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

Nicola Dragoni[1] Eduardo Lostal[1] **Olga Gadyatskaya**[2]
Fabio Massacci[2] Federica Paci[2]

[1]Technical University of Denmark, [2]University of Trento

POLICY-2011, June 6-8, 2011

Agenda

Multi-application smart cards

picture from http://fingerprint-security.net/

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

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 loyaly 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**

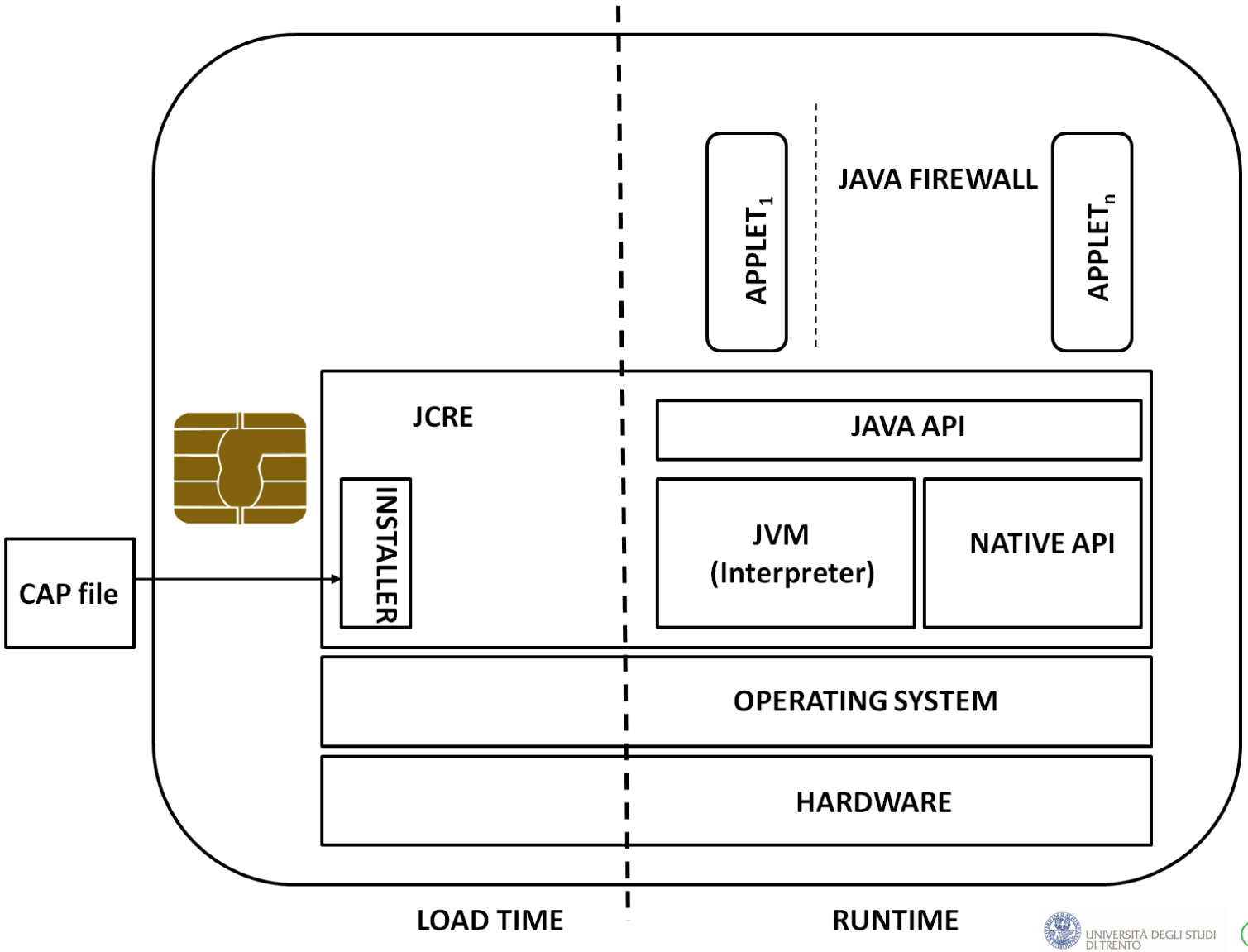
Obstacles

- 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

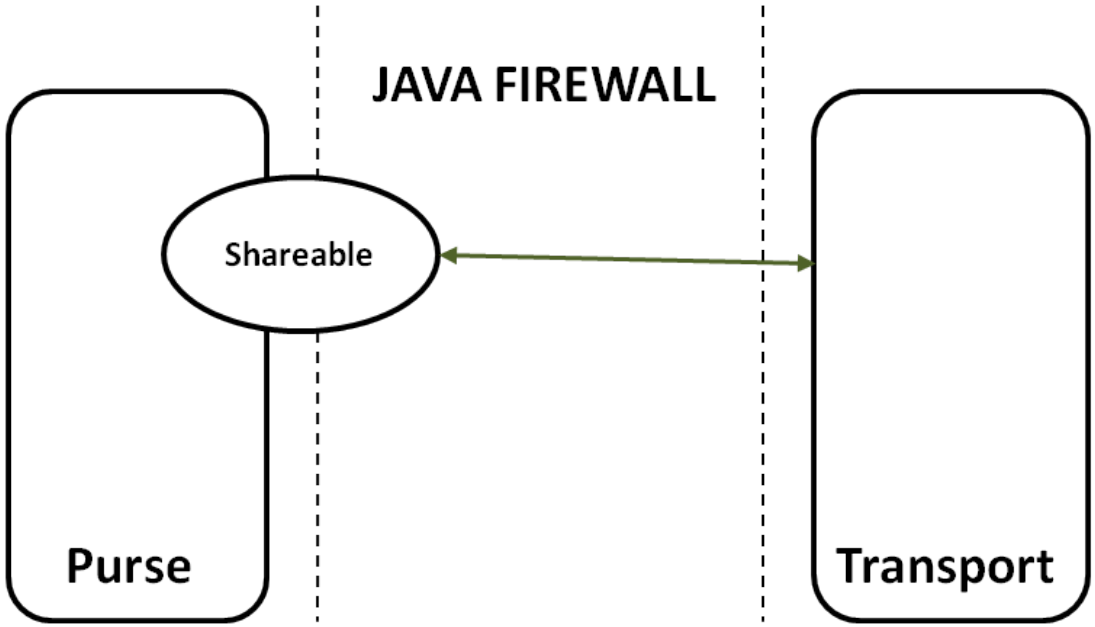
- Each application brings its own policy
- The verification is performed while loading
- Security-by-Contract approach provides incremental verification

