Bootstrapping Privacy Compliance in Big Data Systems

Giulia Fanti
Based on slides by Anupam Datta
CMU

Fall 2019
Administrative

- HW2 will be released this week
  - Stay tuned

- How did Docker recitation go?
Quiz on Canvas

- Take the quiz on your laptops/tablets/devices
- Please do not look back at your notes
  - these quizzes do not affect your grade, so you should just try to do your best without gaming the grade
- 10 minutes
Today’s Lecture

Bootstrap Privacy Compliance in Big Data Systems

S. Sen, S. Guha, A. Datta, S. Rajamani, J. Tsai, J. M. Wing
Proceedings of 35th IEEE Symposium on Security and Privacy
May 2014.
Privacy Compliance for Bing

Setting:

- Auditor has access to source code
The Privacy Compliance Challenge

Legal Team
Crafts Policy

Privacy Champion
Interprets Policy

Developer
Writes Code

Audit Team
Verifies Compliance

English Privacy Policy
Compliant?

Millions of Lines of Undocumented Code
A Streamlined Audit Workflow

Legal Team
Crafts Policy
Interprets Policy

Code analysis
Development annotations

Grok
Data inventory with policy labels

Audit Team
Verifies Compliance

Legalease
A formal policy specification language

Encode
Refine

Checker

Annotated Code
Update Grok
Potential violations

Fix code
A Streamlined Audit Workflow

**Workflow for privacy compliance**

**Legalease**, usable yet formal policy specification language

**Grok**, bootstrapped data inventory for big data systems

**Scalable** implementation for Bing
Privacy as Restrictions on Personal Information Flow

Restrictions

Direct

Purpose & Role based

EPAL
XACML
*-access control

Temporal

FOTLs
[Formal Contextual Integrity,
Reduce audit algorithm,
Basin et al.]

Interference

Purpose → Planning

Jif,
FlowCaml,…

[Hayati & Abadi]

Probabilistic

Information Flow

Experiments

Grok +
Legalease
A Streamlined Audit Workflow

Legal Team
Crafts Policy

Privacy Champion
Interprets Policy

Legalease
A formal policy specification language

Grok
Data inventory with policy datatypes

Code analysis, developer annotations

Checker

Annotated Code
Legalease Policy
Potential violations

Update Grok

Developer
 Writes Code

Audit Team
Verifies Compliance

Fix code
Specification: Legalease

- Usable.
- Expressive.
- Precise.

Usable by lawyers and privacy champs.
Expressive enough for real-world policies.
Precise semantics for local reasoning.
Legalease: Components

- Each statement is a **clause** reasoning about **attributes**
- **Clauses**
  - Allow
  - Deny
  - Except
- **Attributes**
  - InStore
  - UseForPurpose
  - AccessByRole
  - DataType
### Legalease: Syntax

#### Policy Clause $C$ ::= $D \mid A$

#### Deny Clause $D$ ::= \(\text{DENY } T_1 \cdots T_n \ \text{EXCEPT } A_1 \cdots A_m\) \mid \text{DENY } T_1 \cdots T_n$

#### Allow Clause $A$ ::= \(\text{ALLOW } T_1 \cdots T_n \ \text{EXCEPT } D_1 \cdots D_m\) \mid \text{ALLOW } T_1 \cdots T_n$

#### Attribute $T$ ::= ⟨attribute-name⟩ $v_1 \cdots v_l$

#### Value $v$ ::= ⟨attribute-value⟩
DENY Datatype IPAddress
UseForPurpose Advertising

We will not use full IP Address for Advertising.
Designed for Usability

Exceptions
How legal texts are structured
One-to-one correspondence

Local Reasoning
Each exception refines its immediate parent
Formally proven property

H. DeYoung, D. Garg, L. Jia, D. Kaynar, and A. Datta, “Experiences in the logical specification of the HIPAA and GLBA privacy laws”
We will not use full IP Address for Advertising. IP Address may be used for detecting abuse. In such cases, it will not be combined with account information.
Primer on Lattices

- On document camera
- Posets
- Lattices
- Rules for ALLOW and DENY
  - Examples
A Lattice of Policy Labels

- If “IPAddress” use is allowed then so is everything below it
- If “IPAddress:Truncated” use is denied then so is everything above it
Policy Types: Concept Lattices

*InStore* Lattice  *UseForPurpose* Lattice  *AccessByRole* Lattice
Policy Labels: Datatypes

Going down within a lattice:
- Finer specification

TypeState specifies limited temporal properties
Formal Semantics

\[
\begin{align*}
T^G \sqsubseteq T^C &\quad \exists_i D_i \text{ denies } T^G \\
\text{ALLOW } T^C \text{ EXCEPT } D_1 \cdots D_m &\text{ denies } T^G \tag{A_2}
\end{align*}
\]

Based on Lattice Orderings on Policy Types
Formal Semantics

Recursively check exceptions
ALLOW clauses have DENY clauses as exceptions
Top Level clause determines Blacklist/Whitelist