A PATH CONTEXT MODEL FOR COMPUTER SECURITY PHENOMENA IN POTENTIALLY NON-SECURE ENVIRONMENTS

by

WILLEM HENDRIK BOSHOFF

DISSERTATION

Presented in fulfilment of

the requirements of the degree

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

in

FACULTY OF NATURAL SCIENCES

at the

RAND AFRIKAANS UNIVERSITY

PROMOTER : PROF. S.H. VON SOLMS.

I N D E X

CHAPTER 1

INTRODUCTION

Contents

OVERVIEW OF THE PATH CONTEXT MODEL

Synopsis

Having addressed the background, problem definition, scope and approach to the research project which is documented in this dissertation, an overview of the project and the results are presented in this chapter.

This chapter documents the origins and reasons for this research by examining potentially non-secure computer environments and the underlying problems. Thereafter the path context model and its impact are reviewed. In addition various examples are presented to illustrate the underlying principles.

1. BACKGROUND.

Traditional approaches to computer security have evolved along the lines of the so-called secure systems theory [1]. During that period software trends indicated that this type of approach could effectively describe the security issues in any computer environment, denoted E. The concept of an environment has been introduced to describe the occurrence of a set of heterogeneous computer systems, together with software, interconnected in such a way that it can logically be considered as a single system.

Software capabilities have evolved significantly over the last two decades to the extent that on-line real time and data base systems have become the norm. These developments have given rise to architectures and followed strategies which create circumstances which are classified as non-secure according to classical computer security theory. Numerous vendors have actively adopted such directions to the extent that the vast majority of commercial systems which make use of software such as teleprocessing monitors and database management systems are potentially non-secure.

Most publications in dealing with computer security [3], [5], [6] and [7] have extended classical theory in an attempt to accommodate security in the environments described above. Again the rapid advancement of new technologies such as microcomputers, local area networks (LANS) and their inter-connectivity surpassed the ability to

secure them. The untidiness of their software architectures and loose structuring have created problems of their own. The last few years have been characterised by the rapid deployment of the above technologies with inter-connectivity further enhanced by wide area networks (WANS) which often piggy-back on third party value added networks (VANS).

The nature of these types of co-operative cum distributed processing environments which utilise software architectures which in turn are non-secure, have created a need to re-confirm a basic understanding of the theory which underlies computer security. The mere fact that the majority of commercially available computer environments are potentially non-secure, yet with an increased need for security from a business perspective, supports this.

2. PROBLEM DESCRIPTION.

To solve the computer security crisis a myriad of techniques have been developed, many providing an inappropriate degree of comfort to those responsible for computer security. As security packages evolved, their implementation became synonymous with security albeit that they contained inherent restrictions and were implemented on an ad hoc basis.

After details of a number of computer frauds were made public, the demand for better computer security became a major issue. Unfortunately the issue became clouded as some emotionalism and sensation was promoted by numerous persons. The best evidence of this is the number of articles in the popular press which deal with computer fraud.

A study [2] made in fulfilment of a masters degree examined the interrelationships in internal control between manual activities, e.g. division of duties and computer related ones such as computer security. The results led to the development of a model known as the Access Model, details of which have been published [11]. The above study concluded that more research of computer security fundamentals was required in view of the lack of published material in that area.

A summary of the major areas which were identified as constituting the main reason for this project is presented as follows :

(a) The inability of classical computer security to describe security in modern heterogeneous computer environments effectively and efficiently;

(b) The deployment of architectures and structures in software which are potentially non-secure;

(c) Lack of published material which deals with the interrelationship between an organisation's control requirements and computer security; and

(d) The need to introduce a high degree of formalism in any area being investigated. Lack of formalism creates the risk of propagating fragmented adhocracy.

3. SCOPE OF THE RESEARCH

The scientific challenge not always lies in describing what ought to be. Often the greater need lies in providing solutions, sometimes temporary, to the imperfect but real environment from which the demands arise.

Within this context the scope of this research project was threefold :

- To address computer security against the background of potentially non-secure environments.

- To introduce some degree of formalism based on classical computer science thus providing a theoretical basis.

- To explore the potential of automated computer security support for automated exposure evaluation, automated profile generation and automated package evaluation.

It is by no means the intention to suggest that the results of this project provides the only solution, nor that they cannot be enhanced or formalised even further. Instead it is intended to provide a sound basis for addressing some of the computer security problem areas and to introduce a conceptual foundation for either critique or enhancement. Without some bold venture into this difficult yet topical issue of computer security in potentially non-secure environments nothing is gained.

4. APPROACH TO THIS RESEARCH.

Based on the problem statement and scope of this research, it is evident that numerous approaches were possible :

(a) An empirical study of the occurrence of fraud and computer security risks could have been done. The sensitivity of these issues and some of the emotionalism attached created the risk of an uninformed bias towards this research being introduced. We therefore adopted a principle that anybody even vaguely

familiar with computer systems can confirm the issues which have been raised. To have spent time and effort in confirming the obvious provided no added value and hence this avenue was not further pursued.

(b) A very wide literature survey could have been carried out to provide a detailed analysis of the problem, the use of various techniques as solutions or a combination thereof. This research resulted from twelve years experience with various aspects of internal control including eights years specialisation in computer security on an international level as well as a master's dissertation [2] in this area. The danger of the literature route approach lay in the risk of propagating the adhocracy which is evident in the current computer security arena. Instead we adopted an approach by examining organisational requirements for computer security, the problems which are introduced by modern technology and the theory which underlies the topic by modelling each area and their interrelationship. It is acknowledged that classified material may be present, but their lack of availability have placed them outside the scope of this research project.

(c) A more hollistic approach whereby the basic issues of computer security could be formally examined provided another avenue.

Rather than elaborate analysis of detail this avenue provided the opportunity of formally examining the broader issues and their interrelationships. Whilst the risk of this approach lay in it being ambitious and requiring fundamental work, the major contribution was towards the scientific formulation and thinking thus giving rise to further scope for research.

Obviously any avenue adopted would have been subject to normal research fundamentals.

After deliberation the hollistic approach, item (c) above, was adopted in view of the potential contribution that could be made. Essentially this represents a fundamentalist approach best described as "Back to Basics". The danger of this type of research is that it could result in an isoteric discussion which provided little or no contribution. To address this risk it was decided to document the research, and hence this dissertation, as a number of independent chapters on which articles, to be published internationally, could be based. Each article focuses on a particular aspect as follows :

A. Chapter 2 entitled "Overview of the Path Context Model" contains a comprehensive overview of the research project. By way of introduction it sketches issues such as dynamic initiation of network sessions, multi-domain computer environments, use

of multiple executions in single address spaces and multi-domain system functionality which create potentially non-secure environments. These same basic security principles, to which any potential model should adhere, are discussed. Following this, the model which forms the basis of this research project, the Path Context Model (PCM), is introduced. This model is based on Random Context Grammars [8]. The proposed PCM formalises the concepts of accessor transformation and baggaging as well as the following components which provide the structure of a security system :

(a) A Baggage Collection Vehicle which creates the baggage which consists of the information which needs to be collected and transported across system boundaries in order to achieve the objectives of computer security.

(b) A Security Profile which contains the security rules or restrictions that need to be enforced.

(c) The Validator which matches the baggage and security profile and provides the True or False condition for allowing access to secured objects.

A number of examples are introduced to illustrate the application of PCM in providing security in potentially non-secure environments.

B. Having (informally) introduced the PCM in chapter 2, chapter 3 discusses the model in much more detail, concentrating on the formal language aspects, namely Random Context Grammars, on which the PCM is based. Path Context Grammars (PC grammars), and extended PC grammars, derived from Random Context Grammars, are introduced.

C. Having established a model which is capable of addressing aspects of potentially non-secure environments in Chapters 2 and 3, Chapter 4 "Application of the Path Context Model" deals with the application of this model in complex computer systems.

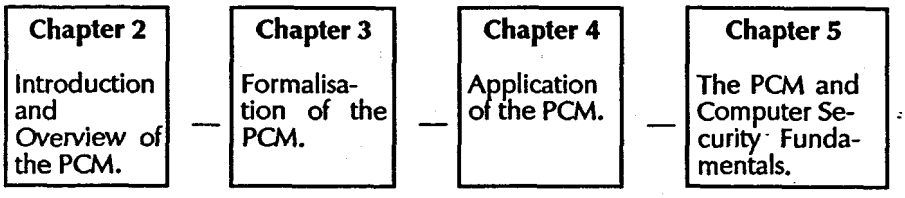
Systems which exist in more than one domain, make use of multiple system software components to access objects and allow multiple executions in a single address space are covered under this heading. In addition a process for evaluating security in terms of the established principles is presented which forms the basis of automated computer security support. This concept which encompasses automatic profile generation, automatic risk evaluation and automatic package evaluation is aimed at

introducing a framework whereby the complexity of computer security in the environments which have been described above can be supported by automated tools. The way in which the proposed PCM can be used to address these aspects, forms the basis of this chapter.

D. As part of the research project on which this dissertation is based, it was found necessary to apply a degree of formalism to computer security fundamentals. The major reason for this development was the lack of such formalism in published material and it being a prerequisite for understanding automated computer security support. Chapter 5 with the heading "Computer Security Fundamentals" deals with these aspects and describes a concept termed the Validity Hierarchy which provides an interface between business principles, computer security and the Path Context Model using regular set theory and Random Context Grammar theory. In terms of structure this chapter contains an informal treatment of the topic as well as an attempt to formalise the concepts. The contribution of the PCM in addressing these fundamental issues, is highlighted

E. Evaluation of the research.

We can therefore summarise the thesis as follows, showing how the PCM forms the continuous thread right through the whole thesis.



At the time of finalising this dissertation the following articles have already been submitted and/or accepted for publication :

ARTICLE	SUBMITTED TO	STATUS
A Path Context Model for Addressing Computer Security in Potentially Non-Secure Environments.	Elsevier for publication in Computers and Security.	Accepted for publication.
Modelling Computer Security in Potentially Non-Secure Systems Using Formal Language Theory.	Elsevier for publication in Computers and Security.	Pending.
Application of a Path Context Model for Addressing Computer Security in Complex Environments.	IFIP/Sec '90 International Security Conference.	Pending.
Application of a Path Context Approach to Computer Security Fundamentals.	Information Age	Accepted for Publication

Inherent in the approach adopted is a degree of redundancy and duplication in the chapters to make each one separately publishable.

From a research methodology perspective the following phases were identified and followed :

(a) Problem definition.

(b) Literature Survey.

(c) Model Synthesis and establishment of the underlying theory.

(d) Experimentation.

(e) Presentation.

Although the various chapters contain further details in relation to each phase some comments on the Literature Survey were deemed necessary.

5. LITERATURE SURVEY.

Whilst there is a myriad of literature available which deals with computer security, few applied to the objective of this research. As a result the more well-known classical approaches and papers which could assist with the synthesis of a theoretical model and explain principles were used. As the objective was not to research or review specific techniques or methodologies few were found to be relevant thus resulting in a relatively small bibliography.

It is acknowledged that there is a risk of a model, such as the Path Context Model which is set out in this paper, may exist as classified, confidential of unpublished material. To date, however, none were discovered which required acknowledgement in this dissertation. On this assumption this is therefore considered original work although a relatively small bibliography has been presented.

6. RESEARCH FINDINGS.

The result of this project is a theoretical model which has been found capable of addressing computer security in a significant number of

situations. In fact, the more isoteric and complex the environment that needs to be modelled, the more effective the model has been found to deal with it.

As research this project has contributed to :

(a) Formalisation of computer security fundamentals and insight into problems areas.

(b) Potential for automated computer security support in the form of profile generation, exposure evaluation and package evaluation.

(c) Application of Random Context Grammars as a basis for handling a variety of computer security restrictions, pre-requisites and/or conditions.

(d) Documenting the computer security concerns and principles.

(e) Applying classic computer science to real world problems and illustrating the power of using these approaches.

7. SUMMARY.

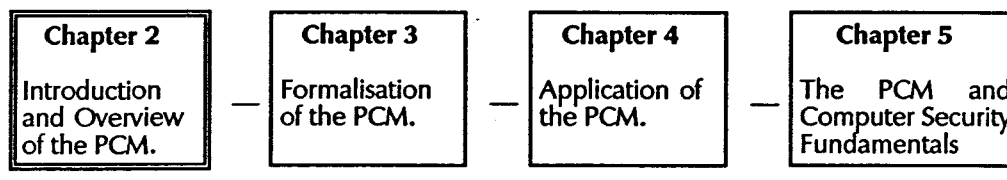
This dissertation documents the results of a research project which deals with computer security in potentially non-secure environments. It provides a fundamental approach to the requirements for better security in computer environments which do not comply with the principles of classic security approaches. It is intended to introduce a rejuvenation of computer security research; the classic research having been done some time ago with relatively little publication of subsequent work along with the rapid advances of computer technology.

CHAPTER 2

OVERVIEW OF THE PATH CONTEXT MODEL

Contents

Synoptic Perspective



OVERVIEW OF THE PATH CONTEXT MODEL

Synopsis

Having addressed the background, problem definition, scope and approach to the research project which is documented in this dissertation, an overview of the project and the results are presented in this chapter.

This chapter documents the origins and reasons for this research by examining potentially non-secure computer environments and the underlying problems. Thereafter the path context model and its impact are reviewed. In addition various examples are presented to illustrate the underlying principles.

1. INTRODUCTION.

Every security system is designed to protect and control access to resources of some real world computer system. At any point in time it is intended to project security policy enforcement by reflecting access capabilities of subjects requesting access to objects in the system. The key question in any environment that needs answering is who, implying granularity of individuals, has access to what, implying granularity of discrete system objects. The term granularity has been used to introduce the requirement for identifying the relationships between individuals and objects in sufficient detail that security can be effective. In its simplest form security constitutes the control of simple and discrete subjects (e.g. Users) who requires access to objects (e.g. Programs and files). The effectiveness or adequacy of the security system is therefore its ability to differentiate at an appropriate level of granularity between individuals, or other subjects, and any system objects as well as the integrity of the security system itself and the environment within which it functions. The classical research in this area is represented by the Bell and LaPadula model which originated from work done at MITRE [1]. The importance of this model lies in the degree of formalism which was achieved by applying classical computer science approaches.

As computers evolved and security became more complex, the simple subject-object mapping no longer satisfies the demands of the environment. As a result the Bell and LaPadula model became viewed as a theoretical system which is unlikely to be implemented in commercial environments. Enhanced models proposed [5], [3] are capable of describing security in more sophisticated environments. The articles by Landwehr [6] and

Summers [7] contain reviews of the various models should further background reading be required. All the published models have a great deal of commonality in the sense that they are based on discrete and fairly simple relationships between subjects and objects. These approaches can take care of situations where primitive chaining takes place during which, say, a user accesses a file for reading/writing by utilisation of one or more processes and programs. In addition the integrity of the security system is enforced by a security kernel with secure and problem states which prevent interference in a user's or system software's compartment or address space by other users and processes. This implies that implementation of such a system is only successful if a mechanism which restricts a user's processes and activity to a single isolated address space, meaning single executions in address spaces, is a place. These are well-known operating and secure systems concepts as most integrity violations have resulted from the ability to compromise these mechanisms.

Over the years most publications which have dealt with security have focused on the application of the classical approaches which have been mentioned above. Meanwhile developers of system and application software have adopted different approaches to software architectures. Some of these directions make the traditional approaches to security difficult if not impossible to apply. Yet, if anything the realities of relying on vendor developed software as well as the increased demands for highly reliable protection of computer resources are a fact of life.

Traditional computer systems were largely single domain, restricted to one host machine, as connectivity and the full use thereof has been a fairly recent phenomena. Although computer research has propagated many of these developments for over a decade it is only in the last few years that we have seen integration, rapid deployment and extensive use of :

- Wide area networks;
- Local area networks;
- Value added networks; and
- Distributed processing where system data and/or application functionality could be distributed.

The implementation of these technologies have resulted in new phenomena in computer environments.

- Dynamic rather than user initiation of network sessions.
- Dynamic initiation and rerouting of network sessions in other domains of a multi-domain environment.
- Common usage of multiple executions in single address spaces (MESAS) of operating systems where multiple users share and execute processes, programs, etc. in one address

space. This has even been extended to processes being exchanged between software components which are resident in different domains.

- Multiple system software components are utilised in carrying out simple on-line requests for processing or retrieving information. One or more of these components may make use of MESAS concepts.

