

Using Mandatory Secrecy and Integrity for Business to Business Applications on Mobile Devices¹

Paul A. Karger, Vernon R. Austel, and David C. Toll
IBM Research Division, Thomas J. Watson Research Center
P. O. Box 704, Yorktown Heights, NY 10598

IBM^{®2} has developed a new mandatory security model, combining both secrecy and commercial data integrity requirements [9]. This model was developed as part of an effort to design a high assurance operating system [5] for the Philips SmartXA chip – an operating system that could be evaluated at the highest security levels of the ITSEC [4] or the Common Criteria [1-3].

This paper shows how using the new security model can permit applications developers to solve security-related problems that could never be addressed before, either in smart cards or in larger computer systems. In particular, the security model and the operating system are designed to permit in-the-field downloading of applications written either in native languages (such as C or assembler) or in interpreted languages (such as Java Card^{™3}). These downloaded applications could be mutually hostile, yet the operating system will prevent unauthorized interference between the applications, yet still allow controlled sharing of information between selected applications, subject to the constraints of the new security model.

For full details of how the model was developed, see [9], [10], and [12].

Integrity Lattices

Secrecy lattices, such as the Bell and LaPadula model [6] while useful for protecting against unauthorized information disclosure, do not deal with unauthorized tampering or sabotage of information. The early military models focused only on secrecy, and even Girard's proposal [8] for mandatory access control on smart cards is only a secrecy model. A commercial system, however, cannot be limited to only protecting the secrecy of information. Assuring that information is not tampered with is often much more important in a commercial setting. Whether a smart card is used as a cash card or as a loyalty card, ensuring that the correct amount of money or loyalty points are transferred may be much more important than keeping secret how much money or how many loyalty points were transferred.

Biba [7] developed a model of mandatory integrity that is a mathematical dual of the Bell and LaPadula mandatory-security model. The principal difficulty with the Biba integrity model is that it does not model any practical system. Unlike the security models that developed from existing military security systems, the Biba integrity model developed from a mathematical analysis of the security models. However, Biba did not suggest how to actually decide which programs deserved a high integrity access class and which did not. This has made practical application of the Biba model very difficult.

Combined Secrecy and Integrity Lattice

How do we actually decide which programs are worthy of a higher integrity level? Since smart card issuers will be particularly worried about the security of applications on their cards (since they might be held liable in a court), we need to improve on the Biba model.

The Biba model also prevents high integrity applications from reading low-integrity data, in fear that the application might be compromised in some form. This makes it difficult to describe applications that have been designed with high integrity to specifically process low integrity data input and to rule on its appropriateness. This processing of low integrity data is called *sanitization*. How do we modify the model to support sanitization (both for integrity and secrecy)?

¹ © Copyright IBM Corporation 2000

² IBM is a trademark of the International Business Machines Corporation in the United States, other countries, or both.

³ Java Card is a trademark of Sun Microsystems, Inc. in the United States, other countries, or both.

Our new model solves the problem of assigning integrity levels by using third-party evaluation. Just as for the ActiveX and Java policies, developers digitally sign their applications. However, we go beyond this. If an application has been independently reviewed and digitally signed by the certifying body, then we can grant it a higher level of integrity. For example, we could define integrity levels for ITSEC [4] or Common Criteria [1-3] evaluated applications. The Commercially Licensed Evaluation Facility (CLEF) would evaluate the application and the certifying body would digitally sign the application and its ITSEC E-level.

To solve the sanitization requirements, we will mark a process with TWO separate integrity access classes – one for read permission and one for write and execute permission. The write/execute integrity access class will be the level determined by the independent evaluator. If the read integrity access class equals the write/execute class, then we have the standard Biba model. If the read integrity access class is lower than the write/execute integrity access class, then we have a process permitted to sanitize and upgrade input that is initially marked at a low integrity access class. Not just any program will be trusted for sanitization, but rather only programs explicitly evaluated for the purpose. Identifying the range of levels across which a program is allowed to sanitize will be specified as part of the digitally signed information from the CLEF. The operating system kernel will read that information when starting a program into execution to know what range of integrity classes to assign the process. Separating the execute permission from the read permission originated in the program integrity model of Shirley and Schell [13]. The policy was further developed in the GEMSOS security model [11] that specified a range of levels within which integrity downgrading could occur.

