INTEGRITY, INTERNAL CONTROL AND SECURITY IN INFORMATION SYSTEMS

CONNECTING GOVERNANCE AND TECHNOLOGY

Edited by
Michael Gertz
Erik Guldentops
Leon Strous

SPRINGER SCIENCE+
BUSINESS MEDIA, LLC

INTEGRITY, INTERNAL CONTROL AND SECURITY IN INFORMATION SYSTEMS

IFIP - The International Federation for Information Processing

IFIP was founded in 1960 under the auspices of UNESCO, following the First World Computer Congress held in Paris the previous year. An umbrella organization for societies working in information processing, IFIP's aim is two-fold: to support information processing within its member countries and to encourage technology transfer to developing nations. As its mission statement clearly states,

IFIP's mission is to be the leading, truly international, apolitical organization which encourages and assists in the development, exploitation and application of information technology for the benefit of all people.

IFIP is a non-profitmaking organization, run almost solely by 2500 volunteers. It operates through a number of technical committees, which organize events and publications. IFIP's events range from an international congress to local seminars, but the most important are:

- The IFIP World Computer Congress, held every second year;
- open conferences;
- working conferences.

The flagship event is the IFIP World Computer Congress, at which both invited and contributed papers are presented. Contributed papers are rigorously refereed and the rejection rate is high.

As with the Congress, participation in the open conferences is open to all and papers may be invited or submitted. Again, submitted papers are stringently refereed.

The working conferences are structured differently. They are usually run by a working group and attendance is small and by invitation only. Their purpose is to create an atmosphere conducive to innovation and development. Refereeing is less rigorous and papers are subjected to extensive group discussion.

Publications arising from IFIP events vary. The papers presented at the IFIP World Computer Congress and at open conferences are published as conference proceedings, while the results of the working conferences are often published as collections of selected and edited papers.

Any national society whose primary activity is in information may apply to become a full member of IFIP, although full membership is restricted to one society per country. Full members are entitled to vote at the annual General Assembly, National societies preferring a less committed involvement may apply for associate or corresponding membership. Associate members enjoy the same benefits as full members, but without voting rights. Corresponding members are not represented in IFIP bodies. Affiliated membership is open to non-national societies, and individual and honorary membership schemes are also offered.

INTEGRITY, INTERNAL CONTROL AND SECURITY IN INFORMATION SYSTEMS

Connecting Governance and Technology

IFIP TC11 / WG11.5 Fourth Working Conference on Integrity and Internal Control in Information Systems November 15–16, 2001, Brussels, Belgium

Edited by

Michael Gertz
University of California, Davis
USA

Erik Guldentops
University of Antwerp Management School
Belgium

Leon Strous
De Nederlandsche Bank NV
The Netherlands

SPRINGER SCIENCE+BUSINESS MEDIA, LLC

ISBN 978-1-4757-5537-4 ISBN 978-0-387-35583-2 (eBook)
DOI 10.1007/978-0-387-35583-2

Library of Congress Cataloging-in-Publication Data

A C.I.P. Catalogue record for this book is available from the Library of Congress.

CONTENTS

PREFACE

IT Governance is finally getting the Board's and top management's attention. The value that IT needs to return and the associated risks that need to be managed, have become so important in many industries that enterprise survival depends on it. Information integrity is a significant part of the IT Governance challenge. Among other things, this conference will explore how Information Integrity contributes to the overall control and governance frameworks that enterprises need to put in place for IT to deliver business value and for corporate officers to be comfortable about the IT risks the enterprise faces.

The goals for this international working conference are to find answers to the following questions:

- what precisely do business managers need in order to have confidence in the integrity of their information systems and their data;
- what is the status quo of research and development in this area;
- where are the gaps between business needs on the one hand and research / development on the other; what needs to be done to bridge these gaps.

The contributions have been divided in the following sections:

- **Refereed papers.** These are papers that have been selected through a blind refereeing process by an international programme committee.
- **Invited papers.** Well known experts present practice and research papers upon invitation by the programme committee.
- **Tutorial.** Two papers describe the background, status quo and future development of CobiT as well as a case of an implementation of CobiT.
- **Vendor white papers.** Vendors of solutions (software) for integrity problems present the background and philosophy of their products.
- **Panel discussion.**

We want to recommend this book to security specialists, IT auditors and researchers who want to learn more about the business concerns related to integrity. Those same security specialists, IT auditors and researchers will also value this book for the papers presenting research into new techniques and methods for obtaining the desired level of integrity.

It is the hope of the conference organizers, sponsors and authors that these proceedings will inspire readers to join the organizers for the next conference on integrity and internal control in information systems. You are invited to take the opportunity to contribute to next year's debate with colleagues and submit a paper or

attend the working conference. Check the websites given below regularly for the latest information.

We thank all those who have helped to develop these proceedings and the conference. First of all, we thank all the authors who submitted papers as well as the keynote and invited speakers, and those who presented papers and participated in the panel. We especially would like to thank mrs. Claudine Gheur for her invaluable support in helping the conference chairs in organizing this conference. Finally, we would like to thank all conference participants, IFIP and the sponsors and supporters of this conference.

November 2001

Michael Gertz
Erik Guldentops
Leon Strous

Websites:

IFIP TC-11 Working group 11.5
http://www.ise.gmu.edu/~csis/faculty/Tc11_5.html

IFIP TC-11
http://www.ifip.tu-graz.ac.at/TC11

IFIP
http://www.ifip.or.at

ACKNOWLEDGEMENTS

Conference chairs:
Michael Gertz, University of California, Davis, USA
Erik Guldentops, University of Antwerp Management School, Belgium
Leon Strous, De Nederlandsche Bank, The Netherlands

Programme Committee:
Co-Chairs:
Michael Gertz, University of California, Davis, USA
Erik Guldentops, University of Antwerp Management School, Belgium
Members/reviewers:
Jacques Beaufays, Eurocontrol, Belgium
Henny Claessens, University of Maastricht, The Netherlands
Bart de Decker, Katholieke Universiteit Leuven, Belgium
Eric Gheur, Galaxia, Belgium
Paul Herremans, Janssen Pharmaceutica, Belgium
Sushil Jajodia, George Mason University, USA
Udo Lipeck, University of Hannover, Germany
Madhavan Nayar, Unitech Systems, USA
Frank Piessens, Katholieke Universiteit Leuven, Belgium
Leon Strous, De Nederlandsche Bank, The Netherlands

Organizing Committee
Leon Strous, De Nederlandsche Bank, The Netherlands
Henny Claessens, University of Maastricht, The Netherlands
Claudine Gheur, Galaxia, Belgium

Organized by:
IFIP TC-11 Working Group 11.5

Sponsored by:

PricewaterhouseCoopers Belgium	(http://www.pwcglobal.com/be)
ThunderStore	(http://www.thunderstore.com)
Unitech Systems	(http://www.unitechsys.com)

Supported by:

BCS and its specialist groups
(British Computer Society and its specialist groups)

BELCLIV - CLUSIB
(Belgische Club voor Informatieveiligheid - Club de la Sécurité Informatique Belge)

IBN - BIN
(Institut Belge de Normalisation / Belgisch Instituut voor Normalisatie)

ICAEW
(Institute of Chartered Accountants in England and Wales)

ISACA BeLux Chapter
(Information Systems Audit and Control Association, Belgium / Luxemburg Chapter)

ISACA NL Chapter
(Information Systems Audit and Control Association, Netherlands Chapter)

ISACA UK Chapter
(Information Systems Audit and Control Association, UK Chapter)

NGI
(Nederlands Genootschap voor Informatica (Dutch computer society))

NOREA
(Nederlandse Orde van Register Edp-Auditors)

SAI
(Studiecentrum voor Automatische Informatieverwerking (Belgian computer society))

PART ONE. REFEREED PAPERS

A CAUTIONARY NOTE REGARDING THE DATA INTEGRITY CAPACITY OF CERTAIN SECURE SYSTEMS

Cynthia E. Irvine
Naval Postgraduate School
irvine@cs.nps.navy.mil

Timothy E. Levin
Naval Postgraduate School
levin@cs.nps.navy.mil

Abstract The need to provide standard commercial-grade productivity applications as the general purpose user interface to high-assurance data processing environments is compelling, and has resulted in proposals for several different types of "trusted" systems. We characterize some of these systems as a class of architecture. We discuss the general integrity property that *systems can only be trusted to manage modifiable data whose integrity is at or below that of their interface components.* One effect of this property is that in terms of integrity these *hybrid-security* systems are only applicable to processing environments where the integrity of data is consistent with that of low-assurance software. Several examples are provided of hybrid-security systems subject to these limitations.

Keywords: integrity, confidentiality, integrity capacity, secure system, multi-level security

1. Introduction

Data integrity is defined as "the property that data has not been exposed to accidental or malicious alteration or destruction." [29] A common interpretation is that high integrity information can be *relied* upon as the basis for critical decisions. However, the protection of high-integrity data in commercial systems has been both problematic to achieve and often misunderstood.

High Assurance Systems are designed to ensure the enforcement of policies to protect the confidentiality and integrity of information. To date, high-assurance systems have been expensive to produce and often lack support for, or compatibility with, standardized user-level applications. *Hybrid security* systems are intended to provide some desired functionality with high assurance of correct policy enforcement by utilizing a combination of high-assurance policy-enforcement components and low-assurance user interface and application components, thus addressing both the expense and compatibility problems typical of high-assurance systems.

In an era when users demand the productivity enhancements afforded by commercial software application suites, hybrid security architectures are of particular interest. Extensive study has demonstrated that hybrid security architectures using commercial user interface components can correctly enforce intended confidentiality policies, e.g. [25]. Less attention has been directed toward the effect of commercial user interface software on the integrity of data managed through those interfaces. Concerns include the integrity of the data modified using these commercial interfaces and then stored by high assurance components, as well as the integrity of data read from high assurance repositories and displayed to users.

While some developers have indicated that this problem is something "we have always known about," the problem may not be fully appreciated by the consumers of these systems. Our premise is that builders and buyers of systems designed to provide high assurance enforcement of security policies should be aware of the impact of component and architectural choices on the integrity of data that users intend to protect. Although the problem is exacerbated in systems designed to implement mandatory integrity models, such as represented by the Biba model [8], it is also significant in systems intended to support confidentiality policies. The former systems have explicit integrity requirements, whereas the latter may have implicit integrity expectations.

1.1 Contributions of this Paper

There is a large body of existing literature regarding integrity enforcement, requirements, and models; most of these address access control and related integrity issues, but do not address integrity capacity problems of system composition.

The National Research Council report on "Trust in Cyberspace" [31] identifies the construction of trustworthy systems from untrustworthy components as a "holy grail" for developers of trustworthy systems. And

a National Computer Security Center guideline[30] that addresses both integrity and confidentiality issues, states that "the ability to run untrusted applications on top of TCBs [1] without undue loss of security is one of the major tenets of trusted computer systems." One of the primary results of our paper is to clarify the limitations of a significant class of these approaches with respect to integrity.

In this paper we examine integrity capabilities of component-based systems and provide a rigorous definition of *system integrity capacity.* This definition can form a basis for reasoning about systems with respect to their suitability for integrity policy enforcement. We also provide examples of several contemporary research-level security systems that exhibit the integrity capacity problem.

Finally, we provide a general conclusion regarding the integrity limitations of hybrid-security system composition: namely, system composition is problematic with respect to maintenance of high integrity data when utilizing commercial-grade products for user interfaces and applications.

1.2 Organization

The remainder of this paper is organized as follows. Section 2 provides a brief discussion of some related efforts involving security and integrity. We review concepts associated with confidentiality, integrity, and assurance in Section 3. Integrity considerations regarding system components and abstract subjects are discussed in Section 4. Section 5 presents the notion of "system integrity capacity," and Section 6 provides a derivation of this capacity for *hybrid security* systems, several of which are described. Our conclusions, in Section 7, complete the body of the paper. A discussion of malicious artifacts in commercial systems is included in Appendix 7.

2. Related Work

The architectural integrity issues we discuss have been addressed only indirectly in the literature. For example, the Seaview papers ([13], etc.) make it clear that the reference monitor will enforce integrity constraints on its subjects, such as the relational database management component; however, they do not explain that the use of a B1-level [2] RDBMS com-

[1] Trusted Computing Base[28]

[2] The terms used in this paper to reflect the evaluation class of systems and components are taken from [28] (e.g., B1 and B2) and [2] (e.g., EAL5).

ponent as the interface to users will limit the integrity range of data that the system can support.

The key issue addressed in our paper is how a system manages modifiable data. The Biba integrity model includes the restriction that a subject may modify an object only if the subject's integrity label "dominates" the object's integrity label. This and related characteristics of the "strict" integrity model are discussed extensively in the literature, starting with [8].

"Program integrity" [39] is related to strict Biba integrity, encompassing all of the restrictions of Biba integrity except for those related to the reading of files, while retaining restrictions for the execution of files. Strict integrity treats execution as a form of reading, whereas program integrity treats them separately [36]. Program integrity is of interest because it can be enforced with simple ring-bracket mechanisms [37], and results in "dominance" or "protection" domains, which can be used to enforce the relationships between the components, subjects and objects discussed in Section 4.

Lipner [24] applies Biba integrity to a real-world business scenario, working through the consistent application of hypothetical integrity labels in the context of a Biba mechanism to protect commercial data from unauthorized modification. This presentation does not address system level integrity problems resulting from the utilization of components with various integrity/assurance levels.

In contrast to low water-mark models, e.g. as discussed in [8], which address changes to the integrity level of a subject as it accesses objects with various integrity levels, we examine how the integrity value of data is affected as it is passed through data-modifying components with heterogeneous integrity properties.

Boebert and Kain [9] recognize the asymmetry of confidentiality and integrity, and remark on the vulnerability of information to corruption when only program integrity is enforced. Their work focussed on the use of domain and type enforcement to construct "assured pipelines" where the integrity level of data is changed as it moves through the pipeline. It does not discuss how software could present intrinsic limitations on the integrity of data to be processed.

Clark and Wilson [11] present a model for the protection of data integrity in commercial systems. In that model, components that modify data or handle user input must be "certified." However, their model does not address the relative integrity of the components and the data, nor does it address the resulting limits to the integrity of data that could be processed by such a system.

With respect to the problem of how to determine integrity labels for objects, Amoroso [4] relates evaluation assurance to software integrity, describing a broad range of (integrity) classes for articulating software trust. Karger suggests that a representation of literal evaluation levels could be used for integrity labels[19].

3. Background

This section sets the context for the presentation of system integrity capacity and attendant problems. Several concepts are examined in relation to integrity, including confidentiality, data versus code, assurance and trust, and multilevel security and the Biba model.

3.1 Integrity and Confidentiality

A given piece of information will have a *confidentiality value* as well as a separate *integrity value* . That is, there will be separately measurable effects (e.g., harm to the information owner) from the leakage vs. the corruption of the information. This is the case whether or not the data has been explicitly *labeled* with confidentiality and integrity designations (as is done in a multilevel-secure system). These labels may indicate both the degree with which we intend to protect the data as well as our assessment of the data's intrinsic value or sensitivity. The *labels* may or may not correspond to the actual integrity or confidentiality *value* of the data (in general, multilevel security models address the security values of the data; whereas the security labels are an implementation issue).

Integrity is, in many ways, the "dual" of confidentiality. Both integrity and confidentiality policies can be represented with labels that represent equivalence classes whose relationships form a lattice [41, 14]. Access control policy decisions can be based on the relative position of labels within a given lattice. Increasing the confidentiality "level" given to a subject (e.g., user) generally *expands* the set of objects that the subject may view; but an increase in integrity may *contract* the set of objects that a subject may view. Such semantic "inversions," and the sometimes non-symmetric nature of integrity and confidentiality properties (e.g., see [9]) can make their differences difficult to reason about. As a result, the analysis of integrity may be overlooked or avoided during the system design or acquisition process, in favor of more familiar confidentiality analyses.

3.2 Integrity of Data and Code

Integrity values are associated with *executable* software (e.g., programs, object code, code modules, components) as well as with *passive* data. In both cases, the integrity value relates to how well the data or program corresponds to its uncorrupted/unmodified original value (e.g., manufactured, installed, or shipped image). For programs, integrity also describes how well a program's behavior corresponds to its *intended behavior* (e.g., documented functionality or design documentation), including the notion that the code does not provide functionality beyond that which was intended (e.g., contain hidden behavioral artifacts). So, "integrity" means that the code has been unaltered, or is faithful to its origin, in both of these ways.

3.3 Assurance and Trust

Integrity of code is also closely related to "assurance" and "trust." Products that have been through security evaluations [28][2] receive an assurance-level designation. A methodical, high-assurance development process may produce code with fewer flaws, and consequently, behavior that is closer to that which is intended, than a low-assurance development process. Suitable security mechanisms and practices must also be in place to ensure the ability of the system to protect itself and provide continued system integrity during operation. This reliable code is sometimes called, or labeled, "high integrity;" it is also referred to as, "high assurance" code. Based on this designation, the product may be deemed suitable for handling data within a certain confidentiality or integrity range. Systems or components with demonstrated capabilities for security policy enforcement are sometimes called "trusted."

3.4 Multilevel Security

Multilevel systems partition data into equivalence classes that are identified by security labels. Data of different sensitivities is stored in different equivalence classes, such that (the data in) some equivalence classes are "more sensitive than," "more reliable than," or "dominate" (the data in) other equivalence classes. The dominance relation forms a lattice with respect to the labels/classes, assuming the existence of labels for universal greatest lower bound, GLB, and universal least upper bound, LUB. A reference validation mechanism (RVM, see "multilevel management component" in Figures 1, 2 and 3), mediates access to objects, controlling object creation, storage, access and I/O, thereby preventing policy-violating data "leakage" across equivalence classes. For

confidentiality policy enforcement, a subject's (e.g., program or component's) ability to write-down or read-up is prevented with respect to the dominance relationship on *confidentiality* labels; for Biba-model integrity, read-down and write-up are prevented with respect to the dominance relationship on *integrity* labels. Most multilevel systems today are designed to enforce confidentiality constraints; some of these are also designed to constrain flow between integrity equivalence classes.

4. Integrity of Components and Subjects

The purpose of this section is to examine how integrity is interpreted with respect to the fundamental building blocks of secure systems.

The abstract architecture we are interested in is one of distributed storage, processing, and interconnection "components." A component is a functional system-level building block made up of software, firmware, hardware or any combination of these. Multiple components may reside on a single computer, but for simplicity's sake, we will assume that a single component does not encompass multiple remotely-coupled computers. Examples of components are shown in Section 6, and include a relational database management system, a security kernel, a client user application, an application server, and a graphical user interface program.

A component can include multiple code modules. The modules may be linked within a process, statically by a compiler/linker, or may have a more dynamic, runtime, linkage. A component can also encompass multiple processes, interconnected through message-passing, remote invocation, or other mechanisms.

Subjects are a modeling abstraction for reasoning about the security behavior of active computer elements such as programs and processes. A primary criteria for identifying a set of active computer elements together as a subject is that each subject has identifiable security attributes (e.g., identity and security level) that are distinct from other subjects. If the security attributes change over time, the elements are sometimes modeled as a different subject.

A component may manifest one or more subjects at a time. Each subject may encompass one or more of the component's modules, for example when they are linked within the same process. In a monolithic architecture, subjects may be identified with separate rings, or privilege levels, of a process [28], especially if the program has different security characteristics in the different rings. Typical systems support confidentiality and integrity labels for abstract subjects that are distinct from the labels on related components and modules (an alternative design would

be to derive the subject label directly from the fixed component label). For example, the assignment of a subject label may be a mapping from the user's current "session level" to the subject representing the user. There are semantic limitations on this assignment with respect to the integrity level of the related modules and components.

4.1 Relation of Component and Subject Integrity

First we consider confidentiality. A subject may be associated with a particular confidentiality equivalence class for enforcement of mandatory access control. The mandatory confidentiality policy is not concerned with what happens between subjects and objects that are in the same confidentiality equivalence class: all reads and writes are allowed. The confidentiality policy is only concerned with what happens when a subject attempts to access an object in another equivalence class. We can interpret the subject reading data from a (source) equivalence class as moving data from that source into the subject's (destination) equivalence class, and writing to a (destination) equivalence class as moving data from the subject's (source) equivalence class to the destination equivalence class. The confidentiality policy says that when data is moved, the confidentiality label of the destination must always dominate the confidentiality of the source (again, data cannot move down in confidentiality).

In contrast, while the integrity policy, too, is concerned with movement of data across equivalence classes (the integrity label of the source must dominate the integrity of the destination), this policy is also concerned with the correctness of modifications, such that even if the subject is in the same equivalence class as the destination object, the modification must be that which has been requested: the *allowed* (e.g., intra-equivalence-class) modifications must be the *correct*, intended, modifications. The tacit assumption in integrity-enforcing systems is that the subject performs the correct modification (only) to its level of integrity (or assurance, if you will). Since an abstract subject's behavior is defined by its code, for coherent enforcement of integrity, the level of integrity assigned to the subject must be no higher than the integrity value of its code.

Components may not always receive an explicit security label, even in a system with labeled modules and other objects. Components may be composed of modules with different security labels. It is conceivable that a given component could be composed of both high-integrity and low-integrity modules, such that subjects with different integrity are

supported by only modules of that same integrity. This would conform to the requirement stated above that a subject's integrity should be no greater than the integrity of its code. However, *most commercial components are not constructed this way. The simplifying assumption for this analysis is that modules within a given component are homogeneous with respect to their integrity, and the integrity of a component is the same as the integrity of its constituent modules.* Thus, we can can generalize the stated requirement to be that the level of integrity assigned to an abstract subject must be no greater than the integrity of the component that manifests the subject.

Combining this component-subject integrity relationship with the subject-object integrity relationship required for data modification (as per the Biba model, above), we arrive at a transitive relationship between the integrity of components and the objects which they access:

> Given the sets of Components, Subjects, and Objects, where each subject in Subjects is an element of one component:
>
> $$\forall\, c \in Components\ ,\ s \in Subjects\ ,\ o \in Objects\ :$$
> $$current\_access(s, o, modify)\ and\ s \in c\ \Rightarrow$$
> $$integrity(c)\ \geq\ integrity(s)\ \geq\ integrity(o)$$

Systems that enforce integrity policies are generally intended to automatically ensure the correct relationship between the integrity level of subjects and the integrity level of accessed objects. However, the enforcement of the relationship between a subject's integrity and its component's integrity may be less clear. Some systems may be able to enforce the relationship between the integrity of subjects and their related modules. For example, this could be enforced by Biba-like labels on executables or other program integrity mechanisms such as rings [38] and ring brackets [1] which can also be represented as attributes on system elements. If these relationships are not enforced during runtime, then the correct relationships may need to be maintained by social convention/procedure.

The relationship between the integrity of a component's subjects and the integrity of the non-software portion of the component is also enforced via social convention (again, component integrity must dominate the subject integrity).

4.2 Component Integrity Labels

This leaves the question of correct integrity labeling of components (and modules). Confidentiality and integrity labels of passive data objects can be correctly assigned based on the data owner's perception of

the object's sensitivity (e.g., harm caused by unauthorized disclosure or modification).

For active objects (viz, code rather than data) integrity labels, as well as confidentiality labels, are usually assigned by the system or network security designer to maximize system security and functionality while being consistent with the principle of least privilege [35]. Best judgment may play a large role in this assignment. For example, if a monolithically-compiled software component is made of up diversely-assured internal modules, it may be the responsibility of a designer, integrator or configuration manager, as stipulated by social convention or procedures, to assign an appropriate integrity level to the executable component. However, the pedigree of the code establishes a real-world limit to the integrity label that can be associated with a component. Intuitively, code that has unknown integrity characteristics, e.g., it is found on the street, should not be accorded a high-integrity label.

The "Yellow Book"[27] is an example of a scheme for determining confidentiality ranges based on the evaluation or assurance level of the components involved, where higher assurance components are allowed to be associated with greater confidentiality ranges. However, there is no "Yellow Book" for integrity to show what integrity label should be allowed or inferred for a code component based on its evaluation/assurance level, although some schemes have been suggested[4, 19].

4.3 Commercial Application Component Integrity

Commercial application components are of particular interest with respect to correct integrity labeling in hybrid security architecture systems (see Section 6). We define *commercial application components* to have been either unevaluated with respect to security policy enforcement, or evaluated below Class B2/EAL5[3]. In the security and evaluation community, components evaluated below B2/EAL5 have historically been considered to be "low assurance" (see, for example, [22]). This is so for several reasons [28, 2]:

- Weak developmental assurance, for example to ensure that unintended malicious artifacts (e.g., Trojan horses and trap doors) are not inserted during manufacture. There is no or very little requirement for system configuration management. There is no requirement for configuration management of development tools.

[3]As there have been few, if any, commercial applications evaluated at B2 or higher, we consider this to be a conservative, non-exclusionary, definition.

- Little or no code analysis, and *no* examination of code for malicious artifacts after manufacture (i.e., during evaluation). There is no requirement for code correspondence [4] to the system specification or for justification of non-policy-enforcing modules. There is no requirement for internal structure (e.g., modularity or minimization) which would enable the meaningful analysis of code functionality.

- Weak assurance that malicious artifacts are not inserted after manufacture. For example, there is no requirement for trusted distribution procedures: no assurance that the system delivered to the end customer is in fact the intended or specified system.

Recall that the semantics of a code integrity label includes an indication of how its behavior corresponds to an intended (e.g., specified) behavior. The fact that there is little assurance that code that has been evaluated below B2/EAL5 functions (only) the way it is supposed to, indicates that there must be a corresponding limit to the value of an integrity label associated with such code (see Appendix 7). We will call this integrity limit, nominally, "low assurance," and assert that components evaluated below B2/EAL5 should be labeled at this, or some lower level. Similarly, code that has not been evaluated at all would be attributed with a (nominal) "no assurance" integrity label. The names of these two labels or the precise evaluation class names are not significant; rather, it is significant to the maintenance of data integrity in hybrid security systems that site security managers/administrators, data owners, and other security policy stake-holders understand the integrity value of their systems' components and of the data entrusted to these systems.

5. Security System Data Capacities

In this section, the notion of *system integrity capacity* will be introduced. This term relates to the ability of a system to handle high-integrity data.

The network architecture of a multilevel system can help to ensure that the actions of other components are constrained by its RVM, for example, through limiting the interconnections or data paths allowed between components. In the architectures discussed in this paper, the separation of data is maintained by either: (1) partitioning the data (and processing elements) into distinct physical equivalence classes and using the RVM to ensure that the security level of the user session matches

[4] Mapping of each specified function to the code that implements it, and accounting for unmapped code.

the security level of the class with which it is connected (e.g., Figure 2), or (2) using the RVM to logically partition the different data equivalence classes and to match the user session level to only the appropriate domain(s) (see Figures 1 and 3).

Our central question is, "for what range of *user data* [5] can we trust such a multilevel system, or any system, to maintain data separation?" Clearly, we would not want to trust a very weak system to protect/separate very highly sensitive information. While our focus is on integrity-related issues, for comparison we will examine cases of both confidentiality and integrity.

5.1 Confidentiality Capacity

For confidentiality, *a multilevel system can be trusted to manage data to the confidentiality range of its RVM.* We call this the *system confidentiality capacity.* For example, if the system's RVM component is assigned or is otherwise deemed capable of managing a range from Unclassified to Secret, we can say the system as a whole is trusted to handle data in that range. This is because the RVM will constrain the actions of the other components to not leak data across equivalence classes, regardless of the level of trust we have in those other components (given a coherent network architecture). To state confidentiality capacity more formally, consider a system, C, comprising a set of components, $\{c\}$, and let RVM be a component in C that enforces the confidentiality policy on other components. Then,

$$c\_capacity(C) = c\_capacity(RVM)$$

5.2 Integrity Capacity

For integrity, on the other hand, *a system can be trusted to manage modifiable data (only) to the integrity limit of its interface components,* where interface components include the various graphical user interfaces and data management applications through which users' data must pass. This is the "system integrity capacity."

System integrity capacity is different from (i.e., not the "dual" of) system confidentiality capacity because we assume that a component will handle modification of objects correctly, only to its level of integrity/assurance. For confidentiality, even if a non-RVM component were infected with malicious code, it could not exfiltrate the information across the equivalence-class boundary, because the RVM component won't let that happen. However, for integrity, once the component has

[5] The ability of a system to protect and maintain *system* data is not addressed in this paper.

approval for modify access, the RVM is powerless to ensure that the correct, and only the correct, modifications are made. Therefore, the assurance level of the individual (viz, non-RVM) component has bearing on its assigned integrity label, but is not necessarily relevant to its assigned confidentiality label.

The input and output mechanisms of a computer system limit the quantity and quality of information that flows through the system, just as the in- and out-flow of water and electricity are limited in hydraulic and electrical systems by their interface devices.

For computers, the I/O mechanisms and related applications, by definition, handle all data entering and leaving the system. Where those mechanisms and applications are configured to be able to modify data, they can potentially effect the integrity of the data entering and leaving the system. The nature of this effect is as follows.

Definitions

C	:	the universal set of components $\{c_1, c_2, \cdots, c_n\}$
O	:	the universal set of objects $\{o_1, o_2, \cdots, o_m\}$
INTEGRITY	:	a lattice of integrity levels: $\{integrity_1, integrity_2, \cdots, integrity_q\}$
modify	:	a relation that defines the fact that a component $c \in C$ has been used to to modify an object $o \in O$
$\mathcal{SYS}$	:	a system comprised of a set of components $c \in C$

Axiom 1

A modified datum is either unchanged in integrity, or takes on an inherent integrity value dominated by the integrity of the data-modifying component.

$$\forall\, c \in \mathcal{SYS},\ o \in O:\ \ modify(c, o) \Rightarrow integrity(c) \geq integrity(o')$$

For example, if a "certified" datum is modified by an "untrusted" code component, the modified datum becomes at best "untrusted," assuming that "certified" dominates "untrusted." If an "untrusted" datum is modified by a "certified" component, the datum becomes at best "certified," indicating it might have been upgraded in integrity.

For a high-assurance integrity-enforcing system, subjects, including the applications that manage user I/O, will be limited by the RVM from modifying protected objects that are above the subject's integrity level. However, if the application is responsible for passing data from one of those objects to, for example, an output device like the computer screen, then the application can simply modify the data in passing without modifying the source object.

Similarly, even if a component does not modify the data directly, it may request that the modification be done by another component, for example, where a user interface component requests from another (e.g., remote) component that an object be created on behalf of the user. Since the requesting component might request the *wrong* modification, we consider it to be a "data-modifying" component. So a system's "data-modifying" components are those components that are able to modify or control the modification of user data. In general, all interface components and other components on the "path" between the user who requests a data access and the ultimate data source (for data reads) or destination (for data writes) are "data-modifying" components, unless they can be guaranteed to not modify, create or delete user data objects or control such operations[6].

Therefore, even for systems that enforce integrity policies, a computer system can only be trusted to manage modifiable data whose integrity is at or below that of its user interface and application components. This is true even if the data is either (1) integrity-upgraded internally by various components, (2) "hand installed" into high integrity internal objects, or (3) imported from specialized high integrity sensing devices, since to be useful, the data will once again be "handled" by the standard interface and application components for access by users. We will note that, theoretically, manual procedures, such as visual inspection of data items retrieved from a hybrid security system, could be used to ensure that processing corruption has not occured, however, this is not generally feasible in commercial or production environments.

As a group, then, the interface components and associated applications determine the integrity limit of the data that a system can handle (*i_capacity*). The interface components are a subset of $\mathcal{SYS}$, indicated $\mathcal{SYS}_{interface}$, and the highest integrity data obtainable from a system $\mathcal{SYS}$ is by way of the user interface component with the highest integrity (viz, the least upper bound of the integrity of all interface components).

$$i\_capacity(\mathcal{SYS}) = \underset{c \in \mathcal{SYS}_{interface}}{\text{LUB}} \left(integrity(c) \right)$$

This gives a "best case" analysis for the integrity that we might expect a system to handle. For example, a high integrity interface application, were it to be available, dependent upon a low integrity database, would not normally improve the integrity of data returned from the database to

[6]The "control" part of this definition makes it broader than the Bell and LaPadula[7] concept of "current access," which indicates only objects with direct access to data.

the user, although this expression of system integrity capacity would indicate that the data accessed through the high integrity interface might be of high integrity. The general case is that the *i_capacity* expression must allow for such upgrades. However, not all systems are designed for data integrity upgrades. A more conservative axiom regarding modification, which does not consider upgrading, results in an *i_capacity* based on the lowest integrity of the components in each path.

Axiom 2

A modified datum takes on an inherent integrity value that is the greatest lower bound of the data and the data-modifying component.

$$\forall\, c \in C\ ,\ o \in O : modify(c, o) \Rightarrow$$
$$integrity(o') = GLB(integrity(c), integrity(o))$$

We now define an individual *data transfer* within the system, a *path* through the system, and the integrity of such a path.

trans: A relation on $C \times C$ that defines an individual transfer of data between components. Data is passed directly from the origin component, c_i, to the terminus component, c_j:

$$c_i\_trans\_c_j$$

path: A sequence of *trans* relations such that for every pair of consecutive relations ($c_i\_trans\_c_j$, $c_j\_trans\_c_k$), the *terminus* of the first and the *origin* of the second coincide[34]. For example, this is a path with n relations: $< c_0\_trans_1\_c_1,\ c_1\_trans_2\_c_2, \cdots,\ c_{n-1}\_trans_n\_c_n >$

The integrity of a path is the greatest lower bound of the components in the path:

$$integrity(path) = \underset{c \in path}{\text{GLB}} \left(integrity(c) \right)$$

Given these definitions, we provide the alternative, more conservative, expression for *i_capacity*.

Let $\pi(\mathcal{SYS})$ be the set of all paths in SYS whose origin or terminus is in $\mathcal{SYS}_{interface}$, then:

$$i\_capacity(\mathcal{SYS}) = \underset{path \in \pi(\mathcal{SYS})}{\text{LUB}} \left(integrity(path) \right)$$

6. Hybrid Security Systems

The systems we are concerned with are those that combine low-assurance commercial components and specialized (e.g., high-assurance) multilevel components specifically to enforce mandatory security policies while using commercial user-level interfaces and applications. These systems, as a class, are composed of the following components:

- commercial terminals or workstations
- commercial user interfaces, applications and application servers
- Storage devices containing multiple levels of data
- Multilevel-management components
- TCB Extensions
- commercial network interconnections

The interested reader is referred to [17] for a detailed description of these components. Of particular note, however, is the description of applications. In the generic "hybrid security" architecture defined in [17], applications interface with the user and participate in the management of all user data. Specifically, the application components have the ability to modify data on behalf of the user (which is to say that read-only systems are not of interest). The general functionality of commercial applications such as word processing, spread sheet, slide presentation, time management, and database tools indicate that, to be useful, they are intended to modify, as well as read, data.

To illustrate the relevance of our concerns for the handling of high integrity data in hybrid security systems, we describe here several systems from the security literature that exhibit dependence on the integrity of commercial components.

A non-distributed version of the model architecture is shown in Figure 1. In this layout, the component interconnections consist of process-internal communications. The lowest layer (viz, "ring") of the process is a multilevel kernel or operating system, with an application (e.g., multilevel-aware RDBMS) and user interface in higher layers. A separate process is created for each security level. An example of this version of the architecture is that of the Seaview project [15, 25], and "Purple Penelope" [33] (the latter includes a degenerate case of a RVM). A variation on this theme is the trusted Virtual Machine Monitor (VMM) architecture, in which a separate version of the OS, in addition to the application and user interface, is created at each security level[20, 26, 6], and multilevel management occurs below that in the VMM layer.

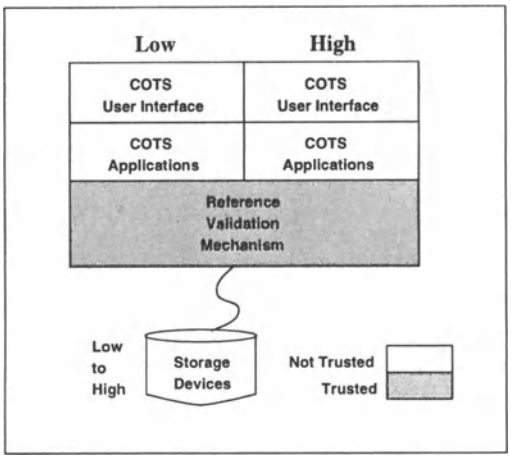


Figure 1. Single Process Architecture (Network Connections are degenerate.)

A simple distributed instantiation is shown in Figure 2. Here, there are logically separate single-level workstations connected by a switch to data management subsystems at different (single) levels. Software associated with the switch ensures that the current level of the workstation matches the level of data subsystem indicated by the switch setting. An example of this version of the architecture is that of the Starlight project [5] (Starlight may allow low confidentiality information to flow through the switch to high sessions, providing "read-down" capability).

The third instantiation of the model architecture is shown in Figure 3. In this layout, there are logically separate single-level terminals (multiplexed onto one physical terminal by purging of state between session-level changes) connected via TCB extensions to multilevel-aware application server(s) running on the multilevel (TCB) component. An example of this version of the architecture is that of the Naval Postgraduate School's Monterey Secure Architecture (MYSEA) system, based on [18].

6.1 Integrity Capacity of Hybrid Security Systems

Based on the preceding discussion, the system integrity capacity of *hybrid security* systems can be summarized as follows:

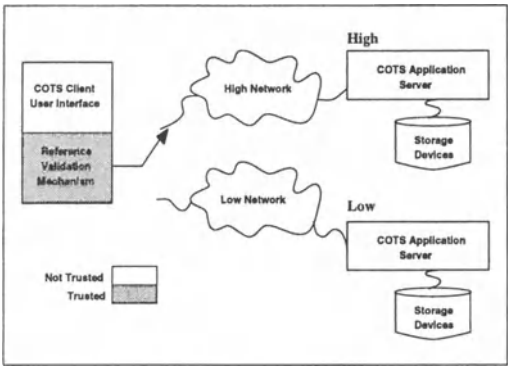


Figure 2. Switch-Based Architecture

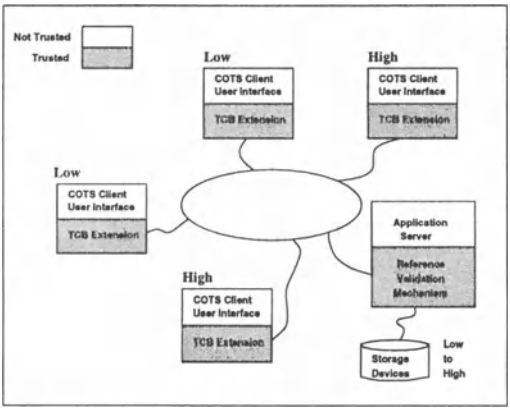


Figure 3. Distributed Multilevel Server Architecture

- A system's integrity capacity is the LUB of the integrity of its interface/application components.
- All interface/application components in hybrid security systems are commercial
- Commercial interface/application components are of "no assurance" or "low assurance" integrity.

- Therefore, the system integrity capacity of a *hybrid security* system is generally no higher than "no assurance" or "low assurance."

An implication of this conclusion is that *hybrid security architecture* systems are not suitable for automated information processing environments in which there are expectations or requirements to maintain data integrity above the nominal "no assurance" or "low assurance" level. Another implication is that composition of trusted systems utilizing only commercial products as interface components is problematic with respect to integrity.

7. Conclusion and Discussion

We have shown that the integrity of a computer system's interface components limits the data "integrity capacity" of the system. This is in contrast to the "confidentiality capacity" of a system, which is determined by characteristics of the system's policy-enforcement component(s), but is not dependent on the interface components.

We have discussed why commercial components should not be attributed with integrity properties above a certain "low-assurance" level, and that *hybrid security* systems should not be trusted with data whose integrity is above that level. An implication from this conclusion is that *hybrid security* systems are not suitable in computing environments where there is an expectation of maintaining data integrity above this basic, low-assurance level.

