DECAF++ (Dynamic Executable Code Analysis Framework): Elastic Whole-System Dynamic Taint Analysis

Ali Davanian, Zhenxiao Qi, Yu Qu, and Heng Yin

University of California, Riverside

HAILONG JIANG

Sept. 3rd , 2019

OUTLINE

- Background
- QEMU
- DECAF
- DECAF++
- Summary
- Review

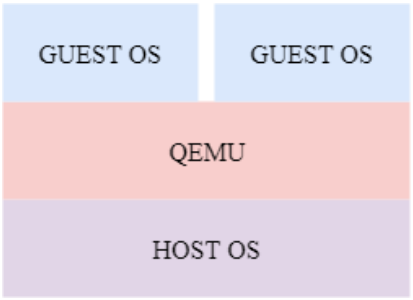
1 BACKGROUND

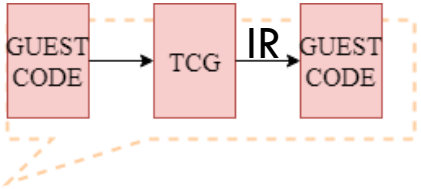
- Dynamic taint analysis (also known as dynamic information flow tracking) marks certain values in CPU registers or memory locations as tainted, and keeps track of the tainted data propagation during the code execution.
  - Process-level: Pin tools
  - Whole-system level a wider analysis scope
    - a better transparency
    - tamper resistance.
- main barrier:
  - prohibitive performance degradation
  - semantic gap between the analysis code and the program being analyzed
  - architecture/OS specificity
  - Being user-mode only
  - lacking APIs, etc.

QEMU

- QEMU [1] is a machine emulator, virtual machine:
  it can run an unmodified guest (target) OS(such as Windows or Linux) and all its applications in a virtual machine.
- QEMU runs on several host OS such as Linux, Windows and Mac OS X. The host and guest CPUs can be different.

- Primary usage
  to run one operating system on another, such as Windows on Linux or Linux on Windows.
- Another usage
  debugging
  because the virtual machine can be easily stopped, and its state can be inspected, saved and restored.



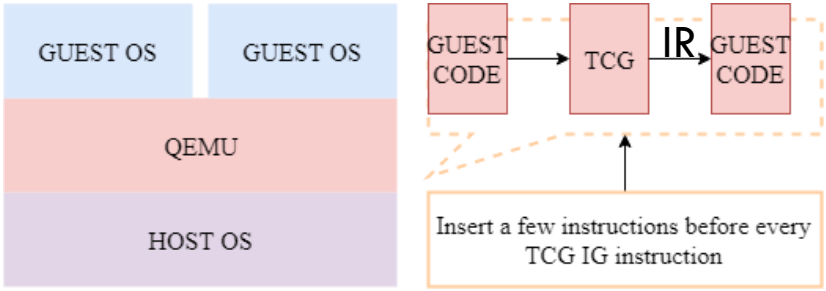


TCG: tiny code generator

[1]Qemu, a fast and portable dynamic translator. (Fabrice Bellard, 2005)

DECAF

- Dynamic binary analysis framework
  - virtual machine (QEMU) based
  - multi-target
  - whole-system
- DECAF (Dynamic Executable Code Analysis Framework) inserts a few instructions before every TCG IR instruction:
  - Read the taint status of the source operand.
  - Decide the taint status of the destination operand based on the instruction tainting rule.
  - Write the taint status of the destination operand to its shadow variable.




```
movi_i32    tmp0,$0x7000
mov_i32     esp,tmp0
movi_i32    tmp4,$0xe070
st_i32      tmp4,env,$0x4c
```

(a) Optimized Qemu generated tcg code

```
movi_i32    tmp23,$0x0
movi_i32    tmp0,$0x7000
mov_i32     taint_esp,tmp23
mov_i32     esp,tmp0
movi_i32    tmp24,$0x0
movi_i32    tmp4,$0xe070
st_i32      tmp4,env,$0x4c
```

(b) DECAF instrumentation for tainting

KEY COMPONENTS OF DECAF

- Just-In-Time VMI
  - This VMI component is able to reconstruct a fresh OS-level view of the virtual machine, including processes, threads, code modules, and symbols, to support binary analysis.
- Precise, lossless dynamic taint analysis
- Dynamic instrumentation management.
- Event-driven programming interface.
  - Compared to many of existing analysis frameworks [15, 16] that provide just the instrumentation interface, DECAF provides an event driven programming interface. .

