



USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

APPATCH: Automated Adaptive Prompting Large Language Models for Real-World Software Vulnerability Patching

*Yu Nong, University at Buffalo; Haoran Yang, Washington State University;
Long Cheng, Clemson University; Hongxin Hu and Haipeng Cai, University at Buffalo*

<https://www.usenix.org/conference/usenixsecurity25/presentation/nong>

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Artifact Appendices to the Proceedings of the 34th USENIX Security Symposium is sponsored by USENIX.



USENIX Security '25 Artifact Appendix: APPATCH: Automated Adaptive Prompting Large Language Models for Real-World Software Vulnerability Patching

Yu Nong¹, Haoran Yang², Long Cheng³, Hongxin Hu¹, and Haipeng Cai^{1*}

¹University at Buffalo, ²Washington State University, ³Clemson University

¹{yunong, hongxin, haipeng}@buffalo.edu, ²haoran.yang@wsu.edu, ³lcheng2@clemson.edu

A Artifact Appendix

A.1 Abstract

APPATCH is an automated LLM-based patching system that elicits LLMs to effectively reason about vulnerable code behaviors with vulnerability semantics reasoning and adaptive prompting. We provide access to the APPATCH datasets, results, source code, and a functional Docker image described in the paper.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

Our research is focused on the development and evaluation of methods for patching software vulnerabilities on the source code level, which does not have any risk for evaluators when executing the artifact with regards to the machines' security, data privacy, or other ethical concerns.

A.2.2 How to access

We have made the APPATCH datasets, results, and source code publicly available on Zenodo: <https://zenodo.org/records/14741018> with a functional Docker image on Docker Hub: <https://hub.docker.com/repository/docker/g2ecb/appatch-demo/>.

A.2.3 Hardware dependencies

Our artifact can run on machines with at least **20GB** spare disk space and **16GB** CPU memory.

A.2.4 Software dependencies

Our functional APPATCH image requires **Docker** to set up and execute.

*Haipeng Cai is the corresponding author.

A.2.5 Benchmarks

Considering the high cost of the commercial LLMs, we only provide 16 interprocedural samples for the functional badge evaluation. For the complete datasets and results, please refer to appatch.zip uploaded on Zenodo.

A.3 Set-up

A.3.1 Installation

We have uploaded the functional Docker image to Docker hub. To pull and install the image, simply run:

```
$ docker run -it -d --name appatch-demo  
g2ecb/appatch-demo bash
```

After installing the image, enter the container with:

```
$ docker exec -it appatch-demo bash
```

After entering the Docker container, please first provide the keys in the api_keys.json file.

```
$ cd ~  
$ vi api_keys.json
```

A.3.2 Basic Test

In the home directory (`/root/`) of the container, run the automatic script provided for the whole pipeline:

```
$ source run_all.sh
```

If the API keys are set correctly, you will see the script run without reporting model related errors.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): APPATCH is able to generate vulnerability patches with its key components: (1) Semantics-Aware Scoping (Slicing), (2) Vulnerability Semantics Reasoning (Root Cause Analysis), (3) Dynamic Adaptive Patch Generation, and (4) Multi-Faceted Patch Validation. We demonstrate the claim through the demo Docker image in Experiment (E1).

A.4.2 Experiments

(E1) Patch Generation (*1 human-minute + 20 compute-minutes*):

How to: Run the code in the execution section below.

Preparation: Set up the API keys described in Section A.3.1. Then, go to the home directory of the container.

Execution: Run the automatic script provided for the whole pipeline:

```
$ source run_all.sh
```

Results: After running the pipeline, the generated slices, root cause analysis, patches, and validation results are stored in `interprocedural_sample_slices`, `root_cause_analysis`, `generated_patches`, and `generated_patches_<model-name>_valid` folders in `/root/appatch/`. By opening any generated files in `root_cause_analysis` and `generated_patches`, you can see results similar to [Figure 5](#) and [Figure 7](#) in the paper.

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/usenixsec2025/>.