Situations where corrupted data could have significant consequences are:

- A legal setting where the "truth" of data might be questioned
- Handling of high integrity intelligence data for critical decision making
- The production of high assurance system components
- Systems where human life might be affected by improper execution of code
- High reliability embedded systems

We have concentrated on issues of integrity in multilevel secure systems, however, the distinctions we have made are germane to other systems where weak integrity components are utilized and stronger data integrity is expected.

One might say, "what difference does it make if a component has too high of an integrity label, and its real integrity value is low? These

commercial software vendors can be generally trusted, since it is in their best interest to ship a product that does not corrupt data." This attitude reflects a common misunderstanding of data integrity enforcement. Certainly, most security analysts and engineers would agree that high-assurance policy-enforcement components are needed to safeguard the *confidentiality* of highly sensitive multilevel data; then, why would there be any lesser concern for the ability of a system to protect the *integrity* of highly sensitive data? From a more technical viewpoint, if a system's objects do not have data sensitivity (confidentiality and integrity) labels that match the objects' real sensitivity values, then the system does not correspond to its model, and its behavior may be undefined. Also, refer to Appendix 7, for a review of common "Subversive Artifacts in Commercial Software."

The result presented in this paper places a limit on what is achievable in system integrity architectures. Such a finding can help to refine the direction for constructive efforts and does not preclude the construction of useful systems any more than other negative results, e.g. [16, 21], have in the past.

One might also ask if high integrity is ever achievable. The answer is yes, but not with the type of commodity application components available today (viz, where commodity implies weak integrity of software functionality). Systems that *could* provide high integrity today are (1) a system composed entirely of high-assurance components, or (2) a system that protects high integrity data from modification by all but high-assurance components. Examples of the first are systems intended to perform safety-critical functions such as avionics and certain medical systems[23]. An example of the second is a client-server system composed of high-assurance client (e.g., web browser) and server components that encrypt their communication such that it is protected from modification during transit through low assurance network components (e.g., via a Virtual-Private-Network-style connection). As noted previously, such systems carry the expense of custom high-assurance development.

Appendix: Subversive Artifacts in Commercial Software

There is clear evidence that subversion of commercial software through hidden entry points (trap doors) and disguised functions (Trojan horses) is more common than generally perceived. Entire web sites [3] are devoted to describing clandestine code which may be activated using undocumented keystrokes in standard commercial applications. Sometimes this code merely displays a list of the software developers' names. Other times the effects are extremely elaborate as in the case of a flight simulator embedded in versions of the Microsoft Excel Spreadsheet software. That these "Easter Eggs" are merely the benign legacy of the programming team is perhaps

a reflection of the general good intentions of the programmers. Malicious insertions, such as long-term time bombs, are just as easily possible.

An indication of the serious nature of the problem was provided in April 2000 when news reports created a mild hysteria surrounding the possibility of a trapdoor in the code of a widely used web server[32]. Subsequent investigations revealed that instead of a trapdoor, the code contained nasty remarks about corporate competitors and well as violations of company coding standards[12]. The fact remains, however, that when rumors of the trapdoor were initially published, few believed that artifices of this type were possible in such a popular software product. However, millions of users do not eliminate the problem of low integrity. Another example of the vulnerability of commercial source code occured in October 2000, when it was revealed that outsiders had access to the development environment of a major software vendor for some period of time[10].

In his Turing Prize Lecture, Ken Thompson described a trapdoor in an early version of the Unix operating system [40]. The cleverness of the artifice was evident in that the artifice was said to have been inserted into the operating system executable code by the compiler, which had been modified so that recompilations of the compiler itself would insert the trapdoor implantation mechanism into its executable while leaving no evidence of the trapdoor in the source code for either the operating system or the the compiler. The presence of this sort of trap door is speculative in *any* compiler and must be addressed through life-cycle assurance of tools chosen for high assurance system development.

References

[1] Gemini Trusted Network Processor (GTNP). In *Information Systems Security Products and Service Catalog Supplement*, Report No.CSC-PB-92/001. April 1992. 4-SUP-3a.3.

[2] ISO/IEC 15408 - Common Criteria for Information Technology Security Evaluation. Technical Report CCIB-98-026, May 1998.

[3] *The Easter Egg Archive.* http://www.eeggs.com/, last modified 19 May 2000.

[4] E. Amoroso, J. Watson, T. Nguyen, P. Lapiska, J. Weiss, and T. Star. Toward an approach to measuring software trust. In *Proceedings 1991 IEEE Symposium on Security and Privacy*, pages 198–218, Oakland, CA, 1991. IEEE Computer Society Press.

[5] M. Anderson, C. North, J. Griffin, R. Milner, J. Yesberg, and K. Yiu. Starlight: Interactive Link. In *Proceedings 12th Computer Security Applications Conference*, San Diego, CA, December 1996.

[6] S. Balmer and C. Irvine. Analysis of Terminal Server Architectures for Thin Clinents in a High Assurance Network. In *Proceedings of the 23rd National Information Systems Security Conference*, pages 192–202, Baltimore, MD, October 2000.

[7] D. E. Bell and L. LaPadula. Secure Computer Systems: Mathematical Foundations and Model. Technical Report M74-244, MITRE Corp., Bedford, MA, 1973.

[8] K. J. Biba. Integrity Considerations for Secure Computer Systems. Technical Report ESD-TR-76-372, MITRE Corp., 1977.

[9] W. Boebert and R. Kain. A practical alternative to hierarchical integrity policies. In *Proceedings 8th DoD/NBS Computer Security Conference*, pages 18–27, Gaithersburg, MD, September 1985.

[10] T. Bridis, R. Bickman, and G. Fields. Microsoft Said Hackers Failed to See Codes for Its Most Popular Products. http://interactive.wsj.com/archive/retrieve.cgi?id=SB972663334793858544.djm, October 2000.

[11] D. Clark and D. R. Wilson. A Comparison of Commercial and Military Computer Security Policies. In *Proceedings 1987 IEEE Symposium on Security and Privacy*, pages 184–194, Oakland, CA, April 1987. IEEE Computer Society Press.

[12] R. Cooper. Re: Security experts discover rogue code in Microsoft software. http://catless.ncl.ac.uk/Risks/20.88.html#subj11, May 2000.

[13] D. Denning, T. F. Lunt, R. R. Schell, W. Shockley, and M. Heckman. The seaview security model. In *Proceedings 1988 IEEE Symposium on Security and Privacy*, pages 218–233, Oakland, CA, April 1988. IEEE Computer Society Press.

[14] D. E. Denning. *Secure Information Flow in Computer Systems.* PhD thesis, Purdue Univeristy, West Lafayette, IN, May 1975.

[15] D. E. Denning, T. F. Lunt, R. R. Schell, W. Shockley, and M. Heckman. Security policy and interpretation for a class a1 multilevel secure relational database system. In *Proceedings 1988 IEEE Symposium on Security and Privacy*, Oakland, CA, April 1988. IEEE Computer Society Press.

[16] M. Harrison, W. Ruzzo, and J. Ullman. Protection in Operating Systems. *Communications of the A.C.M.*, 19(8):461–471, 1976.

[17] C. Irvine and T. Levin. Data integrity limitations in highly secure systems. In *Proceedings of the International Systems Security Engineering Conference*, Orlando, FL, March 2001.

[18] C. E. Irvine, J. P. Anderson, D. Robb, and J. Hackerson. High Assurance Multilevel Services for Off-The-Shelf Workstation Applications. In *Proceedings of the 20th National Information Systems Security Conference*, pages 421–431, Crystal City, VA, October 1998.

[19] P. Karger, V. Austel, and D. Toll. A new mandatory security policy combining secrecy and integrity. Technical Report RC 21717(97406), IBM Research Division, Yorktown Heights, NY, March 2000.

[20] P. A. Karger, M. E. Zurko, D. W. Bonin, A. H. Mason, and C. E. Kahn. A VMM Security Kernel for the VAX Architecture. In *Proceedings 1990 IEEE Symposium on Research in Security and Privacy*, pages 2–19. IEEE Computer Society Press, 1990.

[21] B. Lampson. A Note on the Confinement Problem. *Communications of the A.C.M.*, 16(10):613–615, 1973.

[22] T. M. P. Lee. A Note on Compartmented Mode: To B2 or not B2? In *Proceedings of the 15th National Computer Security Conference*, pages 448–458, Baltimore, MD, October 1992.

[23] N. G. Levenson. *Safeware- System safety and Computers.* Addison-Wesley, 1995.

[24] S. B. Lipner. Non-Discretionary Controls for Commercial Applications . In *Proceedings 1982 IEEE Symposium on Security and Privacy*, pages 2–20, Oakland, 1982. IEEE Computer Society Press.

[25] T. F. Lunt, R. R. Schell, W. Shockley, M. Heckman, and D. Warren. A Near-Term Design for the SeaView Multilevel Database System. In *Proceedings 1988 IEEE Symposium on Security and Privacy*, pages 234–244, Oakland, 1988. IEEE Computer Society Press.

[26] R. Meushaw and D. Simard. Nettop. *Tech Trend Notes*, 9(4):3–10, Fall 2000.

[27] National Computer Security Center. *Computer Security Requirements, Guidance for Applying the Department of Defense Trusted Computer System Evaluation Criteria in Specific Environments*, CSC-STD-003-85, June 1985.

[28] National Computer Security Center. *Department of Defense Trusted Computer System Evaluation Criteria*, DoD 5200.28-STD, December 1985.

[29] National Computer Security Center. *Trusted Network Interpretation of the Trusted Computer System Evaluation Criteria*, NCSC-TG-005, July 1987.

[30] National Computer Security Center. *A Guide to Understanding Covert Channel Analysis of Trusted Systems*, NCSC-TG-030, November 1993.

[31] National Research Council. *Trust in Cyberspace*, Washington, DC, 1999. National Academy Press.

[32] Newsscan.com. Security experts discover rogue code in Microsoft software. http://catless.ncl.ac.uk/Risks/20.87.html#subj8, April 2000.

[33] B. Pomeroy and S. Weisman. Private Desktops and Shared Store. In *Proceedings 14th Computer Security Applications Conference*, pages 190–200, Phoenix, AZ, December 1998.

[34] F. P. Preparata and R. T. Yeh. *Introduction to Discrete Structures*. Addison-Wesley Publishing, Co., Reading, MA, 1973.

[35] J. H. Saltzer and M. D. Schroeder. The Protection of Information in Computer Systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[36] R. Schell and D. Denning. Integrity in trusted database systems. In *Proceedings 9th DoD/NBS Computer Security Conference*, Gaithersburg, MD, September 1986.

[37] M. D. Schroeder, D. D. Clark, and J. H. Saltzer. The Multics Kernel Design Project. *Proceedings of Sixth A.C.M. Symposium on Operating System Principles*, pages 43–56, November 1977.

[38] M. D. Schroeder and J. H. Saltzer. A Hardware Architecture for Implementing Protection Rings. *Comm. A.C.M.*, 15(3):157–170, 1972.

[39] L. J. Shirley and R. R. Schell. Mechanism Sufficiency Validation by Assignment. In *Proceedings 1981 IEEE Symposium on Security and Privacy*, pages 26–32, Oakland, 1981. IEEE Computer Society Press.

[40] K. Thompson. Reflections on Trusting Trust . *Communications of the A.C.M.*, 27(8):761–763, 1984.

[41] K. B. Walter, W. F. Ogden, W. C. Rounds, F. T. Bradshaw, S. R. Ames, and D. G. Shumway. Primitive Models for Computer Security. In *Case Western Reserve University Report*, ESD-TR-74-117, January 1974. Electronic Systems Division, Air Force Systems Command.

DEVELOPING SECURE SOFTWARE

A survey and classification of common software vulnerabilities

Frank Piessens
Dept. of Computer Science
Katholieke Universiteit Leuven, Belgium
Frank.Piessens@cs.kuleuven.ac.be

Bart De Decker
Dept. of Computer Science
Katholieke Universiteit Leuven, Belgium
Bart.DeDecker@cs.kuleuven.ac.be

Bart De Win
Dept. of Computer Science
Katholieke Universiteit Leuven, Belgium
Bart.DeWin@cs.kuleuven.ac.be

Abstract More and more software is deployed in an environment with wide area network connectivity, in particular with connectivity to the Internet. Software developers are not always aware of the security implications of this connectivity, and hence the software they produce contains a large number of vulnerabilities exploitable by attackers.

Statistics show that a limited number of types of vulnerabilities account for the majority of successful attacks on the Internet. Hence, we believe that it is very useful for a software developer to have a deep understanding of these kinds of vulnerabilities, in order to avoid them in new software. In this paper, we present a survey and classification of the most commonly exploited software vulnerabilities.

Keywords: security, software vulnerabilities, software engineering

1. Introduction

At the root of almost every security incident on the Internet are one or more software vulnerabilities, i.e. security-related bugs in the software that can be exploited by an attacker to perform actions he should not be able to perform.

Experience shows that a majority of these software vulnerabilities can be traced back to a relatively small number of causes: software developers are making the same mistakes over and over again. Looking for instance at the list of the ten most exploited software vulnerabilities (see: [11]), one can see that many of these vulnerabilities are actually buffer overflow problems.

Hence, we believe that it is useful to try to survey and classify the most frequently occuring types of vulnerabilities. This paper identifies a number of categories of software vulnerabilities, and gives extensive examples of each of these categories. A software engineer familiar with these categories of problems is less likely to fall prey to these same problems again in his own software.

The structure of this paper is as follows: in the next section, we present a structured classification of software vulnerabilities. In section 3, we present a number of easily remembered guidelines a software developer can keep in mind to steer clear of most of the identified categories of problems. We conclude by discussing related work and summarizing our results.

2. An illustrated survey of software vulnerabilities

2.1 Insufficiently defensive input checking

A developer regularly makes (often implicit, and at first sight very reasonable) assumptions about the input to his programs. An attacker can invalidate these assumptions to his gain. It is important to realize that *input* should be interpreted in a broad way: input could be given to a program through files, network connections, environment variables, interaction with a user etc... If method calls in a program can cross protection domains (as is the case for instance in Java), even method parameters should be considered as non-trustworthy input that needs careful checking.

Examples of this category include: buffer overflows and weak CGI scripts.

2.1.1 Buffer overflows. One of the most successful attacks is certainly a buffer overflow in a server process. What happens is illustrated in figure 1. For every function call, the parameters, the return address[1], and the local variables of the function are put on top of the stack, the so-called current stack frame. In figure 1.a, the normal situation is sketched. The return address points to the correct instruction. If the function is not carefully programmed and does not take into account the actual sizes of the variables, then a buffer overflow might happen. For instance, many servers expect input from client processes, such as a name, a path, an email-address, etc. Often, the server has allocated a buffer for this input in the current stack frame. The buffer is usually oversized and is certainly large enough to accommodate all 'reasonable' input. However, if an attacker sends input that is much larger than expected, and if the server does not take appropriate precautions (e.g. it copies the input until a zero-byte is found), then part of the stack frame is overwritten (see also figure 1.b): the return address is modified and points to code that has been sent as part of the input! Hence, the attacker can make the server execute whatever code he wants to. Usually, the code that is sent as input will make the server spawn a new process that runs the command interpreter (the shell). That way, the attacker gets inside the system without a login procedure. Often, he has superuser privileges, since the command interpreter inherits the privileges of the attacked server.

The last decade, many server programs have been found to be vulnerable to this kind of attack (the finger daemon, bind daemon, ...). Often, these servers used a getstring function that did not limit the length of the string to be read. However, newer attacks do not use 'oversized' input, but cause buffer overflow by other means. For instance, inputs with special characters (wildcards, ...) are sometimes expanded (globalized) by the server, leading to buffer overflow. Also, incomplete inputs may divert (temporarily) the flow of control inside the server; after this diversion, assumptions about the sizes of the variables may no longer be true.

2.1.2 Weak CGI scripts. CGI, an acronym for Common Gateway Interface, is a mechanism for extending the webserver. Instead of sending a webpage in response to a client request, the server starts a new process which will handle the request. Typically the subprocess runs a script (e.g. Perl, Tcl, ...). These languages offer pattern-matching

[1]The return address is the address of the instruction that must be executed after the function call.

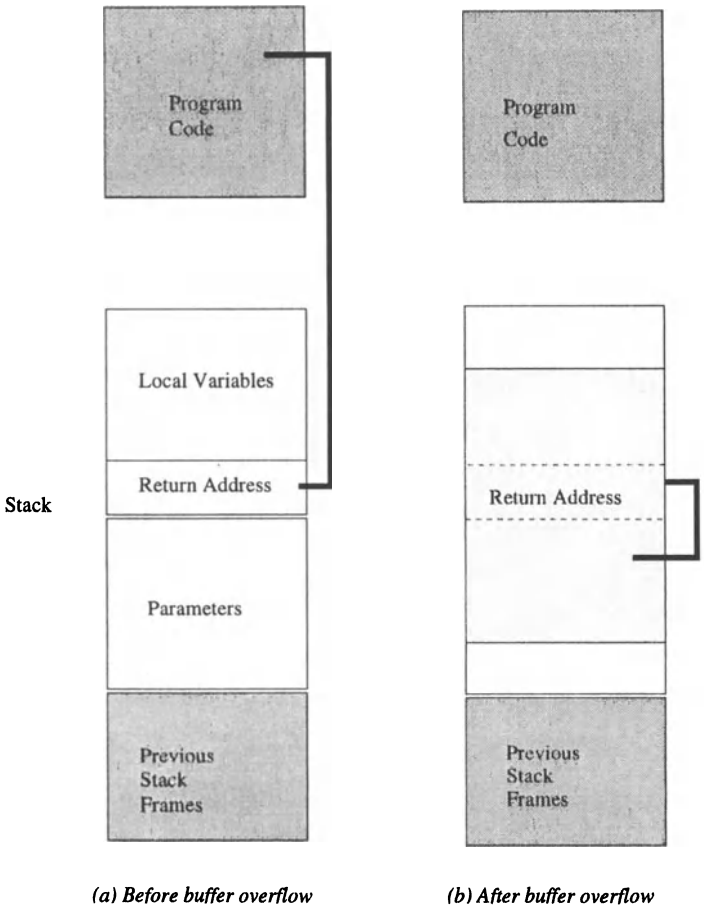


Figure 1. Buffer overflow attack

and many other features, that are useful in this context. However, the expressions are even more powerful than those supported by command interpreters (shells). Since the user inputs are passed to the script, insufficient checking of these inputs may lead to disaster. Figure 2 illustrates the CGI-mechanism. Often, CGI scripts are used to process 'forms' included in certain web pages. The users completes the input fields and submits the form to the server. The server spawns a new process that runs the script, and passes the user inputs either through environment variables, or via standard input. Assume that one of the inputs is an

email-address, that will be used to send a confirmation message to the user, and that the Perl script contains the following lines of code:

```
⋮
$emailaddress = ...;  # fetch the email address
⋮
system("echo \"Your form has been processed\" | mail $emailaddress");
⋮
```

If the email address is not checked by the script, then a user has the ability to have the script execute whatever command the user wants. In this case, if the user gave as email address: `user@dot.com; rm -rf /` then, eventually the following commands will be executed:

```
echo "Your form has been processed" | mail user@dot.com; rm -rf /
```

that is, the acknowledgement is sent to the user, and the file system may be wiped out if the web server is running with superuser privileges! Instead of this denial of service attack, the malicious user could also try to add an extra line to the password file and thus create for him an entrance gate to the system.

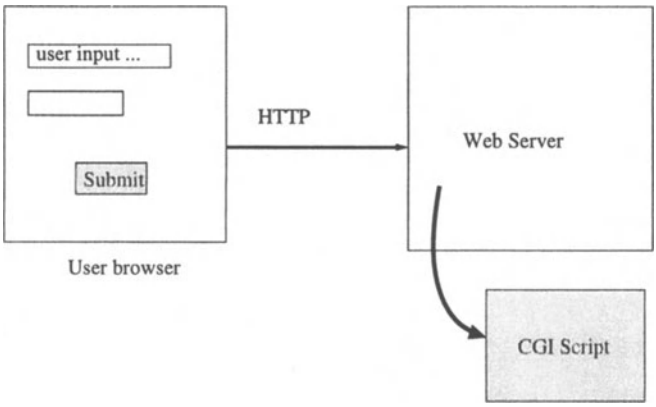


Figure 2. CGI Scripts

2.2 Reuse of software in more hostile environments

Software written for use in a relatively friendly environment (like a mainframe or an intranet) is often reused later in a more hostile en-

vironment (like the Internet). Because the software developer made certain assumptions about the environment in which his program would be running, this change in environment can lead to major security holes. Typical examples include programs using password authentication, and document processing software reused as content viewers on the Internet.

2.2.1 Password authentication. Under the assumption that passwords are well chosen, and well guarded by their owners, password authentication is relatively secure in an environment where the communication between the user typing in the password and the computer verifying the password can not be eavesdropped on by attackers. Typically, for a terminal connected to a mainframe by a dedicated line, a password mechanism is sufficiently secure.

However, in a context where the password is communicated over the Internet, password authentication is extremely weak: eavesdropping on connections is commonplace on the Internet, and once a password has been seen by somebody else, the security of the mechanism is completely broken. Still, many popular programs like telnet and ftp rely on this mechanism to authenticate connections over the Internet.

2.2.2 Document processing software reused as Internet content viewer. If word processing software is used to create, edit and view documents authored by the owner of the software, or a small set of trusted colleagues, then the security requirements of this software are relatively low. If the software contains buffer overflow problems, it might crash occasionally, but it does not represent a major security problem.

This situation changes completely if the same word processing software is reused as a viewer for Internet content. The same buffer overflow problem can now be maliciously exploited: an attacker places a carefully constructed document on the Web, and tries to lure victims into viewing this document. By exploiting the buffer overflow problem, the attacker can do anything he wants on the victims computer.

Since it is difficult to predict in advance in which contexts your software will be used, it is good practice to strive for secure software development even for software that will initially only be used in a friendly environment.

2.3 Trading off security for convenience or functionality

It is well-known that there is a trade-off between security and convenience (i.e. functionality or user-friendliness of the software). Most

security measures tend to add some user-annoyance, and often very powerful and convenient features are easy to abuse. Software developers, tending to think of functionality in the first place, usually emphasize convenience over security. This problem is often an attitude problem: software developers tend to spend a lot of time thinking about how to make things possible. From a security point of view it is as important to spend time thinking about how to make certain things *impossible.*

Examples of vulnerabilities in this category include: executable attachments and powerful scripting languages for applications.

2.3.1 Executable attachments. Many browsers maintain a table which is used to determine how the browser should handle MIME types when it encounters MIME parts in a HTML document, be it an email message, a newsgroup posting, a web page, or a local file. Some of these entries may cause the browser to open the MIME part without giving the end user the opportunity to decide whether the MIME part should be opened. Hence, an intruder may construct malicious content that, when viewed in the browser (or any program that uses the browser's HTML rendering engine), can execute arbitrary code. It is not necessary to run an attachment; simply viewing the document in a vulnerable program is sufficient to execute arbitrary code.

2.3.2 Powerful scripting in applications. More and more applications include an interpreter for a scripting language, which can be used to support 'dynamic' content. Examples are word processors, spreadsheets, web browsers, etc. The problem with these scripting languages is that they are very powerful, and often allow access to local system resources, such as the file system. Although a technique, called sandboxing, can shield off the local system, dynamic content can still mislead the user, and possibly capture confidential information, such as credit card numbers, passwords, etc.

The following attack against web browsers that support JavaScript has been described by Felten, Balfanz, Dean and Wallach ([5]). See also figure 3. An unsuspecting user is lured to the attacker's website[2]. The web document shown to the user is 'booby-trapped': it contains a JavaScript program, which disables the normal functioning of the browser's buttons (by covering the browser with an invisible window), and all URLs are rewritten in order to direct requests to the attacker's site. From now

[2]This is probably the easiest part of the attack. It suffices to offer something for free, to attract many possible victims.

on, the browser is actually captured by the attacker. There is no way to escape. Every URL in the document is of the form:

`http://www.attacker.com/http://www.real.com/page.html`

Hence, the request is sent to `www.attacker.com`, which will forward to request to `www.real.com`. The web document that is returned by that server is then rewritten by the attacker's website, i.e. all URLs are rewritten and a JavaScript program is added to the document. The modified document is finally sent to browser. Note that the JavaScript program can hide these modifications to the user: if the browser is asked to show the 'HTML source' of the document, the script will remove the malicious code and show the original URLs. Since all requests are sent to the attacker's website, including input fields of forms, the site may acquire and abuse confidential information.

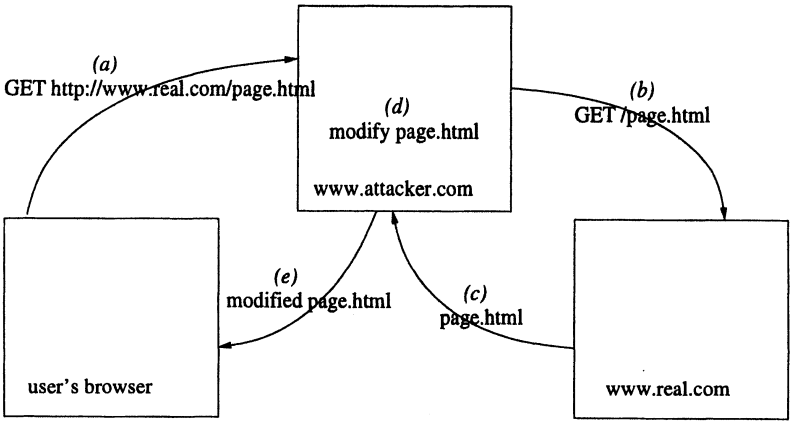


Figure 3. A webspoofing attack

2.4 Relying on non-secure abstractions

Many abstractions offered by a programming language or by an operating system are "complexity-hiding" abstractions more than "tamper-proof" abstractions. Software developers often (implicitly or explicitly) assume that these abstractions are tamper-proof anyway, leading to security breaches. Examples include: buffer overflows (see section 2.1.1), type confusion problems in Java, attacks against smartcards, considering TCP/IP connections as reliable communication channels, unanticipated object reuse,etc... We discuss two examples in more detail.

2.4.1 Type confusion in Java. To allow untrusted code limited access to objects, it seems reasonable to use object oriented access specifiers (like private or protected) on methods or fields that should not be accessible to the untrusted code. The programmer relies on the information hiding aspects of the object oriented language he is working in to achieve a security related goal.

However, it is important to realize that access specifiers are by no means "tamper-proof" in most object oriented languages. For example, in C++, untrusted code can scan the entire memory range in use by a process by casting integers to pointers, and hence untrusted code can also access the private fields of any object. In other words, the OO abstractions offered by C++ are not secure, or C++ is not memory safe or type safe.

The designers of Java tried to make the Java Virtual Machine memory safe and type safe by disallowing pointers, and by checking casts even at runtime. But for many versions of the JVM, bugs in the implementation of the VM have led to breaches in memory and type safety. For example, the well-known classloader-attack (see: [8]), breaks the type safety of all JVM's upto version 1.1.8.

Even in the absence of type safety problems, untrusted code may try to access private fields of an object by serializing the object, and reading the resulting file as a byte array.

2.4.2 Unanticipated resource reuse. The problem here is that the software developer disposes of some object or resource (e.g. deletes a file), and assumes that by disposing of the object, its information content becomes inaccessible. In many cases the object will only be "logically" deleted, and the actual content is still retrievable by an attacker.

2.5 Insecure defaults and difficult configuration

The default configuration of general purpose software is often not secure to guarantee that the majority of customers is able to use it without experiencing too many restrictions. Especially for operating systems, this is common practice. Clearly, the users impression about the system is important and security restrictions might annoy him. However, this should not be a reason to lower the level of security or it should be explicitly and very well documented. And even in this case, system administrators typically do not take the time to read this documentation. They tend to make a default install, and if that works leave it at that. Hence, if the default configuration is an insecure one, many installations will be in an insecure state.

As an example, Microsoft Windows NT 4.0 is a reasonably secure operating system as proven by the ITSEC E3/F-C2 label it received after independent evaluation. However, a default install of the system disables many of the security features. Several documents (see: [4, 2]) provide checklists of tasks an administrator should perform to enable important security features and as such augment the overall security level of the system. However, it is highly questionable how many users will follow all the guidelines described in these checklist documents.

As another point of attention, configuration procedures are sometimes complex and error-prone. For example, securing Windows NT requires changing certain keys in the registry by editing them by hand. Complex configuration procedures must be avoided, since they lead to configuration errors, and a configuration error often introduces a security problem.

2.6 Unanticipated (ab-)use of services and feature interaction

Highly successful services are often used (and abused) in ways never imagined by the designers of the service. Hence, the designers failed to provide safeguards for these abuses.

A typical example is e-mail. The Internet e-mail system, based on SMTP, was designed to provide a simple electronic messaging service for a relatively limited group of people. The unforeseen success of TCP/IP and the Internet has made SMTP a standard for a worldwide electronic mail system. Since sending e-mail is typically much cheaper than sending paper mail, advertisers have been abusing the e-mail system since many years, sending out advertisements to millions of addressees at once. Because the designers did not anticipate the enormous success of their protocol, they did not think of safeguards for protecting against such spam e-mail.

A special case of unanticipated abuse is *feature interaction.* As more and more features are added to a software product, they start interacting in unforeseen and insecure ways. An example is the telephone network, where the introduction of new services, like call-forwarding, conference calls and ringback have led to numerous security breaches ([1]).

2.7 Non-atomic check and use

A typical scenario in a security relevant part of a program is: check if some condition is ok, and if it is, perform some action. Often attacks are based on invalidating the condition between the check and the action.

A typical example is a so-called *race condition.* For example, a program checks to see if a certain filename in the temporary directory is

available (i.e. no file with that name exists already), and if it does not exist, it opens a file with that name and starts writing information to it. An attacker can try to create a link with that specific name to a file he wants to alter between the check of existence and the actual opening of the file. As a consequence, the attacker causes the program to inadvertently add information to an existing file, where the program tried to enforce that it was really opening a new file.

A second, very simple example is simply typing commands at an unattended terminal: the operating system only checks the identity of the user at login-time, and from that moment on assumes that all commands from that terminal come from the authenticated user. A similar problem occurs with session hijacking of telnet sessions over the Internet.

2.8 Programming bugs

Finally, ordinary programming bugs, i.e. flawed algorithmic logic in security sensitive software, are much harder to detect during testing than bugs in the functionality of the software. Security related bugs only show up in the presence of malicious adversaries and hence can not be detected using automatic testing procedures. Moreover, the inherent complexity of cryptographic algorithms and other security related code makes it very hard to understand all relevant details and unfortunately wrong assumptions or small programming errors often introduce big security holes. Several famous examples of this problem exist. First, a weakness in the random generator of Netscape 1.1 where random numbers where based on the current time (which is not random at all !), made it possible to break the keys used in secure connections within seconds (see: [9]). Another example is known as the Java DNS bug. Here (see: [3, 6]), an error in the algorithm used to check whether two hosts are equal provided applets with the opportunity to connect to every computer on the Internet, which was not conforming to the rules of the restricted applet execution environment.

3. Security guidelines for developers

Many of the example software security weaknesses discussed in the previous section could have been avoided if the designers and implementors of the software had been more security-conscious during their design and programming. We feel it is very important for a software engineer to keep a number of security-related design guidelines in the back of his head at all times during the development of a software system. Every design or implementation decision should be verified against these "security rules of thumb". A good set of such guidelines is given below.

It is interesting to note that these guidelines still overlap significantly with the guidelines given in the 25 year old classic paper by Saltzer and Schroeder ([10]).

1 *Defensive programming.* Treat any input your software gets from outside as potentially hostile.

2 *Secure defaults.* While it may be a good idea to make security-related parts of your program configurable, you should realize that many users will use the default configuration without thinking too much about it. Hence, the default configuration should be secure.

   Also, many security checks can be implemented in two ways: deny by default and allow access in selected cases, or allow by default and deny access in selected cases. It should be clear that the first approach is preferable.

3 *Use secure languages where possible.* From a security point of view, a garbage-collected language (like Java) is to be preferred over a language relying on manual memory management (like C or C++). In particular, a type safe language significantly reduces the number of potential security weaknesses in software.

4 *Security-oriented testing.* Software engineers should realize that testing for security is fundamentally different from testing functionality. Testing for security is a creative form of testing, where the testers have to come up with possible attack scenarios.

5 *Economy of mechanism.* Security mechanisms should be as simple as possible (but not any simpler than that). A simple mechanism is easy to understand, easy to verify, and easy to apply.

6 *Need to know principle.* If it is possible to give different parts of your software different privileges (as is possible in Java for example), make sure that you give each part the minimal amount of privileges necessary. This leads to better containment of security breaches.

   As another instance of this rule: make sure your software can run with the minimal amount of privileges from the OS it is running on. Software written for Windows 9X for example, typically assumes having full access to the entire file system, making it difficult to port this software to the more secure NT family of operating systems.

7 *No security decisions by end users.* End users typically have little or no expertise in security, and asking them to do security relevant

configuration easily leads to configuration errors. Also, attackers might try to convince end users to change their configuration to a nonsecure state through social engineering techniques.

4. Related Work

An influential paper surveying and categorizing software vulnerabilities is the paper by Landwehr et al. ([7]). However, this paper is largely focused on system software vulnerabilities, whereas our paper mainly targets application software. A number of books ([1, 6]) give many examples of vulnerabilities, but without an attempt at classification. Also many websites publish lists of software vulnerabilities of varying quality. A column by McGraw and Viega on the IBM DeveloperWorks website is of very high quality ([12]). Finally, our security guidelines were heavily influenced by the seminal paper by Saltzer and Schroeder ([10]).

5. Conclusion

A classification of the most common software vulnerabilities (with many examples) was presented. This classification shows that many software vulnerabilities can be avoided by keeping in mind a number of simple security-related guidelines during design and development of software.

References

[1] Anderson, Ross (2001) *Security Engineering. A Guide to Building Dependable Distributed Systems.* Wiley and Sons publishers.

[2] Paul F. Bartock et al., Guide to Securing Microsoft Windows NT Networks, *National Security Agency*

[3] DNS based attack on Java, *http://www.cs.princeton.edu/sip/news/dns-spoof.html*

[4] Micheal Espinola Jr (Santeria Systems), The Hardening of Microsoft Windows NT, `http://www.networkcommand.com/docs/HardNT40rel1.pdf`

[5] Edward W. Felten, Dirk Balfanz, Drew Dean, and Dan S. Wallach, "Web Spoofing: An Internet Con Game", 20th National Information Systems Security Conference (Baltimore, Maryland), October, 1997.

[6] Gollmann, Dieter (2000) *Computer Security.* Wiley and Sons publishers.

[7] CE Landwehr, AR Bull, JP McDermott, WS Choi, "A Taxonomy of Computer Program Security Flaws, with Examples", ACM Computing Surveys 26, no. 3 (Sep 1994).

[8] Sheng Liang, Gilad Bracha, "Dynamic Class Loading in the Java Virtual Machine", Proceedings of the Conference on Object-oriented programming, systems, languages, and applications (OOPSLA'98), pp. 36 - 44.

[9] Netscape (In)Security Problems, *http://www.demailly.com/ dl/netscapesec/*

[10] Jerome H. Saltzer and Michael D. Schroeder. "The protection of Information in Computer Systems", in *Proceedings of the IEEE*, vol. 63 no. 9 (Mar 1975), pp. 1287–1308.

[11] SANS Institute, The ten most critical security threats, `http://www.sans.org/topten.htm`

[12] http://www.ibm.com/developerworks/security/

ESTABLISHING ACCOUNTING PRINCIPLES AS INVARIANTS OF FINANCIAL SYSTEMS

Naftaly H. Minsky*
minsky@cs.rutgers.edu
Department of Computer Science
Rutgers University
New Brunswick, NJ, 08903 USA

Abstract An enterprise that uses evolving software is susceptible to destructive and even disastrous effects caused either by inadvertent errors, or by malicious attacks by the programmers employed to maintain this software. It is my thesis that these perils of evolving software can often be tamed by ensuring that suitable overarching principles are maintained as *invariants* of the evolution of a given software system. In particular, it would be invaluable to ensure that a financial system satisfies the accounting principle of double-entry bookkeeping, throughout its evolutionary lifetime. We define a concept of *evolution-invariant*, discuss its usefulness, and show how the above mentioned accounting principles can be established as such invariants.

Keywords: perils of software evolution, evolution-invariants, law-governed interaction, accounting principles.

1. Introduction

The inevitable process of software evolution carries serious perils—particularly when the software is embedded in some critical enterprise, such as a power plant or a financial establishment, and when the software evolves in its operational context. The perils of such an evolution are due to the ease of making changes in software, combined with the ability of even a small change to cause large changes in system's behavior. An enterprise that uses an evolving software is thus susceptible to destructive, and even disastrous effects caused either by inadvertent er-

*Work supported in part by NSF grants No. CCR-9710575 and No. CCR-98-03698

rors, or by malicious attacks by the programmers employed to maintain this software.

These dangers are becoming progressively more difficult to manage as the software technology is undergoing a transition from monolithic systems, constructed according to a single overall design and managed by a single organization, into *conglomerates* of semi-autonomous, heterogeneous and independently designed subsystems, constructed with little, if any, knowledge of each other, and often managed and maintained by different organizations. Such software conglomerates, a rarity just few years ago, are becoming more common due to several factors, including: the increased use of COTS, the use of services available via the internet, and the need to support inherently conglomerate institutions, such as large global corporations.

It is my thesis that the perils of software evolution can often be tamed—although not eradicated— by ensuring that some broad principles of a given system be established as *invariants of its evolution.* For example, it could be useful to partition a system into a set of *divisions*, constructing permanent—i.e., evolution-invariant—"firewalls" between them, which will limit the effect that one division can have on the others.

Consider, in particular, a computerized financial enterprise. It has been argued by McKeeman in a paper entitled "Mechanizing Banker' Morality" (McKeeman, 1975), that certain broad principles are so critical to the safety and reliability of such enterprises, that they should not be entrusted to the evolving software running them, but that they should be *"embedded so deeply into the computer, that their violation is improbable to a degree approaching impossibility".* The critical principles cited by McKeeman are:

Principle 1 (double entry bookkeeping) *Money always flows from one account to another, but cannot appear from nowhere, or disappear into thin air.*

Principle 2 (auditing) *Financial activities can be monitored by auditors, without any explicit cooperation with (or the knowledge of) the system being examined, or its programmers.*

It is self evident that for software system to have a property that is invariant of its own evolution, the software must be subject to some higher authority, which ensures this particular property. There are two common mechanisms for establishing such an authority over software, which are very effective, but of a limited range of applicability.

The first such mechanism is the hardware (or firmware) of a computer. Perhaps the most important case of an hardware-induced invariant is

the distinction between *master mode* and *user mode*, which is the basis for the permanent firewall erected in most modern operating system between the kernel and all user code. Hardware enforcement is also what McKeeman had in mind for mechanizing his "bankers' morality."

The second common mechanism for establishing software invariants is the programming language in which a system is written. Examples of useful language-induced invariants abound. They include such things as scope rules, strong typing, and encapsulation. Another case of language induced invariant is the inability of Java applets to access the file system of the host machine.

But computer hardware and programming languages, as mechanisms for establishing invariants in software, have several limitations: First, only very few types of invariants can be built into a given machine, or even into a programming language. Second, an invariant built into the very fabric of a machine or a language tends to be rigid, and not easily adaptable to an application at hand. Finally, these mechanisms are not effective for conglomerate distributed systems, which may be written in a variety of different languages, and may run on a variety of different machines.

In this paper we employ more flexible and general means for establishing invariants of evolving systems. We start, in Section 2, by introducing an abstract concept of evolving systems that can have explicitly defined invariants. We then outline two concrete models for implementing this concept, one for monolithic software, and another for conglomerates. In Section 3 we describe a mechanism, called law-governed interaction (LGI), that can be used for establishing invariants of conglomerate systems. In Section 4 we show how LGI can be used to establish McKeeman's accounting principles as invariants of the evolution of financial systems. We conclude in Section 5.