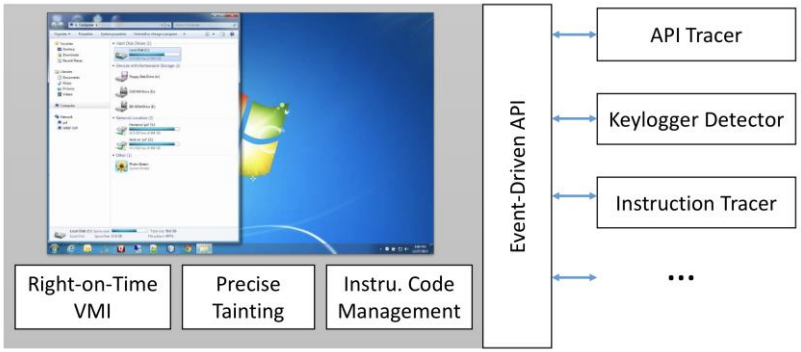


Fig 1 the overall architecture of DECAF

DECAF++

- DECAF incurs around 6 times overhead over QEMU which itself has another 5-10 times slowdown over the bare-metal hardware.
- This overhead for tainting is paid constantly no matter how much tainted data is actually propagated in the software stack.
- DECAF++ is the new version of DECAF Optimized with elastic taint propagation and elastic shadow memory access idea.

ELASTIC TAINT PROPAGATION

- Aims to remove the taint propagation overhead whenever possible.
  - It is based on the intuition that taint analysis can be skipped if the taint analysis operation does not change any taint status value. Taint analysis operations can possibly result in a change only when either of the source or the destination operand of an instruction is tainted

- **Two modes**
  - *track mode:* with taint propagation operations
  - *Check mode:* without taint propagation operations At any given time

the execution mode depends on whether any CPU register is tainted.

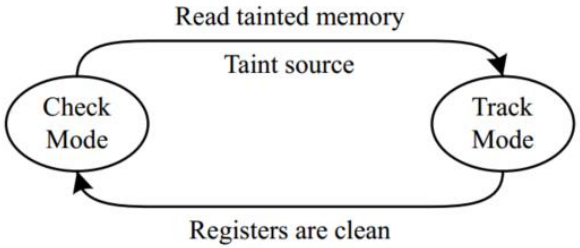


Figure 3: State transitions between check mode and track mode

ELASTIC TAINT STATUS CHECKING

- Main idea: to avoid unnecessary interactions with the shadow memory.
  - In DECAF, checking happens for every memory operation.
  - They can avoid the overhead per memory address if we perform the check for a larger set of memory addresses. Thus, if the larger set doesn't contain any tainted byte we can safely skip the check per address within that set. The natural sets within a system are physical memory pages.

