Policy Auditing over Incomplete Logs: The reduce algorithm

Anupam Datta
Carnegie Mellon University

Fall 2015
Example from HIPAA Privacy Rule

A covered entity may disclose an individual’s protected health information (phi) to law-enforcement officials for the purpose of identifying an individual if the individual made a statement admitting participating in a violent crime that the covered entity believes may have caused serious physical harm to the victim.

- **Concepts in privacy policies**
  - **Actions:** send(p1, p2, m)
  - **Roles:** inrole(p2, law-enforcement)
  - **Data attributes:** attr_in(prescription, phi)
  - **Temporal constraints:** in-the-past(state(q, m))

- **Purposes:** purp_in(u, id-criminal))
- **Beliefs:** believes-crime-caused-serious-harm(p, q, m)
Detecting Privacy Violations

The Oracle

<table>
<thead>
<tr>
<th>Species</th>
<th>Computer Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>A program designed to investigate the human psyche.</td>
</tr>
</tbody>
</table>

The Matrix character

Computer-readable privacy policy

Audit

Automated audit for black-and-white policy concepts

Detect policy violations

Oracles to audit for grey policy concepts

"The Oracle" is a program designed to investigate the human psyche.
Auditing Black-and-White Policy Concepts

With D. Garg (CMU → MPI-SWS) and L. Jia (CMU)

2011 ACM Conference on Computer and Communications Security
Key Challenge for Auditing

Audit Logs are Incomplete

Future: store only past and current events
Example: Timely data breach notification refers to future event

Subjective: no “grey” information
Example: May not record evidence for purposes and beliefs

Spatial: remote logs may be inaccessible
Example: Logs distributed across different departments of a hospital
Abstract Model of Incomplete Logs

Model all incomplete logs uniformly as 3-valued structures

\[ \mathcal{L}(P) \in \{tt, ff, uu\} \]

Define **semantics** (meanings of formulas) over 3-valued structures
reduce: The Iterative Algorithm

\[ \text{reduce} (\mathcal{L}, \varphi) = \varphi' \]
Syntax of Policy Logic

- First-order logic with restricted quantification over infinite domains (challenge for reduce)
- Can express timed temporal properties, “grey” predicates
Example from HIPAA Privacy Rule

A covered entity may disclose an individual’s protected health information (phi) to law-enforcement officials for the purpose of identifying an individual if the individual made a statement admitting participating in a violent crime that the covered entity believes may have caused serious physical harm to the victim.

\[\forall p_1, p_2, m, u, q, t.\]
\[\left(\text{send}(p_1, p_2, m) \land\right.\]
\[\text{tagged}(m, q, t, u) \land\]
\[\text{attr_in}(t, \text{phi})\]
\[\Rightarrow \text{inrole}(p_1, \text{covered-entity}) \land \text{inrole}(p_2, \text{law-enforcement})\]
\[\left(\text{purp_in}(u, \text{id-criminal})\right) \land\]
\[\exists m' \dashv \text{state}(q, m') \land \text{is-admission-of-crime}(m')\]
\[\land \text{believes-crime-caused-serious-harm}(p_1, q, m')\]
reduce: Formal Definition

General Theorem: If initial policy passes a syntactic **mode check**, then finite substitutions can be computed

\( \text{reduce}(L, \forall x. \varphi) \)

Applications: The entire HIPAA and GLBA Privacy Rules pass this check
Example

\[
\varphi = \\
\forall p_1, p_2, m, u, q, t.
\]

- \(\text{send}(p_1, p_2, m) \land \text{tagged}(m, q, t, u) \land \text{attr} \_in(t, \phi)\)

- \(\text{inrole}(p_1, \text{covered-entity}) \land \text{inrole}(p_2, \text{law-enforcement})\)
  \(\land \text{purp} \_\text{in}(u, \text{id-criminal})\)
  \(\land \exists m'. (\leftarrow \text{state}(q, m')\)
    \(\land \text{is-admission-of-crime}(m')\)
    \(\land \text{believes} \_\text{crime-caused-serious-harm}(p_1, m'))\)