2. On the Nature of Evolving Systems

Let us delineate first the type of evolution we have in mind here. It is not the common phenomenon of Darwinian-like evolution of software, where certain systems, such as text-editors, evolve through the independent creation of many variations of existing editors, and through "natural selection" between these variations in the market place. We limit the discussion in this paper to software embedded in some long-term enterprise—such as a medical establishment, or a financial enterprise—which *evolves in its operational context.* In other words, we are dealing here with a time-sequence of systems $\{S_i\}$, operating more or less in the same context[1], where each S_i is a variant of its predecessor.

One often views such a sequence as a single long-lived system, implicitly expecting it to behave in some predictable fashion—that is, to exhibit some invariants. But currently, there is no technical justification for this view, since there is generally nothing definite that can be stated about the structure or behavior of the future stages of such a sequence of systems. This is true even if the enterprise served by a sequence $\{S_i\}$ has an explicit policy P concerning its structure and behavior over time, and even if the enterprise employs good managerial practices and programming tools (like those discussed in (Duby et al., 1992; Murphy et al., 1995; Sefica et al., 1996)) for implementing this policy. Because such informal managerial practices are far from infallible, and the state of art of software development provides for no formal means for ensuring that any given policy is satisfied by an evolving sequence $\{S_i\}$.

To fill this gap, we must (as pointed out in the introduction) subject the time-sequence of systems $\{S_i\}$ to some kind of "higher authority," that enforces a given policy P. This would provide a degree of predictability to $\{S_i\}$, which would then deserve to be viewed as a single long-lived *evolving-system*—to be called an *e-system*, for short, and be denoted by $\overline{\mathcal{S}}$. Such concept is defined below.

Definition 1 *An e-system $\overline{\mathcal{S}}$ is a triple $\langle \mathcal{S}, \mathcal{L}, \mathcal{E} \rangle$, where*

1 *$\mathcal{S}$ is the* system, *at a given moment in time. (That is, at time t, $\mathcal{S}$ is one of the stages $\mathcal{S}_t$ of the evolving system-sequence.)*

2 *$\mathcal{L}$, called the* law *of $\overline{\mathcal{S}}$, is an explicit collection of rules about the structure of the system $\mathcal{S}$, about its process of evolution, and about the evolution of the law itself.*

3 *$\mathcal{E}$ is a mechanism that enforces the law.*

Now, if a certain property of an e-system $\overline{\mathcal{S}}$ is entailed by its law $\mathcal{L}$, then this property is an invariant of the evolution of this system, *as long as the law itself is not changed.* The evolution of the law is, therefore, a critical aspect of an e-system, and needs to be carefully regulated.

So far, we have formulated, and implemented experimentally, two concrete models for this abstract concept of e-systems: one for monolithic systems, and the other for conglomerates. They use different enforcement techniques, support different types of laws, and have different strengths and weaknesses. We now discuss briefly both these models, but then focus on the latter one.

To deal with evolving *monolithic systems*, we introduced the concept of Law-Governed Architecture (LGA) (Minsky, 1996; Minsky, 1997). An e-system under LGA[2] must be constructed and maintained within a software development environment that plays the role of $\mathcal{E}$ in the definition

above. This environment, whose current experimental implementation is called Darwin-E, maintains the law $\mathcal{L}$ of an e-system, and its code $\mathcal{S}$; and it enforces the law over the structure of $\mathcal{S}$, over the evolution of $\mathcal{S}$, and over the evolution of the law itself. The enforcement of the law over the structure of $\mathcal{S}$ is done mostly statically, incurring no run-time overhead. As currently formulated, LGA can be applied only to object-oriented systems, and the current implementation of Darwin-E is for systems written in Eiffel only. One of the disadvantages of LGA is that it requires total commitment to it, and does not lend itself to incremental deployment. This is not the case for our second mechanism.

By its very nature, a *conglomerate system* cannot be subjected to a single overarching regime that regulates its structure and evolution. First, because its sub-systems might be developed and maintained by different organizations; and second, because different components of such a system might be written in several different programming languages, and some of them may be COTS, whose source code may be unavailable. We opt, therefore, for two relaxations of the type of regime provided by LGA. First, we attempt to regulate only the interactions between components of a system, treating the components themselves as black boxes. Second, recognizing that single conglomerate system may involve many different, and sometimes unrelated, activities, we attempt to regulate different activities separately, under different laws. These two relaxations gave rise to the concept of law-governed interaction (LGI) (Minsky, 1991; Ao et al., 2001), briefly presented in the following section. In Section 4 we will show how LGI can be used to establish some of McKeeman's accounting principles as evolution-invariants of a financial system.

3. The Concept of Law-Governed Interaction (LGI)

Broadly speaking, LGI is a message-exchange mechanism that allows an *open group* of distributed agents to engage in a mode of interaction *governed* by an explicitly specified policy, called the *law* of the group. The messages thus exchanged under a given law $\mathcal{L}$ are called $\mathcal{L}$-messages, and the group of agents interacting via $\mathcal{L}$-messages is called a *community* $\mathcal{C}$, or, more specifically, an $\mathcal{L}$-community $\mathcal{C}_{\mathcal{L}}$.

By the phrase "open group" we mean (a) that the membership of this group (or, community) can change dynamically, and can be very large; and (b) that the members of a given community can be heterogeneous. In fact, we make here no assumptions about the structure and behavior of the agents[3] that are members of a given community $\mathcal{C}_{\mathcal{L}}$, which

might be software processes, written in an arbitrary languages, or human beings. All such members are treated as black boxes by LGI, which deals only with the interaction between them via $\mathcal{L}$-messages, making sure it conforms to the law of the community. (Note that members of a community are not prohibited from non-LGI communication across the Internet, or from participation in other LGI-communities.)

For each agent `x` in a given community $\mathcal{C}_{\mathcal{L}}$, LGI maintains, what is called, the *control-state* $\mathcal{CS}_x$ of this agent. These control-states, which can change dynamically, subject to law $\mathcal{L}$, enable the law to make distinctions between agents, and to be sensitive to dynamic changes in their state. The semantics of control-states for a given community is defined by its law, could represent such things as the role of an agent in this community, and privileges and tokens it carries. For example, under law $\mathcal{AC}$ to be introduced in Section 4 the term `role(bank)` in the control-state of an agent denotes that this agent has been certified as a bank, and thus would be able to provide other agents with money.

We now elaborate on several aspects of LGI, focusing on (a) its concept of law, (b) its mechanism for law enforcement, and (c) its treatment of digital certificates. Due to lack of space, we do not discuss here several important aspects of LGI, including the *interoperability* between communities, the concept of *enforced obligation*, and the treatment of *exceptions*. Nor do we discuss here the expressive power of LGI, its implementation, and its efficiency. For these issues, and for a more complete presentation of the rest of LGI, the reader is referred to (Minsky and Ungureanu, 2000; Ungureanu and Minsky, 2000; Ao et al., 2000).

3.1 The Concept of Law

Generally speaking, the law of a community $\mathcal{C}$ is defined over a certain types of events occuring at members of $\mathcal{C}$, mandating the effect that any such event should have—this mandate is called the *ruling* of the law for a given event. The events subject to laws, called *regulated events*, include (among others): the *sending* and the *arrival* of an $\mathcal{L}$-message; the *coming due of an obligation* previously imposed on a given object; and the *submission of a digital certificate* (more about the latter two kinds of events, later). The operations that can be included in the ruling of the law for a given regulated event are called *primitive operations*. They include, operations on the control-state of the agent where the event occured (called, the "home agent"); operations on messages, such as `forward` and `deliver`; and the imposition of an obligation on the home agent.

Thus, a law $\mathcal{L}$ can regulate the exchange of messages between members of an $\mathcal{L}$-community, based on the control-state of the participants; and it can mandate various side effects of the message-exchange, such as modification of the control states of the sender and/or receiver of a message, and the emission of extra messages, for monitoring purposes, say.

On The Local Nature of Laws:. Although the law $\mathcal{L}$ of a community $\mathcal{C}$ is *global* in that it governs the interaction between all members of $\mathcal{C}$, it is enforceable *locally* at each member of $\mathcal{C}$. This is due to the inherent locality of LGI laws, as follows:

- An LGI law $\mathcal{L}$ only regulates local events at individual agents,
- the ruling of $\mathcal{L}$ for an event **e** at agent **x** depends only on **e** and the local control-state $\mathcal{CS}_x$ of **x**.
- The ruling of $\mathcal{L}$ at **x** can mandate only local operations to be carried out at **x**, such as an update of $\mathcal{CS}_x$, the forwarding of a message from **x** to some other agent, and the imposition of an obligation on **x**.

The fact that the same law is enforced at all agents of a community gives LGI its necessary global scope, establishing a *common* set of ground rules for all members of $\mathcal{C}$ and providing them with the ability to trust each other, in spite of the heterogeneity of the community. And the locality of law enforcement enables LGI to scale with community size.

On the Structure and Formulation of Laws:. Abstractly speaking, the law of a community is a function that returns a *ruling* for any possible regulated event that might occur at any one of its members. The ruling returned by the law is a possibly empty sequence of primitive operations, which is to be carried out locally at the location of the event from which the ruling was derived (called the *home* of the event). (By default, an empty ruling implies that the event in question has no consequences—such an event is effectively ignored.)

Concretely, the law is defined by means of a Prolog-like program[4] L which, when presented with a goal **e**, representing a regulated-event at a given agent **x**, is evaluated in the context of the control-state of this agent, producing the list of primitive-operations representing the ruling of the law for this event. In addition to the standard types of Prolog goals, the body of a rule may contain two distinguished types of goals that have special roles to play in the interpretation of the law. These are the *sensor-goals*, which allow the law to "sense" the control-state of the

home agent, and the *do-goals* that contribute to the ruling of the law. A *sensor-goal* has the form `t@CS`, where `t` is any Prolog term. It attempts to unify `t` with each term in the control-state of the home agent. A *do-goal* has the form `do(p)`, where `p` is one of the above mentioned primitive-operations. It appends the term `p` to the ruling of the law.

3.2 The Law-Enforcement Mechanism

We start with an observation about the term "enforcement," as used here: We do not propose to coerce any agent to exchange $\mathcal{L}$-messages under any given law $\mathcal{L}$. The role of enforcement here is merely to ensure that *any exchange of $\mathcal{L}$-messages, once undertaken, conforms to law $\mathcal{L}$*. More specifically, our enforcement mechanism is designed to ensure the following properties: (a) the sending and receiving of $\mathcal{L}$-messages conforms to law $\mathcal{L}$; and (b) a message received under law $\mathcal{L}$ has been sent under the same law (i.e., it is not possible to forge $\mathcal{L}$-messages).

Since we do not compel anybody to operate under any particular law, or to use LGI, for that matter, how can we be sure that all movement of funds would be carried out under law $\mathcal{AC}$ designed for them? The answer is that an agent may be *effectively compelled* to exchange $\mathcal{L}$-messages, if he needs to use services provided only under this law, or to interact with agents operating under it. For instance, if a certain server requires payments for its services only via $\mathcal{AC}$-messages—which, as we shall see, enforces our accounting principles— then anybody needing its services would be *effectively compelled* to operate under law $\mathcal{AC}$. Conversely, if agents in the given enterprise use $\mathcal{AC}$-messages for their financial transactions, then servers would be compelled to accept such messages, if they are to be used.

Distributed Law-Enforcement:. Broadly speaking, the law $\mathcal{L}$ of community $\mathcal{C}$ is enforced by a set of trusted agents called *controllers*, that mediate the exchange of $\mathcal{L}$-messages between members of $\mathcal{C}$. Every member $\mathbf{x}$ of $\mathcal{C}$ has a controller $\mathcal{T}_x$ assigned to it ($\mathcal{T}$ here stands for "trusted agent") which maintains the control-state $\mathcal{CS}_x$ of its client $\mathbf{x}$. And all these controllers, which are logically placed between the members of $\mathcal{C}$ and the communications medium (as illustrated in Figure 1) carry the *same law* $\mathcal{L}$. Every exchange between a pair of agents $\mathbf{x}$ and $\mathbf{y}$ is thus mediated by *their* controllers $\mathcal{T}_x$ and $\mathcal{T}_y$, so that this enforcement is inherently decentralized. Although several agents can share a single controller, if such sharing is desired. (The efficiency of this mechanism, and its scalability, are discussed in (Minsky and Ungureanu, 2000).)

Controllers are *generic*, and can interpret and enforce any well formed law. A controller operates as an independent process, and it may be

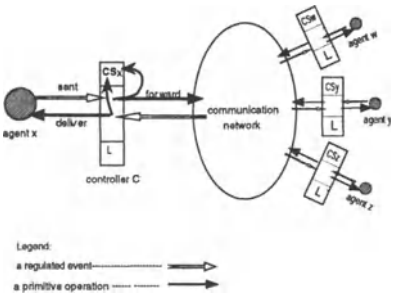


Figure 1. Enforcement of the law.

placed on any machine, anywhere in the network. We have implemented a *controller-service*, which maintains a set of active controllers. To be effective in a widely distributed enterprise, this set of controllers need to be well dispersed geographically, so that it would be possible to find controllers that are reasonably close to their prospective clients.

On the basis for trust between members of a community:. For a members of an $\mathcal{L}$-community to trust its interlocutors to observe the same law, one needs the following assurances: (a) that the exchange of $\mathcal{L}$-messages is mediated by controllers interpreting the *same law* $\mathcal{L}$; and (b) that all these controllers are *correctly implemented.* If these two conditions are satisfied, then it follows that if y receives an $\mathcal{L}$-message from some x, this message must have been sent as an $\mathcal{L}$-message; in other words, that $\mathcal{L}$-messages cannot be forged.

To ensure that a message forwarded by a controller $\mathcal{T}_x$ under law $\mathcal{L}$ would be handled by another controller $\mathcal{T}_y$ operating under the *same* law, $\mathcal{T}_x$ appends a one-way hash (Schneier, 1996) H of law $\mathcal{L}$ to the message it forwards to $\mathcal{T}_y$. $\mathcal{T}_y$ would accept this as a valid $\mathcal{L}$-message under $\mathcal{L}$ if and only if H is identical to the hash of its own law.

With respect to the correctness of the controllers, if an agent is not concerned with malicious violations, then it can trust a controller provided by our controller-naming service, or a controller provided by the operating system – just like we often trust various standard services on the Internet, such as TCP/IP protocols. When malicious violations are a concern, however, the validity of controllers and of the host on which they operate needs to be certified. In this case, the controller-naming service needs to operate as a *certification authority* for controllers. Furthermore, messages sent across the network must be digitally signed by

the sending controller, and the signature must be verified by the receiving controller, allowing the two controllers to trust each other.

3.3 The Treatment of Certificates under LGI

Under LGI, *all agents are made equal* at the time they join an $\mathcal{L}$-community. This is because the control-state of all new members is identical—and control-states provide the only means for a law to make distinctions between agents. We now explain how an agent can acquire extra privileges, thus becoming *more equal than others* (with apologies to George Orwell), by submitting appropriate certificates.

The submission by an agent x, operating under law $\mathcal{L}$, of a certificate `Cert` to its controller, has the following effect: An attempt is made to confirm that `Cert` is a valid certificate, duly signed by an authority that is acceptable to law $\mathcal{L}$, i.e., an authority that is represented by one of the `authority-clauses` in the preamble to the law (See Figure 2 for an example). If this attempt is successful[5], then a *certified-event* is triggered. This event, which is one of the *regulated-events* under LGI, has as its argument the following representation of the submitted certificate:

```
[issuer(I), subject(S), attributes(A)].
```

Here `I` and `S` are internal representations of the public-keys of the CA that issued this certificate, and of its subject, respectively; and `A` is what is being certified about the subject. Structurally, `A` is a list of `attribute(value)` terms. For example, the attributes of a certificate might be `[role(bank)]`, asserting that the subject in question is allowed to function as a bank in this community. Additional components of the attributes field include the expiration time of the certificate, the URL of the server that maintains CRLs for this type of certificates, a certificate id (used to identify it in CRLs), etc. (Currently we support SPKI format of certificates (Ellison, 1999)).

What happens when the `certified` event is triggered depends, of course, on the law. In the case of law $\mathcal{AC}$ of Figure 2, for example, the term `role(bank)` is set in the control-state of the agent that presents this certificate.

4. Establishing Accounting Principles as Laws of a Financial Enterprise

Consider now a conglomerate financial enterprise, viewed as collection of distributed agents interacting via messages. We do not presume any knowledge of, or control over, the internals of these agents, but we wish to ensure that all messages that carry money between agents comply with the principles of *double entry bookkeeping* and of *auditing*. This is done via law $\mathcal{AC}$ (for "accounting") displayed in in Figure 2. The law is

composed of a preamble, and a set of rules. Each rule is followed by a comment (in italic), which, together with the explanation below, should be understandable even for a reader not well versed in the LGI language of laws (which is based on Prolog). We start our discussion of this law with some preliminary observation about its effect, to be justified later.

Each agent operating under law $\mathcal{AC}$ would have a term `cash(c)` in its control-state, which represents the amount of cash currently held by this agent (initially zero, for all agents). Money can be moved from one agent to another by means of $\mathcal{AC}$-messages that contain the term `cash(c)`— they are called *cash-carrying messages*, and they conform to the principle of double entry accounting. Also, the movement of large amount of cash (thousand dollars or more, in this case) is being monitored, in conformance to the auditing principle. The source of money in this system are agents authorized as banks, by the CA called "admin." Such agents would have the term `role(bank)` in their control-state.

The preamble to this law has several clauses: The first is an `authority` clause, which define a certification authorities acceptable to this community, to be used for the certification of banks. Each authority clause provides the public-key of a certification authority, and assign it a local name–"admin". Second, an `initialCS` clause that defines the initial control-state of all agents in this community, which consists of the term `cash(0)` in this case. Finally, there is a `alias` clause assigning the local name "monitor" to the address `auditTrail@enterprise.com`, presumably of the audit-trail server used by this enterprise. We now examine the rules of this law in detail, showing their various effects.

The Flow of Cash:. Rules $\mathcal{R}2$ and $\mathcal{R}3$ of this law regulate the exchange of cash-carrying messages between agents. By Rule $\mathcal{R}2$, if a non-bank agent `x` sends such a message, it will be forwarded to its destination *only if* `x` itself holds sufficient amount of cash, and only after the cash of `x` is reduced by `c`. By Rule $\mathcal{R}3$, when such a message arrives at its destination `y`, it causes the cash of `y` to increase by `c`. The message itself is then delivered to `y` itself..

Thus, cash flows between non-bank agents in the system via cash-carrying messages in full compliance with the principle of double-entry bookkeeping. Note that this law is silent on the structure of cash-carrying messages (except that they need to have a cash-term) and on their effect on anything but the cash balance of the sender and the receiver. So a cash carrying message might be a payment for a previous service, a cash-carrying order, or just a grant of money to the receiver of the message. The specific form and effect of such messages is left to the agents themselves.

Preamble:

```
authority(admin,publicKey).
initialCS([cash(0)]).
alias(monitor, "auditorTrail@enterprise.com").
```

$\mathcal{R}1$
```
certified([issuer(admin),subject(Self),attributes(A)]) :-
        role(bank)@A, do(+role(bank)).
```

Claiming the role of a bank, via certificate issued by the designated CA called `admin`.

$\mathcal{R}2$
```
sent(X,M,Y) :-
        cash(C1)@M, C1>0, cash(C)@CS,
        (C>C1 | role(bank)@CS),
        do(dcr(cash(C)),C1),
        do(forward),
        audit(sent,X,M,Y).
```

An If a message carrying `C1` *dollars (in a "cash" field) is sent by an agent* `X` *with* `C` *dollars in its own cash account, this message is forwarded if* `X` *has enough cash on hand (i.e.,* `C>C1`*) or if* `X` *is a bank. In either case, the cash of* `X` *is decremented by* `C1` *and the message is forwarded. Finally, the audit-rule is invoked.*

$\mathcal{R}3$
```
arrived(X,M,Y) :-
        cash(C1)@M,
        do(incr(cash(C)),C1),
        do(deliver),
        audit(arrived,X,M,Y).
```

A message carrying `C1` *dollars (in a "cash" field) that arrives at an agent* `Y` *causes the cash possessed by* `Y` *to be incremented by* `C1`*. This message is then delivered, and the audit-rule is invoked.*

$\mathcal{R}4$
```
audit(Event,X,M,Y) :-
        Event=arrived,
        cash(C1)@M, C1>1000,
        do(forward(X,[Event,Time,X,M,Y], monitor).
```

The arrival of any message that carries more than 100 dollars is recorded, by sending to `monitor` *all relevant information.*

Figure 2. The Accounting-Law $\mathcal{AC}$

The Role of Banks:. To play the role of a bank under this law, an agent needs to present a certificate signed by by the CA we call here "admin," with the term `role(bank)` in its attributes. By Rule $\mathcal{R}1$, the presentation of such a certificate would add the term `role(bank)` to the control-state of the presenter, which we will call simply "banks" from now on.

The function of banks under this law is to provide agents with cash (without banks in this system there would be no cash to move around, because all agents start with zero balance), and to accepts deposits of cash from agents. By Rule $\mathcal{R}2$, a bank is able to inject arbitrary amount of cash into the system simply by sending it in a cash-carrying message to some agent `y`, even if its own cash-balance is negative.

Presumably, such a grant of cash to an agent `y` would generally be made in response to some kind of withdrawal request from `y`, and only if `y` has some kind of account with this bank, with sufficient balance. But such considerations are orthogonal to the principle of double-entry bookkeeping, and are, therefore, intentionally not covered by this law. Note also that agents may deposit some of their cash in a bank, via some kind of cash-carrying message to it. Thus, the balance of cash in a bank is always the sum of deposits in, minus the sum of withdrawals; and it could be negative.

Auditing:. The audit rule ($\mathcal{R}4$) is invoked by every `sent` or `arrived` events (as specified by Rules $\mathcal{R}2$ and $\mathcal{R}3$). This rule causes the time-stamped record of this event to be forwarded to a distinguished agent `monitor`—thus recording it—provided that the conditions specified in this rule are satisfied. The specific condition for recording an event built into Rule $\mathcal{R}4$, are such that only the arrival of messages that carry at least \$1000 is being monitored. But it is obviously possible to write audit rules that monitors different subsets of event, and that forwards records of such events to different monitors.

Thus, as required by the principle of auditing, somebody who can change the law $\mathcal{AC}$ can specify the type of events to be audited, and to actually carry it out, without the cooperation or knowledge of the system being audited or its programmers.

Discussion:. It is quite remarkable that it is so easy to formulate our two accounting principle by a law that consists of merely four rules. Particularly that this is not just a specification of these principles, but their implementation—because the law is actually enforced under LGI. But this formulation of these principles is oversimplified, particularly as it does not take into account possible faults of the system, such as

communication failures. It is possible to make this law fault tolerant, to a significant extend, but it takes at least twice as many rules to do so, and it is beyond the scope of this paper.

5. Conclusion

The propensity of software for rapid evolution, poses serious dangers to the integrity of any enterprise it is embedded in. We have argued in this paper that these dangers can be tamed by ensuring that some of the architectural principles of a given system are enforced, and thus established as evolution-invariants of the system.

We have used a financial enterprise as an example, showing how two important accounting principles can be established as invariants. And we believe that there are many other accounting principles, and business rules (Ehnebuske et al., 1997), that can be treated similarly.

Notes

1. We say "more or less," because the operational context of such along-lived sequence of systems is itself likely to change, even if relatively slowly.

2. In previous publications about LGA we used the term "project" for what is called here an "e-system".

3. Given the popular usages of the term "agent," it is important to point out that we do not imply by it either "intelligence" nor mobility, although neither of these is being ruled out by this model.

4. Note, however, that Prolog is incidental to this model, and can, in principle, be replaced by a different, possibly weaker, language; a restricted version of Prolog is being used here.

5. If the the certificate is found invalid then an *exception-event* is triggered.

References

Ao, X., Minsky, N., Nguyen, T., and Ungureanu, V. (2000). Law-governed communities over the internet. In *Proc. of Fourth International Conference on Coordination Models and Languages; Limassol, Cyprus; LNCS 1906*, pages 133–147.

Ao, X., Minsky, N., and Ungureanu, V. (2001). Formal treatment of certificate revocation under communal access control. In *Proc. of the 2001 IEEE Symposium on Security and Privacy, May 2001, Oakland California (to be published)*.

Duby, C. K., Meyers, S., and Reiss, S. P. (1992). CCEL: A metalanguage for C++. In *USENIX C++ Conference*.

Ehnebuske, D., McKee, B., Rouvellou, I., and Simmonds, I. (1997). Business objects and business rules. In *OOPSLA'97: Business Object Workshop*.

Ellison, C. (1999). The nature of a usable pki. *Computer Networks*, (31):823–830.

McKeeman, W. (1975). Mechanizing bankers' morality. *Computer Languages*, 1:73–82.

Minsky, N. (1991). The imposition of protocols over open distributed systems. *IEEE Transactions on Software Engineering*.

Minsky, N. (1996). Law-governed regularities in object systems; part 1: An abstract model. *Theory and Practice of Object Systems (TAPOS)*, 2(1).

Minsky, N. (1997). Toward continuously auditable systems. In *Proceedings of the First Conference on Integrity and Internal Control in Information Systems.* IFIP.

Minsky, N. and Ungureanu, V. (2000). Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. *TOSEM, ACM Transactions on Software Engineering and Methodology,* 9(3):273–305.

Murphy, G., Notkin, D., and Sullivan, K. (1995). Software reflection models: Bridging the gap between source and high level models. In *Proceedings of the Third ACM Symposium on the Foundation of Software Engineering.*

Schneier, B. (1996). *Applied Cryptography.* John Wiley and Sons.

Sefica, M., Sane, A., and Campbell, R. (1996). Monitoring complience of a software system with its high-level design model. In *Proceedings of the 18th International Conference on Software Engineering (ICSE).*

Ungureanu, V. and Minsky, N. (2000). Establishing business rules for inter-enterprise electronic commerce. In *Proc. of the 14th International Symposium on DIStributed Computing (DISC 2000); Toledo, Spain; LNCS 1914,* pages 179–193.

Integrity and Internal Control in Modern Banking Systems

Jim Jones
Havenlogic Systems Ltd.
7 Oakcroft Close, West Byfleet, Surrey
UK, KT14 6JQ
Tel.: +44-1932-355716, e-mail: jim-jones@havenlogic.fsnet.co.uk

Abstract: This paper examines the nature of controls traditionally applied in the Banking industry to batch transactions. It then looks at the reasons why batch systems introduce the notion of Risk into financial systems and why modern systems are moving towards real-time in order to overcome this Risk. By analysing real-time banking systems, including those driven by the advent of the Internet, the paper identifies the need for an entirely new family of controls and proposes an architecture of Parallel, Autonomous Audit as a framework in which this might be developed.

Key words: Batch controls; real-time controls; distributed transactions; Parallel, Autonomous Audit

1. INTRODUCTION

"You may not like your bank but you do trust them"

This comment constituted the principal theme of the SIBOS banking conference in San Francisco in October 2000 where the major banks of the world started to come to grips with the risks inherent in using the Internet as a banking medium.

The business of banking has always focussed very heavily on the need for Controls. Double Entry book-keeping itself is a form of control; hash

totals and batch totals are forms of control; transaction reconciliations are forms of control.

Despite this focus, transactions continue to go missing or end up at the wrong destination and in the worst case, banks continue to go bankrupt. Therefore, there still remains a need to enhance controls for the systems we have been familiar with for many years. However, it is clear that we are also being faced by a new generation of banking system and, if our traditional controls are not adequate for the old systems then it must be questioned to what extent they are appropriate for the new systems.

In this paper, it is proposed that these 'traditional' forms of control are no longer adequate for the new generation of banking systems which can be categorised as real-time, distributed transaction systems. We will explore the architecture of a number of these systems, including Real-Time Gross Settlement (RTGS), Continuous Linked settlement (CLS), Straight-Through Processing (STP) as well as Internet banking. We will try to establish what makes these systems different, what new challenges they pose for control regimes and some ideas for ways in which these new risks may be addressed.

2. TRADITIONAL BANKING SYSTEMS

Since banking was systematised by the Florentine bankers in the 15th century, the business of banking can be crystallised as 'the buying and selling of money'. A customer establishes a legal relationship with a bank via one or more bank accounts; in order to come into operation, these bank accounts need to be funded (hence money is paid into them) and they operate by means of the input and output of funds. In some cases, the transfer of funds is an end in itself; for example, if you go to an automated teller machine (ATM) to draw cash from your own bank account, the transfer of funds is purely a financial one. In other cases, however, the movement of funds is used to support some other type of business transaction (often referred to as an 'underlying transaction') such as the sale of purchase of stocks and shares or the repayment of a mortgage or the receipt of interest on a savings account.

These systems are transaction-orientated and are collectively referred to as Payment Systems. Indeed, there is a view that Payments are not restricted to banks but extend to other financial organisations, such as Credit Card companies (such as Visa), Building Societies, Mutual Funds etc.

In traditional systems, the processing of Payments was viewed as a routine task conducted in the back office of a bank. Transactions were submitted on paper (cheques being but one example) and each transaction was hand-written into a large Ledger book. This manual process took some time to perform, especially as any checks on the validity of the transaction, such as credit checking, were also manual. However, the banks were not unhappy about this situation, since they were able to make use of this money (a float) for several days and earn interest on it until the transaction was finalised or settled and they relinquished the money.

From the mid-1960s onwards, the introduction of computer systems began to impact some of these fundamental assumptions about the way Banking should work. For example, it was found that computers worked best by processing a group of transactions together as a batch or file, which meant that, instead of each transaction being handled individually as in the manual system, transactions were accumulated until a large enough set was available for computer processing. Although the computer processed these batches of transactions far quicker than a human being could, nevertheless, there was still a considerable period of time in which a transaction simply waited to be processed.

Batch computer systems found it easy to adopt traditional banking controls. Batches were totalled at various stages of processing and the totals checked to ensure that a transaction had not been corrupted, either accidentally or (in some cases) intentionally. At the end of the processing period – usually a day – a summary or statement would be printed and reconciled back to the initial transactions.

The prime characteristic of these systems was that, even though it might still take several days to complete the processing of a transaction (in the UK, today, it still takes three working days for most payments), the interim checks and balances enabled errors to be picked up and corrected, long before the transaction was finalised and certainly long before the customer could become aware that a problem had occurred. As a result, banks were perceived to handle the overwhelming proportion of their transactions correctly and customers felt comfortable that they did so.

3. THE AGE OF INNOCENCE VANISHES !

In the 1970s, the major banks started to realise that all was not well with their systems. Customers were beginning to demand quicker responses,

based on the fact that banks were spending lots of money on computers, and the systems began to creak. In addition, there was a sudden realisation that Payments were not merely an administrative function in the back office but, if handled incorrectly, would have severe impacts on the bank's ability to fulfill its obligations. In simple terms, if a bank tried to pay out all its obligations before it received any incoming money, it would lack the liquidity or available cash to do so and would stop functioning.

It therefore became clear that there is not only a relationship between incoming and outgoing payments but also a complex relationship between the timings of them. In 1930, the major banks of the world established the Bank of International Settlement (BIS) in Basle, Switzerland, to research the impacts of various flows of Payments and recommend best practices to safeguard these. Throughout the 1980s and 1990s, the BIS published a series of recommendations to the Banking industry. In cases where the commercial banks were slow to implement these recommendations, the Central Bank in a country would intervene and impose a much more onerous solution, typically requiring the commercial bank to submit large amounts of collateral to guarantee that transactions would complete. Of course, such collateral could not be invested and the banks lost interest. Hence, the commercial banks were incentivised to implement BIS recommendations.

4. REAL-TIME GROSS SETTLEMENT (RTGS) – THE FIRST NEW SYSTEM

Even though computers had reduced the time needed to process a Payment, it still took at least one working day before a transaction was finalised. During this period, therwas a risk that something could go wrong. It might be that the customer who had asked for the transaction went bankrupt; it might be that the bank itself lacked liquidity and went bankrupt.

Therefore BIS established a two-tier classification of Payments – High-Value and Low-Value. Although each country is free to determine the boundary between these two types, in practice, most countries come to similar conclusions. In the USA, the Federal Reserve classifies a payment over US$ 1 million as a High-Value payment; in the UK, the Bank of England classifies a payment over UK£ 1 million as High-Value.

In an RTGS, the paying customer (usually a Corporation or a Government department or, in fact, a bank) issues an instruction to his bank to make a high value payment on a certain day. Usually, high value

payments are not triggered on impulse; they are scheduled several days or even weeks in advance. Therefore, on receipt of the instruction, the bank stores it in some form of data warehouse and, during this storage period, the customer is free to cancel his instruction

On the due date, the bank extracts the instruction, checks that the customer still has funds to cover the payment and, if all is correct, sends the instruction to the national bank of its country for immediate finalisation (usually called Settlement). As soon as the instruction is settled, the beneficiary's bank is notified and for practical purposes the funds are available to the beneficiary.

The hypothesis is that a High-Value payment is too important to be left in limbo for a day before it is finalised and the BIS therefore recommended that it should be processed by computer in real-time. The idea was that even if 'real-time' in reality meant 15 minutes, this was substantially better than 8 or 10 hours or worse.

The corollary of this idea, however, was that batch controls were inadequate to safeguard such transactions. In a real-time system, clearly there is no concept of batch controls and so, this weapon was lost. Likewise, if it took from 9.15 in the morning to 9.30 to process a High-Value payment, it was of little use finding out at 5.00 in the evening that it had been processed incorrectly – the money was gone and it was difficult, if not impossible, to get it back.

What was needed was a set of controls, which would monitor each individual transaction, and alert the bank to any error in processing before the transaction was completed.

5. DISTRIBUTED PAYMENTS

In 1974, the Banking industry was rocked by the bankruptcy of Bankhaus Herstatt in Germany.

This bank was processing foreign exchange transactions and a situation arose where it was buying Japanese Yen and selling US dollars. Due to the time differences around the world, it paid out to buy the Yen early in the day while the Tokyo market was awake but had to wait for the US market later in the day before receiving the incoming payment. Unfortunately, when the US market awoke, there were no dollars and Bankhaus Herstatt could not sustain the loss and was declared bankrupt.

This risk – the risk arising from different trading hours around the world became known as Herstatt Risk and the BIS set out to address the problem. This led to a set of recommendations in 1995 and ultimately to the development of a system to solve the problem, known as Continuous Linked Settlement (CLS).

You can look upon CLS as a two-sided RTGS, described earlier. Imagine a typical high value foreign exchange transaction: a major US motor manufacturer wishes to pay for a new factory in Switzerland. So, it sends an instruction to its bank – please transfer the equivalent of US$ 500 million in Swiss francs into our account in Zurich at 10.00 next Wednesday.

The bank therefore has to sell US dollars and buy Swiss francs which it does through it foreign exchange trading department. The usual way in which this is done is for the trader to find another bank which wants to convert Swiss francs to US dollars for its customer and the two banks basically agree an exchange of value.

Again, the transactions are scheduled well in advance and so each bank sends its half of the transaction, in this case, to a data warehouse at CLS bank. On the due date, CLS Bank looks at both halves of the transactions and if they match, it settles the two transactions simultaneously. If either side of the transaction is missing, settlement does not occur and the half of the transaction which was correct is returned, with no money being lost. The same thing happens if both sides of the transaction are present but do not agree in value.

In this way, the CLS system overcomes the Herstatt problem by insisting that both sides of the transaction are available simultaneously.

CLS provides a number of control features, not all of which will be described in this paper. Some controls are the traditional end of day controls, which are the ultimate check that everything has been processed correctly. Other controls relate to the real-time aspects of the system, the mechanics of which have been described above.

However, the study of 'how foreign exchange transactions work' has led to the identification of another issue, the importance if which is just beginning to be realised.

The architecture of the CLS system is illustrated below.

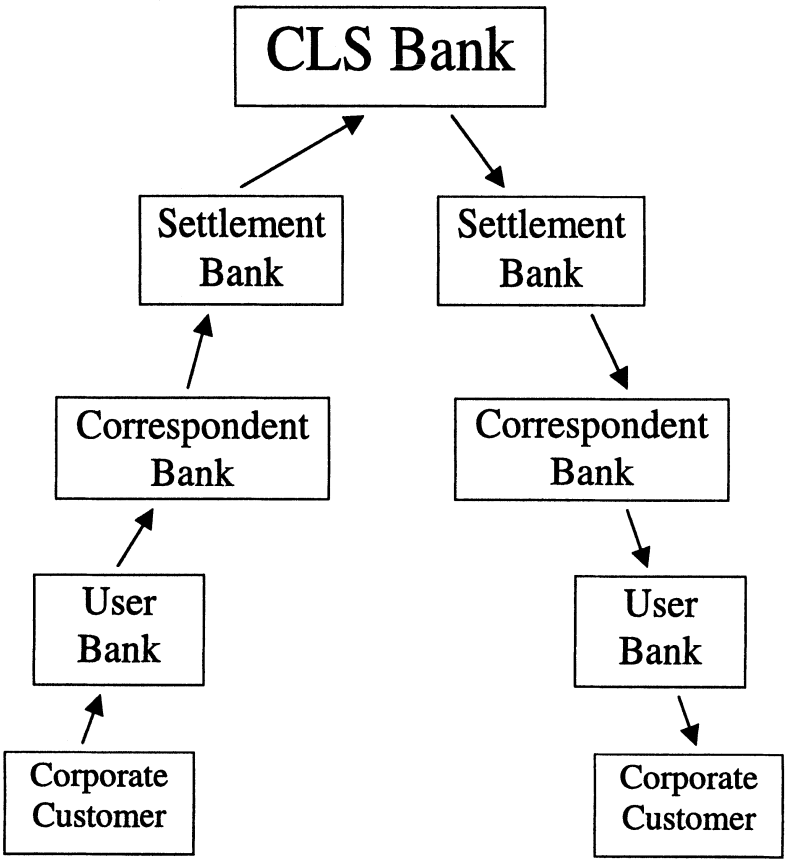


Figure 1. CLS System Architecture

What we can see is that, from the time the transaction is initiated by the customer to the time that the money arrives in the recipient's bank account, the transaction passes through a number of separate organisations, usually banks, on its route to its destination. Each bank is legally only responsible for the transaction for part of its life and there is no single organisation, which has the overall authority and mandate to monitor the integrity of the transaction for the whole of its life cycle.

The question therefore arises – who is supposed to do what if a transaction goes wrong ?

Clearly, any error should be picked up at the end of the day when CLS Bank issues its statements for the participants to carry out their reconciliations. If a participating bank identifies an error, it first has to check its own systems to see if it is the culprit, but if it can find nothing wrong, then it has to pass the problem to the next bank in the chain for it to carry out an investigation and so on. If nobody will admit to having caused the problem and they all point fingers at each other, the resolution could take several days. However, as has been pointed out earlier with RTGS, the actual money is long gone and is difficult to recover. If one of the major banks participating in this system considers a typical transaction to be US$ 500 million, the misprocessing of a single transaction is a serious event not only for the bank involved but also the other banks who could be affected by a knock-on or systemic effect.

Finally, as far as the customer is concerned, he neither knows nor cares how many banks are involved in the processing chain. He requires his own bank to fix the problem, regardless of whether they caused it or not.

6. DISTRIBUTED PAYMENTS – ANOTHER EXAMPLE

CLS is by no means the only new Banking system to adhere to the architecture described above and to suffer from the same risks. Within the Securities market, there exists a similar problem. As shown below, in the buying and selling of Securities, there are a number of organisations involved. Some of them still have manual systems and those that are automated have incompatible systems.

Historically, this has meant that once one system has processed a transaction, it prints out the results and passes the paper record to the next organisation in the chain, which re-enters the details, obviously with the risk of making an error.

To address this, the organisations involved in Securities processing are in the act of launching initiatives to eliminate these manual steps into what is becoming known as Straight-Through Processing (STP). The key idea here is that all the computers will speak a common 'Securities language' with common message formats, so that transactions can be passed automatically from one system to the next without manual intervention.

