Distributed Data Usage Control
Alexander Pretschner
TU München
FOSAD Summer School, September 2014
Jew E. Lovat, F. Kelbert, P. Kumari, M. Büchler, P. Rimatifll, C. Bier,
T. Wüchner, D. Hölting, M. Lüerscher, J. Peschla, P. Wenz

Problem

Don’t copy my data
Delete my data
if I say so

Step 1: Object Clarifications
Copy data
Delete data

Step 2: Action Clarifications
Copy data
Delete data

Domain Meta Model

WBSN Domain Model
System Layers

Step 3: Local Enforcement

Step 4: Tracking Representations

Tracking Representations
Rights and Duties

Distributed Usage Control

Teaser Demo: Data-Driven Cross-Layer Usage Control

Video 1: http://www22.in.tum.de/fileadmin/demos/uc/uc_new_mp4.mp4

Agenda

• Part I: Introduction
• Part II: Event-Based Usage Control
• Part III: Data-Centric Usage Control
• Part IV: Quantitative Usage Control

• Part V: Local Single-Layer Enforcement
• Part VI: Distributed Enforcement
• Part VII: Cross-Layer Enforcement
• Part VIII: Policy Derivation
• Part IX: Discussion

Event-Based Usage Control

• II.1 Requirements
• II.2 Events, refinements, traces
• II.3 Specification-level policies
• II.4 Implementation-level policies
• II.5 Video examples: context-aware UC on Android; UC for camera surveillance
• II.6 Analysis problems
Requirements

- Interviews
- Regulations
- Literature
- Taxonomy of enforcement mechanisms
- System model

- Restrictions and necessary actions
  - Permission and duty
  - Conditions
    - Time, cardinality, events, purpose, environment
  - Examples
    - No distribution
    - Deletion after 30 days
    - At most one copy
    - Notification upon access
    - For statistical purposes only
  - Firewalls certified w.r.t. Common Criteria

Event-Based Usage Control

- II.1 Requirements
- II.2 Events, refinements, traces
- II.3 Specification-level policies
- II.4 Implementation-level policies
- II.5 Video examples
- II.6 Analysis problems

Policies

- Specification-level policies specify the what
  - Expressed as first order future time temporal logic formulas
- Implementation-level policies specify the how
  - Expressed as Event-Condition-Action rules:
    - Conditions first order past time temporal logic formulas
    - Actions describe inhibition, modification, execution

Background

- Events
- Variable events
- Refinement
- System events
- Traces

Policies: first order part

\[ \Gamma = \forall \exists \left[ \text{N \ String} \mid \Gamma \mid \ldots \mid \psi \right] \mid \text{eval}(\Gamma) \mid \text{forall VName in VVal} : \psi \]

Examples: quantification, variables

\[ \forall \text{\upsilon}_1 \in \text{SGNAME} : \forall \text{\upsilon}_2 \in \text{TIME} : \\
E\text{play} \rightarrow \{ \text{obj} \rightarrow \text{\upsilon}_1, \text{time} \rightarrow \text{\upsilon}_2 \} \rightarrow \\
E\text{login} \rightarrow \{ \text{obj} \rightarrow \text{\upsilon}_1, \text{time} \rightarrow \text{\upsilon}_2 \} \]

\[ \forall \text{\upsilon}_1 \in \{0, \ldots, 100\}, \forall \text{\upsilon}_2 \in \{\text{EU, US, JP, AU}\} : \\
E\text{play} \rightarrow \{ \text{obj} \rightarrow \text{sgn} \text{mp3}, \text{quality} \rightarrow \text{\upsilon}_1, \text{location} \rightarrow \text{\upsilon}_2 \} \rightarrow \\
\{ \text{eval}(<\gamma>) \rightarrow \text{eval}(\text{\upsilon}_1 \leq 50) \} \]

Macros for permissions omitted here
Ordering event parameters:  \( S_{\text{ordering}} \)  
- Strengthening only upon re-distribution
- How to strengthen „pay 10€“ or „play with 50% quality“?
- Specify rights as left-open intervals and duties as right-open intervals
  - „Play with at most 75% quality“  
    \( \forall \epsilon \in \{T_p, \ldots, T_o\} \colon \epsilon(\text{play} \Rightarrow \{\text{quality} \Rightarrow \epsilon\}) \Rightarrow \epsilon(\text{play} \Rightarrow \{\text{quality} \Rightarrow \epsilon\}) \)  
  - Strengthened to „play with at most 50% quality“  
  - „Pay at least €10“  
    \( \forall \epsilon \in \{T_p, \ldots, T_o\} \colon \epsilon(\text{pay} \Rightarrow \{\text{amount} \Rightarrow \epsilon\}) \Rightarrow \epsilon(\text{pay} \Rightarrow \{\text{amount} \Rightarrow \epsilon\}) \)  
  - Strengthened to „pay at least €15“

Ordering event names:  \( S_{\text{naming}} \)  
- Expressed as disjunction

Semantics of events  
- Substitutions: for a variable event \( e \)
  - \( v \in \text{Var} \), \( v_n \in \text{dom}(v) \), \( x \in \text{Var}(v_n) \)
  - \( e'[v_n \mapsto x] \) is the result of simultaneously replacing all occurrences of \( v_n \) by \( x \) in \( e \)
- \( \text{VarsIn}(e) \) is the set of variables in a variable event \( e \)
  - \( \text{Inst}_e : \mathcal{V}^O \rightarrow \mathcal{P}(\mathcal{V}) \) generates all ground substitutions of an event
  - \( \text{VarsIn}(e) = \{ v_1, \ldots, v_{N(e)} \} \)  
  - \( \text{Semantics of events} \)
  - \( \forall e' \in \text{maxRefEv}; e \in \mathcal{V} ; \exists v \in \mathcal{E} \)  
  - \( (e', \text{actual}) \models E(e) \Leftrightarrow e' \models \text{refineEv} e' \wedge (e', \text{intended}) \models E(e) \Leftrightarrow e' \models \text{refineEv} e' \wedge (e', \text{intended}) \models E(e) \)

Event-Based Usage Control
- II.1 Requirements
- II.2 Events, refinements, traces
- II.3 Specification-level policies
- II.4 Implementation-level policies
- II.5 Video examples
- II.6 Analysis problems

Specification-level policies:
- Plus the usual macros
- Will use usual symbols instead
Three Examples

Quality of playing sng.mp3 must not exceed 50% before paying €10.

