Reliably Exploiting Audit Logs

Chapman Flack*
Advisor: Mikhail J. Atallah

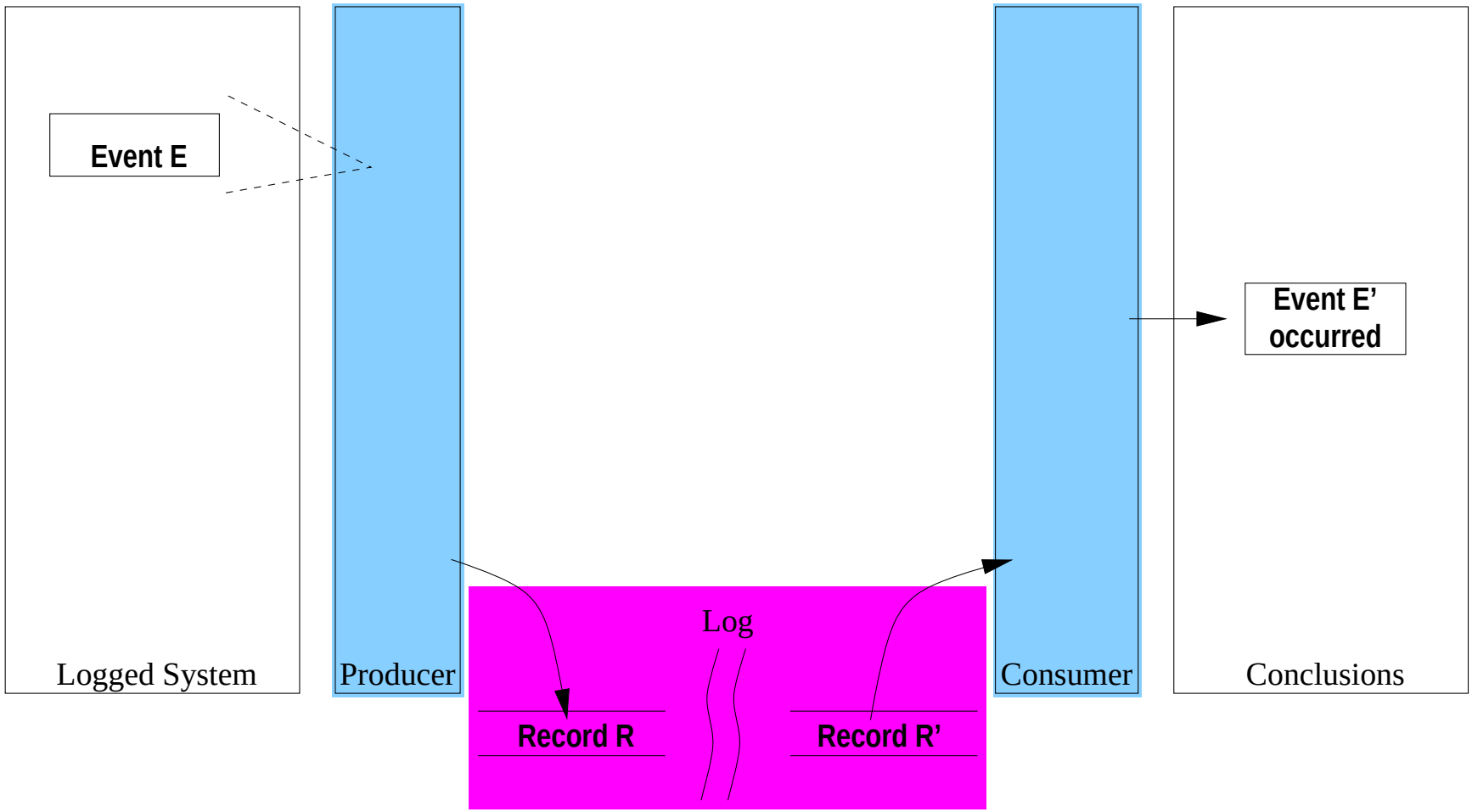
April 20, 2000

*Portions of this work were supported by the Intel Foundation, by contracts MDA904-96-1-0116 and MDA904-97-6-0176 from Maryland Procurement Office, and by sponsors of CERIAS.

Motivation

Many techniques for information assurance and security involve extracting information from logs. Tools that exploit logs may not deliver their expected benefits, and may even add vulnerabilities of their own, if the information extracted is not accurate, or if unexpected log content can cause them to fail.

So logs must obviously be protected from alteration, a problem already studied by others. But log tamperproofing touches only part of the problem.



