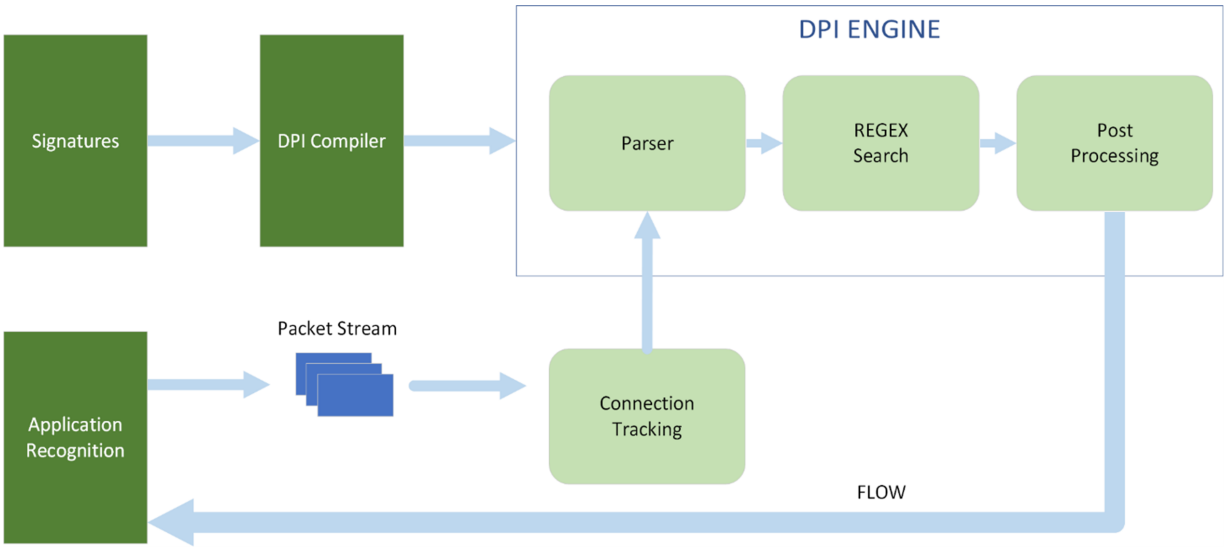
Chapter 2. System Design

The AR application is designed to run as "bump-on-the-wire" on the BlueField-2 instance, it intercepts the traffic coming from the wire, and passes it to the Physical Function (PF) representor connected to the host.


Host
pf0
BlueField
pf0hpf
SF0
OVS-
BR1
AR app
SF1
Arm
DPDK
Regex
engine
OVS-
BR2
P0
Traffic from

Chapter 3. Application Architecture

AR runs on top of Data Plan Development Kit (DPDK) based Stateful Flow Tracking (SFT) to identify the flow that each packet belongs to, then uses DPI to process L7 classification.



1. Signatures are compiled by DPI compiler and then loaded to DPI engine.
2. Ingress traffic is identified using the stateful table module in the DPDK libs which utilizes the connection tracking hardware offloads. This allows flow classifications to be done in the hardware level and be forwarded to the hairpin queue without being processed by the software, which increases performance dramatically.
3. Traffic is scanned against DPI engine compiled signature DB.
4. Post processing is performed for match decision.
5. Matched flows are identified, and actions can be offloaded to the hardware to increase performance as no further inspection is needed.
6. Flow termination is done by the aging timer set in the SFT to 60 seconds. When a flow is offloaded it cannot be tracked and destroyed.

Chapter 4. Configuration Flow

1. Parse application argument.
   ```
   arg_parser_init();
   ```
   a). Initialize arg parser resources.
   b). Register DOCA general flags.
   ```
   register_ar_params();
   ```
   c). Register AR application flags.
   ```
   arg_parser_start();
   ```
   d). Parse DPDK flags and call `rte_eal_init()` function.
   e). Parse app flags.
2. DPDK initialization.
   ```
   dpdk_init();
   ```
   a). Initialize SFT.
   b). Initialize DPDK ports, including mempool allocation.
3. AR initialization
   ```
   ar_init();
   ```
   a). Initialize NetFlow using default configuration `/etc/doca_netflow.conf`.
   b). Initialize signature database.
   c). Initialize DPI engine.
   d). Load signatures to DPI.
4. Configure DPI packet processing.
   ```
   dpi_worker_lcores_run();
   ```
   a). Configure DPI enqueue packets.
   b). Send jobs to RegEx engine.
   c). Configure DPI dequeue packets.
5. Send statistics and write database.
   ```
   sig_database_write_to_csv();
   send_netflow();
   ```
   a). Send statistics to the collector.
   b). Write CSV file with signature statistics.
6. AR destroy.
   ```
   ar_destroy();
   ```
   a). Clear thread.

b). Stop DPI worker.

c). Stop DOCA DPI.

