Ground Truth for Binary Disassembly is Not Easy

Chengbin Pang and Tiantai Zhang, *Nanjing University;* Ruotong Yu, *University of Utah;* Bing Mao, *Nanjing University;* Jun Xu, *University of Utah*

https://www.usenix.org/conference/usenixsecurity22/presentation/pang-chengbin

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

Open access to the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium is sponsored by USENIX.

A Artifact Appendix

A.1 Abstract

Our data set contains nearly 10K binaries which are compiled by our toolchains to obtain the ground truth of binary disassembly. The binaries vary from x86/x64, arm32/aarch64 to mipsle32/mipsle64. To compare popular disassemblers with different ground truths, we also include the result of binary disassembly of popular disassemblers and ground truths in the data set. To validate the result of the paper, we prepare scripts to compare disassemblers with ground truth on major disassembly tasks(instruction recovery, function detection, and jump table reconstruction) and present the accuracy(precision and recall) in the console.

The minimal disk space is about 100 GB. We have tested it in `Ubuntu18.04` and `Ubuntu20.04`. The software requirements are `python3` and `python3-pip`.

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

- **Data set:** The data set contains 10K binaries and ground truths of binary disassembly. We open sourced the data set in https://doi.org/10.5281/zenodo.6566082. The approximate size is 85GB.
- **Run-time environment:** Linux. We tested in Ubuntu 18.04 and Ubuntu 20.04.
- **Metrics:** The accuracy(precision and recall) or the number of false positives and false negatives of disassemblers.
- **Output:** The output is shown in console. The result is numerical results. The expected result is shown in the paper.
- **Experiments:** We prepared `bash` scripts to automate the experiments as possible.
- **How much disk space required (approximately)?:** 100GB.
- **How much time is needed to prepare workflow (approximately)?:** 1-2 hour(s).
- **How much time is needed to complete experiments (approximately)?:** 9 hours.
- **Publicly available (explicitly provide evolving version reference)?:** https://github.com/junxzm1990/x86-sok/tree/25656adbe14/artifact_eval
- **Code licenses (if publicly available)?:** MIT license.
- **Data licenses (if publicly available)?:** Creative Commons Attribution 4.0 International.
- **Archived (explicitly provide DOI or stable reference)?:** https://doi.org/10.5281/zenodo.6566082

A.3 Description

A.3.1 How to access

https://github.com/junxzm1990/x86-sok/tree/25656adbe14/artifact_eval

A.3.2 Hardware dependencies

As we prepared large scale data set and result of binary disassembly and ground truths, our artifact requires at least 100GB storage.

A.3.3 Software dependencies

`python3`, `python3-pip`, `docker`, `qemu`

A.3.4 Data sets

https://doi.org/10.5281/zenodo.6566082

A.3.5 Models

N/A

A.3.6 Security, privacy, and ethical concerns

N/A

A.4 Installation

- Download source code. https://github.com/junxzm1990/x86-sok. After downloading the source code, please change current directory to `x86-sok/artifact_eval`.
- Download data sets. (i) x86/x64 data sets are in https://zenodo.org/record/6566082/files/x86_dataset.tar.xz?download=1. The decompressed size is 56GB. (ii) arm32/aarch64 and mipsle32/mipsle64 data set is in https://zenodo.org/record/6566082/files/arm_mips_dataset.tar.gz?download=1. The decompressed size is 35GB. To evaluate the result easily, please create a new directory named `table_7` and move the second data set into it.
- Set up environment. Please refer to https://github.com/junxzm1990/x86-sok/tree/25656adbe14/artifact_eval#set-up-environment.

A.5 Evaluation and expected results

- **Impacts on Training Accuracy.** To show the impacts on different ground truths for training accuracy, we evaluate instruction recovery of `XDA`. We prepared trained models of `XDA` and test suite. The expected result is shown in paper Table 3. The steps to reproduce the evaluation are in https://github.com/junxzm1990/x86-sok/tree/25656adbe14/artifact_eval#xda1h.
- **Impacts on Tool Evaluation.** We evaluated `dyninst`, `ZAFL`, and `IDA` with different ground truths on x86/x64 testsuite. (i) The steps to reproduce the comparisons between `dyninst` with different ground truths of `dyninst` is in https://github.com/junxzm1990/x86-sok/tree/25656adbe14/artifact_eval#performance-of-dyninst-on-complex-constructs40mins and the expected result is in paper Table 4. (ii) The steps to reproduce the comparisions between `ZAFL` with different ground truths is in https://github.com/junxzm1990/x86-sok/tree/25656adbe14/artifact_eval#performance-

of-zafl-on-instruction-recovery1h and the expected result is in paper Table 5. (iii) The steps to reproduce the result of jump table recovery between `IDA pro` with `OracleGT` is in https://github.com/junxzm1990/x86-sok/tree/25656adbe14/artifact_eval#distribution-of-precision-of-ida20mins and the expected result is in paper Figure 2.

- **Impacts on Tool Comparison.** We compared popular disassemblers on instruction recovery on `openssl`. The steps of reproduction is in https://github.com/junxzm1990/x86-sok/tree/25656adbe14/artifact_eval#accuracy-of-popular-disassemblers-on-recovering-instructions20mins and the expected result is in paper Figure 3.
- **Impacts on improvements of `OracleGT`.** To show the impacts on improvements of `OracleGT`, we present the accuracy of popular disassemblers on recovering jump tables from `glibc`. The expected result is shown in paper Figure 5. The steps of reproduction are in https://github.com/junxzm1990/x86-sok/tree/25656adbe14/artifact_eval#accuracy-of-popular-disassemblers-on-recovering-jump-tables-from-glibc10mins.
- **Evaluation of mainstream disassemblers on binaries with different architectures.** We present Figure 6 to show the recall and precision of mainstream disassemblers on binaries with different architectures. Note that the overall result in x86/x64 is nearly the same as the result presented in Sok [1], we skip the reproduction on binaries in x86/x64. To reproduce the result of arm32/aarch64 and mipsle32/mipsle64, we prepared the tutorial in https://github.com/junxzm1990/x86-sok/tree/25656adbe14/artifact_eval#compare-result-of-arm-and-mips-disassemblers-result-3h. The expected result is shown in paper Table 7.
- `OracleGT` ***v.s.*** **Compilation Metadata** To reproduce the result, the tutorial is in https://github.com/junxzm1990/x86-sok/tree/25656adbe14/artifact_eval#oraclegt-vs-compilation-metadata-20mins and the expected result is in paper Table 6.
- **Extendibility(Optional).** We also provide an example to show how to build a new test suite with our toolchains. The tutorial is in https://github.com/junxzm1990/x86-sok/tree/25656adbe14/artifact_eval#how-to-build-new-testsuite.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20220119.

References

[1] Chengbin Pang, Ruotong Yu, Yaohui Chen, Eric Koskinen, Georgios Portokalidis, Bing Mao, and Jun Xu. Sok: All you ever wanted to know about x86/x64 binary disassembly but were afraid to ask. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 833–851. IEEE, 2021.