Log integrity techniques (studied by others) assure only that R = R'.

Reliable producers and consumers that *demonstrably correspond* (addressed in this work) are also needed for assurance that E = E'.

Approach

Assured correspondence between log producers and consumers is needed to establish *robustness* (nothing the producer can write in the log will cause the consumer to fail on reading) and *semantic accuracy* (what the consumer concludes from a log entry correctly reflects the event and system state observed by the log producer).

- Document log syntax and semantics carefully in a specification sufficiently formal that both producer and consumer can be shown to implement it.
- A natural form is a grammar, arranged and annotated to serve as a reference for semantics as well as syntax.

If the specification allows ambiguous or indistinguishable records to be produced for distinct events, semantic loss is inevitable. A grammar can be machine checked for such problems.

This approach was demonstrated by building a grammar for Solaris BSM logs.

Relation to Content/Semantics

While the community still strives to pin down log content and semantics, is it an extravagance to attend to grammar and syntax?

Claims:

- Syntax overlooked is semantic loss; structure carries meaning.
  Theory: log produced by automaton $\Rightarrow$ all syntax variation reflects state.
  English analogy: French train conductors early to strike / to strike early.
  BSM examples: ioctl, rename.
- Discussions and critiques of log content and semantics require something concrete to discuss; detailed specifications of existing formats provide that focus.
- The process of formalizing a log specification facilitates both automatic and human recognition of weaknesses / ambiguities / omitted content.

The last point was demonstrated as we formalized a BSM specification.

Alternatives

Some information can be extracted without regard to grammatical structure, like skimming a natural language text for key words. Can work when:

- Small fraction of log content is of interest
- That fraction can be characterized in advance and readily distinguished
- Semantic nuances are of no concern

ASAX, IDIOT, and USTAT are examples of ID tools supporting BSM with a skimming approach.

"Skimming"

Advantages:

- Conceptually simple
- Low processing cost *up front*
- Does not require specialized tools like parser generators

Drawbacks:

- Invalid input detected late or not at all
- Semantics carried by syntax lost, recoverable (if at all) only by duplicating some actions of a parser in later processing
- Difficult to identify and check assumptions concerning expected input sequences

Motivation, revisited

For some applications skimming is not suited, such as *deep canonicalization* illustrated by the Common Intrusion Specification Language (CISL) of the Common Intrusion Detection Framework (CIDF).

Shallow vs. Deep Canonicalization
ASAX, IDIOT, and USTAT do not evaluate their detection rules directly against the native log, but convert parts of it first to some canonical form. Their canonical forms may be called *shallow*.

- Simple rearrangements of the native records (discard fields of no interest, align data on word boundaries)
- Semantics not independently specified, require familiarity with native form.
- Can simplify porting ID *engines* between platforms
- but not ID patterns or rules—these deal with native log syntax and semantic issues preserved in the canonical form.

Deep Canonicalization

A deep canonical form, e.g. CISL, has semantics explicitly specified, independent of any native log format. In an ID system based on deep canonicalization, not only evaluation engines but intrusion patterns themselves can be ported between platforms with similar vulnerabilities.

- CISL is rich enough to express the semantic nuances of the native form
- but that means a CISL canonicalizer must correctly and completely discern and translate the original semantics

If tools (such as CIDF E-boxes) do only the familiar skimming of input log formats, the results will be familiar: CISL streams that cannot be properly interpreted without knowledge of the original format (CISL degenerates to a shallow form), or that explicitly and expressively *mis*translate the log. Either way, CISL would not live up to its promise.

Completed work

- Heavily annotated grammar for BSM through Solaris 2.6
- Available in a BSM-parsing package for quick-start BSM-based projects
- Discrepancies and ambiguities in BSM documentation identified in the process are detailed with hyperlinks to original BSM docs for comparison
- Addressed feasibility/efficiency concerns that may have contributed to historical neglect in this area

Efficiency

- A parser for BSM can be efficient, mostly LL(1) with some localized, bounded backtracking.

- Parse early and seldom: distill the semantics into an explicit internal form that can be consulted directly in later processing

- Or else: parsing effort not spent up front is duplicated by all ID rules or other processing affected by the same syntactic feature

Future directions

- Safe skimming
  - When interest is only in a subset of log content known in advance, can parser generator analysis eliminate unnecessary parse-time tests and decisions?
- Variant formats
  - Configurable options in BSM (et al.) slightly alter the grammar of the logs produced
  - Can *environment grammars* (Ruschitzka) be used to avoid proliferation or overcomplication of grammars?