Mobius is a European Integrated Project developing novel technologies for trustworthy global computing, using proof-carrying code to give users independent guarantees of the safety and security of Java applications for their mobile phones and PDAs.

Global computing means that applications today may run anywhere, with data and code moving freely between servers, PCs and other devices; this kind of mobility over the ubiquitous internet magnifies the challenge of making sure that such software runs safely and reliably.

In this context, the Mobius project focuses on securing applications downloaded to the Java MIDP platform: globally deployed across a host of phones, this is the common runtime environment for a myriad mobile applications.

Techniques of static analysis make it possible to check program behaviour by analysing source code before it ever executes. But mobile code means that this assurance must somehow travel with the application to reach the user. Convincing digital signatures using cryptography to identify who supplied a program; the breakthrough of proof-carrying code is to give mathematical proofs that guarantee the security of the code itself. We can strengthen digital signatures with digital evidence.

Key features of the Mobius security architecture are:• Innovative trust management, with digital evidence of program behaviour that can be independently checked by users or any third party. • Static enforcement, checking code before it runs, adaptable to manage a range of user security concerns, and configurable to match the real-world mix of mobile platforms. • Modularity, allowing developers to build up trusted applications from trusted components.

Mobius is a consortium of leading academic and industrial research partners, at 16 sites across ten countries. Members bring international expertise in software security, Java, mobile telecoms and smart devices. All these combine with the aim of delivering a platform for innovative trust management in the next generation of mobile applications.

Proof Carrying Code

Proof Carrying Code (PCC, developed by Necula, Lee and others in the 1990s) is a general technique for mobile code security which associates security information (certificates) to programs.

- **Producer** generates certificate (or proof) of compliance with security policy by using a *certifier* at compile time.
- **Consumer** receives the untrusted package ("program + certificate") and runs a *checker* to verify compliance with security policy.
- PCC has many uses in systems whose trusted computing base is dynamic, either because of mobile code or because of regular updates. Examples include extensible operating systems, Internet browers capable of downloading code, active network nodes and safety-critical embedded controllers. The proof carrying code paradigm can be combined with trust by reputation.
- **Producers** submit programs and certificates to a trusted intermediary.
- **Trusted intermediary** - any trusted third party chosen by consumers, e.g. the phone operator - checks certificates and signs checked programs.
- **Consumers** receive programs from trusted intermediary and check signatures (not certificates).

Challenges

PCC shifts some of the burden of ensuring compliance with a desired security policy from the code consumer to the producer. To enable the use of PCC technology widely, Mobius will:

- define expressive security policies covering a wide range of properties,
- develop easy-to-use certificate generators, and
- design reliable and efficient checkers for certificates.

Mobius Platform

Mobius targets Java-enabled embedded execution frameworks that can run third party applications which must be checked against a platform security policy. Focus on devices (e.g. mobile phones) that support the Mobile Information Device Profile (MIDP version 2) and the Connected Limited Device Configuration (CLDC) of the Java 2 Micro Edition.

Reasons for choosing CLDC/MIDP:

- Uniform access to many resources specific to mobile devices (e.g. persistent store, players, camera, geolocation).
- Available on many devices (> 1 billion).
- Interesting subset of the Java language.
- Thousands of existing applications.
- Standardised through the Java Community Process.

Mobius targets security: confidentiality of data and code, information flow, memory space usage, non-interference and non-disclosure.

Three flavours of PCC

- **Logic-based PCC**
  - Specifications are expressed in a general-purpose program logic, such as Hoare logic or the Java Modeling Language (JML). Certificates are proofs in this program logic, and checking a certificate means checking that a given proof is correct.
  - Flexible due to highly expressive program logic.
  - Proofs in the program logic as a lingua franca for certificates.
  - Certificate generation (= proof search in the program logic) requires human interaction.

- **Type-based PCC**
  - Specifications (e.g. information flow properties, resource boundedness, aliasing relationships) are expressed as typeability in special purpose type systems. Certificates are type derivations, and certificate checking amounts to checking the well-formedness of a given type derivation.
  - Automatic certificate generation by type inference.
  - Type derivations can be standardized to proofs in a program logic.
  - Analysis may be too conservative.
  - Each type system is customized to a particular property.

- **Abstract interpretation PCC**
  - Specifications are expressed as systems of equations in complete lattices. Certificates are representations of fixpoint solutions. Certificate checking amounts to checking that a given representation is a fixpoint.
  - Automatic certificate generation by abstract interpretation.
  - Different properties require customized abstractions.