The Java Card Architecture

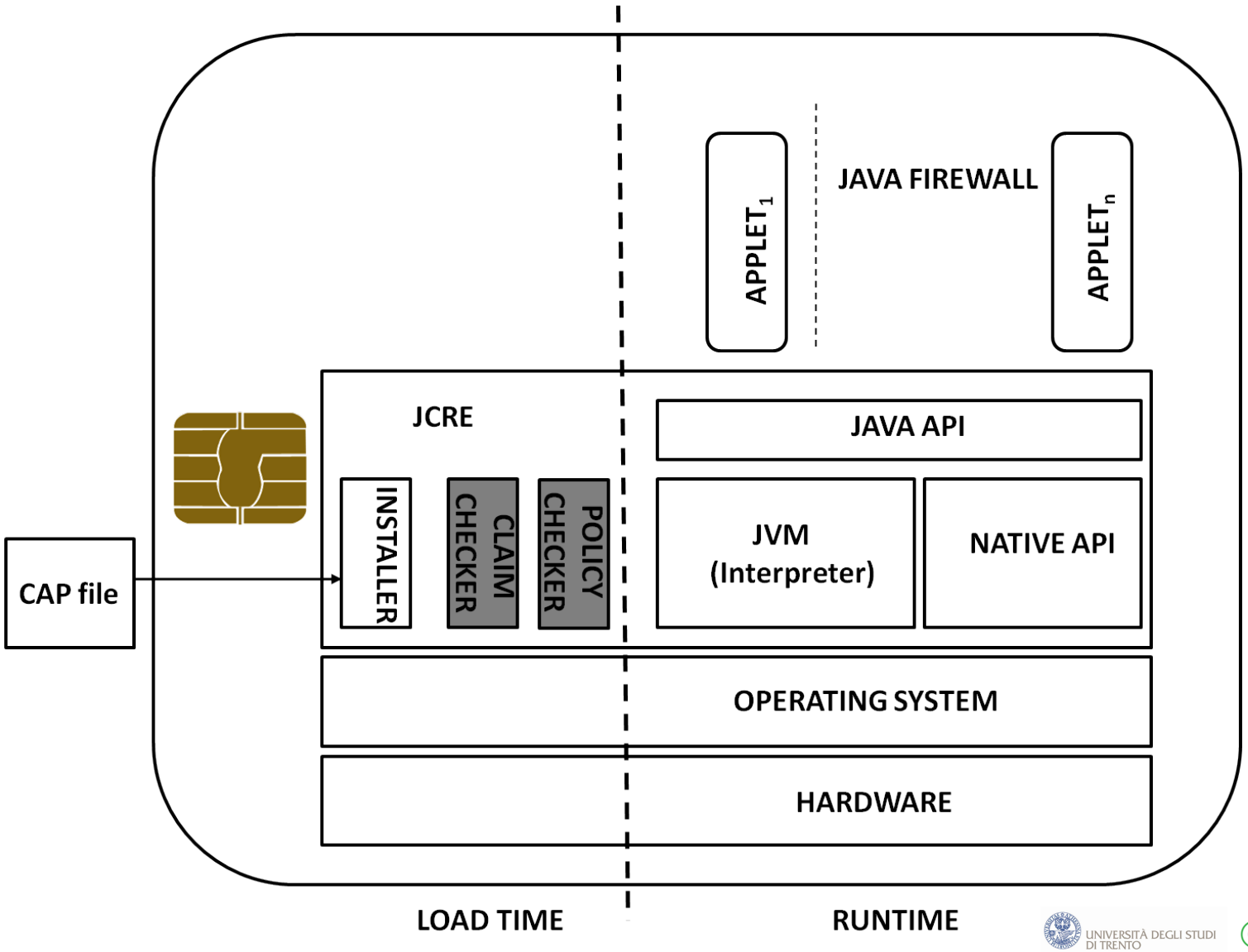


Applet Interactions

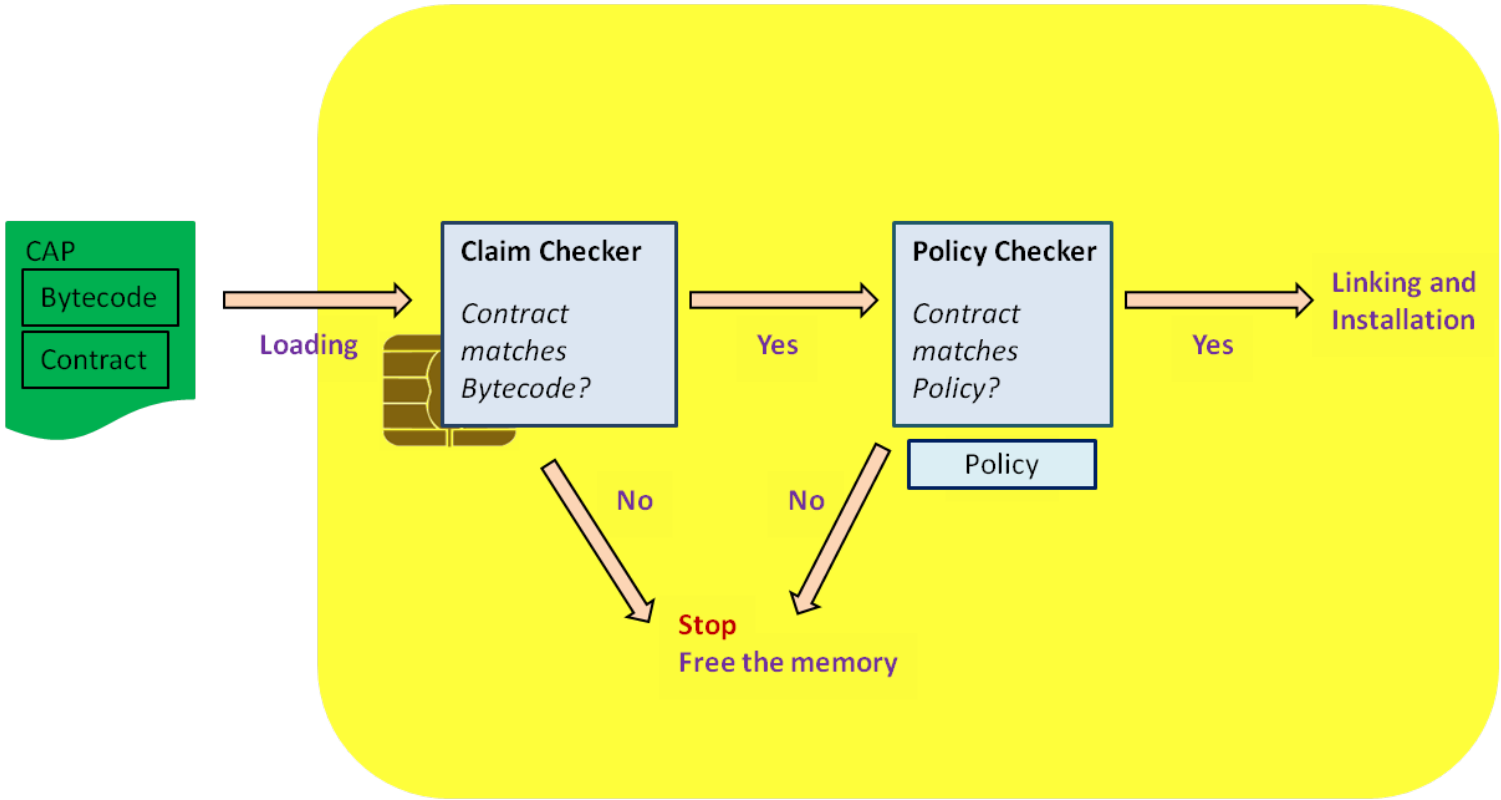
- Applets are isolated by the Java Card firewall
- Only methods of Shareable interfaces (called *services*) are available through the firewall