Non-anonymized data must not leave system without notifying admin.

Accounts are suspended after three consecutive failed login attempts. They can be reactivated within a week upon a signed request, sent over email by their owner. After one week without receiving a reactivation email, suspended accounts will be closed.

Remarks

- No liveness
  - But can be expressed
- Decidability
  - Depends on eval
- Complexity
  - Terrible – but intended to be used for runtime monitoring
- Expressivity
  - Use templates instead

Event-Based Usage Control

- II.1 Requirements
- II.2 Events, refinements, traces
- II.3 Specification-level policies
- II.4 Implementation-level policies
- II.5 Video examples
- II.6 Analysis problems

Implementation-Level Policies

- ILPs are Event-Condition-Action rules
- Conditions
  - \( \Phi^+ ::= (\Phi^-) \mid \Phi^- \mid \Phi^- \implies \Phi^- \mid \forall x \in X. \Phi^- \mid \Phi^- \exists \Phi^- \mid \Phi^- \frac{\text{before} \ N}{}\]
- Actions
  - Inhibition, modification, execution

Semantics of Conditions

- ECA rules
  - describes trigger event and condition:
    \( t(\sigma, \nu) \rightarrow (I(\rho(\sigma)) \land E(\rho(\nu)) \rightarrow \nu) \)
- Effect
  - \( m_{\text{src}}(\sigma, \nu) \rightarrow \forall \nu \in I(\sigma). \nu \rightarrow E(\nu) \)
  - \( m_{\text{dest}}(\sigma, \nu, \text{Mod}) \rightarrow \forall \nu \in I(\sigma). \nu \rightarrow (E(\nu) \land m_{\text{dest}}(\sigma, \nu, \text{Mod})) \)
  - \( m^*_{\text{src}}(\sigma, \nu) \rightarrow \forall \nu \in I(\sigma). \nu \rightarrow E(\nu) \)
- Composition of ILPs
  - \( M := \bigwedge_{i \in I} m^-_{\text{src}}(\sigma, \nu^i) \land m^-_{\text{dest}}(\sigma, \nu^i, \text{Mod}) \land m^*_{\text{src}}(\sigma, \nu^i) \land E(\nu^i) \)
Allow default

\[ M_{\text{default}} \leftrightarrow \bigwedge_{e \in \text{Events}} I(e) \rightarrow \left( E(e) \lor \bigvee_{e' \in \text{Events} \text{ s.t. } e' \equiv e} \exists \text{VarRef}(e') : e \text{ refines} E(e') \land e' \equiv e \right) \]

- Semantics of a set of ILPs: \( \square(M_{\text{complete}}) \) with

\[ M_{\text{complete}} \leftrightarrow M \land M_{\text{default}} \]

Example 1

Quality of playing sng.mp3 must not exceed 50% before paying €10

Example 2

Non-anonymized data must not leave system without notifying admin

Example 3

Accounts are suspended after three consecutive failed login attempts. They can be reactivated within a week upon a signed request, sent over email by their owner. After one week without receiving a reactivation email, suspended accounts will be closed.

Event-Based Usage Control

- II.1 Requirements
- II.2 Events, refinements, traces
- II.3 Specification-level policies
- II.4 Implementation-level policies
- II.5 Video examples event-based UC:
  - Video 2: Context-aware UC on Android
    http://www2.in.tum.de/fileadmin/demos/uc/UC4NEST-Demo3.wmv
  - Video 3: UC for camera surveillance
    http://www2.in.tum.de/fileadmin/demos/uc/Demo4-UC4NEST-Demo-v3.mp4
- II.6 Analysis problems

Example: UC4NEST scenario (I)

- Video surveillance on main floor of Fraunhofer IOSB building
  - People can authenticate as members of some groups using mobile devices and MC-MXT marker
Example: UC4NEST scenario (II)

- Video surveillance on main floor of Fraunhofer IOSB building
  • Intrusion detection service (IDS) observes the immediate proximity of a valuable painting of an ongoing art exhibition

Example: UC4NEST policies (III)

- (Group specific) privacy policies:
  • Show and track unknown persons
  • Show security personnel, but do not track
  • Neither track nor show staff members
- Alarm policy:
  • People causing an IDS alarm are tracked and visualized on the map
  • The video stream of the corresponding camera is temporarily shown to the operator
  • After the alarm has been handled, group specific policies are reactivated

Event-Based Usage Control

- II.1 Requirements
- II.2 Events, refinements, traces
- II.3 Specification-level policies
- II.4 Implementation-level policies
- II.5 Video examples event-based UC:
  - context-aware UC on Android
  - UC for camera surveillance
- II.6 Analysis problems

Analysis Problems

- Satisfiability
- Satisfiability of composed ILPs
- Strengthening policies for distribution
- Entailment of specification-level policies by ILPs
- Configuration of ILPs with free variables

Analysis Problems via

| Use $\Phi = \Phi^+ \cup \Phi^-$ implies $\Phi$ and $\models_f$ with
| $\forall \varphi \in \Phi^+ : \exists e \in \text{Trace}, f \in N : \varphi \in \Phi \Rightarrow ((t,n) \models \varphi \Rightarrow (t,n) \models \varphi)$
| $\land \exists \psi \in \Phi^- : \exists e \in \text{Trace}, f \in N : \psi \models (t,n) \models \varphi$ for consistency:
| $\forall \varphi \in \Phi^+ : \models_f \varphi \Rightarrow \exists e \in \text{Trace}, f \in N : \models_f \varphi$
| Plus a chaotic model $M$, a transformation $\tau_{OSL}$, an encoding of the analysis problems in $\varphi$, a model checker and $M \models \tau_{OSL}(\varphi) \Rightarrow \models_f \varphi$

