Log Analysis for Data Protection Accountability (FM2014)

Denis Butin and Daniel Le Métayer

Context

- Principle of accountability introduced 30 years ago (OECD), more and more popular — 2009 consortium, soon in European Data Protection Regulation
- Key idea: data controller must not only comply with data protection rules but also *demonstrate* compliance
- Accountability often used vaguely, need to distinguish different levels (policy / procedures / practice)
- Here, focus on accountability of practice

Motivations

- Concretely, need to clarify what are the accounts — logs. Good log design not trivial (cf DUMA13 paper)
- Also need to clearly state obligations — privacy policies
- Entire accountability process must be made explicit and be accurate wrt system execution, also include informal aspects
- Formal approach to clarify and integrate all this
- Other issue: gap between levels of abstraction: personal data values vs system memory addresses, duplicate data, etc

In this work:

- Framework for accountability of practice
- Compliance of system-level logs versus compliance of abstract traces — correctness
- Integration of formal framework with overall process and with manual/informal verifications

Privacy policies

- Assume that all personal data received by data controller has attached policy (sticky policies)
- Consider traces and logs on side of the controller
- Privacy policies defined as
  $Policy = Purposes \times Time \times Time \times Contexts \times FwPolicy$
  $\pi \in Policy \quad \pi = (ap, dd, rd, cx, fw)$
- Purposes, global deletion delay, DS request compliance delay, contexts, data forwarding policy
- Example:
  $\pi = (\{Marketing, Statistics\}, 180d, 60m, \{Location\_Europe\}, \uparrow)$

Abstract events (1/2)

- Describe events at level of personal data, abstracting away from system internals (pointer references, duplicates, etc)
- Expressed intuitively wrt privacy policies
- Events: Data Disclosure, Deletion request, Access request, Deletion, Third party deletion order, data forwarding, use of data for specific purpose, break-glass event, context definition (extensible list, e.g. can add notifications, updates, third party update orders . . .)
- *Trace* = sequence of abstract events

Abstract events (2/2)

- Abstract states: $S_A : \mathit{Entity} \times \mathit{Type} \longrightarrow \mathit{Time} \times \mathit{Entity} \times \mathit{Value} \times \mathit{Policy} \times \mathcal{P}(\mathit{Entity} \times \mathbb{N}) \times \mathcal{P}(\mathit{BGtype} \times \mathit{BGcircumstances} \times \mathit{Time})$
  $(ds, \theta) \mapsto (t, or, v, \pi, \mathit{receivers}, bg)$
- origin or = entity from which most recent version of data comes from; $\mathit{receivers}$ = set of third parties that received the data, together with event index
- bg = set of triples containing information about break-glass events
- State expanded for current context
- Semantics defined for abstract events, using abstract state
- Trace compliance properties stated

Log events (1/2)

- Log events describe actual behaviour of system — small number of general purpose low-level operations such as receiving data, sending it, reading, copying, deleting . . .
- Semantics passed through parameters
- "Personal-data-free logs": No data value in parameters of log events, but references. Data subjects identites and data categories are still recorded, but not actual data values
- Logs are sequences of log events

Log events (2/2)

- Concrete state defined:
  $S_C : Reference \longrightarrow Time \times Type \times Entity \times \mathbb{N} \times Entity \times Policy \times \mathcal{P}(Entity \times \mathbb{N}) \times \mathcal{P}(BGtype \times BGcircumstances \times Time)$
  $ref \mapsto (t, \theta, ds, j, or, \pi, receivers, bg)$
- Log compliance properties stated
- Example: Deletions yield third party deletion requests, sent between the last forwarding of the data and its deletion:
  $L_i = (Delete, t', ref) \ \wedge \ State_C(L, i-1)(ref) = (t, \theta, ds, or, \pi, receivers, bg) \implies \forall \ (t_p, l) \in receivers, \exists \ k \mid \exists \ t'' \mid L_k = (Send, DeleteOrder, t'', t_p, ds, \theta) \wedge k \in]\alpha, i[$ with $\alpha = max\{n \mid (t_p, n) \in receivers\}$

Accountability properties

- Relation between abstract states and concrete states
- Relation between traces and logs
- Can then express correctness property relating traces and logs: $Compliant_C(L) \ \wedge \ Abstract_L(L, \sigma) \Longrightarrow Compliant_A(\sigma)$
- No race condition: deletion requests are fulfilled after a finite delay
- If we add update events, same property for update requests

Accountability process

- Manual checks by independent auditors complement automatic verifications
- General system architecture verification: check that logs reflect actual system execution. Done manually because building formal model of entire system is huge task. Formal framework gives guidelines on log event format.
- Specific verifications related to outcome of log analysis — break-glass event circumstances, reasons for use of data for specific purpose, etc
- Audit for accountability cannot provide absolute compliance guarantee, goal is to make cheating more difficult
- In practice, data protection authority controllers or auditors do not check all logs but explore logs for specific types of data

Conclusion

- Accountability is incentive for data controllers to take obligations more seriously
- Important to reconcile different meanings of the principle and to integrate formal with informal approach
- Privacy policy and events format in this work: typical but not set in stone
- Other important topic: log integrity/confidentiality
- Possible to get meaningful compliance checking without storing values of personal data in logs
- Need to consider data aggregation/merging