



Socket

CASHEWS: SOURCE PREPROCESSING FOR LLM-BASED MALICIOUS PACKAGE DETECTION

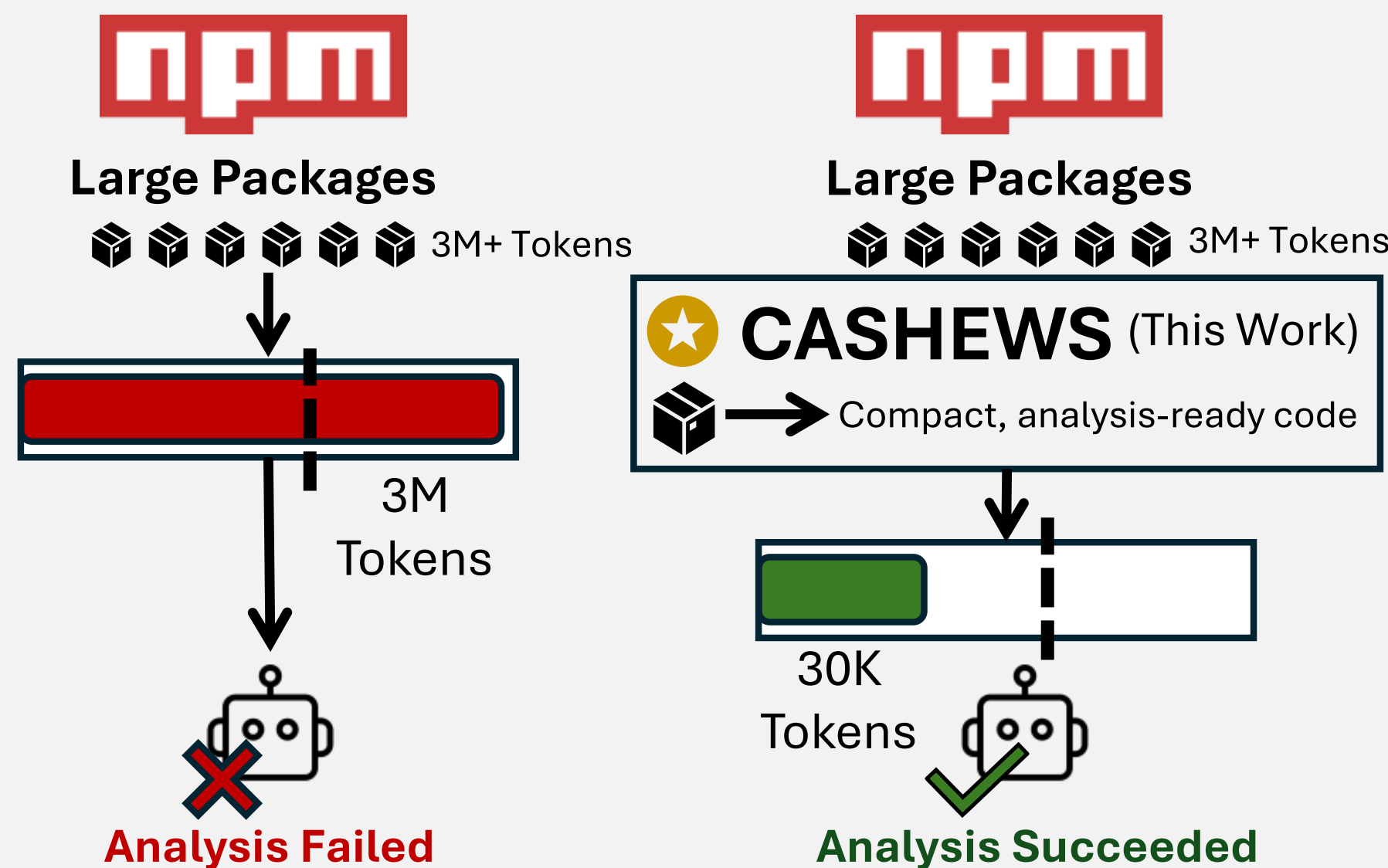
Jean-Charles Noirot Ferrand¹, David Adei², Anders Møller³, Alexandros Kapravelos⁴

¹ University of Wisconsin–Madison, ² Socket



MADS&P

OVERVIEW



- Threat actors embed **malware** in npm packages, leading to increasingly prevalent **supply-chain attacks**
- Malicious package **detectors** increasingly **use LLMs** whose weaknesses are **exploited** through source files
- We build a **source preprocessor** that converts large files into compact, analysis-ready code