Model Checking

| Sanity constraint $C = \bigcap_{e \in S} E(e) \rightarrow I(e)$
| Entailment $M \models \tau_{OSL}(C \sqcap M_{complete}) \rightarrow \varphi$
| Checking capabilities: $M \models \tau_{OSL}(C \sqcap M_{complete}[X \rightarrow x_1] \rightarrow \varphi)$
| Configure ILPs with free variables by enumeration and $M \models \tau_{OSL}(C \sqcap M_{complete}[X \rightarrow x_1] \rightarrow \varphi)$
Examples I: Strengthening

\[ v_1 \land E[download \rightarrow \text{class} \rightarrow \text{song}] \rightarrow E[play \rightarrow \text{obj} \rightarrow \text{add}] \text{ within } 3 \]
\[ v_2 \land \text{repunit}(3,E[display \rightarrow \text{obj} \rightarrow \text{max}]), \forall \tau \in \{5, \ldots, \tau'\} : E[\text{pay} \rightarrow \text{receiver} \rightarrow \text{Bank}, \text{amount} \rightarrow \tau] \]
\[ v_1 \land E[download \rightarrow \text{class} \rightarrow \text{song}] \rightarrow E[\text{play} \rightarrow \text{obj} \rightarrow \text{add}] \text{ within } 1 \]
\[ v_2 \land \text{repunit}(2,E[\text{display} \rightarrow \text{obj} \rightarrow \text{max}]), \forall \tau \in \{5, \ldots, \tau'\} : E[\text{pay} \rightarrow \text{receiver} \rightarrow \text{Bank}, \text{amount} \rightarrow \tau] \]

Examples II: Capabilities and Configuration

Specification-level policy

\[ v_2 = \neg E[\text{play} \rightarrow \text{object} \rightarrow \text{cd}) \text{ until } E[\text{pay} \rightarrow \{\text{amount} \rightarrow 3\}] \]

enforced by ILP

\[ M^0 = \text{maximize} \{ \text{play} \rightarrow \{\text{object} \rightarrow \text{cd}\}, \forall \tau \in \{30, \ldots, \tau'\} : \exists \tau (\neg \text{E[\text{pay} \rightarrow \{\text{amount} \rightarrow \tau}\} \text{ before } 1) \}

and a configurable ILP with free variable X

\[ M^X = \text{maximize} \{ \text{play} \rightarrow \{\text{object} \rightarrow \text{cd}\}, \forall \tau \in \{30, \ldots, \tau'\} : \exists \tau (\neg \text{E[\text{pay} \rightarrow \{\text{amount} \rightarrow X}\} \text{ before } 1) \}

Translation

- Straightforward except for eval and cardinality:

\[ \forall \mu, \nu, \phi, \psi, \chi, \xi, i, m, n, \tau \in 0 \]
\[ \forall \mu, \nu, \phi, \psi, \chi, \xi, i, m, n, \tau \in 0 \]
\[ \forall \mu, \nu, \phi, \psi, \chi, \xi, i, m, n, \tau \in 0 \]
\[ \forall \mu, \nu, \phi, \psi, \chi, \xi, i, m, n, \tau \in 0 \]
\[ \forall \mu, \nu, \phi, \psi, \chi, \xi, i, m, n, \tau \in 0 \]
\[ \forall \mu, \nu, \phi, \psi, \chi, \xi, i, m, n, \tau \in 0 \]
\[ \forall \mu, \nu, \phi, \psi, \chi, \xi, i, m, n, \tau \in 0 \]

Discussion

- Said before: Complexity prohibitive in principle; implementation with NuSMV not that bad
- eval function possibly undecidable
- Simple (manual) abstractions help

Agenda

- Part I: Introduction
- Part II: Event-Based Usage Control
- Part III: Data-Centric Usage Control slides by Enrico Lovat
- Part IV: Quantitative Usage Control
- Part V: Local Single-Layer Enforcement
- Part VI: Distributed Enforcement
- Part VII: Cross-Layer Enforcement
- Part VIII: Policy Derivation
- Part IX: Discussion

Data-Centric Usage Control

- III.1 Motivation and intuition
- III.2 Formalization
- III.3 Demo video
Usage Control: Data not Representations!
Event Processing, Runtime Verification, ...
“Don’t COPY” or “Do DELETE”

Usage Control with Data Flow detection

Example

Information Flow Tracking System

Container 1

Data 1

Container 2

Data 2

Container 3

Data 3
Data-Centric Usage Control

- III.1 Motivation and intuition
- III.2 Formalization
- III.3 Demo video

Four ingredients

- Data state
- Transition relation capturing data flows
- Adjusted refinement and semantics of events
- New operators

Data State

$$\Sigma = (C \rightarrow 2^D) \times (C \rightarrow 2^C) \times (F \rightarrow C)$$

D : Set of Data
C : Set of data-Containers
F : Set of Names for containers
A : Set of Actions
\Sigma : Set of Information Flow States
Four ingredients

- Data state
- Transition relation capturing data flows
- Adjusted refinement and semantics of events
- New operators

Transition Relation

\[ R \subseteq \Sigma \times P(S) \rightarrow \Sigma \]

Ordering of events does not matter:

\( \forall \sigma \in \Sigma : R(\sigma, \emptyset) = \emptyset \)
\( \forall \sigma \in \Sigma : E_S \subseteq S : \sigma \in S : e \in E_S \implies R(\sigma, E_s) = R(\sigma, (e \in E_s \setminus \{e\})) \)

uniquely defines \( R \)