7. DPI destroy

```
doca_dpi_destroy();
```

Chapter 5. Running Application on BlueField

1. Please refer to the DOCA Installation Guide for details on how to install BlueField related software.
2. The application recognition binary is located under `/opt/mellanox/doca/examples/application_recognition/bin/doca_application_recognition`.
3. To build the application:

   a). Run:
   ```
   cd /opt/mellanox/doca/examples/application_recognition/src
   meson /tmp/build
   ninja -C /tmp/build
   ```
   `doca_application_recognition` will be created under `tmp/build`.

   b). The build process depends on the `PKG_CONFIG_PATH` environment variable to locate the DPDK libraries. If the variable was accidently corrupted, and the build fails, please run the following command.

   - For Ubuntu:
     ```
     export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/opt/mellanox/dpdk/lib/aarch64-linux-gnu/pkgconfig
     ```
   - For CentOS:
     ```
     export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/opt/mellanox/dpdk/lib64/pkgconfig
     ```
4. Pre-run setup:

   a). The application recognition example is based on DPDK libraries. Therefore, the user is required to provide DPDK flags, and allocate huge pages. Run:
   ```
   echo 2048 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
   ```

   b). Make sure the regex engine is active:
   ```
   systemctl status mlx-regex
   ```
   If the status is inactive ("Active: failed"), run:
   ```
   systemctl start mlx-regex
   ```
5. To run the application:
   ```
   Usage: doca_application_recognition [DPDK Flags] -- [DOCA Flags] [Program Flags]
   DOCA Flags:
     -h, --help                 Print a help synopsis
     -l, --log-level            Set the log level for the app <CRITICAL=0, DEBUG=4>
   Program Flags:
     -p, --print-match          Prints FID when matched in DPI engine
     -n, --netflow              Collect netflow statistics and send according to
    conf file
   ```

```
  -i, --interactive           Adds interactive mode for blocking signatures
  -o, --output-csv <path>     Path to the output of the CSV file
  -c, --cdo <path>            Path to CDO file compiled from a valid PDD
```

For example:

```
/opt/mellanox/doca/examples/application_recognition/bin/
doca_application_recognition -a 0000:03:00.0,class=regex -a
 auxiliary:mlx5_core.sf.4,sft_en=1 -a auxiliary:mlx5_core.sf.5,sft_en=1 -- -c /
tmp/ar.cdo -p
```

Note: The SFT supports a maximum of 64 queues, thus the application cannot be run with more than 64 cores. To limit the number of cores, run:

```
/opt/mellanox/doca/examples/application_recognition/bin/
doca_application_recognition -a 0000:03:00.0,class=regex -a
 auxiliary:mlx5_core.sf.4,sft_en=1 -a auxiliary:mlx5_core.sf.5,sft_en=1 -l
 0-64 -- -c /tmp/ar.cdo -p
```

This limits the application to using 65 cores (core-0 to core-64) with 1 core for the main thread and 64 others to serve as workers.

To run doca_application_recognition using a JSON file:

```
doca_application_recognition --json [json_file]
```

For example:

```
/opt/mellanox/doca/examples/application_recognition/bin/
doca_application_recognition --json /root/ar_params.json
```

Note: Subfunctions must be enabled according to Scalable Function Setup Guide.

Note: The flags `-a 0000:03:00.0,class=regex -a auxiliary:mlx5_core.sf.4,sft_en=1 -a auxiliary:mlx5_core.sf.5,sft_en=1` are necessary for proper usage of the application. Modifying these flags results in unexpected behavior as only 2 ports are supported. The SF numbers are arbitrary and configurable. The RegEx device, however, is not and must be initiated on port 0.

For additional information on available flags for DPDK, use `-h` before the `--` separator:

```
/opt/mellanox/doca/examples/application_recognition/bin/
doca_application_recognition -h
```

For additional information on the app, use `-h` after the `--` separator:

```
/opt/mellanox/doca/examples/application_recognition/bin/
doca_application_recognition -- -h
```

The application will periodically dump a `.csv` file with the recognition results containing statistics about the recognized apps in the format `SIG_ID`, `APP_NAME`, `MATCHING_FIDS`, and `DROP`.