MOTIVATION

Real Attack: ai-sdk-ollama

- 100K+ weekly downloads**; malicious file is **4.5 MB** → **skipped**

```
eval(function(s, n) {
  return s.replace(/[/[a-zA-Z]/g, function(c) {
    var b = c <= "Z" ? 65 : 97;
    return String.fromCharCode(
      (c.charCodeAt(0) - b + n) % 26 + b
    )
  })
})([40, 105, 97, ..., 41, 40, 41]
.map(function(c) {
  return String.fromCharCode(c)
}).join(""), 18))
```

1.3M-integer array, decoded and run with eval

Large Files Break Detectors

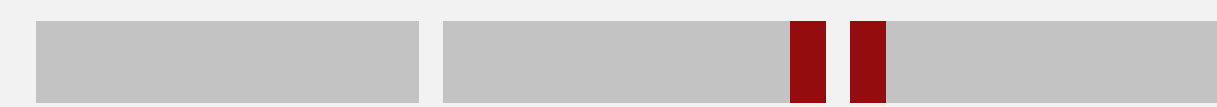
Skip — never analyzed



Truncate — payload past the cutoff is lost

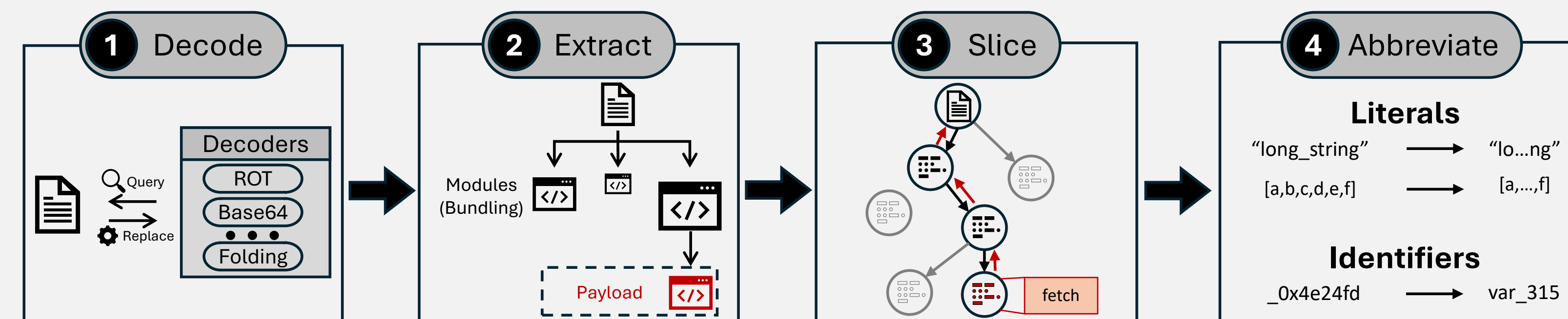


Split — payload broken across calls



.... context window ■ payload ■ analyzed

METHODS



Processing Steps

- 1. Decode:** reveal hidden code
 - Iteratively undo obfuscation (string arrays, Base64, ROT, AES) until nothing changes or a limit is reached
- 2. Extract:** split into analyzable units
 - Recover **bundled modules** and **JS payloads** hidden in strings
- 3. Slice:** keep security-relevant code
 - Keep only statements reaching **sinks** (exec, network, eval...)
- 4. Abbreviate:** reduce context and mitigate prompt injections
 - Rename long identifiers, **elide** long literals

