UNIVERSITY OF CAMBRIDGE
Department of Computer Science and Technology

Comprehensive and Practical Security for Program Binaries

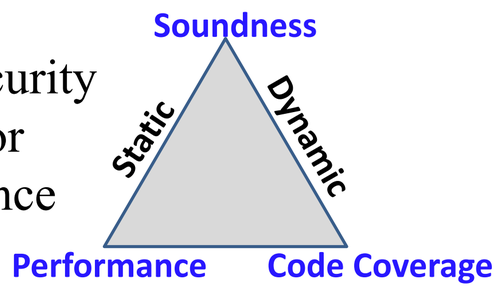
Mahwish Arif

Supervisor: Timothy M. Jones

Background

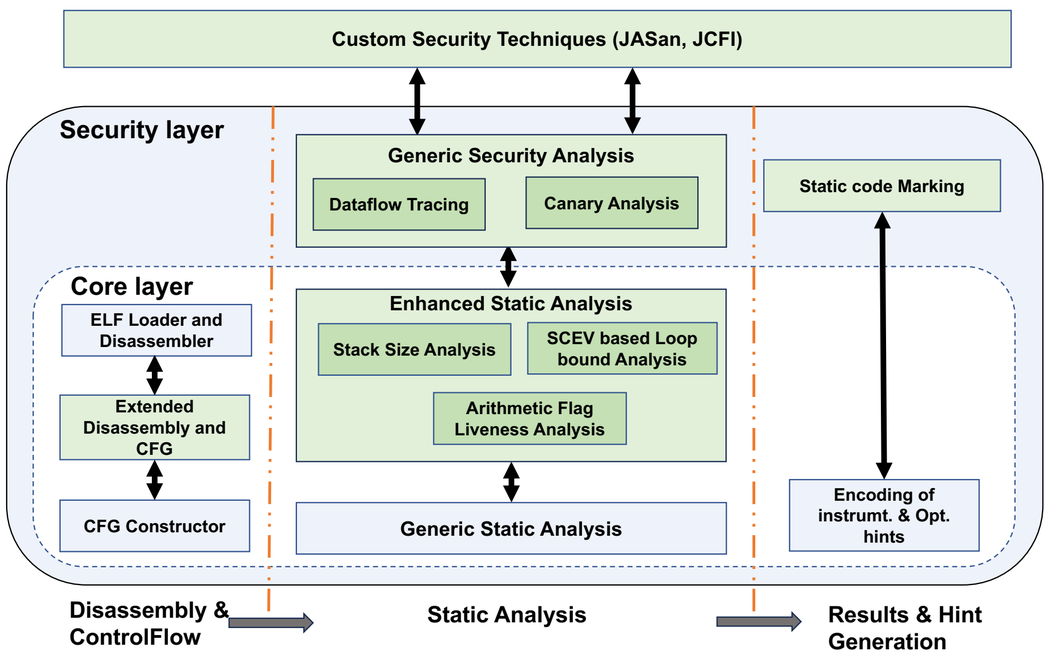
Comprehensive security can only be ensured if all the code that an application executes is protected: any unprotected code, including shared-object library code, becomes a potential attack surface.

Existing static or dynamic binary security tools either lack in code coverage[1] or soundness[2], or incur high performance overhead[3].

