Università degli Studi di
Trento

Matching Midlet's Security Claims with a Platform Security Policy using Automata Modulo Theory

Fabio Massacci, Ida Siahan

(University of Trento)

www.massacci.org

www.s3ms-project.org

NORDSEC-2007

Security of Software and
Services for Mobile Systems

Motivation

- Today's smart phones/nomadic devices have more computing and communication power than PCs 20 years ago, but …
- Not even remotely the amount of third party software available for PCs at that time, and
- A long term market growth cannot be based on selling ring-tones as the only "added-value" services.

Outline

- **Security x Contract**
  - Concepts
- **Automata Modulo Theory (AMT)**
  - AMT Theory
  - Contract/Policy Matching
- **Conclusions**
  - Issues yet to be addressed

Observations

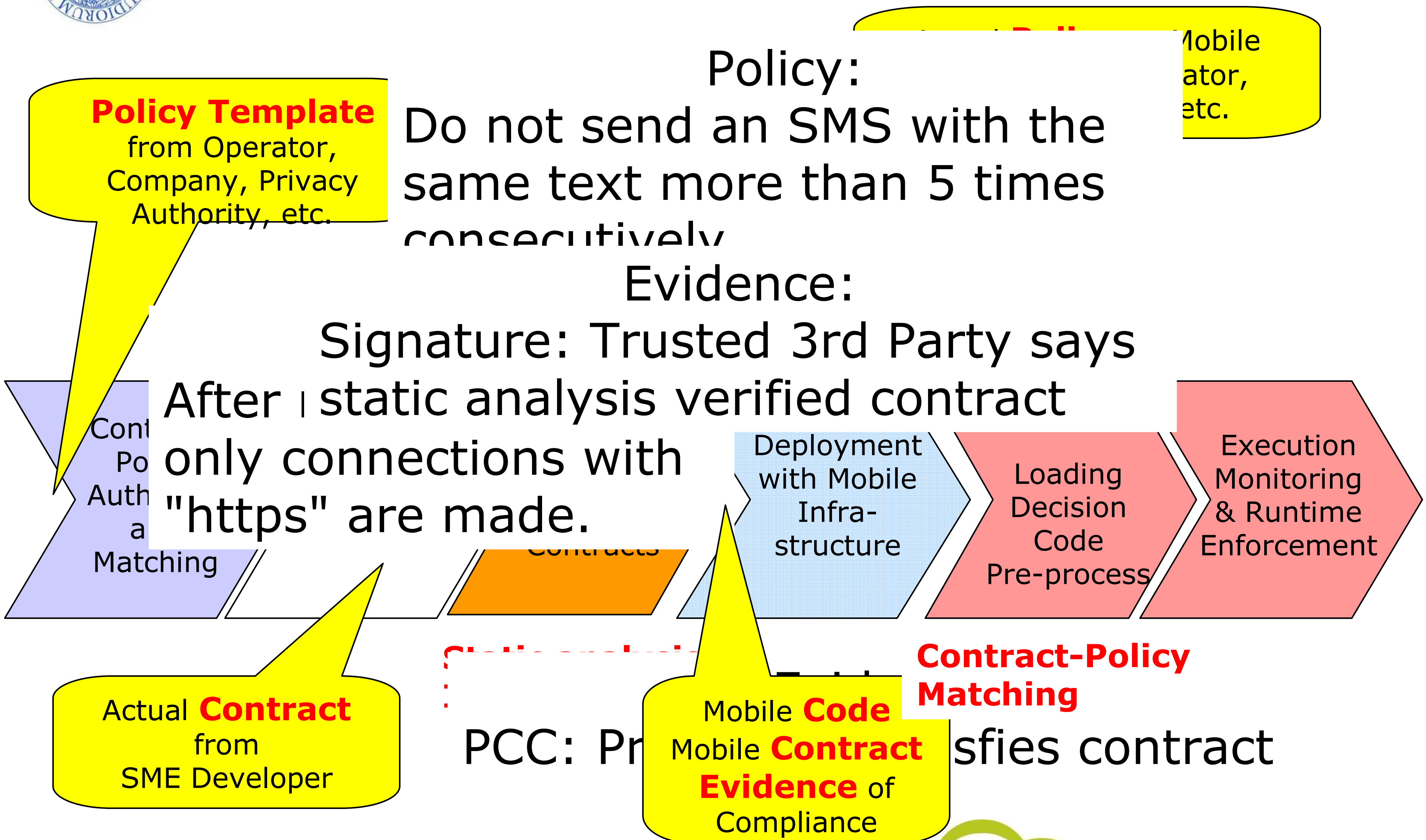
- **A validation infrastructure exists**
  - A signature is checked on the device;
  - No semantics is attached to it.
- **Some technologies exist**
  - Static analysis to prove program properties