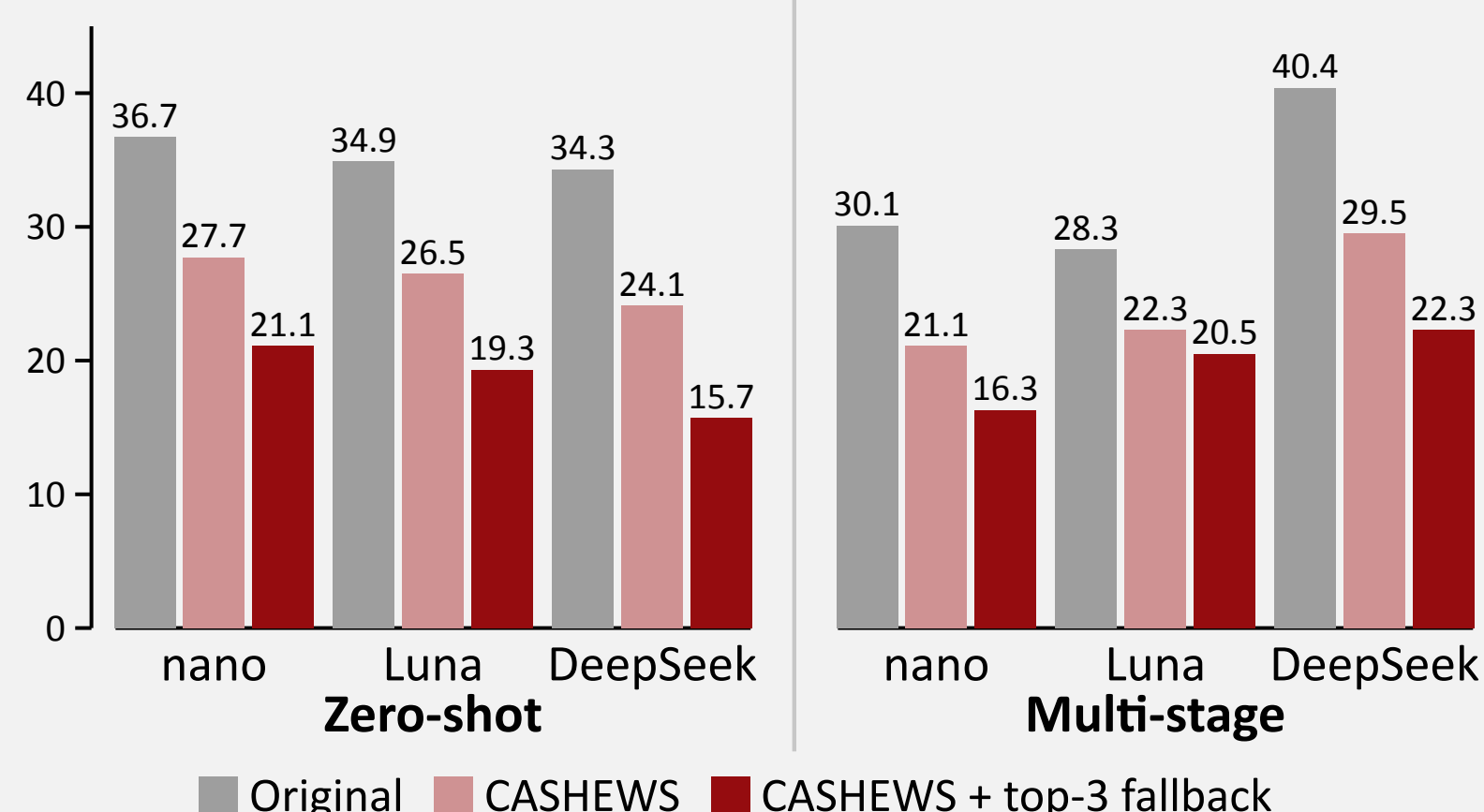
Setup

- Packages:** 512 npm package files over 25K tokens (166 malicious, 346 benign)
- Detectors:** zero-shot, multi-stage
- Models:** GPT-5 nano, GPT-5.6 Luna, DeepSeek V4 Flash
- Inputs:**
 - Original source
 - CASHEWS
 - CASHEWS + top-3 units fallback (ranked by sources and sinks)

