Nleak: Automatic Memory Leak Debugging in Node.js

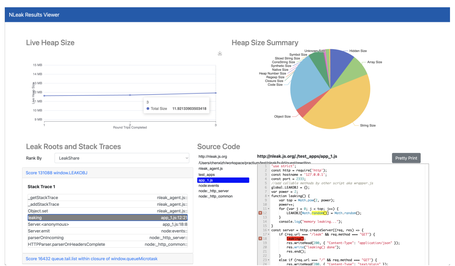
Students: Canchen Li, Chenxi Zhang, Pranathi Alla, Pronoy Roy, Qingyang Shi, Wenting Yeh, Yizhou Liu
Faculty advisor: Hanan Hibshi
Sponsor: Max Dobler, Kripa Ravivarman, Pinda Ndaki, Adobe, Inc.

Introduction

Memory leaks may cause a system to slow down or crash. If an attacker can intentionally trigger a memory leak, the attacker may be able to launch a denial-of-service attack or take advantage of other unexpected program behavior. JavaScript memory leaks are tricky and often time-consuming to identify and fix, as JavaScript is dynamically typed and leaks are fundamentally different from leaks in traditional C, C++, and Java programs. It is a daunting task even for experienced expert developers to effectively identify and fix memory leaks. Our team worked with Adobe to build NLeak, a memory detection tool to automate companies' attempts to locate, diagnose, and rank JavaScript memory leaks in Node.js applications.

Results

NLeak viewer is a tool built in React.js that allows you to visualize the heap snapshot growth of your application. To use it, simply go to https://nleak-viewer.vercel.app/ and upload your nleak_result.json file. A screenshot ofNLeak viewer is provided below.

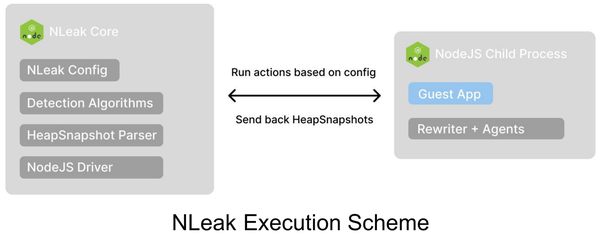


NLeak Viewer Screenshot

Once you've uploaded the file, NLeak viewer will generate a chart showing the growth of your heap snapshots over time. You'll also see a summary of the last heap snapshot's size, as well as the leak location with source map.

Methodology

NLeak adopts the core algorithm from BLeak [1] for memory leak identification. The algorithm takes a series of JavaScript runtime heap snapshots during idempotent operations in the guest application. Memory leaks are identified as heap objects gaining more outgoing references across the heap snapshots.
When leak-related objects are gathered, NLeak re-executes the guest application to gather the related JavaScript stack trace. An illustration of this process is provided below.



NLeak Execution Scheme

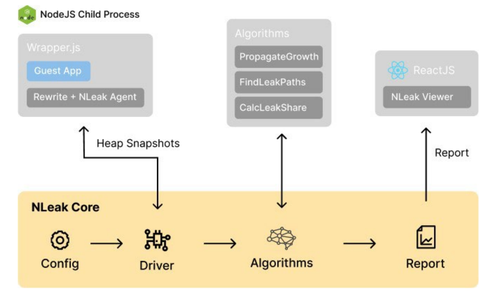
Clients only need to provide a configuration file along with their guest application for executing Nleak. NLeak executes the guest application as one of its child processes in inspection mode. Heapsnapshots are captured via the Chrome debugger protocol. Guest application rewrites are applied in the leak diagnosis phase to obtain the reference stack trace for possible memory leak objects. The rewrite makes object references traceable in the guest application's execution and is integrated into the system seamlessly with a customized module import and compile scheme.

[1] J. Vilk and E. D. Berger, "BLeak: automatically debugging memory leaks in web applications," in Proceedings of the 39thACM SIGPLAN Conference on Programming Language Design and Implementation, 2018.

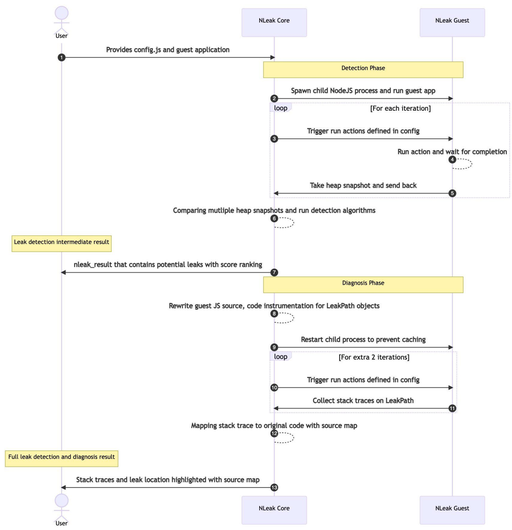
Future Work

- Work on support recursive rewriting and code instrumentation, so that NLeak can run on a full codebase folder instead of single file.
- Setup a use case for integrating NLeak to CI, and expose the NLeak result as API or hooks for data visibility. (e.g., Grafana, Github/Gitlab, etc)
- Expand the tool from Node.js to other JavaScript runtime environments such as Deno, Cloudflare workers.

System Design



NLeak Memory Leak Detection Pipeline



NLeak System Design Sequence Diagram

Scan QR code to view demo

Carnegie Mellon University
Information Networking Institute

CAE IN CYBERSECURITY COMMUNITY