Datacentric Semantics for Verification of Privacy Policy Compliance by Mobile Applications

The context

- Many mobile applications ask for access permission to personal information stored in the device.
- Once these permissions are granted it is often the case that data concerning gender, sex, age, gps location, smartphone ID, contacts, etc. are managed in a way that partially releases them to third parties (e.g. for advertisement profiling, analytics, social computing, etc.) with some degree of obfuscation.
- This leaves the user unaware of the actual leakage of confidential information, and privacy issues may be raised.
- Information-flow analyses in the literature are too conservative to be useful in this scenario.

Overview

The goal: A theoretical framework supporting the design of tools that provide developers and end-users better control of how the values managed by the applications reveal (confidential) data stored in the device

The key-idea: Keep track of the dependence flow of confidential information from data sources in the values managed by the program, by explicitely considering also the obfuscation impact of the program operators.

The theory: Abstract Interpretation [P.Cousot] & Taint Analysis [Pistoia-Tripp]

Plan of the talk

1. Motivating examples
2. Enhanced concrete and abstract semantics for datacentric analysis.
3. Confidentiality and obfuscation values for sources and operators
4. Verification of privacy compliance policies
5. Conclusions

Example 1: Inmobi

```
1 public class IMBanner {
2   public void loadBanner() {
3     UserInfo user = new UserInfo();
4     user.updateInfo();
5     BannerView banner = new BannerView(user);
6     banner.loadNewAd();
7     show(banner);
8   }
9 }
10
11 public class BannerView {
12   private UserInfo user;
13   BannerView(UserInfo user) {
14     this.user = user;
15   }
16   void loadNewAd() {
17       String url = "http :// www.inmobi.com/...?id="
18           + user.id + "&lang="+user.language+
19           "&country=" + user.country + "&loc=" + user.loc;
20     // open an http connection with url
21     // update the new ad to display
22   }
23 }
```

```
25 public class UserInfo {
26   String language;
27   String country;
28   String id;
29   Location loc;
30
31   void updateInfo() {
32     Locale localLocale = Locale.getDefault();
33     language = localLocale.getLanguage();
34     country = localLocale.getCountry();
35     String androidId = Settings.Secure.getAndroidId();
36     id = MessageDigest.hashSHA1(androidId);
37     loc = LocationManager.getLastKnownLocation();
38   }
39 }
```

This code is extracted from the `Inmobi` library, one of the three most popular advertisement engines for Android apps.

Example 2: IMSI

The following code snippet is extracted from internal Android library `com.android.internal.telephony.cdma.RuimRecords`:

```
String mImsi = telephonyManager.getDeviceId();
log("IMSI:" + mImsi.substring (0, 6) + "xxxxxxxxx");
```

- The IMSI code is usually made by 15 characters, where the first 3 characters identify the country, the following 2 or 3 characters identify the mobile network, and the rest is used to identify the device.
- Therefore, we assume that the first 6 characters do not contain confidential information.

Plan of the talk

1. Motivating examples
2. **Enhanced concrete and abstract semantics for datacentric analysis.**
3. Confidentiality and obfuscation values for sources and operators
4. Verification of privacy compliance policies
5. Conclusions

Syntax

Arithmetic, Textual, and Boolean Expressions

$lexp$	$\in$	$\mathbb{E}_{Lab}$	Label Expressions
$lexp$	::=	$\ell \mid \star \mid v$	
$nexp$	$\in$	$\mathbb{E}_N$	Arithmetic Expressions
$nexp$	::=	$n \mid v \mid -\,nexp \mid nexp_1 \oplus nexp_2 \mid read(lexp)$	
		where $\oplus \in \{+, -, *, /, f(-,-)\}$, with $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$	
$sexp$	$\in$	$\mathbb{E}_S$	String Expressions
$sexp$	::=	$s \mid v \mid sexp_1 \circ sexp_2 \mid$	
		$encrypt(sexp, k) \mid sub(sexp, k_1, k_2) \mid$	
		$hash(sexp) \mid read(lexp)$	
$bexp$	$\in$	$\mathbb{E}_B$	Boolean Expressions
$bexp$	::=	true $\mid$ false $\mid v \mid$	
		$nexp_1 \otimes nexp_2 \mid sexp_1 \otimes sexp_2 \mid lexp_1 \otimes lexp_2 \mid$	
		$\neg b \mid b_1 \oslash b_2$	
		where $\otimes \in \{\leq, \geq, ==, >, \neq, \ldots\}$, and $\oslash \in \{\vee, \wedge\}$	