EVALUATION

Effectiveness

False-negative rate (%) ↓



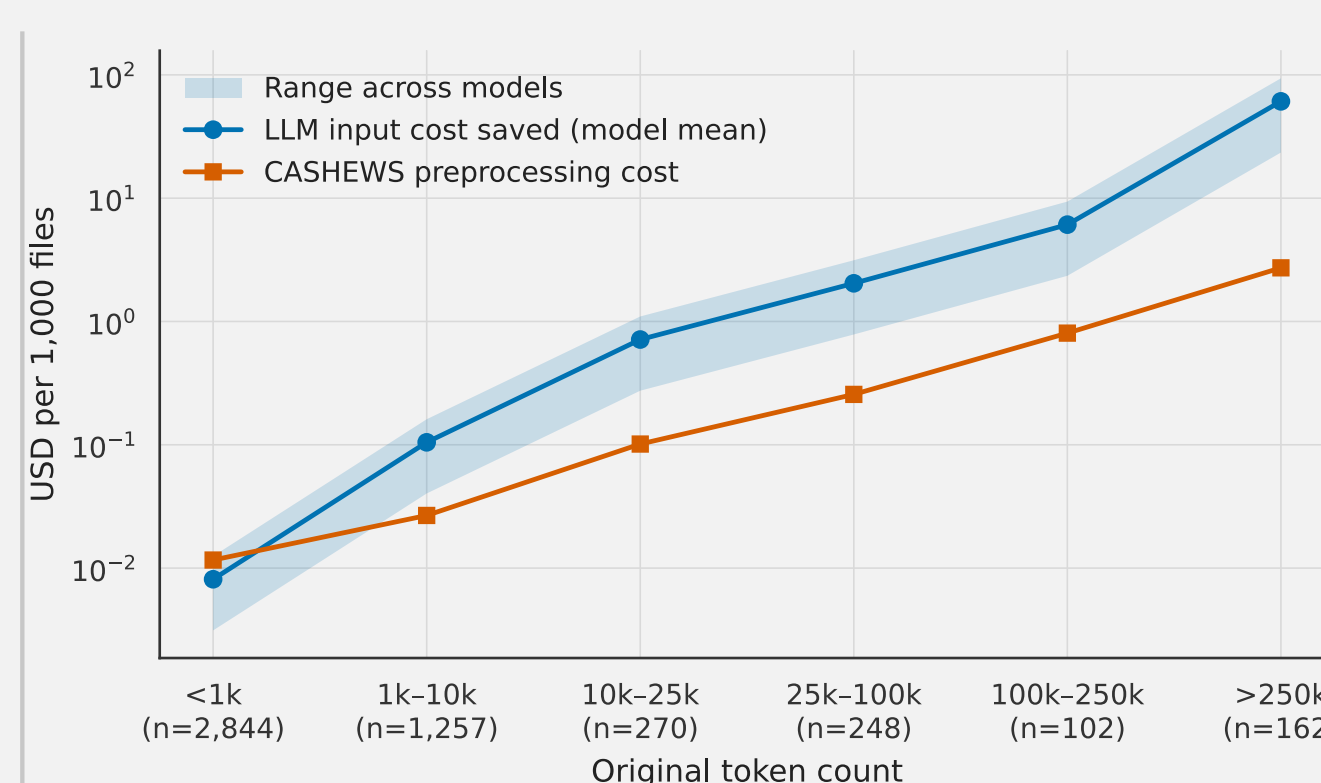
- 69–86% to **99–100%** of files analyzed
- FNR:** down **7.8–18.6 pp** on every setting
- FPR:** stays ≤ **2.9%**

Workflow	Model	Coverage (%)	FPR (%)	Token cost [†]
Zero-shot	GPT-5 nano	69.5 → 100.0	1.2 → 2.9	\$1.63 → \$1.17 (−28%)
	GPT-5.6 Luna	69.7 → 100.0	0.0 → 0.0	\$6.26 → \$4.42 (−29%)
	DeepSeek V4 Flash	69.7 → 100.0	0.0 → 0.9	\$4.67 → \$3.32 (−29%)
Multi-stage	GPT-5 nano	69.1 → 98.8	1.2 → 2.6	\$8.84 → \$7.61 (−14%)
	GPT-5.6 Luna	85.7 → 99.4	0.9 → 0.9	\$50.37 → \$36.19 (−28%)
	DeepSeek V4 Flash	69.1 → 100.0	0.0 → 0.9	\$14.97 → \$11.02 (−26%)

Original → **CASHEWS + top-3 fallback**. [†]Cost on files analyzable under all inputs.

Scalability

- Token cost savings exceed preprocessing cost for files **over 1K tokens**
- Net **−34.6%** cost after preprocessing
- ~30 s** for >25K tokens
~3 min for >250K tokens



Ablation

- Abbreviate** and **selection of extracted units** drive token reduction
- Decode** dominates time but enables later steps

Step	Files changed	Token reduction (%)		Time (s)	
		P50	P90	P50	P90
Decode	456 (89.1%)	0.3	7.3	17.8	261.3
Extract [‡]	244 (47.7%)	43.3	98.9	—	—
Slice	140 (27.3%)	0.6	23.4	8.5	21.6
Abbreviate	470 (91.8%)	19.3	75.7	0.6	5.2

Bars: median reduction. [‡]Share of tokens removed by keeping the top-3 units.

TAKEAWAYS

CASHEWS

- CASHEWS enables **higher coverage** for almost no classification cost
- Such preprocessing is **scalable** to the analysis of the npm ecosystem
- Each step is bounded, preventing **denial-of-service** by adversaries

Deployment Considerations

- New bundlers, JS features, and obfuscators** need to be taken into account as the ecosystem evolves
- Bundled modules reappear, which makes their analysis **reusable**

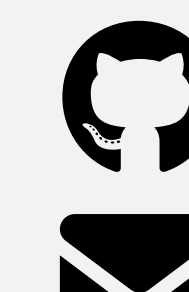
Future Work

- Robustness against **adaptive attacks** that target the preprocessing
- Improvements of the **precision and soundness** of the slicing step
- Tune** the preprocessing **to the analysis** (classification vs. behavior)

Contact



<https://jcnf.me>



jcnf0

jcnf@cs.wisc.edu