Transition Relation determines Data State

\[ \text{states} : (\text{Trace} \times \mathbb{N}) \rightarrow \Sigma \]
\[ \text{states}(t, 0) = \sigma_t \]
\[ n > 0 \implies \text{states}(t, n) = R(\text{states}(t, n - 1), t(n - 1)) \]

Adjusted refinement

- Data state
- Transition relation capturing data flows
- Adjusted refinement and semantics of events
- New operators
Adjusted refinement, formally

\[
\text{refines}_E \subseteq (E \times \Sigma) \times E
\]

\[
\forall e_1, e_2 \in E \forall \sigma \in \Sigma : (e_1, \sigma) \text{refines}_E e_2 \iff
\begin{cases}
(e_1, \sigma) = \text{getclass}(e_2) \land \text{refines}_E e_1 \lor \\
(e_1, \sigma) = \text{dataUsage} \land \text{getclass}(e_2) = \text{containerUsage} \land \\
\exists d \in D \exists c \in C : d = \sigma(e) \land \\
\text{obj} \rightarrow d \in e_1.p \land \text{obj} \rightarrow c \in e_1.p \land e_1.p \setminus (\text{obj} \rightarrow d) \subseteq e_2.p \setminus (\text{obj} \rightarrow c)
\end{cases}
\]

Semantics of events

\[
\models_{\text{E}} (S \times \Sigma) \times \Phi
\]

\[
\forall e' \in \text{maxRefine} \forall e \in E \forall \sigma \in \Sigma \exists e'' \in E : \\
((e', \sigma), \sigma) \models_{\text{E}} e'' \iff \langle e'', \sigma \rangle \text{refines}_E e'' \land e'' \in \text{Inst}_d(e)
\]

Four ingredients

- Data state
- Transition relation capturing data flows
- Adjusted refinement and semantics of events
- New operators

State-Based Operators

\[\Phi_0 := \text{isNotIn}(D, \varnothing) \mid \text{isCombinedWith}(D, D)\]

\[\Phi_0 \equiv \Phi | \Phi_0\]

(plus is\text{OnlyIn}(d, C) \equiv \text{isNotIn}(d, C \setminus C_0)\)

Semantics

\[\models_{\text{E}} (\text{Trace} \times \Sigma) \times \Phi_0\]

\[\forall t \in \text{Trace}, \forall n \in N, \forall \sigma \in \Sigma : (t,n) \models_0 \sigma \iff \sigma = \text{state}(t,n) \land \\
\exists d_1, d_2 \in D, C \subseteq C_0 : \sigma = \text{isNotIn}(d_1, C) \land \\
\exists d' \in \text{Inst}(d, C) : \\
(\sigma \subseteq \text{Inst}(d', C_0) \land d' \subseteq d_{\text{state}}(t,n)) \iff d' \subseteq d_{\text{state}}(t,n)
\]

(plus semantics for \(\Phi_0\) and lifting to past, \(\Phi_0_t\))

Example Transition Relation: OS

\[
\forall s : C \rightarrow \{D\}, \forall v : C \rightarrow \{D\}, \forall f : P \rightarrow C, \forall p \in P, \forall f \in \text{fBlock} : (s,l,f) \subset \text{fBlock} : (s,l,f) \subset \text{fBlock} : (s,l,f)
\]

... and so on
Data-Centric Usage Control

- III.1 Motivation and intuition
- III.2 Formalization
- III.3 Demo video
  - Demo 4: Data-centric UC on Android via TaintDroid
    [http://www22.in.tum.de/fileadmin/demos/uc/uc4android/Demo2.wmv](http://www22.in.tum.de/fileadmin/demos/uc/uc4android/Demo2.wmv)

So far …

- Generalization of access control to the future; permissions and duties
- Specification-level policies (SLP): what
- Implementation-level policies (ILP): how
  - Inhibition, modification, execution
  - Event-based usage control
    - Semantic model: Traces of intended and actual events
    - First order future time temporal logic for SLPs
    - Event-condition-action rules with condition in first order past time temporal logic for ILPs

So far …

- Data-centric usage control
  - Data vs. representations/containers
  - Track data flow in-between representations
  - Policies extended to data usages
    - Data usage = any usage of a container that (potentially) contains the data
    - Operationally, if a system event involves a specific representation, we check which data items this representation potentially contains and if there are any applicable data usage policies

Agenda

- Part I: Introduction
- Part II: Event-Based Usage Control
- Part III: Data-Centric Usage Control
- Part IV: Quantitative Usage Control slides by Enrico Lovat
  - Part V: Local Single-Layer Enforcement
  - Part VI: Distributed Enforcement
  - Part VII: Cross-Layer Enforcement
  - Part VIII: Policy Derivation
  - Part IX: Discussion

Label creep

Quantitative measurements

A. Pretschner: Usage Control. Bertinoro 2014 103

A. Pretschner: Usage Control. Bertinoro 2014 104

A. Pretschner: Usage Control. Bertinoro 2014 105

A. Pretschner: Usage Control. Bertinoro 2014 106
Possibilistic DFT

- Yes, data is (possibly) stored in this representation
- No, it is not

Quantitative DFT

- Yes, data is (possibly) stored in this representation
- No, it is not

(At most) \( x \) different units of data are stored in this representation

Quantitative Data Flow Tracking

- **Goal**: Model *how much* data is stored where
  - Where = in which representation
- **Declassification criteria**
  - Little data = No data?
  - Avoid merging of declassified content
- **Quantitative policies**
  - Enforceable preventively or detectively
  - No more than 10KB of data can leave the system
- **Acceptable exception**
  - A-posteriori policies (e.g. for auditing)
  - Security in practice

Terms:

- **UNIT of data**: smallest part of a representation that can be addressed by an action (e.g. OS Blocks/Bytes, DB Records, Window manager → Pixels)
- **SIZE**: number of units.

