



Consistency Policies for Dynamic Information Systems with Declassification Flows



ICISS 2011 - Kolkata, India

Julien A. Thomas, Nora Cuppens-Boulahia and Frédéric Cuppens

Institut Télécom ; Télécom Bretagne ; SFIS Team
Université Européenne de Bretagne

15-19 December 2011

The study is partially funded by the Délégation Générale pour l'Armement under a DGA / CNRS grant

Outline

- 1 Context
- 2 Principle of Consistency
- 3 Consistency Policy
- 4 Conclusion

Outline

- 1 Context
- 2 Principle of Consistency
- 3 Consistency Policy
- 4 Conclusion

Multilevel Information Systems

- ⊕ An information flow is the transfer of information from a variable x to a variable y in a given *process*
- ⊕ Multilevel Information Systems
 - ⊕ Process information with different sensitivities
 - ⊕ security level lattices
 - ⊕ Permit simultaneous access by users with different security clearances
 - ⊕ Prevent illegal flow from higher level to lower level

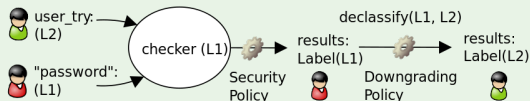
Multilevel Information Systems

- ⊕ An information flow is the transfer of information from a variable x to a variable y in a given *process*
- ⊕ Multilevel Information Systems
 - ⊕ Process information with different sensitivities
 - ⊕ security level lattices
 - ⊕ Permit simultaneous access by users with different security clearances
 - ⊕ Prevent illegal flow from higher level to lower level
- ⊕ Security properties: specify the authorized flows
- ⊕ Security policy: enforce the security properties

The declassification of data

- ⊕ A user u may access to data classified $L2$ but not $L1$ ($L2 \prec L1$)
- ⊕ u try to log in with its password classified $L2$. The result relies on the comparison with the protected password classified $L1$
- ⊕ *result* is classified $L1$... but must be accessed by u
- ⊕ *result* is declassified from $L1$ to $L2$ ($L2 \prec L1$)

Example of declassification : password checking



Dynamic Information Systems

- ⊕ Dynamic Mechanisms
 - ⊕ Databases: Triggers
 - ⊕ Event based information systems
 - ⊕ An action $\alpha_1(P)$ triggers another action $\alpha_2(P_2)$, providing ...

Dynamic Information Systems

- ⊕ Dynamic Mechanisms
 - ⊕ Databases: Triggers
 - ⊕ Event based information systems
 - ⊕ An action $\alpha_1(P)$ triggers another action $\alpha_2(P_2)$, providing ...
- ⊕ Model for Dynamic Information Systems:
 - Event-Condition-Action*-rules and the L_{active} language

- ⊕ actions

```
do( $\alpha$ ) causes op1, ..., opk if q1(X1), ... qn(Xn);
```

- ⊕ events

```
event_name(X) after do( $\alpha$ ) if r1(X1) rm(Xm) ;
```

- ⊕ rules

```
rule: event_name(X) initiates do( $\alpha_1$ ), ... do( $\alpha_k$ )
if t1(X1), ... tp(Xp);
```

Dynamic Multilevel Information Systems

- ⊕ Multilevel ECA rules

ECA Rules Security

do(α, Lc, Li) causes *f*(Y) if $q_1(X_1) \dots q_n(X_n)$
event_name($X, L1, Li1$) after *do*($\alpha, L2, Li2$) if $r_1(Z_1) \dots r_m(Z_m)$
rule_name : *event_name*(X, L_0, Li_0) initiates *do*(α_1, L_1, Li_1) ...
do(α_n, L_n, Li_n) if $t_1(Z_1) \dots t_m(Z_m)$

- ⊕ Multilevel Security Policy
 - ⊕ for confidentiality
 - ⊕ for integrity

Declassification Policy

- ➔ Based on ECA rules

Declassification Rule

```
do(declassify(P,L)) causes  
delete(classification_level(P,L')), insert(classification_level(P,L))  
if classification_level(P,L')  $\wedge$  inf_level(L,L');
```

Declassification Policy

- ⊕ Based on ECA rules

Declassification Rule

```
do(declassify(P,L)) causes
delete(classification_level(P,L')), insert(classification_level(P,L))
if classification_level(P,L')  $\wedge$  inf_level(L,L');
```

- ⊕ Trust Based Downgrading: A user u is allowed to downgrade a piece of information o if u is trusted for the downgrading order on o .

