Hi-Res: Precise Exploit Detection using Object-Granular Memory Monitoring

Ziyang Yang
Rice University
ziyang.yang@rice.edu

Saumya Solanki
Serenitix
saum@serenitix.io

Scott Rixner
Rice University
rixner@rice.edu

Nathan Dautenhahn
Dartmouth College
nathan.d.dautenhahn@dartmouth.edu

Presented by Nathan Dautenhahn

Problem: Exploits violate gaps to find unintended *weird machines*

- Programmer specifies what they think is right
- But may miss some dynamic context leading to weirdness
- Exploits operate within legal kernel operations
- They violate assumptions: scope, lifecycle, access context
- Kernel lacks runtime enforcement of object semantics

2

Gap: detection is low-resolution and bypassable

- Detection systems develop models from events
- Typically, low-level or opaque, lacking context
- Lacks visibility into the *weird machine* layers
- Leads to mimicry

3

Hypothesis: weirdness is weird and should be visible with the right context

Hypothesis: weirdness is weird and should be visible with the right context

If it's used as a duck, it's a duck. If it quacks like a duck, it's a duck.

Solution: lift opaque, low-level traces into behavioral grammars

- Object-Sensitive: Hi-Res tracks memory at object granularity using lexical scopes as types
- Context-Sensitive: Accesses lifted into tuples: syscall, access IP, alloc IP, call stack
- Programmable: dynamic contexts can be selectively explored
- Behavioral Grammars: Valid programs = stable grammar; exploits = violations

Design: Trace Lifting Pipeline

1. Instrument allocation and memory access
2. Maintain address-to-object map
3. Capture syscall, call stack, context per access
4. Construct tuple space for fingerprinting
5. Develop grammars out of behavioral patterns for both benign and exploit

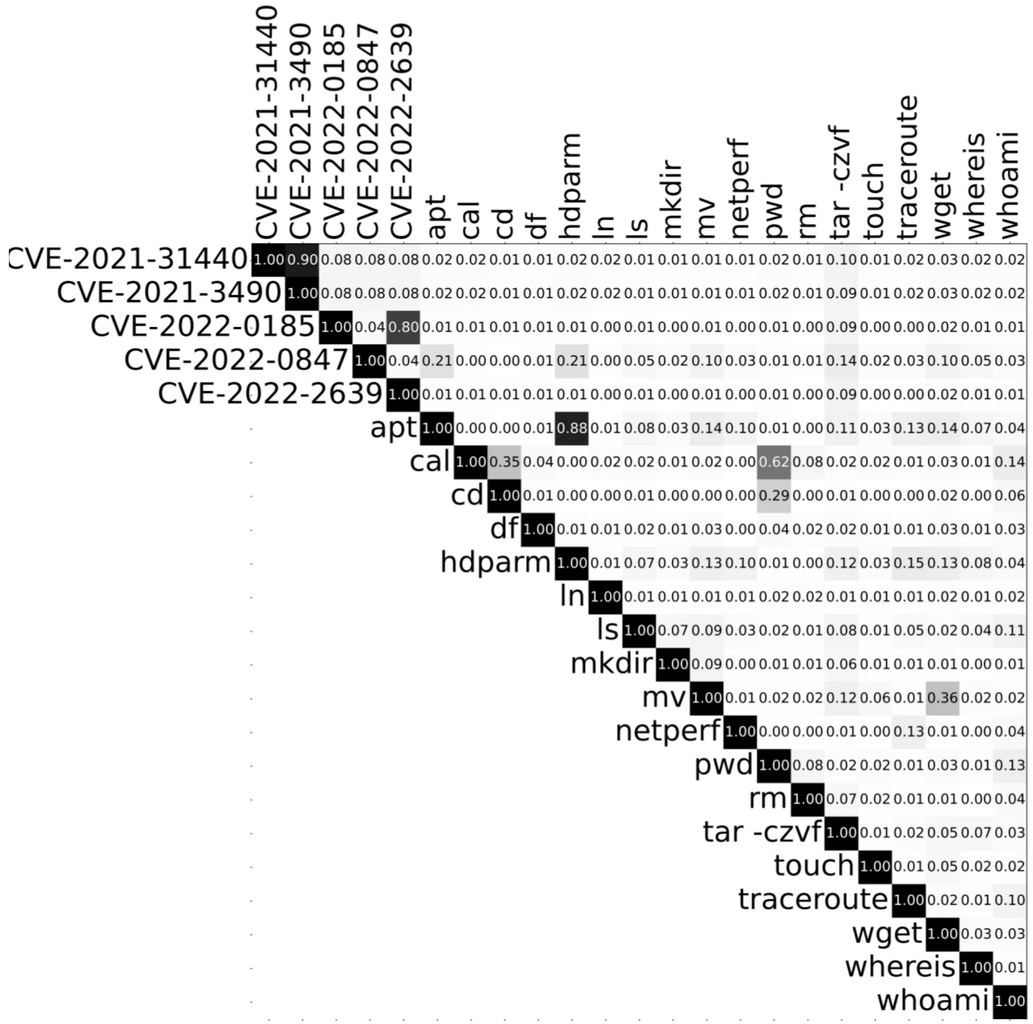
6

Challenges: Semantic Inference Without Labels or Ground Truth and Efficient

- Memory is semantically opaque
- Semantics must be inferred, not declared
- Kernel instrumentation must be efficient and selective
- Detection must be general and interpretable

Results: CVEs are easy to see with object granular access grammars

- 5 kernel exploits, 20 normal workloads
- Exploit traces show sparse, irregular fingerprints
- Hi-Res separates classes with no false positives



(i) (syscall, *access_ip*, *alloc_ip*, callstack)

8

LangSec Perspective

- Hi-Res defines a runtime language over memory use
- It surfaces violations as syntactic outliers
- Not anomaly detection—semantic enforcement

9

Takeaways

- **Hi-Res**, a programmable in-kernel framework for detecting kernel exploits via object-granular memory monitoring
- A method for **lifting low-level memory traces into structured behavioral grammars**, enabling detection of semantic violations without requiring labeled training data
- A lightweight, page-table-based mechanism for **per-process monitoring** that avoids global kernel instrumentation overhead
- An **empirical evaluation** demonstrating great potential for prece, general, and not terrible overhead in identifying exploit behavior

10

Fini!

11