Collecting Semantics: Atomic Data Expressions

- We define atomic data expressions by $\mathbb{D} = \{\langle \ell_i, L_i \rangle : i \in I\}$.
- Given a set of data labels, which identify the locations of the *read-only* datastore a program interacts with, an atomic data expression $adexp$ is a set of elements $\langle \ell_i, \{(op_j, \ell'_j) : j \in J\}\rangle$.
- An element $\langle \ell_i, \{(op_j, \ell'_j) : j \in J\}\rangle$ in $adexp$ says that the value of that expression has been obtained from the datum stored in the location ℓ_i by combining it with data coming from the locations ℓ'_j through the corresponding operations op_j.

  In other words, an atomic data expression keeps track, for each source of the expression value, of the set of other data sources that were used to get that value from it.

```
1  public class IMBanner {
2   public void loadBanner() {
3    UserInfo user = new UserInfo();
4    user.updateInfo();
5    BannerView banner = new BannerView(user);
6    banner.loadNewAd();
7    show(banner);
8   }
9  }
10
11 public class BannerView {
12  private UserInfo user;
13  BannerView(UserInfo user) {
14   this.user = user;
15  }
16  void loadNewAd() {
17    String url = "http :// www.inmobi.com/...?id="
18      + user.id + "&lang="+user.language+
19      "&country=" + user.country + "&loc=" + user.loc;
20   // open an http connection with url
21   // update the new ad to display
22  }
23 }
```

```
25 public class UserInfo {
26  String language;
27  String country;
28  String id;
29  Location loc;
30
31  void updateInfo() {
32   Locale localLocale = Locale.getDefault();
33   language = localLocale.getLanguage();
34   country = localLocale.getCountry();
35   String androidId = Settings.Secure.getAndroidId();
36   id = MessageDigest.hashSHA1(androidId);
37   loc = LocationManager.getLastKnownLocation();
38  }
39 }
```

After the execution of `updateInfo` (line 4) we have that:

- $\texttt{user.language} \mapsto \{\langle \texttt{Language}, \varnothing \rangle\}$,
- $\texttt{user.country} \mapsto \{\langle \texttt{Country}, \varnothing \rangle\}$,
- $\texttt{user.id} \mapsto \{\langle \texttt{AndroidId}, \{(\texttt{hash}, \texttt{AndroidId})\} \rangle\}$,
- $\texttt{user.loc} \mapsto \{\langle \texttt{Location}_1, \varnothing \rangle\}$.

```
1 public class IMBanner {
2   public void loadBanner() {
3     UserInfo user = new UserInfo();
4     user.updateInfo();
5     BannerView banner = new BannerView(user);
6     banner.loadNewAd();
7     show(banner);
8   }
9 }
10
11 public class BannerView {
12   private UserInfo user;
13   BannerView(UserInfo user) {
14     this.user = user;
15   }
16   void loadNewAd() {
17       String url = "http :// www.inmobi.com/...?id="
18           + user.id + "&lang="+user.language+
19           "&country=" + user.country + "&loc=" + user.loc;
20     // open an http connection with url
21     // update the new ad to display
22   }
23 }
```

```
25 public class UserInfo {
26   String language;
27   String country;
28   String id;
29   Location loc;
30
31   void updateInfo() {
32     Locale localLocale = Locale.getDefault();
33     language = localLocale.getLanguage();
34     country = localLocale.getCountry();
35     String androidId = Settings.Secure.getAndroidId();
36     id = MessageDigest.hashSHA1(androidId);
37     loc = LocationManager.getLastKnownLocation();
38   }
39 }
```

We concatenate all this data in a string stored in `url` at line 17.