Trust based downgrading

- ⊕ As an access control requirement
- ⊕ $is_permitted \in USERS \times ACTIONS$

Declassification Policy

- ⊕ Based on ECA rules

Declassification Rule

```
do(declassify(P,L)) causes
delete(classification_level(P,L')), insert(classification_level(P,L))
if classification_level(P,L')  $\wedge$  inf_level(L,L');
```

- ⊕ Automatic Downgrading: the downgrading operation is based on the occurrence of a specific event

Automatic downgrading

- ⊕ `declassify_rule: declassify_event(P,L) initiates do(declassify(P,L));`
- ⊕ Specifying when `declassify_event(P, L)` occurs
- ⊕ `declassify_event(warCost(Cost), \perp) after do(insert(victory)) if endOfWar`

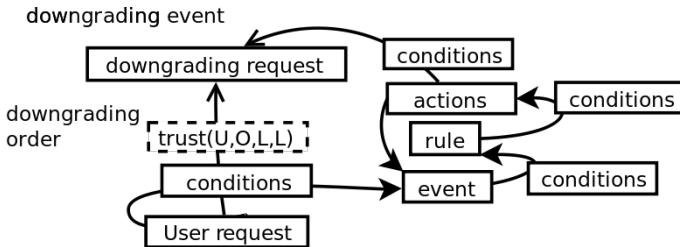


Declassification Model

➔ Based on ECA rules

Declassification

```
do(declassify(P,L)) causes
delete(classification_level(P,L')), insert(classification_level(P,L))
if classification_level(P,L')  $\wedge$  inf_level(L,L');
```



Outline

1 Context

2 Principle of Consistency

- Principle of Consistency
- Need of Consistency
- Consistency

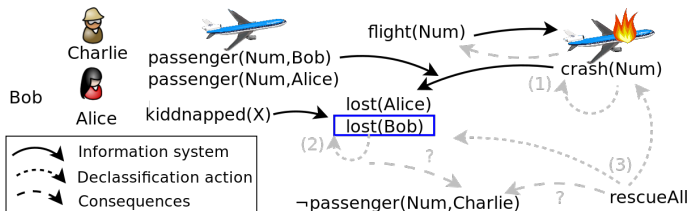
3 Consistency Policy

4 Conclusion

Principle of Consistency

- ⊕ Impossibility to obtain a state where the security policy is inconsistent
 - ⊕ Two classifications are assigned to an object : one explicit, L_r , and the other implicit, L_v
 - ⊕ $L_r \not\subseteq L_v$ [CG01].

Need of Consistency



- ⊕ Some facts
 - ⊕ The declassification of $crash(Num)$ implies the declassification of $flight(Num)$
- ⊕ Some doubts
 - ⊕ Does the declassification of $lost(Alice)$ imply the declassification of $crash(Num)$ and $flight(Num)$?
 - ⊕ Does the declassification of $crash(Num)$ imply the declassification of $lost(Alice)$?

Consistency Property

- ⊕ *Non-Deducibility* property [D.S86]
 - ⊕ A system is consistent if for any user u , the state of any information classified L with $L \sqsubseteq L_U$ may be explained by information flows which only rely on information classified L_2 with $L_2 \sqsubseteq L_U$.
- ⊕ When is a piece of information inferable?
 - ⊕ Inference of a piece of data A classified L_r because of
 - ⊕ a piece of data B declassified to L_v with $L_r \not\sqsubseteq L_v$
 - ⊕ the non-declassification of A
- ⊕ What is an explanation?

System Model

Hypothesis

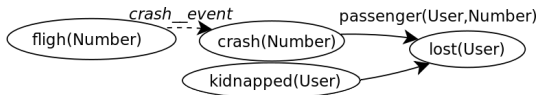
Graphs are well suited to model active rules and build explanations

System Model

Hypothesis

Graphs are well suited to model active rules and build explanations

- ⊕ The system is a direct graph with weights
 - ⊕ Each state is an action
 - ⊕ Each transition is a couple (event-rule)
 - ⊕ a weight is the set of conditions associated to the transition