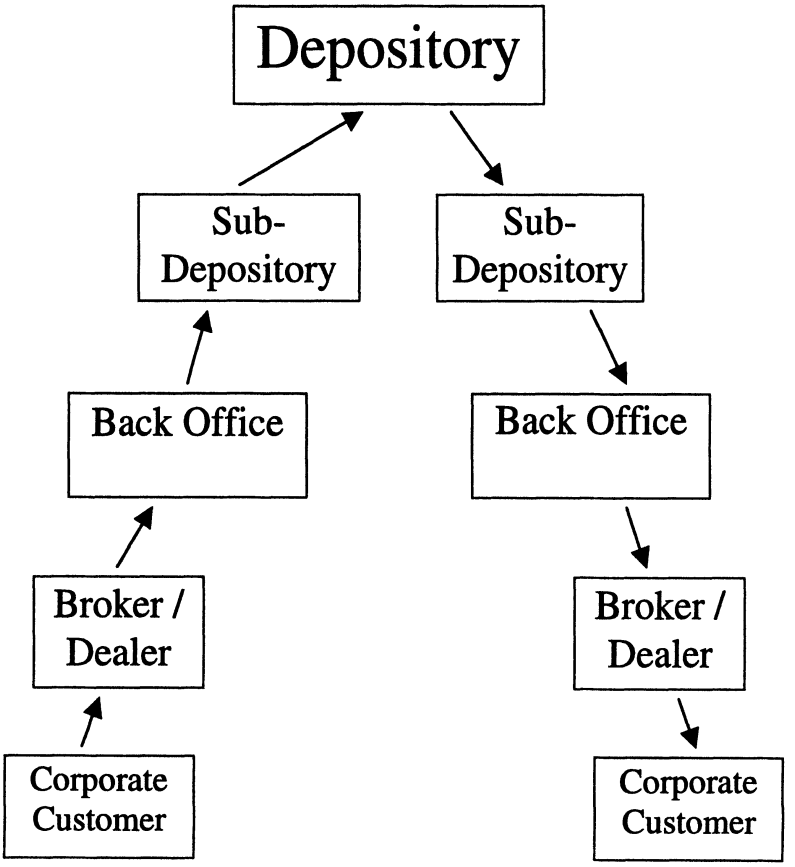


Figure 2. STP System Architecture

Taking a typical transaction, if you instruct your investment broker to buy some shares in a company for you, he looks around to find another broker whose customer wants to sell those shares and when he finds one, the two brokers agree a deal. Today, each broker handwrites on a piece of paper the details of what he thinks he has agreed over the telephone and a messenger takes this piece of paper into the back office.

In the back office, the details are keyed into a computer system and transmitted to the other broker's computer system. Even at this early stage, there is room for error. Handwritten notes are transcribed incorrectly; pieces of paper become detached and fall on the floor or they simply lie on someone's desk unnoticed.

Once past this stage, the transaction moves from the back office to a custodian (which is usually a bank which holds the customer's share certificate) up to a Depository where the transfer of ownership is registered and then all the way down the other side of the pyramid.

The other problem with this type of system is one shared with CLS. None of the organisations in the chain is responsible for the transaction for the whole of its life cycle which means that there is no clear allocation of responsibilities for fixing anything which goes wrong.

One possible solution to this problem is to have a neutral 'referee' sitting outside the system but monitoring the transaction as it passes from one organisation to the next and it has been suggested that SWIFT might play this role. The basis for this is that SWIFT has defined a set of standard message formats which handle securities transactions, SWIFT provides the network which links together the banks which process the corresponding payments and SWIFT has ambitions to use the same network to link together securities firms on a worldwide basis.

It is encouraging to see that this type of control issue (which we look upon as decentralised control) is being addressed but the SWIFT solution is, at best, a partial one. Firstly, many of the larger securities firms have already implemented their own version of STP with their own networks and it could be disruptive to replace this with SWIFT. Secondly, SWIFT can only monitor what goes into an organisation and what comes out of the other side. As such, SWIFT may identify which organisation is causing a problem but cannot tell which system within the organisation is at fault. This paper explores below an alternative approach towards handling decentralised control at a more detailed level.

The elimination of error-prone manual processes is clearly a good thing but, by itself, does not eliminate all the new problems we have identified to date:

- As processing is automated and moves nearer to real-time, traditional end of day controls no longer pick up errors early enough
- In a system made up of autonomous organisations, there is no overall control of a transaction from beginning to end
- Even worse, a Securities transaction is one which we have earlier classified as an underlying transaction; in other words, it involves the buying and selling of equities and the transfer of ownership. However, this transaction is usually accompanied by some form of payment, which is going along in parallel but separately. If the two transactions are not

synchronized, there exists the possibility that either you have received the equities but have not paid for them or, possibly worse, you have paid for them but you are not yet the legal owner !

Hence, you have two systems, each with its own risks, trying to work together, without creating additional risk (this risk is sometimes referred to as Delivery versus Payment)

Hence, this type of system incarnates all of the risks described above and needs to address them all.

7. DISTRIBUTED PAYMENTS – A FINAL EXAMPLE

In the introduction to this paper, we referred to the issue of Internet banking, which caused so much stir at Sibos.

Internet banking is in its infancy and means different things to different banks. At one extreme, the Internet is simply another wire to connect the customer to his bank. The types of transactions, which are carried out, are the same as before and can be controlled in the same way. However, some major banks have ambitions to use the Internet to provide new services. One of these – the concept of acting as an Exchange – is illustrated in figure 3.

The banks observe that the Internet is allowing new organisations to set up so-called Exchanges to bring together Buyers and Sellers in a variety of ways. One typical example is to organise electronic auctions in which the Buyer hopefully ends up with a lower price. Another is the electronic organisation of Tendering – publishing Invitations to Tender (ITTs), establishing bidding consortia, managing the bidding process and managing the contracting process.

Why should the banks want to get involved in this type of business process ? Two reasons are commonly quoted:

- In the life cycle of a total business transaction, the payment (i.e. the part the bank has traditionally handled) is only a small part. If the exchange decides to offer payment services, the bank is cut out of the business completely —it is disintermediated.
- Whoever runs an exchange function learns a huge amount about both the Buyers and Sellers and is therefore well equipped to offer additional services. Banks have always suffered from a lack of understanding of

their customers' habits. Whereas a retail store knows all about the food you buy and the clothes you wear, as a simple by-product of your purchases, and can predict what you are likely to buy in the future, a bank has to explicitly ask for this information and such requests often meet with resistance.

There are many risks involved with a bank going into these areas of its customers business and a bank has to weigh up very carefully what new legal liabilities it will incur with each new service it offers. For example, if it sets up an auction for a buyer and selects a seller, what is its legal liability if the seller disappears with the money but without delivering the goods.

It is not possible in this paper to evaluate all of the risks involved in this new and evolving type of trading, but we can see that all of the risks described earlier apply to this area:

- Transactions are moving to real-time
- A number of organisations (NOT all of them banks) are involved in the chain
- There is an underlying transaction – usually a purchase – which has to be synchronised with the Payment.

However, in addition, it is useful to highlight an additional risk (which does occur in some of the earlier scenarios but which we have chosen to hold off describing up to now). This is the risk associated with a long transaction.

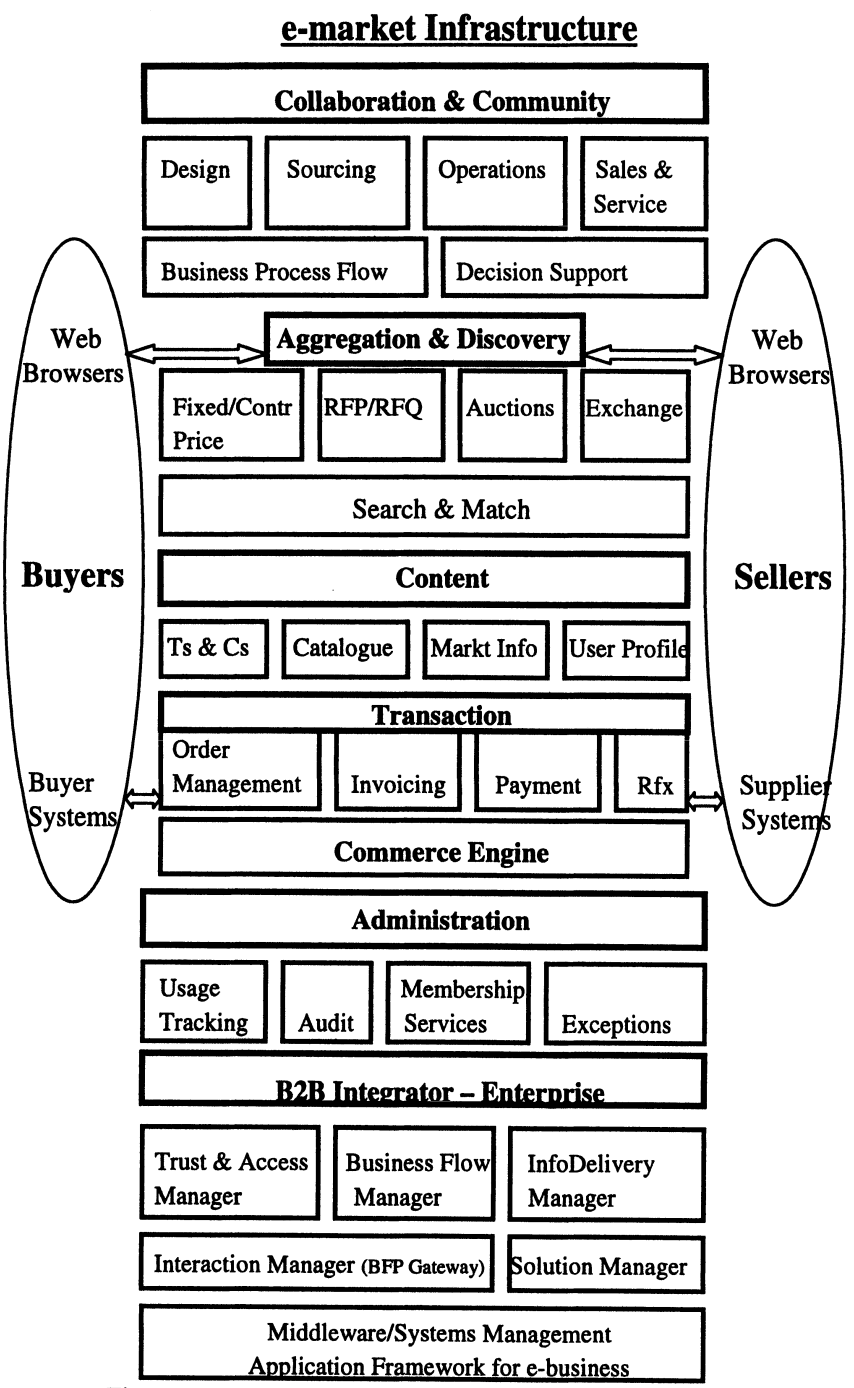


Figure 3. Electronic Exchange Architecture

8. LONG TRANSACTIONS

We have seen that banking transactions now can be processed in a few minutes up to one working day and other parts of the financial industry, such as Securities, is working towards the same position. Nevertheless, there is a type of transaction which is intrinsically different; this is a transaction which is intended to last more than one processing day.

A typical example in a bank or broker is a Futures transaction where the two parties agree to conduct a transaction at some future date. The transaction is recorded immediately but may not be completed for up to one year – and, in fact, may never be completed but cancelled after a year.

In the case of Internet banking, especially in the sector known as Business to Business, if the object of a bid is a major capital item, then payment is typically staged over a period of time and is triggered by certain conditions, such as the completion of a stage of the work or a partial delivery.

The common point is that a transaction, once recorded, lasts for a considerable duration. Of course, such transactions have always occurred and they occur in many industries other than Finance. However, the size of the transaction and therefore the size of the payment involved and its impact on the bank if it is not handled correctly, means that it is not enough simply to record a transaction on a magnetic medium and recall it on the due date: it is necessary to monitor it all the time it is dormant to ensure that if anyone (either within the bank or from the customer) tries to access and change that transaction, they have the appropriate authorisations.

9. SUMMARY OF THE NEW REQUIREMENTS

In the new generation of Banking systems, we have seen that there is a set of new risks, which requires a new set of solutions:

- A need to monitor a real-time transaction during its life cycle
- A need to establish a control regime where ownership of a transaction is distributed over a number of autonomous organisations
- A need to ensure that such systems are properly synchronised
- A need to monitor long transactions

10. TOWARDS A SOLUTIONS ARCHITECTURE

For each of the four categories of new risk listed in section 9, an individual solution either already exists or can be conceived.

10.1 Real-time transactions

There is a need of a software monitor which also runs in real-time and maintains a set of business rules which define key moments in the life of the transaction which need to be reported on and checked – with a corresponding real-time warning or other action

10.2 Distributed control

There are two types of solutions here, which could be used independently or together. Collectively, they may be referred to as Transaction-Orientated Life Cycle Audit Trails (TOLCATs) because they are founded on the idea of detaching an audit trail from a computer system and, instead, attaching it to a transaction.

The first is to take the concept of double entry book-keeping and apply it to a transaction. This could mean that you initiate a transaction on its route through the transaction chain and then you send off a duplicate of this transaction around either the same or an alternative network and you match up the two versions of the transaction at certain intermediate points in order to highlight a problem as early as possible in the cycle.

It might be objected that this doubles the processing required. However, if we recall that this type of processing applies only to High-Value payments (of which there are comparatively few each day), then this could be considered as an acceptable insurance. A more powerful objection might be – what do you do if both transactions do not arrive at a checkpoint at the same time ? Do you wait (and, if so, how long) for the second version which may never arrive ? And, if the two versions do not match, which one do you take to be correct ?

A second approach is to have the financial transaction carry its own control information, as a snail carries its house on its back. On reaching an interim destination, the transaction unpacks the control information, checks itself and if correct, carries on to the next point. This technology already

exists on the Internet and is relatively platform independent. If you are browsing the web and suddenly a special offer appears on your PC screen which you had not asked for, in effect what has happened is that some organisation has downloaded some software onto your PC and set it running to display the offer. In current jargon, such software downloads are known as Non-Persisting Cookies !!

In the sort of chained transaction life cycle we have described above in CLS and STP, the issue is that you can only have a central monitoring system if everyone in the chain agrees to play and lay their systems open to the monitoring station. If only one participant decides against this (possibly because it involves purchasing hardware and software from some vendor it does not otherwise deal with), then the centralised approach breaks down.

A TOLCAT approach therefore appears to be the only feasible alternative in which the control information is implemented in a cookie, which accompanies the transaction. Obviously, the cookie has to be implemented in such a way that it is not trapped as an error by a firewall and to this extent, all participants in the chain have to agree to this.

If, however, someone objects to the idea of a piece of software loading itself into his computer and executing itself, then the community has to accept that it will not be possible to monitor the transaction as closely as they think necessary.

If we look at the practicalities, quite apart from the Internet cookie described above which nobody violently objects to, already one software house is using this technique to monitor Electronic Bill Presentment and Payment – a financial system similar to those described above which we did not have space to analyse. Moreover, System Management tools such as IBM's Tivoli series use the same techniques to check that remote computers are working properly, whether they belong to the one organisation or not.

It may therefore be hoped that this technique will achieve de facto acceptance in Banking and Finance scenarios.

10.3 Synchronisation of systems

Synchronising a Securities trading system with a payments system is usually done by both organisations in collaboration. However, such is the critical nature of the synchronisation and the consequences if it fails that it appears justified implementing an autonomous control to check if it has been

correctly effected. Software is needed to sit on top of the interface to detect errors and arbitrate over whose software caused the problem

10.4 Long transactions

Here there is a need to monitor a transaction as though it were a piece of static data, such as a name and address and to maintain an audit trail of who accessed it under what authorisation during its extended life cycle.

11. CONCLUSIONS

From the four focus areas discussed above (RTGS, CLS, STP, Internet or e-banking), we can see that these systems are compelling us to look at controls in a number of new ways

- We have to take account of real-time transactions and the fact that traditional batch or end of period controls cannot report errors in these quickly enough.

- We have to take account of the fact that financial transactions have a life cycle which extends well beyond the boundaries of any single participant.

- We have to take account of the fact that transactions can no longer be seen as transient events which simply update master file information. They have an extended duration and need to be protected just as carefully as any other persistent data.

These characteristics lead us to conclude that they cannot be addressed solely by tinkering with existing controls. A radical new approach is required and this new approach requires a new control architecture. This architecture has several dimensions, as illustrated in figure 4.

Firstly, we stress that the architecture needs to be autonomous. This means that the control functions are independent of the applications. In handling high value payments, it is essential, in order to reduce the possibility of fraud, that the controls are managed by an entity which is independent of the application developers. It is clearly wrong to allow developers to develop their own controls.

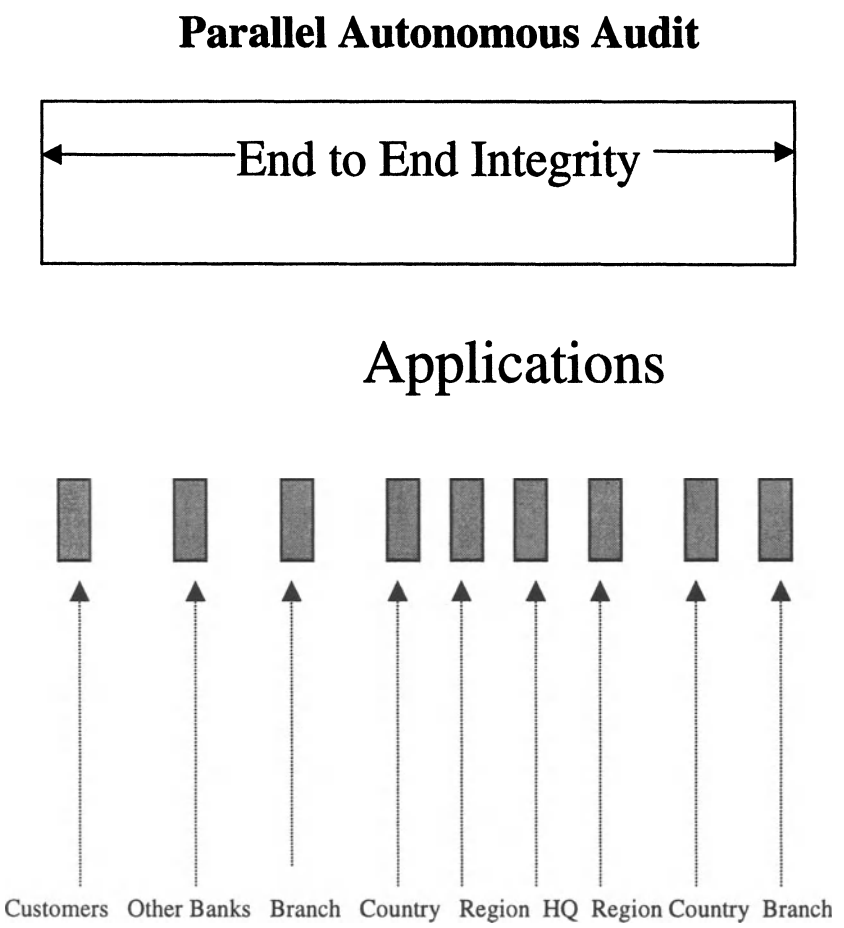


Figure 4. Parallel Autonomous Architecture

Additionally, in real-time systems, since we assume that the new controls will be developed in software, it is logical to suppose that control software may also have bugs. If such control software is in-line with the application code, it becomes part of the problem; whereas, if it is separate, then it can be turned off in the case of problems without affecting the on-going operation.

Secondly, the controls need to run parallel to the transactions they are controlling. If the transactions are occurring in real-time, the controls have to

monitor them in real-time. If an error is detected, it needs to be reported in real-time so that the appropriate correction can take place, while limiting the propagation of the error.

Thirdly, the control regime needs to take account of the total life cycle of the transaction:

- In space – as it travels through a variety of autonomous organisations
- In time – for the duration of its existence, no matter how many years this may represent.

This paper therefore proposes a radically new paradigm for Controls in the Banking and Finance Industry. However, it is true to say that the factors which influence this and the solutions proposed may be applied to many other industries. One could think of airline scheduling, process control in steel mills, car assembly lines, drug prescriptions and a host of others.

12. REFERENCES

Valdez S (1993): An Introduction to Western Financial Markets Macmillan Press

Marshall C (2001): Measuring and Managing Operational Risks in Financial Institutions John Wiley & Sons (Asia) Pte Ltd

Bank For International Settlements (1989): Report on Netting Schemes G10 Committee on Payment and Settlement Systems

Bank For International Settlements (1990): Report on Interbank Netting Schemes (Lamfalussy Report) G10 Committee on Payment and Settlement Systems

Bank For International Settlements (1992): Report on Delivery versus Payment in Securities Settlement Systems (DVP Report) G10 Committee on Payment and Settlement Systems

Bank For International Settlements (1993): Report on Central Bank Payment AND Settlement Services with Respect to Cross-Border and Multi-Currency Transactions (Noel Report) G10 Committee on Payment and Settlement Systems

Governors of the Central Banks of the European Union (1994): Minimum Common Features for Domestic Payment Systems (Padoa-Schioppa Report)

Bank For International Settlements (1995): Report on Cross-Border Securities Settlements G10 Committee on Payment and Settlement Systems

Bank For International Settlements (1996): Report on Settlement Risk in Foreign Exchange Transactions (Allsop Report) G10 Committee on Payment and Settlement Systems

Bank For International Settlements (1997): Report on Real-Time Gross Settlement Systems G10 Committee on Payment and Settlement Systems

Diversity as a Defense Strategy in Information Systems

Does Evidence from Previous Events Support Such an Approach?

Charles Bain, Donald Faatz, Amgad Fayad, Douglas Williams
The MITRE Corporation 7515 Colshire Drive McLean Va 22102 USA

Abstract: One of the challenges facing computer systems is resisting attack and compromise in a networked environment. Today's computing environment is fairly homogeneous, due to a relatively small number of operating systems and application functions running on the vast majority of computers. This environment allows attackers to focus their efforts on the few types of systems deployed. Once an exploit is found, the exploit is effective against a very large number of systems running the same software. The large number of attack methods available on hacker Web sites demonstrates the ease with which attackers can exploit this homogeneous environment. This paper examines several widespread computer attacks to understand the effect of diversity on maintaining the integrity, and hence survivability, of information systems.

Key words: Security, survivability, diversity, intrusion and fault tolerance

1. INTRODUCTION

One of the challenges facing computer systems is resisting attack and compromise in a networked environment. Today's computing environment is fairly homogeneous, due to a relatively small number of operating systems (e.g., variants of UNIX or Microsoft Windows) and application functions (e.g., networking based on TCP/IP) running on the vast majority of computers. This environment allows attackers to focus their efforts on the few types of systems deployed. Once an exploit is found, the exploit is effective against a very large number of systems running the same software. The large number of attack methods available on hacker Web sites

demonstrates the ease with which attackers can exploit this homogeneous environment.

Most systems run similar software and/or support common services. If systems were different, they might have an additional defense against attacker exploits: a vulnerability discovered in one system might not be effective in other systems if the systems are different in ways that avoid the vulnerability. Additionally, this diversity would increase the effort required to compromise systems, since each system would be a unique environment for an attacker to work against. This increase in effort could reduce the number of exploits discovered (because of the additional effort required) and perhaps decrease the attractiveness of exploiting systems.

Diversity as a defense is illustrated in attacks on systems. A typical attack exploits a system and a specific vulnerability: different systems are not (directly) affected. An attack effective against Microsoft Outlook does not affect UNIX mail applications. However, different computer systems support common services, such as the TCP/IP networking protocol used for Internet access, or common applications, such as the Netscape or Internet Explorer web browsers. Thus, the advantages gained through diversity are offset by systems with a common point of failure.

This paper examines several widespread computer attacks to understand the effect of diversity on the survivability of attacked systems. The described attacks are:

- the Morris worm, which spread by exploiting vulnerabilities in TCP/IP capabilities
- the Melissa virus, which infected using the macro capabilities of a Microsoft Word attachment to e-mail
- the LoveLetter worm, which infected using a Visual Basic script attached to e-mail
- the Denial of Service attacks against high-profile web sites from a network of compromised "slave" systems

The attack methods and effectiveness are described. In the conclusion of the paper, the role of diversity in the survival of systems is discussed.

2. THE MORRIS WORM

2.1 Introduction

On Wednesday, November 2, 1988, at 5:01:59 P.M. E.S.T. a worm was released on the Internet [20]. It was brought under control and eliminated from most machines 48-72 hours later [19, 26]. This self-propagating worm easily spread by exploitation of well-known vulnerabilities that had not been closed in the victim systems.

2.2 Technical Summary

The Morris Worm used four main methods for spreading:

- fingerd gets() buffer overflow: Only 4.3BSD VAX machines suffered from this attack [19]. SunOS did not suffer, causing a core dump, only because of different required offset on the stack [18, 21, 26]. Ultrix, for example, was not vulnerable [8].
- Sendmail DEBUG option: Mostly Berkeley derived Unixes, but also other varieties of Unix [25, 26]. SunOS binary releases had this mode [8]. DEBUG was enabled as the default for 4.2BSD, 4.3BSD and derived SunOS, while the commercial release of Ultrix did not have DEBUG enabled as a default [8].
- Trusted logins using .rhosts and /etc/hosts.equiv with rexec and rsh: This affected networking code based on BSD extensions [25]. These are inherently insecure functions.
- Passwords where /etc/passwd file was not shadowed.

Once security was breached, a bootstrap program was sent to the compromised machine, which was compiled after its transfer. This program was then executed and proceeded to copy over two object files (plus the bootstrap source code), one for the VAX BSD and one for the Sun-3 SunOS. The bootstrap program linked the appropriate file against the C library to produce an executable worm. Hence, the worm "supported" only a BSD UNIX and derived operating systems in use at the time of release. There were unused provisions in the worm code to transfer an additional 17 files [17], indicating additional targets may have been planned.

It was estimated that approximately 75 percent of the computers then attached to the Internet used some version of Unix [27], but the worm only affected code that included 4.2 or 4.3 BSD derivatives like SunOS [21]. Furthermore, the worm only propagated over TCP/IP and not UUCP, X.25,

DECNET, or BITNET [21]. The worm did not infect System V systems unless they had been modified to use Berkeley network programs like sendmail, fingerd and rexec [21].

2.3 Extent of Infection

In November 1988 it was estimated that there were approximately 60,000 computers worldwide on the Internet [13, 25], composed of over 500 unclassified national, regional and local networks [27]. The NSF estimated that there were over half a million Internet users [27] at the time.

There are no official estimates of the number of computers attacked, in part because no one organization is responsible for obtaining such information. The actual number of systems infected is impossible to determine, but it's worthwhile to examine the frequently quoted figures.

The first estimate came on Thursday, November 3, 1988, when in the late evening MIT held a press conference stating that they had suffered an estimated 10% infection rate of the 2,000 hosts belonging to MIT. The infection rate was a guess at the time and was given when the Internet was still under attack. The press extrapolated this percentage to the entire Internet and concluded that 6,000 machines, [20, 27] of the 60,000 estimated to comprise the Internet at that time, were infected.

However, not all sites have the same proportion of vulnerable machines as MIT. A Harvard University researcher who queried users over the Internet contends that a more accurate estimate would be between 1,000 and 3,000, or 2% to 5% of the computers infected [27]. Other estimates at the time ranged from 2,000 to 6,000 (3% to 10%), but when the situation stabilized, consensus among published papers centered around 2,000 to 4,000 (3% to 7%) [4, 5, 16, 25].

2.4 Remarks

One of the problems with the available information is that the extent of infection of vulnerable machines is unknown. If this information were available, it would be possible to map this proportion into the total number of Internet hosts to yield an estimate of infection that would have occurred if the Internet had been homogeneous.

Security is a tradeoff, a measurement of the resolve of the attacker and defender to commit resources to gain an advantage. In a homogeneous

computing environment, less expenditure of resources will be required to defend the system. Similarly, a lesser commitment of resource is required to attack the system as Shoch and Hupp found when developing their worm [22]. In a heterogeneous system, the reverse is true for both the defender and attacker. Hence, the issue is whether the defender or the attacker has the resolve to commit greater resources to the problem.

In the worm example, at the time of release, the attacker had only committed resources to permit the attack of a subset of BSD based systems. However, 17 additional operating systems may have been considered as targets [23]. It is arguable that if the worm author had committed the resources to the attack, the Internet would truly have been brought down.

3. THE MELISSA VIRUS

The Melissa virus (W97M.Melissa.A) was released into the Internet in March 1999. Melissa is a Microsoft Word Macro virus which uses electronic mail (e-mail) to spread itself to additional systems; however, it carries no malicious payload.

3.1 Technical Summary

The Melissa Virus spreads by attaching an infected Word document to an e-mail message. Recipients who open the attached document in Word experience two side effects:

- Word documents created after the infection are also infected with the virus.
- E-mail addresses from their address book are used to further spread the virus.

Melissa also reduced the level of security warnings displayed to Word users and modified the Windows Registry to indicate its presence so future re-infections would have no additional affects.

3.2 Extent of Infection

It is difficult to assess the overall extent of the infection. None of the sources located to date could say with any certainty how many systems were infected. Computerworld [17] reported that 80 percent of the 150 organizations that contacted Network Associates for assistance were infected. This article also reports that a single customer had 60,000 desktop

systems infected and over 500,000 e-mail copies in circulation within the company.

The Risks Digest [28] (comp.risks) noted that Microsoft blocked all outgoing e-mail to guard against propagation of the virus outside the company.

In its FAQ, [24] the Software Engineering Institute's (SEI's) Computer Emergency Response Team (CERT) noted first-hand reports from 300+ organizations that had been infected. Across these organizations, over 100,000 computers were infected. This infection spread rapidly. From first reported infection to over 100,000 infections took less than three days.

3.3 Remarks

Melissa, like the Internet worm, targeted very specific software. In the Melissa case, Microsoft Word versions 8 or 9 were the only software that could be infected. Systems that did not use this software could not be infected. Note, however, that Word version 8 was available for both the Microsoft Windows operating system and the Apple MacOS operating system so both systems could be infected.

Melissa used only the Microsoft Outlook e-mail client to propagate itself to other systems. Users of Microsoft Outlook Express, Netscape Mail, Eudora, or other e-mail clients could themselves be infected if they used MS Word, but would not automatically spread the virus. Of course, having been infected, any Word files e-mailed manually would spread the infection.

Further, while users of e-mail clients other than MS Outlook did not automatically propagate the virus, they were frequently victims of colleagues and acquaintances who did use Outlook and were flooded with e-mail sent by Melissa infections on other systems.

4. THE LOVELETTER WORM

The LoveLetter worm (VBS.LoveLetter.A) was released to the Internet in May 2000. LoveLetter is a Microsoft Visual Basic script (VBScript) worm that is delivered to victims as an e-mail attachment.

4.1 Technical Summary

The LoveLetter worm takes advantage of the Windows Scripting Host (WSH) capability of Microsoft Outlook. When a victim clicks on an e-mail script attachment in Outlook, Outlook invokes WSH to execute the VBScript, which infects the victim's system. Other VBS-enabled e-mail clients can also execute the worm script. The worm is restricted to Microsoft environments because the Visual Basic programming language is only available from Microsoft.

The worm performs the following actions:

- Copies of the VBScript program are stored in several folders on the C: drive. The copies ensure that the worm is restarted after a re-boot.
- The Windows registry is modified so that the worm script file is invoked each time the system is restarted. This ensures that the worm is always running.
- The Windows Registry is modified to remove keys that disable password caching and that hide shared passwords.
- The Windows Registry is modified so that starting Microsoft Internet Explorer causes the download of a password-stealing Trojan program. This program sends stolen passwords to an e-mail address at system startup and at certain other times. The Windows Registry is modified to ensure that this Trojan runs at each re-boot of the system.
- A copy of the e-mail and infected script is sent to every entry in the Microsoft Outlook address book. Recipients who open the e-mail attachment become infected, thus spreading the virus. Additionally, the volume of e-mail causes a significant increase in e-mail activity, impacting e-mail servers.
- A copy of the virus, encapsulated inside an HTML file, is sent to users who join IRC chat groups used by the victim.
- Files with certain file extensions are deleted, and a copy of the worm is stored using the name of the deleted file combined with a new extension .vbs. If a user clicks on this new (but familiar-looking) file name, the worm is re-executed.

The worm avoids casual detection by taking advantage of common Microsoft conventions:

- Installation of Microsoft Internet Explorer also installs WSH by default. This also links WSH to Outlook, such that clicking on an e-mail attachment automatically launches WSH to execute the script. While many users had installed Microsoft Internet Explorer, few realized that this installation gave Outlook the capability to execute scripts via WSH.

- A common Windows default is to suppress the display of file extensions. The e-mail attachment is named LOVE-LETTER-FOR-YOU.TXT.vbs. If file extensions are suppressed, the user sees a file name of LOVE-LETTER-FOR-YOU.TXT. Users may assume that the attachment is a text file with no associated application, and assume that it is "safe" to open the attachment. However, rather than displaying a text file, the VBScript file is executed and the victim is infected. The virus also replaces certain files on the user's hard drive with similarly named versions of the virus. Thus clicking on files with certain extensions (.jpeg or .mp3, for example) will re-launch the virus.

Copies of the virus and associated files are stored with "Windows-like" file names: MSKernal32.vbs, Win32DLL.vbs, and WIN-BUGSFIX.EXE. It is difficult for an average user to recognize whether a "Windows-sounding" file name is legitimate or not.

4.2 Extent of Infection

It is difficult to assess the overall extent of this infection. Symantec reports the worm "has wide-spread distribution, infecting millions of computers." [12] Such numbers are, at best, merely guesses. With many organizations reluctant or unwilling to provide accurate numbers of systems infected, the extent of such widespread infections will never be accurately known.

The worm causes much damage when a system is infected:
- Files are destroyed
- Passwords are stolen
- E-mail servers are clogged by copies of the worm

Additionally, it takes time to recover from infection of a host. Even uninfected users typically had to spend time dealing with the worm, either deleting the e-mail sent by infected hosts or updating anti-virus software to prevent infection.

Variants of the LoveLetter worm were easily created and re-introduced. Early variants had a different subject for the e-mail or different text in the e-mail body. (One version purported to be from Symantec Anti-Virus Research Center containing a file to protect against the worm: the file was the worm itself.) Other versions changed the processing of the worm. As of August 2000, there were 29 reported variants of the LoveLetter worm [12], and detection systems continue to report interception of the worm.

4.3 Remarks

The LoveLetter worm is written to exploit both the human "weak-link" and vulnerabilities in the Windows environment. It illustrates how easily a homogeneous environment can be exploited. The worm is written in a Microsoft-specific language, is launched (with a user action) from any e-mail application which supports Microsoft WSH, installs itself on the system using the Microsoft Registry, and spreads itself using the Microsoft Windows Address Book facility. With Microsoft products installed on the vast majority of end-user systems, it is easy to exploit Microsoft vulnerabilities (and normal capabilities) to have a significant impact on users.

However, running a non-Microsoft environment (and thus avoiding Microsoft-only programs) is not free of impact:

- The ILOVEYOU worm flooded e-mail systems. This impacted many servers, as they crashed or were taken offline for repair / protection. This impacted all e-mail users, whether running the Outlook client or not.
- The script was written in VBScript, an ActiveX scripting language. ActiveX can host many scripting languages, including Perl and TCL/TK. Scripts written in these languages have the potential to run on systems other than Microsoft. Using those languages, it would be easy to write a script that could be destructive in additional environments. One variant of the LoveLetter worm is a version written as a generic UNIX shell script.

Other mail clients, such as Netscape Communicator and Eudora, can launch VBScript attachments to e-mail on a Windows platform via WSH. Thus, even different e-mail applications are exposed to vulnerabilities in common facilities.

5. DISTRIBUTED DENIAL OF SERVICE ATTACKS

During February 2000, several high-profile Internet sites were crippled by Distributed Denial of Service (DDoS) attacks. During a 3-day period, Yahoo, Buy.com, eBay, CNN, Amazon.com, ZDNet, and others were flooded with network traffic, either crashing servers or rendering them inaccessible to users. A 16-year old youth is accused of launching the attacks from his home.

5.1 Technical Summary

The DDoS attacks were launched from a network of compromised systems running the attack software. Using this hierarchical network of "slave" and "master" systems, a single attacker is able to mount a massive attack against a victim on a scale that overwhelms it. With a large number of attacking machines, an attacker does not need to exploit vulnerabilities in the victim: the victim can be overwhelmed with an extremely large flood of valid transactions.

The tool used to execute the February DDoS attacks is reported to be the Tribal Flood Network (TFN), which runs on UNIX systems. To install TFN, the UNIX host must first be compromised by exploiting a vulnerability. Once compromised, the host is modified to install both the TFN tool and a "root kit," which helps prevent the detection of tools such as TFN. This compromised host becomes a "slave" or "master" in the attack network hierarchy, and is ready to be used in a DDoS attack.

When an attack is launched, the attacker selects a victim, generally by specifying an IP address. The master systems each direct several slave systems to begin the actual attack. Several attack methods are available, using different methods to flood the victim. The TFN tool can attack with either a UDP flood, a TCP SYN flood, an ICMP flood, or a smurf attack [3].

5.2 Extent of Infection

There are two groups of systems affected by DDoS attacks: victims and compromised hosts.

Victims are selected by the attacker. Therefore, the attacker decides the extent of an attack. If the attack is directed against a server, other systems are affected indirectly, as they cannot access the server's services. For these DDoS attacks, all victims are attached to the Internet, and are important sites on the Web. Every system connected to the Internet is a potential victim of a DDoS attack, so the potential extent of impact is enormous.

Compromised hosts are used as slave or master hosts by the attacker. The TFN tool used in the attacks runs on UNIX systems, primarily Sun Solaris and Linux. Recently, some DDoS programs have been ported to the Windows environment, but UNIX machines are most often used for DDoS attacks. In order to mount a strong attack, a network of compromised hosts is created. These networks are often large. The February DDoS attack

network is reported to consist of at least 50 computers. In a report on the trinoo DDoS tool [23], some attack networks consisted of 227, 888, and 10549 compromised hosts.

It appears to be easy to create a network of compromised systems. Attackers can scan systems connected to the Internet, obtaining information about the operating system level in use. With this information, the attacker selects a tool that can compromise the system. Many of these tools are collected into toolkits and are readily available on the Web. These toolkits effectively "automate" the process of finding and compromising systems.

The I-CAT Metabase [10] has been collecting profile statistics about vulnerabilities reported by CVE. These vulnerabilities are categorized by the targeted operating system. For 1999, 286 new vulnerabilities showed the following distribution of target systems:

- UNIX 51%
- Windows 95 family 23%
- Windows NT family 37%

For the year 2000, the distribution of vulnerabilities is similar. There is a continuous stream of new vulnerabilities found and available for compromising the systems in use today.

5.3 Remarks

DDoS attacks overwhelm the victim by sending far more IP transactions at one time than it can handle. Although other attack methods may crash the victim by exploiting flaws in the software, it is equally effective to overwhelm the victim with a massive amount of legitimate traffic, leaving the victim unable to process other requests.

There are few effective methods for dealing with these attacks. If the victim is disconnected from the network in order to protect it, the attacker has succeeded in removing the victim from normal service. In some cases, the attack network traffic can be re-directed by changes in up-stream components such as routers, but it takes time to determine the source of the attack and implement the configuration changes.

The same TCP/IP connectivity that enables the Web has become a common point of vulnerability for attackers to exploit. Even entirely different systems are vulnerable to attacks on the common components. Each implementation of the common facility has potential vulnerabilities

that can be explored with a common set of approaches and tools. Additionally, entirely different implementations of a common facility will fail when the attack exploits the normal functions of the common facility.