We do essentially the same for secrecy sanitization or downgrading. We define a pair of secrecy access classes, rather than integrity levels, and the read and write rules are reversed. There would be two secrecy access classes – one for read/execute and one for write. Note that for secrecy, we keep execute tied to read permission, because a process at a low secrecy level should not be permitted to execute high secrecy program code. This is not for anti-piracy purposes, but rather to maintain the secrecy of algorithms or constants in the code that must not be revealed, even by mere use of the program. Software piracy protection is outside the scope of this security model.

The combined access rules are shown in Figure 1. Note that we separate execute permission into two subclasses – normal transfers and a special CHAIN operation. A normal transfer is the execution of a branch instruction or a subroutine call instruction. CHAIN is a way to start a separate process executing at some other integrity and secrecy access class. Due to the limited memory of a smart card, the process executing the CHAIN operation is immediately terminated. The intended use of CHAIN is to start a guard or sanitization process or for a guard process to start a recipient of sanitized information.

Read permission	<p>Secrecy read/execute access class (process) \geq secrecy access class (object)</p> <p>Integrity read access class (process) \leq integrity access class (object)</p>
Write permission	<p>Secrecy write access class (process) \leq secrecy access class (object)</p> <p>Integrity write/execute access class (process) \geq integrity access class (object)</p>
Execute permission	
Transfer	<p>Secrecy read/execute access class (process) \geq secrecy access class (object)</p> <p>Integrity write/execute access class (process) \leq integrity access class (object)</p> <p>The target program of a transfer runs at the integrity level of the caller. A high integrity program cannot call or transfer to lower integrity code.</p>
Chain	<p>Secrecy read/execute access class (process) \geq secrecy access class (object)</p> <p>Secrecy write access class (process) \leq runtime read/execute secrecy class (new process)</p> <p>Integrity write/execute access class (process) \geq integrity read access class (new process)</p> <p>The first rule ensures that chain is possible only to files to which the caller has secrecy read permission. Integrity read permission is NOT required, because a high integrity process is always allowed to start a low integrity process. Contrast this with the rule that a high integrity program is not allowed to transfer directly to a low integrity program in the same process. The second rule ensures that the target process must have secrecy read permission to any passed arguments. The third rule ensures that the target process is not contaminated by a low integrity argument. The target program runs in a new process at the integrity access class specified in the digitally signed certificate of the program. This is a very important distinction. The process that issues the CHAIN operation does not determine the access class at which the target process executes. Rather, it is determined by the third party evaluator and certifying body who have digitally signed the code file.</p>