Example 1

\(
\{ p_1 \rightarrow \text{UPMC}, \\
p_2 \rightarrow \text{allegeny-police}, \\
m \rightarrow \text{M2}, \\
q \rightarrow \text{Bob}, \\
u \rightarrow \text{id-bank-robber}, \\
t \rightarrow \text{date-of-treatment} \}
\)

\(
\{ m' \rightarrow \text{M1} \}
\)

Log

Aug 15, 2014
\text{state}(\text{Bob, M1})

Sept 17, 2014
\text{send}(\text{UPMC, allegeny-police, M2})
\text{tagged}(\text{M2, Bob, date-of-treatment, id-bank-robber})

\[
\varphi' = T
\]

- \(\text{purp} \_\text{in}(\text{id-bank-robber, id-criminal})\)
- \(\text{is-admission-of-crime}(\text{M1})\)
- \(\text{believes} \_\text{crime-caused-serious-harm}(\text{UPMC, M1})\)
Correctness of Reduce

**Theorem 3.2** (Partial correctness of reduce). If \( \text{reduce} (\mathcal{L}, \varphi) = \psi \) and \( \mathcal{L} \leq \mathcal{L}' \), then (1) \( \mathcal{L}' \models \varphi \) iff \( \mathcal{L}' \models \psi \) and (2) \( \mathcal{L}' \models \overline{\varphi} \) iff \( \mathcal{L}' \models \overline{\psi} \).
Implementation and Case Study

- Implementation and evaluation over simulated audit logs for compliance with all 84 disclosure-related clauses of HIPAA Privacy Rule

- Performance:
  - Average time for checking compliance of each disclosure of protected health information is 0.12s for a 15MB log

- Mechanical enforcement:
  - reduce can automatically check 80% of all the atomic predicates
Ongoing Transition Efforts

- Integration of reduce algorithm into Illinois Health Information Exchange prototype
  - Joint work with UIUC and Illinois HLN

- Auditing logs for policy compliance
  - Ongoing conversations with Symantec Research
Applications of Reduce

- Audit to detect violations of policy or demonstrate compliance
- Provide explanations for violations (e.g., which clause of HIPAA was violated)
- Help train employees about privacy laws (e.g., check whether a certain type of disclosure is permitted by HIPAA)
Learning Outcomes for You

- Translate privacy laws into first-order logic for use by reduce

- Use reduce tool to check logs for compliance with laws

- Use reduce to check whether certain types of disclosures are permitted by a privacy law

Homework 1 will make you work through these problems
Possible project around other privacy laws such as FERPA, COPPA
Privacy Specification Languages

- P3P [Cranor et al.], XACML [OASIS], EPAL [Backes et al.]:
  Less expressive (no temporal ops,..)
- Logic of Privacy and Utility [Barth et al.]:
  Related specification logic;
  enforcement only for propositional fragment
Logical Specification of Privacy Laws

Smaller fragments of laws

- *Logic of Privacy and Utility* [Barth et al.]: Example clauses from HIPAA and GLBA
- PrivacyAPIs [Gunter et al.]: HIPAA 164.506
- Datalog HIPAA [Lam et al.]: HIPAA 164.502, 164.506, 164.510
Related Work

Runtime monitoring in MFOTL [Basin et al ’10]

- Pre-emptive enforcement
- Efficient implementation
- Assumes past-completeness of logs
- Less expressive mode checking (“safe-range check”)
- Cannot express HIPAA or GLBA
Related Work

Industry practice

Fairwarning Audit Tool
- Customized SQL queries over access logs
- Queries not tied to policy clauses
Detecting Policy Violations

The Oracle

The Matrix character

<table>
<thead>
<tr>
<th>Species</th>
<th>Computer Program</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>A program designed to investigate the human psyche.</td>
</tr>
</tbody>
</table>