6. DISCUSSION

At best, the attacks presented here are "anecdotal" evidence that diversity improves survivability. This is because many of the attacks are targeted for one system: the non-targeted systems are not affected. The available data on incidents is not complete enough to form the basis of a conclusion. However, there is no evidence in any of the examples presented that suggests diversity reduces survivability.

DesWarte *et al.* [6] describe many different examples of the use of diversity in industrial software engineering and in business practices. For example:

- Airbus A-300/600 digital fly-by-wire system is run by two classes of computers with different microprocessors designed independently and provided by different vendors.
- Boeing uses two different compilers to compile separate instances of the fly-by-wire software on the 777 aircraft.
- Separation of duties, a common business practice to prevent/deter fraud is a form of diversity.
- Software testing uses multiple approaches (e.g., code reviews, functional tests, code coverage testing) because each approach is likely to find different types of problems. Therefore, collectively these approaches produce higher software quality.

Essentially, they argue that diversity must be good since it is used in many different ways to provide security and reliability.

Most of the Morris worm's chroniclers took no position on the issue of the advantages or disadvantages of a homogeneous versus heterogeneous networking environment with respect to information system survivability. However, Eichen and Rochlis did make the point at the time that [8]: "Diversity is good. Though the virus picked on the most widespread operating system used on the Internet and on the two most popular machine types, most of the machines on the network were never in danger. A wider variety of implementations are probably good, not bad. There is a direct analogy with biological genetic diversity to be made."

The examples do support a need to better understand the role diversity plays in survivability and defense. For example, it was noted that the Morris Worm could only propagate over TCP/IP connections. Potentially vulnerable systems (those running BSD 4.2 or 4.3 UNIX derivatives) were not affected if they were connected via UUCP or X.25. In the more recent Melissa case, network connection protocol was not considered because all systems used TCP/IP. Connection protocol diversity has been draatically reduced in recent years with the arrival of TCP/IP for virtually all commonly used hardware and software.

Along these lines, the recent denial of service attacks against Yahoo and other sites suggest that use of a single common communication protocol makes everyone vulnerable to the same attacks. Hence, diversity, like other protection mechanisms is likely to require layering or "Diversity in Depth" to provide good protection. Diverse service implementations that rely on a common communication mechanism will not survive attacks on the shared mechanism. This is another example of common mode failures interfering with planned diversity. In other words, a homogeneous information system consists of a logical single point of failure.

Another question that needs consideration is the level of diversity required to derive significant benefit. In the examples presented, the level of diversity was relatively low, a few different implementations to perhaps tens of implementations in the case of UNIX variants. Is this enough to "raise the bar" for a determined adversary or is diversity on a large scale necessary (as in hundreds or thousands of variants)? For example, automatic software mutation as described in Michael *et al.* [15] can make each running copy of a program unique. Other approaches to building diverse computer systems [9] could be effective for attack techniques known today (e.g., the buffer overflow) and suggest other methods for protecting system data (e.g., unique, changeable signatures for files.)

However, it is unknown at this time whether these methods are effective or practical in creating an environment of diversity. Some research indicates that it may be difficult to "create" diversity by modification of software:

- The authors in Michael *et al.* encountered several problems with the software mutation approach. They report "... doubts as to whether source-code mutation is a viable way to create software diversity." Additionally, their work with abstract interpretation of a program resulted in too many constraints to be processed by the system. This lead them to believe that only random constraints (i.e., a subset of the

total number of constraints) could be effectively monitored to detect software modification [15].

- In an e-mail note to the BUGTRAQ list, Crispin Cowan of the Oregon Graduate Institute states "... we investigated the approach of using diversity to resist attack, and found it to be VERY limited in effectiveness" [1]. This was because the things that must be preserved (for a program to work properly) and those that must be changed (to ward off an attack) are largely unknown. Their research efforts turned to "restrictions," which essentially wrap additional checking around critical components [2].

One thing that seems obvious from both the Morris worm and Melissa discussions is that the authors could have made these attacks capable of handling more systems. In the Melissa case, using the Mail Applications Programming Interface (MAPI) instead of spawning Outlook would likely have enabled Netscape and Eudora e-mail clients to spread the infection automatically. The Morris worm had capabilities that were not used that could have supported additional UNIX platforms. Additionally, attack kits combine several attack tools and provide easy selection of the tool which can exploit the target system vulnerabilities. Hence, a determined adversary might easily defeat small-scale diversity.

It may also be the case that extensive diversity creates additional areas of exploitation. Where implementations are different, the possibility exists for new errors caused by these differences. These errors could potentially be exploited r attack, resulting in vulnerabilities that did not exist in the homogeneous environment[1].

7. CONCLUSION

The attacks studied here illustrate that diversity in computer systems appears to be desirable: the specific systems / facilities not targeted do survive an attack. In an environment with different implementations of services, this diversity helps create forms of redundancy: some implementations continue to operate when others have failed. This creates a greater level of system availability and reliability, and as a result there is an improved confidence in the integrity of the information system.

[1] This opportunity to exploit inconsistencies among multiple implementations was suggested by Julie Bouchard of Sandia National Labs during planning for the DARPA Information Assurance program's RT 0001 exercise.

However, the computer industry is moving toward a more homogeneous environment. There has been a steady consolidation of operating systems and applications, despite a tremendous growth in the number of users. Additionally, there is increased popularity in common services based on standards (e.g., TCP/IP) or open (i.e., shared) software (e.g., Linux). This homogeneous environment remains highly vulnerable to attack.

It also appears that diversity may be difficult to "create" in a homogeneous environment. Several researchers have reported complex problems in attempting to modify software to introduce immunity to certain attacks. The large number of attacks discovered each year implies that new "diversity" methods will constantly need to be created for effective defense. It remains to be seen whether practical methods will be created to provide sufficient diversity to help defend against attacks.

8. BIBLIOGRAPHY

1. Cowan, Crispin, e-mail subject "Diversity (was: IIS Remote Exploit (injection code))," BUGTRAQ mailing list, June 16, 1999.
2. Cowan, Crispin and Pu, Calton, "Death, Taxes, and Imperfect Software: Surviving the Inevitable," http://www.cse.ogi.edu/DISC/projects/immunix/publications.html; presented *New Security Paradigms Workshop 1998.*
3. Criscuolo, Paul J., "Distributed Denial of Service," CIAC-2319, Department of Energy Computer Incident Advisory Capability, February 14, 2000.
4. Denning, Dorothy, *Information Warfare and Security*, Addison-Wesley, Reading, 1999.
5. Denning, Peter J., "The Internet Worm," in Peter J. Denning, ed., *Computers Under Attack: Intruders, Worms, and Viruses*, ACM Press, N.Y., 1990.
6. DesWarte, Kanoun, and Laprie, "Diversity against Accidental and Deliberate Faults," Computer Security, Dependability, & Assurance: From Needs to Solutions, IEEE Press, 1998.
7. Dittrich, David, "The DoS Project's "trinoo" distributed denial of service attack tool," University of Washington; http://staff.washington.edu/dittrich/misc/trinoo.analysis.
8. Eichin, Mark W. and Rochlis, Jon A., "With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988," Massachusetts Institute of Technology, Cambridge, 1988.

9. Forrest, Spephanie, Somayaji, Anil, and Ackley, David H., "Building Diverse Computer Systems," *Proceedings of the 6th Workshop on Hot Topics in Operating Systems*, IEEE Computer Society Press, Los Alamitos, CA., pp. 67-72 (1997).
10. http://csrc.nist.gov/icat/
11. http://www.attrition.org/mirror/attrition/stats.html
12. http://www.symantec.com/avcenter/venc/data/vbs.loveletter.a.html
13. Kahney, Leander and Polly Sprenger , "Melissa, Spawned by Spam," Wired News, http://www.wired.com/news/news/technology/story/18819.html
14. Lottor, Mark, "Internet Growth (1981-1991)," RFC 1296, Network Working Group, January 1992.
15. Michael, C.C., Aron Bartle, John Viega, Alexandre Hulot, Natasha Jarymowyzc, J. R. Mills, Brian Sohr, Brad Arkin, "Two Systems for Automatic Software Diversification," DISCEX, 2000.
16. Montz, Lynn B., "The Worm Case: From Indictment to Verdict," in Peter J. Denning, ed., *Computers Under Attack: Intruders, Worms, and Viruses*, ACM Press, N.Y., 1990.
17. Ohlson, Kathleen, "Melissa: The Day After," Computerworld Online News, 30 March 1999.
18. Page, Bob, "A Report on the Internet Worm," Computer Science Department, University of Lowell, November 7, 1988.
19. Reynolds, Joyce K., "The Helminthiasis of the Internet," RFC 1135, Network Working Group, December 1989.
20. Rochlis, Jon A. and Eichin, Mark W., "With Microscope and Tweezers: The Worm from MIT's Perspective," in Peter J. Denning, ed., *Computers Under Attack: Intruders, Worms, and Viruses*, ACM Press, N.Y., 1990.
21. Seely, Don, "A Tour of the Worm," Department of Computer Science, University of Utah, n.d.
22. Shoch, John F. and Hupp, Jon A., "The 'Worm' Programs - Early Experience with a Distributed Computation," in Peter J. Denning, ed., *Computers Under Attack: Intruders, Worms, and Viruses*, ACM Press, N.Y., 1990.
23. Slade, Rob, "Melissa Macro Virus," The Risks Digest Vol 20, Issue 26, 1 April 1999.
24. Software Engineering Institute (SEI) Computer Emergency Response Team (CERT), "Melissa FAQ," http://www.cert.org/tech_tips/Melissa_FAQ.html
25. Spafford, Eugene H., "The Internet Worm Incident," Technical Report CSD-TR-933, Department of Computer Sciences, Purdue University, West Lafayette, September 19, 1991.

26. Spafford, Eugene H., "The Internet Worm Program: An Analysis," Purdue Technical Report CSD-TR-823, Department of Computer Sciences, Purdue University, West Lafayette, November 29, 1988, revised December 8, 1988.
27. United States General Accounting Office, "Computer Security: Virus Highlights Need for Improved Internet Management," GAO/IMTEC-89-57, United States General Accounting Office, June 1989.
28. Woods, Lloyd, The Risks Digest Vol 20, Issue 26, 1 April 1999.

PART TWO. INVITED PAPERS

Data Quality: Developments and Directions

Bhavani Thuraisingham* and Eric Hughes
The MITRE Corporation, Bedford MA, USA
** On Leave at the National Science Foundation, Arlington VA, USA*

Abstract: This paper first examines various issues on data quality and provides an overview of current research in the area. Then it focuses on research at the MITRE Corporation to use annotations to manage data quality. Next some of the emerging directions in data quality including managing quality for the semantic web and the relationships between data quality and data mining will be discussed. Finally some of the directions for data quality will be provided.

Key words: Data Quality, Annotations, Semantic Web, Data Mining, Security

1. INTRODUCTION

There has been a lot of research and development on data quality for the past two decades or so. Data quality is about understanding whether data is good enough for a given use. Quality parameters will include the accuracy of the data, the timelines of the data and the precision of the data. Data quality has received increased attention after the revolution of the Internet and E-Commerce. This is because organizations now have to share data coming from all kinds of sources, and intended for various uses, and therefore it is critical that organizations have some idea as to the quality of the data. Furthermore, heterogeneous databases have to be integrated within and across organizations. This also makes data quality more important. Another reason for the increased interest in data quality is warehousing. Data warehouses are being developed by many organizations and data is brought into the warehouse from multiple sources. The sources are often inconsistent, so the data in the warehouse could appear to be useless to users.

This paper describes various aspects of data quality. In Section 2 we discuss some of the developments in data quality. Section 3 discusses some of the research at the MITRE Corporation in data quality. We discuss some of the emerging directions in data quality in Section 4. In particular, data quality for the semantic web will be discussed. Data quality and data mining will be the subject of Section 5. Security, Integrity and Data quality will be discussed in Section 6. Finally the paper is concluded in Section 7.

2. DEVELOPMENTS IN DATA QUALITY

There has been a lot of work in data quality for the last several years. Notable among the efforts is the work carried out at Sloan School of the Massachusetts Institute of Technology (MIT) [TDQM]. Some of the early work focussed on incorporating data quality primitives for data models such as the entity relationship model. Data quality primitives would include information such as the timeliness of the data, the accuracy of the data and precision of the data.

As mentioned earlier, much interest in data quality is as a result of the web. There is now so much data on the web that we need some measure as to the accuracy of the data. The accuracy depends on the source of the data. Furthermore, data gets propagated from one source to another and as a result the quality could very well deteriorate. Therefore, we need some way to determine the quality as data is passed from one to another.

Data quality is also an important factor when integrating heterogeneous data sources. An organization often cannot be responsible for the quality of the data belonging to another organization. Therefore, when organizations share data, they need to have some idea of the quality of the data.

High quality data is critical for e-commerce transactions. These transactions may involve millions of dollars and therefore of the data is of poor quality, the consequences may be disastrous. Many of the businesses that have studied data quality have found that quality problems can be directly traced to significant loss of revenue.

There are web sites for data quality related information (see, e.g., [DQ]). These have information about data quality journals, products including data cleansing tools as well as information about the various organizations that do data quality work. These organizations include government organizations as

well as commercial organizations around the world. In addition, MIT also now has a series of workshops on information quality.

3. ANNOTATIONS FOR DATA QUALITY

Now that we have provided a brief overview, in this section we will discuss briefly the research that i.e. being carried out at the MITRE Corporation on data quality (see [HUGH01]).

In this effort, we have spent about two years learning about the data quality problems and efforts of a set of government organizations that share military intelligence information. These organizations share a database system, and connect it to many other databases and tools in a complex system of systems. We have begun advising these organizations on ways to address data quality problems with these systems, based on both our experience and the techniques that have been published by the data quality research community.

In summary, we have found much from the academic community that can be used by government organizations. Data quality is a significant problem, and one of the main barriers to addressing it is the need to include both technical changes to systems and managerial changes to the processes used in the organization. Many of the data quality annotations (precision, timeliness, accuracy, etc.) that have been defined for business information systems are equally relevant to government and military systems.

We also identified a need for data quality annotations, i.e., metadata that describe qualities of individual data records, rather than only using aggregate metrics (as is the focus of several prior efforts). For example, many organizations have devised ways of estimating the average accuracy of their data sets, improving that accuracy, and benefitting from the improvement. We showed that, in addition to such aggregate measures, our customers also needed to understand the quality of individual data records.

One case where these *per item* quality annotations are important is when quality varies significantly from one record to the next. In this case, if users cannot understand the quality of presented data, they may not differentiate the good from the bad. If the data is used to make decisions (as is usually the case), then either bad data can lead to bad decisions, or data of uncertain quality can lead to less well-informed decisions. As a result, users may not trust the system as a whole, even though it contains valuable data.

We defined a methodology for adding such annotations to a system, and using them to improve data quality. We are now working with two organizations to apply this methodology, and measure its effects.

4. SEMANTIC WEB AND DATA QUALITY

In the previous sections we provided an overview of the developments in data quality as well as one approach to manage data quality. In this section e discuss data quality for the semantic web.

The semantic web is essentially about machine understandable web pages. In the article on semantic web by Berners Lee et al, [LEE01], semantic web is described to be a web that can understand and interpret web pages and manage activities for people. These activities could be maintaining appointments, giving advice, and essentially making the life of the human as easy as possible.

If the semantic web is to be effective, then we need to ensure that the data and information on the web is timely, accurate, and precise. Note that with bad data one cannot make good decisions. Therefore, the techniques being used to understand data need to be widened to understand data quality. These technologies include XML, (eXtensibe Markup Language), RDF (Resource Description Framework), and agents. There is little work reported on data quality for the semantic web. In fact, data quality is an essential ingredient in understanding the semantics of data, and the semantic web approaches should take advantage of the work that has been done to define data quality annotations and use them in database systems.

5. DATA MINING AND DATA QUALITY

As we have stated, having good data is essential for making effective decisions. Data mining is about posing queries and extracting information previously unknown from large quantities of data using pattern matching and statistics (see [THUR98]). One of the challenges in data mining is to get the data ready for mining. One has to determine where the data is, assemble the data and clean the data. This would also include removing redundant data as well as inconsistent data. Data also has to be complete. Various tools ate being developed to ensure that the data is ready for mining. However, much remains to be done.

Another relationship between data mining and data quality is to use data mining to improve the quality of the data. That is, while data quality is important for mining, we could use mining to improve the quality of the data. Data mining techniques could be used to determine the bad data as well as data that is incomplete. That is, in the same way we use mining to detect unusual occurrences and patterns, our patterns here are bad data. We need to define what bad data means and perhaps train tools like neural networks to detect bad data. This is an area that is getting much attention recently.

6. SECURITY AND DATA QUALITY

There have been many discussions on the tradeoffs between security, and integrity. For example, if we are to enforce various access control rules, then the transactions may miss the timing constraints due to the time it takes to make all the checks. As a result, the data may not be current. As a result, the quality of the data may be compromised. Another issue is that to enforce high levels of security, one may need to give out cover stories. This will also compromise data quality, as the data given out may not be accurate. In other words, there are tradeoffs between security and data quality.

There have been various studies carried out between security and integrity as well as between security sand real-time processing. WE need all aspects. That is, security, integrity, real-time processing and as well as data quality are all important. The question is how can have all of them enforced in a system. This is where quality of service comes in. We need to develop flexible policies that can deal with security, integrity, real-time processing and data quality essentially on a case by case basis. There will be cases where security will be absolutely critical and there will be cases where we cannot live with bad data. Some issues were discussed in [THUR99].

7. SUMMARY AND DIRECTIONS

This paper has provided a brief overview of the developments and directions for data quality. We discussed the need for data quality and then showed an approach to manage data quality. This approach is to use annotations. That is, quality information is defined and treated as part of the data. Next we discussed some of the emerging trends such as data quality for the semantic web and the relationship between data mining and data quality.

Because of the developments of the web and the importance of technologies such as data warehousing and data mining, the need to have good data cannot be overemphasized It is critical that for many applications we need high data quality. There is already a lot of work that has been done on data quality. However much remains to be done.

8. REFERENCES

1. [DQ] www.dataquality.com
2. [HUGH01] Rosenthal, A., et al, Data Quality in the Small: Providing Consumer Information, Information Quality Workshop (IQ 2001), November 2001.
3. [LEE01] Berners Lee, T., et. al., The Semantic Web, Scientific American, May 2001.
4. [TDQM] MIT Sloan School, MIT Total Data Quality Management Program, http://web.mit.edu/tdqm/www/.
5. [THUR98] Thuraisingham, B., Data Mining: Technologies, Techniques, Tools and Trends, CRC Press, 1998.
6. [THUR99] Thuraisingham, B., and J. Maurer, Information Survivability for Evolvable Interoperable Command and Control Systems, IEEE Transactions on Knowledge and Data Engineering, January 1999.

Developments in Electronic Payment Systems Security

EMV and CEPS

Mike Ward
Europay International, Brussels, Belgium

Abstract: This paper provides an outline of the security features for Europay's chip card applications for Credit, Debit and Electronic Purse payments. It gives an overview of the EMV chipcard specifications and the Common Electronic Purse Specifications (CEPS).

Key words: electronic payment systems, EMV, CEPS, electronic purse, Clip

1. INTRODUCTION

1.1 Background

In June 1994, Europay became the first international payment organisation to commit to chip as the replacement technology for the magnetic stripe. At that time Europay's Chip Business Case focused on four key areas: Fraud Reduction, Telecommunications Cost Reduction, Credit Risk Management, and Value-added Services such as Electronic Purse.

To develop the global standards necessary to usher in the new technology, chip, Europay initiated a joint working group with MasterCard International and Visa International, known within the industry as EMV.

EMV'96, released in July 1996, marked the completion of the final phase of the global chip card specifications which served as the framework for chip card and terminal manufacturers world-wide. The EMV specifications have now been revised as EMV2000 version 4.0.

In addition to this activity, Europay have participated in the development of the Common Electronic Purse Specifications (CEPS).

This article provides an overview of the EMV'96 specifications and of Europay's supporting public key services. It also provides a brief overview of Europay's Electronic Purse, Clip.

2. EMV

2.1 Card Payments

Debit and credit card transactions normally take place between two parties that do not know one another. This is made possible by the contractual relationship that exists between the bank that issued the card to the cardholder and the acquiring bank of the merchant. The relationship between the two banks operating from two different countries is established by membership of a payment system, which also provides the network for authorising and clearing of cross-border payment transactions, and sets the rules of membership and operation along with an often complex set of guarantees.

Payment cards carry a magnetic stripe, a hologram, one or more payment brands, and a specimen signature of the cardholder. The card is also embossed with the cardholder's name and account number as well as the expiry date of the card.

The ultimate purpose of the EMV standard is to replace the magnetic stripe on the card by an Integrated Circuit or chip, thereby making it a 'smart card' with memory and processing capabilities. This added intelligence enables better issuer risk management, including improved offline and online authentication and authorisation. The characteristics of these cards are based on the ISO/IEC 7816 series, which is a generic cross-industry standard for IC cards.

2.2 Overview of EMV Application

The EMV application specification defines the terminal and integrated circuit card (ICC) procedures necessary to effect a payment system transaction in an international interchange environment.

It describes the following processes:
- Offline Data Authentication
- Cardholder Verification
- Terminal Risk Management
- Terminal Action Analysis
- Card Action Analysis
- Online Processing and Issuer to Card Script Processing

2.2.1 Offline Data Authentication

Offline Data Authentication is the process whereby the terminal verifies by means of a digital signature the authenticity of critical card data. This process is known as an "offline CAM" (Card Authentication Method) and can either be static SDA or dynamic DDA (see later).

2.2.2 Cardholder Verification

Cardholder verification is performed to ensure that the person presenting the ICC is the person to whom the application in the card was issued. The terminal uses the data in the ICC to determine whether one of the issuer-specified cardholder verification methods (CVMs) is to be executed:
- Offline PIN processing enables the ICC to verify a plaintext or enciphered PIN presented to it by the terminal
- Online PIN processing enables issuer verification of the PIN sent by the terminal
- A (paper) signature may be required, in which case this corresponds to the 'conventional' signature verification by the terminal attendant.

2.2.3 Terminal Risk Management

Terminal risk management is performed by the terminal to protect the acquirer, issuer, and system from fraud. It provides positive issuer authorisation for high-value transactions and ensures that transactions initiated from ICCs go online periodically to protect against threats that might be undetectable in an offline environment.

Terminal risk management consists of:
- Floor limit checking
- Random transaction selection
- Velocity Checking

With Velocity Checking, the terminal can compare the difference between the ATC (Application Transaction Counter) and the Last Online ATC Register with the Lower Consecutive Offline Limit to see if this limit has been exceeded.

2.2.4 Terminal Action Analysis

Once terminal risk management and application functions related to a normal offline transaction have been completed, the terminal makes the first decision as to whether the transaction should be approved offline, declined offline, or transmitted online. If the outcome of this decision process is to proceed offline, the terminal obtains a MAC (Message Authentication Code) from the card that can be used as a transaction certificate. If the outcome of the decision is to go online, the terminal initiates a challenge and response between the card and its issuer that enables online authorisation or decline (see below).

2.2.5 Card Action Analysis

An ICC may perform its own risk management to protect the issuer from fraud or excessive credit risk. Details of card risk management algorithms within the ICC are specific to the issuer and are outside the scope of the specification, but as a result of the risk management process, an ICC may decide to complete a transaction online or offline or request a referral or reject the transaction.

2.2.6 Online Processing

Online processing is performed to ensure that the issuer can review and authorise transactions - or reject transactions that are outside acceptable limits of risk defined by the issuer, the payment system, or the acquirer.

In general, online processing is the same as today's online processing of magnetic stripe transactions. The primary exception being that with EMV the terminal obtains an Application Cryptogram (AC) from the ICC. The terminal may then choose to send this AC in an authorisation request message to the Issuer.

Although actions performed by the acquirer or issuer systems are outside the scope of EMV, for online authorisations it is generally assumed that the AC is a cryptogram generated by the card from transaction data using an issuer key stored in the card and known at the issuer authorisation system.

The issuer uses this key to authenticate the AC and thereby authenticate the card. Subsequent to card authentication, the issuer may generate a cryptogram on selected data included in the authorisation response or already known to the card. This cryptogram is sent to the terminal in the authorisation response and the ICC may use it to authenticate that the response message originated from the issuer.

2.2.7 Issuer-to-Card Script Processing

An issuer may provide command scripts to be delivered to the ICC by the terminal to perform functions that are not necessarily relevant to the current transaction but are important for the continued functioning of the application in the ICC. An example might be unblocking of an offline PIN, which might be done differently by various issuers or payment systems. Europay requires that such script processing be cryptographically secured using Secure Messaging.

2.3 EMV Offline Authentication

2.3.1 Static Data Authentication

Static data authentication is performed by the terminal using a digital signature based on public key techniques to confirm the legitimacy of critical ICC-resident static data and to detect unauthorised alteration of data after personalisation.

Static data authentication requires the existence of a certification authority, which is a highly secure cryptographic facility that 'signs' the issuer's public keys. Every terminal conforming to the specification contains the appropriate certification authority's public key(s). The relationship between the data and the cryptographic keys is shown below.

2.3.2 Dynamic Data Authentication

In the case of Dynamic Data Authentication, a three-layer public key certification scheme is used where the ICC owns a public key pair. The private key is used for signing the dynamic data, and the ICC's public key is stored on the ICC in the form of a ICC Public Key Certificate, signed by the issuer.

The corresponding issuer public key is also stored on the ICC in the form of a Issuer Public Key Certificate, signed by the Card Scheme.

2.3.3 The Europay-MasterCard Certification Authority

Europay and MasterCard have developed a Public Key Certification Service for their members for the management of the joint Europay-MasterCard public key pairs and the certification of the issuer's public keys. The Europay-MasterCard Certification Authority provides the 'top layer' of the Static and Dynamic Data Authentication schemes described above.

3. ELECTRONIC PURSE

3.1 Clip

This section provides a very brief overview of Clip, a pre-paid product from Europay based on CEPS (Common Electronic Purse Specification).

Unlike debit and credit payment applications, Clip is specifically designed to enable low value offline transactions.

Although there is a wide range of domestic pre-paid schemes in operation all over Europe, generally these schemes are incapable of inter-operating with each other and thereby lack an international dimension to transactions using pre-paid products. One of the purposes of Clip is to provide this international dimension.

Clip includes a minimum set of functionality that will be available internationally, wherever the product is used:

- Loading of electronic value in the local currency of the Load Device, a service typically provided by ATMs or by dedicated devices.
- Purchases of goods, provided by dedicated purchase devices or as a new feature to POS terminals.

Electronic value is stored as a number in a 'slot' of the ICC Electronic Purse application, one slot per currency.

3.2 CEPS Online Transactions

CEPS defines two types of online transactions:
- Load;
- Currency Exchange.

A successful load transaction results in an increase to the balance of electronic value in one of the Electronic Purse's slots. A successful Currency Exchange transaction results in an increase to the balance of electronic value in one currency slot and the appropriate decrease in another slot.

Both types of transaction involve mutual authentication between the Electronic Purse card and its issuer by means of the exchange of cryptograms computed with a double length DES key shared between the two entities.

3.3 CEPS offline Transactions

CEPS defines two types of offline transactions
- Purchase;
- Cancel Last Purchase.

The Purchase transaction enables the Electronic Purse card to make payments to a merchant PSAM (Purchase Secure Application Module) in one step or a series of incremental steps. The Cancel Last Purchase transaction enables a merchant to cancel the previous purchase transaction or, in the case of an incremental purchase, the final step thereof.

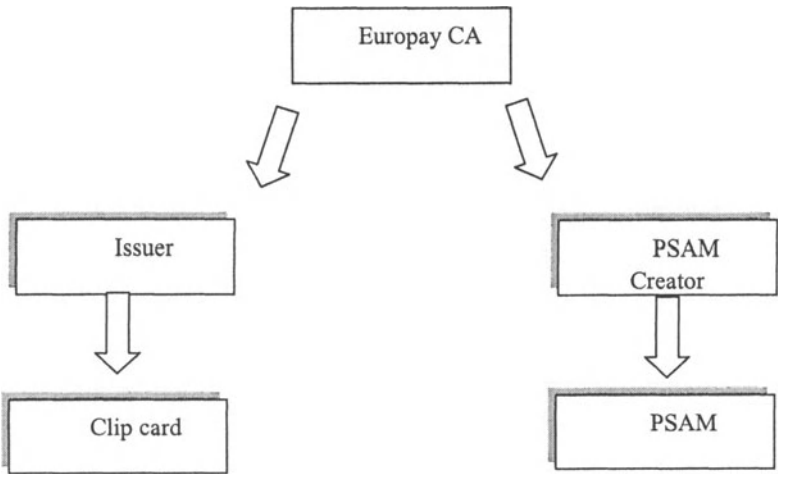
The purchase transaction requires that the Electronic Purse appropriately decrement the value in one of its slots and provide to the PSAM a cryptogram. For Europay this cryptogram is a MAC computed using a 112-bit key shared with the card issuer. The merchant can submit this 'transaction certificate' to the issuer (via the merchant acquirer) as evidence that the Electronic Purse was correctly debited.

In order that the merchant can obtain confidence that the transaction certificate is legitimate, the PSAM and the Electronic Purse perform a mutual authentication using RSA public key cryptography. A by-product of this mutual authentication is the establishing of a shared symmetric session key that can then be used for authenticating subsequent incremental purchase steps and purchase cancellations.

3.3.1 The Clip Certification Authority

In order that the Clip card be able to authenticate itself to the PSAM it is personalised with a private RSA key and public key certificates. The top-layer public key is provided by the Europay Clip Certification Authority and this public key must be installed in all PSAMs that accept Clip transactions. This situation is very similar to that of an EMV DDA card.

In order that the PSAM be able to authenticate itself to the Clip card it too is personalised with a private RSA key and public key certificates. The top-layer public key is again provided by the Europay Clip Certification Authority and this public key must be installed in all Clip cards.



4. CONCLUSIONS

This article has provided an outline of the security features for Europay's chip card applications for Credit, Debit and Electronic Purse payments. Further information can be found at Europay's web site:

http:\\www.europay.com

including the downloadable EMV specifications themselves:

EMV2000 Version 4.0: December 2000
Integrated Circuit Card Specifications for Payment Systems:

- Book 1 - Application Independent ICC to Terminal Interface Requirements
- Book 2 - Security and Key Management
- Book 3 - Application Specification
- Book 4 - Cardholder, Attendant and Acquirer Interface Requirements

PART THREE. TUTORIAL

COBIT and IT Governance

Governing Information Technology Through COBIT

Erik Guldentops CISA
Advisor, IT Governance Institute Board
Brussels, Belgium
erik.guldentops@pandora.be

Abstract: This paper[1] covers the following questions: - what is IT governance and why is it important; - whom does it concern; - what can they do about it; - what does it cover; - what questions should be asked; - how is it accomplished; - how does your organisation compare. After an introduction to COBIT, the COBIT Framework is explained and specific attention is given to the COBIT management guidelines.

Key words: IT Governance, COBIT, corporate governance, control objectives, risk assessment, management guidelines, benchmarking, performance indicators, critical success factors

1. WHAT IS IT GOVERNANCE AND WHY IS IT IMPORTANT?

As information technology has become a critical driver of business success, boards of directors have not kept pace. IT demands thorough and thoughtful board governance, yet such oversight has often been lacking

[1] This presentation is based on *Board Briefing on IT Governance*, published in 2001 by the IT Governance Institute™. It is available for complimentary download, as an open standard, from www.ITgovernance.org/resources.htm. The presentation is also based on *Control jectives for Information and related Technology (COBIT®) 3rd Edition©*, published in 1998 by the IT Governance Institute and distributed through the Information Systems Audit and Control Association® (ISACA™). All portions of COBIT, with the exception of the Audit Guidelines, are an open standard and are available for complimentary download at *www.isaca.org/cobit.htm*

because IT has been seen as an operations matter best left to management, and board members lacked interest or expertise in technology issues.

While boards have always scrutinized business strategy and strategic risks, IT has tended to be overlooked, despite the fact that it involves large investments and huge risks. Reasons include:

- The technical insight required to understand how IT enables the enterprise — and creates risks and opportunities;
- The tradition of treating IT as an entity separate to the business;
- The complexity of IT, even more apparent in the extended enterprise operating in a networked economy.

Closing the IT governance gap has become imperative as it becomes more difficult to separate an organisation's overall strategic mission from the underlying IT strategy that enables that mission to be fulfilled.

IT governance is ultimately important because expectations and reality often do not match. Boards expect management to juggle a myriad of responsibilities: deliver quality IT solutions on time and on budget, harness and exploit IT to return business value and leverage IT to increase efficiency and productivity while managing IT risks. However, boards frequently see business losses, damaged reputations or weakened competitive positions, unmet deadlines, higher-than-expected costs, lower-than-expected quality and failures of IT initiatives to deliver promised benefits.

IT governance extends the board's mission of defining strategic direction and ensuring that objectives are met, risks are managed and resources are used responsibly. Pervasive use of technology has created a critical dependency on IT that calls for a specific focus on IT governance. Such governance should ensure that an organization's IT sustains and extends its strategies and objectives.

Effective IT governance:

- Protects shareholder value;
- Makes clear that IT risks are quantified and understood;
- Directs and controls IT investment, opportunity, benefits and risks;
- Aligns IT with the business while accepting IT is a critical input to and component of the strategic plan, influencing strategic opportunities;
- Sustains current operations and prepares for the future;
- Is an integral part of a global governance structure.

2. WHOM DOES IT CONCERN?

Like most other governance activities, IT governance intensively engages both board and executive management in a cooperative manner. However, due to complexity and specialisation, this governance layer must rely heavily on the lower layers in the enterprise to provide the information needed in its decision-making and evaluation activities. To have effective IT governance in the enterprise, the lower layers need to apply the same principles of setting objectives, providing and getting direction, and providing and evaluating performance measures. As a result, good practices in IT governance need to be applied throughout the enterprise.

3. WHAT CAN THEY DO ABOUT IT?

Among the board's responsibilities are reviewing and guiding corporate strategy, setting and monitoring achievement of management's performance objectives, and ensuring the integrity of the organisation's systems.

3.1 How Should the Board Address the Challenges?

The board should drive enterprise alignment by:

- Ascertaining that IT strategy is aligned with enterprise strategy;
- Ascertaining that IT delivers against the strategy through clear expectations and measurement;
- Directing IT strategy to balance investments between supporting and growing the enterprise;
- Making considered decisions about where IT resources should be focused.

The board should direct management to *deliver measurable value* through IT by:

- Delivering on time and on budget;
- Enhancing reputation, product leadership and cost-efficiency;
- Providing customer trust and competitive time-to-market.

The board should also *measure performance* by:

- Defining and monitoring measures together with management to verify that objectives are achieved and to measure performance to eliminate surprises;

- Leveraging a system of balanced business scorecards maintained by management that form the basis for executive management compensation.

The board should *manage enterprise risk* by:

- Ascertaining that there is transparency about the significant risks to the organisation;
- Being aware that the final responsibility for risk management rests with the board;
- Being conscious that risk mitigation can generate cost-efficiencies;
- Considering that a proactive risk management approach can create competitive advantage;
- Insisting that risk management be embedded in the operation of the enterprise;
- Ascertaining that management has put processes, technology and assurance in place for information security to ensure that:
  - Business transactions can be trusted;
  - IT services are usable, can appropriately resist attacks and recover from failures;
  - Critical information is withheld from those who should not have access to it.

3.2 How Should Executive Management Address the Expectations?

The executive's focus is generally on cost-efficiency, revenue enhancement and building capabilities, all of which are enabled by information, knowledge and the IT infrastructure. Because IT is an integral part of the enterprise, and as its solutions become more and more complex (outsourcing, third-party contracts, networking, etc.), adequate governance becomes a critical factor for success. To this end, management should:

- *Embed clear accountabilities* for risk management and control over IT into the organisation;
- *Cascade strategy*, *policies* and *goals* down into the enterprise and *align the IT organisation* with the enterprise goals;
- *Provide organisational structures* to support the implementation of IT strategies and an *IT infrastructure* to facilitate the creation and sharing of business information;
- *Measure performance* by having outcome measures[3] for business value and competitive advantage that IT delivers and performance drivers to show how well IT performs;

- *Focus on core business competencies IT must support*, i.e. those that add customer value, differentiate the enterprise's products and services in the marketplace, and add value across multiple products and services over time;
- *Focus on important IT processes* that improve business value, such as change, applications and problem management. Management must become aggressive in defining these processes and their associated responsibilities;
- *Focus on core IT competencies* that usually relate to planning and overseeing the management of IT assets, risks, projects, customers and vendors;
- *Have clear external sourcing strategies*, focussing on the management of third-party contracts and associated service level and on building trust between organisations, enabling interconnectivity and information sharing.

3.3 What Does It Cover?

Fundamentally, IT governance is concerned about two things: that IT delivers value to the business and that IT risks are mitigated. The first is driven by strategic alignment of IT with the business. The second is driven by embedding accountability into the enterprise. Both need measurement, for example, by a balanced scorecard. This leads to the four main focus areas for IT governance, all driven by stakeholder value. Two of them are outcomes: value delivery and risk mitigation. Two of them are drivers: strategic alignment and performance measurement.

3.3.1 IT Strategic Alignment — "IT alignment is a journey, not a destination."

The key question is whether a firm's investment in IT is in harmony with its strategic objectives (intent, current strategy and enterprise goals) and thus building the capabilities necessary to deliver business value. This state of harmony is referred to as "alignment." It is complex, multifaceted and never completely achieved. It is about continuing to move in the right direction and being better aligned than competitors. This may not be attainable for many enterprises because enterprise goals change too quickly, but is nevertheless a worthwhile ambition because there is real concern about the value of IT investment.

Alignment of IT has been synonymous with IT strategy, i.e., does the IT strategy support the enterprise strategy? For IT governance, alignment

encompasses more than strategic integration between the (future) IT organisation and the (future) enterprise organisation. It is also about whether IT operations are aligned with the current enterprise operations. Of course, it is difficult to achieve IT alignment when enterprise units are misaligned.

3.3.2 IT Value Delivery—"IT value is in the eye of the beholder."

The basic principles of IT value are delivery on time, within budget and with the benefits that were promised. In business terms, this is often translated into: competitive advantage, elapsed time for order/service fulfillment, customer satisfaction, customer wait time, employee productivity and profitability. Several of these elements are either subjective or difficult to measure, something all stakeholders need to be aware of.

The value that IT adds to the business is a function of the degree to which the IT organisation is aligned with the business and meets the expectations of the business. The business has expectations relative to the contents of the IT deliverable:

- Fit for purpose, meeting business requirements;
- Flexibility to adopt future requirements;
- Throughput and response times;
- Ease of use, resiliency and security;
- Integrity, accuracy and currency of information.

The business also has expectations regarding the method of working:

- Time-to-market;
- Cost and time management;
- Partnering success;
- Skill set of IT staff.

To manage these expectations, IT and the business should use a common language for value which translates business and IT terminology and is based wholly on fact.

3.3.3 Performance Measurement — "In IT, if you're playing the game and not keeping score, you're just practising."

Strategy has taken on a new urgency as enterprises mobilise intangible and hidden assets to compete in an information-based global economy. Balanced scorecards translate strategy into action to achieve goals with a performance measurement system that goes beyond conventional

accounting, measuring those relationships and knowledge-based assets necessary to compete in the information age: *customer* focus, *process* efficiency and the ability to *learn* and grow. At the heart of these scorecards is management information supplied by the IT infrastructure. IT also enables and sustains solutions for the actual goals set in the financial (enterprise resource management), customer (customer relationship management), process (intranet and workflow tools) and learning (knowledge management) dimensions of the scorecard.

IT needs its own scorecard. Defining clear goals and good measures that unequivocally reflect the business impact of the IT goals is a challenge and needs to be resolved in co-operation among the different governance layers within the enterprise. The linkage between the business balanced scorecard and the IT balanced scorecard is a strong method of alignment.