Therefore, when building `url` we obtain the following atomic data expression with label `AndroidId`:

$$\langle \texttt{AndroidId}, \{(\texttt{hash}, \texttt{AndroidId}), (\circ, \texttt{Language}), (\circ, \texttt{Country}), (\circ, \texttt{Location}_1)\}\rangle$$

while for $\texttt{Location}_1$ we obtain $\langle \texttt{Location}_1, \{(\circ, \texttt{AndroidId}), (\circ, \texttt{Language}), (\circ, \texttt{Country})\}\rangle$.

Collecting Semantics: Concrete Domain Σ

- We focus our collecting semantics on the variables referring to values coming from the datastore.
- A data environment maps local variables in $\texttt{Var}$ to atomic data expressions ($D : \texttt{Var} \longrightarrow \wp(\mathbb{D})$).

  Note that each variable may contain data about different sources (e.g., the concatenation of the strings representing the Android identifier and the location), and therefore each variable is related to a set of atomic data expressions.
- In addition, the concrete state tracks value information as well ($V : \texttt{Var} \longrightarrow (\mathbb{Z} \cup \mathbb{S})$).
- Formally, $\Sigma = D \times V$.

Concrete Datastore

A concrete datastore is a set $\{\langle \ell_i, \varnothing \rangle\} : i \in I\} \subseteq \mathbb{D}$ such that $\forall i, j \in I : i \neq j \Rightarrow \ell_i \neq \ell_j$, and $\ell_i \neq \star\}$.

Example:

	Surname	FamName	SSN	Rank	Sal	Perf
1	John	Taylor	346-19-1754	1	120,000	+4
2	Paul	Stenton	316-49-6574	2	90,000	0
3	Pieter	Prince	464-19-8177	2	88,000	-2
4	Donna	Bean	643-12-8555	1	130,000	-4
5	Joanna	Foster	323-22-9941	1	110,000	-1
6	Carl	Bowl	336-88-7754	2	70,000	+5

The atomic data expression corresponding to *Bowl* is $\{\langle (6, \texttt{FamName}), \varnothing \rangle\}$.

Concrete Semantics

$$
\begin{array}{rcl}
S_A[\![x]\!](a,v) & = & a(x) \\
S_A[\![read(lexp)]\!](a,v) & = & \{\langle S_L[\![lexp]\!](a,v), \varnothing\rangle\} \\
S_A[\![encrypt(sexp,k)]\!](a,v) & = & \{\langle \ell_1, L_1 \cup \{([encrypt,k],\ell_1)\}\rangle : \langle \ell_1, L_1\rangle \in S_A[\![sexp]\!](a,s,n)\} \\
S_A[\![s]\!](a,v) & = & \{\langle \star, \varnothing\rangle\} \\
S_A[\![sexp_1 \circ sexp_2]\!](a,v) & = & \{\langle \ell_1, L_1 \cup \{(\circ,\ell_2)\}\rangle, \langle \ell_2, L_2 \cup \{(\circ,\ell_1)\}\rangle : \\
 & & \langle \ell_1, L_1\rangle \in S_A[\![sexp_1]\!](a,v)\ , \langle \ell_2, L_2\rangle \in S_A[\![sexp_2]\!](a,v)\} \\
S_A[\![sub(sexp,k_1,k_2)]\!](a,v) & = & \{\langle \ell_1, L_1 \cup \{([sub,k_1,k_2],\ell_1)\}\rangle : \langle \ell_1, L_1\rangle \in S_A[\![sexp]\!](a,v)\} \\
S_A[\![hash(sexp)]\!](a,v) & = & \{\langle \ell_1, L_1 \cup (hash,\ell_1)\rangle : \langle \ell_1, L_1\rangle \in S_A[\![sexp]\!](a,v)\}
\end{array}
$$

- A standard concrete evaluation of numerical ($S_N : nexp \times V \to \mathbb{Z}$) and string ($S_S : sexp \times V \to \mathbb{S}$) expressions is provided, as well as the evaluation of Boolean conditions ($S_B : bexp \times V \to \{\texttt{true}, \texttt{false}\}$), and label expressions ($S_L : lexp \times \Sigma \to \texttt{Lab}$).
- The special label $\star$ is used to represent data coming either from the input of the program or from the constant set of the program itself.