- Transparent multidomain access to data.

- Transparent, multidomain application functionality.

- Loss of single user or individual identity during the processing path, particularly where MESAS concepts are utilised.

- Dynamic sharing of routines and processes among multiple software components.

- Parallel execution of processes in any of the above situations.

- Presence of modules, programs and processes often nearing 10^6 in order of magnitude.

The scope of this chapter is to provide an overview of an alternative approach for applying classical computer science approaches to the situations listed above. It is therefore not our intention to analyse each issue and its impact on security in this chapter. A more detailed discussion can be found in chapter 5. On the other hand a conceptual understanding of the security issues in computer environments described above are necessary to appreciate some of the problem areas and an example is presented in figure 1 for illustrative purposes.

Figure 1

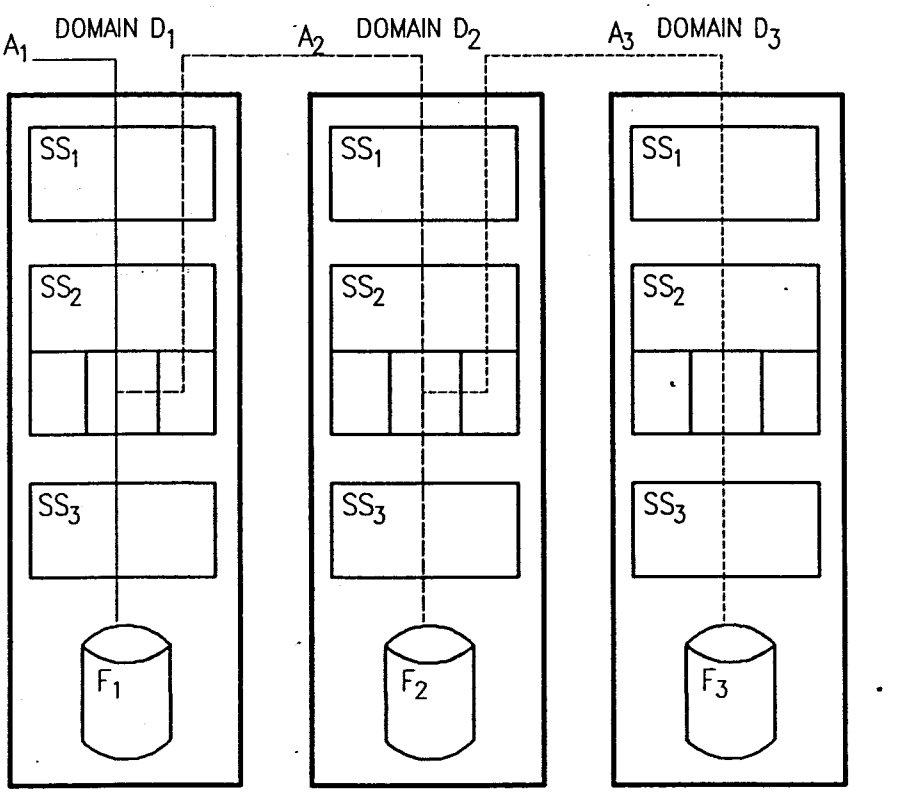


Fig. 1 sketches a fairly simple set of accesses which are initiated at one end by a user A_1 who retrieves information from files F_1, F_2, F_3 in three different domains without necessarily being aware or able to control any of it. In a typical situation A_1 would logon to system software components SS_2 in domain 1 (D_1) whereafter the latter assumes control over all subsequent processing. In fact, after A_1 has been identified to SS_2 in domain 1 it is quite common to find that once the processes which execute in the same address space as SS_2 assume control all subsequent activities do not recognise A_1 nor even take cognisance of the person's existence. In domain 3 (D_3) SS_3 gains access to the data with the A_1 granularity not even accessible at all. In domain 3 there is often no system software component which even knows where the requests were initiated. It requires little imagination to identify some of the concerns. Even if one were to suggest cryptography and sophisticated access control mechanism in an attempt to control A_1 to a larger extent, it is clear that they have very little impact; perhaps only as a deterrent to unauthorised access. In addition the vast number of models or processes, many of them called by others disqualifies a view of discrete processes and simple subject-object relationships in such an environment.

The demands for a higher degree of security, the prevalence of such systems today and the apparent lack of published material has given rise to the research in which this chapter is based. The objective of this chapter is to explore the development of a model which can be applied to computer security in complex multi-domain computer environments.

2. PRINCIPLES OF COMPUTER SECURITY

A major danger when researching computer security is becoming engrossed with the myriad of techniques and emotional issues which are present today. Much of the classical work and the principles have been ignored by more technique orientated researchers at the expense of making security a battle of wits. Invariable one finds that losing sight of the basic principles results in the issues under consideration becoming blurred or distorted. It is submitted that there are only four basic principles that need to be borne in mind :

- Firstly security is based on the construct of ultimately restricting the capabilities of an individual who has access to a computer system. It implies that the responsibility for implementing, accessing or activating any process or task ultimately vests with that person. Similarly chaining, proxy login and even accesses across domains do not affect this principle.

- Secondly, having established the authorised user base and restrictions there is a risk that a number of unauthorised individuals will either attempt to access the system or, alternatively, authorised users may attempt to act outside the restrictions imposed on them. A security system therefore has two functions : to apply the restrictions and try to prevent and/or detect unauthorised activity.

- Security can be implemented in a preventative and/or detective mode. Preventative mode is aimed at gathering enough data about a user to decide whether the person is authorised, or not, to be granted access to a protected object or resource. Detective mode on the other hand, is directed at collecting enough data about a user to decide, at a point in time after the access to the object or resource was granted, whether the person was appropriately authorised to do so. Both modes have relative advantages and disadvantages and are recognised as the main approaches to computer security.

- Ultimately security must deal with reality. If reality means dealing with computer environments which differ significantly from the criteria which are specified by traditional models then be it so. The scientific challenge lies in developing alternatives whereby the security requirements which are demanded by our environments can be met.

3. APPROACH TO THE RESEARCH.

The research project on which this chapter is based originated from an analysis or real world computer system architectures, increasing demands for security and the difficulty of using traditional security models to describe and implement security. In fact reality has necessitated the need for security

mechanisms in environments which, in traditional terms, are potentially non-secure. At the outset it is stressed that we are not suggesting that other security models are incorrect or invalid. In fact the Bell LaPadula model is very sound. The problem lies in its application where the environment under consideration is for example one such as described in fig. 1.

The criteria which were imposed on the project is that any alternative approach should extend traditional foundations and be able to accommodate them as special cases. In addition the principles of computer security as described in the previous section should not be compromised in any way. Surprisingly enough it was possible to construct a relatively simple model which, with some experimentation, was found to accommodate a variety of situations. We refer to this model as a path context model (PCM) of security, or simply the PCM.

4. PATH CONTEXT MODEL OF COMPUTER SECURITY.

The Path Context Model (PCM) is based on two basic concepts :

- Firstly, the concept of an access path which is formed by the various components that need to be activated or utilised in order for a typical user's request to be executed. It is submitted that the initial originator of any service request is always an individual termed a primary accessor say A_1. Assume that Software Components $S_1, S_2 \ldots S_n$ in a single domain D_1 are required to access some object, say a file

or a block of information on a file, O_1. Then $A_1 S_1 S_2 .... S_n O_1$ is defined as an access path to O_1. The notation implies that A_1 initiates S_1, S_1 in turn activates S_2 and so on.

Notes

(i) n is finite for any environment.

(ii) There may be multiple access paths to O_1.

(iii) The total number of access paths in any environment, E, is finite.

(iv) Any given S_i is best described as a software component major node which utilises or activates hardware or other resources. Typically S_i consist of a set of tasks, processes, routines or programs. It is submitted that having the ability to control access via the software is in line with software or firmware based directions in computer engineering.

(v) The access path concept applies equally to cross domain environment. In this case a corresponding S_i simply exists in another domain.

- Secondly there are obvious advantages in having a security model which accommodates preventative and detective modes even though the latter may only serve as a back-up mechanism. PCM is primarily geared to prevention of unauthorised access by attempting to determine invalid access requests prior to allowing access to a protected object. This is particularly important where accessors are transformed and domains crossed such as in figure 1. To achieve this a principle of baggaging is introduced. Baggage is defined as the minimum amount of information that has to be collected and must accompany the access request or its route in order that responsibility and access authority checking can be performed even though various transformations or domain crossing may occur. Baggaging has been found to be a useful concept when taking cognisance of not only accessor transformations and domain crossing, but also integrity parameters which are associated with S_i, $i = 1, n$. An example would be S_i executing in supervisor or privilege versus problem state. By introducing baggaging this research project has clarified some of the traditionally complex areas of computer security while leading to some interesting discoveries, particularly concerning the role of many commercial techniques and security packages. The detection option of PCM is quite simply a process of baggage retention and post access checking.

4.1. Notation

The following notation is presented to formulate PCM in an environment E :

$E = \{D \times S \times I \times O \times A\}$

D ={Valid domains}$\times K$

S ={Valid software components}$\times K$

I ={Integrity states}$\times K$

O ={Valid objects}$\times K$

A ={Valid Accessors}$\times K$, with A_1 always being an individual person or primary Accessor

K ={Valid access classes}.

Fig. 2 illustrates the application of this notation in a complex environment.

Figure 2.

Element	Example excluding Cartesian Products with K
D	LAN, WAN, VAN
S	Network software, teleprocessing monitor, DBMS.
I	Problem state, supervisor state, MESAS, Encrypted transmission
O	Program, file, block of data, data element
A	User-transformed accessor
K	Read, execute, write, delete, passthru, pre-access checking, post access checking

The concepts of an access path $A_1 S_1 S_2 \ldots S_n O_1$ and baggaging have already been introduced. To formalise the concept of baggaging a baggage vector $BV = (A, D, S, I, O)$ is defined where BV reflects the values of A, D, S and O at different points in the access path. Where any value is

unknown or cannot be determined it is assigned a nil value denoted ∅. The baggage B is defined as $\{BV_1, BV_2, \ldots BV_j\}$ where $BV_i \neq BV_{i+1}$. Reasons for not requiring a more elaborate definition will become evident later on.

4.2. Structure

After a number of avenues had been explored, the simplest and cleanest approach for developing PCM was the utilisation of formal grammar concepts. Obviously the other motivation is the benefits provided by having automata theory available for implementing such a model, particularly as PCM could form the basis for a variety of systems, including expert systems.

Like any security model PCM consists of three components. They have been identified as the following distinct and mutually exclusive elements :

- A set of restrictions which specify the security procedures to be applied termed the security profile.

- A mechanism which collects information against which the security profile can be applied termed the baggage collection vehicle.

- A mechanism which performs the actual checking termed the validator.

Using the notation in section 4.1. a formal grammar was developed to accommodate this structure of the security model. Although it essentially involves one grammar, separate definitions for the three components are

presented. As the grammar is quite simple and only requires a basic knowledge of formal grammar, we have not elaborated on the structure. Instead a number of examples have been prepared to illustrate the application of PCM.

Specification of Baggage Collection Vehicle

$A \rightarrow A_1 \mid A_2 \mid A_3 \mid .... \mid A_k \mid C$

$D \rightarrow D_1 \mid D_2 \mid D_3 \mid .... \mid D_m \mid \emptyset$

$S \rightarrow S_1 \mid S_2 \mid S_3 \mid .... \mid S_n \mid \emptyset$

$I \rightarrow I_1 \mid I_2 \mid I_3 \mid .... \mid I_p \mid \emptyset$

$O \rightarrow O_1 \mid O_2 \mid O_3 \mid .... \mid O_r \mid \emptyset$

$C \rightarrow ADSIC$

$C \rightarrow O$

The extent of the baggage collection process is the most significant element in PCM. Insofar as a computer environment's control blocks, logs or status vectors are unable to provide information about the activities of primary accessors, there is a security deficiency which constitutes a risk. For example a baggage vector $A_1 D_1 SS_1 SS_2 A_2 SS_3 O_1$ would allow tracing of A_1's activities to the point of accessing O_1. Should BV assume a format of $SS_2 SS_3 O_1$ or $SS_3 O_1$, no information about A_1 is known. No matter what the capabilities of the security profile and validator are, it would be impossible to apply appropriate security principles under these

circumstances. Some of these situations are frequently solved with some form of proxy principles [4]. Interestingly enough it was found that where BV was deficient e.g. SS_3O_1, even proxy login constituted risk as it is potentially possible for anybody to access O_1 with no definite assignment of responsibilities. This is, for example, aggravated by wild card proxies. It is also shown later on that O can be extended to any combination of A, D, S, I or O.

Specification of Security Profile

$A \rightarrow | A_1 | A_2 | A_3 | .... | A_k | \emptyset$

$D \rightarrow | D_1 | D_2 | D_3 | .... | D_m | \emptyset$

$S \rightarrow | S_1 | S_2 | S_3 | .... | S_n | \emptyset$

$I \rightarrow | I_1 | I_2 | I_3 | .... | I_p | \emptyset$

$O \rightarrow | O_1 | O_2 | O_3 | .... | O_r | \emptyset$

$C \rightarrow ADSIC$

$C \rightarrow O(B;F)$

$B \rightarrow C$

$B \rightarrow < C >$

$B \rightarrow \ll C \gg$

$F \rightarrow C$

$F \rightarrow < C >$

$F \rightarrow \ll C \gg$.

This section deserves more of an explanation as it reflects the required security restrictions in a given environment. B and F have been introduced to reflect compulsory and prohibitive context respectively. B therefore determines conditions which are compulsory to gain access to O while F those which disqualify access to O. This can, for example, be used to implement specific routes or integrity constraints. In addition the conditions $< C >$ and $<< C >>$ have been introduced to reflect random and fixed adjacency to accommodate compulsory productions. For example, access can be restricted to a single domain and specific software or paths can be specified in order of access. Thus in enhancing overall security and flexibility. These concepts are relatively simple applications of random context grammars [8].

The functionality of the security profile is to cater for possible restrictions to enforce division of duties in an environment E. It is submitted that an issue such as confidentiality is a natural extension of division of duties. With a random context grammar some very interesting restrictions are possible. Although some case studies are presented below, we point out that conditions such as $A_1 O_1 (\ll D_1 I_3 D_2 >> ; D_3)$ can be used to prohibit access from LANS, say D_3, and enforce access via encrypted line, say I_3, between domains D_1 and D_2. As the power of random context grammars is available to PCM a variety of security restrictions can be represented.

Validator.

The validator applies the risk profile conditions to the baggage reflected in the Baggage Collection System and determines the validity of the access request to any secured object. The strength of this approach lies in the focus on the previous two components as the basis for security while the validator simply performs the "if...then" logic.

Rather than elaborate on the pure theoretical aspects of PCM, some short case studies have been formulated to illustrate the underlying principles further.

5. APPLICATION OF PCM.

Using Fig. 1 as the basis for illustrating various principles and applications of PCM are presented in this section.

Case 1.

The first principle of computer security is based on the construct of restricting the capabilities of an individual. Assuming F_2 is a secured object and that the access is initiated by A_1 as illustrated. In many current security systems the baggage would only be available for D_2 and may be something like $A_2D_2SS_1SS_2SS_3F_2$ or even simply $D_2SS_2F_2$ where A_2 is best described as a transformed accessor which was generated in to sign on into D_2. Under these circumstances of inadequate baggaging any security checking is severely restricted and there is no way in which any responsibility

can be assigned to A_1. In fact, no system could even detect security problems in D_2. This problem manifests itself fairly frequently in multi domain environments where network software, teleprocessing monitors and data base management systems are present. Taking this one step further one finds that even though user authentication of A_1 in D_1 may have been carried out by a very sophisticated mechanism, it becomes meaningless in D_2 where the actual secured object being accessed is located. Even having an encrypted line between D_1 and D_2 doesn't solve the issue as we still cannot determine anything about A_1.

One may argue that reliance needs to be placed on the integrity of the various environments. In the above example there is total ignorance about the integrity of D_1 or D_2 ; not particularly useful if D_1 and D_2 are potentially non-secure. Readers who are familiar with commercial environments may recognise some of these issues as they occur more frequently than one would perhaps like. Even though one would like to enforce an access restriction $A_1 F_2$ across two domains, the absence of adequate baggage makes it impossible to enforce.

