Reaching Ubiquity through Invisibility

Eric Vétillard
Chief Architect
Trusted Logic
www.trusted-logic.fr

Cassis’04 – Marseille – March 2004
Ubiquity

- Ubiquity, *n.*
  The state of being everywhere at once.
Invisibility

- Invisibility, *n.*
  The quality of not being perceivable by the eye.
Levels of Invisibility

- First degree
  - Visible
  - Invisible

- Second degree
  - Visible
  - Invisible

- Many other things are invisible here
  - TL Logos, Windows, …
  - The eye sees them, but not the brain
Ubiquity

- What is everywhere is invisible
  - Ubiquitous computing
  - Pervasive computing

- a.k.a. “Commodity”
Ubiquity of Smart Cards

- For the end user
  - Using a smart card

- For the IT architect
  - Accessing a smart card

- For the developer
  - Programming a smart card
Smart Cards?

The debate we won’t have

• Pros
  - Tamper-resistant hardware
  - Closely associated to a person

• Cons
  - Not better than a server in a connected world
  - Does not greatly simplify the infrastructure
Hypotheses

- Smart cards are useful
  - They can securely hold personal information
  - They are an interesting element in a security infrastructure

- Security is a concern
  - Fair hypothesis since 9/11
For the end-user
Anecdotes

- **1997, JavaOne keynote by Scott McNealy**
  - Positive: Smart cards are mentioned
  - Negative: Scott mentions “swiping” a smart card

- **2003, Java Card Forum meeting, Brighton, England**
  - Positive: There is a smart card terminal
  - Negative: My French banking card does not work
Breakthroughs

- Contactless transit cards
  - No need to get the card out of the wallet
  - Use is very natural

- SIM cards
  - We never think about them …
  - … except when reading « SIM is not ready »

- Both are invisible
Not so good

• Use of banking cards in France
  ➢ At the end of the counter
  ➢ Efficient and secure (PIN is protected)
  ➢ Inconsistent with current security rules
    ✓ The clerk should check the hologram
    ✓ No physical security is possible

• Debate about “Smart Tags” in the U.S.
  ➢ Only bad use cases for the end-user
  ➢ Strong privacy concerns
  ➢ Invisibility is not always good for the end-user
On the Right Track

- Appropriate technologies exist today
  - Portable/simple card readers
  - Contactless cards

- The technology is well accepted
  - Use standard patterns
  - Simple is beautiful

- It is simply a question of time
  - Technology needs to be deployed
  - It also needs to be well integrated
For the IT architect
The infamous APDU

- A bad, old-style, serial protocol …
  - Very slow half-duplex protocol
  - Only 256 bytes at a time
  - Not even deadlock proof

- … is the first thing you hear about
  - Most tutorials and APIs are a nightmare

- APDU’s are far too visible
APDU’s are not that bad

- This is a very slow protocol
  - Cards are not fast
- This is a low-level protocol
  - So is TCP
- The card must be the slave
  - But SIM Toolkit was built on top of it
High-level protocols exist

- The application model has evolved
  - First with Java Card itself
  - Java Card 2.2 includes RMI
  - Experiments have been made with Jini

- But no application uses them …
Specs are the problem

- Most of today’s applications are old-fashioned
  - Exchanging standard APDU’s
  - Based on a card file system (GSM/3G, EMV)

- New architectures don’t get to the specs
  - So card (and APDU’s) remain highly visible
  - Nobody wants to deal with them
Traditional applications?

- Backward compatibility issue
  - Terminals are strongly linked to cards
  - Terminals are late compared to cards
    - STIP remains confidential
    - MIDP is just starting its deployment
  - French EMV cards need to support B0’

- The migration will be extremely slow
  - J-CRMI is a difficult protocol for terminals
  - The change is too costly
New designs for new applications

• There are good news on cards …
  ➢ DoD makes extensive use of sharing
  ➢ They even have defined plug-in applets

• … and on terminals
  ➢ JSR177 gives a way to access cards
  ➢ They have selected JCRMI as one of the protocols
  ➢ They also make extensive use of PKCS#15 / WIM

• The card becomes invisible
The security issue

- Smart cards are used for security
  - To protect some assets
  - To provide guarantees (authentication, …)

- Security is difficult to assess
  - Smart card security is not simple
  - System security is much harder
Building a secure system

Nothing really exists in that field about smart cards
More invisibility

- Security is a key factor
  - Some data needs to be protected
  - Operations need to be performed on this data

- Splitting applications automatically sounds good
  - Even extremely good on mobile phones

- Application management is the next issue
  - How to increase the trust?
For the developer
The Java Card promise

- Millions of Java programmers
  - Smart card application architecture is specific
  - Secure programming remains difficult

- Thousands of new applications
  - Developers don’t make applications
  - Issuers make applications
We did it!