3.3.4 Risk Management — "It's the IT alligators you don't see that will get you."

Enterprise risk comes in many varieties, not only financial risk. Regulators are specifically concerned about operational and systemic risk, within which technology risk and information security issues are prominent. Infrastructure protection initiatives in the US and the UK point to the utter dependence of all enterprises on IT infrastructures and the vulnerability to new technology risks. The first recommendation these initiatives make is for risk awareness of senior corporate officers.

Therefore, the board should manage enterprise risk by:

- Ascertaining that there is *transparency* about the significant risks to the organisation and clarifying the risk-taking or risk-avoidance policies of the enterprise;
- Being aware that the final *responsibility* for risk management rests with the board so, when delegating to executive management, making sure the constraints of that delegation are communicated and clearly understood;
- Being conscious that the system of internal control put in place to manage risks often has the capacity to generate *cost-efficiency;*
- Considering that a transparent and proactive risk management approach can create *competitive advantage* that can be exploited;
- Insisting that risk management is *embedded in the operation* of the enterprise, responds quickly to changing risks and reports immediately to appropriate levels of management, supported by agreed principles of escalation (what to report, when, where and how).

4. WHAT QUESTIONS SHOULD BE ASKED?

While it is not the most efficient IT governance process, asking tough questions is an effective way to get started. Of course, those responsible for governance want good answers to these questions. Then they want action. Then they need follow-up. It is essential to determine, along with the action, *who* is responsible to deliver *what* by *when*.

An extensive checklist of questions is provided in *Board Briefing on IT Governance*. The questions focus on three objectives: questions asked to discover IT issues, to find out what management is doing about them, and to self-assess the board's governance over them. For example:

To Uncover IT Issues
- How often do IT projects fail to deliver what they promised?
- Are end users satisfied with the quality of the IT service?
- Are sufficient IT resources, infrastructure and competencies available to meet strategic objectives?

To Find Out How Management Addresses the IT Issues
- How well are enterprise and IT objectives aligned?
- How is the value delivered by IT being measured?
- What strategic initiatives has executive management taken to manage IT's criticality relative to maintenance and growth of the enterprise, and are they appropriate?

To Self-assess IT Governance Practices
- Is the board regularly briefed on IT risks to which the enterprise is exposed?
- Is IT a regular item on the agenda of the board and is it addressed in a structured manner?
- Does the board articulate and communicate the business objectives for IT alignment?

5. HOW IS IT ACCOMPLISHED?

Action plans for implementing effective IT governance, from both a board and an executive management point of view, are provided in detail in *Board Briefing on IT Governance*. These plans consist of the following elements:

- *Activities* list what is done to exercise the IT governance responsibilities and the *subjects* identify those items that typically get onto an IT governance agenda.
- *Outcome measures* relate directly to the subjects of IT governance, such as the alignment of business and IT objectives, cost-efficiencies realised by IT, capabilities and competencies generated and risks and opportunities addressed.
- *Best practices* list examples of how the activities are being performed by those who have established leadership in governance of technology.
- *Critical success factors* are conditions, competencies and attitudes that are critical to being successful in the practices.
- *Performance drivers* provide indicators on *how* IT governance is achieving, as opposed to the outcome measures that measure *what* is being achieved. They often relate to the critical success factors.

The plans list IT governance activities and link a set of subjects and practices to them. Practices are classified to reflect the IT governance area(s) to which they provide the greatest contribution: value delivery, strategic alignment, risk management and/or performance (V, A, R, P). A list of critical success factors is provided in support of the practices. Finally, two sets of measures are listed: outcome measures that relate to the IT governance subjects and performance drivers that relate to how activities are performed and the associated practices and critical success factors.

6. HOW DOES YOUR ORGANISATION COMPARE?

For effective governance of IT to be implemented, organisations need to assess how well they are currently performing and be able to identify where and how improvements can be made. This applies to both the IT governance process itself and to all the processes that need to be managed within IT.

The use of maturity models greatly simplifies this task and provides a pragmatic and structured approach for measuring how well developed your processes are against a consistent and easy-to-understand scale:

0 = Non-existent. Management processes are not applied at all.
1 = Initial. Processes are ad hoc and disorganised.
2 = Repeatable. Processes follow a regular pattern.
3 = Defined. Processes are documented and communicated.
4 = Managed. Processes are monitored and measured.
5 = Optimised. Best practices are followed and automated.

(For a complete description of the various maturity levels, see *Board Briefing on IT Governance.*)

Using this technique the organisation can:

- Build a view of current practices by discussing them in workshops and comparing to example models;
- Set targets for future development by considering model descriptions higher up the scale and comparing to best practices;
- Plan projects to reach the targets by defining the specific changes required to improve management;
- Prioritise project work by identifying where the greatest impact will be made and where it is easiest to implement.

7. INTRODUCING COBIT

Control Objectives for Information and related Technology (COBIT) was initially published by the Information Systems Audit and Control Foundation™ (ISACF™) in 1996, and was followed by a second edition in 1998. The third edition, which incorporates all-new material on IT governance and Management Guidelines, was issued by the IT Governance Institute in 2000. COBIT presents an international and generally accepted IT control framework enabling organisations to implement an IT governance structure throughout the enterprise.

Since its first issuance, COBIT has been adopted in corporations and by governmental entities throughout the world.

All portions of COBIT, except the Audit Guidelines, are considered an open standard and may be downloaded on a complimentary basis from the Information Systems Audit and Control Association's web site, *www.isaca.org/cobit.htm.* The Audit Guidelines are available on a downloadable basis to ISACA members only.

8. THE COBIT FRAMEWORK

Business orientation is the main theme of COBIT. It begins from the premise that IT needs to deliver the information that the enterprise needs to achieve its objectives. It is designed to be employed as comprehensive guidance for management and business process owners. Increasingly, business practice involves the full empowerment of business process owners

so they have total responsibility for all aspects of the business process. In particular, this includes providing adequate controls. COBIT promotes a process focus and process ownership.

The COBIT Framework provides a tool for the business process owner that facilitates the discharge of this responsibility. The Framework starts from a simple and pragmatic premise:

In order to provide the information that the organisation needs to achieve its objectives, IT resources need to be managed by a set of naturally grouped processes.

The Framework continues with a set of 34 high-level Control Objectives, one for each of the IT processes, grouped into four domains:

- **Planning and Organisation**—This domain covers strategy and tactics, and concerns the identification of the way IT can best contribute to the achievement of the business objectives. Furthermore, the realisation of the strategic vision needs to be planned, communicated and managed for different perspectives. Finally, a proper organisation as well as technological infrastructure must be put in place.
- **Acquisition and Implementation**—To realise the IT strategy, IT solutions need to be identified, developed or acquired, as well as implemented and integrated into the business process. In addition, changes in and maintenance of existing systems are covered by this domain to make sure that the lifecycle is continued for these systems.
- **Delivery and Support**—This domain is concerned with the actual delivery of required services, which range from traditional operations over security and continuity aspects to training. In order to deliver services, the necessary support processes must be set up. *This domain includes the actual processing of data by application systems, often classified under application controls.*
- **Monitoring**—All IT processes need to be regularly assessed over time for their quality and compliance with control requirements. This domain thus addresses management's oversight of the organisation's control process and independent assurance provided by internal and external audit or obtained from alternative sources.

Corresponding to each of the 34 high-level control objectives is an Audit Guideline to enable the review of IT processes against COBIT's 318 recommended detailed control objectives to provide management assurance and/or advice for improvement.

The Management Guidelines further enhances and enables enterprise management to deal more effectively with the needs and requirements of IT governance. The guidelines are action-oriented and generic and provide management direction for getting the enterprise's information and related processes under control, for monitoring achievement of organisational goals, for monitoring performance within each IT process and for benchmarking organisational achievement.

COBIT also contains an Implementation Tool Set that provides lessons learned from those organisations that quickly and successfully applied COBIT in their work environments. It has two particularly useful tools—Management Awareness Diagnostic and IT Control Diagnostic—to assist in analyzing an organisation's IT control environment.

Over the next few years, the management of organisations will need to demonstrably attain increased levels of security and control. COBIT is a tool that allows managers to bridge the gap with respect to control requirements, technical issues and business risks and communicate that level of control to stakeholders. COBIT enables the development of clear policy and good practice for IT control throughout organisations, worldwide. Thus, COBIT is designed to be *the* break-through IT governance tool that helps in understanding and managing the risks and benefits associated with information and related IT.

9. THE COBIT CONTROL OBJECTIVES

For the purposes of COBIT, the following definitions are provided. "Control" is adapted from the COSO Report *(Internal Control—Integrated Framework*, Committee of Sponsoring Organisations of the Treadway Commission, 1992) and "IT Control Objective" is adapted from the SAC Report *(Systems Auditability and Control Report*, The Institute of Internal Auditors Research Foundation, 1991 and 1994).

Control is defined as the policies, procedures, practices and organisational structures designed to provide reasonable assurance that business objectives will be achieved and that undesired events will be prevented or detected and corrected.

IT Control Objective is a statement of the desired result or purpose to be achieved by implementing control procedures in a particular IT activity.

To satisfy business objectives, information needs to conform to certain criteria, which COBIT refers to as business requirements for information. In establishing the list of requirements, COBIT combines the principles embedded in existing and known reference models:

- **Quality requirements**—Quality, Cost, Delivery;
- **Fiduciary requirements** (COSO Report)—Effectiveness and Efficiency of operations; Reliability of Information; Compliance with laws and regulations;
- **Security requirements**—Confidentiality; Integrity; Availability.

Starting the analysis from the broader Quality, Fiduciary and Security requirements, seven distinct, certainly overlapping, categories were extracted. COBIT's working definitions are as follows:

- **Effectiveness** deals with information being relevant and pertinent to the business process as well as being delivered in a timely, correct, consistent and usable manner.
- **Efficiency** concerns the provision of information through the optimal (most productive and economical) use of resources.
- **Confidentiality** concerns the protection of sensitive information from unauthorised disclosure.
- **Integrity** relates to accuracy and completeness of information as well as to its validity in accordance with business values and expectations.
- **Availability** relates to information being available when required by the business process now and in the future. It also concerns the safeguarding of necessary resources and associated capabilities.
- **Compliance** deals with complying with those laws, regulations and contractual arrangements to which the business process is subject, i.e., externally imposed business criteria.
- **Reliability of Information** relates to the provision of appropriate information for management to operate the entity and for management to exercise its financial and compliance reporting responsibilities.

The IT resources identified in COBIT can be explained/defined as follows:

- **Data** are objects in their widest sense (i.e., external and internal), structured and non-structured, graphics, sound, etc.
- **Application Systems** are understood to be the sum of manual and programmed procedures.
- **Technology** covers hardware, operating systems, database management systems, networking, multimedia, etc.
- **Facilities** are all the resources to house and support information systems.
- **People** include staff skills, awareness and productivity to plan, organise, acquire, deliver, support and monitor information systems and services.

COBIT consists of high-level control objectives for each process which identify which information criteria are most important in that IT process, state which resources will usually be leveraged and provide considerations on what is important for controlling that IT process. The underlying theory for the classification of the control objectives is that there are, in essence, three levels of IT efforts when considering the management of IT resources. Starting at the bottom, there are the activities and tasks needed to achieve a measurable result. Activities have a lifecycle concept while tasks are more discrete. The lifecycle concept has typical control requirements different from discrete activities. Processes are then defined one layer up as a series of joined activities or tasks with natural (control) breaks. At the highest level, processes are naturally grouped together into domains. Their natural grouping is often confirmed as responsibility domains in an organisational structure and is in line with the management cycle or lifecycle applicable to IT processes.

Thus, the conceptual framework can be approached from three vantage points: (1) information criteria, (2) IT resources and (3) IT processes.

It is clear that all control measures will not necessarily satisfy the different business requirements for information to the same degree.

- **Primary** is the degree to which the defined control objective directly impacts the information criterion concerned.
- **Secondary** is the degree to which the defined control objective satisfies only to a lesser extent or indirectly the information criterion concerned.
- **Blank** could be applicable; however, requirements are more appropriately satisfied by another criterion in this process and/or by another process.

Similarly, all control measures will not necessarily impact the different IT resources to the same degree. Therefore, the COBIT Framework specifically indicates the applicability of the IT resources that are specifically managed by the process under consideration (not those that merely take part in the process). This classification is made within the COBIT Framework based on a rigorous process of input from researchers, experts and reviewers, using the strict definitions previously indicated.

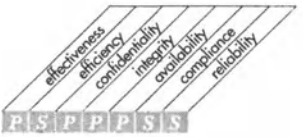
Each high-level control objective is accompanied by detailed control objectives, 318 in all, providing additional detail on how control should be exercised over that particular process. In addition, extensive audit guidelines are included for building on the objectives.

Sample high-level control objectives, with their related detailed control objectives, follow for PO9, the Assess Risks process in the Planning and Organisation domain, and DS5, the Ensure System Security process in the Delivery and Support domain.

PO9 Planning & Organisation
Assess Risks

COBIT

HIGH-LEVEL CONTROL OBJECTIVE



Control over the IT process of

assessing risks

that satisfies the business requirement

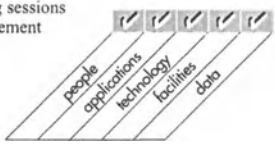
of supporting management decisions through achieving IT objectives and responding to threats by reducing complexity, increasing objectivity and identifying important decision factors

is enabled by

the organisation engaging itself in IT risk-identification and impact analysis, involving multi-disciplinary functions and taking cost-effective measures to mitigate risks

and takes into consideration

- risk management ownership and accountability
- different kinds of IT risks (technology, security, continuity, regulatory, etc.)
- defined and communicated risk tolerance profile
- root cause analyses and risk brainstorming sessions
- quantitative and/or qualitative risk measurement
- risk assessment methodology
- risk action plan
- timely reassessment



CONTROL OBJECTIVES PO9

DETAILED CONTROL OBJECTIVES

9 ASSESS RISKS

9.1 Business Risk Assessment
CONTROL OBJECTIVE
Management should establish a systematic risk assessment framework. Such a framework should incorporate a regular assessment of the relevant information risks to the achievement of the business objectives, forming a basis for determining how the risks should be managed to an acceptable level. The process should provide for risk assessments at both the global level and system specific level, for new projects as well as on a recurring basis, and with cross-disciplinary participation. Management should ensure that reassessments occur and that risk assessment information is updated with results of audits, inspections and identified incidents.

9.2 Risk Assessment Approach
CONTROL OBJECTIVE
Management should establish a general risk assessment approach which defines the scope and boundaries, the methodology to be adopted for risk assessments, the responsibilities and the required skills. Management should lead the identification of the risk mitigation solution and be involved in identifying vulnerabilities. Security specialists should lead threat identification and IT specialists should drive the control selection. The quality of the risk assessments should be ensured by a structured method and skilled risk assessors.

9.3 Risk Identification
CONTROL OBJECTIVE
The risk assessment approach should focus on the examination of the essential elements of risk and the cause/effect relationship between them. The essential elements of risk include tangible and intangible assets, asset value, threats, vulnerabilities, safeguards, consequences and likelihood of threat. The risk identification process should include qualitative and, where appropriate, quantitative risk ranking and should obtain input from management brainstorming, strategic planning, past audits and other assessments. The risk assessment should consider business, regulatory, legal, technology, trading partner and human resources risks.

9.4 Risk Measurement
CONTROL OBJECTIVE
The risk assessment approach should ensure that the analysis of risk identification information results in a quantitative and/or qualitative measurement of risk to which the examined area is exposed. The risk acceptance capacity of the organisation should also be assessed.

9.5 Risk Action Plan
CONTROL OBJECTIVE
The risk assessment approach should provide for the definition of a risk action plan to ensure that cost-effective controls and security measures mitigate exposure to risks on a continuing basis. The risk action plan should identify the risk strategy in terms of risk avoidance, mitigation or acceptance.

9.6 Risk Acceptance
CONTROL OBJECTIVE
The risk assessment approach should ensure the formal acceptance of the residual risk, depending on risk identification and measurement, organisational policy, uncertainty incorporated in the risk assessment approach itself and the cost effectiveness of implementing safeguards and controls. The residual risk should be offset with adequate insurance coverage, contractually negotiated liabilities and self-insurance.

continued on next page

DETAILED CONTROL OBJECTIVES *continued*

9.7 **Safeguard Selection**
CONTROL OBJECTIVE
While aiming for a reasonable, appropriate and proportional system of controls and safeguards, controls with the highest return on investment (ROI) and those that provide quick wins should receive first priority. The control system also needs to balance prevention, detection, correction and recovery measures. Furthermore, management needs to communicate the purpose of the control measures, manage conflicting measures and monitor the continuing effectiveness of all control measures.

9.8 **Risk Assessment Commitment**
CONTROL OBJECTIVE
Management should encourage risk assessment as an important tool in providing information in the design and implementation of internal controls, in the definition of the IT strategic plan and in the monitoring and evaluation mechanisms.

DS5 Delivery & Support
Ensure Systems Security

COBIT

HIGH-LEVEL CONTROL OBJECTIVE

Control over the IT process of

ensuring systems security

that satisfies the business requirement

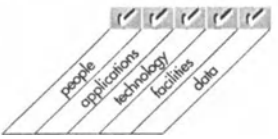
to safeguard information against unauthorised use, disclosure or modification, damage or loss

is enabled by

logical access controls which ensure that access to systems, data and programmes is restricted to authorised users

and takes into consideration

- confidentiality and privacy requirements
- authorisation, authentication and access control
- user identification and authorisation profiles
- need-to-have and need-to-know
- cryptographic key management
- incident handling, reporting and follow-up
- virus prevention and detection
- firewalls
- centralised security administration
- user training
- tools for monitoring compliance, intrusion testing and reporting



CONTROL OBJECTIVES DS5

DETAILED CONTROL OBJECTIVES

5 ENSURE SYSTEMS SECURITY

5.1 Manage Security Measures

CONTROL OBJECTIVE

IT security should be managed such that security measures are in line with business requirements. This includes:

- Translating risk assessment information to the IT security plans
- Implementing the IT security plan
- Updating the IT security plan to reflect changes in the IT configuration
- Assessing the impact of change requests on IT security
- Monitoring the implementation of the IT security plan
- Aligning IT security procedures to other policies and procedures

5.2 Identification, Authentication and Access

CONTROL OBJECTIVE

The logical access to and use of IT computing resources should be restricted by the implementation of adequate identification, authentication and authorisation mechanisms, linking users and resources with access rules. Such mechanisms should prevent unauthorised personnel, dial-up connections and other system (network) entry ports from accessing computer resources and minimise the need for authorised users to use multiple sign-ons. Procedures should also be in place to keep authentication and access mechanisms effective (e.g., regular password changes).

5.3 Security of Online Access to Data

CONTROL OBJECTIVE

In an online IT environment, IT management should implement procedures in line with the security policy that provides access security control based on the individual's demonstrated need to view, add, change or delete data.

5.4 User Account Management

CONTROL OBJECTIVE

Management should establish procedures to ensure timely action relating to requesting, establishing, issuing, suspending and closing of user accounts. A formal approval procedure outlining the data or system owner granting the access privileges should be included. The security of third-party access should be defined contractually and address administration and non-disclosure requirements. Outsourcing arrangements should address the risks, security controls and procedures for information systems and networks in the contract between the parties.

5.5 Management Review of User Accounts

CONTROL OBJECTIVE

Management should have a control process in place to review and confirm access rights periodically. Periodic comparison of resources with recorded accountability should be made to help reduce the risk of errors, fraud, misuse or unauthorised alteration.

5.6 User Control of User Accounts

CONTROL OBJECTIVE

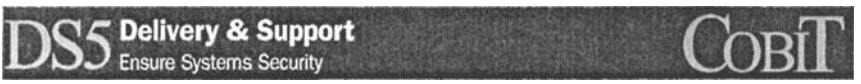
Users should systematically control the activity of their proper account(s). Also information mechanisms should be in place to allow them to oversee normal activity as well as to be alerted to unusual activity in a timely manner.

5.7 Security Surveillance

CONTROL OBJECTIVE

IT security administration should ensure that security activity is logged and any indication of imminent security violation is reported immediately to all who may be concerned, internally and externally, and is acted upon in a timely manner.

continued on next page



DETAILED CONTROL OBJECTIVES *continued*

5.8 **Data Classification**
CONTROL OBJECTIVE
Management should implement procedures to ensure that all data are classified in terms of sensitivity by a formal and explicit decision by the data owner according to the data classification scheme. Even data needing "no protection" should require a formal decision to be so designated. Owners should determine disposition and sharing of data, as well as whether and when programs and files are to be maintained, archived or deleted. Evidence of owner approval and data disposition should be maintained. Policies should be defined to support reclassification of information, based on changing sensitivities. The classification scheme should include criteria for managing exchanges of information between organisations, addressing both security and compliance with relevant legislation.

5.9 **Central Identification and Access Rights Management**
CONTROL OBJECTIVE
Controls are in place to ensure that the identification and access rights of users as well as the identity of system and data ownership are established and managed in a unique and central manner to obtain consistency and efficiency of global access control.

5.10 **Violation and Security Activity Reports**
CONTROL OBJECTIVE
IT security administration should ensure that violation and security activity is logged, reported, reviewed and appropriately escalated on a regular basis to identify and resolve incidents involving unauthorised activity. The logical access to the computer resources accountability information (security and other logs) should be granted based upon the principle of least privilege, or need-to-know.

5.11 **Incident Handling**
CONTROL OBJECTIVE
Management should establish a computer security incident handling capability to address security incidents by providing a centralised platform with sufficient expertise and equipped with rapid and secure communication facilities. Incident management responsibilities and procedures should be established to ensure an appropriate, effective and timely response to security incidents.

5.12 **Reaccreditation**
CONTROL OBJECTIVE
Management should ensure that reaccreditation of security (e.g., through "tiger teams") is periodically performed to keep up-to-date the formally approved security level and the acceptance of residual risk.

5.13 **Counterparty Trust**
CONTROL OBJECTIVE
Organisational policy should ensure that control practices are implemented to verify the authenticity of the counterparty providing electronic instructions or transactions. This can be implemented through trusted exchange of passwords, tokens or cryptographic keys.

5.14 **Transaction Authorisation**
CONTROL OBJECTIVE
Organisational policy should ensure that, where appropriate, controls are implemented to provide authenticity of transactions and establish the validity of a user's claimed identity to the system. This requires use of cryptographic techniques for signing and verifying transactions.

CONTROL OBJECTIVES DS5

5.15 **Non-Repudiation**
CONTROL OBJECTIVE
Organisational policy should ensure that, where appropriate, transactions cannot be denied by either party, and controls are implemented to provide non-repudiation of origin or receipt, proof of submission, and receipt of transactions. This can be implemented through digital signatures, time stamping and trusted third-parties, with appropriate policies that take into account relevant regulatory requirements.

5.16 **Trusted Path**
CONTROL OBJECTIVE
Organisational policy should ensure that sensitive transaction data is only exchanged over a trusted path. Sensitive information includes security management information, sensitive transaction data, passwords and cryptographic keys. To achieve this, trusted channels may need to be established using encryption between users, between users and systems, and between systems.

5.17 **Protection of Security Functions**
CONTROL OBJECTIVE
All security related hardware and software should at all times be protected against tampering to maintain their integrity and against disclosure of secret keys. In addition, organisations should keep a low profile about their security design, but should not base their security on the design being secret.

5.18 **Cryptographic Key Management**
CONTROL OBJECTIVE
Management should define and implement procedures and protocols to be used for generation, change, revocation, destruction, distribution, certification, storage, entry, use and archiving of cryptographic keys to ensure the protection of keys against modification and unauthorised disclosure. If a key is compromised, management should ensure this information is propagated to any interested party through the use of Certificate Revocation Lists or similar mechanisms.

5.19 **Malicious Software Prevention, Detection and Correction**
CONTROL OBJECTIVE
Regarding malicious software, such as computer viruses or trojan horses, management should establish a framework of adequate preventative, detective and corrective control measures, and occurrence response and reporting. Business and IT management should ensure that procedures are established across the organisation to protect information systems and technology from computer viruses. Procedures should incorporate virus protection, detection, occurrence response and reporting.

5.20 **Firewall Architectures and Connections with Public Networks**
CONTROL OBJECTIVE
If connection to the Internet or other public networks exists, adequate firewalls should be operative to protect against denial of services and any unauthorised access to the internal resources; should control any application and infrastructure management flows in both directions; and should protect against denial of service attacks.

5.21 **Protection of Electronic Value**
CONTROL OBJECTIVE
Management should protect the continued integrity of all cards or similar physical mechanisms used for authentication or storage of financial or other sensitive information, taking into consideration the related facilities, devices, employees and validation methods used.

10. COBIT'S MANAGEMENT GUIDELINES

COBIT's Management Guidelines consist of maturity models, critical success factors (CSFs), key goal indicators (KGIs) and key performance indicators (KPIs). This structure delivers a significantly improved framework responding to management's need for control and measurability of IT by providing management with tools to assess and measure their organisation's IT environment against COBIT's 34 IT processes.

COBIT's Management Guidelines are generic and action-oriented for the purpose of addressing the following types of management concerns:

- Performance measurement — What are the indicators of good performance?
- IT control profiling — What's important? What are the critical success factors for control?
- Awareness — What are the risks of not achieving our objectives?
- Benchmarking — What do others do? How do we measure and compare?

An answer to these requirements of determining and monitoring the appropriate IT security and control level is the definition of specific:

- **Benchmarking** of IT control practices (expressed as maturity models);
- **Performance indicators** of the IT processes—for their outcome and their performance;
- **Critical success factors** for getting these processes under control.

The Management Guidelines are consistent with and build upon the principles of the balanced business scorecard.[4] In "simple terms," these measures will assist management in monitoring their IT organisation by answering the following questions:

1. *What is the management concern?* Make sure that the enterprise needs are fulfilled.
2. *Where is it measured?* On the balanced business scorecard as a key goal indicator, representing an outcome of the business process.
3. *What is the IT concern?* That the IT processes deliver on a timely basis the right information to the enterprise, enabling the business needs to be fulfilled. This is a critical success factor for the enterprise.
4. *Where is that measured?* On the IT balanced scorecard, as a key goal indicator representing the outcome for IT, which is that information is delivered with the right criteria (effectiveness, efficiency, confidentiality, integrity, availability, compliance and reliability).

5. *What else needs to be measured?* Whether the outcome is positively influenced by a number of critical success factors that need to be measured as key performance indicators of how well IT is doing.

Each element of the Management Guidelines will be examined in further detail.

10.1 Maturity Models

IT management is constantly on the lookout for benchmarking and self-assessment tools in response to the need to know what to do in an efficient manner. Starting from COBIT's processes and high-level control objectives, the process owner should be able to incrementally benchmark against that control objective. This creates three needs:

- A relative measure of where the organisation is;
- A manner to decide efficiently where to go;
- A tool for measuring progress against the goal.

The approach to maturity models for control over IT processes consists of developing a method of scoring so that an organisation can grade itself from non-existent to optimised (from 0 to 5). This approach is based on the maturity model that the Software Engineering Institute defined for the maturity of the software development capability.[5] Whatever the model, the scales should not be too granular, as that would render the system difficult to use and suggest a precision that is not justifiable.

In contrast, one should concentrate on maturity levels based on a set of conditions that can be unambiguously met. Against levels developed for each of COBIT's 34 IT processes, management can map:

- The current status of the organisation — where the organisation is today;
- The current status of (best-in-class in) the industry — the comparison;
- The current status of international standard guidelines — additional comparison;
- The organisation's strategy for improvement — where the organisation wants to be.

For each of the 34 IT processes, there is an incremental measurement scale, based on a rating of 0 through 5. The scale is associated with generic qualitative maturity model descriptions ranging from Non-existent to Optimised as follows:

- **0 Non-existent**. Complete lack of any recognisable processes. The organisation has not even recognised that there is an issue to be addressed.
- **1 Initial**. There is evidence that the organisation has recognised that the issues exist and need to be addressed. There are no standardised processes but instead there are ad hoc approaches that tend to be applied on an individual or case-by-case basis. The overall approach to management is disorganised.
- **2 Repeatable**. Processes have developed to the stage where similar procedures are followed by different people undertaking the same task. There is no formal training or communication of standard procedures and responsibility is left to the individual. There is a high degree of reliance on the knowledge of individuals and therefore errors are likely.
- **3 Defined**. Procedures have been standardised and documented, and communicated through training. It is, however, left to the individual to follow these processes, and it is unlikely that deviations will be detected. The procedures themselves are not sophisticated but are the formalisation of existing practices.
- **4 Managed**. It is possible to monitor and measure compliance with procedures and to take action where processes appear not to be working effectively. Processes are under constant improvement and provide good practice. Automation and tools are used in a limited or fragmented way.
- **5 Optimised**. Processes have been refined to a level of best practice, based on the results of continuous improvement and maturity modelling with other organisations. IT is used in an integrated way to automate the workflow, providing tools to improve quality and effectiveness, making the enterprise quick to adapt.

The maturity model scales help professionals explain to managers where IT management shortcomings exist and set targets for where they need to be by comparing their organisation's control practices to the best practice examples. The right maturity level will be influenced by the enterprise's business objectives and operating environment. Specifically, the level of control maturity depends on the enterprise's dependence on IT, its technology sophistication and, most importantly, the value of its information.

A strategic reference point for an organisation to improve security and control could also consist of looking at emerging international standards and best-in-class practices. The emerging practices of today may become the expected level of performance of tomorrow and is therefore useful for planning where an organisation wants to be over time.

In summary, maturity models:

- Refer to business requirements and the enabling aspects at the different maturity levels;
- Are a scale that lends itself to pragmatic comparison, where differences can be made measurable in an easy manner;
- Help setting "as-is" and "to-be" positions relative to IT governance, security and control maturity;
- Lend themselves to gap analysis to determine what needs to be done to achieve a chosen level;
- Avoid, where possible, discrete levels that create thresholds that are difficult to cross;
- Increasingly apply critical success factors;
- Are not industry-specific nor always applicable. The type of business defines what is appropriate.

10.2 Critical Success Factors

Critical success factors provide management with guidance for implementing control over IT and its processes. They are the most important things to do that contribute to the IT process achieving its goals. They are activities that can be of a strategic, technical, organisational, process or procedural nature. They are usually dealing with capabilities and skills and have to be short, focused and action-oriented, leveraging the resources that are of primary importance in the process under consideration.

A number of critical success factors can be deduced that apply to most IT processes.

Applying to IT in general

- IT processes are defined and aligned with the IT strategy and the business goals.
- The customers of the process and their expectations are known.
- Processes are scalable and their resources are appropriately managed and leveraged.
- The required quality of staff (training, transfer of information, morale, etc.) and availability of skills (recruit, retain, retrain) exist.
- IT performance is measured in financial terms, in relation to customer satisfaction, for process effectiveness and for future capability. IT management is rewarded based on these measures.
- A continuous quality improvement effort is applied.

Applying to most IT processes

- All process stakeholders (users, management, etc.) are aware of the risks, of the importance of IT and the opportunities it can offer, and provide strong commitment and support.
- Goals and objectives are communicated across all disciplines and understood; it is known how processes implement and monitor objectives, and who is accountable for process performance.
- People are goal-focused and have the right information on customers, on internal processes and on the consequences of their decisions.
- A business culture is established, encouraging cross-divisional co-operation, teamwork and continuous process improvement.
- There is integration and alignment of major processes, e.g., change, problem and configuration management.
- Control practices are applied to increase efficient and optimal use of resources and improve the effectiveness of processes.

Applying to IT governance

- Control practices are applied to increase transparency, reduce complexity, promote learning, provide flexibility and scalability, and avoid breakdowns in internal control and oversight.
- Practices that enable sound oversight are applied: a control environment and culture; a code of conduct; risk assessment as a standard practice; self-assessments; formal compliance on adherence to established standards; monitoring and follow-up of control deficiencies and risk.
- IT governance is recognised and defined, and its activities are integrated into the enterprise governance process, giving clear direction for IT strategy, a risk management framework, a system of controls and a security policy.
- IT governance focuses on major IT projects, change initiatives and quality efforts, with awareness of major IT processes, the responsibilities and the required resources and capabilities.
- An audit committee is established to appoint and oversee an independent auditor, drive the IT audit plan and review the results of audits and third party opinions.

In summary, critical success factors are:

- Essential enablers focused on the process or supporting environment;
- A thing or a condition that is required to increase the probability of success of the process;
- Observable—usually measurable—characteristics of the organisation and process;
- Either strategic, technological, organisational or procedural in nature;

- Focused on obtaining, maintaining and leveraging capability and skills;
- Expressed in terms of the process, not necessarily the business.

10.3 Key Goal Indicators

A key goal indicator, representing the process goal, is a measure of *what* has to be accomplished. It is a measurable indicator of the process achieving its goals, often defined as a target to achieve. By comparison, a key performance indicator is a measure of *how well* the process is performing.

How are business and IT goals and measures linked? The COBIT Framework expresses the objectives for IT in terms of the information criteria that the business needs in order to achieve the business objectives, which will usually be expressed in terms of:

- Availability of systems and services;
- Absence of integrity and confidentiality risks;
- Cost-efficiency of processes and operations;
- Confirmation of reliability, effectiveness and compliance.

The goal for IT can then be expressed as delivering the information that the business needs in line with these criteria. These information criteria are provided in the Management Guidelines with an indication whether they have primary or secondary importance for the process under review. In practice, the information criteria profile of an enterprise would be more specific. The degree of importance of each of the information criteria is a function of the business and the environment in which the enterprise operates.

Key goal indicators are *lag* indicators, as they can be measured only after the fact, as opposed to key performance indicators, which are *lead* indicators, giving an indication of success before the fact. They also can be expressed negatively, i.e., in terms of the impact of not reaching the goal.

Key goal indicators should be measurable as a number or percentage. These measures should show that information and technology are contributing to the mission and strategy of the organisation. Because goals and targets are specific to the enterprise and its environment, many key goal indicators have been expressed with a direction, e.g., increased availability, decreased cost. In practice, management has to set specific targets which need to be met, taking into account past performance and future goals.

In summary, key goal indicators are:

- A representation of the process goal, i.e., a measure of *what*, or a target to achieve;
- The description of the outcome of the process and therefore lag indicators, i.e., measurable after the fact;
- Immediate indicators of the successful completion of the process or indirect indicators of the value the process delivered to the business;
- Possibly descriptions of a measure of the impact of not reaching the process goal;
- Focused on the customer and financial dimensions of the balanced business scorecard;
- IT-oriented but business-driven;
- Expressed in precise, measurable terms wherever possible;
- Focused on those information criteria that have been identified as most important for this process.

10.4 Key Performance Indicators

Key performance indicators are measures that tell management that an IT process is achieving its business requirements by monitoring the performance of the enablers of that IT process. Building on balanced business scorecard principles, the relationship between key performance indicators and key goal indicators is as follows: key performance indicators are short, focused and measurable indicators of performance of the enabling factors of the IT processes, indicating how well the process enables the goal to be reached. While key goal indicators focus on *what*, the key performance indicators are concerned with *how*. They often are a measure of a critical success factor and, when monitored and acted upon, identify opportunities for the improvement of the process. These improvements should positively influence the outcome and, as such, key performance indicators have a cause-effect relationship with the key goal indicators of the process.

While key goal indicators are business-driven, key performance indicators are process-oriented and often express how well the processes and the organisation leverage and manage the needed resources. Similar to key goal indicators, they often are expressed as a number or percentage. A good test of a key performance indicator is to see whether it really does predict success or failure of the process goal and whether or not it assists management in improving the process.

Some generic key performance indicators follow that usually are applicable to all IT processes:

Applying to IT in general
- Reduced cycle times (i.e., responsiveness of IT production and development);
- Increased quality and innovation;
- Utilisation of communications bandwidth and computing power;
- Service availability and response times;
- Satisfaction of stakeholders (survey and number of complaints);
- Number of staff trained in new technology and customer service skills.

Applying to most IT processes
- Improved cost-efficiency of the process (cost vs. deliverables);
- Staff productivity (number of deliverables) and morale (survey);
- Amount of errors and rework.

Applying to IT governance
- Benchmark comparisons;
- Number of non-compliance reportings.

In summary, key performance indicators:
- Are measures of how well the process is performing;
- Predict the probability of success or failure in the future, i.e., are lead indicators;
- Are process-oriented, but IT-driven;
- Focus on the process and learning dimensions of the balanced business scorecard;
- Are expressed in precisely measurable terms;
- Help in improving the IT process when measured and acted upon;
- Focus on those resources identified as the most important for this process.

11. MANAGEMENT GUIDELINES FOR SELECTED COBIT PROCESSES

Although COBIT consists of 34 high-level IT control practices, through extensive testing and surveying, the 15 most important have been identified. On the following pages, COBIT's Management Guideline for seven of these 15 processes is included, outlining critical success factors, key goal indicators, key performance indicators and a maturity model for each.

PO1 Planning & Organisation
Define a Strategic Information Technology Plan

COBIT

Control over the IT process **Define a Strategic IT Plan** with the business goal of *striking an optimum balance of information technology opportunities and IT business requirements as well as ensuring its further accomplishment*

ensures delivery of information to the business that addresses the required Information Criteria and is measured by Key Goal Indicators

is enabled by *a strategic planning process undertaken at regular intervals giving rise to long-term plans; the long-term plans should periodically be translated into operational plans setting clear and concrete short-term goals*

considers Critical Success Factors that leverage specific IT Resources and is measured by Key Performance Indicators

Information Criteria	IT Resources
P effectiveness	✓ people
S efficiency	✓ applications
confidentiality	✓ technology
integrity	✓ facilities
availability	✓ data
compliance	
reliability	

(P) primary (S) secondary (✓) applicable to

Critical Success Factors

- The planning process provides for a prioritisation scheme for the business objectives and quantifies, where possible, the business requirements
- Management buy-in and support is enabled by a documented methodology for the IT strategy development, the support of validated data and a structured, transparent decision-making process
- The IT strategic plan clearly states a risk position, such as leading edge or road-tested, innovator or follower, and the required balance between time-to-market, cost of ownership and service quality
- All assumptions of the strategic plan have been challenged and tested
- The processes, services and functions needed for the outcome are defined, but are flexible and changeable, with a transparent change control process
- A reality check of the strategy by a third party has been conducted to increase objectivity and is repeated at appropriate times
- IT strategic planning is translated into roadmaps and migration strategies

Key Goal Indicators

- Percent of IT and business strategic plans that are aligned and cascaded into long- and short-range plans leading to individual responsibilities
- Percent of business units that have clear, understood and current IT capabilities
- Management survey determines clear link between responsibilities and the business and IT strategic goals
- Percent of business units using strategic technology covered in the IT strategic plan
- Percent of IT budget championed by business owners
- Acceptable and reasonable number of outstanding IT projects

Key Performance Indicators

- Currency of IT capabilities assessment (number of months since last update)
- Age of IT strategic plan (number of months since last update)
- Percent of participant satisfaction with the IT strategic planning process
- Time lag between change in the IT strategic plans and changes to operating plans
- Index of participants involved in strategic IT plan development, based on size of effort, ratio of involvement of business owners to IT staff and number of key participants
- Index of quality of the plan, including timelines of development effort, adherence to structured approach and completeness of plan

MANAGEMENT GUIDELINES PO1

PO1 Maturity Model

Control over the IT process **Define a Strategic IT Plan** with the business goal of *striking an optimum balance of information technology opportunities and IT business requirements as well as ensuring its further accomplishment*

0 Non-existent IT strategic planning is not performed. There is no management awareness that IT strategic planning is needed to support business goals.

1 Initial/Ad Hoc The need for IT strategic planning is known by IT management, but there is no structured decision process in place. IT strategic planning is performed on an as needed basis in response to a specific business requirement and results are therefore sporadic and inconsistent. IT strategic planning is occasionally discussed at IT management meetings, but not at business management meetings. The alignment of business requirements, applications and technology takes place reactively, driven by vendor offerings, rather than by an organisation-wide strategy. The strategic risk position is identified informally on a project-by-project basis.

