IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com
ITP4457

Formal Methods and Secure Systems Development

Gurpreet Dhillon[1] and Ian Hosein[2]

[1]College of Business, University of Nevada, Las Vegas, Las Vegas, NV, 89154-6009, dhillon@nevada.edu

[2]Computer Security Research Center, London School of Economics, Houghton Street, WC2A 2AE, UK

ABSTRACT

This paper argues that extensive reliance on formal methods to design security for computer based systems within modern business organizations falls short of fulfilling the purpose. The argument is conducted by evaluating various formal methods and reviewing the nature and scope of modern organizations. Since models are abstractions of reality, this paper presents an outline of the necessity conditions, restrictions and the nature and scope of the applicability because of the abstractions.

INTRODUCTION

Indeed designing security of information systems within organizations is a nebulous task. Organizations attempt to make their information assets secure in varying ways – may it be by incorporating latest technology or by applying generally accepted models, criteria and policies, with the hope that the systems would be secure. However various pieces of statistics indicating the level of security of computer based systems suggests otherwise (see Dhillon & Backhouse, 2000; Dhillon, 2001). Clearly this questions either the appropriateness of technology use in securing information systems or the use of various models, criteria and other formalisms in designing security of systems.

In this paper we explore the nature and scope of the formal methods in the design and development of secure systems. The paper is organized into five sections. Following a brief introduction, section two reviews various formal methods used in the design of secure systems. Section three discusses the merits and demerits of formal models in addressing security needs of modern enterprises. Section four presents a discussion and conclusion.

FORMAL METHODS FOR SECURE SYSTEMS DEVELOPMENT

A formal method for secure systems development is a process for building security into computer based systems while exploiting the power of mathematical notation and proofs. A formal method relies on formal models to understand reality and subsequently implement the various components. A model can be construed as an abstraction of reality and a mental construction that is embodied into a piece of software or a computer-based information system. Any function of a computer-based system can be viewed at two levels (Wordsworth, 1999):

The user view. The user view, which is elicited during requirement analysis for a system, records what a system should do. The user view is generally an aggregate of views of various stakeholders and is to a large extent independent of the details on the manner in which it will be implemented. The model that embodies the user view is the specification of the system

The implementation view. This view is build during system design and records as to how the system is to be constructed. The model that embodies the implementation view is commonly referred to as a design. In an ideal state the design and specification should adequately reflect each other.

An example of such formal methods and models is evidenced in the Trusted Computer System Evaluation Criteria (TCSEC) developed by the US Department of Defense (DoD). The TCSEC were a means to formalize specifications such that the vendors could develop applications according to generally accepted principles. The criteria were attempting to deal with issues of trust and maintaining confidentiality, integrity and availability of data from the perspective of the vendor. Hence the TCSEC represented the user view of the model. Soon many other such criteria were developed, such as the European Information Technology Security Evaluation Criteria, the Canadian Trusted Computer Product Evaluation Criteria, the US Federal Criteria for Information Technology Security, and most recently the Common Criteria.

In the realm of secure systems development, the user and implementation views have largely been overlooked and the formal security model tends to embody in itself the security policy. There are various kinds of policies such as the organizational security policy and the technical security policy. The language, goals and intents of various security policies are different, though the ultimate aim is to secure the systems. An example of a technical security policy is access control. The intent behind restricting access is generally to maintain confidentiality of data. Access control could either be mandatory or discretionary. Clearly the policy is motivated by the lack of trust in application programs communicating with each other, not necessarily the people though. This would mean that employees of a particular organization could make an unclassified telephone call despite having classified documents on their desks. This necessitates the need for an organizational security policy. However it might be difficult to describe the desired behavior of the people in formal language. Models tend to be simple, abstract, easy to comprehend and prove mathematically (Gasser, 1988) and hence have limited utility in specifying technical security measures alone.