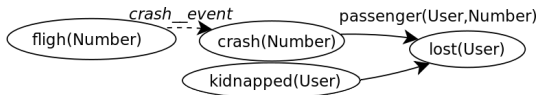


System Model

Hypothesis

Graphs are well suited to model active rules and build explanations

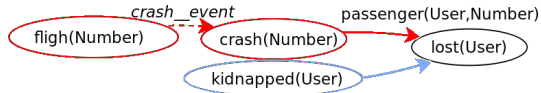
- ⊕ The system is a direct graph with weights
 - ⊕ Each state is an action
 - ⊕ Each transition is a couple (event-rule)
 - ⊕ a weight is the set of conditions associated to the transition



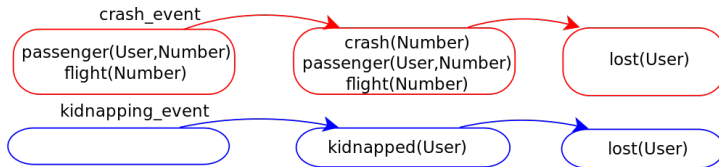
- ⊕ In the algorithms, transitions are refined, based on the considered security levels
 - ⊕ A conditioned transition is valid for L if each condition C of $P_S \upharpoonright L$ is valid

Explanations

- ⊕ An explanation at the security level L consists in a valid evolution of the system state where
 - ⊕ Each node is a valid state
 - ⊕ Each conditioned transition is valid for L
- ⊕ Considering the declassification of $Lost(User)$

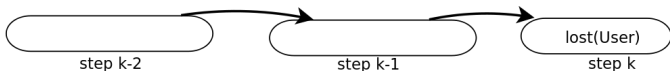


- ⊕ The valid explanations are the following

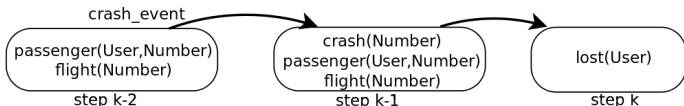


Evaluation of the consistency

- ⊕ The evaluation of the system consistency consists in building the graph G_{all} where
 - ⊕ Each node is a state of the system
 - ⊕ Each fact of a state is certain (*Non-Deducibility*)
 - ⊕ Each action responsible for a transition is certain
 - ⊕ The step number is evaluated based the initial declassification action
- ⊕ Examples
 - ⊕ If $\neg kidnapped(User)$ is secret



- ⊕ If $\neg kidnapped(User)$ is public



Outline

1 Context

2 Principle of Consistency

3 Consistency Policy

- Evaluation of the Explanation Graphs
- Consistency Policy
- Composition of the Consistency and Confidentiality Policies

4 Conclusion

Evaluation of the Explanation Graphs

- ⊕ System invariants and hypotheses
 - ⊕ When a fact is revealed in a state, it has to be revealed in the parent and child ones
 - ⊕ When a fact is inserted in a state, it has to be revealed in the child ones

Evaluation of the Explanation Graphs

- ⊕ System invariants and hypotheses
 - ⊕ When a fact is revealed in a state, it has to be revealed in the parent and child ones
 - ⊕ When a fact is inserted in a state, it has to be revealed in the child ones
- ⊕ Explicit rules for deduction based inference
 - ⊕ When a node N_i with $\alpha(P)$ is created, a graph is created for each valid conditioned transition to $\alpha_2(P_2)$
 - ⊕ When a node N_i with $\alpha(P)$ is created, a graph is created for each possible falling condition that would stop the action flow

Evaluation of the Explanation Graphs

- ⊕ System invariants and hypotheses
 - ⊕ When a fact is revealed in a state, it has to be revealed in the parent and child ones
 - ⊕ When a fact is inserted in a state, it has to be revealed in the child ones
- ⊕ Explicit rules for deduction based inference
 - ⊕ When a node N_i with $\alpha(P)$ is created, a graph is created for each valid conditioned transition to $\alpha_2(P_2)$
 - ⊕ When a node N_i with $\alpha(P)$ is created, a graph is created for each possible failing condition that would stop the action flow
- ⊕ Explicit rules for abduction based inference
 - ⊕ When a node N_i with $\alpha(P)$ is created, a graph is created for each valid conditioned transition to $\alpha(P)$
 - ⊕ For each possible failing condition, a graph is also created

Concrete Algorithms

- ⊕ *Non-Deducibility*:
 - ⊕ History at the action level
 - ⊕ Costly computations (abductive and deductive reasonings)
- ⊕ Refinement of the security: Non Interference
 - ⊕ Inference at the object layer: greedy algorithm
 - ⊕ $dependencies(P)$ and $child_dependencies(P)$ sets
 - ⊕ the declassification of P implies the declassification of each $o \in dependencies(P)$

