Design and Development of a Custom Browser Fuzzing Tool for Detecting Security Vulnerabilities

---

Dhiaeddine Kraini

Content

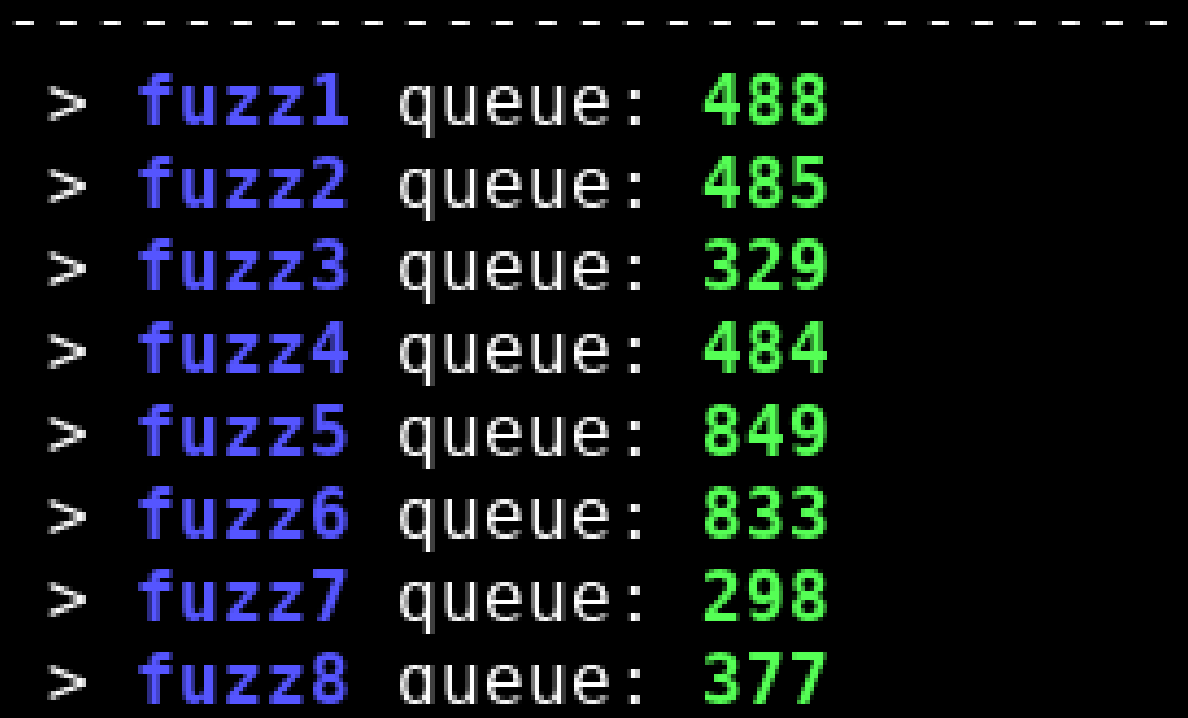
- → Problem Statement
- → Aim & Objectives
- → Research Background
- → Methodology
- → Achievements
- → Examples
- → Bibliography & URLs

The Vulnerability Landscape

- Modern Browsers: Constantly process untrusted inputs (HTML, CSS, JavaScript), making them susceptible to attacks such as cross-site scripting, memory corruption, and more.
- High Impact: Browser vulnerabilities can lead to severe security breaches, impacting millions of

Limitations of Existing Tools:

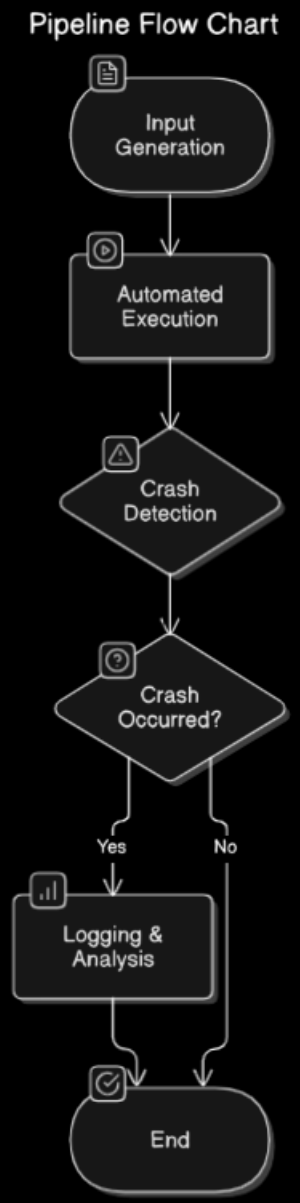
- Domato: Uses a seed-based approach for DOM fuzzing and Limited mutation strategies that can miss subtle edge cases.
- AFL (American Fuzzy Lop): Excellent for simple applications but struggles with the complexity of modern browser engines and Inefficient in exploring deep code paths due to the lack of integrated browser-specific feedback.