  [Leroy et al, and many others]
  - Monitor generation for complex properties

  [Havelund & Rosu, Erlingsson & Schneider, Krukow et al. Ligatti et al.]
- **Security-by-Contract (SxC) puts them together**
  - Use contracts as semantics for the signatures;
  - Use static analysis and monitors as basis;

Key Concepts

- **Contract carried by application;**
  - Claimed Security behavior of application;
  - (Security) interactions with its host platform;
  - Maybe with Proof that code satisfies contract.
- **Policy specified by a platform.**
  - Desired Security behavior of application;
  - Fine-grained resource control
- **But I trust nobody, I just need policy monitor**
  - Monitoring ONLY a part of the story…



S3MS Security of Software and Services for Mobile Systems

Contract vs Policy

Claimed Behavior

Rules

- Used Methods
- Bounds on Methods Args
- Bouns on ret Values
- Allowed Sequences
- Achievable Obligations

Contract

Matching

Desired Behavior

Rules

- Possible Methods
- Constraints on Methods Args
- Constraints on ret Values
- History-based access control
- Desired Obligations

Policy

Language Containment of Automata Modulo Theory

What's Automata Modulo Theory ($\mathcal{AMT}$)?

- **Finite State Automata**
  - They represent the security behavior (claimed or desired)
  - You should know that…
- **With "Infinite" Edges**
  - Url starting with "https://" are not that few…
  - Battery Levels less than 30%
- **BUT Finitely represented with Expressions**
  - m=Java.IO.Connector &&
  - protocol(x)==https && protocol(x)==http
  - applicationType(x)!=jpg || appType(x)=appType(y)
- **Decidable theory for satisfiability of expressions**

Why Modulo Theory?

- **Matching = Language Containment**
  - Actions allowed by the contract subset actions allowed by the policy
- **Failure of Matching**
  - Path allowed by contract but NOT allowed by policy
  - Path allowed by contract and allowed by NEG policy
- **Path allowed by contract and by neg policy**
  - At run-time: two sequence of actions
  - Symbolically: two sequences of expressions
  - IF conjuction of pair of expressions SAT (modulo theory)
  - THEN exists common action…

Contract vs Policy Example

Claimed Behavior

Desired Behavior

"After PIM was opened no connections are allowed".

Connector.open() method is executed only if PIM.openPIMList() method was never called before.

Contract

Matching

"After PIM was accessed only secure connections can be opened".

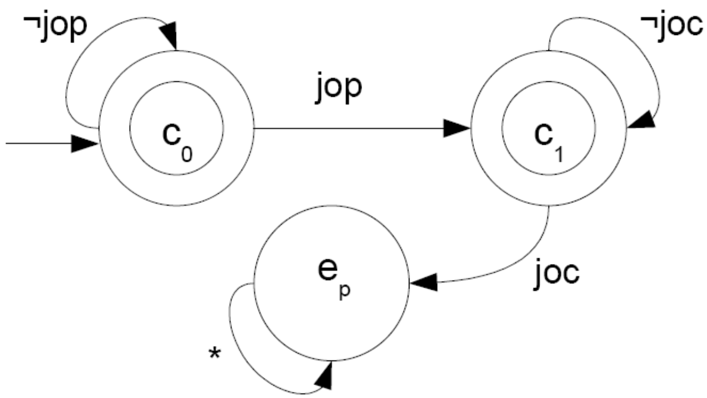
Connector.open(string url) method is executed only if the started connection is a secure one (url starts with "https://")

Policy

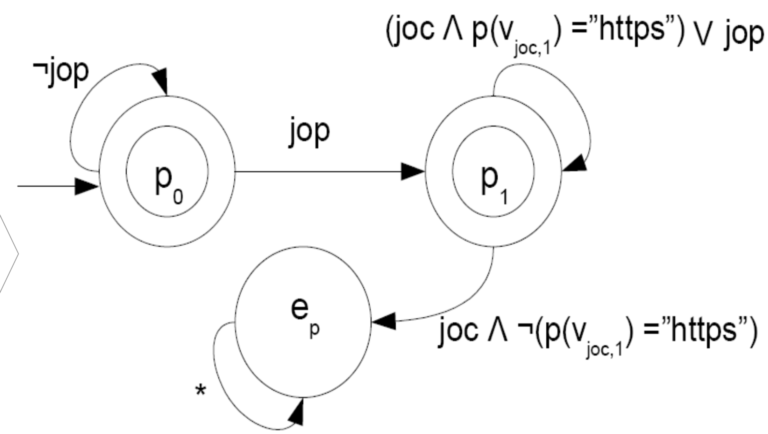
Language Containment of Automata Modulo Theory

