USENIX Security '25 Artifact Appendix: The DOMino Effect: Detecting and Exploiting DOM Clobbering Gadgets via Concolic Execution with Symbolic DOM

Zhengyu Liu, Theo Lee, Jianjia Yu, Zifeng Kang, and Yinzhi Cao
{zliu192, imeal1, jyu122, zkang7, yinzhi.cao}@jhu.edu
Johns Hopkins University

A Artifact Appendix

A.1 Abstract

In the paper, we present Hulk, the first dynamic analysis framework to automatically detect and exploit DOM Clobbering gadgets. Hulk operates in three phases. First, it performs dynamic taint analysis in the browser to track values from attacker-clobberable sources to sinks. Next, it leverages the recorded taint traces to construct *Symbolic DOM*, a formalized representation for defining and resolving DOM-related constraints in gadgets, and generates corresponding HTML payloads. Finally, Hulk injects the generated payloads into target webpages to verify their exploitability.

This artifact provides the source code, datasets, and instructions to reproduce the evaluation results, namely, detecting all zero-day gadgets in the *dom-clobbering-collection* and additional ones from the Tranco Top 5,000 websites.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The artifact poses no security or privacy risks to artifact reviewers, testing websites, or their users. The tool identifies DOM Clobbering vulnerabilities and verifies them with non-destructive proof-of-concept exploits. On the *dom-clobbering-collection* benchmark, all tests are conducted locally, ensuring no impact on real users or live systems. During large-scale evaluations on live websites, the artifact exclusively analyzes publicly accessible client-side scripts. It does not interact with server-side interfaces, user data, or website functionalities.

A.2.2 How to access

The artifacts and the *dom-clobbering-collection* benchmark are publicly accessible at the following links. We also provide the Zenodo.org link for persistent access.

- https://github.com/jackfromeast/TheHulk
- https://github.com/jackfromeast/dom-clobbering-collection
- https://zenodo.org/records/15054169

A.2.3 Hardware dependencies

There is no specific hardware features required for this artifact evaluation. During our research, we performed the experiments on an Intel(R) Core(TM) i9-14900K CPU with 192 GB RAM and 100 GB of disk space.

A.2.4 Software dependencies

The experiments require Google Chrome, Node v18.18.2, and Python $\geq$ 3.10. In our setup, we used Google Chrome version 126.0.6478.126, installed at `/usr/bin/google-chrome` on Ubuntu 24.04.1 LTS. We recommend using the same version for consistency. Other modern versions may also work but are not guaranteed. All additional software dependencies can be installed during the Set-up step.

A.2.5 Benchmarks

To evaluate the functionality and reproducibility of this artifact, we provide the following two benchmarks:

- The *dom-clobbering-collection* benchmark. This dataset includes 28 zero-day gadgets discovered by Hulk from widely used client-side libraries. Each gadget is documented by a README file detailing the vulnerable code and a proof-of-concept (PoC).
- The large-scale benchmark. This dataset consists of 497 zero-day gadgets identified by Hulk across 354 domains from the Tranco Top 5,000 websites. This benchmark is used to evaluate Hulk's effectiveness in detecting gadgets on real-world live websites at scale.

A.3 Set-up

A.3.1 Installation

- Step 1: Clone our GitHub repository recursively using the following command: `$ git clone -recursive`

`https://github.com/jackfromeast/TheHulk.git`

- Step 2: Navigate to the project root and run the dependency installation script:

  `$ cd TheHulk && ./install.sh`

- Step 3: Configure the network proxy for HTTP(S) traffic to `http://127.0.0.1:8899`. This step is crucial as the experiment requires the browser to route its traffic through a man-in-the-middle (MITM) proxy. For detailed instructions on configuring a network proxy, refer to the Ubuntu official guide[1].

A.3.2 Basic Test

To make sure that the setup is ready and well configured, we provide a command to run Hulk against our motivating example as a basic test:

```
$ ./tasks/ae-run-basic-check/run.sh
```

After running the command, the results will be stored under directory: `tasks/ae-run-basic-check/output/domain/url-hash/crawler/`, where `domain` is `127.0.0.1`, and `url-hash` is the hash value of the tested URL. The taint analysis results are stored in `taintflows.json` and the generated exploits are stored in `exploit.txt`.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): We present a dynamic taint analysis tool that can detect DOM Clobbering gadgets at runtime and generate detailed taint traces. This claim is supported by Experiment (E1), where we show Hulk can detect all the gadgets in the *dom-clobbering-collection* benchmark.

