Accountability by Design for Privacy

Denis Butin, Marcos Chicote and Daniel Le Métayer

Introduction

- ICT growth adds to concern about sensitive data use
- Individuals share more & more PII
- Stronger privacy guarantees needed
- Regulations exist through EU directives, country-specific laws
- Not enough — practical, specific means required

Background

- *Legal* privacy protection — EU directives 95/46 (Data Protection), 2002/58 (Privacy & Electronic Communications)
- Privacy & PII security are subtle, context-dependent
- Need bridge between broadly-defined concepts & actual ICT systems

Privacy Impact Assessment

- Modern analytic approach to mitigate privacy risks in ICT systems
- Done before system deployment
- No guarantees to users about actual running system

Motivation (1/2)

- Runtime / a posteriori verifications needed!
- Provide “proven trust” instead of “blind trust”
- Data controllers should be accountable to data subjects
- Practical requirements?

Motivation (2/2)

- Need to provide the means to check that agreements were fulfilled
- Approach: check PII handling event histories (logs) against agreements with an automatic tool!
- Duality — if PIA done right (*implies* design choices), accountability possible (*depends* on design)

What is Accountability?

- Obligation to accept responsibility for actions
- Attributability: who did what?
- Non-repudiable evidence that cannot be falsified
- Transparent use of information

Enabling Accountability (1/2)

- Accountability does not emerge spontaneously
- Feasibility of comprehensive a posteriori verification?
- Depends directly on technical architecture!

Example — requirements on logs for accountability

Timestamps needed in logs if notification to data subject within an hour required when sharing their age with a third party

Enabling Accountability (2/2)

Need to define:

- Obligations to be met $\Longrightarrow$ Policy language
- Compliance checking evidence $\Longrightarrow$ Log architecture
- Compliance checking procedure $\Longrightarrow$ Log analyzer

Usage Policy Languages

- Usage policy languages allow data handling details to be set
- On both sides: data subject (preferences), data controller (policies)

Example – data handling preference

Data controller may use data subject's email address to send security alerts, but may not share it with third parties.

Primelife Policy Language (PPL)

- Automated matching of data subject & data controller policies yields *Sticky Policies* (agreements)
- Wide range of obligations possible (trigger + action)
- Only informal specification available until our work

Example — informal obligation

If PII accessed by data controller for purpose marketing, anonymize it within a day

PII Event Logging

- Data Controller must provide evidence that agreements met
- Audit possible through inspection of event histories (logs) wrt data handling agreements
- Structure of logs conditions auditability, hence accountability
- Deciding what to include in logs — not a trivial task

Formalising PPL

- Relevant events precisely defined (syntax)
- Compliance properties described (semantics)
- Tool built for automated compliance checking (implementation)
- Reasoning over compliance can be generalised

Guidelines for Log Design

- Importance of explicitness — sufficiently detailed event information needed
- Avoid ambiguity; reflect causal relationships
- Accountability definitions shape log structure & vice versa
- Include contextual information if obligation of performance

Future Work

- Implications of audit process role definition (third party, data subject, certificated authority. . .)
- Accountability-oriented, standardised log format (policy language-independent)
- Detailed case studies illustrating design guidelines

Questions?