# SmartFaultInjector: LLM-Driven Kernel Fault Injection and Testing Framework

**Jiahuan Liu** [1] **Chen Liang** [1 2] **Jiahao Zhao** [1] **Lili Zhang** [1 2]

## Abstract

In Linux kernel development and testing, fault injection(Natella et al., 2016) techniques play a crucial role. Kernel developers need an effective method to simulate and test kernel behavior under various abnormal conditions. Traditional hardware fault injection methods are costly and complex operationally, typically requiring specialized equipment while affect hardware lifespan.

We propose **SmartFaultInjector**, combining Extended Berkeley Packet Filter(eBPF) and Large Language Models(LLMs)(Hoffmann et al., 2022; Kaplan et al., 2020) for fault injection to enable hardware fault simulation and any kernel subsystem fault injection in kernel space. The `kprobe_override` feature of eBPF enables precise interception and manipulation of kernel functions, SmartFaultInjector leverages the `kprobe_override` feature of eBPF, which allows us to override the return value of a specific function in a hardware driver within the kernel, enabling fault injection from the kernel to the hardware. Kernel developers can use SmartFaultInjector to simulate hardware faults in kernel space without any modifications to the hardware, and perform in-depth analysis of kernel functions based on LLM to generate fault injection strategies, significantly enhancing kernel testability.

The architecture of the system is shown in Figure 1. SmartFaultInjector consists of Target Scanner, Comprehension Engine, Fault Injection Module, and Positioning Cause. Target Scanner is responsible for analyzing driver code, scanning frequently called parts in combination with LLM, and serving as candidates for fault injection. Comprehension Engine utilizes LLM to conduct in-depth analysis of target functions based on their logical structure and context(Brown et al., 2020), generating fault specification strategies. Fault Injection Module is responsible for seamlessly integrating fault injection functionality into the existing system based on eBPF technology. Positioning Cause is an independent observability module used to analyze changes in system behavior after fault injection, providing feedback for the next iteration.

Compared to traditional fault injection methods, SmartFaultInjector has the following advantages:

- **Minimal Cost:** Developers only need to write eBPF code to simulate hardware faults in kernel space, no specialized equipment is required for testing.
- **No Impact on Hardware:** Fault injection occurs in kernel space, and no actual modifications are made to the hardware.
- **Automated Fault Injection Strategy Generation:** Combining LLM for in-depth analysis of kernel functions to generate fault injection strategies automatically.

Based on this framework, we conducted simulated card drop tests on our internally developed network card, verifying its effectiveness in kernel space fault injection. In the future, we will build a pipeline system that automatically generates test programs for newly emerging driver modules, and automates the generation of eBPF code for fault injection(Zheng et al., 2024) enabling the `kprobe_override` feature.

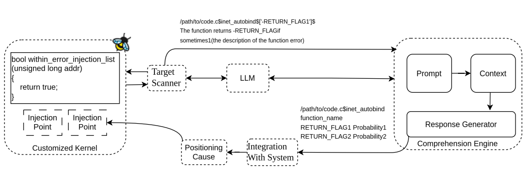


*Figure 1.* SmartFaultInjector Architecture

*Equal contribution [1]Xi'an University of Posts and Telecommunication, China [2]Shaanxi Key Laboratory of Network Data Analysis and Intelligent Processing. Correspondence to: Chen Liang <mumulc@xupt.edu.cn>.

*Proceedings of the $8^{th}$ MLSys Conference*, Santa Clara, CA, USA, 2025. 

## Acknowledgements

This work was supported by ZTE Industry-University-Institute Cooperation Funds under Grant No. IA20221221020, Undergraduate Training Programsfor Innovation and Entrepreneurship of ShaanxiProvince No. S202411664170 and No. S202411664179, and the Tencent's T-Spark Program.

## References

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. Training compute-optimal large language models, 2022.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models, 2020.

Natella, R., Cotroneo, D., and Madeira, H. S. Assessing dependability with software fault injection: A survey. *ACM Comput. Surv.*, 48(3):44:1–44:55, February 2016. ISSN 0360-0300. doi: 10.1145/2841425. URL http://doi.acm.org/10.1145/2841425.

Zheng, Y., Yang, Y., Chen, M., and Quinn, A. Kgent: Kernel extensions large language model agent. In *Proceedings of the ACM SIGCOMM 2024 Workshop on EBPF and Kernel Extensions*, eBPF '24, pp. 30–36, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400707124. doi: 10.1145/3672197.3673434. URL https://doi.org/10.1145/3672197.3673434.