Concrete semantics - Statements

$$
\begin{array}{rcl}
S[\![x := sexp]\!](a,v) & = & (a[x \mapsto S_A[\![sexp]\!](a,v)], v[x \mapsto S_S[\![sexp]\!](v)]) \\
S[\![send(sexp)]\!](a,v) & = & (a,v) \\
S[\![c_1; c_2]\!](a,v) & = & S[\![c_2]\!](S[\![c_1]\!](a,v))) \\
S[\![\text{if } bexp \text{ then } c_1 \text{ else } c_2]\!](a,v) & = & \begin{cases} S[\![c_1]\!](a,v) & \text{if } S_B[\![bexp]\!](v) \\ S[\![c_2]\!](a,v) & \text{otherwise} \end{cases} \\
S[\![\text{while } bexp \text{ do } c]\!](a,v) & = & S[\![\text{ if } (bexp) \text{ (c; while } bexp \text{ do } c)]\!](a,v)
\end{array}
$$

Abstract Semantics

- **Values** can be abstracted by means of well known either relational or non-relational domains for numerical and textual values.
- **Labels** can be abstracted by any abstract domain for categorical data, like a flat constant propagation domain.
- As required by the Abstract Interpretation Framework, the computation on abstract values is performed by abstract operators op'_i that are required to be correct with respect to their concrete counterpart op_i.

Atomic Data Abstraction

Given a set of atomic data, an abstract element will be a set of tuples $\{\langle \ell_j^a, L_j^{a\sqcap}, L_j^{a\sqcup} \rangle : j \in J\}$, where

- ℓ_j^a is an element of an abstract domain that abstracts labels in $\mathtt{Lab}$
- $L_j^{a\sqcap} = \{(op_{ij}^a, \ell_{ij}^a) : i \in I\}$ is an under-approximation of the set of operators applied to the sources represented by ℓ_j^a with values coming from sources represented by ℓ_{ij}^a
- $L_j^{a\sqcup} = \{(op_{ij}^a, \ell_{ij}^a) : i \in I'\}$ is an over-approximation of the set of operators applied to the sources represented by ℓ_j^a with values coming from sources represented by ℓ_{ij}^a
- $L_j^{a\sqcap} \subseteq L_j^{a\sqcup}$.

Ordering

The order on the abstract elements is given by the order on the Cartesian product of the components' domain, and the least upper bound and greatest lower bound operators are defined accordingly.

Given two abstract atomic data $d_1 = \{\langle \ell^a_{1i}, L^{a\sqcap}_{1i}, L^{a\sqcup}_{1i}\rangle : i \in I_1\}$ and $d_2 = \{\langle \ell^a_{2i}, L^{a\sqcap}_{2i}, L^{a\sqcup}_{2i}\rangle : i \in I_2\}$ on the same abstract domains for values and labels,

$$d_1 \sqsubseteq d_2 \Leftrightarrow \forall i \in I_1\ \exists j \in I_2 : \ell^a_{1i} = \ell^a_{2j},\ L^{a\sqcap}_{1i} \supseteq L^{a\sqcap}_{2j},\ L^{a\sqcup}_{1i} \subseteq L^{a\sqcup}_{2j}$$

Abstract Semantics

$$
\begin{array}{rcl}
S^a[\![x := sexp]\!](a^a, v^a) & = & (a^a[x \mapsto S^a_A[\![sexp]\!](a^a, v^a)], v^a[x \mapsto S^a_S[\![sexp]\!](v^a)]) \\
S^a[\![send(sexp)]\!](a^a, v^a) & = & (a^a, v^a) \\
S^a[\![c_1; c_2]\!](a^a, v^a) & = & S^a[\![c_2]\!](S^a[\![c_1]\!](a^a, v^a)) \\
S^a[\![\text{if } bexp \text{ then } c_1 \text{ else } c_2]\!](a^a, v^a) & = & S^a[\![c_1]\!](a^a, S^a_B[\![bexp]\!](v^a)) \sqcup \\
 & & S^a[\![c_2]\!](a^a, S^a_B[\![\neg bexp]\!](v^a)) \\
S^a[\![\text{while } bexp \text{ do } c]\!](a^a, v^a) & = & fix(S^a[\![\text{ if } (bexp) \text{ (while } bexp \text{ do } c)]\!](a^a, v^a))
\end{array}
$$

