Overview of the Multos construction process

Chad R. Meiners

Outline

- Examine the characteristics of Multos
- Examine the project requirements
- Examine the formal methodology
- Examine the developmental results
- Evaluate Praxis’s methods
- Evaluate the paper’s focus
- Evaluate infrastructural requirements of the methodology

Multos

- Multos is a secure operating system for running separate applications within a single smart card.
- Multos depends on a centralized certification authority to provide permission to load and delete applications from a card.
- Praxis was hired to develop the Multos certification authority.
Project Requirements

• ITSEC level E6 certification
  – Formal methods required
  – System requirements must be unambiguous
• Minimum developmental risk
  – Avoid pitfalls in later development cycles
  – Keep within developmental and budget constraints

System Requirements

• COTS hardware and infrastructure
  – Minimize the cost
  – Minimize time
• Multi-user requirements
  – System distributed for high throughput
  – System must be usable to clients
• Security
  – System must be trusted and usable

Methodology

“Correctness by construction depends on knowing what the system needs to do and being sure that it does it.”
- Anthony Hall and Roderick Chapman

• Requirements
• Specification
• Design
• Code
Requirements

- Translated business objectives into requirements
- Labeled each requirement
- Traced each label to a source
  - Includes tracing security requirements to threats
- Validate all requirements with the client

Specification

- System specified as a series of black boxes
  - User Interface
    - The look and feel of the black boxes
    - Prototype verified by clients
  - Formal top-level specification
    - The functionality of the black boxes
    - Typechecker used to verify and check system consistency

High Level Design

- Described how the black boxes function together
- Described the black boxes’ internal structures
- Addressed security concerns between the boxes
Security with Untrusted Components

- System security must not really on the security of untrusted components
  - Commercial database
  - Untrusted operating system
- Security achieved by (from the paper)
  - Hardware separation
  - Information encryption
  - Authentication codes
  - Individual processing of security-critical code

Detailed Design

- Describes the actual software modules to construct the system
- Most modules are directly derived from the formal top-level specification
- Complex areas are further modeled in Z
  - Key management
  - System startup

Formal Security Policy Model

- Security requirements written in Z
- Four types of requirements
  - Security invariants
  - Functional requirements
  - Operational constraints
  - Information separation
- Requirements checked with a typechecker
Code

- Code life of twenty years
  - Avoided using COTS software components
  - Implemented these components from scratch
- Multiple languages used for appropriate jobs
  - Spark Ada: Security Kernel
  - Ada 95: System critical parts
  - C: Reuse cryptography code
  - C++: GUI

Review

- All artifacts of the development cycle were reviewed and check for correctness and consistency
- Automated checking was used when possible
- Desk checking was performed when necessary

Testing

- System built incrementally in a top down fashion
- Each system test was a real system in the environment
- Tests derived from specifications and automated
- Tests were instrumented to check statement and branch coverage
Results

- The delivered system satisfied client requirements
- Low defect rate in the system
  - 0.04 defects per KLOC
  - 100,000 line of code
- Minimized the amount of effort spent fixing errors (6% of total effort)

Results of Tool Support

- Developers could concentrate on insuring correct system wide behavior
  - Z automatically checked for consistency
  - CSP model checked
  - SPARK eliminated an entire domain of errors
- Tool support was utilized whenever available

But was it a commercial success?

- Mobile Communication
- Mobile e-Commerce
- Mass Transit
- CA in UK
- CA in Japan
- Future CA in Hong Kong
  - Government Immigration card project
Other Praxis Projects

- SHOLIS
- Hercules II (C-130J) Flight Software Development

Praxis’s Methodology

- Specify early; specify often
- Black boxes
  - Allows for the formalization of interfaces and expectations
  - Works even for untrusted components
- Untrusted COTS components integrated via incorporating the lack of trust in the system design

Paper’s focus

- Concentrated on the developmental process of incorporating various formal tools
- Advocated the “right tool for the right job” approach
- Designed to demonstrate the approachability of formal methods
Developmental Infrastructure

- Requires the developer's commitment to the approach
  - Personnel need to be trained in formal method techniques before starting the project
  - Formal methods must be considered at every step of the project
  - Must understand that FM is not a magic bullet

- Requires the client's acceptance of the approach
  - Client must be willing to provide proper feedback to the developers
  - Must understand that FM is not a magic bullet

So where do the formal methods for security fit into this process?

- Used to model protocols before implementation
- Used to establish a set of trusted protocol for use in formally developed systems
- The methods we have learned so far must be grafted into a real development process

References

- The Multos Website http://www.multos.com
- Praxis’s SPARK Website http://www.sparkada.com
Thanks

- Zhenxiao Yang: For questioning about the commercial success of Multos
- Brian Kellogg: For questioning about the existence of any bias in the paper