2 Repeatable but Intuitive IT strategic planning is understood by IT management, but is not documented. IT strategic planning is performed by IT management, but only shared with business management on an as needed basis. Updating of the IT strategic plan occurs only in response to requests by management and there is no proactive process for identifying those IT and business developments that require updates to the plan. Strategic decisions are driven on a project-by-project basis, without consistency with an overall organisation strategy. The risks and user benefits of major strategic decisions are being recognised, but their definition is intuitive.

3 Defined Process A policy defines when and how to perform IT strategic planning. IT strategic planning follows a structured approach, which is documented and known to all staff. The IT planning process is reasonably sound and ensures that appropriate planning is likely to be performed. However, discretion is given to individual managers with respect to implementation of the process and there are no procedures to examine the process on a regular basis. The overall IT strategy includes a consistent definition of risks that the organisation is willing to take as an innovator or follower. The IT financial, technical and human resources strategies increasingly drive the acquisition of new products and technologies.

4 Managed and Measurable IT strategic planning is standard practice and exceptions would be noticed by management. IT strategic planning is a defined management function with senior level responsibilities. With respect to the IT strategic planning process, management is able to monitor it, make informed decisions based on it and measure its effectiveness. Both short-range and long-range IT planning occurs and is cascaded down into the organisation, with updates done as needed. The IT strategy and organisation-wide strategy are increasingly becoming more coordinated by addressing business processes and value-added capabilities and by leveraging the use of applications and technologies through business process re-engineering. There is a well-defined process for balancing the internal and external resources required in system development and operations. Benchmarking against industry norms and competitors is becoming increasingly formalised.

5 Optimised IT strategic planning is a documented, living process, is continuously considered in business goal setting and results in discernable business value through investments in IT. Risk and value added considerations are continuously updated in the IT strategic planning process. There is an IT strategic planning function that is integral to the business planning function. Realistic long-range IT plans are developed and constantly being updated to reflect changing technology and business-related developments. Short-range IT plans contain project task milestones and deliverables, which are continuously monitored and updated, as changes occur. Benchmarking against well-understood and reliable industry norms is a well-defined process and is integrated with the strategy formulation process. The IT organisation identifies and leverages new technology developments to drive the creation of new business capabilities and improve the competitive advantage of the organisation.

PO9 Planning & Organisation
Assess Risks

COBIT

Control over the IT process **Assess Risks** with the business goal of *supporting management decisions in achieving IT objectives and responding to threats by reducing complexity, increasing objectivity and identifying important decision factors*

ensures delivery of information to the business that addresses the required Information Criteria and is measured by Key Goal Indicators

is enabled by *the organisation engaging itself in IT risk-identification and impact analysis, involving multi-disciplinary functions and taking cost-effective measures to mitigate risks*

considers Critical Success Factors that leverage specific IT Resources and is measured by Key Performance Indicators

Information Criteria
P effectiveness
S efficiency
P confidentiality
P integrity
P availability
S compliance
S reliability

(P) primary (S) secondary

IT Resources
✓ people
✓ applications
✓ technology
✓ facilities
✓ data

(✓) applicable to

Key Goal Indicators

- Increased degree of awareness of the need for risk assessments
- Decreased number of incidents caused by risks identified after the fact
- Increased number of identified risks that have been sufficiently mitigated
- Increased number of IT processes that have formal documented risk assessments completed
- Appropriate percent or number of cost effective risk assessment measures

Critical Success Factors

- There are clearly defined roles and responsibilities for risk management ownership and management accountability
- A policy is established to define risk limits and risk tolerance
- The risk assessment is performed by matching vulnerabilities, threats and the value of data
- Structured risk information is maintained, fed by incident reporting
- Responsibilities and procedures for defining, agreeing on and funding risk management improvements exist
- Focus of the assessment is primarily on real threats and less on theoretical ones
- Brainstorming sessions and root cause analyses leading to risk identification and mitigation are routinely performed
- A reality check of the strategy is conducted by a third party to increase objectivity and is repeated at appropriate times

Key Performance Indicators

- Number of risk management meetings and workshops
- Number of risk management improvement projects
- Number of improvements to the risk assessment process
- Level of funding allocated to risk management projects
- Number and frequency of updates to published risk limits and policies
- Number and frequency of risk monitoring reports
- Number of personnel trained in risk management methodology

MANAGEMENT GUIDELINES PO9

PO9 Maturity Model

Control over the IT process **Assess Risks** with the business goal of *supporting management decisions in achieving IT objectives and responding to threats by reducing complexity, increasing objectivity and identifying important decision factors*

0 Non-existent Risk assessment for processes and business decisions does not occur. The organisation does not consider the business impacts associated with security vulnerabilities and with development project uncertainties. Risk management has not been identified as relevant to acquiring IT solutions and delivering IT services.

1 Initial/Ad Hoc The organisation is aware of its legal and contractual responsibilities and liabilities, but considers IT risks in an ad hoc manner, without following defined processes or policies. Informal assessments of project risk take place as determined by each project. Risk assessments are not likely to be identified specifically within a project plan or to be assigned to specific managers involved in the project. IT management does not specify responsibility for risk management in job descriptions or other informal means. Specific IT-related risks such as security, availability and integrity are occasionally considered on a project-by-project basis. IT-related risks affecting day-to-day operations are infrequently discussed at management meetings. Where risks have been considered, mitigation is inconsistent.

2 Repeatable but Intuitive There is an emerging understanding that IT risks are important and need to be considered. Some approach to risk assessment exists, but the process is still immature and developing. The assessment is usually at a high-level and is typically applied only to major projects. The assessment of on-going operations depends mainly on IT managers raising it as an agenda item, which often only happens when problems occur. IT management has not generally defined procedures or job descriptions dealing with risk management.

3 Defined Process An organisation-wide risk management policy defines when and how to conduct risk assessments. Risk assessment follows a defined process that is documented and available to all staff through training. Decisions to follow the process and to receive training are left to the individual's discretion. The methodology is convincing and sound, and ensures that key risks to the business are likely to be identified. Decisions to follow the process are left to individual IT managers and there is no procedure to ensure that all projects are covered or that the ongoing operation is examined for risk on a regular basis.

4 Managed and Measurable The assessment of risk is a standard procedure and exceptions to following the procedure would be noticed by IT management. It is likely that IT risk management is a defined management function with senior level responsibility. The process is advanced and risk is assessed at the individual project level and also regularly with regard to the overall IT operation. Management is advised on changes in the IT environment which could significantly affect the risk scenarios, such as an increased threat from the network or technical trends that affect the soundness of the IT strategy. Management is able to monitor the risk position and make informed decisions regarding the exposure it is willing to accept. Senior management and IT management have determined the levels of risk that the organisation will tolerate and have standard measures for risk/return ratios. Management budgets for operational risk management projects to reassess risks on a regular basis. A risk management database is established.

5 Optimised Risk assessment has developed to the stage where a structured, organisation-wide process is enforced, followed regularly and well managed. Risk brainstorming and root cause analysis, involving expert individuals, are applied across the entire organisation. The capturing, analysis and reporting of risk management data are highly automated. Guidance is drawn from leaders in the field and the IT organisation takes part in peer groups to exchange experiences. Risk management is truly integrated into all business and IT operations, is well accepted and extensively involves the users of IT services.

PO10 Planning & Organisation
Manage Projects

COBIT

Control over the IT process **Manage Projects** with the business goal of *setting priorities and delivering on time and within budget*

ensures delivery of information to the business that addresses the required Information Criteria and is measured by Key Goal Indicators

is enabled by *the organisation identifying and prioritising projects in line with the operational plan and the adoption and application of sound project management techniques for each project undertaken*

considers Critical Success Factors that leverage specific IT Resources and is measured by Key Performance Indicators

Information Criteria	IT Resources
P effectiveness	✓ people
P efficiency	✓ applications
confidentiality	✓ technology
integrity	✓ facilities
availability	data
compliance	
reliability	

(P) primary (S) secondary (✓) applicable to

Key Goal Indicators

- Increased number of projects completed on time and on budget
- Availability of accurate project schedule and budget information
- Decrease in systemic and common project problems
- Improved timeliness of project risk identification
- Increased organisation satisfaction with project delivered services
- Improved timeliness of project management decisions

Critical Success Factors

- Experienced and skilled project managers are available
- An accepted and standard programme management process is in place
- There is senior management sponsorship of projects, and stakeholders and IT staff share in the definition, implementation and management of projects
- There is an understanding of the abilities and limitations of the organisation and the IT function in managing large, complex projects
- An organisation-wide project risk assessment methodology is defined and enforced
- All projects have a plan with clear traceable work breakdown structures, reasonably accurate estimates, skill requirements, issues to track, a quality plan and a transparent change process
- The transition from the implementation team to the operational team is a well-managed process
- A system development life cycle methodology has been defined and is used by the organisation

Key Performance Indicators

- Increased number of projects delivered in accordance with a defined methodology
- Percent of stakeholder participation in projects (involvement index)
- Number of project management training days per project team member
- Number of project milestone and budget reviews
- Percent of projects with post-project reviews
- Average number of years of experience of project managers

MANAGEMENT GUIDELINES PO10

PO10 Maturity Model

Control over the IT process **Manage Projects** with the business goal of *setting priorities and delivering on time and within budget*

0 Non-existent Project management techniques are not used and the organisation does not consider business impacts associated with project mismanagement and development project failures.

1 Initial/Ad Hoc The organisation is generally aware of the need for projects to be structured and is aware of the risks of poorly managed projects. The use of project management techniques and approaches within IT is a decision left to individual IT managers. Projects are generally poorly defined and do not incorporate business and technical objectives of the organisation or the business stakeholders. There is a general lack of management commitment and project ownership and critical decisions are made without user management or customer input. There is little or no customer and user involvement in defining IT projects. There is no clear organisation within IT projects and roles and responsibilities are not defined. Project schedules and milestones are poorly defined. Project staff time and expenses are not tracked and compared to budgets.

2 Repeatable but Intuitive Senior management has gained and communicated an awareness of the need for IT project management. The organisation is in the process of learning and repeating certain techniques and methods from project to project. IT projects have informally defined business and technical objectives. There is limited stakeholder involvement in IT project management. Some guidelines have been developed for most aspects of project management, but their application is left to the discretion of the individual project manager.

3 Defined Process The IT project management process and methodology have been formally established and communicated. IT projects are defined with appropriate business and technical objectives. Stakeholders are involved in the management of IT projects. The IT project organisation and some roles and responsibilities are defined. IT projects have defined and updated milestones, schedules, budget and performance measurements. IT projects have formal post system implementation procedures. Informal project management training is provided. Quality assurance procedures and post system implementation activities have been defined, but are not broadly applied by IT managers. Policies for using a balance of internal and external resources are being defined.

4 Managed and Measurable Management requires formal and standardised project metrics and "lessons learned" to be reviewed following project completion. Project management is measured and evaluated throughout the organisation and not just within IT. Enhancements to the project management process are formalised and communicated, and project team members are trained on all enhancements. Risk management is performed as part of the project management process. Stakeholders actively participate in the projects or lead them. Project milestones, as well as the criteria for evaluating success at each milestone, have been established. Value and risk are measured and managed prior to, during and after the completion of projects. Management has established a programme management function within IT. Projects are defined, staffed and managed to increasingly address organisation goals, rather than only IT specific ones.

5 Optimised A proven, full life-cycle project methodology is implemented and enforced, and is integrated into the culture of the entire organisation. An on-going programme to identify and institutionalise best practices has been implemented. There is strong and active project support from senior management sponsors as well as stakeholders. IT management has implemented a project organisation structure with documented roles, responsibilities and staff performance criteria. A long-term IT resources strategy is defined to support development and operational outsourcing decisions. An integrated programme management office is responsible for projects from inception to post implementation. The programme management office is under the management of the business units and requisitions and directs IT resources to complete projects. Organisation-wide planning of projects ensures that user and IT resources are best utilised to support strategic initiatives.

AI6 Acquisition & Implementation
Manage Changes

COBIT

Control over the IT process **Manage Changes** with the business goal of *minimising the likelihood of disruption, unauthorised alterations and errors*

ensures delivery of information to the business that addresses the required Information Criteria and is measured by Key Goal Indicators

is enabled by *a management system which provides for the analysis, implementation and follow-up of all changes requested and made to the existing IT infrastructure*

considers Critical Success Factors that leverage specific IT Resources and is measured by Key Performance Indicators

Information Criteria	IT Resources
P effectiveness	✓ people
P efficiency	✓ applications
confidentiality	✓ technology
P integrity	✓ facilities
P availability	✓ data
compliance	
S reliability	

(P) primary (S) secondary (✓) applicable to

Critical Success Factors

- Change policies are clear and known and they are rigorously and systematically implemented
- Change management is strongly integrated with release management and is an integral part of configuration management
- There is a rapid and efficient planning, approval and initiation process covering identification, categorisation, impact assessment and prioritisation of changes
- Automated process tools are available to support workflow definition, pro-forma workplans, approval templates, testing, configuration and distribution
- Expedient and comprehensive acceptance test procedures are applied prior to making the change
- A system for tracking and following individual changes, as well as change process parameters, is in place
- A formal process for hand-over from development to operations is defined
- Changes take the impact on capacity and performance requirements into account
- Complete and up-to-date application and configuration documentation is available
- A process is in place to manage co-ordination between changes, recognising interdependencies
- An independent process for verification of the success or failure of change is implemented
- There is segregation of duties between development and production

Key Goal Indicators

- Reduced number of errors introduced into systems due to changes
- Reduced number of disruptions (loss of availability) caused by poorly managed change
- Reduced impact of disruptions caused by change
- Reduced level of resources and time required as a ratio to number of changes
- Number of emergency fixes

Key Performance Indicators

- Number of different versions installed at the same time
- Number of software release and distribution methods per platform
- Number of deviations from the standard configuration
- Number of emergency fixes for which the normal change management process was not applied retroactively
- Time lag between the availability of the fix and its implementation
- Ratio of accepted to refused change implementation requests

MANAGEMENT GUIDELINES AI6

AI6 Maturity Model

Control over the IT process **Manage Changes** with the business goal of *minimising the likelihood of disruption, unauthorised alterations and errors*

0 Non-existent There is no defined change management process and changes can be made with virtually no control. There is no awareness that change can be disruptive for both IT and business operations, and no awareness of the benefits of good change management.

1 Initial/Ad Hoc It is recognised that changes should be managed and controlled, but there is no consistent process to follow. Practices vary and it is likely that unauthorised changes will take place. There is poor or non-existent documentation of change and configuration documentation is incomplete and unreliable. Errors are likely to occur together with interruptions to the production environment caused by poor change management.

2 Repeatable but Intuitive There is an informal change management process in place and most changes follow this approach; however, it is unstructured, rudimentary and prone to error. Configuration documentation accuracy is inconsistent and only limited planning and impact assessment takes place prior to a change. There is considerable inefficiency and rework.

3 Defined Process There is a defined formal change management process in place, including categorisation, prioritisation, emergency procedures, change authorisation and release management, but compliance is not enforced. The defined process is not always seen as suitable or practical and, as a result, workarounds take place and processes are bypassed. Errors are likely to occur and unauthorised changes will occasionally occur. The analysis of the impact of IT changes on business operations is becoming formalised, to support planned rollouts of new applications and technologies.

4 Managed and Measurable The change management process is well developed and consistently followed for all changes and management is confident that there are no exceptions. The process is efficient and effective, but relies on considerable manual procedures and controls to ensure that quality is achieved. All changes are subject to thorough planning and impact assessment to minimise the likelihood of post-production problems. An approval process for changes is in place. Change management documentation is current and correct, with changes formally tracked. Configuration documentation is generally accurate. IT change management planning and implementation is becoming more integrated with changes in the business processes, to ensure that training, organisational changes and business continuity issues are addressed. There is increased co-ordination between IT change management and business process re-design.

5 Optimised The change management process is regularly reviewed and updated to keep in line with best practices. Configuration information is computer based and provides version control. Software distribution is automated and remote monitoring capabilities are available. Configuration and release management and tracking of changes is sophisticated and includes tools to detect unauthorised and unlicensed software. IT change management is integrated with business change management to ensure that IT is an enabler in increasing productivity and creating new business opportunities for the organisation.

DS5 Delivery & Support
Ensure Systems Security

COBIT

Control over the IT process **Ensure Systems Security** with the business goal of *safeguarding information against unauthorised use, disclosure or modification, damage or loss*

ensures delivery of information to the business that addresses the required Information Criteria and is measured by Key Goal Indicators

is enabled by *logical access controls which ensure that access to the systems, data and programmes is restricted to authorised users*

considers Critical Success Factors that leverage specific IT Resources and is measured by Key Performance Indicators

Information Criteria
effectiveness
efficiency
P confidentiality
P integrity
S availability
S compliance
S reliability

(P) primary (S) secondary

IT Resources
✓ people
✓ applications
✓ technology
✓ facilities
✓ data

(✓) applicable to

Critical Success Factors

- An overall security plan is developed that covers the building of awareness, establishes clear policies and standards, identifies a cost-effective and sustainable implementation, and defines monitoring and enforcement processes
- There is awareness that a good security plan takes time to evolve
- The corporate security function reports to senior management and is responsible for executing the security plan
- Management and staff have a common understanding of security requirements, vulnerabilities and threats, and they understand and accept their own security responsibilities
- Third-party evaluation of security policy and architecture is conducted periodically
- A "building permit" programme is defined, identifying security baselines that have to be adhered to
- A "drivers licence" programme is in place for those developing, implementing and using systems, enforcing security certification of staff
- The security function has the means and ability to detect, record, analyse significance, report and act upon security incidents when they do occur, while minimising the probability of occurrence by applying intrusion testing and active monitoring
- A centralised user management process and system provides the means to identify and assign authorisations to users in a standard and efficient manner
- A process is in place to authenticate users at reasonable cost, light to implement and easy to use

Key Goal Indicators

- No incidents causing public embarrassment
- Immediate reporting on critical incidents
- Alignment of access rights with organisational responsibilities
- Reduced number of new implementations delayed by security concerns
- Full compliance, or agreed and recorded deviations from minimum security requirements
- Reduced number of incidents involving unauthorised access, loss or corruption of information

Key Performance Indicators

- Reduced number of security-related service calls, change requests and fixes
- Amount of downtime caused by security incidents
- Reduced turnaround time for security administration requests
- Number of systems subject to an intrusion detection process
- Number of systems with active monitoring capabilities
- Reduced time to investigate security incidents
- Time lag between detection, reporting and acting upon security incidents
- Number of IT security awareness training days

MANAGEMENT GUIDELINES DS5

DS5 Maturity Model

Control over the IT process **Ensure Systems Security** with the business goal of *safeguarding information against unauthorised use, disclosure or modification, damage or loss*

0 Non-existent The organisation does not recognise the need for IT security. Responsibilities and accountabilities are not assigned for ensuring security. Measures supporting the management of IT security are not implemented. There is no IT security reporting and no response process to IT security breaches. There is a complete lack of a recognisable system security administration process.

1 Initial/Ad Hoc The organisation recognises the need for IT security, but security awareness depends on the individual. IT security is addressed on a reactive basis and not measured. IT security breaches invoke "finger pointing" responses if detected, because responsibilities are unclear. Responses to IT security breaches are unpredictable.

2 Repeatable but Intuitive Responsibilities and accountabilities for IT security are assigned to an IT security co-ordinator with no management authority. Security awareness is fragmented and limited. IT security information is generated, but is not analysed. Security solutions tend to respond reactively to IT security incidents and by adopting third-party offerings, without addressing the specific needs of the organisation. Security policies are being developed, but inadequate skills and tools are still being used. IT security reporting is incomplete, misleading or not pertinent.

3 Defined Process Security awareness exists and is promoted by management. Security awareness briefings have been standardised and formalised. IT security procedures are defined and fit into a structure for security policies and procedures. Responsibilities for IT security are assigned, but not consistently enforced. An IT security plan exists, driving risk analysis and security solutions. IT security reporting is IT focused, rather than business focused. Ad hoc intrusion testing is performed.

4 Managed and Measurable Responsibilities for IT security are clearly assigned, managed and enforced. IT security risk and impact analysis is consistently performed. Security policies and practices are completed with specific security baselines. Security awareness briefings have become mandatory. User identification, authentication and authorisation are being standardised. Security certification of staff is being established. Intrusion testing is a standard and formalised process leading to improvements. Cost/benefit analysis, supporting the implementation of security measures, is increasingly being utilised. IT security processes are co-ordinated with the overall organisation security function. IT security reporting is linked to business objectives.

5 Optimised IT security is a joint responsibility of business and IT management and is integrated with corporate security business objectives. IT security requirements are clearly defined, optimised and included in a verified security plan. Security functions are integrated with applications at the design stage and end users are increasingly accountable for managing security. IT security reporting provides early warning of changing and emerging risk, using automated active monitoring approaches for critical systems. Incidents are promptly addressed with formalised incident response procedures supported by automated tools. Periodic security assessments evaluate the effectiveness of implementation of the security plan. Information on new threats and vulnerabilities is systematically collected and analysed, and adequate mitigating controls are promptly communicated and implemented. Intrusion testing, root cause analysis of security incidents and pro-active identification of risk is the basis for continuous improvements. Security processes and technologies are integrated organisation wide.

DS10 Delivery & Support
Manage Problems and Incidents

COBIT

Control over the IT process **Manage Problems and Incidents** with the business goal of *ensuring that problems and incidents are resolved, and the cause investigated to prevent any recurrence*

ensures delivery of information to the business that addresses the required Information Criteria and is measured by Key Goal Indicators

is enabled by *a problem management system which records and progresses all incidents*

considers Critical Success Factors that leverage specific IT Resources and is measured by Key Performance Indicators

Information Criteria	
P	effectiveness
P	efficiency
	confidentiality
	integrity
S	availability
	compliance
	reliability

(P) primary (S) secondary

IT Resources	
✓	people
✓	applications
✓	technology
✓	facilities
✓	data

(✓) applicable to

Key Goal Indicators

- A measured reduction of the impact of problems and incidents on IT resources
- A measured reduction in the elapsed time from initial symptom report to problem resolution
- A measured reduction in unresolved problems and incidents
- A measured increase in the number of problems avoided through pre-emptive fixes
- Reduced time lag between identification and escalation of high-risk problems and incidents

Critical Success Factors

- There is clear integration of problem management with availability and change management
- Accessibility to configuration data, as well as the ability to keep track of problems for each configuration component, is provided
- An accurate means of communicating problem incidents, symptoms, diagnosis and solutions to the proper support personnel is in place
- Accurate means exist to communicate to users and IT the exceptional events and symptoms that need to be reported to problem management
- Training is provided to support personnel in problem resolution techniques
- Up-to-date roles and responsibilities charts are available to support incident management
- There is vendor involvement during problem investigation and resolution
- Post-facto analysis of problem handling procedures is applied

Key Performance Indicators

- Elapsed time from initial symptom recognition to entry in the problem management system
- Elapsed time between problem recording and resolution or escalation
- Elapsed time between evaluation and application of vendor patches
- Percent of reported problems with already known resolution approaches
- Frequency of coordination meetings with change management and availability management personnel
- Frequency of component problem analysis reporting
- Reduced number of problems not controlled through formal problem management

MANAGEMENT GUIDELINES DS10

DS10 Maturity Model

Control over the IT process **Manage Problems and Incidents** with the business goal of *ensuring that problems and incidents are resolved, and the cause investigated to prevent any recurrence*

0 Non-existent There is no awareness of the need for managing problems and incidents. The problem-solving process is informal and users and IT staff deal individually with problems on a case-by-case basis.

1 Initial/Ad Hoc The organisation has recognised that there is a need to solve problems and evaluate incidents. Key knowledgeable individuals provide some assistance with problems relating to their area of expertise and responsibility. The information is not shared with others and solutions vary from one support person to another, resulting in additional problem creation and loss of productive time, while searching for answers. Management frequently changes the focus and direction of the operations and technical support staff.

2 Repeatable but Intuitive There is a wide awareness of the need to manage IT related problems and incidents within both the business units and information services function. The resolution process has evolved to a point where a few key individuals are responsible for managing the problems and incidents occurring. Information is shared among staff; however, the process remains unstructured, informal and mostly reactive. The service level to the user community varies and is hampered by insufficient structured knowledge available to the problem solvers. Management reporting of incidents and analysis of problem creation is limited and informal.

3 Defined Process The need for an effective problem management system is accepted and evidenced by budgets for the staffing, training and support of response teams. Problem solving, escalation and resolution processes have been standardised, but are not sophisticated. Nonetheless, users have received clear communications on where and how to report on problems and incidents. The recording and tracking of problems and their resolutions is fragmented within the response team, using the available tools without centralisation or analysis. Deviations from established norms or standards are likely to go undetected.

4 Managed and Measurable The problem management process is understood at all levels within the organisation. Responsibilities and ownership are clear and established. Methods and procedures are documented, communicated and measured for effectiveness. The majority of problems and incidents are identified, recorded, reported and analysed for continuous improvement and are reported to stakeholders. Knowledge and expertise are cultivated, maintained and developed to higher levels as the function is viewed as an asset and major contributor to the achievement of IT objectives. The incident response capability is tested periodically. Problem and incident management is well integrated with interrelated processes, such as change, availability and configuration management, and assists customers in managing data, facilities and operations.

5 Optimised The problem management process has evolved into a forward-looking and proactive one, contributing to the IT objectives. Problems are anticipated and may even be prevented. Knowledge is maintained, through regular contacts with vendors and experts, regarding patterns of past and future problems and incidents. The recording, reporting and analysis of problems and resolutions is automated and fully integrated with configuration data management. Most systems have been equipped with automatic detection and warning mechanism, which are continuously tracked and evaluated.

DS11 Delivery & Support
Manage Data

COBIT

Control over the IT process **Manage Data** with the business goal of *ensuring that data remains complete, accurate and valid during its input, update and storage*

ensures delivery of information to the business that addresses the required Information Criteria and is measured by Key Goal Indicators

is enabled by *an effective combination of application and general controls over the IT operations*

considers Critical Success Factors that leverage specific IT Resources and is measured by Key Performance Indicators

Information Criteria	
	effectiveness
	efficiency
	confidentiality
P	integrity
	availability
	compliance
P	reliability

(P) primary (S) secondary

IT Resources	
	people
	applications
	technology
	facilities
✓	data

(✓) applicable to

Critical Success Factors

- Data entry requirements are clearly stated, enforced and supported by automated techniques at all levels, including database and file interfaces
- The responsibilities for data ownership and integrity requirements are clearly stated and accepted throughout the organisation
- Data accuracy and standards are clearly communicated and incorporated into the training and personnel development processes
- Data entry standards and correction are enforced at the point of entry
- Data input, processing and output integrity standards are formalised and enforced
- Data is held in suspense until corrected
- Effective detection methods are used to enforce data accuracy and integrity standards
- Effective translation of data across platforms is implemented without loss of integrity or reliability to meet changing business demands
- There is a decreased reliance on manual data input and re-keying processes
- Efficient and flexible solutions promote effective use of data
- Data is archived and protected and is readily available when needed for recovery

Key Goal Indicators

- A measured reduction in the data preparation process and tasks
- A measured improvement in the quality, timeline and availability of data
- A measured increase in customer satisfaction and reliance upon the data
- A measured decrease in corrective activities and exposure to data corruption
- Reduced number of data defects, such as redundancy, duplication and inconsistency
- No legal or regulatory data compliance conflicts

Key Performance Indicators

- Percent of data input errors
- Percent of updates reprocessed
- Percent of automated data integrity checks incorporated into the applications
- Percent of errors prevented at the point of entry
- Number of automated data integrity checks run independently of the applications
- Time interval between error occurrence, detection and correction
- Reduced data output problems
- Reduced time for recovery of archived data

MANAGEMENT GUIDELINES DS11

DS11 Maturity Model

Control over the IT process **Manage Data** with the business goal of *ensuring that data remains complete, accurate and valid during its input, update and storage*

0 Non-existent Data is not recognised as a corporate resource and asset. There is no assigned data ownership or individual accountability for data integrity and reliability. Data quality and security is poor or non-existent.

1 Initial/Ad Hoc The organisation recognises a need for accurate data. Some methods are developed at the individual level to prevent and detect data input, processing and output errors. The process of error identification and correction is dependent upon manual activities of individuals, and rules and requirements are not passed on as staff movement and turnover occur. Management assumes that data is accurate because a computer is involved in the process. Data integrity and security are not management requirements and, if security exists, it is administered by the information services function.

2 Repeatable but Intuitive The awareness of the need for data accuracy and maintaining integrity is prevalent throughout the organisation. Data ownership begins to occur, but at a department or group level. The rules and requirements are documented by key individuals and are not consistent across the organisation and platforms. Data is in the custody of the information services function and the rules and definitions are driven by the IT requirements. Data security and integrity are primarily the information services function's responsibilities, with minor departmental involvement.

3 Defined Process The need for data integrity within and across the organisation is understood and accepted. Data input, processing and output standards have been formalised and are enforced. The process of error identification and correction is automated. Data ownership is assigned, and integrity and security are controlled by the responsible party. Automated techniques are utilised to prevent and detect errors and inconsistencies. Data definitions, rules and requirements are clearly documented and maintained by a database administration function. Data becomes consistent across platforms and throughout the organisation. The information services function takes on a custodian role, while data integrity control shifts to the data owner. Management relies on reports and analyses for decisions and future planning.

4 Managed and Measurable Data is defined as a corporate resource and asset, as management demands more decision support and profitability reporting. The responsibility for data quality is clearly defined, assigned and communicated within the organisation. Standardised methods are documented, maintained and used to control data quality, rules are enforced and data is consistent across platforms and business units. Data quality is measured and customer satisfaction with information is monitored. Management reporting takes on a strategic value in assessing customers, trends and product evaluations. Integrity of data becomes a significant factor, with data security recognised as a control requirement. A formal, organisation-wide data administration function has been established, with the resources and authority to enforce data standardisation.

5 Optimised Data management is a mature, integrated and cross-functional process that has a clearly defined and well-understood goal of delivering quality information to the user, with clearly defined integrity, availability and reliability criteria. The organisation actively manages data, information and knowledge as corporate resources and assets, with the objective of maximising business value. The corporate culture stresses the importance of high quality data that needs to be protected and treated as a key component of intellectual capital. The ownership of data is a strategic responsibility with all requirements, rules, regulations and considerations clearly documented, maintained and communicated.

12. ENDNOTES AND REFERENCES

[1] In this document, "stakeholder" is used to indicate anyone who has either a responsibility for or an expectation from the enterprise's IT, e.g., shareholders, directors, executives, business and technology management, users, employees, governments, suppliers, customers and the public.

[2] In this document, "board of directors" and "board" are used to indicate the body that is ultimately accountable to the stakeholders of the enterprise.

[3] The COBIT control framework refers to key goal indicators (KGIs) and key performance indicators (KPIs) for the balanced business scorecard concepts of outcome measures and performance drivers.

[4] "The Balanced Business Scorecard — Measurements that Drive Performance," Robert S. Kaplan and David P. Norton, *Harvard Business Review*, January-February 1992

[5] "Capability Maturity Model SM for Software," Version 1.1. Technical Report CMU/SEI-93-TR-024, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, February 1993

Control Objectives for Information and related Technology (COBIT) 3rd Edition, IT Governance Institute, 1998, *www.isaca.org/cobit.htm* (All sections of COBIT, except the Audit Guidelines, can be downloaded on a complimentary basis. Print copies of all components, including the Audit Guidelines, may be purchased from the ISACA Bookstore; contact *bookstore@isaca.org* for availability.)

Board Briefing on IT Governance, IT Governance Institute, 2001, *www.ITgovernance.org/resources.htm* (May be downloaded on a complimentary basis. Print copies may be purchased from the ISACA Bookstore; contact *bookstore@isaca.org* for availability.)

Information Security Governance: Guidance for Boards of Directors and Executive Management, IT Governance Institute, 2001, *www.ITgovernance.org/resources.htm* (May be downloaded on a complimentary basis. Print copies may be purchased from the ISACA Bookstore; contact *bookstore@isaca.org* for availability.)

About the Author

Erik Guldentops, CISA, was until recently security advisor for the Society of Worldwide Interbank Financial Telecommunication (SWIFTsc) in Brussels, Belgium, where he previously held the positions of chief inspector and director of information security. SWIFT provides secure global communication to more than 7,000 financial institutions in more than 190 countries. More than five million messages valued in trillions of dollars are sent over SWIFT's network every business day. Guldentops is advisor to the board of thIT Governance Institute and an executive professor in the management school of the University of Antwerp, Belgium, where he teaches on the subjects of IT security and control, IT governance and risk management. He initiated and has headed up the developments of Control Objectives for Information and related Technology (COBIT) since the early nineties and is currently the chair of Information Systems Audit and Control Association's COBIT Steering Committee.

Implementation of the COBIT-3 Maturity Model in Royal Philips Electronics

Alfred C.E. van Gils
Philips International BV
Corporate Information Technology
Eindhoven, The Netherlands

Abstract: Philips has an ongoing Business Excellence program for all business functions, including IT. As part of this program the COBIT standard from the IT Governance Institute™ is used to do maturity (self) assessments on IT processes. The program is implemented worldwide using 10 steps including, a workshop approach, improvement management, relation to the internal control framework, organisation and communication.

Key words: IT governance, audit, COBIT, ISACA, internal control, maturity

1. ROYAL PHILIPS ELECTRONICS

Royal Philips Electronics is a global electronics company established in 1891. Headquartered in Amsterdam, The Netherlands, it has a multinational workforce of more than 225,000 and offers sales and services in 150 countries. Listed on the New York, London, Amsterdam and other stock exchanges, Philips had net income in 1999 of Euro 1.8 billion (US $1.9 billion). Divisions of Philips include Consumer Electronics, Lighting, Semiconductors, Medical Systems, Domestic Appliances & Personal Care and Components.

2. COBIT CONTROL OBJECTIVES FOR INFORMATION TECHNOLOGY

Control Objectives for Information Technology (COBIT), was originally an audit framework. It is now a comprehensive IT Governance framework, organised around 34 high-level processes and control objectives. COBIT is released by the COBIT Steering Committee and the IT Governance Institute™

COBIT recognises the following processes:

Planning and Organisation
PO1 Define a Strategic IT Plan
PO2 Define the Information Architecture
PO3 Determine Technological Direction
PO4 Define the IT Organisation and Relationships
PO5 Manage the IT Investment
PO6 Communicate Management Aims and Direction
PO7 Manage Human Resources
PO8 Ensure Compliance with External Requirements
PO9 Assess Risks
PO10 Manage Projects
PO11 Manage Quality

Acquisition and Implementation
AI1 Identify Automated Solutions
AI2 Acquire and Maintain Application Software
AI3 Acquire and Maintain Technology Infrastructure
AI4 Develop and Maintain Procedures
AI5 Install and Accredit Systems
AI6 Manage Changes

Delivery and Support
DS1 Define and Manage Service Levels
DS2 Manage Third-Party Services
DS3 Manage Performance and Capacity
DS4 Ensure Continuous Service
DS5 Ensure Systems Security
DS6 Identify and Allocate Costs
DS7 Educate and Train Users
DS8 Assist and Advise Customers
DS9 Manage the Configuration

DS10 Manage Problems and Incidents
DS11 Manage Data
DS12 Manage Facilities
DS13 Manage Operations

Monitoring
M1 Monitor the Processes
M2 Assess Internal Control Adequacy
M3 Obtain Independent Assurance
M4 Provide for Independent Audit

For each of the 34 processes, COBIT specifies:
- Detailed control objectives (318 in total)
- Key goal indicators
- Key performance indicators
- Critical success factors
- Maturity models

Key to the implementation was the addition of maturity models in COBIT. The maturity model in COBIT is taken from the Software Engineering Institute's Capability Maturity Model (CMM) for Software Engineering [Pauli, M.C., Curtis, B., Chrissis, M.B. and Weber, C.V, 1993]. CMM defines the following maturity stages:

1) **Initial** The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort.
2) **Repeatable** Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.
3) **Defined.** The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.
4) **Managed.** Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.
5) **Optimizing.** Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

It is not the intention of this article to give a full description on COBIT. Detailed information can be found on http://www.isaca.org.

3. COBIT IN PHILIPS

The Internal Audit department within Philips has a long-standing tradition of using COBIT and encouraging CISA® (Certified Information Systems Auditor™) certification for IT audit staff.

In addition to its extensive Internal Audit implementations, the Corporate IT Department of Philips International used the COBIT framework when participating in two company-wide initiatives. Support at Supervisory Board level was achieved by linking with the following executive programs:

1. **The BEST (Business Excellence through Speed and Teamwork) quality improvement program.**

This program has strong, visible support from senior management and is one of the five top items on the Management Agenda. As part of this program, Philips uses Balanced Scorecards and "Process Survey Tools" to measure the relative maturity of business processes within organizations. Process Survey Tools now exist for key business processes:

- Supply Chain Management
- Manufacturing Maintenance
- Manufacturing Operations
- Purchasing
- Demand Generation
- Innovate to Market

And also for some functional areas:

- Finance and Accounting
- Human Resource Management
- Information Technology

The Process Survey Tool for IT is based on the COBIT 3rd Edition. The maturity definitions from the COBIT Management Guidelines were copied and published in the format of a Process Survey Tool with only some minor formatting changes in order to align with other business functions.

2. **The Statement on Business Controls program.**

Each organizational unit within Philips issues a formal annual statement on the quality of internal controls. The process is based on controlled self-

assessment and is subject to validation by internal and external auditors. It is consolidated into the Annual Report's internal control statement and therefore has the full support of senior management. As part of this process, organizations are also asked to complete a section on IT. The IT section of the Statement on Business Controls is also based on the COBIT control objectives. In fact, completing a COBIT maturity assessment is considered sufficient basis for submitting an IT internal control statement provided that:

- The self-assessment scores are based upon sufficient supporting evidence.
- A proper action plan addresses shortcomings on all material processes scoring in or below the "Initial/Ad Hoc" range of maturity. This means that if the process performs ad hoc it is regarded as insufficient from an internal control point of view.

4. IMPLEMENTATION

The BEST program's Process Survey Tool for IT was initiated in 1999 and developed during the second and third quarters of 2000. At the beginning of the project, only the second edition of COBIT was available which did not yet include maturity models. The COBIT framework was used for the pilots, but the general ISO15504 standard (which also includes a similar maturity model), was used for establishing maturity levels.

Key to the implementation of COBIT is a workshop approach, based on self-assessments.

- The course of assessing 34 processes takes one to one and a half days;
- The process of scoring should preferably be done by a group of 6 to 8 persons;
- The group should consist of IT staff and representatives from the user community (depending on the user organisation key users, F&A representatives);
- The actual scoring should cover the following items:
  - Introduction and training into COBIT and maturity levels;
  - Scope definition (decide what to score and what not to score);
  - Scoring per process based on a self-assessment, i.e. individual scoring, discussion and consensus building;
  - Defining improvement actions and levels for the next year.
- The use of a facilitator for the process is recommended

After undergoing testing in 10 pilot workshops, the Process Survey Tool was released with two implementation paths:

- Per Product Division, where one contact person per division and/or business group was responsible for rollout
- Per region (i.e. Asia Pacific, East and West Europe, Latin America and North America), where rollout was facilitated per country

In July 2000, the 3rd edition of COBIT, which now included a fully worked out maturity model per process, was released. COBIT 3rd edition is a significant improvement.

Elements for implementing COBIT based Maturity Self Assessments

A comprehensive implementation of COBIT addresses the following 10 elements, not all of which have currently reached completion. In addition, many of the specific details are left to the discretion of different divisions, business groups and countries.

