DeepType: Refining Indirect Call Targets with Strong Multi-layer Type Analysis

Tianrou Xia, Hong Hu, and Dinghao Wu, *The Pennsylvania State University*

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

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

USENIX Security '24 Artifact Appendix: DEEPTYPE: Refining Indirect Call Targets with Strong Multi-layer Type Analysis

Tianrou Xia
Pennsylvania State University

Hong Hu
Pennsylvania State University

Dinghao Wu
Pennsylvania State University

A Artifact Appendix

A.1 Abstract

As presented in the paper, the artifact is a prototype of Strong Multi-Layer Type Analysis (SMLTA), named DEEPTYPE. It is a static analysis tool that employs SMLTA to precisely and efficiently identify indirect call targets. Compared to TypeDive, the prototype of MLTA, DEEPTYPE is able to narrow down the scope of indirect call targets and reduce runtime and memory overhead. This appendix describes how to reproduce the following experiment results in the paper:

- Average number of indirect call targets (ANT) of DEEPTYPE, DT-weak, and DT-noSH.
- Runtime overhead of DEEPTYPE and DT-nocache.
- Memory overhead of DEEPTYPE.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The artifact performs static analysis on benchmarks to identify indirect call targets, which does not cause any security, privacy, or ethical concern to evaluators' machines.

A.2.2 How to access

The artifact is available on GitHub: https://github.com/s3team/DeepType/tree/AE.

A.2.3 Hardware dependencies

This artifact does not have specific requirements on hardware, but hardware features different from those for the experiments in the paper (8-core Intel Core i9-9880H CPU @ 2.30GHz and 16GB DDR4 RAM) may result in different runtime overhead.

A.2.4 Software dependencies

The experiments in the paper were performed on Ubuntu 20.04 (64-bit). The following tools and environments are also required: cmake-3.5.1, LLVM-15.0, and valgrind-3.15.0.

A.2.5 Benchmarks

The bitcode (compiled with O0 optimization level) of benchmarks in the paper are available at https://drive.google.com/file/d/1U9rMr4UC0uxVhAH7p0R3127lJpaaQMuj/view?usp=sharing.

A.3 Set-up

A.3.1 Installation

Download the source code of DEEPTYPE and follow the Setup Guide in README.md.

A.3.2 Basic Test

Follow the steps in How to Use to perform a basic test. The expected output is listed in Analysis Results.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): DEEPTYPE reduces average number of indirect call targets (ANT) across most benchmarks when compared to the state-of-the-art tool TypeDive. This is proven by the experiment (E1), which reproduces results described in Section 7.1 of the paper.

(C2): DEEPTYPE significantly reduces runtime overhead when compared to TypeDive. The variant DT-nocache also demonstrates reduced runtime overhead. This is proven by experiment (E2), which reproduces results described in Section 7.2 of the paper.

(C3): DEEPTYPE shows lower memory overhead than TypeDive, while the subtle difference between two tools remains consistent across all benchmarks. This is proven by experiment (E3), which reproduces results described in Section 7.2 of the paper.

(C4): The ANT of DT-noSH is close to that of DEEPTYPE, indicating that disabling the special handlings has minimal impact on the effectiveness of DEEPTYPE, which reveals the primary role of SMLTA in accuracy improvement. This is proven by experiment (E4), which reproduces results described in Section 7.3 of the paper.

(C5): DT-weak demonstrates a higher ANT than DEEPTYPE across most benchmarks, indicating that recording entire multi-layer types, rather than two-layer types, effectively refines indirect call targets. This is proven by experiment (E5), which reproduces results described in Section 7.3 of the paper.

A.4.2 Experiments

(E1): [Measure ANT of DEEPTYPE]
How to: Follow the steps in How to Use to analyze each benchmark.
Results: See "Avg. Number of indirect-call targets" in output.
Time consumption: 1 to 2 minutes for linux, several seconds per other benchmarks.
Memory consumption: 4 GB memory for linux, 70 to 140 MB memory per other benchmarks.

(E2): [Measure runtime overhead]
How to: Build DT-nocache following the guide in Configurations. For each benchmark, run DEEPTYPE and DT-nocache multiple times to achieve average execution time. Note that, the order of running DEEPTYPE and DT-nocache affects the runtime overhead because the first execution may need to load data and code from disk into memory and cache while the afterwards executions may quickly access data from memory and cache. So, we ran DEEPTYPE and DT-nocache 3 times for each benchmark after warming up and calculated the average execution time to mitigate the impact of hardware conditions and operating system states.
Results: See the execution time at the end of the output.
Time consumption: 1 to 2 minutes for linux, several seconds per other benchmarks.
Memory consumption: 4 GB memory for linux, 70 to 140 MB memory per other benchmarks.

(E3): [Measure memory overhead]
How to: Use Massif tool in Valgrind tool suite to evaluate memory consumption. Here are the steps:

1. Generate Massif's profiling data, which will be written in a file named massif.out.<pid>. Replace `filename.bc` with each benchmark.

```
$ cd path/to/DeepType/build/lib
$ valgrind --tool=massif --pages-as-
    heap=yes ./kanalyzer filename.bc
```

2. Show the information gathered by massif.

```
$ ms_print massif.out.<pid>
```

3. Determine the snapshot indicating the peak of memory consumption. For example, the peak is 76 when DEEPTYPE is analyzing sqlite.

```
Detailed snapshots: [1, 17, 23, 28, 33,
     41, 50, 52, 62, 72, 76 (peak)]
```

4. Find the memory consumption at the peak snapshot. For example, the memory used at snapshot 76 is 139.2 MB when DEEPTYPE is analyzing sqlite.

Results: The memory consumption of DEEPTYPE for each benchmark.
Time consumption: Several minutes per benchmark, including the execution time of Valgrind tools and manual analysis.
Memory consumption: 4 GB memory for linux, 70 to 140 MB memory per other benchmarks. Valgrind's memory overhead can range from around 100% to 500% of the original memory usage.

(E4): [Measure ANT of DT-noSH]
How to: Follow the guide in Configurations to build DT-noSH. Use DT-noSH to analyze each benchmark.
Results: See "Avg. number of targets" in output.
Time consumption: 1 to 2 minutes for linux, several seconds per other benchmarks.

(E5): [Measure ANT of DT-weak]
How to: Follow the guide in Configurations to build DT-weak. Use DT-weak to analyze each benchmark.
Results: See "Avg. number of targets" in output.
Time consumption: 1 to 2 minutes for linux, several seconds per other benchmarks.

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/.