Figure 1. Access Control Rules

References

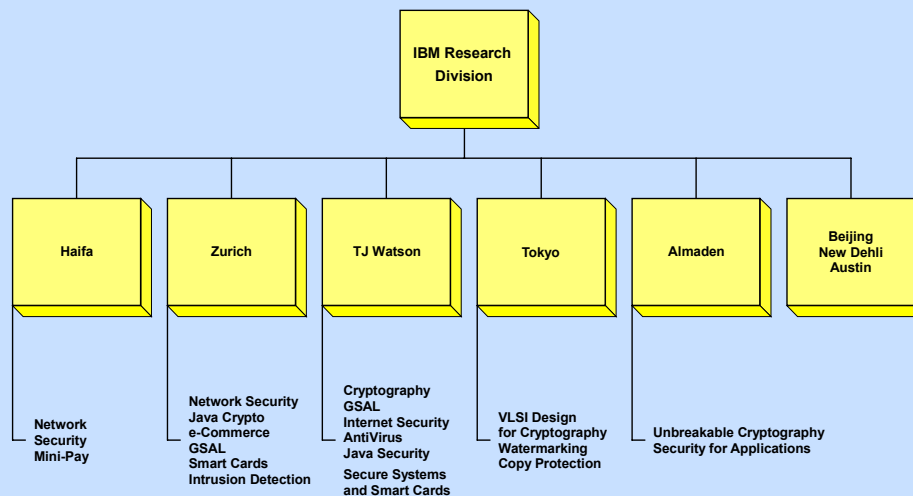
1. *Information technology - Security techniques -- Evaluation criteria for IT security -- Part 1: Introduction and general model*, ISO/IEC 15408-1, 1999, International Standards Organization.
2. *Information technology - Security techniques -- Evaluation criteria for IT security -- Part 2: Security functional requirements*, ISO/IEC 15408-2, 1999, International Standards Organization.
3. *Information technology - Security techniques -- Evaluation criteria for IT security -- Part 3: Security assurance requirements*, ISO/IEC 15408-3, 1999, International Standards Organization.
4. *Information Technology Security Evaluation Criteria (ITSEC)*, June 1991, Commission of the European Communities: Brussels, Belgium.
5. *Philips Semiconductors and IBM Research to co-develop secure smart cards: Highly secure operating system and processor, suitable for multiple applications*, 4 February 1999. URL: http://www.semiconductors.philips.com/news/content/file_384.html
6. Bell, D.E. and L.J. LaPadula, *Computer Security Model: Unified Exposition and Multics Interpretation*, ESD-TR-75-306, June 1975, The MITRE Corporation, Bedford, MA: HQ Electronic Systems Division, Hanscom AFB, MA.

7. Biba, K.J., *Integrity Considerations for Secure Computer Systems*, ESD-TR-76-732, April 1977, The MITRE Corporation, Bedford, MA: HQ Electronic Systems Division, Hanscom AFB, MA.
8. Girard, P. *Which Security Policy for Multiapplication Smart Cards?* in **Proceedings of the USENIX Workshop on Smartcard Technology**. 10-11 May 1999. Chicago, IL The USENIX Association. p. 21-28.
9. Karger, P.A., V.R. Austel, and D.C. Toll, *A New Mandatory Security Policy Combining Secrecy and Integrity*, RC 21717 (97406), 15 March 2000, IBM Research Division, Thomas J. Watson Research Center: Yorktown Heights, NY. URL: <http://domino.watson.ibm.com/library/CyberDig.nsf/home>
10. Karger, P.A., V.R. Austel, and D.C. Toll. *Using a Mandatory Secrecy and Integrity Policy on Smart Cards and Mobile Devices*. in **EUROSMART Security Conference**. 13-15 June 2000. Marseilles, France . p. 134-148.
11. Schell, R.R., T.F. Tao, and M. Heckman. *Designing the GEMSOS Security Kernel for Security and Performance*. in **Proceedings of the 8th National Computer Security Conference**. 30 September - 3 October 1985. Gaithersburg, MD DoD Computer Security Center and National Bureau of Standards. p. 108-119.
12. Schellhorn, G., W. Reif, A. Schairer, P. Karger, V. Austel, and D. Toll. *Verification of a Formal Security Model for Multiapplicative Smart Cards*. in **6th European Symposium on Research in Computer Security (ESORICS 2000)**. 4-6 October 2000. Toulouse, France:Lecture Notes in Computer Science Vol. 1895. Springer-Verlag. p. 17-36.
13. Shirley, L.J. and R.R. Schell. *Mechanism Sufficiency Validation by Assignment*. in **Proceedings of the 1981 Symposium on Security and Privacy**. 27-29 April 1981. Oakland, CA IEEE Computer Society. p. 26-32.

Using Mandatory Secrecy and Integrity for Business to Business Applications on Mobile Devices

Paul A. Karger, Vernon R. Austel, and David C. Toll
Secure Systems and Smart Cards
IBM Thomas J. Watson Research Center

Security Research in IBM



Secure Systems and Smart Cards

Our focus: Software and tamper-protected devices that enable electronic commerce.