- Thousands of SIM Toolkit applications
  - Most likely at least one on your mobile
  - You don’t even know you have a Java Card

- The number of developers did not follow
  - Card manufacturers still dominate the market
  - Even they do not always master card programming
How to get there?

- Education
  - Teaching more about security
  - Smart cards are ideal test cases

- Rationalization
  - Definition of a process

- Certification
  - Making sure applications are right

- Automation
  - Generating the applications right
Education

- Computer security is not much taught
  - There are specific programs
  - Other programs are not very good

- The basics are missing
  - Security is not a reflex
  - Students are not aware of security issues …
  - … or they have bad principles
Security 101

- The basic principles
  - From assets and attacks to risks and countermeasures
  - Identity, authentication, authorization and other tools

- Application to computers
  - Protecting data and code
  - A plentiful of use cases

- Practical case: A smart card application
  - Ideal kind of application: simple and secure
  - Easily leads to an extension to a system
Defining a process

- This sounds quite simple
  - Defining some good principles
  - Making sure that they are applied

- Reality check …
A Development Cycle

Specification

Card Issuer

Applet

Acceptance Process

Development Process

Application Developer

Applet

Applet

ok

ok

ok

ok
Enhancing the Process

Java Card App.
Developer Handbook of Rules

Interop and Security Guides
Smart Card

Defined by the issuer
Applied by the providers
Enhancing the Process

Applied by the Issuer
Checks the application of guides
Writing the Guides

1. Gather information
   - Define the target environment
   - Perform a risk analysis

2. Split the responsibility
   - Some duties for the card provider
   - Some duties for the developer

3. Consolidate the guides
   - Define detailed rules for developers
Example: Sharing in SIM Toolkit

The Issue

Basic Requirement
No sharing is possible

Practical Constraint
The SIM Toolkit API uses sharing

Contradiction!
This can be addressed by defining more precise development guidelines
Example: Sharing in SIM Toolkit

The Basic Guidelines

Two guidelines:

- Applications can only share objects with the GSM application
- Applications can only use objects shared by the GSM application

Security policy is merged with the context constraints
Example: Sharing in SIM Toolkit

Share objects with the GSM application:
- Only applets can implement ToolkitInterface
- Only applets can be returned as shared objects
- Before sharing, check the AID of the GSM application

Use objects shared by the GSM application:
- Applet should not use getAppletShareableInterfaceObject
- Only the SIMView shareable interface can be used

Rules are directly applicable by developers
Getting to automation

1. Automate the validation phase
   - Performed by experts
   - Definitely simplifies their work

2. Provide tools to developers
   - Help them make better applications
   - Need to be more explicit

3. Generate code automatically
   - Let developers focus on security principles
   - Automate the implementation
Demo

- Application certification
- Use by developers
Code generation

- Simple at the level of an application
  - User authentication
  - Secure channel management
  - Life cycle management
  - Sensitive data protection

- More complex at the level of a system
  - Splitting responsibilities
  - Dealing with card life cycle
Secure Java Beans?

- The developer focuses on its main tasks
  - Writing application-specific code
  - Defining security requirements
- The platform does everything else
  - Generating the required security code
  - Providing the security-related libraries
- Security becomes invisible
Liste à la Prévert

- Smart cards = Security
- Developers have no security culture
- Issuers have no developer culture
- Researchers just solve the basic problems
- End-to-end solutions are uncommon
A few leads

• Lots of good things to come …

• About virtual machines
  ➢ Can you use our proof information?

• About proof or validation …
  ➢ Readability of results (and failures)

• About modeling
  ➢ Can the developer just do a few clicks?