(C2): We introduce a *Symbolic DOM*-based method to automatically generate exploitable HTML markups from taint flow records. This claim is supported by Experiment (E2), where we show Hulk is able to generate valid exploits of all the gadgets on the *dom-clobbering-collection* benchmark.

(C3): Our approach is capable of detecting and exploiting DOM Clobbering gadgets on live websites at scale. This claim is supported by Experiment (E3), where we show Hulk can detect and exploit all the zero-day gadgets from Tranco Top 5,000 websites.

A.4.2 Experiments

All experiments of Hulk are driven by scripts in the `task` directory. Each subdirectory under `task` corresponds to a specific task, where users can define inputs, outputs, and configurations using `.yaml` files. To reproduce the experiments, refer to the tasks prefixed with `ae-`, as these are pre-configured for artifact evaluation.

(E1): *[Gadgets Detection] [3 human-minutes + 10 compute-minutes]: Running Hulk on the dom-clobbering-collection benchmark to generate detailed taint flow records.*
Preparation: No additional preparation required. For environments with a display (check with `echo $DISPLAY`) , one can set `chrome.HEADLESS` to `false` in `config.browser.yml` under the same directory to visualize the running.
Execution: Run gadget detection on *dom-clobbering-collection* benchmark with following command:

```
$ ./tasks/ae-gadget-detection-e1/run.sh
```

Results: After the task completes, the results are available under `ae-gadget-detection-e1/output` directory. The overall summary is located at `tasks/ae-gadget-detection-e1/AE-E1-Timestamp/127.0.0.1/taintflows_summary.json`, where all 28 cases are expected to be successful. Detailed taint flows for each gadget can be found at `AE-E1-Timestamp/127.0.0.1/url_hash/crawler/taintflows.json`.

(E2): *[Exploit Generation and Verification] [3 human-minutes + 30 compute-minutes]: Running Hulk to generate and verify the exploits for the gadgets in the dom-clobbering-collection.*
Preparation: In the configuration file: `tasks/ae-exploit-generation-e2/config.browser.yml`, update the `others.EXPLOIT_INPUTS_FOLDER` with the path of the output directory of E1, which is `tasks/ae-gadget-detection-e1/AE-E1-Timestamp`.
Execution: Run exploit generation and verification with the following command:

```
$ ./tasks/ae-exploit-generation-e2/run.sh
```

Results: After the task is completed, the results are available under the `ae-gadget-detection-e2/output` directory. Each web page folder should contain two newly generated files: `exploit.txt` and `verified_exploit.txt`. The former contains the generated exploits, and the latter contains the verified exploits. Each web page is expected to have at least one verified exploit in `verified_exploit.txt`.

(E3): *[Large-scale Testing] [5 human-minutes + 7 compute-hours]: Running Hulk on the large-scale benchmark to detect the zero-day gadgets that Hulk found from the Tranco Top 5,000 websites.*
Preparation: No additional preparation required.
Execution: The large-scale test encompasses the entire pipeline of Hulk, including gadget detection, exploit generation, and verification. The execution should be conducted in the following steps:

[1] https://help.ubuntu.com/stable/ubuntu-help/net-proxy.html.en

First, run the command to detect taint flows:

`$ ./tasks/ae-gadget-detection-e3-1/run.sh`.

Next, update the `others.EXPLOIT_INPUTS_FOLDER` in `tasks/ae-gadget-detection-e3/config.browser.yml` with the path to the output directory generated in the previous step (same as E2).
Finally, run the command to generate exploits and verify the gadgets:

`$ ./tasks/ae-gadget-detection-e3-2/run.sh`.

Results: After the task completes, the results are available under the `ae-gadget-detection-e3-2/output` directory. Note that the results may differ slightly from the original results (i.e., the results we obtained between July 2024 and August 2024) due to regular updates on live websites or patches applied after our disclosure.

A.5 Notes on Reusability

The major components of Hulk—the dynamic taint analysis tool and the *Symbolic DOM*-based exploit generator—are designed to facilitate future client-side security testing. The dynamic taint analysis tool is extensible, allowing users to customize taint configurations (e.g., sources, sinks, and propagation rules). The exploit generator enables researchers to define DOM-related constraints and automatically generate corresponding HTML markups.

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/usenixsec2025/.