Enhancing Network Attack Detection with Distributed and In-Network Data Collection System

Seyed Mohammad Mehdi Mirnajafizadeh and Ashwin Raam Sethuram, *Wayne State University;* David Mohaisen, *University of Central Florida;* DaeHun Nyang, *Ewha Womans University;* Rhongho Jang, *Wayne State University*

https://www.usenix.org/conference/usenixsecurity24/presentation/mirnajafizadeh

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.

USENIX Security '24 Artifact Appendix: Enhancing Network Attack Detection with Distributed and In-Network Data Collection System

Seyed Mohammad Mehdi Mirnajafizadeh
Wayne State University

Ashwin Raam Sethuram
Wayne State University

David Mohaisen
University of Central Florida

DaeHun Nyang
Ewha Womans University

Rhongho Jang
Wayne State University

A Artifact Appendix

A.1 Abstract

The paper proposes In-network Serverless Data Collection (ISDC), a middleware that resides in distributed network switches operating system (NOS), forming a data layer with the main mission of eliminating the bottleneck between the infrastructure layer and the application layer for data collection. Interacting with its data plane functions, ISDC is responsible for task allocation and migration to effectively and efficiently utilize in-network resources. Meanwhile, ISDC performs serverless data migration and aggregation for data integrity to serve as a reliable and distributed data source for ML/DL-based security applications.

The artifact accompanies all codes for software/hardware data/control plane implementation of ISDC, simulation code in the P4 Mininet environment, and Python codes for running the distributed learning of collected data.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

There are no associated risks for the evaluators.

A.2.2 How to access

The artifact can be accessed through the stable GitHub URL: https://github.com/NIDS-LAB/ISDC/tree/90b1bbe813d8711004e967d7cfabed4566135fe7

A.2.3 Hardware dependencies

No specific hardware is required to run the software simulation. However, we recommend having an x86 machine with at least 32 cores, 64GB of RAM, and 20GB of free disk space.

A.2.4 Software dependencies

All artifacts should be run on Ubuntu 20.04 with Python 3.8, and Mininet simulation v2.3.1b1, TcpReplay 4.3.4. You also need Jupyter Notebook to open .ipynb files for federated learning. Lastly, we used the Flower v1.6.0 Python library for the federated learning implementation.

A.2.5 Benchmarks

To run the simulation and replay the traffic, we used three datasets: To simplify the data preparation to replay the traffic, we provided an example of already processed data in https://github.com/NIDS-LAB/ISDC/tree/main/Simulation/example/pcap.

A.3 Set-up

A.3.1 Installation

First, we need to install the P4 Mininet environment. The simplest yet the quickest way to install is using script install-p4dev-v5.sh, provided by p4-guide GitHub repository with detailed step-by-step installation instructions. After you are done with the installation, make sure you also have Tcpreplay to replay the traffic. To install Flower, you can refer to this resource.

A.3.2 Basic Test

To run the simulation, we refer to the steps mentioned in https://github.com/NIDS-LAB/ISDC/tree/main/Simulation.

A.4 Evaluation workflow

We list all of our claims and the experiments we performed to support them in the following. For the artifacts functional badge, we provide a minimal working example in Section A.4.2, and for Completeness, we provide the instructions in Section A.4.3.

A.4.1 Major Claims

(C1): ISDC's feature collection demonstrates high performance across different network topologies in terms of

top-K flow coverage and quality of data.

(C2): With the high-quality data provided by ISDC, the global model trained in a distributed manner demonstrates superior detection performance.

A.4.2 Exercisability

In this section, we provide a minimal working example of how to run the simulation for feature collection.

(E1): *Minimal Working Example [30 human-minutes + 1.5 compute-hour]:*

How to: Please follow the instruction provided in https://github.com/NIDS-LAB/ISDC/tree/main/Simulation to install and run the simulation for the selected small-scale topology.

Results: After the experiment is done, the switches now have the collected features for the data, which can be used for further evaluation. One sample of such a file is placed at https://github.com/NIDS-LAB/ISDC/tree/main/ML/CIC/dataset/isdc.

A.4.3 Completeness

For ML evaluation with the collected data from ISDC, we now leverage federated learning to train the model.

(E1): *Minimal Working Example [10 human-minutes + 1 compute-hour]:*

How to: We have made the collected datasets for Covert Channel and CIC available at https://github.com/NIDS-LAB/ISDC/tree/main/ML (located in the CIC and CovertChannel directories). Detailed codes for training a global model can be found in the FL.ipynb notebook within each directory.

Results: After training, the global model will be exported and saved for testing and evaluation in a `.keras` format. For detailed steps, refer to the FL.ipynb notebook.

A.5 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at https://secartifacts.github.io/usenixsec2024/.