Concrete Algorithms

- ⊕ *Non-Deducibility*:
 - ⊕ History at the action level
 - ⊕ Costly computations (abductive and deductive reasonings)
- ⊕ Refinement of the security: Non Interference
 - ⊕ Inference at the object layer: greedy algorithm
 - ⊕ $dependencies(P)$ and $child_dependencies(P)$ sets
 - ⊕ the declassification of P implies the declassification of each $o \in dependencies(P)$
 - ⊕ Inference at the action layer: history based algorithm
 - ⊕ $dependencies(\alpha)$ set
 - ⊕ the declassification of α implies the declassification of each $\alpha_i \in dependencies(\alpha)$ and the maintenance of the action chain consistency

Concrete Algorithms

- ⊕ *Non-Deducibility*:
 - ⊕ History at the action level
 - ⊕ Costly computations (abductive and deductive reasonings)
- ⊕ Refinement of the security: Non Interference
 - ⊕ Inference at the object layer: greedy algorithm
 - ⊕ $dependencies(P)$ and $child_dependencies(P)$ sets
 - ⊕ the declassification of P implies the declassification of each $o \in dependencies(P)$
 - ⊕ Inference at the action layer: history based algorithm
 - ⊕ $dependencies(\alpha)$ set
 - ⊕ the declassification of α implies the declassification of each $\alpha_i \in dependencies(\alpha)$ and the maintenance of the action chain consistency
 - ⊕ Management of the non-declassifications at the object layer: greedy algorithm
 - ⊕ $implicit_dependency(L_P)$ and $implicit_parent(P)$ sets

Consistency Policy

- ⊕ Several consistency policies are possible
 - 1 Intransitive Declassification: the action α performed in the state s_{n-1} is authorized if the resulting state s_n engenders no additional declassification at s_n
 - 2 Strict Non Interference: the action α performed in the state s_{n-1} is authorized if the resulting state s_n engenders no additional declassification
 - ⊕ The declassification of a fact engenders the declassification of the predicate
 - 3 Authoritative Declassification: the action α performed in the state s_{n-1} is authorized, not matter the declassifications it engenders

Composition of the Consistency and Confidentiality Policies

- ⊕ The consistency policy may engender additional information flows
 - ⊕ The declassification may be refused due to non public conditions
 - ⊕ refusal of declassification due to the declassification of another non public piece of data
 - ⊕ The security policy must not generate insecure information flows

Composition of the Consistency and Confidentiality Policies

- ⊕ The consistency policy may engender additional information flows
 - ⊕ The declassification may be refused due to non public conditions
 - ⊕ refusal of declassification due to the declassification of another non public piece of data
 - ⊕ The security policy must not generate insecure information flows
- ① If the user issuing the declassification order has a clearance level superior or equal to the ones of the objects to declassify ...
 - ⊕ The refusal of declassification engenders no additional knowledge
 - ⊕ Any of the consistency policies may be used

Composition of the Consistency and Confidentiality Policies

- ⊕ The consistency policy may engender additional information flows
 - ⊕ The declassification may be refused due to non public conditions
 - ⊕ refusal of declassification due to the declassification of another non public piece of data
 - ⊕ The security policy must not generate insecure information flows
- ① If the user issuing the declassification order has a clearance level superior or equal to the ones of the objects to declassify ...
 - ⊕ The refusal of declassification engenders no additional knowledge
 - ⊕ Any of the consistency policies may be used
- ② Otherwise, the security policy must rely on the *Authoritative Declassifications* policy

Outline

- 1 Context
- 2 Principle of Consistency
- 3 Consistency Policy
- 4 Conclusion

Conclusion

- ⊕ Contributions
 - ⊕ Formalization of the consistency property
 - ⊕ For dynamic information
 - ⊕ For declassification flows

Conclusion

- ⊕ Contributions
 - ⊕ Formalization of the consistency property
 - ⊕ For dynamic information
 - ⊕ For declassification flows
 - ⊕ Formalization of the consistency policy
 - ⊕ Consistency policies
 - ⊕ Consistency and Confidentiality

Conclusion

⊕ Contributions

- ⊕ Formalization of the consistency property
 - ⊕ For dynamic information
 - ⊕ For declassification flows
- ⊕ Formalization of the consistency policy
 - ⊕ Consistency policies
 - ⊕ Consistency and Confidentiality
- ⊕ Evaluation of the consistency
 - ⊕ Abstract evaluation for Non Deducibility
 - ⊕ Concrete algorithms for Non Interference