By introducing this extent of formalism into the various components, in this case the baggage collection vehicle, PCM facilitates :

- evaluating whether the baggaging in an environment E is such that the activities of accessors and their responsibilities can be controlled in the first place. No security profile or validator can compensate for inadequate baggaging;

- automatic evaluation of security risk by analysing baggaging deficiencies in E;

- given a specified security profile, PCM can be used to specify the baggage requirements to enforce it; and

- evaluation of the true capabilities of a security system by evaluation of the baggaging it uses or has access to.

Case 2

Assuming that we have more complete baggage through a comprehensive baggage collection vehicle. The baggage for access to F_2 can, for example, be represented by $A_1 D_1 SS_1 I_1 SS_2 I_2 SS_3 F_1 SS_3 SS_2 I_1 SS_1 I_3 A_2 D_2$ $SS_1 I_1 SS_2 I_2 SS_3 F_2$ where I_1 could represent problem state execution, I_2 could be MESAS and I_3 encryption between D_1 and D_2. Providing this degree of baggaging is available, Fig 3 contains some of the security profiles which can be constructed in PCM :

Fig. 3

Security Profile	Meaning
$A_1 F_2$	A_1 has unrestricted access to F_2.
$A_2 F_{2_r}$	A_1 has, say, read access to F_2 (F_{2_r} is actually a different object in PCM to F_2). On this basis anything in E can be classified as a secured object and protected.
$A_1 F_2(D_1; D_3)$	A_1 can access F_2 from D_1 but not from D_3.
$A_1 F_2(\ll D_1 I_3 D_2 \gg; D_3)$	A_1 can access F_2 provided it is done from D_1 via encrypted line, say I_3, to D_2. No access from D_3.
$A_1 F_2(I_1; I_2)$	A_1 can access F_2 in problem state, say I_1, but not where MESAS, say I_2, present.

No restrictions have been found which could not be accommodated by this notation. This includes random or fixed ordering of any component in n domains. Accessor transformation is simply accommodated in a security profile which states that F_2 is accessed by A_2 transformed from A_1, e.g. $A_1 \ldots . A_2 \ldots . F_2$.

The above examples provide some illustration of the random context grammar capabilities which are utilised by, in this case, the security profile component, to provide :

- the ability to secure any resource in *E* and to allow nearly any type of restriction;

- introduction of a flexi-secure environment where certain objects can be highly secured whilst others may remain unsecured;

- accommodation of any environment even where LANS, WANS are present; and

- research into the potential of automated profile generation and maintenance.

Case 3.

The capabilities of the Validator are a function of the baggaging collection vehicle and the security profile. By implication the reverse is true in the sense that security is limited by the baggage collection process and the security profile that can be established. It is submitted that given any environment *E* (see section 4.1.) which has a security system comprising a baggage collection vehicle, security profile and validator it is possible to derive the actual security requirements as well as the limitations of the security system in terms of either baggage collection, security profile capabilities or both. The same principle applies where standard logging facilities serves as the baggage collection vehicle.

Proof :

Given any environment E the number of access paths to its secured objects is finite. The set of all access paths P is the determinant of the degree of restrictions R that need to be placed in E. P therefore determines the baggage $B = \{BV_1, BV_2 .. BV_j\}$ which is required to ensure that R can be enforced. As B and R serve as the specifications for a security system, we can write them as B_s and R_s.

Given any security system for E we decompose it into the actual baggaging collection and security profile to give B_a and R_a. Insofar as productions of B_s and R_s cannot be accommodated by B_a and R_a respectively the security system cannot enforce the required degree of security in E. Should $B_a < B_s$ the baggage collection is inadequate or where $R_a < R_s$ the security profiler is inadequate. Assuming the validator functions in terms of B_a and R_a we can determine where E is at risk for lack of adequate protection. PCM provides a mechanism to compute the deficiency and risk with a high degree of formalism associated with it.

Case 4.

PCM can be used to model a classic secure environment such as those described in the Bell and LaPadula model [1] as a special case. Such an environment is associated with one domain and two integrity states, say I_1 as problem and I_2 as privileged state and provided that $B = \{BV_1 .. BV_j\}$

incorporates I_1 and I_2 in its productions, e.g. $A \rightarrow A_1 S_1 I_1 O_1$ or $A \rightarrow A_1 S_1 I_2 O_1$ and its security profile, e.g. $A \rightarrow A_1 O_1 (I_1 ; I_2)$ or $A \rightarrow A_2 O_1 (I_2 ; \emptyset)$, the environment can be modelled. The relative simplicity of these environments is evident from the above examples, particularly the fact that baggage is not really a major issue due to its triviality.

Case 5 : Accommodation of Specific Security Issues.

There are always a variety of questions when dealing with a security model. Quite often they reflect current security concerns and the intention of this final case is to provide a brief summary of some issues that can make PCM a powerful model.

Issue	**Solution**
Terminal security	Extend D_i to include a terminal identification and protect as a secured object.
Passwords	Not a mechanism which provides security but a technique to authenticate $A_1 \ldots . A_n$.
Public network	As a domain, say D_i, with an integrity indicator I_j to reflect the risk issues.
Microcomputer	As a separate domain with a device and integrity indicator associated with it.
On-line transactions	As a software S_i associated with system software, say S_{i-1}.

6. SCOPE OF PCM.

While conducting a research on which this chapter is based, a number of areas were explored to research the scope of PCM. Of particular significance was the relative simplicity with which the security issues in very complex environments could be modelled and explored. Even environments where some areas required no restrictions whilst other needed to be highly secured could be accommodated. This makes it possible to evaluate the appropriateness and functionality of security software in such environments as well as determine the risk factors associated with it. In fact, by using PCM a total flexi-secure set of restrictions can be defined, monitored or evaluated. Even O can be extended to accommodate most elements of E.

Areas currently being explored includes the use of PCM as a basis for automatic profile generation and maintenance. This can be achieved by simply extending the accessor to include a security and/or job classification without any loss of generality. In the event of the existing definition of $E = \{D \times S \times I \times O \times A\}$ no longer being able to describe it, it is a trivial exercise to extend PCM by introducing another factor. Theoretically E can therefore be infinitely extended. This kind of scenario may become applicable to describe security in highly parallel environments or to unravel access requests in fifth generation systems. Another natural extension may be the introduction of fuzzy logic to include probabilistic factors.

7. CONCLUSION.

Security requirements have become major issues in organisations as more of the business activities have become computer supported. Enhanced by strategic directions such as using information technology for competitive advantage businesses now also have vast amounts of data available, usually on-line. Unfortunately commercially available system software and technological development such as micro computer, LANS, etc., have not helped to alleviate security issues. In fact, the strategic directions adopted by many organisations have actually aggravated the problem.

Based on experience with the organisational and technical aspects of computer security, this research project was conducted in an attempt to address the shortcomings of existing approaches, particularly in the light of the computer environments being created. The approach adopted was to revisit the basic principles of computer security with the objective to formulate a model which could accommodate the complexity of current and future environments.

The most significant contribution of this research project was the development and formalisation of the baggaging and access paths concepts. These concepts allowed us to overcome the restrictions of classical approaches to computer security in modern and future computer environments, many of which are potentially non-secure for a variety of reasons. Application of the concepts gave rise to the specification of a model termed a Path Context Model (PCM) to address security issues in a significant number of different computer environments. At first the approach appeared

very simplistic, but was found to accommodate an amazing number of different environments. In fact the more complex the environment the clearer the model becomes, mainly as a result of the baggaging and access paths concepts.

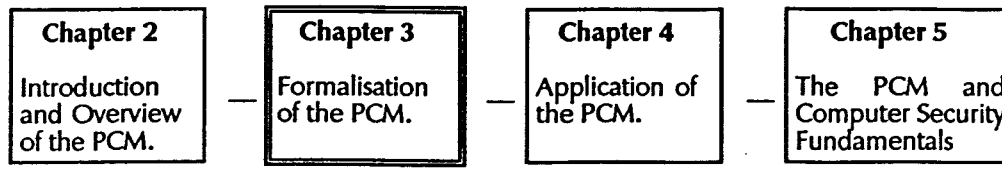
The Path Context Model (PCM), an overview of which is contained in this chapter, has enabled formal research in the areas of computer security in potentially non-secure to an extent not yet found in published literature. It is hoped that researchers will again revive computer security as an area of research as it is likely to become a major area of concern in the future unless formally addressed to accommodate new technology and system principles.

CHAPTER 3

FORMAL THEORY OF THE PATH CONTEXT MODEL

Contents

Synoptic Perspective



FORMAL THEORY OF THE PATH CONTEXT MODEL

Synopsis

At this stage the background to this project has been covered (chapter 1) as well as an overview thereof, including the origins, results and implications (chapter 2).

The following chapters have been structured to cover various topics which were presented in chapter 2 in more detail. In this chapter the formal theory of the path context model which was presented in Chapter 2 is further developed and formalised. Consequently the format of this chapter is formal grammar oriented in the sense that the application of Random Context Grammars, as the basis of the path context model, is presented.

1. INTRODUCTION.

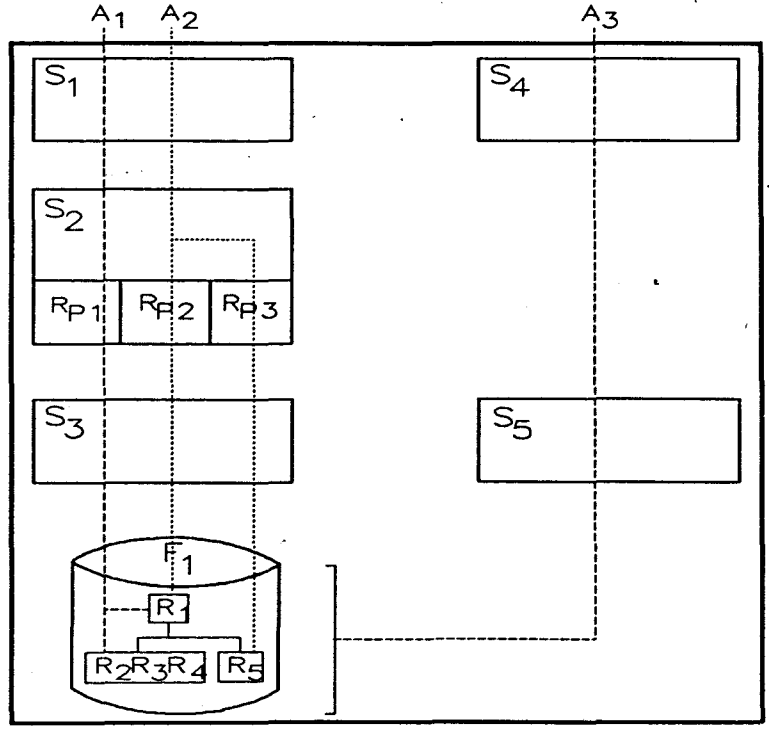
Most of the published material which deal with theoretical models of computer security concentrate on relatively simple subject-object security mechanisms which exist in single domains computer resources under the control of a single computer. Whilst very valuable work [1], [3] and [5] have been published, their effective application in distributed and co-operative processing environments is often limited. The main limitation is that modern computer environments are often based on architectures which have traditionally been declared non-secure. One example is the phenomena whereby system facilities allow multiple executions, initiated by more than one accessor, to take place concurrently in one address space. Another is the transparent or dynamic initiation of sessions between two or more domains without the ability to take cognisance of the origination or history of the service request.

Rather than attempt to rehash existing secure system theory, which is already well developed, the objective of this chapter is to present the theory and formalism of a model which is aimed at addressing computer security in potentially non-secure environments. Potentially non-secure systems are conveniently defined as typical Local area networks (LANS), Wide area networks (WANS), Value added networks (VANS) together with the computer systems and underlying system software which have been linked and do not comply with the principles of secure system theory.

The scope of this chapter is to present an alternative theoretical model which utilises classic computer science theory and addresses the problems in potentially non-secure environments as well as automated security. The model, which has already been described in chapter 1, is further formalised in this one.

Probably the best way of introducing some of the problems of applying computer security definitions to a formal grammar based model is by way of an example. Fig. 1 represents a fairly conventional computer system with three primary accessors, always individual persons, issuing requests to access various elements of a file F_1.

Figure 1



The constraints that need to be enforced are that A_1 and A_2 should only access elements of F_1, say R_1, R_3 and R_1, R_5 respectively, while A_3 is permitted to control F_1 without gaining access to individual elements. In a typical computer environment each primary accessor could initiate various system software components, denoted S_j for the j^{th} component, or application programs, denoted R_{pj} for the j^{th} application program. To reflect the complete access process, the concept of an access path which is formed by the various system components that need to be activated to service a request is introduced. For example $A_1S_1S_2R_{p1}S_3F_1R_1R_2$ and $A_3S_4S_5F_1$ are two access paths in the environment with $R_1 to R_5$ representing data elements which are contained in the file F_1. Obviously multiple access paths do exist, the issue being that in any environment E the number of access paths are finite. A short discussion of some typical security problems are presented by reference to the above example :

(a) The access paths which were used to illustrate its definition show intuitively that simple object-subject relationships are no longer an effective way of describing the security requirements in environments such as in Fig. 1. The effectiveness of object-subject relationships is further reduced by phenomena in system software design which, firstly, transforms the primary accessor, say A_1, to A_{11} and A_{12} for system software components S_1 and S_2 which means that subsequent activities are carried out under the accountability of a substitute or

transformed accessor. It implies that a more accurate description of an access path would be $A_1 S_1 S_2 A_{11} R_{p1} S_3 A_{12} F_1 R_1 R_2$ with F_1 "thinking" it is being accessed by A_{12} using S_3. The fact that A_1 is the individual actually initiating the request is often unknown from S_2 onwards thus forcing reliance on system integrity or proxy access. This is all very well in secure computer environments, except that potentially unsecure environments, e.g. multi-domain components with multiple executions in single address space, may give rise to circumstances where significant exposures arise.

Secondly, system software may incorporate principles whereby, say, S_3 controls access to all the elements in F_1. S_3 demands unrestricted or universal access to F_1 with no consideration to the restrictions that apply to A_1.

Thirdly, applications can be developed in such a way that access to multiple application functions or computer resources are allowed. For example program R_{p1} could be used to access $R_1 \ldots R_5$. Unless the individual functions which are imbedded in R_{p1} can be protected as objects, A_1 cannot be restricted to R_1 and R_3 only. Definition of everything as both objects and subjects with the underlying complex and multiple security

interrelationships in any sizeable environment can only be described as ad hoc and intuitive with significant closure and accuracy risks in maintaining the underlying security system.

(b) The security implemented in the access paths used by A_1 and A_2 in Fig. 1 is irrelevant when A_3 uses S_4 and S_5 to access F_1. Perhaps A_3 needs to back-up F_1 and should not be permitted to gain access to the individual elements $R_1 \ldots R_5$ in F_1. Under these circumstances yet another security strategy needs to be devised.

(c) How is it actually determined what $A_1 \ldots A_3$ can access and are restrictions in the access paths necessary? Ad hoc approaches or exhaustive interviewing seem very antiquated and risky in relation to the technology being applied in complex environments.

(d) The number of permutations that need to be thought through in large systems make an intuitive "protect all" strategy impracticable in most environments.

2. NOTATION AND STRUCTURE.

The above has only been presented to illustrate the motive for researching more formal frameworks which can accommodate the issues in a more scientific manner. To provide solutions for the security problems which are associated with non-secure systems, the Path Context Model (PCM) was derived. The notation and structure which has been used is consistent with those developed in chapter 2. Using that structure, the following propositions are motivated:

(a) It is obvious that an automaton can be constructed which accommodates the baggage collection vehicle, security profile and validator. Whether the exercise is initiated by multiple machines, tapes or heads, classic computer science holds to the extent that they can be reduced to one automata such as a Turing Machine.

(b) The function of the validator is to examine a baggage vector and to apply the restrictions contained in the security profile in order to generate a condition which grants or rejects access to an object. Consequently there should be commonality between the descriptions of the baggage collection vehicle and security profile thus suggesting that only one grammar needs to by synthesised and presented. It is therefore not necessary to deal with each component of the security system separately.

(c) It could be argued that classic secure system theory should apply to computer security in general and that an attempt to secure potentially non-secure environments is invalid. It is submitted that reality dictates otherwise and that potentially non-secure phenomena will be present in the foreseeable future. Another paradigm whereby secure computer environments represent a special case of a more general computer security philosophy is therefore necessary. In this respect the following additional points are made :

(i) Secure computer environments in the traditional sense represents a special case of potentially non-secure environments as defined in this chapter.

