Secure Systems Group, Aalto University

Alexander Colb, Philip Ginzboorg (Aalto University)

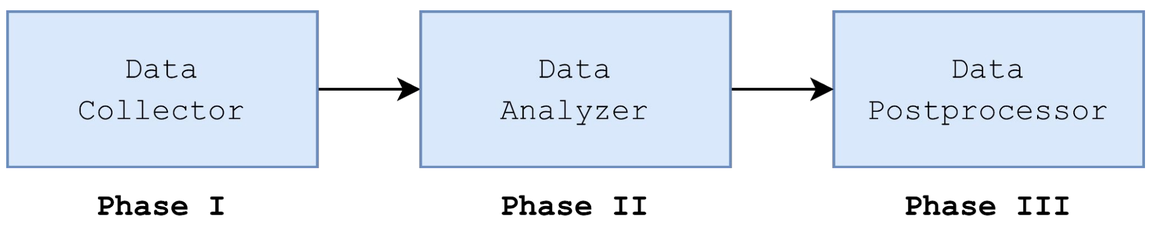
Mapping the Attack Surface Hidden in Unauthenticated JavaScript — An Analysis of Finland's Critical Infrastructure

Background

- **Legacy Web Sites**
  - Most logic on the back end
  - Fetched new HTML for each user interaction
- **Single Page Applications (SPAs)**
  - Fetch an initial HTML document, further actions without refreshing page
  - Most application logic run in-browser (JavaScript code)

Objectives

- Compare *static* crawling of JavaScript to (headless-browser based) *dynamic* crawling
- Map out front-end attack surface from statically crawled JavaScript via static analysis
- Identify front-end attack surface from a sample of Finnish critical-infrastructure websites



Experiment

- Developed a three-stage pipeline
  - I: Static JavaScript collection
  - II: Static attack-surface analysis†
  - III: Results post-processing
- Published an intentionally vulnerable SPA
- Tested 100 web applications from Finnish critical infrastructure

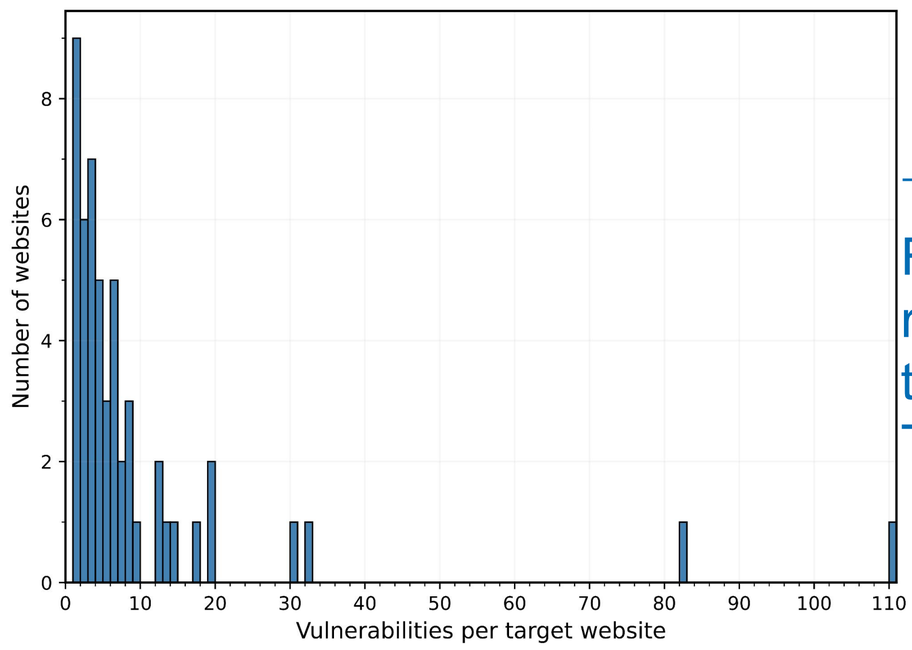
† *taint analysis (hybrid property graphs) for vulnerable flows [1], source-map discovery, vulnerable-dependency detection [2], LLM-assisted analysis, fingerprinting of vulnerable functions [3]*

Crawler comparison

- In 46% of targets, ours was better
- In 19% of targets, theirs was better
- In 35% of targets, equal coverage

Attack-surface results

- 33/95 targets with **vulnerable dependencies**
- 24/95 targets with with **exposed source maps**
- 3/95 targets with **DOM-clobbering flows**
- 15/95 targets with **open-redirect flows**



→ Findings reported to Traficom.

Sector	Targets		Vulns.
	total	vulnerable	
Healthcare & Pharmaceuticals	10	5	130
Transportation & Logistics	8	4	95
Education & Research	10	6	61
Food & Agriculture	10	7	61
International Relations	9	6	40
Government & Public Administration	10	4	40
Energy & Utilities	10	4	29
Manufacturing & Industry	10	5	27
Finance & Banking	10	8	25
Communications & Media	8	3	7
Total	**95**	**52**	**515**

Our analysis shows that Finnish organizations should improve their software lifecycle processes. We found static crawling to be a viable JavaScript extraction method and that high-quality attack-surface data can be collected using static methods.

Aalto University

[1] S. Khodayari et al., “Do (not) follow the white rabbit: Challenging the myth of harmless open redirection”, Feb. 2025. doi: 10.60882/cispa.28606865.v1

[2] Retire.js, Accessed 2025-12-8, 2025. [Online]. Available: https://github.com/RetireJS/retire.js

[3] M. Kluban et al., “On detecting and measuring exploitable JavaScript functions in real-world applications”, ACM Trans. Priv. Secur., vol. 27, no. 1, 8:1–8:37, Feb. 2024. doi: 10.1145/3630253