Emerging Threats: A Perspective from Mobius (EU project on Mobility, Ubiquity, and Security)

Andrei Sabelfeld, Chalmers

http://mobius.inria.fr

Contract n° 015905
Fact sheet

- Integrated Project within FET Global Computing II, started Sept 2005, 4 years duration.
- 16 members

INRIA
RU Nijmegen
U. Edinburgh
Tallinn U.
UC. Dublin
UP. Madrid
SAP Research
Trusted Logic

LMU München
ETH Zürich
Chalmers
Imperial College
U. Warsaw
TU Darmstadt
France Telecom
TLS

- Scientific Advisory Board:
  Martin Abadi   Amy Felty   Rustan Leino

- End User Panel
The goal of MOBIUS is

- to build a *security infrastructure*
- for *mobile code*
- in the context of *global computing*
- by means of *certificate-based verifiable evidence* (PCC)
- which captures *expressive security policies*
Computational model

- Very large networks of (JVM-enabled) devices:
- No central trust authority: trust infrastructures must allow verifiable evidence (cryptography is not enough).
- Devices contain computational infrastructures that can be updated/extended remotely.
- Sandboxing is not enough. No sharp distinction between static Trusted Computing Base and mobile applications.
An Application Scenario

- MOBIUS promotes *trust through verifiable evidence* (PCC). Can be combined with *trust by authority* or *trust by reputation*.

- Phone operators/manufacturers can act as trusted intermediaries:

  ![Diagram showing the interaction between producers and consumers through PCC and PKI]

- Provides an appropriate trade-off between feasibility and flexibility which will be exploited in the rest of the project.
Certificate-based Mobile Code Safety

PRODUCER

CONSUMER

Program

Safety Policy

Program

Certificate

Safety Policy

OK

OK
Proof-Carrying Code (PCC)

- *Proof Carrying Code* (PCC) is a general technique for mobile code security which associates security information (*certificates*) to programs.
Proof-Carrying Code (PCC)

- *Proof Carrying Code* (PCC) is a general technique for mobile code security which associates security information (*certificates*) to programs.
  - *Producer*: Generates a certificate (or proof) at compile time by using a *certifier*. Then, submits it to the consumer.
  - *Consumer*: Receives (or downloads) the untrusted package “program + certificate”. Then, runs a *checker* to verify compliance with the security policy.

Key benefit: burden of ensuring compliance with desired security policy (mostly) shifted from consumer to producer.

Fundamental challenges:

1. defining expressive security policies covering a wide range of properties,
2. obtaining easy-to-use certificate generators and,
3. designing simple, reliable, and efficient checkers for the certificates.
Proof-Carrying Code (PCC)

- *Proof Carrying Code (PCC)* is a general technique for mobile code security which associates security information (*certificates*) to programs.
  - **Producer:** Generates a certificate (or proof) at compile time by using a *certifier*. Then, submits it to the consumer.
  - **Consumer:** Receives (or downloads) the untrusted package “program + certificate”. Then, runs a *checker* to verify compliance with the security policy.

- Key benefit: burden of ensuring compliance with desired security policy (mostly) shifted from consumer to producer.

- Fundamental challenges:
  1. defining *expressive security policies* covering a wide range of properties,
  2. obtaining *easy-to-use certificate generators* and,
  3. designing *simple, reliable, and efficient checkers* for the certificates.
Enabling technologies

- The *Enabling technologies*: should address the three issues mentioned.
- Enabling technologies should provide enough precision and automation to guarantee applicability and scalability.
  - Type systems/static analyses:
    - efficient and automatic, but
    - specialized and imprecise (due to approximation)
    - used for information flow, resource usage, aliasing
Enabling technologies

- The *Enabling technologies*: should address the three issues mentioned.
- Enabling technologies should provide enough precision and automation to guarantee applicability and scalability.
  - Type systems/static analyses:
    - efficient and automatic, but
    - specialized and imprecise (due to approximation)
    - used for information flow, resource usage, aliasing
  - Program logics:
    - general, precise, but
    - often interactive
    - used for high-level security policies and functional correctness
Main flavors of PCC

**Type-based PCC**

- Widely deployed in CLDC (and soon in J2SE)
- On-device checking is possible

**Logic-based PCC**

- Original scenario
- Applications to type safety and memory safety
Mobius vision

Source specification (types + logics)

Certificate generation

Interactive proofs

Code producer

Requirements

Source program

Certificate

Certificate

Code consumer

Bytecode program

Certificate

Code producer

Bytecode program

Certificate

Certificate checker

Runtime environment
MIDP: Assets

- MIDP: Mobile Information Device Profile (MIDP) of J2ME
  - 1/3 world’s phones support MIDP

- Operator assets
  - Billable events
  - Support infrastructure
  - Reputation
  - Network infrastructure

- End-user assets
  - Billed events
  - Private information
    - Personal Information Manager (PIM)
    - passwords
    - geo-location
    - application-specific data (e.g., secure storage)
  - Mobile phone
  - Information with no back-up
MIDP: Attacker’s goals

- Make money
- Steal sensitive information
- Hurt operator/user
- Perform a hacker stunt
- Perform a terrorist act
MIDP: Attacks

- Information flow
  - Disclosure of sensitive data
    - Sources: PIM, passwords, geo-location, application-specific
    - Sinks: persistent store, network access, covert channels
  - Modification of sensitive data

- Resource control
  - Abuse of billable events
    - Short messages
    - Phone calls
    - Network accesses
  - Memory/CPU usage
    - Synchronization
    - Exceptions
    - Network connections