Remote Direct Memory Introspection

Hongyi Liu, Jiarong Xing, and Yibo Huang, *Rice University;* Danyang Zhuo, *Duke University;* Srinivas Devadas, *Massachusetts Institute of Technology;* Ang Chen, *Rice University*

https://www.usenix.org/conference/usenixsecurity23/presentation/liu-hongyi

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

Open access to the Artifact Appendices to the Proceedings of the 32nd USENIX Security Symposium is sponsored by USENIX.

USENIX'23 Artifact Appendix: Remote direct memory introspection

Hongyi Liu Jiarong Xing Yibo Huang Danyang Zhuo[†] Srinivas Devadas[‡] Ang Chen
Rice University [†]*Duke University* [‡]*MIT*

A Artifact Appendix

A.1 Abstract

This artifact appendix describes the workflow to setup and run RDMI. It includes an artifact check-list, description of hardware/software dependencies to install RDMI as well as setup instructions and experiment workflows. Please refer to the GitHub repository for further installation and execution details.

A.2 Description & Requirements

We provide a check-list for meta-information here.

- **Compilation:** GCC v7.5.0, Tofino SDE v8.4.0.
- **Binary:** Source code included to generate binaries
- **Run-time environment:** End host codes are tested on x86 servers with Ubuntu18.04 OS.
- **Hardware:** Intel/Barefoot Wedge 100BF-32X Tofino switch $\times$1, x86 server with Mellanox ConnectX-4 RNICs $\times$2.
- **Metrics:** Throughput, latency, CPU utilization, defense effectiveness.
- **Output:** The compiler will output configuration files used for configuring the programmable switch to enforce policies. Latency and traffic volume can be measure by tools like `tcpdump` or using in-switch telemetry. CPU utilization can be measurement by tools like `top`.
- **Experiments:** DSL compilation, connection establishment, switch reconfiguration and policy execution.
- **How much disk space required (approximately)?:** 1GB (dependencies not included)
- **How much time is needed to prepare workflow (approximately)?:** Compiling all programs needs about 1 hour (installation of software dependencies and hardware is not included)
- **How much time is needed to complete experiments (approximately)?:** About 2 hours to see the effect of all defenses.
- **Publicly available?:** Yes, code is available on GitHub.
- **Code licenses:** MIT license

A.2.1 Security, privacy, and ethical concerns

There is no security, privacy, and ethical concerns.

A.2.2 How to access

Our artifact and guidelines for installing and evaluating RDMI are publicly available at the following GitHub repository: commit: 7b8b15cf9a.

A.2.3 Hardware dependencies

To run RDMI, it requires two x86 servers connected by an Intel/Barefoot Tofino switch through Mellanox ConnectX-4 RNICs.

A.2.4 Software dependencies

Our experiments are performed on x86 servers running 64-bit Ubuntu 18.04, but similar Linux distributions should also work. To enable RDMA, Mellanox MLNX_OFED driver must be installed on the servers. RDMI's P4 code is compiled by proprietary toolchains provided by the switch vendors.

A.2.5 Benchmarks

None.

A.3 Set-up

To run RDMI, user needs to install all the dependency listed in check-list as well as install the NIC driver. We provide more details in the GitHub repository.

A.3.1 Installation

We list the main steps to install RDMI here. More details can be found in our GitHub repository.

- Install RNIC drivers to enable RDMA on end hosts.
- Install and setup the programmable switch following the vendor instructions.

A.3.2 Basic Test

To test compiler, run `make & python parse.py & ./RDMI 1000 100 1` inside *compiler* directory. It should result in a generated `cmd` file used for configuring the switch. To test the connections, run `sudo ./rdmatry_server -a SERVER_IP`

`-n 10 -m 1 -M 1000000000 -d 0` in the introspected machine side and `./rdmatry_client -a SERVER_IP -n 10 -M 1000000000 -r 10000000 -c 1 -t 99999999999 -p 1` in the remote side inside *switch* directory, and follow the instructions to establish the connections. The program should print out connection success information if the connection is setup correctly. To test the switch, run `./run_switchd.sh -p master` on the corresponding Tofino SDE environment. The load success information will be printed if the switch environment and program is correct. Then the user can follow the vendor provided instructions to configure the switch with the generated configuration files.

A.4 Evaluation workflow

We listed detailed workflows to conduct the experiments of the system in the GitHub repo. Here we provide three key steps below for the evaluation workflow. Please refer to our GitHub repository for further details:

- Establish the RDMA connections.
- Compile the policy and generate the corresponding configuration files.
- Configure the switch and run the program.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20220926. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2023/.