(ii) In potentially non-secure environments additional information and tracking, the content of which can be derived from first principles, is necessary to manage computer security.

(iii) As a result of the risks in potentially non-secure environments simple issues such as passing of access rights, establishing security classes and simple subject-object relationships cannot fully describe and provide the degree of computer security which is normally acquired.

(iv) Many commercial systems and vendor system software follow the potentially non-secure route. This is evident from the small number of claims about environments which are secure in the classical sense. The application of the theory contained in this chapter therefore lies in a wider and more complex area.

(v) Theoretically an object can be any component and can therefore be located anywhere in E. It is submitted that any service request is ultimately the responsibility of a person or primary accessor A_1 although accessor transformations may take place among domains or other components. For purposes of this chapter, however, it is appropriate to consider a set A of accessors. To access an object in E, any A_1 would make use of various components of E. The combination of A, D, S, I and O for a given service request is defined as an access path in E thus creating a set of access paths M which consist of the various components in E. It is also possible to define for any $M_i \in M$ a set of conditions or restrictions in terms of any of its components or combination thereof which need to be enforced for access to be granted to a pre-defined object O_i.

Another way of defining restrictions is by creating a concept of permitting and forbidding conditions. The former dictates under which

circumstances access to O_j can be obtained whereas the latter prohibits access in the event of certain circumstances being present. The following examples illustrate this concept :

(a) An object O_i may be accessed via M_1, M_2, M_3 (permitting condition) but not via M_4, M_5 (forbidding conditions).

(b) Object O_1 may only be accessed via domain D_j (permitting condition) but not if $S_1 S_2 I_1$ forms part of any access path (forbidding condition).

(c) Object O_1 may only be accessed if the request was initiated from domain D_2 with the specific request if $I_i \in I$ in E represented an encrypted transmission (permitting condition) but not if this request was in the clear (forbidding condition).

It therefore appears that the mechanism of permitting and forbidding context conditions can be used very constructively in potentially non-secure environments to create more flexible security systems.

3. RANDOM CONTEXT GRAMMARS.

The properties of Random Context Grammars [10] provide some interesting ways in handling circumstances where it is necessary to determine the occurrence, presence or absence of events or conditions, or even a combination thereof, anywhere in a formal presentation of the string being examined, e.g. [9] and [10].

A random context grammar [8] is a 4-tuple $G=(V_N, V_T, P, S)$, where V_N, V_T and S have the usual formal language meaning, and P is a finite set of productions of the form $A \rightarrow \alpha(U;T)$ where $A \in V_N, \alpha \in (V_N \cup V_T)^*, U, T \subseteq V_N, U \cap T = \emptyset$. U is called the permitting context, and T is called the forbidding context.

Assume $x = \alpha_1 A \alpha_2$ is a sentential string over $(V_N \cup V_T)^*$. The production $A \rightarrow \alpha(U;T)$ can be applied to x, resulting in $Y = \alpha_1 \alpha \alpha_2$, if

(i) all elements of U appear somewhere in $\alpha_1 \alpha_2$, and

(ii) no element of T appears anywhere in $\alpha_1 \alpha_2$

The language L(G) generated by G, is defined as $L(G) = \{w / w \in V_T^+ \wedge S* \Rightarrow w\}$ where $*\Rightarrow$ has the usual formal language meaning.

4. PATH CONTEXT GRAMMAR

The objective is to define a path context grammar, (PC-grammar), which will generate valid access path, depending on the context restrictions of the production rules. By describing the grammar and its operation rather informally the intention is to illustrate the controlling mechanism of the grammar.

Assume the following environment E :

Accessors	$A = \{A_1, A_2 .. A_i .. A_r\}$
Objects	$O = \{O_1, O_2, .. O_j .. O_t\}$
Domain	$D = \{D_1, D_2, .. D_k\}$

To illustrate how such a Grammar functions, a number of examples have been used to show how restrictions are enforced. To achieve this objective in a clear and simplistic manner, a unique variable V_j which is associated with Object

O_j has been introduced to reflect the secured object prior to granting access. If access to O_j is permitted, O_j will be substituted for V_j thus resulting in a terminal symbol. This simulates the comparison of the baggage against the security profile. Otherwise the generation will terminate without a terminal string. Set $V = \{V_1, V_2, .. V_t\}$. Let $V_T = A \cup O \cup D$ be the set of terminal symbols and $V_N = V \cup \{S, \overline{S}\}$ the set of non-terminal symbols. Note that V is not the traditional V which represents $V_N \cup V_T$.

The PC grammar now is :

$G = (V_N, V_T, P, S)$ which can now be applied to the environment which has been described above.

(a) $S \rightarrow A_i \overline{S} D_k V_j$ where V_j is the unique variable associated with object O_j. This production means that any subject A_i can claim access to any object O_j which exists in domain D_k. S is the start symbol of the defined PC grammar.

(b) $\overline{S} \rightarrow D_j \overline{S} \qquad 1 \leq i \leq k$

(c) $\overline{S} \rightarrow D_j$.

Using productions (b) and (c), any access path, via any domain, between A_i and V_j (representing O_j), can be generated. The result after a number of attempts may be $x = A_i D_2 D_3 D_6 D_8 V_j$ where D_2, D_3, D_6 and D_8 are valid domains. Because V_j is not a terminal symbol in the PC-grammar, x is not a terminal string, and therefore not (yet) a valid access path. The objective is that V_j will change to O_j, signifying the granting of access of A_i to O_j if and only if the relevant conditions concerning domain crossings, had been satisfied. Suppose now that A_i can only access O_j if :

(i) A_i is cleared to access O_j,

(ii) domains $D_{k1}, D_{k2}, \ldots, D_{kr}$ must have been used with an access class of Passthru, and

(iii) no connection has been made with domains $D_{t1}, \ldots, D_{tn}$.

(d) The production $V_j \rightarrow O_j(A_i, D_{k1}, \ldots, D_{kr}; D_{t1}, \ldots, D_{tn})$ where $A_i, D_{k1}, \ldots, D_{kr}$ represent permitting conditions and $D_{t1}, \ldots D_{tn}$ forbidding conditions in this production, will now enforce precisely the conditions described above. In this way, any domain conditions can be included in the final production

which in the final instance, determines whether access is granted by examining the "history" of the access path generation process up to that specific stage.

To illustrate the concepts, another example which is based on the following environment is presented :

Assume the following in an environment E :

$\zeta = \{A_1, A_2, A_3, A_4\}$ as the set of accessors,
$\sigma = \{O_1, O_2, O_3\}$ as the set of objects,
$\delta = \{D_1, D_2, D_3, D_4, D_5, D_6\}$ as the set of domains, and
$V = \{V_1, V_2, V_3\}$ as the set of unique object associated variables as defined previously.

Let $V_T = \zeta \cup \sigma \cup \delta$ be the set of terminal symbols and $V_n = \{V_1, V_2, V_3, S, \overline{S}\}$ the set of non-terminal symbols.

The PC-grammar now is :
$G = (V_N, V_T, P, S)$, where P is defined below. (Note : only a subset of P has been defined).

Also assume the following restrictions of accessors and objects :

- Only A_1 and A_2 may access O_1
- O_1 resides in domain D_5
- A_1 and A_2 can only access O_1 via domains D_1, D_4 and D_5, or via domains D_1, D_5 and D_6
- Access by A_1 to O_1 may not be via domains D_2 and D_3 and access by A_2 to O_1 may not be via D_2.

The following productions in D will now enforce these conditions :

(a) $S \rightarrow A_i \overline{S} D_5 V_1 \qquad 1 \leq i \leq 4$

(b) $\overline{S} \rightarrow D_i \overline{S} \qquad 1 \leq i \leq 6$

(c) $\overline{S} \rightarrow D_i$

(d) $V_1 \rightarrow O_1(A_1, D_1, D_4, D_5; D_2, D_3)$

(e) $V_1 \rightarrow O_1(A_1, D_1, D_5, D_6; D_2, D_3)$

(f) $V_1 \rightarrow O_1(A_2, D_1, D_4, D_5; D_2)$

(g) $V_1 \rightarrow O_1(A_2, D_1, D_6, D_5; D_2)$

Productions (a) to (c) can generate any access path between any accessor A_i, $1 \leq i \leq 4$ and V_1.

Productions (d) to (g) control the actual granting of access. Only A_1 and A_2 appear in the permitting contexts of these

productions, hence should any other $A_j, i \neq 1,2$ be selected in production 1, the generation will terminate without a terminal string.

Production (d) requires an access path including domains D_1, D_4 and D_5, and excluding domains D_2 and D_3. This forces the access path to either passthru' via D_1 and $D_4 to D_5$, without initiating a session with D_2 or D_3. The other productions are similarly explained.

Note that the only reason why the validity of the access path could be checked was due to the availability of the complete history, termed the baggage, of the generation process. Given appropriate baggage it is possible to control access to objects to any degree of granularity using the components in E.

5. PC-LANGUAGES.

The language generated by a PC-grammar $G = (V_N, V_T, P, S)$ is $L(G) = \{w / w \in A_i \alpha O_j, A_i, O_j \in V_T, \alpha \in V_T^+$ and $S \Rightarrow * w\}$. L(G) consists of all valid access paths as specified by the production rules in P.

6. EXTENDED PC-GRAMMARS.

Path context grammars are "pure" random context grammars but are, per their definition, however, not powerful enough to describe some of the problematical situations in an environment E. For example specific sequences of domain occurrences in the generated access path or adjacency of domains in the path cannot be checked. If there is an access control requirement that a session with domain D_i must be initiated immediately after domain D_j had been exited, the model defined in the previous section is not powerful enough to provide access control. It is, however, possible to extend the model by allowing the permitting and forbidding context conditions of the production rules to be context sensitive. Obviously this holds for any component A, D, S, I, O in an environment E.

Assume that $< X , Y >$ indicates that X must occur directly adjacent to Y in a sentential string. Using this terminology in permitting and forbidding contexts conditions, enforcement of domain adjacency or forbidding thereof is possible.

Assuming the example in section 4 contained an additional requirement that the access path had to initiate sessions from domain D_1 directly to domain D_6, or from D_6 to D_1 the production (e) would be changed to $V_1 \rightarrow O_1(A_1, < D_1, D_6 >, D_5; D_2, D_3)$. This introduces

the enforcement of more stringent criteria to the extent that specific access paths could be specified as pre-requisites for access to objects whilst others could be prohibited. In this case $< D_1, D_6 >$ specifies that both D_1 and D_6 need to be present in any order. Assuming that the presence of D_1 and D_6 as well as their fixed adjacency are compulsory, D_6 is required to follow D_1, it is specified as $<< D_1, D_6 >>$. Potentially any restriction can be specified without loss of generality.

Given the notation in an environment E it is possible to include an indication of the integrity class of a domain such as differentiating whether the domain or a specific component is secure or insecure. This can be indicated by substituting the symbol D for a domain to $D(I_j)$ thus indicating that domain D has integrity class I_j.

Production (b) in the example now becomes to $\overline{S} \rightarrow D_i(I_j)\overline{S}$. Integrity class I_1 could for example mean that the data has been encrypted during transfer from one domain to another domain.

The symbols $D_i(I_1)D_j(I_k)$ in a sentential string therefore means that any data transmitted from D_i to D_j must be encrypted. By using this extension, as well as the context sensitivity of context conditions, the power of the model is significantly enhanced. To illustrate this an example is presented.

Assume the following additional requirements to the specifications in section 4 :

- O_1 is a secure object, and can only be accessed via either a dedicated line without encryption, or via a dial-up line, but then using encryption;

- the lines from domain D_1 to D_4 and from D_4 to D_5 are dial-up lines; and

- the lines from domain D_1 to D_6, D_1 to D_5, D_6 to D_5 and from D_5 to D_1 are dedicated lines.

Assuming that I_1 indicates that transmission has been encrypted while I_2 indicates the contrary. The resulting productions of the extended PC-grammar will now be :

(a) $S \rightarrow A_i \overline{S} D_5(I_j) V_1, 1 \leq i \leq 4, 1 \leq j \leq 2$

(b) $\overline{S} \rightarrow D_i(I_k)\overline{S}, 1 \leq i \leq 6, 1 \leq k \leq 2$

(c) $\overline{S} \rightarrow D_i(I_k)$

(d) $V_1 \rightarrow O_1(A_1, \ll D_1(I_1), D_4(I_1), D_5(I_i) \gg ; D_2(I_i), D_3(I_i))$

(e) $V_1 \rightarrow O_1(A_1, < D_1(I_2), D_5(I_1), D_6(I_i) > ; D_2(I_i), D_3(I_i))$

(f) $V_1 \rightarrow O_1(A_2, \ll D_1(I_1), D_4(I_1), D_5(I_i) \gg ; D_2(I_1))$

(g) $V_1 \rightarrow O_1(A_2, < D_1(I_2), D_6(I_1), D_5(I_i) > ; D_2(I_1))$.

The following conclusions can now be made :

(i) Production (e) requires domains D_1, D_5 and D_6 to be connected to in any order but without any other sessions being introduced. The integrity class of D_1 is not necessarily I_1, thus not enforcing encryption even though it may have been done.

(ii) Production (d) requires connections directly from D_1 to D_4 to D_5 in that order, but as the integrity class of D_1 and D_4 is I_1, encryption is required.

(iii) There still remains the issue of preventing a domain from appearing more than once in a generated access path. This is reasonable as multiple sessions with the same domain is possible in reality. In the example above an access path string

x may be created as follows :

$$x = A_1 D_1(I_1) D_4(I_1) D_5(I_i) D_5(I_i) D_4(I_1) D_1(I_i) V_1$$

The restrictions of production (e) above can be applied to x, resulting in A_1 gaining access to O_1, as the specified conditions are apparently satisfied. There is, however, an additional session between D_1, D_5 and D_4 in that order, which is not allowed. This results from a production being satisfied provided it finds at least one set of conditions satisfying its context conditions, and which is true for x. This is an inherent characteristic of random context grammars and it will therefore be necessary to check (e) whether **all** domains D_1 appearing are adjacent to a D_4 and not only one! This is easily done with the use of classical manipulation techniques in the field of Random Context Grammars.

9. CONCLUSION.

The reason for presenting this chapter somewhat informally is to illustrate the power of a relatively simple model in computer environments where security is a major problem area. It utilises simple production rules yet the principles of baggaging [e.g. logging] can be

simulated in complex, multidomain computer environments.

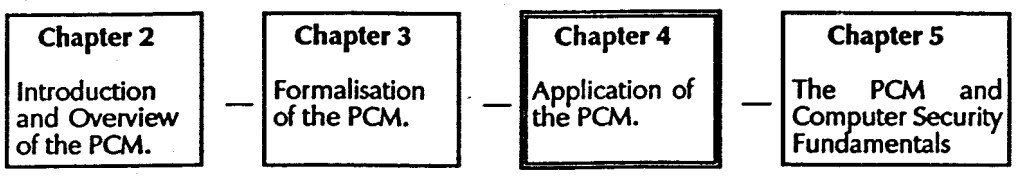
Initially this application of Random Context Grammars may appear simplistic, but experimentation suggests that a wide variety of heterogeneous computer systems such as LANS connected to hosts each utilising its own array of software components can be accommodated. In fact, the more complex the environment the more clearly and effective the model becomes. This is mainly the result of introducing the concepts of baggaging and access paths.

CHAPTER 4

APPLICATION OF THE PATH CONTEXT MODEL

Contents

Synoptic Perspective



APPLICATION OF THE PATH CONTEXT MODEL

SYNOPSIS

At this stage the reader has been exposed to the path context model and its implication as well as the underlying theory which is based on Random Context Sensitive Grammars.

This chapter has been structured to illustrate its application in complex computer environments and to establish the principles whereby the path context model can assist to automate computer security. Specific areas covered are heterogeneous multidomain computer environments, non-secure environments, automated security profile generation, automated exposure evaluation and automated security package evaluation. The objective is to highlight the power of the path context model in sophisticated computer environments.

1. INTRODUCTION.

The last few years have placed renewed demands on computer security as organisations have increasingly automated more of their activities. Technology has evolved rapidly over the same period to the extent that automation could be done at affordable costs. Resulting from this deployment of technology has been very large and complex systems which handle substantial activity volumes. Securing these kinds of environments requires an alternative perspective of computer security in the sense that there are demands for flexi-secure approaches, automated profile generation and automated exposure and security package evaluation.