- **Secure coprocessors**
Physically secure devices that do fast crypto and general purpose computation.
 - IBM 4758 - First ever device to pass FIPS 140-1 Level 4
- **Secure smart card operating system**
Provable security, despite mutually hostile applications
- **Power Analysis**
attacks. . . and defenses
- **Applications which require these devices**

Current Smart Cards

- Operating system put in mask ROM at chip foundry
- 8-bit processors, 512 bytes RAM, 16K EEPROM
- Limited software updates in the field
- Applications written by a handful of specialists
- Applications written in hand-tuned assembler language or C
- All applications on multi-function cards are written by or reviewed by one company.

Emerging Smart Cards

- Interpreted code (JavaCard, Multos, Smart Card for Windows)
- stronger crypto assist hardware (e.g. 1024-bit RSA)
- more memory, 16- and 32-bit processors
- standardized interfaces to card readers

Goals of IBM Research's Secure Embedded Operating System Project

- code written by companies who don't trust each other's programs (or programmers)
- programming by anyone
- field loadable machine code for best performance
- a provably secure operating system (very high level ITSEC or Common Criteria evaluation)
- JavaCard and native OS interface to applications inside

Project goals (financial application example)

- code written by companies who work independently of one another or don't trust each other's software to be bug-free

A bank issues a corporate travel smart card to employees in a large company. The card is used as a credit card, a corporate discount card, an identity authentication card, and to accrue loyalty points.

The large company has contracts with two different car rental companies, one used primarily in Europe, the other in the US.

The large company has contracts with twenty worldwide hotel chains.

Each car rental company, and each hotel chain writes its own card application. Each company uses its own programming team.

Project goals (financial application example)

- field loadable code
When a new card is issued to an employee, it contains only a credit application. When the employee travels, he/she inserts the card into a kiosk and downloads one or more of the car rental and hotel applications. The applications can be removed later to make room for a different set of car rental companies and hotels.
- programming by anyone for small trials
A company wants to set up a small Internet trial for 200 of its top customers. The smart card runs custom code that logs transactions which match a behavior profile.
- a certified operating system
An independent third party assures that the card meets its security claims.
- open interfaces (JavaCard and native OS) and applications inside
Applications can run on cards issued by multiple companies. An application vendor can add high performance code to support the new US Advanced Encryption Standard algorithm two years after a card is issued.

Philips Semiconductors' SmartXA Microcontroller

- supervisor/user mode
- memory protection (hardware firewall)
 - segmented architecture
- 16-bit CISC processor
 - derived from Philips XA (extension of 8051 and 30x faster)
 - better security than Philips XA
- lots of memory (48K ROM, 32K EEPROM, 2.5K RAM)
- 1024-bit modular math hardware assist (for RSA, DSA, EC)
- hardware random number generator
- UART