Theorem

The abstract semantics of a terminating program P with an abstract datastore A is a conservative (sound) over-approximation of the enhanced concrete semantics of P with a concrete datastore in $\gamma(\mathtt{A})$.

Plan of the talk

1. Motivating examples
2. Enhanced concrete and abstract semantics for datacentric analysis
3. **Confidentiality and obfuscation values for sources and operators**
4. Verification of privacy compliance policies
5. Conclusions

Confidentiality and Obfuscation

So far, we made no distinctions among data contained in the data-store, with respect to their confidentiality level.

- We can consider a lattice of confidentiality levels $\mathbb{S}$, and we can associate to each label ℓ in $\mathtt{Lab}$ an element $\eta(\ell) \in \mathbb{S}$.
- On the operation side, we introduce the notion of *obfuscation degree* $\zeta(op)$.
- This can be seen as a generalization of the all/nothing tainting approach, where only declassification operators (like encrypting functions) are tracked.

Confidentiality of Atomic Data - Monotonic Ops.

Let S be a lattice representing confidentiality levels of labels.

Let O be a lattice representing the obfuscation power of operators.

Let η and ζ be functions assigning confidentiality/obfuscation values in S and O to labels and operators, respectively.

The confidentiality value of $\{\langle \ell_i, L_i \rangle : i \in I\}$ with respect to (η, ζ), when the combination of operators in $\bigcup_{i \in I} L_i$ is monotonic with respect to the obfuscation order in the lattice O, is:

$$(sc_{min}, sc_{max}, lc_{min}, lc_{max})$$

$$sc_{min} = \sqcap_{\mathsf{D}}\{\eta(\ell_i) : i \in I\} \qquad lc_{min} = \sqcap_{\mathsf{O}}\{\zeta(op_{ij}) : (op_{ij}, \ell_j) \in L_i, i \in I\}$$

$$sc_{max} = \sqcup_{\mathsf{D}}\{\eta(\ell_i) : i \in I\} \qquad lc_{max} = \sqcup_{\mathsf{O}}\{\zeta(op_{ij}) : (op_{ij}, \ell_j) \in L_i, i \in I\}$$

Inmobi and IMSI Examples - confidentiality value

Imagine that we have $\mathtt{L} < \mathtt{M} < \mathtt{H}$ as both the confidentiality and obfuscation lattice.

$$
\begin{array}{lrcl}
\zeta: & [encrypt, k] & \mapsto & \mathtt{H} \\
 & hash & \mapsto & \mathtt{M} \\
 & \circ & \mapsto & \mathtt{L} \\
 & [sub, k_1, k_2] & \mapsto & \begin{cases} \mathtt{H} & \text{if } k_2 < 7 \\ \mathtt{M} & \text{otherwise} \end{cases} \\
\eta: & \texttt{Language} & \mapsto & \mathtt{L} \\
 & \texttt{Country} & \mapsto & \mathtt{L} \\
 & \texttt{Location}_i & \mapsto & \mathtt{M} \\
 & \texttt{AndroidId} & \mapsto & \mathtt{H} \\
 & \texttt{IMSI} & \mapsto & \mathtt{H}
\end{array}
$$

The confidentiality value of the expression assigned in `Inmobi` to `url` is $(\mathtt{L}, \mathtt{H}, \mathtt{L}, \mathtt{M})$.

For $\langle \texttt{IMSI}, \{([sub, 0, 6], \texttt{IMSI})\} \rangle$, we get $(\mathtt{H}, \mathtt{H}, \mathtt{H}, \mathtt{H})$. This says that even if sensitive data items are leaked, a powerful obfuscation is definitely applied.