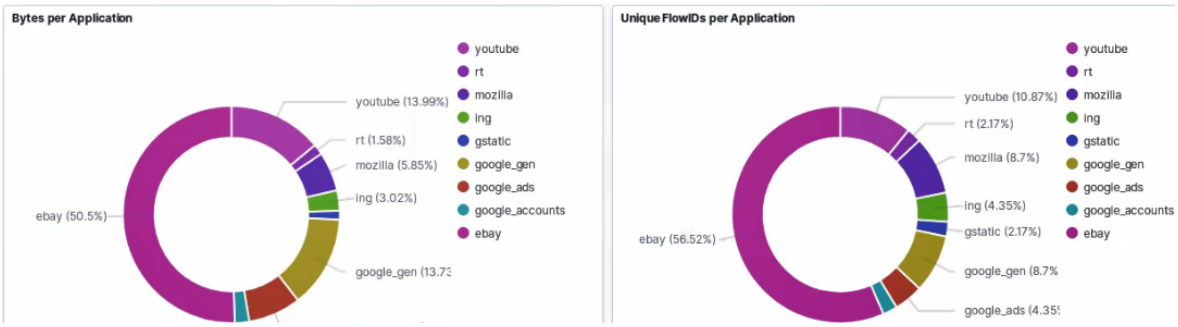
As per the example above, a file called `ar_stats.csv` will be created.

Additional features can be triggered by using the shell interaction. This allows blocking and unblocking specific signature IDs using the following commands:

- `block <sig_id>`
- `unblock <sig_id>`

The TAB key allows autocompletion while the `quit` command terminates the application.

NetFlow collector UI example:



6. To use the supplied signature file (`suricata_rules_example`), which is installed to the bin directory, the DPI compiler must be used, as the RegEx engine accepts only `.cdo` files. The CDO files are constructed by compiling a signature file written in the Suricata open-source format.

   To compile the signature file, run the following:

   ```
   doca_dpi_compiler -i /opt/mellanox/doca/examples/application_recognition/bin/
   ar_suricata_rules_example -o /tmp/ar.cdo -f suricata
   ```

   A `.cdo` will be created in the output path flagged as the `-o` input path of the compiler. That file can be used when executing the reference application using the `-c` flag as can be seen in previous bullet.

Chapter 6. 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
DPDK flags	a	devices	Add a PCI device into the list of devices to probe	`"devices": [ {"device": "regex",” id": "0000:03:00.0"}, {"device": "sf", "id": “4”,"sft": true}, {"device": "sf", "id": “5”,"sft": true}, ]`
	l	core-list	List of cores to run on	`"core-list": "0-4"`
General flags	l	log-level	Sets the log level for the application: ▶ CRITICAL=0 ▶ ERROR=1 ▶ WARNING=2 ▶ INFO=3 ▶ DEBUG=4	`"log-level": 4`
	h	help	Print a help synopsis	N/A
Program flags	p	print-match	Prints FID when matched in DPI engine	`"print-match": true`
	n	netflow	Collect netflow statistics and send according to conf file	`"netflow": false`

Flag Type	Short Flag	Long Flag/JSON Key	Description	JSON Content
	i	interactive	Adds interactive mode for blocking signatures	`"interactive": false`
	o	output-csv	Path to the output of the CSV file	`"output-csv": "/tmp/ar_stats.csv"`
	c	cdo	Path to CDO file compiled from a valid PDD	`"cdo": "/tmp/ar.cdo"`