The various formal methods for designing security tend to model three basic principles – confidentiality, integrity and availability. These have also often been touted as the core principles if information security management. In fact maintaining confidentiality was the prime motivation behind TCSEC. The US DoD wanted to develop systems which would allow for only authorized access and usage. For this reason the computer science community created ever so sophisticated models and mechanisms that considered confidentiality as the panacea of security. Clearly security of systems, more so for the commercial organizations, went beyond confidentiality to include issues of integrity and availability of data. The notion of integrity deals with individual accountability,

This paper appears in *Managing Information Technology in a Global Economy,* the proceedings of the Information Resources Management Association International Conference.

auditability and separation of duties. It can be evaluating by considering the flow of information within a system and interpreting areas where the integrity is at risk.

EVALUATION CRITERIA AND THEIR CONTEXT

The Trusted Computer Systems Evaluation Criteria were first introduced in 1983 as a standard for the development of systems to be used within the US Government, particularly within the DoD. This document established the DoD procurement standard, which is even in use today, albeit with some modifications. Although the original standards were set within the particular context of the military, their subsequent use has underplayed the importance of contextual issues. The DoD was largely concerned with safeguarding the classified data while procuring systems from vendors (Longley, 1991). The criteria list different levels of trusted systems, from level D with no security measures to A where the security measures are highly regarded. As one moves from level D to A the systems become more secure through the use of dedicated policies operationalized by formal methods and provable measures. Formal models such as the Bell La Padula, the Denning Information Flow Model for access control, and Rushby's model provide the basis.

BELL LA PADULA

The Bell La Padula model, published in 1973, sets the criteria for class A and class B systems in the TCSEC. It deals with controlling access to objects. This is achieved by controlling the abilities to read and write information. The Bell La Padula model deals with mandatory and discretionary access controls. It's two basic axioms being:

1. a subject can not read information for which it is not cleared (no read up rule)
2. a subject can not move information from an object with a higher security classification to an object with a lower classification (no write down rule)

A combination of the two rules forms the basis of a trusted system, i.e. a system that disallows an unauthorized transfer of information. The classification and the level in the model are not one-dimensional, hence the entire model ends up being more complex than it appears to be. The system is based on a tuple of *current access set*, *hierarchy*, *access permission matrix*, and *level function*.

The current access set addresses the abilities to extract or insert information into a specified object, based on four modes: execute, read, append and write, addressed for each subject and object. The hierarchy is based on a tree structure, where all objects are organized in a structure of either trees or isolated points, with the condition that all nodes of the structure can only have one parent node. The access permission matrix is the portion of the model that allows for discretionary access control. It places objects vs. subjects in a matrix, and represents access attributes for subject to a corresponding object. This is based on the access set modes. The level function classifies the privileges of objects and subjects in a strict hierarchical form with the labels: top secret, secret, confidential, and unclassified. These information categories are created based on the nature of the information within the organization and are designated a level of access, so that a subject could receive the relevant security designation. With respect to the level function, considering the two classes c1 and c2, the basic theorem is is that (c1,A) dominates (c2,B) if and only if c1 is greater than or equal to c2, and A includes B as a subset.

The development of the Bell La Padula model was based on a number of assumptions. First, there exists a strict hierarchical and bureaucratic structure, with well-defined responsibilities. Second, people in the organization will be granted clearance based on the their need to know in order to conduct work. Third, there is a high level of trust in the organization and people will adhere to all ethical rules and principles, since the model deals with trust within applications as opposed to people. For example it is possible to use covert means to take information from one level to the other.

DENNING INFORMATION FLOW MODEL

While the Bell La Padula model focused attention on the mandatory access control, the Denning Information Flow Model is concerned with the security of information flows. The Information Flow Model is based on the assumption that information is constantly flowed, compared and merged. Hence establishing levels of authority and compiling information from different classes is a challenge. The Denning models is defined, first, as a set of objects, such as files and users, that contain information; second, as active agents responsible for information flows; third, as security classes where each object and process are associated with a security class; fourth, a 'determination operator', which decides the security of an object that draws information from a pair of objects; fifth, a 'flow operator' that indicates if information will be allowed to flow from one security class to another.