Conclusion




⊕ Contributions

- ⊕ Formalization of the consistency property
 - ⊕ For dynamic information
 - ⊕ For declassification flows
- ⊕ Formalization of the consistency policy
 - ⊕ Consistency policies
 - ⊕ Consistency and Confidentiality
- ⊕ Evaluation of the consistency
 - ⊕ Abstract evaluation for Non Deducibility
 - ⊕ Concrete algorithms for Non Interference

⊕ Future Works

- ⊕ Complexity of the algorithms
- ⊕ Proof of the completeness and correctness of the explanations
 - ⊕ formal and temporal logics (in progress)

References I

-  Frédéric Cuppens and Alban Gabillon, *Cover story management*, Data Knowl. Eng. **37** (2001), no. 2, 177–201.
-  D.Sutherland., *A model of information*, Proceedings of the 9th National Computer Security Conference, 1986.
-  Julien Thomas, Nora Cuppens-Boulahia, and Frédéric Cuppens, *Expression and Enforcement of Confidentiality Policy in Active Databases*, MEDES 2010, 5th International ACM Conference on Management of Emergent Digital EcoSystems, 26 October - 29 October 2010, Bangkok, Thailand, LUSI - Dépt. Logique des Usages, Sciences Sociales et de l'Information (Institut Télécom-Télécom Bretagne), 2010.

References II



_____, *Declassification Policy Management in Dynamic Information Systems*, ARES 2011, The Sixth International Conference on Availability, Reliability and Security. 22 August - 26 August 2011, Vienna, Austria, LUSI - Dépt. Logique des Usages, Sciences Sociales et de l'Information (Institut Télécom-Télécom Bretagne), 2011.

Security Properties based on traces

- ⊕ Without declassification action [TCBC10]
 - ⊕ $ext(t)$: external (users) actions of t
 - ⊕ $ext(t) \sim_L ext(t') \Leftrightarrow s_0 \approx_L s'_0 \wedge ext(\alpha) \upharpoonright L = ext(\alpha') \upharpoonright L$

Security Properties based on traces

- ⊕ Without declassification action [TCBC10]
 - ⊕ $ext(t)$: external (users) actions of t
 - ⊕ $ext(t) \sim_L ext(t') \Leftrightarrow s_0 \approx_L s'_0 \wedge ext(\alpha)[L = ext(\alpha')][L$
- ⊕ With declassification actions [TCBC11]
 - ⊕ $t = (s_0, \alpha)$ and $t' = (s_0, \alpha')$ are declassification equivalent (\approx_{L_D}) if
 - 1 The subsequence of declassification actions at L is identical in t and t'
 - 2 For every declassification action $d = declassify(P, L)$, if $indexof(d, \alpha) = i$ and $indexof(d, \alpha') = i'$ then $\Pi(P, s_i) = \Pi(P, s'_{i'})$

Security Properties based on traces

- ⊕ Without declassification action [TCBC10]
 - ⊕ $ext(t)$: external (users) actions of t
 - ⊕ $ext(t) \sim_L ext(t') \Leftrightarrow s_0 \approx_L s'_0 \wedge ext(\alpha)[L = ext(\alpha')][L$
- ⊕ With declassification actions [TCBC11]
 - ⊕ $t = (s_0, \alpha)$ and $t' = (s_0, \alpha')$ are declassification equivalent (\approx_{L_D}) if
 - 1 The subsequence of declassification actions at L is identical in t and t'
 - 2 For every declassification action $d = declassify(P, L)$, if $indexof(d, \alpha) = i$ and $indexof(d, \alpha') = i'$ then $\Pi(P, s_i) = \Pi(P, s'_{i'})$

Confidentiality in Information Systems with Declassification

- 1 $ext(t_n) \sim_{L_D} ext(t'_{n'}) \implies \alpha_n[L_D = \alpha'_{n'}][L_D$
- 2 $t_n \sim_{L_D} t'_{n'} \implies s_n \approx_L s'_{n'}$.

Integrity in Information Systems with Declassification

$$ext(t_n) \sim_{\underline{L}} ext(t'_{n'}) \wedge t_n \sim_{L_D} t'_{n'} \implies t_n \approx_{D_L} t'_{n'}$$