Chapter 7. Running Application on Host

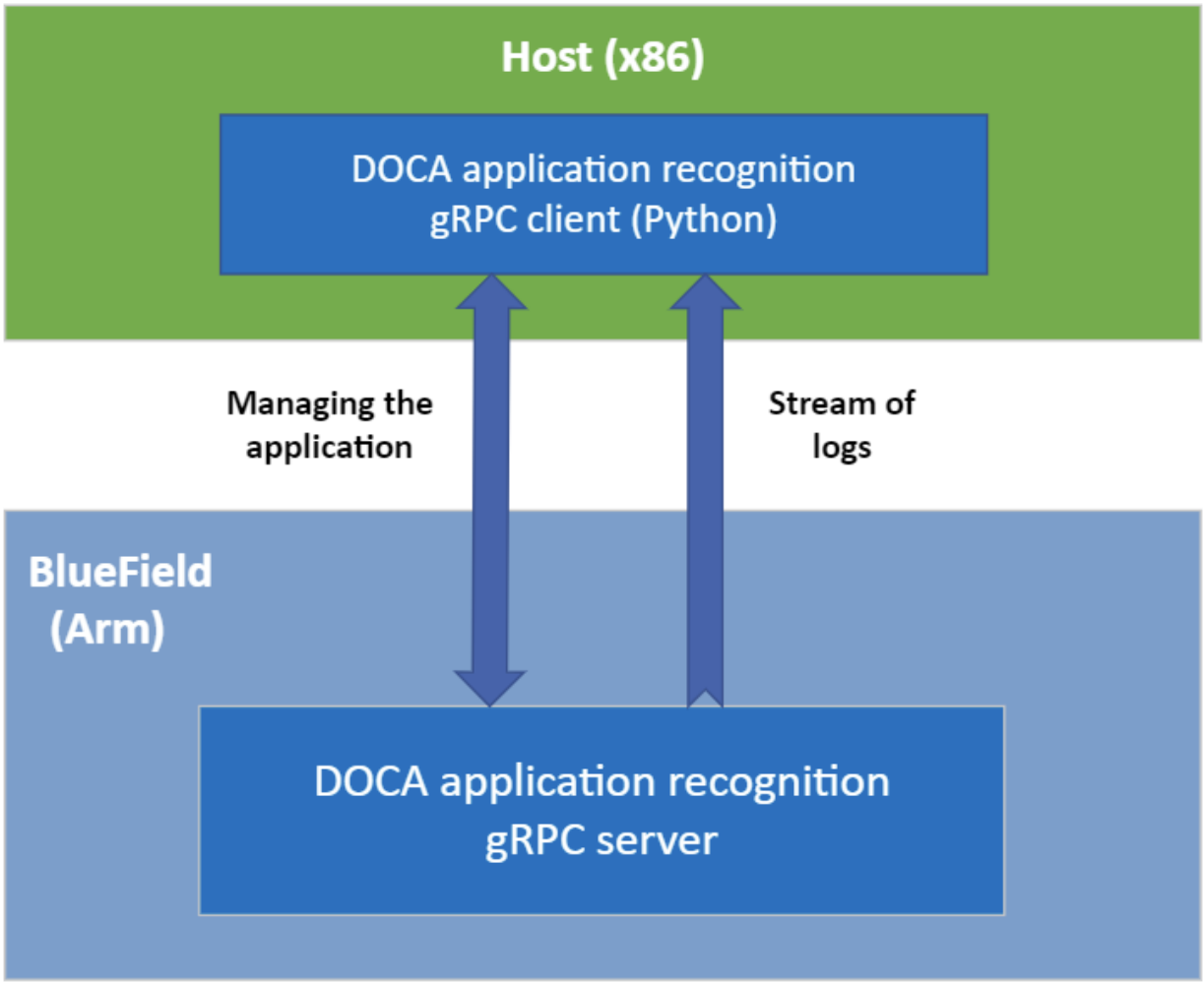
Host execution example:

```
doca_application_recognition -a 0000:04:00.0,class=regex -a 04:00.3 -a 04:00.4 -v --
 -c suricata_rules_example.cdo -o /tmp/check.csv -p
```

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

Chapter 8. Managing gRPC-Enabled Application from Host

Refer to NVIDIA DOCA gRPC Infrastructure User Guide for instructions on running the gRPC application server on the BlueField.



To run the Python client of the gRPC-enabled application:

```
./doca_application_recognition_gRPC_client.py -d/--debug <server address[:server
 port]>
```

For example:

```
/opt/mellanox/doca/examples/application_recognition/bin/grpc/client/
doca_application_recognition_gRPC_client.py 192.168.104.2
```

Note: Refer to known issue 2872883 in the NVIDIA DOCA Release Notes regarding the execution of the gRPC Python client.

Chapter 9. Deploying Containerized Application

The application recognition example supports a container-based deployment:

1. Refer to the NVIDIA DOCA Container Deployment Guide for details on how to deploy a DOCA container to the BlueField.
2. Application-specific configuration steps can be found on NGC under the application's container page.

Chapter 10. References

- `/opt/mellanox/doca/examples/application_recognition/src/application_recognition.c`
- `/opt/mellanox/doca/examples/application_recognition/src/grpc/application_recognition.proto`
- `/opt/mellanox/doca/examples/application_recognition/bin/ar_suricata_rules_example`

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