1. Process for Initial Assessments
2. Process for Re-Assessments
3. Relation to Balanced Scorecards
4. Relation to SBC and Internal Control Framework
5. Communication of Action Plans
6. Consolidation of scores
7. Exchange of Best Practices
8. Process for calibration of scores
9. Scoring Directives and Guidance of Improvement Actions
10. Organisation and communication

4.1 Process for Initial Assessments

The process for doing maturity self-assessments is built upon the following elements:

- The process uses a facilitated self-assessment workshop based on COBIT 3rd edition maturity models
- The total duration is 1 - 1,5 days including introduction, training and definition of improvement actions.
- The optimal group size is 6-8 people, including one or two representatives from the user community.
- The scoring is based on consensus discussions
- A presentation and reporting standard

During the initial assessment, the definition of a set of good improvement actions is key for continuation of the program the next year. After the first

assessment round, participants are asked to select a limited number (3-5) from the 34 COBIT processes, where improvements should be focussed on the next year. The consolidated votes of the group are used to focus on a limited number of tangible improvement actions up to the next assessment round.

Improvements should balance the materiality, customer impact, cost and maturity score of a particular process. Consequently, the improvement program does not automatically have to address the lowest scoring processes in the first place. It may be more important to achieve a very high maturity score on a critical process ("Ensure Systems Security"). That however depends on the type of business, information and environment.

As a rule, processes scoring in the "Initial/Ad Hoc" range of maturity or lower, are also recommended for improvement.

4.2 Process for Re-Assessments

As a first step, an organisation needs to define the length of time to the next assessment. Experience with re-assessments in Philips is still limited and tends to use the following agenda:

- A 2-3 hour session
- The period is set to a maximum of +1 year from the initial assessment
- Setting of agreed actions and other improvements during the period
- Definition and assessment of new scores for applicable processes
- Definition of new improvement actions, scores and re-assessment period

4.3 Relation to Balanced Scorecards

The use of Balanced Score Cards for IT (BSC-IT) is starting to develop. Whenever a BSC-IT is in use, an organisation also needs to address how to include the COBIT based maturity assessments in the processes part of the scorecard. An organisation then needs to address how to include maturity targets in the BSC-IT. Examples are:

- The PST-IT done/re-assessed
- A PST action plan available
- The PST score on average X, or PST score 80% higher than Y and no score lower than Z

The relation between a maturity assessment and the BSC-IT can be directed towards a certain scoring target, or non-directive, i.e. checking whether or not the assessment took place and is properly followed-up.

4.4 Relation to the Internal Control Framework

Any implementation of COBIT should also address the relation to the organisation's formal internal control framework and auditors.

In consultation with the external auditors it was concluded that conducting a COBIT based maturity assessment is sufficient basis for submitting a control self-assessment statement to the external auditor, provided there is sufficient evidence in support of the scores.

From an internal control point of view, processes scoring in the "Initial/Ad Hoc" range of maturity or lower are not acceptable and should be addressed by (at least) defining proper follow up actions.

4.5 Communication of Action Plans

Another item in the implementation of COBIT based maturity assessments, is whether or not action plans resulting from assessments need to be communicated. If so an agreed format and communication tool has to be established.

Are action plans subsequently evaluated on their content if they exist? In practice the following variations can be found in different divisions, business groups and countries:

- Action plans are not communicated at all;
- There is a registration if an action plan has been established;
- Action plans are consolidated into an overall action plan and the content is known to all concerned.

If, as in the latter case, the content of action plans is also part of the organisation's implementation plan for COBIT, the logical consequence is to use assessment results to identify areas for improvement and guide improvement programs.

4.6 Consolidation of scores

A related issue is whether or not scores should be communicated and consolidated.

If an organisation decides to consolidate scores from self-assessments, a supporting process needs to be developed with proper tooling. Experience shows that consolidation of scores can be of some value in the sense that:

- It provides a benchmark for organisations against which scores can be evaluated, not in a precise statistical sense but more in support of a general outlook of being "in-line" within a business community.
- Consolidation of scores provides a benchmark in order to track, identify and benchmark changes over time.
- A database of scores is an efficient way of not only identifying "best practices", but also problem areas where scores drop in or below an "Initial/Ad Hoc" level of maturity.

Consolidation of scores just by the figures has some limitations and drawbacks. The scope of the assessment needs to be taken into consideration when comparing scores. An assessment may have taken a very limited scope, say SAP R/3 Managed Operations, making it very difficult to compare it with a full scope assessment including legacy systems and office support.

4.7 Exchange of Best Practices

Implementing COBIT-based maturity assessments may lead to the identification of "Best Practices" in an organisation. Firstly, there needs to be at least some way of communicating scores. Then a definition of which processes qualify as a Best Practice is needed to determine whether this is the highest score available or the score above a certain value

Best practices have been identified within Philips, for almost all 34 COBIT processes based on a rule of thumb that, a score should be at least "Managed and Measurable", with some scores reaching "Optimised" levels of performance.

Following the identification of Best Practices the process for exchanging information can be addressed by means of meetings, presentations or the Intranet.

Another important consideration is the validity of scores. So far, there is no process for verification of cases that have been submitted as a Best Practice in Philips and the information is taken at face value. In a more advanced stage of implementation, audits or peer-audits may become useful tools in the actual verification and validation of best practices within an organization.

4.8 Process for calibration of scores

Since the resulting scores are based on self-assessments, a process needs to be in place to address the issue of calibrating scores, in order to ensure that assessment results present a true and fair view of the actual maturity levels.

Comparable processes within CMM [Pauli, M.C. et. al. 1993] are highly evolved into an extensive set of assessor training, curricula and audit programs, the results of which may even be published [Mark C. Paulk 2000].

Maturity definitions for COBIT processes are relatively new (July 2000) and supporting processes and applications are still far from the level of institutionalisation that has been achieved in CMM.

In practice, results from self-assessments are taken at face value. Still, there are some considerations that may result in more reliable scores:

- The first control is the assessor group composition. The recommendation is to have a balanced group comprising of IT, as well as user representatives. Involving different stakeholders in a group, results in a more balanced and reliable scoring process.
- A second control is to have a facilitator, who has experienced more than one assessment.
- A final control is to implement an audit or peer-audit process on the actual scores. Results are subject to audit as they are part of the Philips Internal Control Framework. Assessments that were combined with audits show that the assessment results did in fact provide a fair and true view, although based on a limited experience. There are some organisations that include peer-auditors in the re-assessment exercises.

4.9 Scoring Directives and Guidance of Improvement Actions

The scoring directive is optional and may be linked into the BSC-IT. It can use a very fixed format (e.g. all scores at least on a "Defined Level" of maturity) or a more open format (80% of scores at least on a "Defined Level").

Philips organisations use many different formats, from no directive at all to a very fixed quantitative target.

As stated earlier, any scoring directive should balance the materiality, customer impact, cost and maturity score of an IT process in a particular business operation.

The question then becomes, whether improvement efforts can focus beyond the level of independent self-assessments to broader improvement efforts. Typical examples in this area are ISO and ITIL implementation projects, which focus on a broad scope of IT and COBIT processes for improvement. If carried out properly, they can leverage the performance of IT processes to defined and higher levels of maturity in a single integrated effort.

4.10 Organization and communication

The final and most important implementation point is to define and set up a proper structure for communication and organization.

The introduction of COBIT Maturity Assessments in Philips is linked to a highly visible Business Excellence Program with the full support of senior managment. This has been a critical factor in the successful implementation.

Next has been the selection and development of an adequate tool for maturity self-assessment. This was done in a joint effort including representatives from Philips divisions and results were based on 10 pilot assessments.

The roll out was done per division as primary business drivers. In addition the program was also introduced worldwide as part of regional IT meetings. In some cases a facilitator provided guidance in order for participants to conduct their own workshops.

Communication and organisation is embedded in each division and business unit with varying degrees of formality and institutionalisation. In addition there are regional programs where synergy requires a concerted effort.

5. CONCLUSION

Using COBIT to establish organizational capabilities on a maturity level, gives a clear indication of where improvement is possible and how to achieve it.

The experience in Philips includes more than 100 assessments worldwide, in a mixture of organizations and cultures.

One feature of COBIT is that it is flexible and allows users to customise applications. Philips developed its approach based on internal group workshops including presentations, training and scoring material.

COBIT has a number of outstanding benefits, in particular:

- It is an open standard;
- The documentation is clear and understandable;
- The professional organization; the IT Governance Institute™ is leading the development of COBIT by using a broad range of international references (e.g. ISO standards);
- COBIT is part of a larger program; it is kept up to date, more detailed information is available, plus there are translations, training programs and the related CISA certification;
- COBIT version 3 brings together three extremely rich methodologies in one framework, namely:
  1. The 34 COBIT processes, along with key performance indicators, key goal indicators, critical success factors and control objectives;
  2. The Business Balanced Scorecard for IT and
  3. The maturity model derived from the Capability Maturity Model for Software Engineering.

In addition, generally applied frameworks for auditing such as COSO, Control Self Assessment and quality frameworks based on ISO are well integrated in the COBIT methodology.

Some other points with respect to using COBIT for maturity assessments are:

- The language used in COBIT assumes a certain background, affinity and fluency which is not available in all (IT) groups. COBIT has a strong focus on internal control and may not be perceived as self-explanatory as is required for doing self-assessments. It may be required to establish COBIT "champions" and/or training programs to increase the level of familiarisation.

- Implementing COBIT needs to explicitly address application context. It is important to do a scope definition (what applications, processes, functions and business processes will be assessed). The maturity model also assumes a certain size of IT operations and typically only applies to operations of a certain critical mass.
- Finally, there are huge cultural differences in self-assessment scoring. Without being specific, some cultures take written documentation as law while others have a much more pragmatic and lenient attitude towards COBIT. There is also cultural diversity in the degree of openness in group sessions and affects of the particularities of group composition. There is no simple answer to any of these contextual factors.

The Philips rollout per product division and region is ongoing, and concrete activities include providing ongoing support for COBIT 3rd Edition-based assessment workshops

The rollout itself will focus on institutionalising and grounding the ten steps into the organisation.

After the rollout, Philips will focus on:
- Assessing actual outcomes of the process (based on key goal indicators and maturity levels);
- Identifying problem areas (for IT processes with low maturity scores);
- Defining best practices ('defined process' maturity level and higher);
- Improving management processes and actions;
- Benchmarking score.

6. REFERENCES

Information Systems Audit and Control Association & IT Governance Institute. **COBIT. Control Objectives for Information Technology.** 3rd Edition. Rolling Meadows, IL. ISBN 1-893209-13-X. http://www.isaca.org/

Kaplan, R. S., and Norton, D. **The Balanced Scorecard: Measures that Drive Performance.** Harvard Business Review 70, no. 1 (January-February 1992): 71-79.

Mark C. Paulk, Dennis Goldenson, and David M. White, "**The 1999 Survey of High Maturity Organizations,**" Software Engineering Institute, Carnegie Mellon University, CMU/SEI-2000-SR-002, February 2000.

National Commission on Fraudulent Financial Reporting; COSO Treadway Commission. **Report of the National Commission on Fraudulent Financial Reporting.** COSO, October 1987. http://www.coso.org/.

Pauli, M.C., Curtis, B., Chrissis, M.B. and Weber, C.V. **Capability Maturity Model for Software, Version 1.1.** Technical Report, CMU/SEI-93-TR-024 ESC-TR-93-177 February 1993. http://www.sei.cmu.edu/cmm/

PART FOUR. VENDOR WHITE PAPERS

Business Process Security

Managing the new security challenge with X-Tra Secure

Bartjan Wattel
ThunderStore BV
's-Hertogenbosch, The Netherlands
info@thunderstore.com

Abstract: The process of security is destined to fail if it does not protect the process of business. Despite a record investment in security by corporations around the globe, the frequency, variety and financial cost of attacks continues to rise. And while most organizations focus their defenses on external threats from hackers and virus authors, the real security challenge is much closer to home. If security professionals want to avoid falling into the trap of trying to protect everything all the time, they need to refocus their defenses on authorized users and the resources they have access to.

Key words: business process security, X-Tra Secure, security policy, monitoring, privileges

1. THE HOLE IN THE FENCE

The rush to protect the enterprise from the Internet has resulted in a widely adopted security strategy that is full of gaping holes, and one which often has security administrators facing the wrong way.

Traditional security thinking will always be flawed for two fundamental reasons:

- It assumes that the most serious attacks come from beyond the corporate perimeter;
- It assumes that the most serious attacks will come from hackers, virus authors, and other unauthorized outsiders.

For this reason most security spending had been focused on keeping hackers and virus authors from penetrating the outermost perimeter. And most of the technology focus has been on security tools that will defend these types of attacks, most notably network monitoring, intrusion detection, firewall, and anti virus.

But the significant increase in attacks on corporate networks has exposed perimeter security as little more than fighting wild fires. As the security team rushes to patch a newly discovered breach in the perimeter, two more appear.

And just as they think they have accurately mapped the perimeter, it changes again. The growing army of mobile, home, and tele workers has extended the corporate perimeter into hotels, taxis, and employee living rooms.

According to IDC there are now more than 38 million home and mobile workers in the United States, and growing fast. And every time an employee takes an unprotected laptop out of the office, they punch a hole straight through the perimeter that might only be discovered by a successful attack.

The perimeter will continue to grow, to incorporate branch and international offices, and to accommodate the need for customers, partners, and suppliers to access corporate intranets. And the increase in use of wireless devices and wireless networks will not only reshape the already fluid perimeter, it will introduce an entirely new set of security vulnerabilities already discovered in wireless protocols.

And while security professionals have long recognized the importance of increasing security awareness amongst employees, the focus of this awareness is usually on external intruders, and not on the employees themselves.

The role of security policy has been recognized as the foundation of any effective security strategy, but security professionals have yet to find a way to enforce that security policy around the clock, on every employee, and for every business process, short of stationing a security guard at every desktop, laptop, and wireless device.

2. THE USER SECURITY TRAP

Perimeter defenses like firewalls and intrusion detection systems are essential components in any defense, but their effectiveness is severely limited when the threat is trusted, and is located behind the perimeter.

"Employees are your security," claimed former hacker Mudge in an interview with CIO magazine. He was echoing a widely held view, even in the hacking community, that employees and authorized users now play a critical role in security. According to the most recent security survey by the Computer Security Institute and the FBI, 86% of respondents cited trusted insiders as a major security concern.

An earlier report from the CSI found that while the average cost of an attack by an outside hacker cost $57,000, the average cost on an attack by a trusted insider was $2.1 million, or thirty-six times as much.

And while many organizations acknowledge the role employees must play in protecting the enterprise, that role is usually limited to better vigilance and awareness against those very same external attacks on the perimeter.

Few companies have been able to successfully address the more important user security challenge – that of their own everyday behavior in the workplace, and their compliance with security rules and policies.

According to a recent report by research group Forrester "despite an expected 300 percent spending increase on information technology security spending over the next four years, bad decision-making will leave U.S. companies almost as vulnerable to security breaches as they are today."

Why? The report explains that "rather than focusing on key business assets, companies will waste billions of dollars on external security spending, falling into the trap of trying to protect everything."

The view is shared by research firm Gartner Group: "Despite the recent notoriety of external attacks, internal threats continue to pose the greatest threat to the exposure and compromise of sensitive corporate information. Identifying vendors who offer a realistic solution to this very complex monitoring problem will be the next challenge facing all security professionals."

By overlooking the security vulnerabilities of their own automated business processes, says Forrester, companies put themselves at great risk.

"A company is not secure if it installs a firewall system, but does not have application security in place to ensure an approved user does not fraudulently wire money to his Swiss bank account or does not trade above his daily limit," said Lorenzo De Leon, CEO of Saecos Corporation, a Chicago-based security firm focused on the financial community.

3. SECURITY IN AN IDEAL WORLD

How much easier security would be if policies could be created and tested with ease, distributed instantly to all users, and never challenged, ignored, or circumvented by a single user.

For any security policy to be worth the time invested in developing it, it must meet certain minimum requirements. It must be easy to create, and administrators must be able to test it thoroughly before it's cast in stone. There must be an easy way to distribute the policy to potentially tens of thousands of users, across dozens of offices, speaking many different languages.

Users must have no choice but to comply with every element of the policy, whether they like it or not. And if they don't like it, it must be explained to them the consequences of any alternatives.

Ideal Security Scenarios

- An employee attempting to send in clear mode a sensitive document that is supposed to be sent encrypted, might be halted, reminded of the specific security rule relating to sensitive documents, told why the rule is important, and informed that the document cannot be sent unencrypted. If the user agrees, then the document is automatically encrypted and sent. The policy is effective, the user is educated, and security is maintained.
- An employee attempting to open a databases file with anything other than Microsoft Access (a policy rule), when the rules state that Access is the only options available, is unable to open the database.
- An employee seeking access to specific applications after 7pm when the rules forbid any access to such applications after 5 pm, is denied access, informed why, and a record of the event is sent to security administrators.

- An employee who downloads a file from the internet and attempts to save it on a local hard drive, when the relevant security rule insists that such files be stored only on a specific server, will be unable to save the file to that drive.
- Any document containing sensitive information such as customer credit card data is automatically encrypted when an employee attempts to transfer the document to a removable disc, or send it across the internet or any non-secure channel.

While these scenarios may seem too unreasonable to realize, success in making them a reality could forever change the way users impact security.

4. THE IMPORTANCE OF BUSINESS PROCESS SECURITY

Core to the foundation of every enterprise are its business processes. Computer systems, networks, even the workforce are just contributors to the processes that create product, drive revenues, and generate profits. Networks, intranets, and computer systems simply support these processes.

The effects of attacks on networks and computer systems can always be mitigated, but attacks that compromise the availability and integrity of critical business processes can cause irreparable harm to very core of the enterprise. And the applications and information that enable these business processes are vulnerable to the very people entrusted with their safe use.

According to Forrester "businesses will focus their security efforts on trying to hold onto customers, ignoring the more potent threats waiting to pounce from inside the company itself....despite multibillion-dollar spending, firms will miss the new challenge: business-process security."

In order for security to become truly effective, it must change its focus from the outer perimeter, and instead target security at the applications and data that enable all critical business processes.

To achieve this, the enterprise must focus on the trusted user, to minimize the vulnerability created by inappropriate user behavior, and to maximize the effectiveness of security policies designed to protect business processes from dishonest, malicious, careless, or reckless users.

"There is an emerging need for auditing and security solutions that enable companies to monitor the authorized activity of their employees, contractors and business partners," according to Gartner.

"There is a new security equation. It's not just about keeping the bad guys out; it's also about enabling the good guys to do what they are entitled to do, and only what they are entitled to do," said Saecos CEO De Leon.

5. THE IMPACT OF POLICY ON BUSINESS PROCESS

A good security policy is not difficult to create, and security professionals have a wealth of guides, templates, tools, and experience with which to create the perfect policy. The real challenge lies in making policy work, a process that must include testing, distribution, compliance monitoring, enforcement, education, and updating.

In that ideal security world, a policy designed to protect business processes should:

- Focus on how users access and use applications, information, and business processes;
- Eliminate the need to simply trust users to do the right thing;
- Automatically monitor all access and use for compliance;
- Correct inappropriate behavior as it happens, without adding to the administrative overhead or burdening security;
- Teach users why the behavior was incorrect, by explaining to them in real time the official policy for that action;
- Test and simulate any policy for effectiveness before it's launched, to avoid confusing the user;
- Gather the necessary evidence and forensics, in case the user continuously ignores policy rules, or engages in criminal behavior;
- Automate this process to free security teams from the task of manually identifying policy breaches or inappropriate behavior, halting the action, informing the user, and correcting any harm.

Given these significant challenges it's easy to understand why business process security is so essential to business integrity, and why the effective enforcement of policy is so difficult

6. THE X-TRA SECURE RESPONS

From its roots in the early anti virus industry, ThunderStore recognized the importance and value of linking individual behavior to the individual. The firm also recognized a major security challenge – to find a way to enforce the correct behavior on individual users and employees, in compliance with corporate policies, but to do so in a way that did not burden security administrators or users, and which didn't impact the pace and drive of the business.

Their response is called **X-Tra-Secure**, reflecting its position as a complimentary defense to existing security solutions to manage a significant threat - protecting the integrity of information, applications and business processes from careless or malicious actions by authorized users.

X-Tra-Secure solves the current disconnect between desired activities and actual behavior of "authorized users" through enforcement of agreed security rules that ensure the necessary compliance with corporate information, business and security policies.

At the core of the technology is the X-Tra-Secure Framework, which logs, monitors and analyzes the activities of "authorized users" and then enforces the desired behavior as it happens." X-Tra Secure™ enforces an organization's policies in real time, with or without notifying the user.

By using X-Tra- Secure™, companies no longer have to depend solely on the knowledge, behavior, good will and discipline of users. Instead, X-Tra-Secure™ keeps users in compliance with policy and holds them accountable.

7. THE BENEFITS OF X-TRA SECURE

- Delivers Business-Process Security by applying a policy-based management system that monitors and controls user activities and assigned privileges against corporate policies, standards and guidelines;
- Promotes policy awareness;
- Guarantees compliance;
- Encourages and enforces desired behavior;
- Eliminates dependence on user cooperation;
- Eliminates dependence on user cooperation;
- Reduces the total cost of ownership;

- Enables enterprises to deliver accountability above and beyond authorization and authentication.

8. HOW X-TRA SECURE WORKS

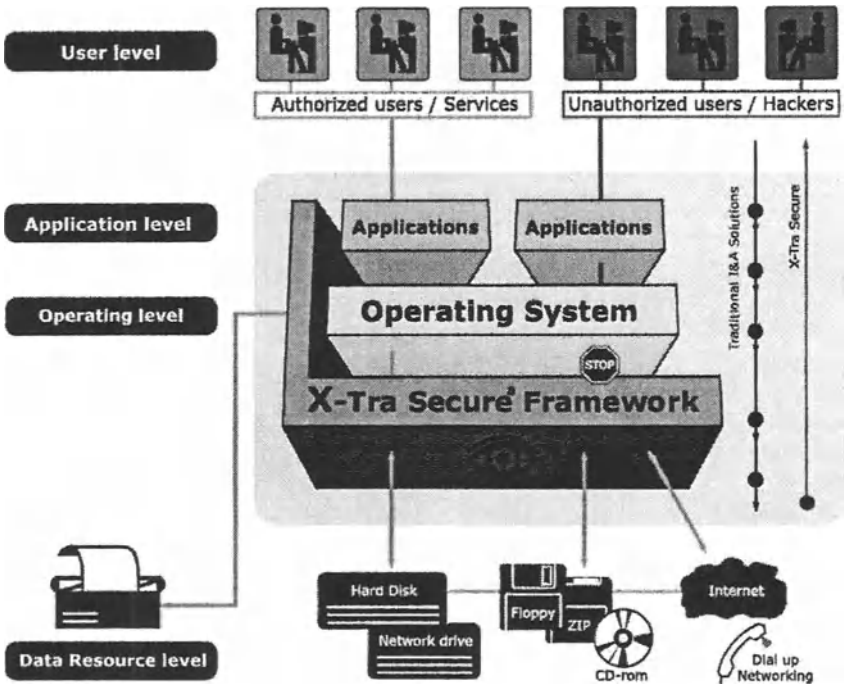
X-Tra-Secure actively monitors, logs, analyses and enforces security policy in business information systems. The **X-Tra-Secure** Framework monitors and manages how users are allowed to handle corporate information and data, by comparing user activities and assigned privileges against a set of rules that are based on corporate information and security policies.

Activities that attempt to violate these rules (such as unauthorized copying and/or printing of confidential files, or transmission of unencrypted information) are prevented. As a result, companies do not have to rely on voluntary employee compliance with information, business and security policy. **X-Tra-Secure** encourages and enforces policy compliance automatically.

9. THE X-TRA SECURE™ FRAMEWORK

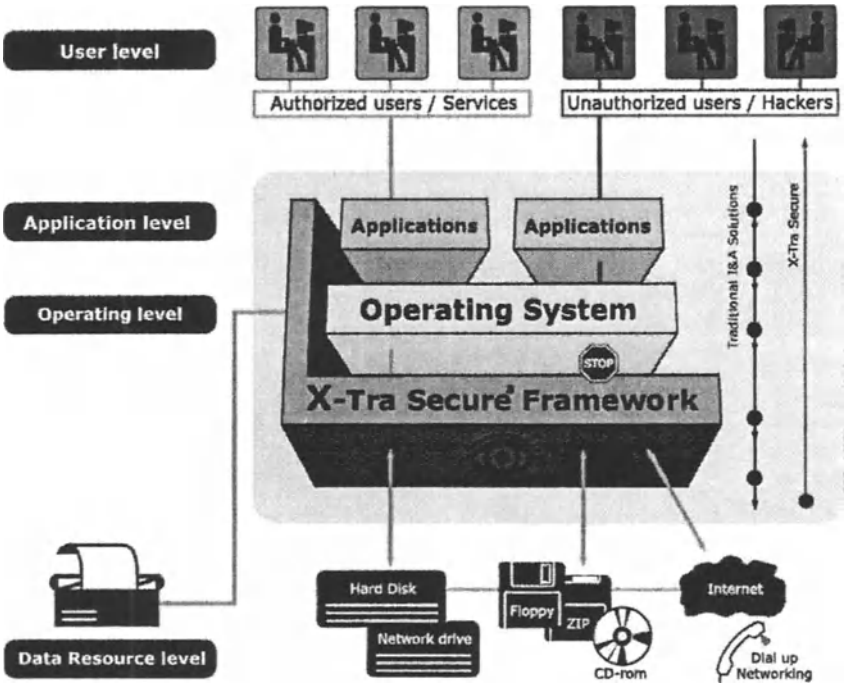
The **X-Tra Secure™** framework occupies a strategic position below the network operating system level where it is able to monitor and analyze all data before it reaches the operating system. For each operation, **X-Tra-Secure™** poses questions such as: Should this file be copied, moved, or deleted? Should the file be stored locally? Should it be opened by a particular application?

Should the contents of a Website be stored locally? If an assigned policy rule attached to a file attempts to perform a violation, **X-Tra-Secure™** will immediately intercept and stop the request, before it reaches the operating system.



X-Tra-Secure™ Technology can log, monitor, analyze and enforce pre-defined policy and rules with respect to any type of data or application. For example, it can identify critical words (and or semantics), sentences and strings (e.g. credit card numbers) before files are opened, saved, read, copied, deleted or sent. This capability can be coupled with X-Tra-Secure's ability to attach security rules to files. As a result, certain rules can be invoked if the file is determined to contain critical pieces of information.

This combination of existing features, combined with a suite of new tools and planned enhancements, makes X-Tra-Secure the perfect complement to existing security mechanisms, and the Number One choice for managing Business Process Security.


User level
Authorized users / Services
Unauthorized users / Hackers
Application level
Applications
Applications
Operating level
Operating System
STOP
X-Tra Secure Framework
Traditional I&A Solutions
X-Tra Secure
Hard Disk
Network drive
Floppy
ZIP
CD-rom
Internet
Dial up Networking
Data Resource level

The Information Integrity Imperative

Madhavan K. Nayar
Unitech Systems Inc.
1240 East Diehl Road
Suite 300, Naperville, IL 60653-1439, USA
mnayar@unitechsys.com

Abstract: This white paper examines several dimensions of information integrity. It then explains why it is so imperative that we mobilize appropriate resources to create and implement a new, universal framework for achieving information integrity.

Key words: information integrity, data quality, imperative, pervasive, information integrity space

1. INTRODUCTION

Information integrity is the dependability or trustworthiness of information. More specifically, in the context of today's information systems, it is the accuracy, consistency and reliability of the information content, process and system.

Every organization in business, government and society is concerned about information integrity and impacted by information integrity failures. Until now, though, the issue of information integrity has not been recognized as a pervasive, universal phenomenon, despite the fact that it costs the economy hundreds of billions of dollars. This situation offers opportunities for research, education and improvement.

In the past, we have been able to achieve an acceptable level of information integrity because we either had the time to detect and correct

information integrity failures or the failures were sufficiently isolated to prevent wide-scale contamination (due to the loosely coupled or discontinuous nature of our systems and processes). However, several recent trends in information technology and the business environment are dramatically raising the seriousness of the problem and the urgency of finding radically different approaches and solutions for achieving information integrity.

2. DATA QUALITY VERSUS INFORMATION INTEGRITY

Many organizations have been engaged in major initiatives to build data warehouses of customer information. "Data quality" has become part of the vernacular of the information systems professionals involved in these initiatives. They also use such terms as "Data integrity", "Information quality", "Information integrity", "Data accuracy" and "Information accuracy" to refer to various aspects of the usefulness, usability and integrity of data and information. In this document, we use the term "Information integrity" to describe the accuracy, consistency and reliability of the information content, processes and systems. In this sense, information integrity is the basis and prerequisite for the usefulness and usability of information. It is a specific, objective attribute that lends itself to standards, measurement and improvement.

3. THE INFORMATION INTEGRITY SPACE

Conceptually, information integrity can be viewed and understood from many different perspectives or dimensions, ranging from purely technical issues such as the accuracy of prices recorded by a supermarket scanner to more philosophical issues such as the integrity of financial reports when they affect personal gains or losses through incentive plans and share prices.

On a more practical level, every organization performs a variety of activities and spends a significant amount of resources to ensure the integrity of the information it delivers to customers, partners, suppliers, employees and shareholders. While these activities and resources reside in different parts of the organization, they are all aimed at ensuring the accuracy, consistency and reliability of information and information systems.

They may be viewed as part of an "Information integrity space", the dimensions of which may include activities related to:

- Prevention, monitoring, detection, verification and correction of information errors;
- Security, audit and control;
- Protection against corruption of information due to accidental failures or deliberate fraud;
- Data scrubbing and cleansing in the creation of data warehouses;
- Design, development, operation, use and maintenance of information systems;
- Conversion of existing systems due to mergers, acquisitions and consolidations;
- Modification of existing systems to accommodate changes such as new legislation or new technology;
- Information integrity requirements of specific industries such as banking, finance, telecommunications, engineering, transportation, defense, etc.

Because different functional areas within an organization perform these activities, each addresses information integrity as an isolated, unique challenge, even though the underlying problems are inter-related and could be solved more effectively through an enterprise-wide approach. Consequently, many organizations are paying an inordinately high price for the level of information integrity they are able to achieve.

4. COSTS OF INFORMATION INTEGRITY

Most organizations have systems and procedures to verify the integrity of information they receive from both external and internal sources. These systems and procedures are usually custom-developed, unique to each organization and very labor-intensive. Many information integrity related activities are performed in all parts of the organization, and thus are often duplicated.

Although there are no readily available studies about the total cost of information integrity within an enterprise, many executives and industry experts concur that such costs would, as a conservative estimate, range from 1 percent to 5 percent of revenue. Effective enterprise- and industry-wide approaches to information integrity using standards, best practices and innovative technology would significantly reduce this cost.

5. INFORMATION INTEGRITY RISKS

Many software vendors and information technology specialists claim their products and systems are not susceptible to information errors. However, all organizations are increasingly vulnerable to errors caused by information integrity failures, simply because such failures are primarily due to factors outside their systems and beyond their control. These exogenous factors include:

- Change: organizational structure, legislation, personnel, hardware, software, etc.;
- Complexity: the number and variety of components and interfaces; the volume and speed of information processing;
- Communication: the risk of duplicate, missed or partial transmission or receipt of information;
- Conversion: data consolidation, decomposition or transformation;
- Corruption: accidental failures and deliberate fraud.

When information integrity failures do occur, they can be enormously expensive, embarrassing and sometimes fatal. An enterprise-wide information integrity architecture, which minimizes an organization's exposure to the aforementioned exogenous factors, could effectively manage this risk.

6. VALUE OF INFORMATION INTEGRITY

Information integrity can create significant added value to organizations, especially those:

- Whose primary product is information;
- Whose core business processes are driven by information; or
- Who use their information base to develop new products, enter new markets or launch other strategic initiatives.

Organizations can realize this added value by proactively ensuring the integrity of the information about customers, markets, products and processes over a sufficient span of time.

7. TECHNOLOGY EVOLUTION

The rapid evolution of technology has steadily increased the impact and importance of information integrity, while at the same time making it

possible to implement sophisticated solutions for achieving information integrity.

Stand-alone mainframe systems, for the first time, made it possible to automate the validation of input data and the programming of internal controls to verify the integrity of the processing logic. However, the implementation of the validation and controls was often *ad hoc*, arbitrary and ineffective. Many systems failed or produced erroneous results, and therefore required frequent reruns. These failures, however, did not always cause serious business disruptions, because systems had enough slack in the batch processing "window" to allow errors to be corrected.

The implementation of online systems for customer service shrank the batch processing window and made information integrity failures immediately visible to customers. This led to the implementation of automated balancing and reconciliation of the batch processing outputs. Despite these controls, the complexity of the online systems- and the lack of continuity between batch and online controls - exacerbated information integrity problems.

The introduction of client/server architectures further added to the complexity of information systems by increasing the number of hardware / software components and requisite interfaces. Distributed computing also increased the volume and frequency of data communication, thereby introducing the potential for duplicate, incomplete, missing or delayed data transmissions. The development of data warehouses, because it entails the conversion and consolidation of data from many disparate systems, has also been fraught with information integrity issues.

8. RECENT TRENDS

Several recent trends in information technology and the business environment dramatically intensify the urgency of finding radically different approaches and solutions to information integrity. These include:

- Information explosion;
- Application integration;
- Zero latency enterprise.

8.1 Information explosion

Business, government and society are rapidly becoming information-driven, with more and more of their functions based on information about customers, suppliers, employees, communities, citizens, products, capital and everything else. At the same time, we no longer can physically observe, measure or verify the name, address or other information about a customer; the financial capability of a supplier; or the amount of money in the bank. Instead, we manage our assets and deploy our resources based on the information we receive from our systems, databases or third-party resources. The Internet has dramatically multiplied the number of such third-party resources, as well as the amount of information available from each source. Since all this information is subject to the same types of errors and information integrity risks that were discussed earlier, we are forced to deal with massive amounts of information which, at best, is suspect.

8.2 Application integration

As organizations rapidly ramp up their presence on the World Wide Web and exponentially grow their e-commerce business, they quickly realize that their front-office systems must be integrated with both their back-office and distributed departmental systems. This forces them to undertake major application integration projects to provide straight-through processing for most of their mission-critical applications. These projects will reveal the serious information integrity deficiencies in their legacy applications and databases.

8.3 Zero latency enterprise

The net result of application integration will be to completely eliminate the batch cycles that have been characteristic of the information systems in every organization. Instead, customer's order or the supplier's invoice will be automatically processed the moment it is presented to the system, without any human intervention or verification, thus creating zero latency within the enterprise. Zero latency enterprises will have zero tolerance for information integrity errors.

9. CONCLUSION

Information integrity is a pervasive universal issue, which impacts business, government and society in profound ways in this electronic

information age. Today, the knowledge and understanding about information integrity is rudimentary, fragmented and insufficient. Because of the lack of industry standards and best practices, we pay an inordinately high price for the current level of information integrity in our organizations. The recent trends in information technology and business dramatically heighten the seriousness and urgency of finding radical, different approaches and solutions for achieving information integrity.

Information integrity has the potential for becoming a new discipline, a new science, even a new industry, very much like the environmental science and industry, which emerged as a result of society's concerns about the quality of air, water and the earth.

Many of the answers to information integrity issues already exist in various other disciplines such as robotics, aeronautics and statistics. Similarly, much can be learned from the practices of other, more mature industries such as communication, transportation, utilities and even food.

The emergence of the new information integrity science, technology and industry will require the crafting and communication of clear, compelling and consistent messages about the pervasive and critical nature of information integrity, and the formation of a coalition of academia, thought leaders, professionals, practitioners and organizations interested in information integrity.

It will entail a broad spectrum of research, education, technology, standards, products and services. It is an opportunity for the champions of quality to make a difference.

PART FIVE. PANEL SESSION

The way forward

Leon Strous
De Nederlandsche Bank
Amsterdam, The Netherlands
strous@iaehv.nl

Abstract: The closing session of the working conference was a panel discussion. Panel members were in (alphabetical order): Eric Gheur (Galaxia, Belgium), William List (Wm. List & Co, UK), Frank Piessens (Katholieke Universiteit Leuven, Belgium), Bhavani Thuraisingham (MITRE Corporation / National Science Foundation, USA). The panel addressed issues raised during the conference and worthwhile pursuing in the next event.

Key words: data quality, standards, software engineering, safety critical systems

1. INTRODUCTION

In preparing the panel session, some topics were identified where members from both academia and industry definitely could make a worthwhile contribution. The theme that we had in mind was "Integrity and Internal Control in Information Systems: old and new challenges". The outcome of this panel session is to serve as input for the next IICIS conference.

To address the current problems in the area of integrity and internal control a number of questions were prepared. For example:

- Has the problem become more complex because the systems for which we want to develop integrity and internal control methods have become more complex?

- Have the systems we are interested in actually become more complex, e.g., current banking and e-business environments?
- Do we still have the right methods and tools to do the job?
- What are the specific problems (regarding these new types of systems)?
- Do we have a sufficient understanding of the systems we are always talking about?
- Do existing standards help in dealing with the problems?
- What are emerging techniques that could help solving the problem? For example, do data mining and knowledge discovery techniques provide security personnel with appropriate tools?
- What are the systems people from industry are currently struggling with regarding the issue of integrity and internal control? What (new) achievements does academia have to offer?

2. DISCUSSION

Thanks to the lively and interactive presentations during the two days of the conference, sufficient discussion topics had already been raised during the days. The panel session therefore smoothly continued the discussions started earlier.

One of the major issues of concern was the question how to sell integrity and security to management. Here, the participants and panel members pointed out an obvious relationship with internal control and the recent standards in this area, like COBIT (as explained also during the tutorial). It was considered helpful if the auditors in their management letters and statements could include integrity and security issues. But also security professionals should try to use at least some of the internal control language which management often understands better than security terminology because the link to business processes is closer. Another option to help "selling" the good cause is to coordinate with areas like data quality which might be more appealing to management.

A problem encountered by vendors of security software that monitors the activity within the computer and its interaction with the outside world, is the fact that solutions embedded in software are not always accepted in a number of countries or in a number of organizations. The reasons for this can be found in social / cultural habits and/or in legal restrictions. Many employees find it threatening to be monitored (either literally or figuratively), it gives them a "big brother is watching you" feeling. If not for legal reasons, this is again a matter of how to sell your cause. Monitoring

can also be beneficial for employees in cases where their innocence can be proved if fraud is suspected. The question is whether monitoring can contribute to the integrity of data and systems or whether this technique only / mainly serves other purposes.

The conclusion from the discussion about the first two topics is that further work needs to be done in investigating how integrity (and other security) solutions can be presented as an integral part of a system, business process or even enterprise architecture.

Moving towards the question whether standards can help to solve the above problems, the discussion focussed on a few standards currently available. These varied from baseline controls (ISO/IEC 17799) and security management guidelines (ISO 13335) to software engineering standards that include integrity aspects (ISO 15026). Besides the discussion about well known problems (e.g., it takes too long to develop standards, often compromise which means not the best standard) there was also a clear consensus about the fact that there are many standardization efforts, both through formal standards bodies and industry fora, which very often lack knowledge about each others activities, let alone cooperation to draft standards that cover more than one (small) area. Also the combination of technical and non-technical standards is an issue to look into.

Some of the questions listed in the introduction were not specifically addressed during the panel session, but were briefly discussed during other presentations. Examples of this are the questions "What are the systems people from industry are currently struggling with regarding the issue of integrity and internal control? What (new) achievements does academia have to offer?". In the presentation about modern banking systems for example, the increasing complexity of online real-time distributed banking applications was demonstrated and a new paradigm for controls in the finance and banking industry was proposed. It is a challenge for researchers and vendors to come up with useful solutions for this paradigm.

3. CONCLUSION

The panel chair concluded from the discussion during the panel session, the discussions during the presentations and the discussions outside the conference room that the work of working group 11.5 needed continuation. It was a general feeling that the chosen formula of different types of

contributions was a very rewarding one for all participants. The conference should take place annually rather than bi-annually.

For the next event, submissions should be invited that address the questions listed in the introduction of this section. To ensure that progress is made in the area of integrity and internal control, it is essential to continue the dialogue between:

- Disciplines: data quality, software engineering, safety critical systems, standards, etc.;
- Professions: researchers, practitioners, vendors, standards developers, customers.

INDEX OF CONTRIBUTORS

INDEX OF KEYWORDS