Philips Semiconductors' SmartXA Microcontroller

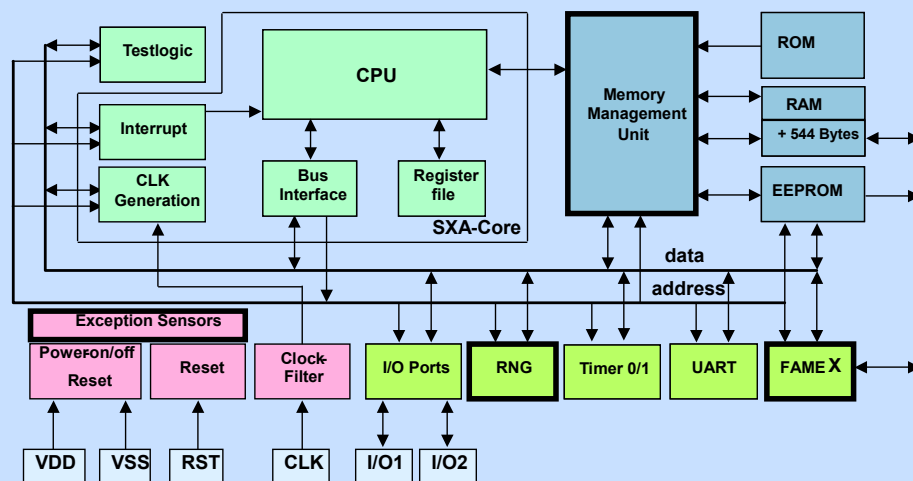


diagram courtesy of Philips Semiconductors

Software Layers on the SmartXA

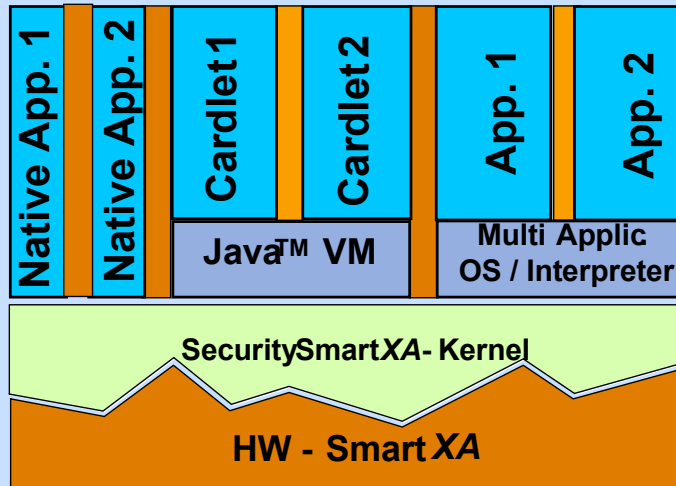


diagram courtesy of Philips Semiconductors

New Security Policy

- Need for a strong security policy to provided controlled sharing for business to business applications on a smart card
- Want to download code from multiple developers who may be mutually hostile
- formal mathematical model of the policy under development by University of Ulm and DFKI in Germany
 - **Ulm team recently moved to University of Augsburg**

Bell and LaPadula Access Rules

- For read permission
 - level (object) [level (subject)
 - category set (object) ` category set (subject)
- For write permission
 - level (object) m level (subject)
 - category set (object) r category set (subject)
- Level and category set can be combined into a single entity, called the access class

Integrity Models

- protecting against tampering and sabotage
- Biba integrity model
 - mathematical dual of Bell and LaPadula model
 - never really used for anything in 20 years
 - solution in search of a problem?

Biba Access Rules

- **For read permission**
 - level (object) m level (subject)
 - category set (object) r category set (subject)
- **For write permission**
 - level (object) [level (subject)
 - category set (object) ` category set (subject)

Problems with the Biba Model

- **How do you assign integrity levels?**
- **What do integrity categories mean?**
- **High integrity program is not allowed to validate potentially low integrity data, yet that is precisely what a high integrity application ought be good at doing**

Integrity of Downloaded Code

- **Java and ActiveX models of digitally signed code resemble the Biba model**
 - still have no basis for assigning integrity levels
 - just because IBM wrote and digitally signed some code doesn't make it secure or trustworthy

Modified Biba Model Combined with Digital Signatures

- **Why not use ITSEC or Common Criteria evaluation to assign integrity levels?**
 - independent third party evaluation of applications answers the question of how much can you trust this particular piece of software
 - wouldn't have to be full ITSEC evaluation
 - could use other evaluation schemes that are less formal
- **each downloaded application has TWO digital signatures**
- **developer signs code for identification and to ensure code is not modified**
- **third-party evaluator (actually the certifying body under the ITSEC scheme) ALSO signs code and specifies the integrity level**
- **could have a third signature of card issuer if desired**
- **application is assigned a pair of integrity access classes (more later on this)**

Actual Use of the Model in an Operating System

- **Must distinguish between a program as stored in a file and a process that is in execution**
- **File containing a program has a mandatory secrecy and integrity access class for protecting the file itself**
- **When executing, the process will have two secrecy and two integrity access classes**
 - This is based on the GEMSOS model, but with some differences
- **The four access classes of a process are stored inside the program file and are digitally signed by the certifying body**
 - The certifying body determines the access classes from information provided in the applications' security target