Confidentiality of Atomic Data - General Case

Let S be a lattice representing confidentiality levels of labels.

Let O be a lattice representing the obfuscation power of operators.

Let η be a function assigning confidentiality values in S to labels,

Let ζ be a function assigning to each set of operators an **interval** in $\mathsf{O} \times \mathsf{O}$ representing its min and max obfuscation power.

The confidentiality value of $\{\langle \ell_i, L_i \rangle : i \in I\}$ wrt (η, ζ) is:

$$(sc_{min}, sc_{max}, lc_{min}, lc_{max})$$

$$\begin{aligned}
sc_{min} &= \sqcap_{\mathsf{D}}\{\eta(\ell_i) : i \in I\} \\
sc_{max} &= \sqcup_{\mathsf{D}}\{\eta(\ell_i) : i \in I\} \\
lc_{min} &= \sqcap_{\mathsf{O}}\{\pi_1(\zeta(\{op_{ij} : (op_{ij}, \ell_j) \in L_i\})) : i \in I\} \\
lc_{max} &= \sqcup_{\mathsf{O}}\{\pi_2(\zeta(\{op_{ij} : (op_{ij}, \ell_j) \in L_i\})) : i \in I\}
\end{aligned}$$

where π_1 and π_2 denote the min and max element of the interval, respectively.

Confident. Value of Abstract Atomic Data

Let $\mathbb{S}$ and $\mathbb{O}$ be the lattices representing the labels' confidentiality and the obfuscation power of operators, respectively.

Let η and ζ be functions assigning confidentiality/obfuscation values in $\mathbb{S}$ and $\mathbb{O}$ to (concrete) labels and operators, respectively.

Let $\eta^a(\ell^a) = [\sqcap\{\eta(\ell) : \ell \in \gamma(\ell^a)\}, \sqcup\{\eta(\ell) : \ell \in \gamma(\ell^a)\}]$.

Let ζ^a be the function assigning to each abstract operator the same obfuscation value assigned by ζ to the concrete operator it corrresponds.

If the combination of operators in $\mathbb{O}\mathfrak{p}$ appearing in $\bigcup_{i\in I} L_i^{a\sqcup}$ is monotonic with respect to the obfuscation order in $\mathbb{O}$, then the confidentiality value of $\{\langle \ell_i^a, L_i^{a\sqcap}, L_i^{a\sqcup}\rangle : i \in I\}$ is $(sc^a_{min}, sc^a_{max}, lc^a_{min}, lc^a_{max})$:

$$sc^a_{min} = \sqcap_{\mathbb{S}}\{\pi_1(\eta^a(\ell_i^a)) : i \in I\} \qquad lc^a_{min} = \sqcap_{\mathbb{O}}\{\zeta^a(op^a_{ij}) : (op^a_{ij}, \ell^a_j) \in L_i^{a\sqcap}, i \in I\}$$

$$sc^a_{max} = \sqcup_{\mathbb{S}}\{\pi_2(\eta^a(\ell_i^a)) : i \in I\} \qquad lc^a_{max} = \sqcup_{\mathbb{O}}\{\zeta^a(op^a_{ij}) : (op^a_{ij}, \ell^a_j) \in L_i^{a\sqcup}, i \in I\}$$

Plan of the talk

1. Motivating examples
2. Enhanced concrete and abstract semantics for datacentric analysis.
3. Confidentiality and obfuscation values for sources and operators
4. **Verification of privacy compliance policies**
5. Conclusions

Confidentiality Policy

Given the set of data source labels $\mathtt{Lab}$, and the confidentiality/obfuscation lattices $\mathtt{S}$ and $\mathtt{O}$ for labels and operations, respectively, a **confidentiality policy** is a tuple $\pi = (\eta, \zeta, \kappa_{sc\_max}, \kappa_{lc\_min})$ such that

- η, ζ assign each label and each operator a corresponding value in $\mathtt{S}$ and $\mathtt{O}$.
- κ_{sc_max} is a source confidentiality threshold (the max confidentiality level allowed for sources).
- κ_{lc_min} is an obfuscation threshold (the min obfuscation level required for operators).