Assumptions:

- Each action of the system has a size.
- We can observe how many units of representation are transferred by an action, but not which ones

Idea

- Assumes a size for each action in the system
- Size of action → **Upper bound**
- Amount of different data in source → **Upper bound**
Data Provenance

Result depends on the ‘history’ of data transfer

Provenance graph

Provenance Graph

- $G = (N, E)$
- $N = (C \times N)$
- $E = (N \times N \times N)$

An edge represents an action that adds data to or removes data from a container

Provenance Graph

1: Init $(A, 20)$

2: Transfer $(A, B, 5)$
Provenance Graph

- $G = (N,E)$
- $N = (C \times \mathbb{N})$
- $E = (N \times N \times N)$

- $\text{Init}(c,m)$
- $\text{Transfer}(c_1,c_2,m)$
- $\text{Truncate}(c,m)$

5: Transfer $(B,D,2)$

7: Truncate $(C,5)$

What is the maximum amount of different sensitive data possibly stored in $D$ at time 6?

MAX-FLOW $((S,0),(D,6))$

Formalization

- $\text{Node} = \text{Container} \times \mathbb{N}$
- $\text{Graph} = \text{Node} \times \text{Node} \rightarrow \mathbb{N}$
- $\text{Step}: (\text{Graph} \times \text{Event}) \rightarrow \text{Graph}$
- $\Sigma = \text{Data} \rightarrow \text{Graph}$
- $\sigma: (\mathbb{N} \times \text{Event}) \rightarrow \Sigma$
  - $g(\sigma(a)) = \sigma' \Rightarrow \forall \sigma \in \text{Data}: \sigma'(d) = \text{step}(\sigma(d),a)$

Formalization (2)

- $\text{states}(\mathbb{N} \times \Sigma) \rightarrow \Sigma$
  - $\text{states}(\mathbb{N},0) = \sigma_0$
  - $\text{states}(\mathbb{N},t) = g(\text{states}(\mathbb{N},t-1),\mathbb{N}(t-1))$
Formalization (2)
Let $\mathcal{S} : \text{Trace} \times N \to \Sigma$
- $\mathcal{S}_t((r, 0)) = s_t$
- $\mathcal{S}_t((r, t)) = \{ \mathcal{S}_t((r, t-1), tr(t-1)) \}$

Let $\Phi_q = \text{atMostInEach}(\text{Data}, N, \mathcal{2}^{\text{Container}})$

AtMostInSet(\text{Data}, N, \mathcal{2}^{\text{Container}})

- $\forall r \in \text{Trace}, \forall t \in N, \forall \phi \in \Phi_q, \forall s \in \Sigma : (tr, t) \Rightarrow \phi \iff$
  $s = \mathcal{S}_t((r, t)) \land \exists d \in \text{Data}, \exists C \in \mathcal{2}^{\text{Container}}, \exists Q \in N :$
  $\phi = \text{atMostInEach}(d, Q, C) \land \forall c \in C : K(s(d), c) \leq Q \lor$
  $\phi = \text{atMostInSet}(d, Q, C) \land \sum_{c \in C} K(s(d), c) \leq Q$

State-based operators

- $\text{inData}(D_1, C_1, C_2)$ true if data $D_1$ is stored in none of containers $C_1$ and $C_2$
- $\text{inData}(D_2, C_1, C_2)$ true if no container other than $C_1$ or $C_2$ contains data $D_1$
- $\text{combinedWith}(D_1, D_2)$ true if data $D_1$ and $D_2$ are stored in the same container
- $\text{atMostInEach}(D_1, n, (C_1, C_2))$ true if both $C_1$ and $C_2$ contain less than $n$ units of data $D_1$
- $\text{atMostInSet}(D_1, n, (C_1, C_2))$ true if $C_1$ and $C_2$ contain in total less than $n$ units of data $D_1$

Experimental results

- Scenario: Phone specification

Experimental results

- Scenario: Phone specification
  - Implementation: OpenBSD, Systrace
  - Settings:
    - Repositories (→Files)
    - Monitor usage by Alice
    - Observe sent mails (→Files)
    - Enforce quantitative policies
    - Preventive enforcement

Theorem

- Estimated number of tainted units in each node greater than or equal to actual number of tainted units
- Induction over size of the provenance graph: different steps add nodes in a specific way
Results

- Precision = \( \frac{\text{Size} - \text{estimated # of tainted blocks}}{\text{Size} - \text{exact # of tainted blocks}} \)
  - If \( \text{Size} = \text{Exact} = \text{Estim} \), Precision = 1

- Variables:
  - \( PS = \) Probability that source of a transfer is a specification (100% sensitive) vs existing report
  - \( PN = \) Probability that destination of a transfer is a new report vs update existing report
  - No meaningful impact on precision
  - Meaningful impact on precision

- 100 experiments for each \( (\text{num_rep}, PS, PN) \) triple

Experimental results

- Performance
  - Scalability issues
  - Graph simplification
  - Trace dependent
  - Asymptotic behavior

- Precision

Related work: quantitative information flow

- Information leakage defined by different measures: min-entropy, Shannon entropy, guessing entropy
- Require probability distribution of secrets (input to channel or system)
- In contrast, we want to protect one data item for which, in general, no probability distribution can be known

Concerns

- Precise meaning of numbers?
- Performance
- Coding and compression
  - May be assumed in controlled environments
- Technology possibly better suited for anomaly detection in IDS

Agenda

- Part I: Introduction
- Part II: Event-Based Usage Control
- Part III: Data-Centric Usage Control
- Part IV: Quantitative Usage Control
- Part V: Local Single-Layer Enforcement
- Part VI: Distributed Enforcement
- Part VII: Cross-Layer Enforcement
- Part VIII: Policy Derivation
- Part IX: Discussion

THURSDAY
Agenda

- Part I: Introduction
- Part II: Event-Based Usage Control
- Part III: Data-Centric Usage Control
- Part IV: Quantitative Usage Control
- Part V: Local Single-Layer Enforcement
- Part VI: Distributed Enforcement
- Part VII: Cross-Layer Enforcement
- Part VIII: Policy Derivation
- Part IX: Discussion

Prevention

- Provider can guarantee adherence to an obligation
- Control mechanisms in DRM
- Adobe LiveCycle, Windows RMS
- Amount of control platform-dependent

Detection

- Provider can detect violation of an obligation
- Take compensating action
- 100% security not desired
- Trustworthy signalers and monitors

Different requirements and trust models. Technically very similar.