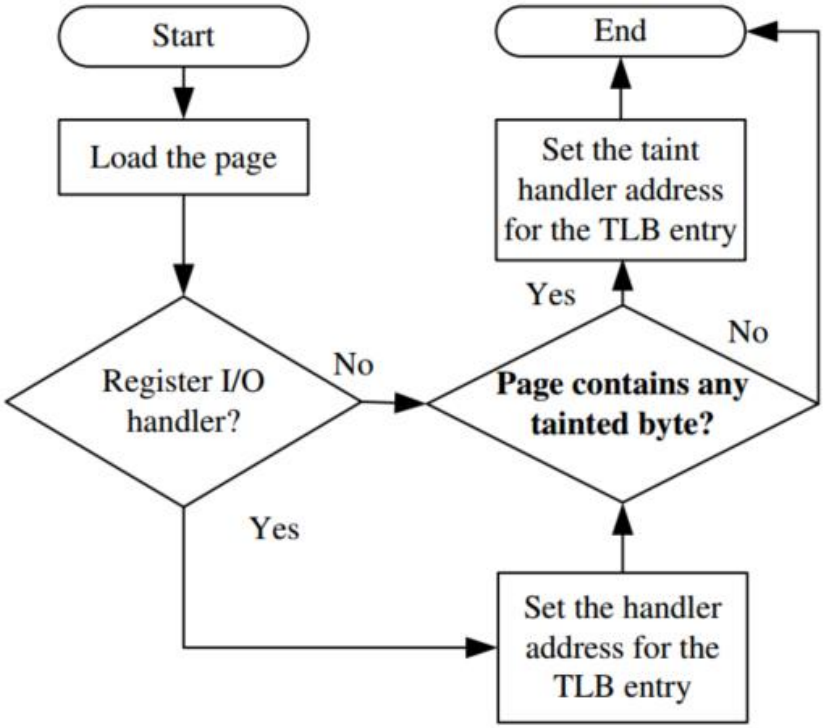


Figure 4: Fill TLB routine

TLB: Translation Lookaside Buffer

EVALUATION

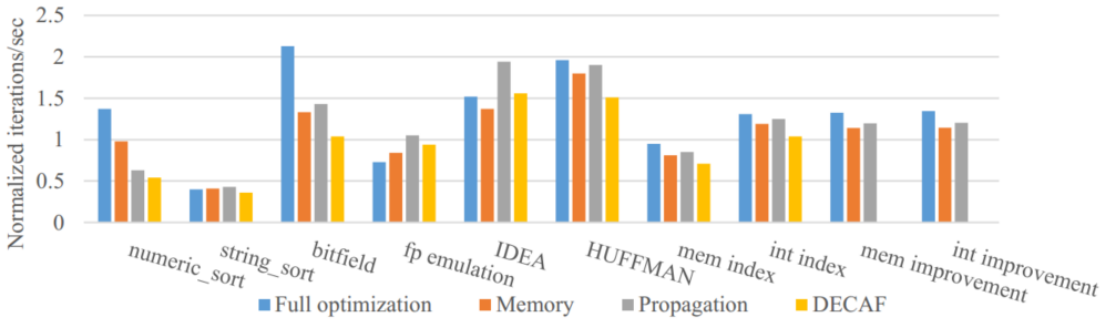


Figure 6: Comparing DECAF and DECAF++ performance.

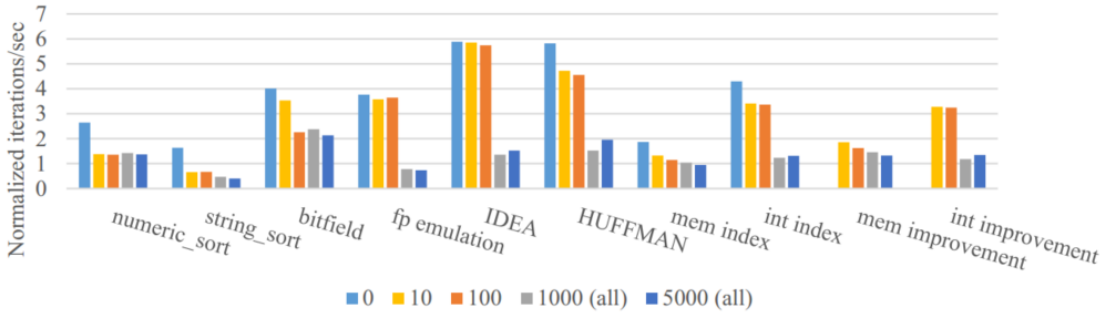


Figure 7: Evaluation of DECAF++ with full optimization under different number of tainted bytes; performance values are normalized by nbench based on a AMD k6/233 system.

SUMMARY

- Whole-system dynamic taint analysis framework
- The essence of DECAF++
  - *elasticity*
- Two independent optimizations to achieve the elasticity
  - elastic taint status checking: 2.6 times overhead
  - elastic taint propagation: 1.8 times overhead.

REVIEW

- Promotion from DECAF: Elasticity
- Whole-system dynamic taint analysis applications like intrusion detection systems and honeypots can greatly benefit from the elastic property
- Good idea for other taint analysis tools

THANK YOU

hjiang13@kent.edu

Primary Papers Included:

- DECAF++: Elastic Whole-System Dynamic Taint Analysis[C]

  Davanian A, Qi Z, Qu Y, et al.

  22nd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2019). 2020.

- QEMU, a fast and portable dynamic translator.

  Bellard, Fabrice.
  USENIX Annual Technical Conference, FREENIX Track. Vol. 41. 2005.

- Make it work, make it right, make it fast: building a platform-neutral whole-system dynamic binary analysis platform[C]

  Henderson A, Prakash A, Yan L K, et al.

  Proceedings of the 2014 International Symposium on Software Testing and Analysis. ACM, 2014: 248-258.

BACK UP