Given a program P, let X be the set of concrete/abstract atomic data P generated as an output. We say that P satisfies the confidentiality policy $\pi = (\eta, \zeta, \kappa_{sc\_max}, \kappa_{lc\_min})$ if:

$\forall d \in X$, if $(sc_{min}, sc_{max}, lc_{min}, lc_{max})$ is the confidentiality value of d with respect to (η, ζ), then, $sc_{max} \sqsubseteq_{\mathtt{S}} \kappa_{sc\_max}$ and $lc_{min} \sqsupseteq_{\mathtt{O}} \kappa_{lc\_min}$.

Confidentiality Policy Verification

Theorem

Consider a program P, an abstract datastore A, and a confidentiality policy $\pi = (\eta, \zeta, \kappa_{sc\_max}, \kappa_{lc\_min})$. If the program P terminates, and the output of the analysis on P and A satisfies the confidentiality policy π, then any actual execution of program P on a concrete datastore in $\gamma(A)$ satisfies the confidentiality policy π.

Inmobi Example

In the Inmobi example, when building `url` we obtained the following atomic data expressions:

$$\{\langle \texttt{AndroidId}, \{(\texttt{hash}, \texttt{AndroidId}), (\circ, \texttt{Language}), (\circ, \texttt{Country}), (\circ, \texttt{Location}_1)\}\rangle,$$
$$\langle \texttt{Location}_\texttt{i}, \{(\circ, \texttt{AndroidId}), (\circ, \texttt{Language}), (\circ, \texttt{Country})\}\rangle,$$
$$\langle \texttt{Country}, \{(\circ, \texttt{AndroidId}), (\circ, \texttt{Language}), (\circ, \texttt{Location}_\texttt{i})\}\rangle,$$
$$\langle \texttt{Language}, \{(\circ, \texttt{AndroidId}), (\circ, \texttt{Location}_\texttt{i}), (\circ, \texttt{Country})\}\rangle\}$$

A reasonable privacy policy: *a datum can be released only if its obfuscation level is equal or higher than its confidentiality level*.

This program satisfies this model for `Country` and `Language` (whose confidentiality level is `L` and they are released without any obfuscation), but not for $\texttt{Location}_\texttt{i}$ (with confidentiality level `M` and released without any obfuscation) and `AndroidId` (whose confidentiality level is `H` and it is released after invoking $hash$, that is, with obfuscation level `M`).

Sources' Confidentiality Policies

As an orthogonal approach, we may define a confidentiality policy as a propositional formula that captures constraints on the allowed releasing levels of (confidential) data in the datastore.

Let $\texttt{Lab}$ denote as usual the set of data source labels, and $\texttt{Op}$ denote the set of operators in the program. Consider a set of propositional variables $\texttt{V}$, and a function λ that maps elements of V into either labels or links.

$$\lambda : \texttt{V} \longrightarrow \texttt{Lab} \cup \{(op, \ell) : op \in \texttt{Op}, \ell \in \texttt{Lab}\}$$

A *policy formula* is a positive propositional formula on $\texttt{V}$, i.e. a propositional formula using only $\wedge, \vee$ and $\leftrightarrow$ logical operators

Example

We can express the fact that we can leak the Android ID if encrypted and the location, or the hashed Android ID, or the first six characters of the `IMSI`, by means of the formula:

$$\varphi = (x \wedge y) \vee z \vee w$$

where

$$\begin{aligned} \lambda(x) &= ([encrypt, k], \texttt{AndroidId}) \\ \lambda(y) &= \texttt{Location} \\ \lambda(z) &= (hash, \texttt{AndroidId}) \\ \lambda(w) &= ([sub, 0, 6], \texttt{AndroidId}) \end{aligned}$$