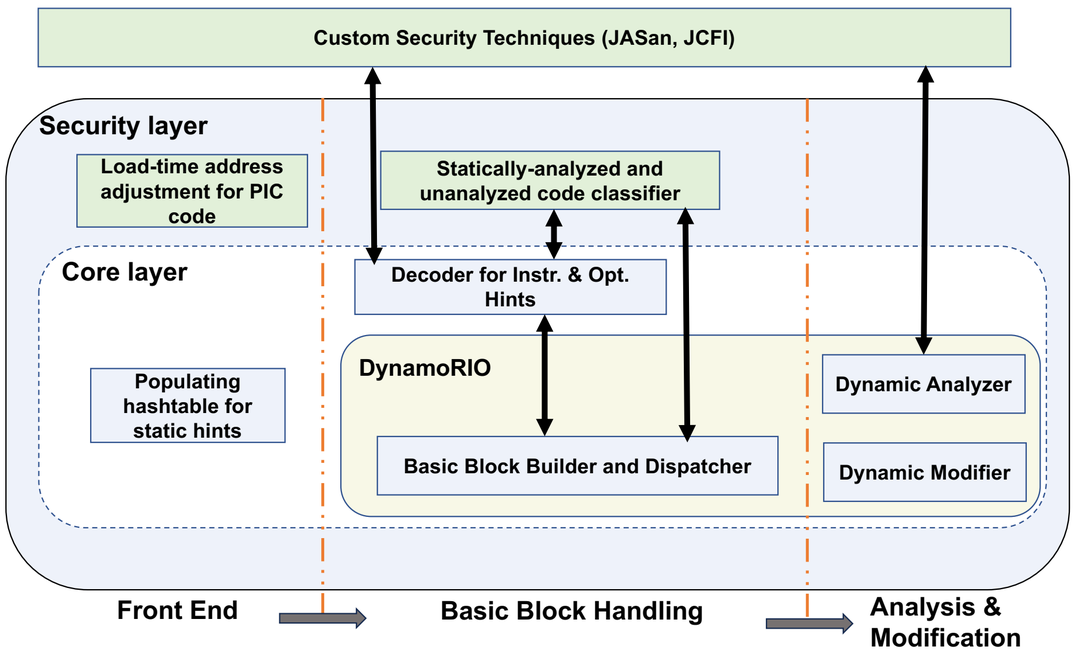


Janitizer: A Hybrid Binary Mechanism for Security

- Complex cross-block static analysis to *offset runtime overhead* for static code (PIC/non-PIC)
- Optimisation or transformation hints passed on to dynamic translator

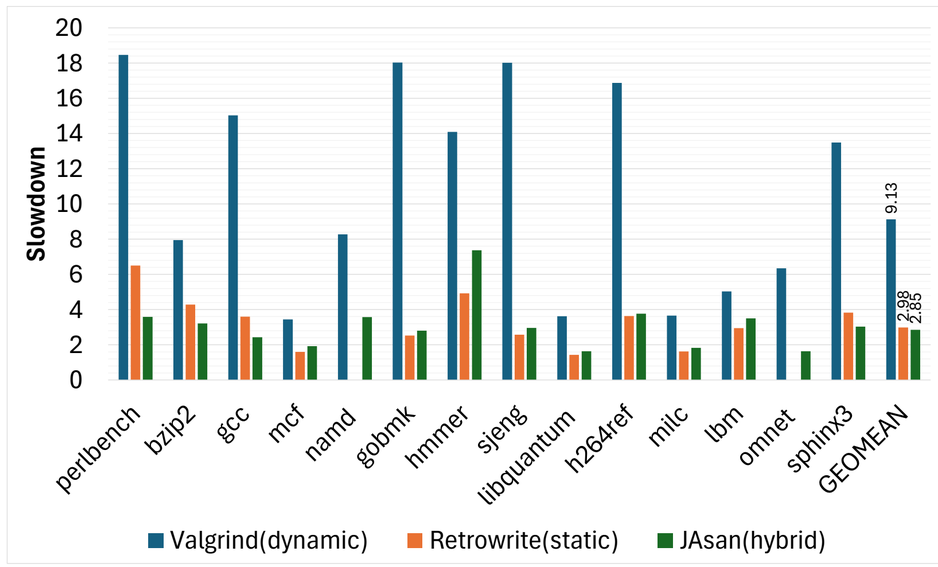


- Weaker/simpler dynamic analysis for statically unseen/unavailable code (e.g. dynamically-loaded libraries, dynamically-generated code)
- Instrument/transform at runtime to *ensure soundness*



Evaluation

For evaluation of Janitizer, we provide hybrid binary implementations of
a) memory address sanitiser – **JASan**
b) forward and backward Control-flow Integrity (CFI) scheme – **JCFI**
We use different static analyses such as liveness, scalar evoultion and cross-block control-flow target analysis to improve performance and precision of these scheme.



Performance overhead of binary address sanitizer implementations (compared to native implementation) on SPEC CPU2006 – lower is better

benchmark	lockdown	JCFI	binCFI
perl	97.74	99.87	97.89
bzip2	86.85	99.19	99.37
gcc	99.49	99.9	98.34
mcf	83.37	99.45	99.25
gobmk	98.81	99.93	99.2
hmmer	87.95	99.78	98.61
sjeng	87.23	99.73	99.1
libquantum	85.6	99.56	98.89
h264ref	94.26	99.76	99.52
milc	91.34	99.69	98.65
lbm	82.47	99.35	
sphinx3	91.46	99.78	98.64
GEOMEAN	90.37	99.66	98.83

Average Indirect-target Reduction (AIR) of lockdown(dynamic), JCFI (hybrid) and binCFI(static) on SPEC CPU2006 – higher is better

Conclusion

Our hybrid binary mechanism, *Janitizer*, provides comprehensive code coverage equivalent to that of high-overhead dynamic techniques, while maintaining performance levels of low-coverage static techniques.

References

[1] S. Dinesh *et al.*, "Retrowrite: Statically Instrumenting COTS binaries for Fuzzing and Sanitization", S&P'20
[2] M. Zhang *et al.*, "Control Flow Integrity for COTS Binaries", USENIX Sec'13
[3] N. Nethercote *et al.*, "Valgrind: A framework for heavy-weight dynamic binary instrumentation," PLDI'07

Email: mahwish.arif@cl.cam.ac.uk

Computer Architecture Group