SniP: An Efficient Stack Tracing Framework for Multi-threaded Programs

MSR 2022

Arun KP\textsuperscript{1} \quad Saurabh Kumar\textsuperscript{1} \quad Debadatta Mishra\textsuperscript{1} \quad Biswabandan Panda\textsuperscript{2}

\textsuperscript{1}Indian Institute of Technology, Kanpur

\textsuperscript{2}Indian Institute of Technology, Bombay
Multi-threaded Program Stack Tracing

- Stack captures the state of a program.
Multi-threaded Program Stack Tracing

- Stack captures the state of a program.
- Trace and analysis approach for stack
  - Requires dynamic run-time techniques to trace stack.
  - Needs to know stack range to filter stack specific accesses for analysis.
Multi-threaded Program Stack Tracing

- Stack captures the state of a program.

- Trace and analysis approach for stack
  - Requires dynamic run-time techniques to trace stack.
  - Needs to know stack range to filter stack specific accesses for analysis.

- Challenge for multi-threaded program
  - Identifying thread’s stack range.
Multi-threaded Program Stack Tracing

- Stack captures the state of a program.
- Trace and analysis approach for stack
  - Requires dynamic run-time techniques to trace stack.
  - Needs to know stack range to filter stack specific accesses for analysis.
- Challenge for multi-threaded program
  - Identifying thread’s stack range.
**SniP Design**

- OS extension (Monitor) captures thread’s stack range information.
SniP Design

- OS extension (Monitor) captures thread’s stack range information.
- Pin tool uses this stack range info and records only stack accesses in trace file.
SniP Design

- OS extension (Monitor) captures thread’s stack range information.
- Pin tool uses this stack range info and records only stack accesses in trace file.
- Driver program coordinates & manages tracing.
Benefits of SniP in Multi-threaded Program Stack Tracing
Trace File Size

- Reduces file size for long running applications. — ~ 98% reduction for TD.
- Marginal reduction (~ 6% for MS) in file size for short running, heavy stack usage applications.

[ MS: Merge-Sort, HS: Python3 Http Server, DT: Decision Tree Classifier, BD: BabyDBM, CD: CacheDBM, TD: TinyDBM, G500: Graph500 BFS ]
Y-axis is in log scale
Tracing Time

- SniP benefits long running applications. — \( \sim 96\% \) reduction in time for TD.

- Marginal benefit (\( \sim 2\% \) reduction for MS) for short running, heavy stack usage applications.

[Y-axis is in log scale]

[ MS: Merge-Sort, HS: Python3 Http Server, DT: Decision Tree Classifier, BD: BabyDBM, CD: CacheDBM, TD: TinyDBM, G500: Graph500 BFS ]
Example Use Cases of SniP
Tracing ML Classification Algorithms

- Studied stack read-write access pattern of popular ML algorithms.

- As an example, in Decision Tree Classifier, reads dominated writes.
Detecting Uninitialized Memory in Stack

- Analysed program with uninitialized memory bug.
- Identified instances where read from stack happened before write.

Parser output of uninitialized memory bug

```json
{
    "0x7fffffffe2d8": [
        "0x7ffff7ac8a0c"
    ],
    "0x7fffffffe2f8": [
        "0x7ffff7ac8913"
    ],
    "0x7fffffffe308": [
        "0x7ffff7ac8927"
    ],
    "0x7fffffffe310": [
        "0x7ffff7ac8928"
    ],
    "0x7fffffffe318": [
        "0x7ffff7ac8929"
    ],
    "0x7fffffffe2d8": [
        "0x7ffff7ac8a0c"
    ],
    "0x7fffffffe2f8": [
        "0x7ffff7ac8913"
    ],
    "0x7fffffffe308": [
        "0x7ffff7ac8927"
    ],
    "0x7fffffffe310": [
        "0x7ffff7ac8928"
    ],
    "0x7fffffffe318": [
        "0x7ffff7ac8929"
    ]
}
```
Conclusion

• Program stack tracing is key in gaining insights, exposing security loopholes in applications.
• We introduced SniP, an efficient stack tracing framework for run-time tracing of multi-threaded application stack.
• SniP combines Intel’s Pin with an intelligent OS extension, reducing trace file size and tracing time.
• SniP can be easily adapted for vast variety of use cases.
For more details contact:
Arun KP
kparun@cse.iitk.ac.in