Smart Learning to Find Dumb Contracts

Tamer Abdelaziz, *National University of Singapore;*
Aquinas Hobor, *University College London*

https://www.usenix.org/conference/usenixsecurity23/presentation/abdelaziz

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: Smart Learning to Find Dumb Contracts

Tamer Abdelaziz[†]
tamer@comp.nus.edu.sg
(†) National University of Singapore
Singapore

Aquinas Hobor[‡,†]
a.hobor@ucl.ac.uk
(‡) University College London
London, United Kingdom

A Artifact Appendix

A.1 Abstract

The artifact is an implementation of *Deep Learning Vulnerability Analyzer (DLVA)*, a vulnerability detection tool for Ethereum smart contracts based on powerful deep learning techniques for sequential data adapted for bytecode. We benchmark DLVA against nine competitors.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

N/A

A.2.2 How to access

Both DLVA and $\text{DLVA}_{\text{large}}$ are available as Docker images at https://hub.docker.com/u/dlva.

A.2.3 Hardware dependencies

The test machine is a desktop with a 12-core 3.2 GHz Intel(R) Core(TM) i7-8700 and 16 GB of memory.

A.2.4 Software dependencies

Ubuntu OS and Docker should be installed.

A.2.5 Benchmarks

In this artifact, we use three benchmarks to compare DLVA with the-state-of-the-art tools. $\text{Elysium}_{\text{benchmark}}$ https://bit.ly/Elysium_benchmark, $\text{Reentrancy}_{\text{benchmark}}$ https://bit.ly/Reentrancy_benchmark, $\text{SolidiFI}_{\text{benchmark}}$ https://bit.ly/SolidiFI_benchmark.

A.3 Set-up

A.3.1 Installation

The instructions to install DLVA as follows: Open the terminal and enter the following command to create a "dlva folder" in the home directory: `mkdir ~/dlva`

- To install DLVA (trained on SolidiFI's labels or on Slither's labels of small-length contracts):
  1. Pull the latest version of the DLVA Docker image by running the following command:
     `docker pull dlva/dlva:latest`
  2. Run the DLVA Docker container with the following command:
     `docker run -i -t -v ~/dlva/:/DLVA_Tool/dlva/ dlva/dlva`
  3. After executing the above commands, you will be inside the DLVA Docker container.
- To install $\text{DLVA}_{\text{large}}$ (trained on Slither's labels of large-length contracts):
  1. Pull the latest version of the $\text{DLVA}_{\text{large}}$ Docker image by running the following command:
     `docker pull dlva/dlva-large:latest`
  2. Run the $\text{DLVA}_{\text{large}}$ Docker container with the following command:
     `docker run -i -t -v ~/dlva/:/DLVA_Tool/dlva/ dlva/dlva-large`
  3. After executing the above commands, you will be inside the $\text{DLVA}_{\text{large}}$ Docker container.

A.3.2 Basic Test

Now, follow the command-line interface instructions to interact with DLVA Docker container.

1. Press 1 to run DLVA trained on SolidiFI labels, or press 2 to run DLVA trained on Slither labels (small contracts).
2. Enter 1 for a single contract mode, or 2 for a batch of contracts mode.
3. For a single contract mode: enter contract address *e.g.* `0x01f8c4e3fa3edeb29e514cba738d87ce8c091d3f` or insert the bytecode in "dlva/input.bin" file at "dlva folder" then enter: `b`

4. For batch mode: copy the dataset file (*e.g.*, batch.csv with "address" and "bytecode" columns) to the "dlva folder", then enter `../dlva/batch.csv` Alternatively, the user can use the provided test dataset by entering `Testset10.csv` The analysis results will be written in file named "DLVA_Predictions_batch.csv" at "dlva folder".

To test $\text{DLVA}_{\text{large}}$ Docker container follow the same aforementioned steps without step 1.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): DLVA performs well in practice for smart contract vulnerability detection: Figure 1 illustrates DLVA performance against nine competitors. DLVA is on the far right. We use bar-and-whiskers where star $\star$ represents the mean and plus $+$ represents outliers. Our average Completion Rate (*i.e.*, the percentage of contracts for which a tool produces an answer, the higher the better) is 100.0%. Our average accuracy is 99.7% (the higher the better), with a True Positive Rate (*i.e.*, detection rate; the higher the better) of 98.7% and a False Positive Rate (*i.e.*, false alarm rate; the lower the better) of 0.6%. Our average analysis time per contract (the graph is in log scale, lower better) is 0.2 seconds. Smart learning pays off: DLVA beats Slither on every statistic except for TPR (where it lags by 0.7%). Recall also that Slither requires source whereas DLVA needs only bytecode.
This is proven by the experiment (E1) described in §4.4 whose results are illustrated/reported in Figure 1, the data underlying Figure 1 is in Tables 4, 5, and 6.

A.4.2 Experiments

(E1): *30 human-minutes + machine with 12-core and 16 GB memory + 12 GB disk*
How to: Run the DLVA Docker container with the following command:
`docker run -i -t -v ~/dlva/:/DLVA_Tool/dlva/ dlva/dlva`
Preparation: After executing the previous command, the three benchmarks A.2.5 will be downloaded automatically and saved to the "dlva folder".
Execution: Run DLVA on the three benchmarks:

1. For $\text{Elysium}_{\text{benchmark}}$:
   (a) Press 2 to select DLVA trained on Slither labels (small contracts).
   (b) then, press 2 to select the batch of contracts mode.
   (c) then, enter: `../dlva/Elysium_benchmark.csv`
   (d) then, wait until the analysis is complete, the raw results will be stored at "dlva/DLVA_Predictions_for_Elysium_benchmark.csv"
2. For $\text{Reentrancy}_{\text{benchmark}}$:
   (a) Press 2 to select DLVA trained on Slither labels (small contracts).
   (b) then, press 2 to select the batch of contracts mode.
   (c) then, enter: `../dlva/Reentrancy_benchmark.csv`
   (d) then, wait until the analysis is complete, the raw results will be stored at "dlva/DLVA_Predictions_for_Reentrancy_benchmark.csv"
3. For $\text{SolidiFI}_{\text{benchmark}}$:
   (a) Press 1 to select DLVA trained on SolidiFI labels.
   (b) then, press 2 to select the batch of contracts mode.
   (c) then, enter: `../dlva/SolidiFI_benchmark.csv`
   (d) then, wait until the analysis is complete, the raw results will be stored at "dlva/DLVA_Predictions_for_SolidiFI_benchmark.csv"
4. Press 0 to exit from the DLVA Docker image.

Results: Enter: `cd ~/dlva`
then enter: `pip3 install -r requirements.txt`
then: `python3 print_dlva_results.py` that will show the DLVA results on the three benchmarks.
then: `python3 print_competitors_results.py` that will show the competitors results on the three benchmarks.
Open the "dlva folder", seven files have been added: "tools_results.txt" contains the log performance for all tools based on raw data in "10_tools_files" folder, "tools_results.csv" represents the same results as a spreadsheet, and five files of "tools_predictions_benchmark.csv" contain labels of all tools for each benchmark.
The FN and FP produced numbers in this experiment should match the numbers in Tables 4, 5, and 6.
Any number in Figure 1 is the average for each tool using the three benchmarks.

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/.