FV8: A Forced Execution JavaScript Engine for Detecting Evasive Techniques

Nikolaos Pantelaios and Alexandros Kapravelos, *North Carolina State University*

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

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: FV8: A Forced Execution JavaScript Engine for Detecting Evasive Techniques

Nikolaos Pantelaios
North Carolina State University

Alexandros Kapravelos
North Carolina State University

A Artifact Appendix

A.1 Abstract

FV8 (pronounced "favorite") is a specialized version of the V8 engine enhanced with unique capabilities. It also integrates patches from VisibleV8 and its custom patches to deliver a powerful tool for code analysis and execution. FV8 forces execution of code paths to detect evasive techniques in JavaScript, enhancing visibility and detecting hidden or malicious code. It is used in both Node.js and Chromium environments.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

There are no specific risks associated with executing the FV8 artifact. The original experiments, were run inside a controlled and isolated environment and involved malicious npm packages and malicious extensions potentially. Since for this artifact evaluation we are visiting only secure websites, there is no danger involved whatsoever.

A.2.2 How to access

The artifact can be accessed via the GitHub repository at: https://github.com/wspr-ncsu/FV8/tree/57fc2b26995633169586039 34aba572907171162. The specific version of the artifact used for this evaluation is tagged as `v1.0`.

A.2.3 Hardware dependencies

None. FV8 can run on any standard x86_64 hardware with support for Docker.

A.2.4 Software dependencies

The primary software dependencies include:

- *Operating System: Linux (Ubuntu 20.04 recommended)*
- *Docker: For containerized execution*
- *Python > 3.10: For running scripts*
- *(provided) Chromium: Version 94 to 122, specifically Chromium 122 patches as a deb file*
- *Required Python packages as specified in* `requirements.txt`

A.2.5 Benchmarks

For our purposes we will run FV8 on Tranco top 1k or Tranco top 10k. In the paper, the benchmarks involve running FV8 on datasets of browser extensions and npm packages to detect evasive techniques.

A.3 Set-up

A.3.1 Installation

```
# Clone the repository
git clone https://github.com/wspr-ncsu/FV8.git
cd FV8

# Install required Python packages
pip install -r scripts/requirements.txt

# Setup Docker
export DOCKER_BUILDKIT=0
python scripts/vv8-cli.py setup

# Patch FV8 through the patches (Optional)
patch -p1 < patches/LATEST_FV8_PATCH_FILE


## Otherwise (directly run the .deb file)

# Install the FV8 Chromium .deb package (optional)
dpkg -i deb_files/chromium-browser-stable_ \
122.0.6261.111-1_amd64.deb
```

A.3.2 Basic Test

```
# Run FV8 Chromium with a simple test URL

chromium-browser-stable --headless \
--no-sandbox --disable-gpu \
--disable-features=NetworkService \
```

```
--js-flags='--no-lazy' https://google.com

# Run the FV8 Crawler
python crawler_queue_tranco.py
```

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): *FV8 operates seamlessly on top of V8 without any disruptions.*

(C2): *FV8 is compatible with VisibleV8, and the patches can be integrated without conflicts.*

(C3): *FV8 increases code execution visibility, effectively identifying JavaScript evasions and enforcing code execution. Notably, FV8 achieves 11% higher code coverage compared to the default V8, as demonstrated in the experiments detailed in Section 5.1 of the paper.*

A.4.2 Experiments

(E1): *[Functionality and Performance] [0 human-minutes + 5 compute-minutes] Install the .deb file. Run the latest version of FV8 (equivalent to Chromium 122) using the .deb file and verify that it operates correctly without any issues, following the example command for chromium-browser-stable.*

Execution: sudo dpkg -i deb_files/chromium-browser-stable_122.0.6261.111-1_amd64.deb
chromium-browser-stable –headless –no-sandbox –disable-gpu –disable-features=NetworkService –js-flags='–no-lazy' https://google.com

Results: If even one forced execution happens, you should see a message with the API that triggered the execution and where it is in the code:
Example: *Found match for API: setTimeout and varproxy name: 0x7e0a001c7c1d <String[10]: #setTimeout>on position: 8066*

(E2): *[Compatibility and Patches] [20 human-minutes + 1 compute-hour] Apply patches to V8, incorporating both FV8 and VisibleV8 patches. The crawler automatically handles this for the latest version, as detailed in the README.md file on the GitHub repository https://github.com/wspr-ncsu/FV8.*

Execution: # Setup the crawler – that also handles the pathes of FV8
Install required Python packages
pip install -r scripts/requirements.txt
export DOCKER_BUILDKIT=0
python scripts/vv8-cli.py setup

Results: Crawler setup should complete without any errors.

(E3): *[Higher Code Coverage] [30 human-minutes + 1 compute-hour + <50GB disk] Execute the FV8-modified Chromium and Node.js environments on the Tranco top 1k sites. For this evaluation, we do not use malicious datasets for the artifact assessment.*

Execution: # Run the FV8 Crawler to visit Tranco top X websites
python crawler_queue_tranco.py
To create full logs:
i) Modify 'crawler_queue_tranco.py' script:
from 'cmd = f"python3 ./scripts/vv8-cli.py crawl -pp Mfeatures –no-headless –show-chrome-log –disable-screenshot –disable-artifact-collection –disable-har –disable-gpu –disable-features=NetworkService –js-flags='–no-lazy' timeout -u url"'
to 'cmd = f"python3 ./scripts/vv8-cli.py crawl -pp Mfeatures –no-headless –show-chrome-log –disable-gpu –disable-features=NetworkService –js-flags='–no-lazy' timeout -u url"'
ii) Connect to the docker container:
'docker exec -it DOCKER_ID /bin/bash' where DOCKER_ID belongs to the 'vv8_log_parser_worker' container.

Results: You can check the docker environment for the created files and the forced executions that happened because of FV8. You can modify the 'tranco/tranco10.txt' part to 'tranco/tranco100.txt' or 'tranco/tranco1k.txt' to run the browser on more Tranco websites. Check tranco/* for more options.

A.5 Notes on Reusability

We emphasize that the artifact evaluation aims to demonstrate FV8's usability rather than its performance on specific datasets. FV8 is a versatile tool applicable in various scenarios and fully customizable. By creating a proof of concept for a particular set of APIs and validating its functionality, the research community can adapt both the list of APIs and the datasets it operates on. FV8 can also be extended to analyze other JavaScript-based environments beyond the scope of this paper, such as MongoDB or Electron, using the same V8 patches. Researchers can leverage FV8 to investigate other evasion techniques and apply forced execution strategies across different JavaScript ecosystems. Hotcrp Stable URL: https://sec24winterae.usenix.hotcrp.com/paper/98.

A.6 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/.