Modified Bell and LaPadula Secrecy Rules

- **A process has two secrecy access classes**
 - read/execute secrecy access class
 - write secrecy access class
- **To read or execute an object**
 - read/execute secrecy access class (process) m secrecy access class (object)
- **To write an object**
 - write secrecy access class (process) [secrecy access class (object)

Modified Biba Integrity Rules

- **A process has two integrity access classes**
 - read integrity access class
 - write/execute integrity access class
- **Note that execute permission is different for integrity and secrecy**
- **To read an object**
 - read integrity access class (process) [secrecy access class (object)
- **To write an object**
 - write/execute integrity access class (process) m integrity access class (object)
- **To execute an object**
 - to branch or transfer in the same process
 - write/execute integrity access class (process) m integrity access class (object)

Modified Biba Model Also Good for Secrecy Downgrading

- **Can establish mandatory policy that says application must be of specified integrity level to be allowed to downgrade Bell and LaPadula secrecy markings**
- **Analogous to NSA's Yellow Book that says what level of evaluation is needed for what risk range of security clearances and classifications.**

NSA Yellow Book Example

Maximum Data Sensitivity

	U	N	C	S	TS	1C	MC
U	C1	B1	B2	B3	*	*	*
N	C1	C2	B2	B2	A1	*	*
C	C1	C2	C2	B1	B3	A1	*
S	C1	C2	C2	C2	B2	B3	A1
TS(BI)	C1	C2	C2	C2	C2	B2	B3
TS(EBI)	C1	C2	C2	C2	C2	B1	B2
1C	C1	C2	C2	C2	C2	C2	B1
MC	C1	C2	C2	C2	C2	C2	C2

Minimum User Clearance

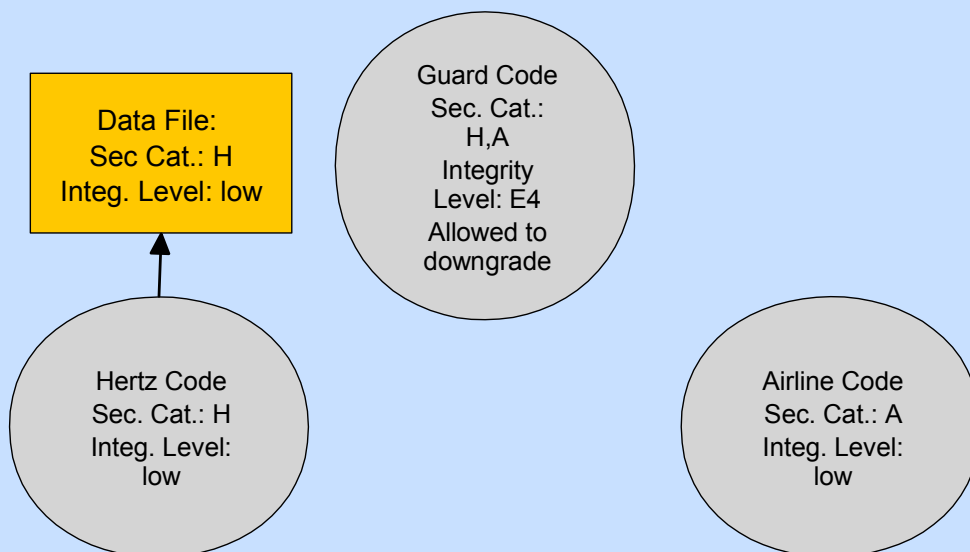
Ideas for Commercial Table

- It is clear than an analogous table can be created for commercial applications
- Concept of risk ranges is much less understood in commercial world
- construct a set of weighting factors
- if sum of all weights exceeds a certain value, then higher evaluation is required
- possible weighting factors
 - Are two companies direct competitors?
 - How much money can be handled in a single transaction?
 - How many transactions can be carried out per unit time?
 - Is the application in a restrictive scripting language?

