Decoupling dynamic program analysis from execution in virtual environments

Jim Chow    Tal Garfinkel    Peter M. Chen
VMware
Dynamic program analysis
Dynamic program analysis

Many useful analysis tools
- Discovering races: Intel ThreadChecker, Errat, HeGrind
- Finding bugs: Purify, Valgrind
- Checking security invariants: TaintCheck, TaintBooK, Program Shepherdling
- Profiling: VTune, DTrace

Dynamic program analysis

Can we do better?
Aftersight: an overview

Decoupling properties

Isolation:
- Analysis is self-contained: events, data self-generated
- Eliminates communication bottleneck
- Faults and regressions
- Add analysis in situ

Parallelism:
- Analysis and target can run separately
- Make analysis go faster
- Many analyses can run in parallel

Aftersight

Supported targets

Decoupling properties

Parallelism:
- Analysis and target can run separately
- Make analysis go faster
- Many analyses can run in parallel

Decoupling properties

Ex-post facto analysis

- Ex-post facto:
  - Analysis not known at the time of recording can be created and applied to prior runs
Challenging implications

Separating environments

- Low overhead, speed
- Important ability: different environments for recording and replaying
- Tuned to their task
- Rich analysis environment

Heterogenous replay

- Important ability: different environments for recording and replaying
- Tuned to their task
- Rich analysis environment

Heterogenous replay

- Modify VM replay: make it simpler
- Lump all devices into one big abstract device, never processes output

Hardware to software

- Interpretation execution: lack hardware performance counters on replaying end
- Emulate counters in software: emit translations to increment counts, check for event delivery
- Different virtual HW: replay log unusable

Heterogenous replay

- Modify VM replay: make it simpler
- Lump all devices into one big abstract device, never processes output
  - Input only
  - May not be as efficient
- Simple interface: interrupts, DMA, reply to MM reads, port IN

Abstract device
Benefits of heterogenous replay
- Use analyses impractical to do in VMM
- Lots of bugs: ESX (critical pre-production), Linux (old undiscovered), putty

Parallel analysis
- Made easy with recording
- Important capability: timeliness
- Compelling: performance (more cores)

Implications of parallel analysis
- Different rates
- Stacked odds
- Block target (low but not all lost)

Implications of parallel analysis
- Blocking is unavoidable, but with buffering it's possible we never have to do it
- Key: make analysis faster (play catch up)
- Slow analysis but keeps up (intuition: oracle)

Playing catchup
- Faster instructions: wait for interrupts (HLT)
- Speculation: memoize/forward cache

Keeping pace: web crawling
- No analysis
- Inline analysis
- Aftermath target VM analysis VM

Analysis
App
OS
VMM
Analysis tool
Analysis platform
Analysis
App
OS
VMM
Analysis tool
Analysis platform
Analysis
App
OS
VMM
Analysis tool
Analysis platform
Analysis
App
OS
VMM
Analysis tool
Analysis platform
Keeping pace: web crawling

- No analysis
- Inline analysis
- After target VM analysis

Keeping pace

- Analysis results can be timely
- Implication: must speed up analysis

Aftersight

- System presents decoupled analysis
- Shows heterogeneous replay as a method for extending scope of analysis tools
- Enables parallel analysis and examines its impact on analysis
- ... and more: synchronous safety, relogging, idletime boost, feedback modes, memoizing simulation...

Synchronous safety

- Security important part of some dynamic analysis
- Key attribute: block target before it does damage
- Synchronous systems block for mixed reasons
  - Deliver events/data, block target to prevent damage
- Asynchronous systems are better
  - At least as good, because blocking is a choice
  - Better, avoid blocking for irrelevant items: events/data
  - Key: choosing to block target before it does damage
  - In record/replay: most damage can be undone the only damage that can’t be undone is output to the outside world