Architecture

At Runtime

Deployment

Runtime enforcement

- PMP manages policy lifecycle
- Independent of deployed policies:
  - PEP intercepts intended and actual events
  - PIP implements transition relation and tracks data flows
- Depending on deployed policies:
  - PDP configured by ILPs
  - Technology: runtime verification, complex event processing
- Possible distinction in PEPs: signalers and monitors
  - Detective ~ preventive?
Instantiations

Agenda

- Part I: Introduction
- Part II: Event-Based Usage Control
- Part III: Data-Centric Usage Control
- Part IV: Quantitative Usage Control
- Part V: Local Single-Layer Enforcement
- Part VI: Distributed Enforcement slides by Florian Kelbert
- Part VII: Cross-Layer Enforcement
- Part VIII: Policy Derivation
- Part IX: Discussion

Architecture

- So far: single-system data flow tracking
- E.g. at the level of the operating system
- Intercepting system calls using systrace
- Enforcement of state-based policies
Goal

- Track the flow of data across systems
- Connection Establishment
- Data Transmission
- Enforce policy on receiving system
- Sticking policies to data

Containers
- Files
- Processes
- Pipes
- Network sockets

Names
- Filenames
- File descriptors
- Process IDs
- Socket names

Principals
- Actions
  - Systemcalls: open(), read(), write(), close(), pipe(), dup(), unlink(), kill(), rename(), execve(), fork(), socket(), bind(), connect(), accept(), send(), recv(), ...
Process with PID 11 running on host with IP Address A
Process with PID 22 running on host with IP Address B

socket()
- Create an endpoint (Socket) for communication
- Socket is identified by a file descriptor relative to the process

bind()
- Bind a name (IP Address + Port) to a socket
- Either explicit or implicit on succeeding system call
- PID 11 binds socket to IP Address A and Port X
- PID 22 binds socket to IP Address B and Port Y
Bind a name (IP Address + Port) to a socket
Either explicit or implicit on succeeding system call
- PID 11 binds socket to IP Address A and Port X
- PID 22 binds socket to IP Address B and Port Y

listen()
- Marks a socket as passive and listening for connections
- Socket will accept incoming connection requests using accept()

connect()
- Initiate a connection on a socket
- Connects to the socket A:X that must be listening for connections

accept()
- Accept a connection on a socket
- Creates a new connected socket
- Original socket unaffected by this call
- Will listen for further connection requests
Accept a connection on a socket
- Accepts a new connected socket
- Original socket unaffected by this call
- Will listen for further connection requests

Distributed Data Flow State
- Things are slightly more complicated: no global PIP
- Each system keeps track of its own Data Flow state
  - One PIP per system
- PIPs of independent systems must be synchronized
- Union of the local Data Flow states results in the global Data Flow state

| PIP 1 | + | PIP 1 | → | PIP 1 |
Recall the Problem
Performance

(video overhead by one control and one data channel)

A. Pretschner: Usage Control. Bertinoro 2014

Performance

<table>
<thead>
<tr>
<th>number of syscalls</th>
<th>FTP Server: vsftpd</th>
<th>HTTP Server: Apache</th>
</tr>
</thead>
<tbody>
<tr>
<td>100KB</td>
<td>100KB</td>
<td>128MB</td>
</tr>
<tr>
<td>4MB</td>
<td>4MB</td>
<td>4MB</td>
</tr>
<tr>
<td>128MB</td>
<td>128MB</td>
<td>128MB</td>
</tr>
</tbody>
</table>

write()  
- 44  139  417  
- 122  996  32740

total        
- 148  355  5294  
- 52  1523  49130

 performance overhead factor w.r.t. native execution
- best case  
  - 11.66  0.61  0.14  
  - 7.29  0.53  0.25
- worst case 
  - 13.15  1.16  0.05  
  - 9.01  4.90  4.65

Video: Distributed Data-Centric UC

- Video 5: usage control via ftp
  
http://www22.in.tum.de/fileadmin/demos/uc/FTPDemo.avi

Agenda

- Part I: Introduction
- Part II: Event-Based Usage Control
- Part III: Data-Centric Usage Control
- Part IV: Quantitative Usage Control
- Part V: Local Single-Layer Enforcement
- Part VI: Distributed Enforcement
- Part VII: Cross-Layer Enforcement
- Part VIII: Policy Derivation
- Part IX: Discussion

Why multiple layers?

- Alternative: tracking at level of machine code
- Loss of semantic information: how to determine „screenshot“ at level of machine code?
  - Precision
  - Performance

Different levels of abstraction (not stacked!)

- Business Process
- User application
- Application wrapper
- Service wrapper
- Infrastructure Application
- Runtime System
- Operating System
- CPU/VM (ring -1)

A. Pretschner: Usage Control. Bertinoro 2014
Problem description

How can we track flows of data in-between representations across different l.o.a.?

Thunderbird example: save email

considering OS only would add e as well!