To achieve security in complex computer environments has proved to be a formidable task and consequently the research project on which this chapter is based was undertaken. Although the initial scope was solely directed at computer security in complex systems, it was found necessary to re-visit many of the fundamental concepts in order to gain a better understanding of the main issues. In addition it was found more efficient to utilise a Random Context Grammar [8] to describe a model which is capable of accommodating complex technologies.

The purpose of this chapter is firstly to review some of the potential security problems which are associated with complex systems. A simple classification of multi-domain systems and those which allow multiple executions in a single address space has been used. Such phenomena are known to give rise to potentially non-secure conditions. Local area

networks, wide area networks and all facets of distributed and co-operative processing fall under the former while environmental software such as teleprocessing monitors and database management systems under the latter. Secondly, the whole question of flexi-secure approaches, automatic profile generation, automatic risk evaluation and computer assisted package assessment is dealt with using a path context model.

It is stressed at the outset that it is not being advocated that the approach adopted in this chapter is the only approach, or, at this stage, that it is perfect. The path context model which is described in this chapter does, however, address many of the known security concerns in complex potentially non-secure systems effectively and with a high degree of efficiency. Also of importance is its ability to provide a basis for automating many of the traditional manual security processes, a typical example being maintenance of security profiles.

To facilitate a comprehensive discussion, the chapter has been divided into sections which deal with security issues and the application of the Path Context Model in the environments which have been mentioned above while the remaining describe the flexibility and automation of computer security administration and evaluation.

2. MULTI-DOMAIN SYSTEMS

Multi-domain systems manifest themselves by the presence of more than one domain which consists of one or more of Wide Area Networks (WAN), Local Area Networks (LANS), Value Added Networks (VAN), Intelligent Data Terminal Equipment (IDTE) and distributed processing where system, data and/or application functionality could be distributed. In themselves a multi-domain system not necessarily create the problems; their implementation and features actually originate them. Typical examples are :

- Dynamic rather than user initiation of various network sessions.

- Dynamic initiating and rerouting of network sessions in other domains.

- Use of multiple system software components to carry out simple on-line requests for processing or retrieving information.

- Transparent multi-domain access to data.

- Transparent multi-domain application functionality.

- Loss of single user or individual identity during the processing path.

- Dynamic sharing of routines and processes among multiple software components.

- Parallel and/or concurrent execution of processes.

To illustrate these concepts a hypothetical system has been constructed, fig. 1.

Figure 1

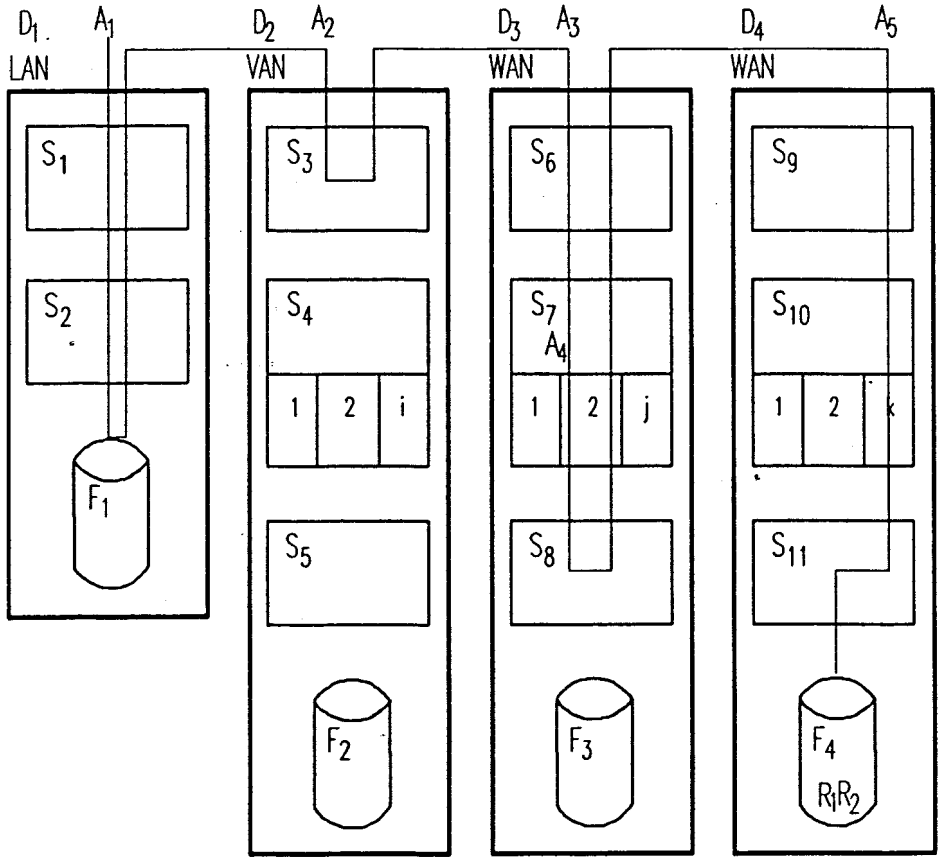


Fig 1. which consists of four domains covering a LAN, VAN and two WANS. The example provides the following service requests :

(a) A primary accessor A_1 signs onto a LAN denoted domain D_1 and utilises software components S_1 and S_2 with integrity I_1, representing for example problem state integrity and K_1 passthru' access class. F_1 is accessed for read only. S_2 dynamically initiates a VAN session in domain D_2 using a transformed accessor A_2.

(b) Within D_2 system software component S_3 is used with integrity state I_1 and access class K_2 as above. S_3 in turn initiates a session in D_3 by means of transformed accessor A_3.

(c) In D_3, S_6, S_7 and S_8 are used with S_7 initiating another accessor transformation to A_4. Integrity state I_1 and access class K_2 apply except that an application program denoted S_{72}, formed by S_7 and an application program block 2 in figure 1, has an integrity state I_2.

(d) S_8 in turn initiates a session in D_4 with transformed accessor A_5. S_9, S_{10}, and S_{11} are similar to D_4 with the only exception that F_4 is accessed with, say, an access class denoting update to elements R_1 and R_2.

The following represent the various baggage vectors for every domain :

$BV_{D_1}: A_1 D_1 S_1 I_1 S_2 I_1 F_1$

$BV_{D_2}: A_2 D_2 S_3 I_1$

$BV_{D_3}: A_3 D_3 S_6 I_1 S_7 I_1 S_{72} I_2 A_4 S_8$

$BV_{D_4}: A_5 D_4 S_9 I_1 S_{10} I_1 S_{10_k} S_{11} F_4 R_1 R_2$

where the symbols are as used througout this dissertation and already accommodate the access class where appropriate. Note that any particular domain does not recognise the identity of the accessors external to itself and therefore the need for transformation.
The complete baggage vector BV_c is simply a concatenated string of the above, i.e. $BV_c = BV_{D_1} || BV_{D_2} || BV_{D_3} || BV_{D_4}$.

It is shown in chapter 5 that computer security is based on the construct of restricting the capabilities of individuals in such a way that organisational segregation of duties is enforced. The only individual in the example which is described in figure 1 is, in fact, A_1 and the objective is then to determine whether A_1 can in fact gain access to R_1 and R_2 with a certain access class. In addition it is shown that unless there is a one to one mapping between the primary accessors and the transformed ones, or the baggage vector is comprehensive enough to deduce this, the system is non-secure. The implication of fig. 1 is that

unless A_1 actually "owns" A_2, A_3, A_4 and A_5 exclusively or that BV_c is available to the validator when R_1 and R_2 are accessed the system is potentially non-secure as it cannot be determined conclusively that it is, in fact, A_1 which is accessing R_1 and R_2.

At this point ad hoc, trial and error, scenario and/or if approaches are often resorted to in an attempt to introduce some security into the system. Where a large number of users, computer activities, devices and domains are present, the task is formidable. The issue now becomes of illustrating how the Path Context Model can be used to deal with such situations. Bearing in mind that the ultimate objective is to determine whether a primary accessor which requires access to objects operates within the scope of their segregation of duties in the organisation, the following steps address the process :

(a) Define the primary accessors.

(b) Define the resources to be accessed, e.g. files, data elements, etc.

(c) Determine the underlying access paths.

(d) Determine the Baggage Vector in every domain which could vary from system control blocks to logging mechanisms.

(e) Insofar as the complete Baggage Vector or even the underlying Baggage Vectors contain insufficient information to permit conclusive proof that the computer security objectives have been achieved, there are potential security exposures.

(f) Sometimes the environment can be simplified by not making use of some of the features thereby providing the basis for being able to secure the system. For example in fig. 1 A_1 need not sign on to the LAN but directly onto D_3 thus eliminating the necessity to deal with D_1 and D_2 from a security perspective.

To formalise this process generically the following steps are provided which use PCM and the underlying structure :

i. Define the set of access paths P in environment E within which computer security needs to be implemented.

ii. Define the set of Baggage Vectors BV according to the capabilities of the Baggage Collection Vehicle.

iii. Insofar as BV is deficient in that it cannot accommodate the full set of access paths, security exposures exist thus creating a potentially non-secure environment. The capabilities of the baggage collection vehicle is therefore the major determinant of

the environment's securability.

It is concluded that provided an environment E can be defined in terms of D, S, I, O and A and the Baggage Collection Vehicle is adequate any environment which could be contemplated was securable provided that the Security Profile and/or the Validator did not impose any further constraints.

3. MULTIPLE EXECUTIONS IN SINGLE ADDRESS SPACES (MESAS)

A phenomena which is frequently encountered in modern mainframe system software design is the execution of subtasks such as application programs in the same address space and under the control of a controlling task. Typical examples include teleprocessing monitors and database management systems.

Using fig. 1 as an example, S_4 S_7 and S_{10} represent components which utilise a MESAS architecture. Under these circumstances it is not unusual that an accessor say A_3 gains access to S_7. Once S_7 is activated, however, S_7 assumes control on behalf of A_3 but gains execute and passthru' access to application program S_{7_j} and S_8 respectively in its own right as accessor A_4. All subsequent initiated service requests take

place under the auspices of A_4 unless further accessor transformation takes place. In fact it can be said as a general statement that MESAS principles usually involve :

(a) Some form of accessor transformation.

(b) Transformed accessor control over a significant, if not all, portion of subsequent activities.

(c) Where an application program, say S_{7j}, is an integral component in restricting the capabilities of an individual and MESAS principles are present, there is a high probability that S_{7j} needs to be contributed towards a baggage vector to ensure effective security.

This concept can be formalised by simply stating that any MESAS based software component which cannot contribute to baggaging where functionality is incorporated as, say, subtasks that functionality is non-secure. Contribution to the baggage vector via the baggage collection vehicle may be inherent in the software itself or installation written and includes the environment's control blocks, status words, logs and vectors.

By way of conclusion it is stated that software which is MESAS based and often classified as non-secure, can be accommodated by PCM as another integrity state and the exposure assessed by the baggage deficiencies which may exist.

4. FLEXI-SECURITY.

Multi-domain environments are often characterised by large numbers of primary accessors, activities and hence access paths. Many activities often require no security, classed as public domain, whilst other may require a very high degree of security. The impracticability of securing everything to the highest level requirements and the resulting administrative burden is well known and requires no further comment.

Of interest then is to explore the possibility of a security system in an environment *E* which is capable of differentiating among access paths and applying different rules depending on the circumstances. For example certain activities may be classed as only accessible via encrypted lines while other may be available via VANS. The definition of PCM as a Baggaging Collection Vehicle and Security Profile with the validator matching the two achieves this as follows :

(a) Assuming that the Baggage Collection Vehicle is effective, it is only necessary to concentrate on the security profile.

(b) Within the Security Profiles any production is possible as the specification as described in chapter 2 can accommodate any component in E. In addition conditions which can reflect fixed or random adjacency are possible together with compulsory or prohibitive contexts. Either are able to handle restrictions of the nature as described previously. The power of Random Context Grammars allow very effective and efficient handling of simple and complex Security Profiles which in fact permits a flexi-secure approach. A more rigourous treatment of these concepts has been presented in chapter 3.

5. AUTOMATED COMPUTER SECURITY SUPPORT.

Having introduced complex computer environments and flexi-secure principles, it is submitted that unless support for security in such environments can be automated, it is unlikely that workable security can be effectively implemented and maintained without substantial administration support. The real benefit of this type of research project would therefore lie in its ability to provide the ground rules for structuring automated computer security support. Three areas, namely automatic profile generation, automatic risk evaluation and computer assisted security package evaluation have been explored using PCM.

5.1. Automatic Profile Generation.

Let PCM be the grammar with specifications which define the baggage collection vehicle (BCV) and security profile (SP). The Validator then serves as the mechanism which compares the underlying Baggage Vector (BV) and Security Profile derivations and allocates a value of TRUE if the security profile can be derived from the Baggage Vector and FALSE if not. By introducing a notation of individual accessor profiles SP_j representing deviations of SP the foundation for automatic profile generation can be established. An accessor profile SP_j is defined as well constructed iff :

(a) It must be derived from BCV.

(b) It is well formed by containing derivations which incorporate A, D, S, I and O.

(c) There is closure to the extent of mapping the primary accessor A_1 with secured objects to which A_1 has access.

An Automatic Profile Generator can therefore be described as an automaton which generates well constructed accessor profiles SP_j from an external input source. The formalism already achieved resolves the details about the automatic profile generator itself while giving rise to questions about the input. Chapter 5 provides ground rules to determine

the capabilities of the various primary accessors and provides insight into interrelationships between the various components such as policies, users, ownership access rights, etc. It is submitted that unless these criteria can be incorporated into the computer system by means of resources such as a resource data directory, true automatic profile generation is impracticable. The implications are that this type of security problem is no longer solely a function of computer security systems, but should, in fact, migrate to other areas thus necessitating a shared responsibility for computer security. Essentially the inadequacy of security theory or techniques are no longer the limitation factor. Chapter 3 for example introduces a high degree of formalism to a number of computer security issues. In practical terms it means that the theoretical foundations to automatically generate well constructed security profiles exist provided that appropriate external input is available.

5.2. Automatic Risk Evaluation.

Using the notation which was used to introduce Automatic Profile Generation the definition of risk is defined as :

(a) SP_j not being derivable from the BCV.

(b) BV and SP_j not being well formed.

(c) BV and SP_j not being closed.

Essentially this means that risk can be viewed as the set of partial derivations which results from (a), (b) and (c) above. Risk is therefore the elements in the derivations which, if they were included in the above definitions, would have constituted no risk. Risk can be further extended to reduce the probability of primary accessors exceeding their security profile by means of "Unauthorised Access". Under these circumstances compulsory and/or prohibitive conditions have been introduced against secured objects, i.e. $O(B;F)$ with B being compulsory and F prohibitive conditions. The term "conditions" broadly means any derivation of BV.

In real world environments it is a fairly trivial exercise to model the Baggage Collection Vehicle and Security Profile to determine the risks based on the above-mentioned criteria. The more complex the environment being addressed, the more effective PCM has been found to deal with risk, particularly in multiple heterogeneous cross domain environments.

5.3. Automatic Package Evaluation.

There are two ways in which this approach can be applied to assist with the evaluation of a computer security package :

(a) Use the principles which have been described under automatic profile generation. Insofar as the various package's security profiles cannot accommodate the SP_j which can be described as requirements, the package is deficient.

(b) Use the Automatic Risk Evaluation criteria to the extent of applying them to an environment E and determining whether the package addresses the risks or, alternatively, apply them to the security package and determining the potential risk. The power of Random Context Grammars provide very interesting opportunities in modelling various security alternatives. To appreciate this a great deal of formalism is required. Chapter 3 contains further detail in this regard.

6. CONCLUSION.

A computer security model termed a Path Context Model which is based on Random Context Grammars has been applied with interesting results in complex computer environments which could be classified as potentially non-secure according to traditional security theory. It also provides the basis for defining and structuring automated computer security support.

The advantage of this approach is the efficient and effective way in which controversial security issues can be handled. Its real potential, however, lies in providing the basis for automatic profile generation, automatic risk evaluation and automatic package evaluation.

