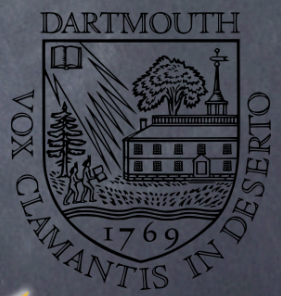
SegSlice: new primitives for trustworthy computing

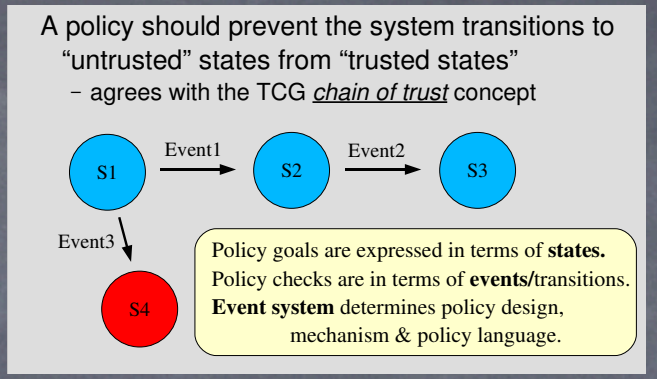
Sergey Bratus, PKI/Trust Lab,
Dartmouth College



Michael Locasto, George Mason U.
Brian Schulte, George Mason U.

GEORGE MASON UNIVERSITY

Policy model

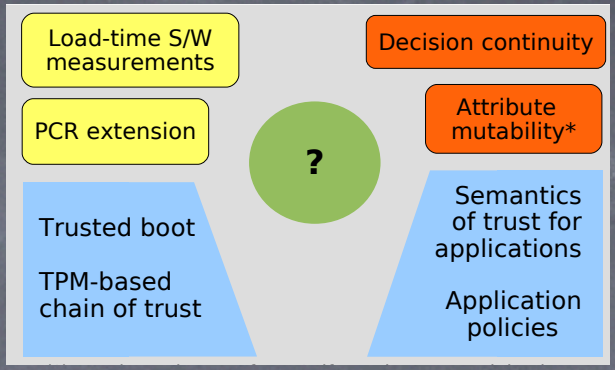


- Policy + mechanism =
  - definition of trustworthy system states (as derived from policy goals)
  - definition of events that cause state transitions (cf. [F.B.Schneider, 2000])
  - trapping and mediation of events that might cause transition to an untrustworthy state
    - re-measuring of system state, TPM ops

Vision

- Example: TPM-aware selective memory immutability (TRUST '08)
  - Accesses to selected RAM regions (and all page tables) trapped (Xen; wish: "MMU+")
  - Trap handler re-measures them, may call TPM's seal/unseal, or zeroize PCRs
- Position: Smarter MMUs for finer, faster trapping of memory events, more context than just "page read/written/fetched from" (FTC '08)

"Policy Gap"

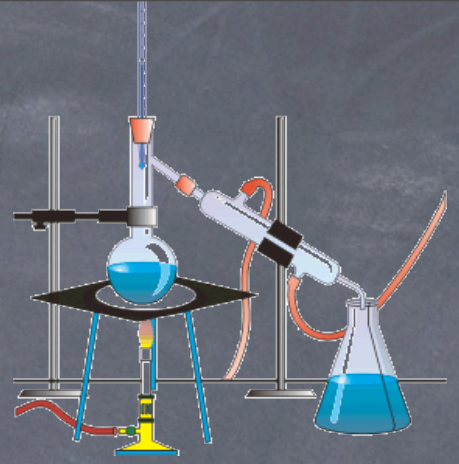


- The "Policy Gap" in TCG architecture:
  - Policy mechanisms do not allow developers to express their knowledge of expected or trustworthy behaviors, relative data value
  - Application-specific or special-purpose policy enforcement resembles debugging with predicates and actions to catch particular behavior
- Position: Policy requires developer knowledge about app internals, e.g., symbol tables

Platform Desiderata

- Developers should be given tools to express policy-critical behaviors of software at runtime, just like they are given tools for correctness-critical behaviors while debugging
- Systems should support Boolean logic-enhanced, object-granular, developer-friendly expressions of trappable conditions
- In the meantime, we should distill toolchain-supported & (x86-) implementable primitives

The distillation



- "Some thoughts on security after 10 years of q-mail 1.0" [DJ Bernstein, 2007]
  - Eliminating hidden data flow is more helpful than minimizing privilege
- "Exploit engineering principle": convert a hidden data flow to a control flow
  - code reading/writing data it is not meant to read/write

DJB quote

"I have become convinced that this 'principle of least privilege' is fundamentally wrong."

"Minimizing privilege might reduce the damage done by some security holes but almost never fixes the holes. Minimizing privilege is not the same as minimizing the amount of trusted code, does not have the same benefits as minimizing the amount of trusted code, and does not move us any closer to a secure computer system."

Code-data ownership

- Developer intuition: "This data unit is exclusively owned by this code unit"
  - Explicit scoping to catch hidden data flows at compile time (e.g., file-level "static")
- How many code/data units exist in binaries?
  - ELF sections (semantically different contiguous memory areas) -- about 30 in executable, could be > 70 in shared libs

Special Relationships

- A lot of information about which code units own which data units
  - .init & .fini with .ctors and .dtors
  - .plt with .got (and other runtime linking and relocation relationships)
  - .o file-scope relationships
  - Heap management code with heap (meta)data

“Lost at runtime”

“...And when the little sections woke up, it was Runtime, and no one could tell them apart, and no one cared about how special they were, or about their special relationships”

SegSlice model

- ELF executable image is divided into OS- and user-defined (GCC's _section pragma) slices
- CPL2 virtualizes user-level program segments
- Not optimal w.r.t. x86 32bit, but virtualization support on x86 started out much worse
- A full lattice of code-date units relations can be implemented (as in Bell-LaPadula or Biba)

Developers wanted!

- SegSlice extends explicit data-flow controls from the source code through ELF to loadable binary image
- Loaded image is instrumented with traps, incurring least performance loss
- x86 segmentation can be RE-optimized to avoid these losses
- Integrates naturally with TCG platforms: lattice violations cause TPM PCR ops, re-measuring and/or clearing PCRs

Thanks!