Five ingredients

• To define cross-layer flows: Three kinds of behaviors
• Scope
• Intermediate containers
• Cross-layer aliases
• Cross-layer transition relations
Behaviors \( \text{BEHAV} = \{\text{IN}, \text{OUT}, \text{INTRA}\} \)

- Container receives data from container at same level
- Container receives data from container at different level
- Container sends data to container at different level

Cross-layer flow: A pair of causally related IN and OUT events

Five ingredients

- Three kinds of behaviors
- Scope
- Intermediate containers
- Cross-layer aliases
- Cross-layer transition relations

Scopes I

Events have duration. Scopes capture start, events at other layers, end.

Scopes II

Possibly multiple scopes at the same time.
Yet, one event belongs to exactly one scope:
\( X_{\text{BEHAV}} : (\Sigma \times E) \rightarrow (\text{BEHAV} \times \text{SCOPE}) \)

Events may activate or deactivate scopes:
\( X_{\text{EVENT}} : (\Sigma \times E) \rightarrow P(\{\text{OPEN}, \text{CLOSE}\} \times \text{SCOPE}) \)

Scopes example

Intermediate Containers

Five ingredients

- Three kinds of behaviors
- Scope
- Intermediate containers: the pipe between two layers
- Cross-layer aliases
- Cross-layer transition relations

Intermediate Containers example
Example: TB+OS – read

```
load_s(fn) open(fn,fd) read(fd,b1) read(fd,bn) close(fd)...
```

Example: TB+OS – write

```
save_s(file,fn) open(fn,fd) write(fd,b1) write(fd,bn) close(fd)...
```

Five ingredients

- Three kinds of behaviors
- Scope
- Intermediate containers
- Cross-layer aliases
- Cross-layer transition relations

Cross-Layer Aliases

Cross-layer alias example. In correspondence of event `DECRYPT(G)` at TB layer a new data $d$ appears in the container $\gamma_B$ but no corresponding event takes place at the OS layer. Nevertheless, data $d$ is propagated via cross-layer alias to the container that represents the memory of the Thunderbird process at the OS layer $\gamma_O$.

\[
\begin{align*}
\lambda_B: (\gamma_B \rightarrow P(G_{TB})) \cup (\gamma_B \rightarrow P(G_{OS})) \\
\lambda_O: (\gamma_O \rightarrow P(\gamma_B)) \cup (\gamma_O \rightarrow P(\gamma_B)) \\
X_{ALIAS}: (\gamma \times \xi) \rightarrow (\gamma_{TB} \rightarrow P(G_{TB})) \cup (\gamma_{OS} \rightarrow P(G_{OS}))
\end{align*}
\]

Cross-layer transition relation

... copy data to intermediate container

\[
\begin{align*}
\sigma_A = (\theta_A, I_A, F_A, SC_A) \\
\theta_B = (\theta_B, I_B, F_B, SC_B) \\
\mathbf{S}_{TB}^A = \mathbf{S}_{TB}^B \\
\mathbf{R}_{TB}^A = \mathbf{R}_{TB}^B(\mathbf{F}_{TB}^A) \\
\mathbf{S}_{OS}^A = \mathbf{S}_{OS}^B
\end{align*}
\]
Example Thunderbird+OS

Algorithm 2: X_THUNDERBIRD

1. for each IDE₁ in Thunderbird do
2.   if e = SAVING_EMAIL₁ then
3.     return ([[OPEN] ‘-saving email to file ErrorHandler’]);
4.   else if e = SAVING_EMAIL₂ then
5.     return ([[CLOSE] ‘TB saving mail to file ErrorHandler’]);
6.   else if e = LOAD_EMAIL₁ then
7.     return ([[OPEN] ‘TB attaching file ErrorHandler to mail ErrorHandler’]);
8.   else if e = LOAD_EMAIL₂ then
9.     return ([[CLOSE] ‘TB attaching file ErrorHandler to mail ErrorHandler’]);
10. end
11. return (null)

Example TB+OS

Algorithm 3: X_TB\_OS

1. if e = OPEN_EMAIL₁ then
2.   if m =-ending message then
3.     return ([[OPEN] ‘TB adding file ErrorHandler to mail ErrorHandler’]);
4.   else
5.     return (null);
6. end
7. return (null)

Not trivial:

- Identification of start and stop events

Video: Cross-Layer Data-Centric UC

- Video 6: distributed data-driven usage control for smart meter readings rendered in a social network
  [http://www22.in.tum.de/fileadmin/demos/pec/uc4win6_internet.mp4](http://www22.in.tum.de/fileadmin/demos/pec/uc4win6_internet.mp4)
- Video 7: distributed data-driven usage control in a social network application
  [http://www22.in.tum.de/fileadmin/demos/uc/final-thund-cloud.htm](http://www22.in.tum.de/fileadmin/demos/uc/final-thund-cloud.htm)

Agenda

- Part I: Introduction
- Part II: Event-Based Usage Control
- Part III: Data-Centric Usage Control
- Part IV: Quantitative Usage Control
- Part V: Local Single-Layer Enforcement
- Part VI: Distributed Enforcement
- Part VII: Cross-Layer Enforcement
- Part VIII: Policy Derivation (sketch)
  slides by Prachi Kumari
- Part IX: Discussion

TIME …

…for policy derivation?
A Motivating Use Case

Problem

Domain Meta Model

WBSN Domain Model

Excerpt

Usage Control + Domain meta-models

Combine the two models.