CHAPTER 5

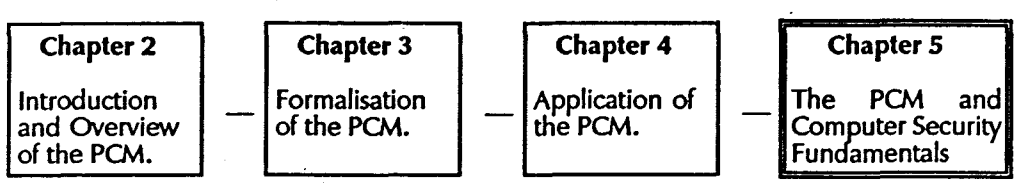
COMPUTER SECURITY FUNDAMENTALS

1. **Synopsis.**

2. **The PCM and Computer Security Fundamentals.**

3. **Appendix A - Towards Formalising Computer Security Fundamentals.**

Synoptic Perspective



COMPUTER SECURITY FUNDAMENTALS

Synopsis

The previous chapters have concentrated on the path context model, the underlying theory and its application. One area, however, has been omitted as it deals with business or organisational administrative issues. This area originates from problems in determining where the rules which are necessary to automate computer security support (chapter 4) originates. It was deemed necessary to explore computer security fundamentals further and attempt to introduce some formalism. Whilst the article which relates to this chapter covers the fundamentals it does not introduce complex theory as it requires extensive multi-disciplinary skills. An attempt has, however, been made in Appendix A to formalise the principles as a basis to stimulate further research.

1. INTRODUCTION.

It is generally accepted that computer security is associated with the control of access and protection of computer controlled resources. Whilst this may appear to be a fairly obvious statement, there are nevertheless difficulties when one is trying to deal with the potential ambiquity of terms such as protection, control and resources. This is highlighted even further when attempting to develop principles for the automation of security administration and access control across multiple, heterogeneous computer system boundaries. It is therefore a situation of not only dealing with the intricacies of security in complex computer environments, but also to determine the "resources" requiring securing, on what basis this securing needs to be done and who the various players are. One way of addressing these issues is to create an organisational policy which reflects users and their positions, computer controlled resources involved, the operations that need to be performed, the domains for which the policy applies, the authority to implement policies and, finally, the access rules which reflect authority. A good exposition of this type of approach is outlined in [12]. Essentially this attempts to mobilise an organisation into defining its security requirements. It assumes that the various people in an organisation actually comprehend the principles which ultimately result in access restrictions. This is, however, sometimes not the case with the result that computer security becomes an ad hoc exercise, often with many flaws. Probably the best way to illustrate this is by way of example. Assume a situation where two business transactions T_1 and T_2 need to be performed by two separate persons A_1 and A_2. Now assume that T_1 and T_2 are subsequently automated and incorporated into one application program R_{p1}.

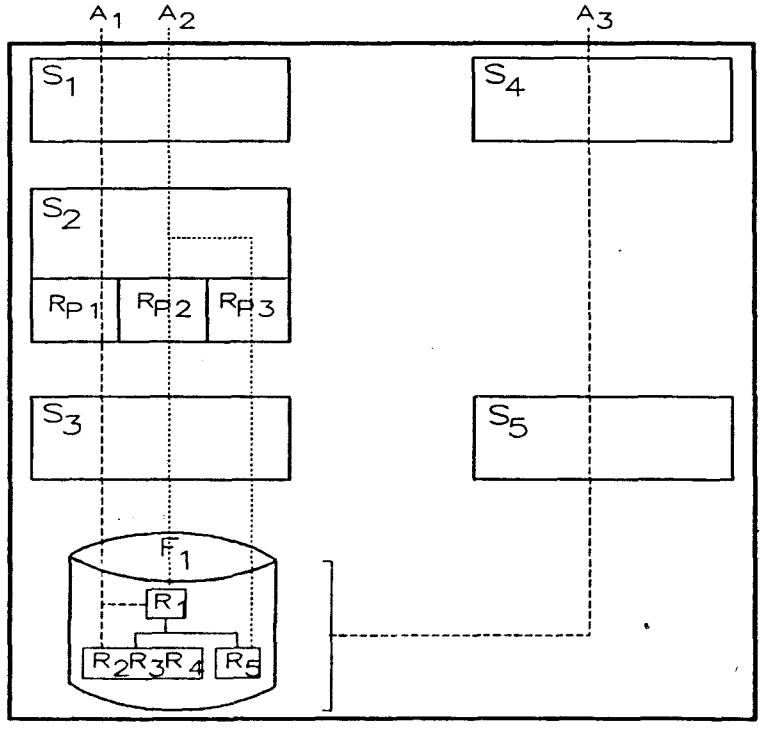
Readers familiar with application development, even using CASE tools, are very likely to be familiar with such situations. Often well-known application packages incorporate this type of arrangement. A situation arises where the business requires A_1 to have access to the T_1 component of R_{p1} and A_2 to the T_2 component in R_{p1}. Obviously the question in security terms becomes the definition of the resource to be protected; in this case the functions embedded in R_{p1}. Hopefully this simple example shows the necessity for formalising the derivation of policy.

The objective of this chapter is to describe a framework or model which provides insight into the complexities which underly computer security principles. It takes the concepts in discussions such as [12] a step further by attempting to explore organisational fundamentals. This chapter has been derived from a research project which was aimed at providing an understanding of security in potentially non-secure computer environments. Computer environments which contain commercially available local area networks, wide area networks, teleprocessing monitors and database management systems often give rise to non-secure environments, usually as a result of the architectures involved. These environments, however, frequently require a high degree of computer security and risk evaluation and hence this type of research has wide application.

One way of introducing some of the computer security fundamentals in potentially non-secure systems is by way of example. The term potentially

non-secure will become more obvious later on. Fig. 1 represents a fairly conventional computer system with three primary accessors, A_1, A_2 and A_3, always individual persons, issuing requests to access various elements R_1 to R_5 which are contained in a file F_1. R_1 to R_5 could represent data elements in a database record of which the database itself is contained in F_1.

Figure 1



Assume that the restrictions that need to be enforced are that A_1 should only be able to access R_1 to R_4 in F_1, A_2 to R_1 and R_5 and A_3 can control or manage F_1 without gaining access to the elements in F_1. In a typical computer environment each primary accessor would initiate various system software components, denoted S_i for the i^{th} component, or application programs,

denoted R_{pj} for the j^{th} application program. To reflect the complete access process, the concept of an access path which is formed by the various system components that need to be activated to service requests is introduced. For example $A_1 S_1 S_2 R_{p1} S_3 F_1 R_3$ and $A_3 S_4 S_5 F_1$ are two access paths in this environment. Obviously multiple access paths do exist, the issue being that in any environment E the number of access paths are finite. A short discussion of some typical security problems are presented by reference to the above example :

(a) The access paths which were used to illustrate its definition show intuitively that simple object-subject relationships are no longer an effective way of describing the security requirements in environments such as in Fig. 1. This effectiveness of object-subject relationships is further reduced by phenomena in system software design which, firstly, transforms the primary accessor, say A_1, to A_{11} and A_{12} for system software components S_1 and S_2 which means that subsequent activities are carried out under the capabilities of a substitute or transformed accessor. It implies that a more accurate description of an access path would be $A_1 S_1 S_2 A_{11} R_{p1} S_3 A_{12} F_1 R_3$ with F_1 "thinking" it is being accessed by A_{12} using S_3. The fact that A_1 is the individual actually initiating the request is often unknown from S_2 onwards thus forcing reliance on system integrity or proxy access. This is all very well in secure computer environments, except that potentially

unsecure environments, e.g. multi-domain components with multiple executions in single address space may give rise to circumstances where significant exposures arise.

Secondly, system software, such as a data base management system, may incorporate principles whereby, say, S_3 controls all access to the elements in F_1. S_3 usually demands unrestricted or universal access to F_1 with no consideration to the restrictions that apply to A_1.

Thirdly, applications can be developed in such a way that access to multiple system functionality is allowed. For example R_{p1} could be used to access R_1 to R_5 by incorporating multiple functions which access the various elements in it. Unless these individual functions in R_{p1} can be protected as objects, A_1 cannot be restricted to R_1 to R_3 as R_{p1}, defined as one object, allows access to all the data elements. By defining everything as both objects and subjects with the underlying complex and multiple security interrelationships in any sizeable environment the result can only be described as ad hoc and intuitive with significant closure and accuracy risks in maintaining the underlying system.

(b) The security implemented in the access paths which are used by A_1 and A_2 in Fig. 1 is irrelevant as A_3 uses a separate access path defined

by S_4 and S_5 to access to F_1. For example assume that A_3 has the responsibility of backing-up F_1. A_3 should therefore not be permitted to gain access to the individual elements R_1 *to* R_5 which are stored in F_1. Under these circumstances yet another security strategy needs to be devised whereby A_3 cannot access F_1 at the data element level of granularity.

(c) How is it actually determined what $A_1 \ldots A_3$ can access and are restrictions in the access paths necessary? Ad hoc approaches or exhaustive interviewing seem very antiquated and risky in relation to the technology being applied in complex environments.

(d) The number of permutations that need to be thought through in large systems make an intuitive "protect all" strategy impracticable in most environments.

The above has been presented to illustrate the motive for researching more formal frameworks which can accommodate computer security issues in a more scientific manner. Of particular significance, however, is to question what computer security is all about as is evident from the above example that there is both a organisational administrative and technology component involved.

2. COMPUTER SECURITY FUNDAMENTALS

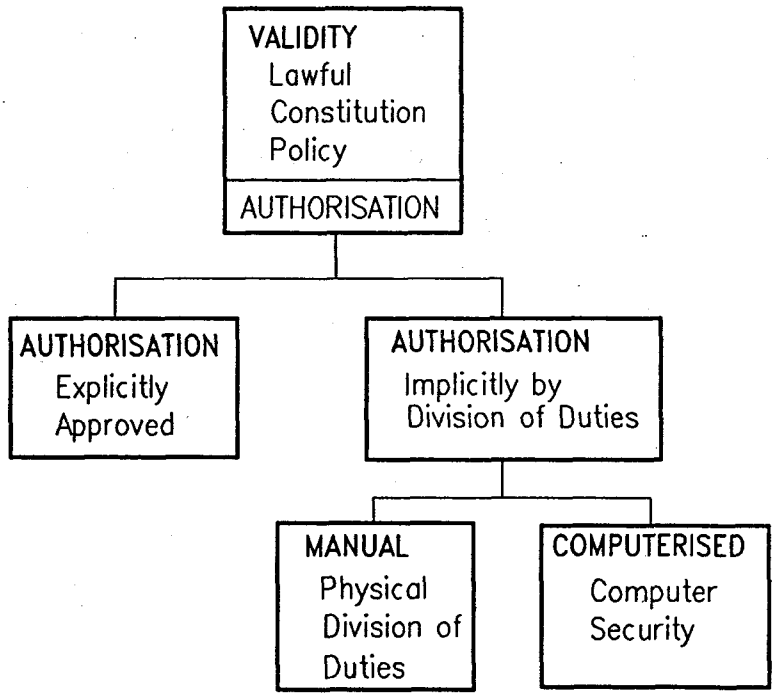
Principles of internal control in organisations have received a great deal of attention over the last few decades to the extent that they provide a sound point of departure for deriving formal propositions relating to computer security, particularly in commercial environments. Although numerous references which deal with internal control are available, [11] is presented as an internationally acceptable source of reference. Its usefulness further extends to its coverage of computer environments.

The objective of computer security or protection of any resource is ultimately an organisational issue. For example the components that need to be secured and the degree of security required can, and often is, overridden by business principles. It is submitted that these so-called business principles are not always understood and is often a source of confusion. Even if a perfect model of computer security could be developed and it could not reflect organisational or business related principles, there is a high probability that its implementation would not be successful. This is particularly evident when the potential for automated computer security support is explored. As a result it was found necessary to introduce some formalism into the organisational arena as it ultimately affects computer security.

A convenient point to initiate a discussion of this nature is with the organisation's business activities. Consider a hierarchy which deals with

business activities in an organisation in terms of validity, authorisation, approval and segregation of duties. Validity refers to business activities in terms of being lawful, within the scope of the organisation's constitution and in terms of organisational policy. To ensure validity, however, it is necessary that any activity be authorised as being valid. Authorisation can be achieved explicitly by a technique of somebody simply approving an activity or implicitly by dividing the activity into a number of sub-activities which are then performed by a number of different persons. Authorisation is then said to be implicitly achieved on the basis that the involvement of a number of persons provides an acceptable degree of risk that the activity is valid. The same holds for retrieval of information as well as privacy and confidentiality. Fig. 3 represents this structure diagrammatically.

Figure 3



It is submitted that where explicit authorisation takes place by an approval process, no segregation of duties at a level below that of the approval process is necessary. This implies that a complex segregation of duties hierarchy in addition to an approval process constitutes redundancy. This process is evident in a small organisation where the owner by means of an approval process negates the need of a large staff complement which is necessary for implicit authorisation. The latter again is often found in larger organisations.

Now consider the person in a manual environment who is responsible for determining the procedures according to which business, and therefore segregation of duties, is conducted. The principle is quite simple. That person cannot participate in an implicit authorisation process as this could compromise the process. Consequently business system and procedures are documented by independent persons and an authorisation process is applied to it, that is explicitly or implicitly. In the event of this set of manual activities being computerised, the same principles apply. By implementing segregation of duties in the form of a programmer, operator, or security administrator the implicit authorisation of the computerised systems and procedures is achieved. If it were possible to obtain reliable explicit authorisation of a system, the above may not be necessary. In a micro computer environment one person is capable of performing all the roles; hence the issues which arise when attempting to introduce security.

In complex computer environments with a significant number of components the implicit authorisation process of systems and procedures is far more involved. One would therefore expect more segregation of duties and hence more functions need to be introduced. From the user side the issue is somewhat simpler. In the event of any business transaction which has been decomposed into multiple sub-transactions performed by different individuals to achieve implicit authorisation and then subsequently computerised, the same principle applies. This means that one has to examine the segregation

of duties in a computer environment in exactly the same way as a manual one. It is not a question of arbitrary policy or protection of resources, but that of generally accepted business internal control procedures.

The conclusion which can be drawn from the above is simple yet far reaching. It implies that computer security is the technique whereby segregation of duties is enforced in a computer environment. Per definition unauthorised access or unauthorised disclosure can be viewed as a contravention of the segregation of duties principles. It is, however, necessary to introduce the concept of risk. In a manual environment masquerading by means of forged signatures does occur. The degree of control and supervision over the process affects the risk. Similarly the nature and extent of computer security is used to achieve the same objective. Note that in both instances absolute control cannot be achieved as authorisation always has a degree of inherent risk.

It is not the intention of this chapter to introduce any formalism as an informal approach was deemed to provide a clearer presentation. However, an attempt has been made in appendix A to this chapter to formalise these concepts by making use of regular set theory. Having established some basic principles, their impact on the study of computer security fundamentals is considered significant and therefore presented in the following sections. The term "Validity Hierarchy" has been adopted to represent the fundamental concepts which underly computer security.

3. SEGREGATION OF DUTIES AND COMPUTER SECURITY

Having established a conceptual foundation which focuses on the Validity Hierarchy and access paths, it is possible to experiment with various issues surrounding computer security as well as to substantiate the usefulness of the approaches which have been adopted.

3.1. Computer Security as Implementing Segregation of Duties.

It is possible in practice to develop a validity hierarchy for an organisation which reflects the segregation of duties which are necessary to authorise those business activities which are best done so implicitly. In addition redundant procedures may be introduced arbitrarily to further reduce the risk of error and/or fraud. Assume some of these business activities have been automated. The only way in which an individual can perform these computerised activities is by activating an access path (in Fig. 1), say $A_1 S_1 S_2 R_{p1} S_3 F_1 R_3$ to perform activity T_1. If A_1 represents an individual with the delegated right to perform T_1 there is no problem. If A_1, say, is not entitled to perform T_2 (which, for example, could result in a service request $S_1 S_2 R_{p2} S_3 F_1 R_1$), and an access path $A_1 S_1 S_2 R_{p2} S_3 F_1 R_1$ exists, or $S_1 S_2 R_{p2} S_3 F_1 R_1$ is available to everyone, then some major concerns are evident. Essentially being represented is the similarity between segregation of duties and computer security. This implies that, in the event of the new access paths which are available in computer environment E not reflecting organisational structures,

there is a major problem. Assume that one application program R_{pT} incorporated the functionality of R_{p1}, R_{p2} and R_{p3}. Unless it is possible to construct a baggage collection vehicle and a security profile (see section 5) which differentiates between the various functions in R_{pT} a situation arises where computer security serves little purpose. The key is not simply to implement security on whatever access paths are present to satisfy an uninformed audience, but to have computer security reflect the required organisational segregation of duties. The result is that unless a security system can be synthesised such that certain requirements are met, the computer security system is ineffective and a situation entailing risk is present. The following criteria are proposed :

(a) Valid activities by means of implicit authorisation are only possible if the security system and the access paths meet the organisational prerequisites in terms of segregation of duties.