Example Airline Loyalty Application

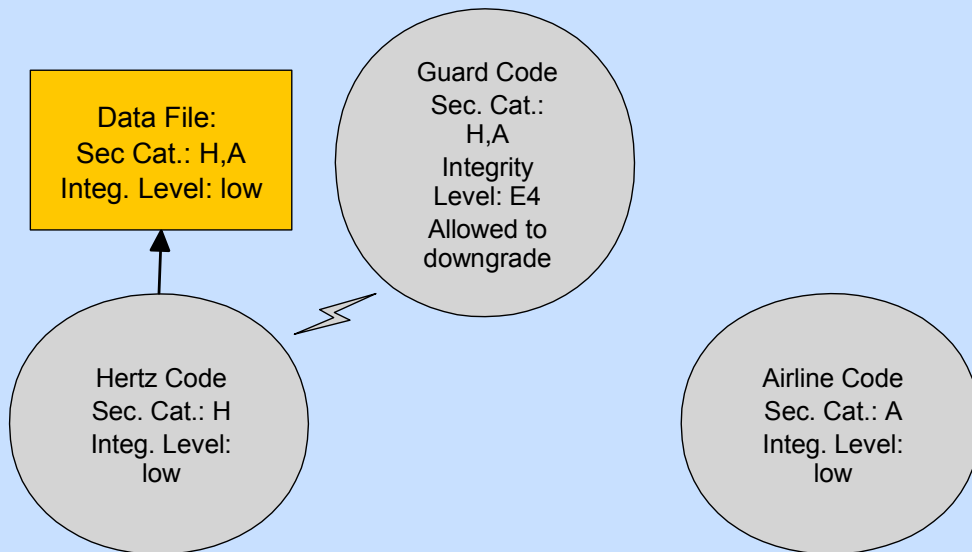
- Airline card with multiple car rental and hotel partners who want to write and load their own code
- Partners do NOT trust one another
- Partners also concerned about possible denial-of-service attacks by competitors
- Need controlled sharing between some partners, but not others
 - If you fly, rent from car company A, and stay at hotel B, you get bonus points, but data must be protected from car company C and hotel D who are legitimate partners, but not part of the bonus deal
- Existing Java-based security policies have not successfully provided either full isolation or this level of controlled sharing between mutually suspicious applets

How Would Loyalty Application Work? Step 1



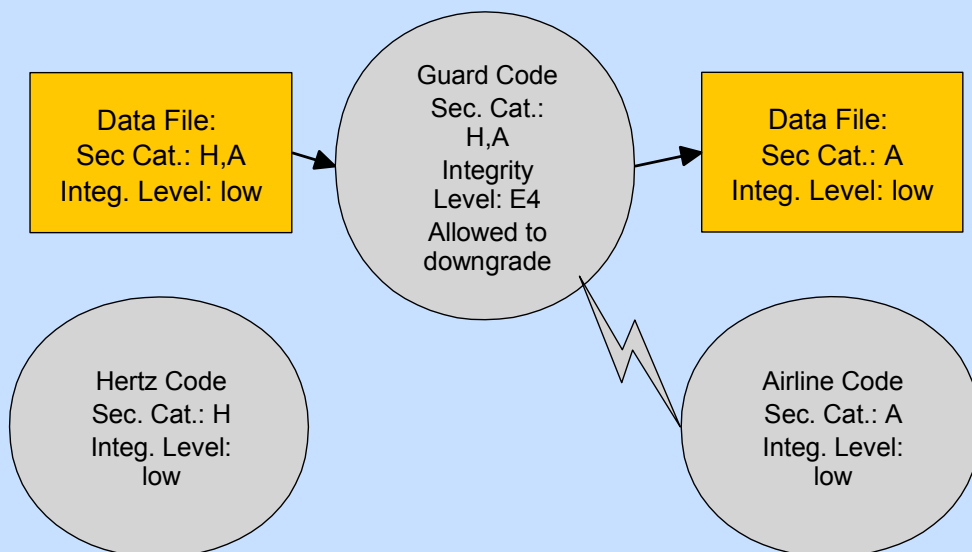
1. Hertz application writes loyalty data.

