A Artifact Appendix

A.1 Abstract

We implement a GPU-based information flow query tool and make its source code public for access. The project is available at https://github.com/mimicji/FlowMatrix/. In this Artifact Evaluation, we are applying for:

- "Artifact Available" badge

For "Artifact Available" badge application, we do not require hardware and software for this badge evaluation. However, to run our tool, NVIDIA GPU(s) and NVIDIA CUDA toolchain are required. In the paper, we claim the source code is available on GitHub (on page 6, Implementation Section). Thus, the "Artifact Available" badge would support our claims of availability.

A.2 Artifact check-list (meta-information)

- **Algorithm:** We do not propose any new algorithm.
- **Program:** We do not use public benchmark.
- **Compilation:** Besides the general compiler (GCC>=7.5), we also require a public compiler for our project: NVCC, the CUDA compiler driver. The version of the NVCC must be or greater than 11.3.
- **Transformations:** We do not require program transformation tool.
- **Binary:** We do not provide binaries.
- **Model:** We do not include models.
- **Data set:** We used recent CVEs and popular open-source programs for evaluation. The exploit of CVEs can be found in NVD.
- **Run-time environment:** We require Ubuntu (>=16.04). The `glibc` version should be greater than 2.30. We do not require root access.
- **Hardware:** We require special hardware for running our project: NVIDIA GPU(s). Any NVIDIA GPUs that CUDA toolkit supports would meet the requirement. However, we evaluate FlowMatrix with two V100 cards and different hardware other than V100 provided may affect the performance of FlowMatrix.
- **Run-time state:** Our tool is not sensitive to run-time state.
- **Execution:** We do not have specific conditions for execution. However, to gain the best performance, we recommend running FlowMatrix as the only task on GPUs.
- **Security, privacy, and ethical concerns:** There is no specific security, privacy, or ethical concerns.
- **Metrics:** The reported metrics include execution time, data flow throughput, performance profiling.
- **Output:** The tool will show the results to the console in numbers.
- **Experiments:** We have system scripts to run the experiments automatically.
- **How much disk space required (approximately)?:** Around 900GBs.
- **How much time is needed to prepare workflow (approximately)?:** Around 15 minutes.
- **How much time is needed to complete experiments (approximately)?:** Around 20 hours, depending on the devices.
- **Publicly available (explicitly provide evolving version reference)?:** The source code for FlowMatrix can be found at https://github.com/mimicji/FlowMatrix. The submitted version is https://github.com/mimicji/FlowMatrix/tree/c4a809f6c76ac447d0baf542db9e04b8d4600436.
- **Code licenses (if publicly available)?:** The code license is GPL3.0. Check the file `LISENSE` for more details.
- **Data licenses (if publicly available)?:** Not provided.
- **Workflow frameworks used?:** No workflow frameworks are used.
- **Archived (explicitly provide DOI or stable reference)?:** Please check https://github.com/mimicji/FlowMatrix/tree/c4a809f6c76ac447d0baf542db9e04b8d4600436.

A.3 Description

A.3.1 How to access

Please follow this URL https://github.com/mimicji/FlowMatrix/tree/c4a809f6c76ac447d0baf542db9e04b8d4600436.

A.3.2 Hardware dependencies

Any NVIDIA GPUs which are supported by CUDA.

A.3.3 Software dependencies

NVIDIA CUDA toolkit, SQLite C++ package, Protocol Buffers C++ package, Capstone C++ version.

A.3.4 Data sets

Not provided.

A.3.5 Models

No models are used.

A.3.6 Security, privacy, and ethical concerns

No Security, privacy, and ethical concerns.

A.4 Installation

After download, just run `make` command at the project home directory.

A.5 Experiment workflow

After compilation, a folder named `bin` with executable binaries will be generated under the project home directory. A system script has been used to automatically initialize the tool, pre-proceed all traces in the database, query the information flows with specified sources and destinations, and finally report results and performance numbers to the console. Following are the steps if a user would like to run the experiment step by step manually (which is also the workflow of the auto script).

Step 1: Initialization. To run FlowMatrix, the user needs to specify an SQLite database file for storage as the only parameter in the command line:

$$\$\ ./bin/QueryCLI\ [path\_to\_database]$$

This will open FlowMatrix console. Next, the user can choose one trace project stored in the database to work with in the console:

$$FMQuery>\ WorkOn\ [TraceName]$$

The user may check the list of traces via *ListTraces* command:

$$FMQuery>\ ListTraces$$

Step 2: Pre-processing. If the trace has never been pre-proceeded (pre-summarized), users need to tell FlowMatrix to pre-proceed parts of or all of it:

$$FMQuery>\ PrepareQuery\ [start]\ [end]$$

To check the trace length, we suggest the commend *TraceSize* which shows users the current trace length:

$$FMQuery>\ TraceSize$$

Also, although traces have their own trace reader, FlowMatrix allows users to view traces in the console:

$$FMQuery>\ TracePrintRange\ [start]\ [end]$$

Step 3: Querying. Once Step 2 has been done, the trace is ready to be queried for its information flows. The users may specify a sub-range of the pre-proceeded range to query information flows using *Query* command:

$$FMQuery>\ Query\ [V1Type]\ [V1]\ [Direct]\ [V2Type]\ [V2]$$

In this command, we support four types of data variables (*V1Type, V2Type*) to be queried, including system calls, instructions, registers, memory slots. Detailed usage can be found in the README.md document at https://github.com/mimicji/FlowMatrix/blob/main/README.md#query-usage. In the experiments, we usually choose *mmap*, *read* and *receive* system calls as the sources for CVEs, depending on the exploit. We choose input system calls or simply random instructions as the sources of a common program. The destination differs from CVEs, which can be *write* and *send* system calls, a register, or a memory slot. When it comes to common programs, destinations are output system calls and random instructions.

A.6 Evaluation and expected results

The claims have been made in this work:

- We analyze offline dynamic information flow operations on binaries and identify their linearity property.
- We propose FlowMatrix, a novel way of representing DIFT operations using matrices that enabling off-the-shelf GPUs to be used as a hardware co-processor for DIFT.
- We design an efficient solution to support interactive DIFT queries on offline execution traces. Our prototype demonstrates sub-second response time for several DIFT queries in common DIFT workloads.
- Our tool is open-sourced at GitHub.

First two claims are elaborated in the manuscripts and do not require experiments to support them.

The third claim can be supported by the performance results. Once the auto script finishes, it reports the performance for pre-processing and queries. The expected results can be found in the paper. The variation of empirical results depends on the provided GPUs and hard disks. In our test environment with two NVIDIA Tesla V100 cards and SSD, the pre-processing may have a 20% variance while the variance of query time may be 50% (especially in small cases).

The fourth claim can be supported by link access.

A.7 Experiment customization

NA.

A.8 Notes

NA.

A.9 Version

Based on the LaTeX template for Artifact Evaluation V20220119.