(b) In the event of the computer environment E containing additional access paths, it implies that computer activities which exceed business restrictions are possible.

(c) Where the computer environment E and the security system cannot reflect business segregation of duties requirements at the level of an

individual person, then E has not necessarily been constructed to accommodate the business requirements. A computer security system may in place but has limited application.

In this context all security related policies, administrative arrangements and approaches are aimed at ensuring the effective implementation and operation of segregation of duties with the use of a computer security system as a technique in a computer environment. This affects the definition of terms such as protection and resource which have little meaning in a large computer environment.

3.2. Linking Individuals and Accessors.

In a computer environment E which incorporates a security system, one can introduce the concept of unactivated access paths on the basis that they can be predetermined without having to actually use them. An unactivated access path can accordingly be said to contain unactivated accessors. For example an access path $A_1 S_1 S_2 R_{p1} S_3 F_1 R_3$ is only activated when the unactivated accessor A_1 is associated with an individual person. In this context the identification and authentication of an accessor by whatever means only serves in establishing a link between the person and unactivated accessor. It serves no purpose in the enforcement of segregation of duties. This is important as an organisation can spend vast amounts of money and other resources on this

aspect without taking cognisance of the further restrictions that need to be applied. The end result could be a sophisticated encryption system but yet security being ineffective for other reasons. Unless a security profile which details the accessors rights is associated with the accessor at the correct level of granularity within the various access paths, E is said to be non-secure.

3.3. Encryption.

A great deal of literature is available on encryption and this chapter is not intended to discuss the techniques available. Instead it is deemed important to consider the role of encryption in the light of this approach to security.

Firstly it is submitted that access to information is a natural component of segregation of duties in the sense that an authorisation function is involved. The access to information can therefore be explicitly or implicitly authorised on the same principles which have already been dealt with in this section. Encryption is therefore considered as a technique whereby information relating to access paths and security systems is transformed to prevent disclosure of underlying information which could result in subsequent violation of segregation of duties. In itself it serves no purpose in establishing whether the validity hierarchy is complied with in the first place. This is the function of the security system. Again this is significant in the sense that the presence of encryption is a computer environment E which is potentially non-secure may have limited effectiveness from an overall security perspective.

3.4. Substitute Authorisation.

Substitute authorisation, or proxy logins or any other process whereby accessors are transformed will always give rise to non-secure conditions unless one of the following apply :

(a) Given any primary accessor A_1 there is always a one-to-one mapping between A_1 and any subsequent transformed accessors.

(b) In a given access path P enough information is collected within the access path that when the object is accessed it can be determined whether the primary accessor A_1 is permitted access to that object within the context of the information which has been collected and the security profile (see section 5) of A_1.

Unless one of these criteria are complied with, the security system cannot establish compliance with the validity hierarchy and the environment is non-secure.

3.5. Unauthorised Access.

It is submitted that unauthorised access is a natural extension of the segregation of duties concept. Essentially it relates to the risk of a person correctly activating

an unactivated access path. The issue becomes one of linking Individuals and Accessors as described above.

4. PRINCIPLES OF PATH CONTEXT CONCEPTS.

The value of this more business-like approach to security fundamentals lies in providing a better understanding of computer security fundamentals as well in its application to real world security issues. Whilst formal derivation provides evidence of theoretically sound concepts and propositions, their ability to describe and provide solutions to these problems is the ultimate test. This section is presented as a natural evolution of classic computer security which was developed in the seventies. [1], [3], [5]. [6] and [7] document these works.

Firstly, it is generally accepted that a security system is adequate if and only if it consists of the following three components :

(a) A tracking mechanism of accessors and access paths, termed a baggage collection vehicle in this chapter.

(b) A mechanism which describes and contains the restrictions which are imposed on accessors and objects. This has been termed the security profile.

(c) A mechanism which applies the rules and restrictions which are contained in the security profile to a specific value which is reflected by the baggage collection vehicle during a particular access or service request : The validator.

The reason for having introduced the Validity Hierarchy as a fundamental element of computer security is hopefully now apparent. It establishes a basis for sound evaluation of computer security requirements and the risk of ad hoc policies, standards and procedures which could even mirror the restrictions of a particular computer security system. Ultimately an objective test needs to confirm the adequacy of computer security or highlight exposures.

Security exposures can be defined as the degree of certainty which the three components of a security system provides that the principles of validity hierarchy can be implemented to meet organisational demands. This implies that an evaluation of baggage deficiency, profile deficiency and validator deficiency provide the key towards risk evaluation. Where the baggage collection vehicle as reflected in the computer system's control blocks, task vectors, logs and status words is incapable of recording events which are necessary to meet the underlying specifications, an exposure exists. In computer environments this would, for example, be reflected by a database update which is controlled by the database management system which was activated by an application program which in turn is controlled by a teleprocessing monitor. All the security system may see is the teleprocessing

monitor and database management systems and a series of transformed accessors. The true identity of the primary accessor may even reside in another part of the network. On the other hand a security profile which is incapable of describing the necessary restrictions that need to be enforced and a validator which cannot detect exceptions similarly create exposures. Essentially all three components need to be in place before implementation of computer security is possible. Note that issues such as split baggage collection, use of multiple physical mechanisms, or distributed security profiles do not affect the underlying concepts provided that logically the security criteria are complied with.

Even having the capability of building a very sophisticated security system does not ensure complete success. It is like a safe; there is always a more powerful device which is capable of cracking it. This introduces the concept of risk which is a measure of robustness. Any computer security system which is based on current technology is subject to risk mainly because of a phenomena best termed discreteness. Discreteness refers to the separate existence of the individual components which constitute the set of access paths in a computer environment E. As the individual accessor is separated from the unactivated accessor this gives rise to the risk of impersonation. Risk can only be reduced by the capabilities of the security system. The focus of the research on which this chapter is based is in this area of modeling requirements and security systems and matching them to determine exposures and risk.

5. DESCRIPTION OF A PATH CONTEXT SECURITY SYSTEM.

Two aspects concerning computer security have now been introduced. Firstly the concept of an access path as the various system components which are activated to service requests to access computer controlled resources and, secondly, the validity hierarchy. Obviously it is now necessary to examine their interrelationship in order to show how these concepts actually function. Again a rather informal approach has been adopted as the principles are deemed more important than their formal proof.

To explore the "interrelationship" mentioned, a number of options were explored in the original research :

(a) Multi-tape Turing Machines.

(b) Multi-head Turing Machines.

(c) Context free grammars.

(d) n-Dimensional Context Sensitive Language.

(e) Random Context Sensitive Language.

(f) Graph Theory.

(g) Various calculus approaches.

The following selection criteria were used to evaluate the suitability of the above options :

(a) Its ability to describe possible productions of a security system.

(b) The degree of formalism which it could provide in accommodating complex computer systems.

(c) Its ability to describe the dynamics of a security system with the Validator applying the Security Profile to the Baggage Collection Vehicle.

(d) Relative simplicity in applying it to the above items.

Turing machines and n-Dimensional Context Sensitive Language satisfies all the criteria except the point which deals with application simplicity. Context Free Grammars, graph theory and calculus approaches were discarded on the basis of an inability in their simple forms to describe the systems being researched. By far the most powerful, yet simple, approach was the use of Random Context Sensitive Languages [8].

A useful way of defining a security system was found to be a Path Context Model (PCM) which exists of three components, like any other security system :

- A set of restrictions which specify the security restrictions to be applied termed the Security Profile.

- A mechanism which collects information from the computer system during a service request against which the security profile can be applied. This has been termed the baggage collection vehicle of which a baggage vector reflects the individual values of an access path at any given point in time.

- A mechanism which performs the actual checking termed the validator.

6. PATH CONTEXT MODEL RELATED ISSUES

Although it does not fall within the scope of this chapter to provide a discussion of PCM and its actual operation, there are some issues which relate to the fundamentals which deserve some attention.

To achieve computer security objectives a computer security system consisting of three components, a baggage collection vehicle, security profile and validator together with their functions and specifications can be defined. A number of questions can be posed which explain how PCM fits together :

(a) The first question is how the baggage collection vectors are constructed? Essentially the idea is that the baggage collection vehicle should be viewed as logical mechanisms in the sense that the process of collection and storage of baggage vectors can be physically handled by more than one mechanism. The key is that the required information should be collected and transported so that appropriate checking of security objectives can take place. The baggage collection vehicle can take the form of logging mechanisms, exits, control blocks task vectors, etc., while the baggage vectors may manifest themselves in logs, control blocks, system status vectors, etc. Baggage deficiency is therefore a logical concept which may span any number of domains or system software components.

(b) The validator is invoked once access to a secured object is attempted. Two issues arise : How PCM detects a secured object and, secondly, how the validator is invoked.

It is submitted that the most effective identification of secured objects is the record of their definition in E. Alternatively such a situation can be simulated by an add on dictionary type system of resources and their security status. Components in E should be capable of invoking the validator when encountering a secured object.

(c) PCM contains the definition of an object on the security profile. It contains compulsory and prohibitive conditions which describe object access criteria rather than accessor restrictions to which it is linked. Again the concept involves a logical perspective. The actual implementation of PCM allows criteria which are unique to an environment E to dictate physical implementation whilst retaining the logical design and conceptual soundness at a higher level of abstraction.

(d) The role of identification, authentication and the risk profile have been defined. Techniques such as PINS and Passwords all relate to the risk of impersonation and can therefore be selected in terms of a specific environment's requirements. The tools and techniques which are available to enhance identification and authentication simply do that and nothing more.

(e) Finally, one of the interesting applications of PCM is by using it to provide a level of abstraction or a global view of security in nearly any environment in existence today. By means of the baggage collection vehicle, security profile and validator and simulation thereof, shortcomings in many situations can be identified and resolved. This area which was a major criteria of the research is referred to as automated computer security support and the results of the research form part of a separate chapter.

7. CONCLUSION.

In dealing with computer security issues the definitions of certain fundamental concepts have often confused the real issues. This chapter has concentrated on one framework which can be used to clarify some of the ambiguity which surrounds computer security. Starting with basic segregation of duties principles, the approach has been to formalise computer security fundamentals and showing how, ultimately, a path context model for addressing computer security in a wide variety of environments can apply them.

Although a detailed discussion of the model is not contained in this chapter, its major contribution lies in the investigation and formalising of the principles which form the base for automatic security profile generation, automatic security evaluation and flexi-secure computer security mechanisms.

Its application lies mainly in the area of potentially non-secure computer environments which represents a substantial portion of real-world computer environments. The model also permits a level of abstraction and simulation which can deal efficiently with heterogeneous and distributed computer environments.

CHAPTER 6

EVALUATION

Contents

APPENDIX A

TOWARDS FORMALISING COMPUTER SECURITY FUNDAMENTALS

To introduce some formalism for a given organisational environment a set of valid business activities or tasks BT is defined. For these activities to be carried out the organisation requires a set of people A. $BT \times A$ represents all the possible combinations of activities and people. In terms of the hierarchy which was presented above a set approval and/or segregation of duties are required to ensure that BT are authorised as being valid. This clearly prohibits the implementation of $BT \times A$ as is and necessitates the introduction of a segregation function f such that $BT_a = f(BT \times A)$. BT_a reflects the organisation's policy in terms of the activities which various persons in the organisation can perform. Note that $BT_a \subseteq f(BT \times A)$.

Given a computerised environment E as described in the notation section BT would be implemented by means of computer activities CT and manual

activities M. To reflect $B_a = (BT \times A)$ two further sets CT_a and M_a are defined such that $CT_a = g(CT \times A_{c_t})$ and $M_A = h(M \times A_m)$ where $A = A_{c_t} \cup A_m$ and g, h are segregation of duties functions. Whilst BT and M are intuitively clear, CT is not readily apparent.

The question surrounding CT is its relation to the computer environment E. An examination of CT suggests that to perform any CT_i, $CT_i \in CT$ requires an access $P_i P_i \in P$ where P represents all access paths in P. Note that the definition of P includes a set of primary accessors. An interesting phenomena arises whereby P represents an unactivated path. An access path is only activated when $A_c T_i$, $ACT_i \in A_c T$ is associated with it by means of a logon process which creates the linkage or mapping between the two. An invalid access is created when a given A_{CT_j}, $A_{CT_j} \in A_{CT}$ is logged on to an accessor which has not been assigned to A_{CT_j} . CT_a can now be redefined as $CT_a = g(P \times A_{CT})$ where g represents the segregation of duties function as described above.

Finally, by examining BT and its relation to CT and M, it is concluded that $BT = CT \cup M$. In the event of $BT \subset CT \cup M$ it implies that invalid activities have been implemented in the organisation whereas $BT \supset CT \cup M$ implies not all valid activities are carried out in the organisation.

PROPOSITION 1.

Computer security can be defined as a technique for implementing segregation of duties in a computer environment.

Proof.

Given a set BT of valid business activities implemented by means of a set CT of computerised and M of manual activities such that $BT = CT \cup M$. To implement CT and M it is necessary to introduce a set of persons A, A_{CT} and A_m such that $A = A_{CT} \cup A_m$ and segregation of duties such that $BT_a = g(CT \times A_{CT}) \cup h(M \times A_m)$. The actual implementation of CT is by means of P. Note that P is the implementation of CT which in turn is related to BT. CT therefore reflects the organisational requirements whereas P is the actual implementation. Unless $BT_a = S(P \times A_{CT}) \cup h(M \times A_m)$ the above validity hierarchy is potentially violated with S being the security function. Furthermore, unless there is a unique one-to-one relationship between the accessor in P, say A_i and A_{CT_i} such that $A_i = A_{CT_i}$ then $S(P \times A_{CT}) \neq g(CT \times A_{CT})$. Note that P represents the unactivated paths whereas $S(P \times A_{CT})$ represent the activated access paths. Concentrating on the computer component only :

(a) Valid activities by means of implicit authorisation is only possible if $S(P \times A_{CT}) = g(CT \times A_{CT})$.

(b) $S(P \times A_{CT}) \subset g(CT \times A_{CT})$ implies that E has not necessarily been constructed according to the validity hierarchy thus compromising the concept of authorisation.

(c) $S(P \times A_{CT}) \supset g(CT \times A_{CT})$ implies that computer activities which are not of the business are possible.

Conclusion.

Computer security actually deals with ensuring that $S(P \times A_{CT}) = g(CT \times A_C)T$ with the objective of ensuring that the validity hierarchy can be satisfied. Unless specific approval procedures override $S(P \times A_{CT})$ computer security can be regarded as a technique which can be used in a computer environment to ensure implicit authorisation. This implies that it enforces segregation of duties. Where any invalid activity, including any form of access by unvetted accessors, say A_u are performed the activity is said to fall outside the sphere of activities which constitute validity and are there invalid. All policies, administrative arrangements and approaches are therefore geared to ensure effective implementation and operation of computer security.

The definition of objects therefore need to be done in such a way that it can be determined whether $S(P \times A_{CT}) = g(CT \times A_{CT})$. To illustrate the concepts which have been presented the following example is presented :

Examples of Validity Hierarchy.