Aim & Objectives

- Aim: Develop an extensible fuzzing tool tailored for browser engines that uncovers security vulnerabilities efficiently.
- Objectives:

1. Realistic Input Generation:Use a mix of seed-based and mutation strategies to create HTML, CSS, and JavaScript test cases
2. .Automated Execution:Integrate with ASan-enabled browsers for enhanced memory error detection.Utilize Selenium for seamless browser automation.
3. Crash Logging & Analysis:Capture detailed crash logs (e.g., use-after-free, buffer overflows).Provide a structured output for security researchers to analyze.



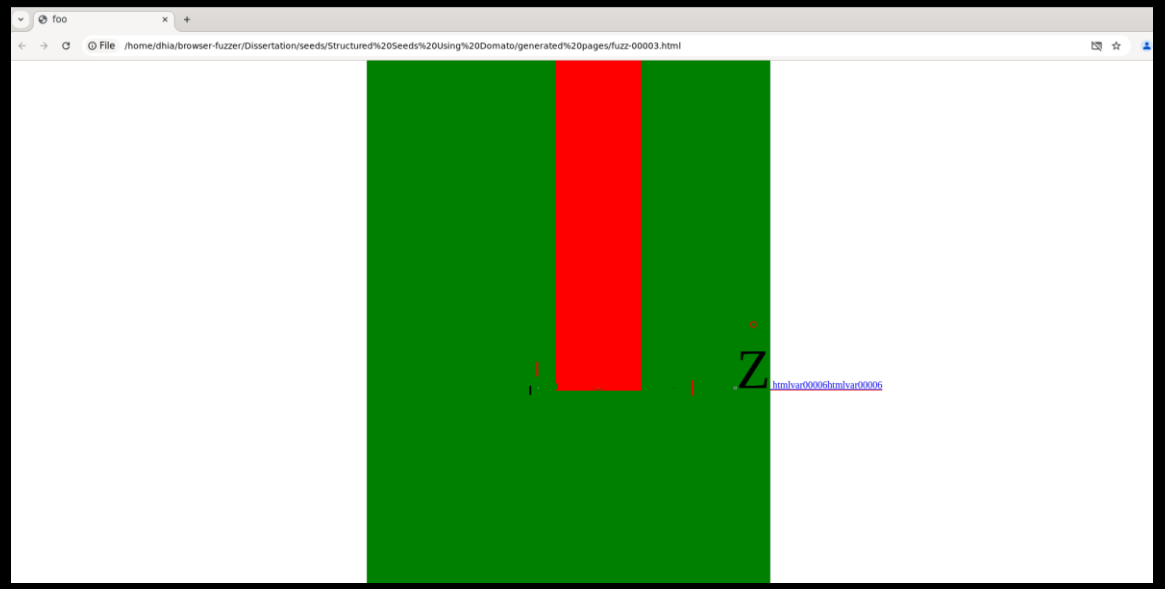
Research Background

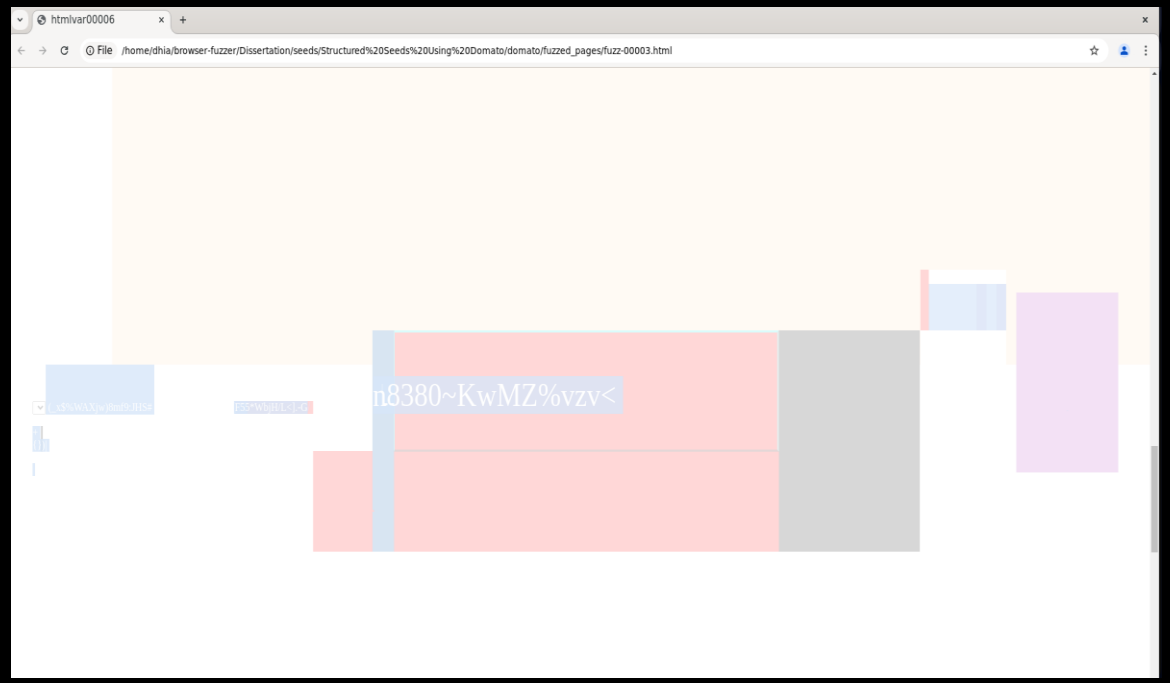
Domato (Google Project Zero):

- Significance: Demonstrated the potential of DOM fuzzing for discovering vulnerabilities.
- Lesson Learned: Relying solely on seed-based mutation limits coverage; expanding mutation strategies can expose more edge cases.

Coverage-Guided Fuzzing (AFL & AFL++):

- Significance: Introduces feedback loops where code coverage guides further mutation.
- Lesson Learned: Incorporating coverage information can help in dynamically augmenting the test corpus, leading to more efficient vulnerability detection.





Methodology

Phase 0

Setup and Preparation

- Configure ASan-enabled browsers (Chrome, Firefox).
  Set up Selenium for headless and automated testing.
  Initialize version control using Git for reproducible experiments.

Phase 2

Corpus Generation and Mutation

- Input Sources:
  Leverage Web-Platform-Tests and Domato for initial seed files.
  Mutation Engines:
  Radamsa: Introduces random mutations to mimic real-world corrupted inputs.
  AFL++: Provides coverage-guided mutations to efficiently explore execution paths.

Phase 1

Requirements & Architecture Design

- Define the system architecture with modular components for input generation, execution, and logging. Plan integration points for future expansion (new mutation engines).

Phase 3

Crash Detection and Logging

- Automated Crash Detection:
  Implement a system that continuously monitors browser execution for crashes and abnormal behavior using ASan.
- Log Collection & Analysis:
- Feedback Loop:
  Utilize crash data to refine input generation strategies, enhancing overall testing effectiveness.

Achievements

Technical Progress:

Seed Generation: Over 500 seed files produced using Domato and Web-Platform-Tests.

Mutation Success: More than 1,200 mutated test cases generated via Radamsa.

Integration of AFL++: Implemented coverage-guided fuzzing, enabling deeper exploration of code paths.

Environment Setup & Automation:

Headless Browser Setup: Configured headless versions of Chrome and Firefox with AddressSanitizer (ASan) enabled, allowing efficient and automated testing without a graphical user interface.

Selenium Integration: Developing a robust Selenium framework to automate browser interactions. This setup for now seamlessly launches headless browsers.

Bibliography

•Domato (Google Project Zero): “Domato: DOM Fuzzing Tool,” GitHub. [Online]. Available: https://github.com/googleprojectzero/domato

•Zalewski, M. (2014). American Fuzzy Lop: M., American Fuzzy Lop: A Security-Oriented Fuzzer, 1st ed., Google Inc., 2014