Policy Formulas' Satisfiability

Given a set of atomic data S, a set of propositional variables $\mathtt{V}$ and an assignment λ on $\mathtt{V}$, we say that S satisfies the policy formula φ on $\mathtt{V}$ if $S, \lambda \vDash \varphi$, as defined inductively as follows:

$$
\begin{array}{lcl}
S, \lambda \vDash v \in \mathtt{V} & \Leftrightarrow & \begin{cases} \lambda(v) \in \bigcup_{d \in S} \mathtt{src}(d) & \text{if } \lambda(v) \in \mathtt{Lab} \\ \lambda(v) \in \bigcup_{d \in S} \mathtt{links}(d) & \text{otherwise.} \end{cases} \\
S, \lambda \vDash \varphi_1 \vee \varphi_2 & \Leftrightarrow & S, \lambda \vDash \varphi_1 \text{ or } S, \lambda \vDash \varphi_2 \\
S, \lambda \vDash \varphi_1 \wedge \varphi_2 & \Leftrightarrow & S, \lambda \vDash \varphi_1 \text{ and } S, \lambda \vDash \varphi_2 \\
S, \lambda \vDash \varphi_1 \leftrightarrow \varphi_2 & \Leftrightarrow & S, \lambda \vDash \varphi_1 \text{ iff } S, \lambda \vDash \varphi_2.
\end{array}
$$

Observe that, by construction, if the data resource denoted by $\lambda(x)$ contributes to any of the values represented by the atomic data S, then $S, \lambda \vDash x$.

Lifting to the abstract atomic data

Let S^a be a set of abstract atomic data, and
$\lambda^a : \mathtt{V} \longrightarrow \mathtt{Lab^a} \cup \{(op^a, \ell^a) : op \in \mathtt{Op}, \ell^a \in \mathtt{Lab^a}\}$ be a function mapping propositional variables into abstract labels and links.
We apply the three value assignment $assign(S^a, \lambda) : \mathtt{V} \to \{\mathtt{true}, \mathtt{false}, \top\}$:

$$assign(S^a, \lambda^a)(v) = \begin{cases} \mathtt{false} & \text{if } \lambda^a(v) \in \mathtt{Lab^a} \text{ and } \lambda^a(v) \notin \bigcup_{d \in S^a} \mathtt{src}(d) \\ & \text{or} \\ & \text{if } \lambda^a(v) \notin \mathtt{Lab^a} \text{ and } \lambda^a(v) \notin \bigcup_{d \in S^a} \mathtt{L}^{\sqcup}(d) \\ \mathtt{true} & \text{if } \lambda^a(v) \in \mathtt{Lab^a} \text{ and} \\ & \quad (\lambda^a(v) \in \bigcup_{d \in S^a} \mathtt{src}(d) \text{ or } \lambda^a(v) \in \bigcup_{d \in S^a} \mathtt{links\_lab}(\mathtt{L}^{\sqcap}_d)) \\ & \text{or} \\ & \text{if } \lambda^a(v) \notin \mathtt{Lab^a} \text{ and } \lambda^a(v) \in \bigcup_{d \in S^a} \mathtt{L}^{\sqcap}(d) \\ \top & \text{otherwise} \end{cases}$$

Soundness

Theorem

Let φ be a positive formula on a set $\mathtt{V}$ of propositional variables, S^a be a set of abstract atomic data, and $\lambda^a : \mathtt{V} \longrightarrow \mathtt{Lab^a} \cup \{(op^a, \ell^a) : op \in \mathtt{Op}, \ell^a \in \mathtt{Lab^a}\}$ be a function mapping propositional variables into abstract labels and links. If $assign(S, \lambda)(\varphi) = \mathtt{true}$, then there is a set of atomic data $S \subseteq \bigcup_{d \in S^a} \gamma(d)$ and a function λ satisfying $\forall v \in \mathtt{V} : \lambda(v) \in \gamma_{\mathtt{Lab}}(\lambda^a(v))$, such that $S, \lambda \vDash \varphi$.

Conclusions

- We designed a framework that enhances tainting analysis
- Main novelty: finer granularity
- It allows to link access control and information flow
- It allows to support user-defined privacy compliance policy verification
- Waiting for experimental results... (coming soon!) and the treatment of indirect flow (work in progress)

Thanks!