A Load Time Policy Checker for Open Multi-Application Smart Cards

Nicola Dragoni\textsuperscript{1} \quad Eduardo Lostal\textsuperscript{1} \quad \textbf{Olga Gadyatskaya}\textsuperscript{2} \quad Fabio Massacci\textsuperscript{2} \quad Federica Paci\textsuperscript{2}

\textsuperscript{1}Technical University of Denmark, \textsuperscript{2}University of Trento

POLICY-2011, June 6-8, 2011
Agenda

1. Motivations
2. The Java Card Technology
3. The Security-by-Contract Solution
4. The Policy Checker
5. Conclusions
Multi-application smart cards

Integration of multiple applications on one chip

Supported by current technologies

Applets can come from different providers

Cards can evolve on the field

Applications can interact

Exchange of loyalty points/miles/bonuses

Money transfers/payment operations

Information services

...
Integration of multiple applications on one chip

- Supported by current technologies
- Applets can come from different providers
- Cards can evolve on the field

But in reality they don't!
Multi-application smart cards

Integration of multiple applications on one chip

- Supported by current technologies
- Applets can come from different providers
- Cards can evolve on the field

Applications can interact

- Exchange of loyalty points/miles/bonuses
- Money transfers/payment operations
- Information services
- ...

But in reality they don’t!
The Goal

- The card has to verify that the security policy concerning application interactions is satisfied during the on the field evolution

Obstacles

- Security policy is provided by all the stakeholders together
- The verification has to be performed by the device itself
- Runtime monitoring is not possible
- The approach should be incremental
The Goal

- The card has to verify that the security policy concerning application interactions is satisfied during the on the field evolution

Obstacles

- Security policy is provided by all the stakeholders together
- The verification has to be performed by the device itself
- Runtime monitoring is not possible
- The approach should be incremental

Solution

- Each application brings its own policy
- The verification is performed while loading
- Security-by-Contract approach provides incremental verification
The Java Card Architecture

- **LOAD TIME**
  - CAP file
  - INSTALLER
  - JCRE

- **RUNTIME**
  - JAVA FIREWALL
  - APPLET
  - JAVA API
  - JVM (Interpreter)
  - NATIVE API
  - OPERATING SYSTEM
  - HARDWARE
Applet Interactions

- Applets are isolated by the Java Card firewall
- Only methods of Shareable interfaces (called services) are available through the firewall
The Security-by-Contract Architecture

CAP file

JCRE
INSTALLED
CLAIM CHECKER
POLICY CHECKER
JVM (Interpreter)
NATIVE API
OPERATING SYSTEM
HARDWARE
LOAD TIME
RUNTIME
JAVA FIREWALL
APPLET
JAVA API
APPLET

Olga Gadyatskaya (UNITN)
A Policy Checker for Smart Cards
POLICY-2011 7 / 15
The Security-by-Contract Process

CAP
Bytecode
Contract

Claim Checker
Contract matches Bytecode?

Policy Checker
Contract matches Policy?

Policy

Loading
Yes
No

Stop
Free the memory

Yes

Linking and Installation
The Contract of an applet $A$ consists of two parts:

- **Claim**: specifies provided and called services;
- **Application Policy**: specifies who is authorized to call $A$’s services and which services $A$ needs;

The Security Policy

- Security policy is a union of all the contracts of the applets on the card
The Formal Model

Contract formally

- The $Claim_A = \langle Provides_A, Calls_A \rangle$ where
  $Provides_A$ is a set of services that applet $A$ provides as a server.
  $Calls_A$ is a set of services of other applets that $A$ can try to invoke.

- The $AppPolicy_A = \langle Allows_A, Needs_A \rangle$ where
  $Allows_A$ contains service access rules as pairs $(s, B)$, where $s$ is a service of $A$ and $B$ is some applet.
  $Needs_A$ is a set of functionally necessary services.

Security Properties

- **Stable Security**
  If an applet $A$ invokes a service of an applet $B$, then $A$ is authorized to do it: if $B.s \in Calls_A$ then $(s, A) \in Allows_B$

- **Stable Functionality**
  All functional needs are satisfied: if $B.s \in Needs_A$ then $s \in Provides_B$
### An example

<table>
<thead>
<tr>
<th>Application</th>
<th>Claim</th>
<th>AppPolicy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Provides</td>
<td>Calls</td>
</tr>
<tr>
<td>EMV@BANK</td>
<td>transaction</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>fill_purse</td>
<td>-</td>
</tr>
<tr>
<td>ePurse@BANK</td>
<td>payment</td>
<td>fill_purse</td>
</tr>
<tr>
<td></td>
<td>account_balance</td>
<td>transaction</td>
</tr>
<tr>
<td>jTicket@Transport</td>
<td>buy_ticket</td>
<td>payment</td>
</tr>
<tr>
<td>Weather@Sky</td>
<td>weather_info</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>weather_RSS</td>
<td>-</td>
</tr>
<tr>
<td>eTicket@SASTravel</td>
<td>-</td>
<td>weather_info</td>
</tr>
</tbody>
</table>
The Policy Checker verifies that the *Contract* is compliant with the security policy of the card.

The Policy Checker maintains the security policy across updates.

As a proof-of-concept we implemented it as an applet.
## Performance

<table>
<thead>
<tr>
<th>Application</th>
<th>Claim</th>
<th>AppPolicy</th>
<th>Contract</th>
<th>PolAllows</th>
<th>PolNeeds</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMV@BANK</td>
<td>104</td>
<td>144</td>
<td>272</td>
<td>96</td>
<td>96</td>
<td></td>
</tr>
<tr>
<td>ePurse@BANK</td>
<td>112</td>
<td>144</td>
<td>280</td>
<td>96</td>
<td>112</td>
<td></td>
</tr>
<tr>
<td>jTicket@Transport</td>
<td>104</td>
<td>122</td>
<td>240</td>
<td>56</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>weather@Sky</td>
<td>104</td>
<td>144</td>
<td>272</td>
<td>96</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td>eTicket@SASTravel</td>
<td>96</td>
<td>112</td>
<td>232</td>
<td>56</td>
<td>56</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1352</strong></td>
<td><strong>860</strong></td>
<td><strong>2220</strong></td>
<td><strong>96</strong></td>
<td><strong>96</strong></td>
<td><strong>4</strong></td>
</tr>
</tbody>
</table>

![Graph showing memory consumption (bytes) vs. applications](image-url)
Conclusions and future/ongoing work

- The Security-by-Contract framework can ensure at loading time that the application interactions policies of each stakeholder will be preserved on the card;
- The Policy Checker component was implemented as an applet;
- Very difficult to implement something on a smart card!
Thank you!

Send your applets and comments  gadyatskaya@disi.unitn.it