WOOT'26 Artifact Appendix: Squeezing Juicy Variant Bugs Out of Modern Browsers

Han Zheng
EPFL

Flavio Toffalini
Ruhr-Universität Bochum

Qiang Liu
EPFL

Mathias Payer
EPFL

A Artifact Appendix

This document describes the artifacts accompanying the USENIX WOOT '26 publication Squeezing Juicy Variant Bugs Out of Modern Browsers. The artifact include variant analysis queries we use to find Chrome / VSCode vulnerabilities.

A.1 Description & Requirements

A.1.1 Security, privacy, and ethical concerns

This artifact does not pose any threats to the user system, as it only scan the code that follow the variant patterns described in the paper. Moreover, all the bugs mentioned in the paper are reported to the vendors and users are safe to these patterns as long as they update their software.

A.1.2 How to access

The artifact is uploaded to Github repo https://github.com/HexHive/Grape. Alternatively, it is also available in https://doi.org/10.5281/zenodo.20020117. The artifact includes the queries and scripts to reproduce our results, including semgrep rules and the script that automatically fill the *violation* functions into the rules and run the queries. Users may still need to download the Chromium VSCode and AzureDataStudio source code for analysis.

A.1.3 Hardware dependencies

We recommend a personal Desktop with 16GB memory and over 200GB free disk space. We only test our artifact on Ubuntu 22.04 thus we recommend same OS. This artifact dont have any CPU requirement.

A.1.4 Software dependencies

To reproduce our results, we recommend Python 3.10.12 and SemGrep 1.93.0.

A.2 Set-up

A.2.1 Installation

Download the artifact, install the Python 3.10.12 (default python in Ubuntu 22.04) and SemGrep 1.93.1 using *pip3 install semgrep==1.93.1*.

Also download the chromium following the chromium document, Checkout to test version 126.0.6465.2 using git checkout 126.0.6465.2 && gclient sync −D.

A.3 Evaluation workflow

A.3.1 Major Claims

(C1): Grape finds all bugs described in Table 4 in the original paper. This is supported by experiment 1 (E1). Note that bug 10 is introduced in chrome 130.0.6710.0 thus can only be found when switching chromium to the corresponding version.

(C2): Grape suffers from the same false positive rate described in Table 5 in the original paper. This is also supported by experiment 1 (E1).

A.3.2 Experiments

(E1): [10 human-minutes + 0.5 computer-hour] This evaluation runs Grape with predefined variant queries and report bugs we discovered.
How to: Delete directory 'config//false_positive' to get full results. Run src/main.py following the description in the README. Each $BUG_VARIANT represent a bug variant in the Table 3.
Results for Claim 2: The stdout output should contain total number of variants found by Grape, this number should match the results in the Table 5.
Results for Claim 1: The stdout also include each item of variants found by Grape. Compare the bug locations in Table 4 (find the original report in chrome issue tracker, and you'll find analysis and bug location there. The b/1234 means crbug.com/1234.) and validate if they're in query's output. All the bugs in Table 4 should be found by Grape.

A.4 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/.