Clearly the 'flow operator' is the critical part of the model since it determines if information will be allowed to flow from say a top secret file to an existing secret file. The 'flow operator' is also the major delimiting factor that prohibits the flow of information within the system. The definition of a secure information flow follows directly from these definitions. The flow model is secure if and only if a sequence of operations cannot give rise to an information flow that violates the flow operation. Together these properties are drawn into a universally bounded lattice. The first set of requirements of this lattice is that the flow operation is reflexive, transitive, and antisymmetric. The reflexive requirement is that information in a specified security class, be it Confidential{cases}, must be able to flow into other information containers within that same security class. The transitive rule requires that if information is allowed to travel from a file with security class Confidential{cases} to another information container file with security class Confidential{case_detail}, and that information is permitted to flow from Confidential{case_detail} to Confidential{case_summary}, then it must be permitted that information from file with clearance Confidential{cases} can flow directly to Confidential{case_summary}. Finally, the antisymmetric requirement is that if information can flow between two objects with different security classes, both from the first to the second, and from the second to the first, then we can set the security classes as equivalent.

The second set of requirements for this lattice is that there exists lower and upper bounds operations. That is, for all security classes, there should exist a security class such that information from an object with that security class would be permitted to flow to any other object. This requires that when information between two classes are merged, that it is possible to select the minimum of the security levels in order to allow the intersection of the information. For example, to use the lower bound operation on Confidential{cases, names} and TopSecret{names} to derive what the intersection of that information would result in, the result would be that the access to the output information would have the classification of Confidential{names}.

The upper bound requirement is already denoted as the flow operation. It is in line with the lower bound, with the exception that the maximum of the security levels and the union of the information is chosen. So, if information is merged together, the security level of the merged information assumes the highest previous form. Thus, for Confidential{cases} and TopSecret{results}, when information is merged between these two, the level of the object would be TopSecret{cases, results}.

As a result, the lattice allows for the Bell La Padula *no read up* and *no write down*, since an object with the highest class within a system can receive information from all other classes within the lattice, but can not send information to any other object, while the lowest security class can send information to any other security class in the lattice, but can not receive information from any of them. Together, the upper bound and lower bound provide the secure flow. If two objects with different classes, such as Confidential{case1, names} and Confidential{case2, names} are merged to create a new object, the resulting security level would have to be a more restrictive Confidential{case1, case2, names}. As for the lower bound, it restricts the flow of information downwards, so that objects with security class Confidential{cases, results} and Confidential{cases, names} can only receive an item classified no higher than Confidential{cases}.

Another result is that information can not flow between objects with incompatible security classes, which returns us to the restrictive nature of strict access control models. So, information within objects of class TopSecret{names} and TopSecret{results} can not be drawn together unless it is accessed by an object with a higher security level. This maintains the need-to-know nature of strict access controls, so that users and files are given the ability to collect information only for which domains they are designated.

THE REFERENCE MONITOR AND RUSHBY'S SOLUTION

To enforce the access control policy of the previously mentioned models, the TCSEC discusses the use of the reference monitor concept. The reasoning for this monitor is to the efficient and able enforcement of the policy, because there is a need for making sure that all interactions within the information system occur with some type of mediation that implements the policy at all times. This monitor must be accessed whenever the system is accessed, while it must be small and well identified in order for the system to be able to call on it whenever it is needed. To meet this need, three design requirements were specified by the TCSEC: the mechanism must be tamper proof, the reference validation mechanism must always be invoked, and the reference validation mechanism must be small enough to be subject to analysis, tests, and have an assurable completeness.

The emergence of the idea of a security kernel is rooted in this need. The security kernel is the small module in which all security features are located, and thus allows for intensive evaluation, testing, and formal verification. Rushby, however, argued against the use of a security kernel because in practice it ended up being inapplicable without the use of trusted processes, which must be permitted to break some of the rules normally imposed by the kernel. The reason for this is because of the restrictive and rigid natures of the security requirements demanded by the aforementioned models, because in the end, there are a number of classifications to deal with, and Rushby outlined a particular situation where they would fail: the print spool.