Automated audit for black-and-white policy concepts

Detect policy violations

Oracles to audit for grey policy concepts

Computer-readable privacy policy

Audit
Thanks!
More Technical Details
Definition of $\widehat{\text{sat}}$

Assume: The function $\text{sat}(L, P)$ computes all substitutions $\sigma$ for variables in $P$ such that $L \models P\sigma$, if certain argument positions in $P$ are ground.

\[
\begin{align*}
\widehat{\text{sat}}(L, p_O(t_1, \ldots, t_n)) &= \text{sat}(L, p_O(t_1, \ldots, t_n)) \\
\widehat{\text{sat}}(L, \top) &= \{\bullet\} \\
\widehat{\text{sat}}(L, \bot) &= \{\} \\
\widehat{\text{sat}}(L, c_1 \land c_2) &= \bigcup_{\sigma \in \widehat{\text{sat}}(L, c_1)} \sigma + \widehat{\text{sat}}(L, c_2\sigma) \\
\widehat{\text{sat}}(L, c_1 \lor c_2) &= \widehat{\text{sat}}(L, c_1) \cup \widehat{\text{sat}}(L, c_2) \\
\widehat{\text{sat}}(L, \exists x. c) &= \widehat{\text{sat}}(L, c) \backslash \{x\} \quad (x \text{ fresh})
\end{align*}
\]
Mode Analysis: Idea

- Example 1: \( \text{addless}(x, y, a) = x + y < a \)

- Key idea: If input positions are grounded, then only finite number of satisfying substitutions for output positions.

- Example 1 moding: \( \text{addless}(+, -, +) \)

- Example 2: \( \theta = \text{send}(p1, p2, m) \land \text{tagged}(m, q, t, u) \)

- \( \text{send}(-,-,-) \): all positions are output mode
- \( \text{tagged}(+,-,-,-) \): message position is input mode
Mode Analysis: Predicates

1. $\{\} \vdash \text{send}(p_1, p_2, m): \{p_1, p_2, m\}$
2. $\{p_1, p_2, m\} \vdash \text{tagged}(m, q, t, u): \{p_1, p_2, m, q, t, u\}$

$$\forall k \in I(p_0). \text{fv}(t_k) \subseteq \chi_I \quad \chi_O = \chi_I \cup \left( \bigcup_{j \in O(p_0)} \text{fv}(t_j) \right)$$

$$\chi_I \vdash p_0(t_1, \ldots, t_n): \chi_O$$
Mode Analysis: Conjunction

1. \( \{\} \vdash \text{send}(p_1, p_2, m): \{p_1, p_2, m\} \)

2. \( \{p_1, p_2, m\} \vdash \text{tagged}(m, q, t, u): \{p_1, p_2, m, q, t, u\} \)

3. \( \{\} \vdash \text{send}(p_1, p_2, m) \land \text{tagged}(m, q, t, u): \{p_1, p_2, m, q, t, u\} \)

\[
\begin{align*}
\chi_I \vdash c_1: \chi & \quad \chi \vdash c_2: \chi_O \\
\hline
\chi_I \vdash c_1 \land c_2: \chi_O
\end{align*}
\]
Mode Analysis and \( \text{sat} \)

Example: \( \theta = \text{send}(p_1, p_2, m) \land \text{tagged}(m, q, t, u) \)

- \( \text{send}(-,-,-) \): all positions are output mode
- \( \text{tagged}(+,-,-,-) \): message position is input mode
- \( \text{sat}(\theta) = \text{sat}(\text{send}(p_1, p_2, m)) + \text{sat}(\text{tagged}(m, q, t, u) \circ \sigma) \)

\[
\{ \begin{array}{l}
p_1 \rightarrow \text{UPMC}, \\
p_2 \rightarrow \text{allegeny-police}, \\
m \rightarrow M2, \\
q \rightarrow \text{Bob}, \\
u \rightarrow \text{id-bank-robber}, \\
t \rightarrow \text{date-of-treatment} \\
\end{array} \}
\]
Mode Analysis: Termination of

General Theorem: If initial policy passes a syntactic **mode check**, then finite substitutions can be computed.

Applications: The entire HIPAA and GLBA Privacy Rules pass this check.