The Security-by-Contract Architecture



The Security-by-Contract Process



Contracts and security policy

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

Application	*Claim*		*AppPolicy*	
	Provides	*Calls*	*Needs*	*Allows*
EMV@BANK	transaction fill_purse	-	-	(transaction, ePurse@BANK) (fill_purse, ePurse@BANK)
ePurse@BANK	payment account_balance	fill_purse transaction	fill_purse	(payment, jTicket@Transport) (account_balance, jTicket@Transport)
jTicket@Transport	buy_ticket	payment	payment	-
Weather@Sky	weather_info weather_RSS	-	-	(weather_info, eTicket@SASTravel) (weather_RSS, eTicket@SASTravel)
eTicket@SASTravel	-	weather_info weather_RSS	weather_info	-

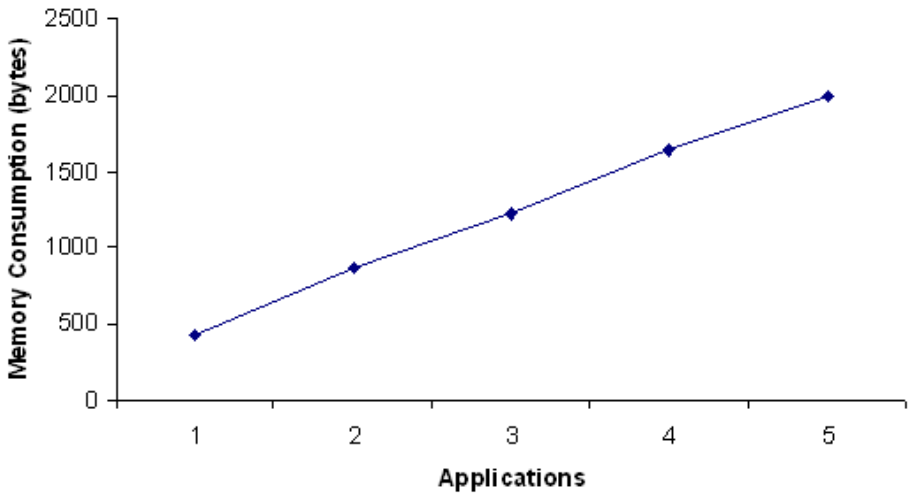
The Policy Checker

- 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

The Policy Checker Implementation

- Performance

	Claim	AppPolicy	Contract	PolAllows	PolNeeds	System
EMV@BANK	104	144	272	96	96	
ePurse@BANK	112	144	280	96	112	
jTicket@Transport	104	122	240	56	56	
weather@Sky	104	144	272	96	56	
eTicket@SASTravel	96	112	232	56	56	
Total	1352			860		2220



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