How Would Loyalty Application Work? Step 2



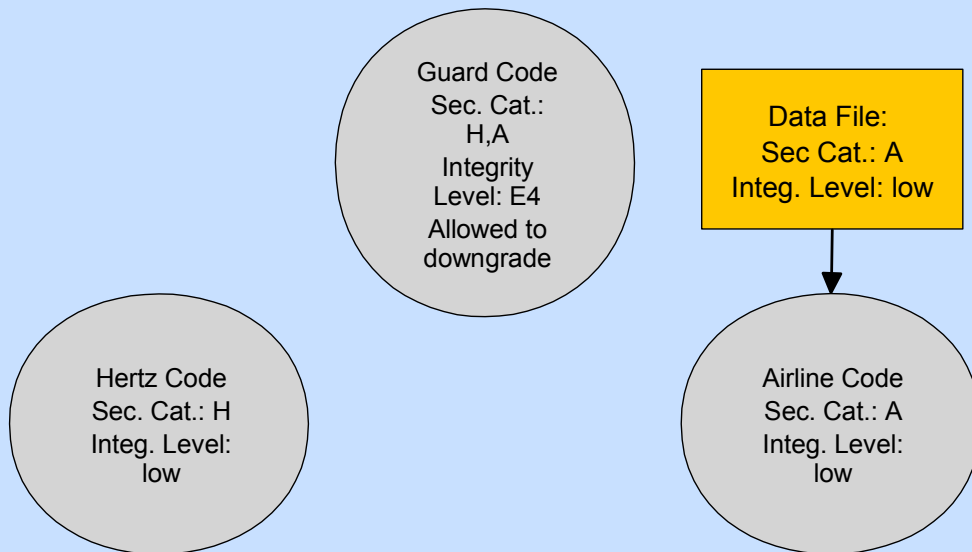
2. Hertz application upgrades data to H,A and starts guard code in separate process.

How Would Loyalty Application Work? Step 3



3. Guard inspects file, verifies that no Hertz secrets are revealed, downgrades to A-only, and starts Airline code.

How Would Loyalty Application Work? Step 4



4. Airline code reads loyalty points.

Important Benefits

- This model clearly separates which application code has to be correct in order to be secure.
- Only the guard application needs to undergo evaluation.
- This clearly assigns responsibilities to different code modules, and reduces overall costs to application developer who wants to deliver high-assurance applications at the lowest possible cost.

Why not just get a JavaCard implementation validated?

Proving that a software-based approach is secure

In a nutshell, it's a harder problem.

To prove that a software-based JavaCard security model is secure, we have to prove that:

- There is no way to generate a bogus address in downloaded code. This is very hard, particularly since the JavaCard VM does fewer run-time checks than a Java VM, due to memory limitations.
- There is no way to download a bogus address into the card.
- The compiler must run in a secured system like the IBM 4758.
- The byte code verifier and the code signer must also run in a secured system like the IBM 4758, and must be formally verified.
- The JavaCard VM itself must be formally verified.

Proving that a hardware-based approach is secure

To prove that a hardware-based approach is secure we have to prove that:

- The hardware memory management and supervisor state work.
- The OS makes proper use of the hardware protection features.
- Even if there is a bogus address in downloaded code, it can do no damage.

Most high assurance systems use this approach.

Security Evaluations: Advantages and Advice

- "Never trust the vendor!"
- Legal requirements
 - German Digital Signature Law (ITSEC E4, CC EAL5)
 - US Government (FIPS 140-1)
- Know what level of security is enough to protect your asset.
- "Designed to meet" does not equal "Meets" or "Validated".
- Ask to see evaluation reports. Note that evaluation reports of specific products are often considered confidential and may not be available to you.

For more information

- on IBM's secure embedded operating system project. . .
www.research.ibm.com/compsci/security/secsystems/scard.htm
- downloadable technical reports on the security policy
www.research.ibm.com/resources/paper_search.html
and then search on the authors' names (Karger, Austel, or Toll)
- on Philips Semiconductors' SmartXA. . .
www-us.semiconductors.philips.com/