Assume the following :

BT	A		BT_a
B_1 B_2 B_3 B_4	A_1 A_2 A_3 A_4	$f(BT \times A) =$	B_1A_1 B_2A_1, B_2A_3 B_3A_1, B_3A_2 B_4A_3, B_4A_4

M	A_m		M_a
$B_1 \rightarrow M_1$ $B_2 \rightarrow M_2$ $B_3 \rightarrow M_3$ $B_3 \rightarrow M_4$ $B_4 \rightarrow M_5$	A_1 A_3 A_4	$h(M \times A_m) =$	M_1A_1 M_2A_3 M_3A_1 M_4A_2 M_5A_3, M_5A_4

CT	A_{CT}		CT_a
$B_2 \rightarrow C_1$ $B_2 \rightarrow C_2$ $B_3 \rightarrow C_3$ $B_3 \rightarrow C_4$	A_1 A_2 A_3	$g(CT \times A_{CT}) =$	C_1A_1 C_2A_1, C_2A_3 C_3A_1 C_4A_2

P	A_{CT}		P-activated
$C_1 \rightarrow A_1S_1S_2F_1R_1$ $C_2 \rightarrow A_1S_1S_2F_1R_2$ $C_2 \rightarrow A_3S_1S_2F_1R_2$ $C_3 \rightarrow A_1S_1S_2F_1R_3$ $C_4 \rightarrow A_2S_1S_2F_1R_4$	A_1 A_2 A_3	$S(P \times A_{CT}) =$	$A_1 \rightarrow A_1S_1S_2F_1R_1$ $A_1 \rightarrow A_1S_1S_2F_1R_2$ $A_3 \rightarrow A_3S_1S_2F_1R_2$ $A_1 \rightarrow A_1S_1S_2F_1R_3$ $A_2 \rightarrow A_2S_1S_2F_1R_4$

The actual implementation of BT_a in terms of the manual and computerised activities are :

$B_1A_1 = M_1A_1$

$B_2A_1 = C_1A_1 \cup C_2A_1 = A_1 \rightarrow A_1S_1S_2F_1R_1 \cup A_1 \rightarrow A_1S_1S_2F_1R_2$

$B_2A_3 = M_2A_3 \cup C_2A_3 = M_2A_3 \cup A_3 \rightarrow A_3S_1S_2F_1R_2$

$B_3A_1 = M_3A_1 \cup C_3A_1 = M_3A_1 \cup A_1 \rightarrow A_1S_1S_2F_1R_3$

$B_3A_2 = M_4A_2 \cup A_2 \rightarrow A_2S_1S_2F_1R_4$

$B_4A_3 = M_5A_3$

$B_4A_4 = M_5A_4$

The implications of computer security are that unless P is structured to enforce the requirements of BT there is a risk of invalid activities. For example assume there existed C_4 with $A_3S_1S_2F_1R_2$. Under these circumstances the segregation of duties requirements of B_3, B_3, A_1 and B_3A_2 are not satisfied. This illustrates the principle that computer security is geared towards enforcing division of duties.

PROPOSITION 2

Identification and authentication of an accessor only serves in verifying its claim of being who it is purported to be. Unless a security profile which detail these rights is associated with the accessor, E is potentially non-secure.

Proof

Assume an access path denoted by $A_iC_1C_2...C_nO_j$ with $C_i \in E\,(i = 1, n)$. Insofar as any accessors exist any given identification and authentication only serve to establish the credentials of one accessor. In terms of an access path definition $A_1C_1C_2...C_nO_j$ the identification process can only give rise to $A_i\emptyset$ where $\emptyset = NIL$. This does not permit any form of subsequent checking thus showing that an access path $A_i\emptyset$ cannot be used to prove anything conclusively. Additional information $C_1...C_nO_j$ termed baggaging is therefore necessary to allow subsequent checking.

Given an access path $P = A_i C_1 C_2 ... C_n O_j$ with identification and authentication as established previously. To achieve the objective of establishing $S(P \times A_{CT}) = g(CT \times A_{CT})$ (as described in the previous proof) it is necessary that $g(CT \times A_{CT})$ be reflected for an external reference to P in a form that $S(P \times A_{CT}) = g(CT \times A_{CT})$ can be verified. The existence of $g(CT \times A_{CT})$ is termed the security profile and obviously insofar as P cannot refer to a security profile, a similar situation to $A_i \varnothing$ is created. This leaves E non-secure.

PROPOSITION 3

Encryption is a technique whereby information relating to an access path P or a security profile $g(CT \times A_{CT})$ is transformed to prevent disclosure of the underlying information which could result in $S(P \times A_{CT}) = g(CT \times A_{CT})$ (as in proposition 1) being compromised. It services no function in establishing whether $S(P \times A_{CT}) = g(CT \times A_{CT})$.

Proof

Given a set of access paths P and a security profile $g(CT \times A_{CT})$ and that by applying an encryption or transformation algorithm T to P and $g(CT \times A_{CT})$ or any individual components $T(P)$ and $T(g(CT \times A_{CT})$ is derived. Assume that encryption can perform an active function in establishing that

$S(P \times A_{CT}) = g(CT \times A_{CT})$ then $T(g(CT \times A_{CT})$ and $g(CT \times A_{CT})$ do not necessarily enforce the same division of duties or $T(P)$ and P represent different access paths.

The basic principle of encryption T and de-encryption T^{-1} is that $T^{-}T(g(CT \times A_{CT})$ and $T^{-}T(S(P \times A_{CT})) = S(P \times A_{CT})$ thus implying that T does not impact on the security profile or the access path.

PROPOSITION 4

Substitute authorisation, proxy logins or any other process whereby accessors are transformed will always give rise to non-secure conditions unless one of the following apply :

(a) Given any primary accessor A_1 there is always a one to one mapping between A_1 and accessors $A_j, j > 1$ which is generated within an access path P.

(b) In a given access P enough information about P is maintained to establish when an object is accessed whether $g(CT \times A_1) = S(P \times A_1)$ for a primary accessor A_1. Define this information as a set of baggaging vectors BV.

Proof

Given an access path set P and a security profile set $g(CT \times A_{CT})$, A_{CT} can be extended to A'_{CT} with accessors generated by proxy logins, substitute authorisation or any accessor transformation. The security issue is still to show $S(P \times A_{CT}) x g(CT \times A_{CT})$ which can only be true if $S(P \times A'_{CT}) = g(CT \times A_{CT})$. This is so if there exists a one to one mapping such that effectively $A_{CT} = A'_{CT}$. Insofar as this is not the case, enough information needs to exist about P in BV in order that it can be established whether $S(P \times A'_{CT}) = g(CT \times A_{CT})$.

4. APPLICATION OF PATH CONTEXT CONCEPTS.

The value of this kind of research lies in it providing a better understanding of computer security as well in its application to real world issues. Whilst formal derivation provides evidence of theoretically sound concepts and propositions, their ability to describe and provide solutions to these problems is the ultimate test. The objective of the propositions which have been described above was to introduce some of the conceptual foundations as well as a notation and structure which can be used to provide a better understanding of computer security. This section is presented as an evolution of classic security theory which was developed in the seventies.

Having provided some formalism to the basic concepts in the previous section, the logical progression is to derive the specification of the underlying security model. Again use is made of a propositional format.

PROPOSITION 5.

A computer security system is adequate iff it contains the following three components :

(a) A tracking mechanisms of accessors and access, termed a Baggage Collection Vehicle in this paper.

(b) A mechanism which describes and contains the restrictions of an accessor or object and the rules which are associated with the accessing of objects. This has been termed the Security Profile.

(c) A mechanism, termed the Validator, which applies the rules and restrictions which are contained in the security profile to a specific baggage vector which is constructed by the baggage collection vehicle during a particular access or service request.

Proof.

In proposition 1 it was shown that unless $g(CT \times A_{CT}) = s(P \times A_{CT})$ there is a risk of invalid activities. This, however, addresses the macro or universal level thus creating the necessity for defining $CT_i \in CT_1 A_{CTj} \in A_{CT} 1 P_k \in P$ where $i, j, k > O$. The requirement for an individual accessor A_{CTj}' is $A_{CTj}CT_j = A_{CTj}P_j$ where j implies a relationship such that ultimately $g(CT \times A_{CT}) = S(P \times A_{CT})$ is satisfied. To ensure $A_{CTj}CT_j = A_{CTj}P_j$ it is necessary for $A_{CTj}CT_j$ to be defined (the security profile), P_j (the baggage vector) and a mechanism for comparing the two (the validator).

PROPOSITION 6.

Security exposures can be defined as the degree of certainty which the three components of a security system provides that $g(CT \times A_{CT}) = S(P \times A_{CT})$.

Proof.

In proposition 5 it was shown that for a single service request it is necessary that $A_{CTj}CT_j = AP_j$ with $A_{CT}CT'_j$ being the security profile and P_j the baggage vector and the comparison being made by the validator. In section 2 the definition of an environment E which is used to formulate access paths comprises $E = \{D \times S \times I \times O \times A\}$ with D ={Valid domains}$\times k$, S ={Valid Software Components}$\times k$, I ={Integrated States}$\times k$, O ={Valid Objects}$\times k$,

A ={Valid Accessors}$\times k$ and K ={Valid Access Classes}. In its simplest form $A_i O_J = P_j$ with all the risks associated with D, S and I and O capable of circumventing it. The form with $CT' = f(D, S, I, O)$ represents a finer degree of control over P. Insofar as an accessor is restricted in terms of D, S, I and O a less risky situation is produced as an accessor is highly restricted in terms of capabilities within p. By defining CT and P well formed if they contain restrictions in terms of D, S, I and O such that $CT = f(D, S, I, O)$ and $P = g(D, S, I, O)$ then security exposures are high to the extent that $S(CT' \times A_{CT}) = P$ with CT and P are not well formed. Insofar as $CT' \times A_{CT}$ and P, with both CT and P well formed, those components of D, S, I and O within CT and P which cannot be compared by the validator are redundant. A situation is created as if CT and P are not well formed thus resulting in high security exposures.

PROPOSITION 7.

A security exposure exists where an accessor A_i is transformed into or proxied by another accessor $A_j, i \neq j$ and the baggage vector bears no evidence of this fact when the validator is invoked to verify that $S(A_{CT} \times CT'_j) = P_j$.

Proof.

Where accessor transformation or proxy access takes place a consolidated baggage vector has a format $[A_i \ldots . A_j \ldots . O_j]$. Under these circumstances the

validator is capable to performing a comparison with the security profile and conclude A_i has accessed O_j. Assume split baggage vectors $[A_i.....]$ and $[A_j.....O_j]$. When the validator performs the above function there is inconclusive evidence unless $A_i=A_j$ or a one to one mapping between A_i and A_j exists. It cannot therefore be shown that $CT' \times A_{CT} = P$ and hence a security exposure exists. It is, however, obvious that a split baggage vector $[A_i...A_j]$ and $[A_j...O_j]$ does not fall under the above restriction provided both are available to the validator for reference when required.

CHAPTER 6

EVALUATION

Contents

1. INTRODUCTION.

The objective of this chapter is to evaluate the research project and the resulting dissertation in terms of scientific contribution. Whilst the results and their acceptance for international publication may be deemed noteworthy, it is felt that the scientific process and contribution to the development of a paradigm for computer security are equally important. Consequently the scope of this evaluation is wider than the original scope as set out in the introduction.

2. DIRECT CONTRIBUTION TO THE FIELD OF COMPUTER SECURITY.

The most significant achievement of this research lies in the Path Context Model (PCM) which originated. Although fairly simple, the structure and definition permits the implementation, evaluation and monitoring of computer security in environments which are potentially non-secure because of the manner in which technology is applied. Experiments with the Path Context Model shows its applicability in a wide variety of environments, particularly complex ones. Typically these consist of multi-domain, distributed and heterogeneous network environments such as wide area networks and local area networks.

A second area of achievement is the success with which organisational administrative principles and computer security principles could be

formally integrated to provide a number of fundamental laws which govern computer security. These laws essentially comprise the Validity Hierarchy or segregation of duties; identification, authentication and security profile relationships; accessor transformation and/or substitute authorisation, the essential components of a computer security system and risk of security exposures.

The above have contributed to a comprehensive or hollistic definition of computer security which has direct application in many organisations which utilise computer technology.

3. APPLICATION OF CLASSIC COMPUTER SCIENCE THEORY.

Any development in an area such as computer security needs to be based on formal theoretical principles. Absence to achieve this gives rise to the risk of propagating fragmented adhocracy. Whilst the theory which is utilised for such a purpose is not necessarily restricted, it was deemed more appropriate to attempt application of classic computer science theory. In this arena a new application of Random Context Grammars was found. It was established that the opportunities of these Grammars afforded unique properties in describing elaborate protection or restriction capabilities of secured objects. Two grammars, a Path Context (PC) Grammar and an Extended PC Grammar originated

from specifying the Path Context Model. Both rely on the properties of Random Context Grammars to create an alternative model for computer security.

4. REPRESENTING COMPUTER SECURITY.

One of the characteristics of mature research is the ability to accept the possibility that improvements, enhances or major leaps over any initial research can be made. The real contribution any individual researcher may make to any area should therefore not be restricted to the achievement of direct results but extended to include the understanding of the field or area being researched.

Our view of the above in relation to the field of computer security is that there are currently shortcomings in the following areas :

(a) Conceptual framework

(b) Reference disciplines

(c) Representation issues.

The project on which this dissertation is based has attempted to communicate the development of thinking which address the above. Specifically the Path Context Model which consists of the Baggage

Collection Vehicle, Security Profile and validator and interfaces with the Validity Hierarchy establishes a theoretically sound conceptual framework for addressing computer security. On such a basis it is possible to further evolve an understanding of what is becoming a very complex organisational issue.

Computer security is unique in the sense that it is a field which combines organisational administrative principles, technology and computer science. These serve as the reference disciplines which need to be considered when proposing alternative theories. In this context we have used set theory, access path principles, Random Context Grammars and have interfaced them to provide the contents of this dissertation. Of interest then is the contribution of this project of linking multiple disciplines to solve a multi-dimensional problem; in this case computer security.

Having established a conceptual framework and reference disciplines does not necessarily guarantee success. It is only when they are combined and represented in such a way that its actual functioning can be modelled and experimented with, that progress is made. The ultimate in representation is the ability to apply artificial intelligence or expert system principles to the problem as this requires a comprehensive understanding of the subject's representation. In this dissertation the proposals for automated computer security support

provide evidence of successful representation. Specifically the concepts of baggaging, an access path and a Validity Hierarchy permit representation of the conceptual framework within the scope of the reference disciplines.

By way of concluding remarks it is submitted that each of the various papers which constitute this dissertation contain the evolution of this representation concept from an overview to the detailed treatment of the individual areas and provides an unique contribution towards not only an understanding the field of computer security but also representation of multi-discipline problem areas.

5. INTERNATIONAL ACCEPTANCE.

The unique contribution of this research is evidenced by the justification to publish the complete dissertation as four independent articles in international publications.

6. CONCLUSION.

From the account provided above of the research and therefore of the dissertation, it is submitted that a significant contribution has been

made, not only to the field of computer security, but also to the areas of applying classic computer science theory, problem representation and the scientific process in multi-reference discipline problem domains.

CHAPTER 7

BIBLIOGRAPHY

Contents

REFERENCES.

1. D.E. Bell and L.J. LaPadula, "Secure Computer System : Unified Exposition and Multics Interpretation", Report ESD-TR-75-306, MITRE Corporation, Bedford, MA (March 1976).

2. W.H. Boshoff, "The Interface between Application and Integrity Controls in Modern Computer Systems", Dissertation in fulfilment of a Masters Degree in Economic Sciences, Rand Afrikaans University, May 1985.

3. D.E. Denning, "A Lattice Model of Secure Information Flow", Communications of the ACM 19, No. 5, 243-250 (May 1970).

4. P.A. Karger, "Authentication and Discretionary Access Control in Computer Networks, Computers & Security 5, 314-324 (1986).

5. B.W. Lampson, "Protection", Proceedings of the 5th Annual Princeton Conference on Information Science and Systems (1971), pp 437-443, Reprinted in ACM Operating Systems Review 8, No. 1, 18-24 (January 1970).

6. C.E. Landwehr, "The Best Technologies for Computer Security", Computer 16, No. 7, 86-99 (July 1983).

7. R.C. Summers, "An Overview of Computer Security", IBM Systems Journal 23, No. 4, 309-325 (1984).

8. Van der Walt, "Random Context Languages Symposium on Formal Languages", Oberwolfach. W-Germany, (1970). Abstracted in "Mitteilingen der Gesellschaft für Mathematik und Datenverarbeitung", Bonn.

9. S.H. von Solms, "Random Context Array Grammars", Proceedings of IFIP '80, Tokyo, Japan.

10. S.H. von Solms, "Node-label controlled graph grammars with Context Conditions", International Journal of Computer Mathematics, 1984, Vol. 15.

11. S.D. Halper, G.C. Davis, P.J. O'Neil-Dunne, P.R. Pfau, "Handbook of EDP Auditing and Supplements", Warren, Gorham & Lamont (1986).

12. J.D. Moffett and M.S. Sloman, "The Source of Authority for Commercial Access Control", Computer Vol. 21, No. 2, 59-69 (February 1988).