Contract vs Policy Example in $\mathcal{AMT}$

Claimed Behavior **Desired Behavior**



Matching



Contract **Policy**

Language Containment of Automata Modulo Theory

$\mathcal{AMT}$ - Deterministic $\mathcal{AMT}$

- $\mathcal{AMT}$ is a tuple **<*E*, *S*, q_0, Δ, *F*>**
  - *E* **is a set of formulae in the language of the theory** $\mathcal{T}$
  - *S* **is a finite set of states**
  - q_0 **is the initial stat**
  - Δ **is labeled transition function**
  - *F* **is a set of accepting states**
- Deterministic $\mathcal{AMT}$:
  - **for every (q,e_1,q_1) and (q,e_2,q_2) in Δ and $q_1 <> q_2$ then in theory $\mathcal{T}$ the expression $e_1 \wedge e_2$ is unsatisable.**

Università degli Studi di Trento

$\mathcal{AMT}$ Run-Complementation-Intersection

- **Run:**
  - Finite (resp. infinite) word (trace) $w=<\alpha_0,\alpha_1,\alpha_2,\ldots>$ of assignments
  - Accepting finite run: $s_{|w|}$ goes through some accepting states
  - Accepting infinite run: the automaton goes through some accepting states infinitely often (as in BA)
- **Complementation:**
  - Given: a deterministic automaton A_T
  - The complement nondeterministic automaton A^c_T accepts language not accepted in A_T
- **Intersection:**
  - Given: a non deterministic automaton A_C and a nondeterministic automaton A^c_P
  - The intersection automaton A runs both given automata simultaneously on input word.

S3MS Security of Software and Services for Mobile Systems

Matching Language Inclusion Algorithm

- **Finding counterexamples faster:**
  - combine algorithm based on Nested DFS [S. Schwoon & J. Esparza] with decision procedure for SMT
- **Input:**
  - Midlet's claim and mobile platform's policy
- **Process:**
  - Start a depth first search procedure over the initial state
  - If an accepting state in $\mathcal{AMT}$ is reached:
    - Suspect state contains an error state of complemented policy: security policy violation without further ado.
    - Suspect state does not contain an error state:
      Start a new depth first searches to determine whether it is in a cycle
      If it is, then we report availability violation.

$\mathcal{AMT}$ main result

Let the theory $\mathcal{T}$ be decidable with an oracle for the SMT problem in the complexity class C then:

- The non-emptiness problem for $\mathcal{AMT}_{\mathcal{T}}$ is decidable in $LIN\text{-}TIME^{C}$.
- The non-emptiness problem for $\mathcal{AMT}_{\mathcal{T}}$ is $NLOG\text{-}SPACE^{C}$.

Conclusions

- **Security-by-Contract**
  - Ideas stolen from Design-by-Contract (Bertrand Meyer) and Model-Carrying-Code (Sekar et al.)
- **Security must takes into account complete lifecycle**
  - Enforcement but also Development & Matching
- **Matching Policy and Contract**
  - Mapped into $\mathcal{AMT}$
  - If theory for deciding edges polynomial (most cases) => Practical

Issues yet to be addressed

- Problem with security automata and infinity:
  - **Encoding of history dependent policies: allow certain strings that we have seen in the past.**
- Interesting problem for future work:
  - **Missing claimed security contract (current MIDP applications case)**
  - **Approximation automaton:**
    - **By static analysis based on the platform security policy**
    - **Code monitoring becomes unnecessary**
  - **Feasibility: depends on the cost of inferring approximation automata on-the-fly**

References

- N. Dragoni, F. Massacci, K. Naliuka, and I. Siahaan.
  Security-by-Contract: Toward a Semantics for Digital Signatures on Mobile Code. In Proc. of EuroPKI’07, 2007.
- N. Dragoni, F. Massacci, C. Schaefer, T. Walter, and E. Vetillard.
  A security-by-contracts architecture for pervasive services. In 3rd International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing. IEEE Computer Society Press, 2007.
- R. Sekar, V. Venkatakrishnan, S. Basu, S. Bhatkar, and D. DuVarney.
  Model-carrying code: a practical approach for safe execution of untrusted applications. In Proceedings of the 19th ACM symposium on Operating systems principles (SOSP-03), pages 15–28. ACM Press, 2003.
- S. Schwoon and J. Esparza. A note on on-the-y verication algorithms. Technical Report 2004/06,Universitat Stuttgart, Fakultat Informatik, Elektrotechnik und Informationstechnik, November 2004.