Define an action in terms of:

1. Sets/sequences of transformers
2. Resulting system state
1- Refinement as disjunction

(Filter irrelevant transformers)

2- Refinement as sequence

2- Refinement as sequence

2- Refinement as sequence

3- Refinement using States

- Defining state formula encoded in definition of actions

\[ \text{not(isNotIn}(d, \{c_1, c_2, c_3\})) \]
Action Refinement using States

- Defining state formula encoded in definition of actions

\[ \text{not(isNotIn}(d, \{c_1, c_2, c_3\})) \]

Action Refinement using States

- Defining state formula encoded in definition of actions

\[ \text{not(isNotIn}(d, \{c_1, c_2, c_3\})) \]

Action Refinement using States

- Actual data and containers not known at declaration time
  - Use ISM/PSM containers (classes of containers) ...
  - Use variables ...

\[ \Phi_2 := \text{inNotIn}(\text{Data}, \text{P Container}) \mid \text{isOnlyIn(Data, Data, Data)} \mid \text{isCombinedWith(Data, Data)} \]

\[ \Phi_3 := \text{inNotIn(Data, P ISMContainer)} \mid \text{inNotIn(Data, P PSMContainer)} \mid \text{isOnlyIn(Data, P ISMContainer)} \mid \text{isOnlyIn(Data, P PSMContainer)} \mid \text{isCombinedWith(Data, Data)} \]

- and decide at runtime which class an actual container belongs to

Action Refinement using States

- Actual data and containers not known at declaration time
  - Use ISM/PSM containers (classes of containers)
  - Use variables ...

\[ \Phi_2 := \text{inNotIn(Data, P ISMContainer)} \mid \text{inNotIn(Data, P PSMContainer)} \mid \text{isOnlyIn(Data, P ISMContainer)} \mid \text{isOnlyIn(Data, P PSMContainer)} \mid \text{isCombinedWith(Data, Data)} \]

\[ \text{Var} := V(k) \]

\[ \text{VerData} := \text{Var} \cup \text{Data} \]

\[ \Phi_3 := \text{inNotIn(VerData, P ISMContainer)} \mid \text{inNotIn(VerData, P PSMContainer)} \mid \text{isOnlyIn(VerData, P ISMContainer)} \mid \text{isOnlyIn(VerData, P PSMContainer)} \mid \text{isCombinedWith(VerData, Data)} \]

- and bind these variables to actual data at deployment time

So, we'll check at runtime if a container is of class Clipboard

\[ \text{not(isNotIn}(d, \text{Clipboard})) \]
Action Refinement using States

Copy photo

\[ \text{not(isNotIn}(d, \text{Clipboard})) \]

Action Refinement using States

… and we’ll bind a concrete data item to the variable \( d \) at deployment time

Copy photo

\[ \text{not(isNotIn}(\text{photo}, \text{Clipboard})) \]

Finally …

Combining event and state–based refinements

- Complete action refinement is
  - A disjunction over the event- and state-based refinements

Policy derivation: from SLPs to ILPs

- Once the domain models are defined, we need to transform future-time SLPs to past-time ILPs
- No constructive methods known
- Use templates in graphical editor

Disjunctions and sequences of events

- Events may be refined into disjunction of events at more than one layer of abstraction
  - „Copy“ becomes „save email“ at Thunderbird level and „open() write() close“ at OS level
- Two (plus a third, later!) possible strategies
  - Deploy only one policy that captures all layers simultaneously
  - Project policy to layers of abstraction and deploy each projection

Disjunctions and sequences of events II

- Either way, a policy may be evaluated and enforced multiple times at different layers!
  - „Don’t copy more than twice“: one physical „copy“ action captured by
    - one „save“ event at TB layer
    - one „write“ event at OS level
- Sometimes no problem: \( \text{always(not copy)} \) can be simultaneously enforced at several layers
Disjunctions and sequences of events III

- Sometimes problematic:
  - Executors at multiple layers lead to duplicate effects, e.g., `always(copy implies notify)`
  - Critical: Counting and "stateful" temporal formulas (repmax, until; minimized monitor has more than one state)
  - In these cases, pick one layer and ignore the other one
  - Possible unless a policy correlates events at different layers, e.g., "no screenshot until mail client disabled"

Related: Distributed Systems

- "Distribute at most n times"
  - Local enforcement: each node can do n distributions, overall number not restricted
  - Global enforcement: overall number restricted to n
- Here, in contrast, we want to make sure that events in multiple systems (~ at multiple layers) are counted each
- Requires shared data state

Agenda

- Part I: Introduction
- Part II: Event-Based Usage Control
- Part III: Data-Centric Usage Control
- Part IV: Quantitative Usage Control
- Part V: Local Single-Layer Enforcement
- Part VI: Distributed Enforcement
- Part VII: Cross-Layer Enforcement
- Part VIII: Policy Derivation
- Part IX: Discussion

Assumptions

- Layer-specific events interceptable
- Omnipresence of UC infrastructures
- Integrity and authenticity of policies ensured
- End-to-end confidentiality of data ensured
- UC infrastructure and underlying technology secure

Attacker Models and Guarantees

- Assets, goals, detective vs. preventive enforcement
- Benevolent and malevolent regular users and administrators; external intruders
- Malware
- Guarantees layer-specific and depending on attacker

Protecting the infrastructure

- BonaFides system: record logs of file changes (iNotify) and protect them using TPM
  - Detective not preventive
Insecure Security

- Yes solution can be circumvented
- Like any other security solution: a matter of commitment and resources
- Security as a sub-discipline of risk management!
- How much technology will be in a „security solution“?

Concerns

- Overapproximations
  - Quantitative measurements (but what does this mean?)
  - Declassification
- Performance
- (Formal) guarantees
- Collection of usage information in itself likely to impact privacy
- Business model - omnipresence of UC infrastructures?
- DEPENDS ON SPECIFIC APPLICATION SCENARIO!

Related Work

- Access control
- Usage control
- Possibilistic information flow control
- Quantitative information flow measurements
- Complex event processing
- DRM
- Data loss prevention
- Android security
- Runtime verification
- Enforcement automata
- Model-based development
- Semantics of sequence charts

Wrap-Up

- Part I: Introduction
- Part II: Event-Based Usage Control
- Part III: Data-Centric Usage Control
- Part IV: Quantitative Usage Control
- Part V: Local Single-Layer Enforcement
- Part VI: Distributed Enforcement
- Part VII: Cross-Layer Enforcement
- Part VIII: Policy Derivation
- Part IX: Discussion

Is this a good idea?