The print spool reads files and forwards them to the printer. If the printer spool was given the highest possible classification, it could read the user files, and then write them as spool files with the highest level of security classification. However, this requires that users be disallowed to access their own spool files, even to check the status of the printer queue. An option would be to allow spool files to retain the classification of the original user files, so that the spool could still *read down* the files. Then, the inability of the printer spool to *write down* would prevent the spool from even deleting the lower classification files after having been processed. The security kernel solution would be to declare the spooler a *trusted process*, which would allow it to contravene the *no write down* rule.

Rushby's model uses a method called the separation of users. That is, no user would be able to read or modify data or information belonging to another user. Meanwhile, users could still communicate using common files, provided by a file server, where the file server performs a single function of sharing files, as opposed to the complex operating system. This forms the basis of the Rushby Separation Model.

The reference monitor assumes that users access a common mass of information under the jurisdiction of a single unit. The separation approach assumes that it would be easier to offer users their own domains, thus simplifying all models and policies. However, users do need to cooperate and share data within an information system, and the file server's task is to provide this communication. The purpose of the separation kernel is to create an environment that suggests an appearance of separation amongst machines, and allows only for communication from one machine to the other through external communication lines, even though this physical distribution is not in fact in existence. In essence, this method simplifies the need for a reference monitor, and focuses on the need for logical separation while sharing resources, such as processor and communication lines. The end result is that the users' access to their own data needs no security control, thus effectively maintaining security without worrying about the restrictions of the above models, hence offering a little more flexibility.

AWAY FROM THE MILITARY

The aforementioned models concentrated solely on access controls for a very reasonable reason: the TCSEC, and even most of the range of security prior to the mid to late 1980s was concerned mostly with confidentiality. The TCSEC was quite overt with the importance of confidentiality, after all it was setting the standard for systems that were to be implemented within the Department of Defense. Within this institution, trust was predominant with its employees, and the organization thrived off it, while all that was required was the ability to trust the technology, the hardware and the software. Thus, the covert channels that the Bell La Padula model left unguarded were not of grave concern, because it was the trust of the application that was in doubt, not necessarily the users.

The rigidity of the models only seemingly complemented the rigidity of the organization, as structures within the military organization remain relatively constant, responsibilities and duties are often clear and compartmentalized much like the security classes, and where the philosophy of *need to know* reigned supreme as the status quo.

It is interesting to see the effectiveness of the TCSEC in what it set out to achieve, that is a set of standards for a military type organization. In that sense, it achieved its mission quite simply, yet the effect it had on the field of security is somewhat akin to the chicken and the egg. It was the culmination of years of research into the confidentiality of systems because security was more importantly deemed about keeping secrets secret. Meanwhile it also spawned a market acceptance of this type of solution to security, where the TCSEC are still considered the ultimate in evaluation criteria. If we are to speak in the language of formal methods and mathematics, this is where the problem arises: the models are wonderful models for the abstraction of the system they were abstracting, yet we have expected to apply the models to different systems with different abstractions. So, we see that the TCSEC are not meeting the requirements for what the average organization of today requires, while originally, the models and their abstractions never presumed that they could, however implicitly.

MILITARY AND NON-MILITARY: TOWARDS INTEGRITY

It is not being argued within this paper that confidentiality is key to information systems in all organizations. However, it is noticeable that non-military organizations secure their systems with another idea in mind: it costs them money if a system has incomplete or inaccurate data (Chalmers, 1986). The military organization is built on protecting information from an enemy, and thus the philosophy of *need-to-know* based on efforts to classify information and maintain strict segregation of people from information they are not allowed to see.

The TCSEC-type models were more interested in preventing unauthorized read access; businesses are far less concerned with who reads information than with who changes it. This demand for maintaining the integrity of information, and in general catering for the real needs of the non-military sector prompted research into other models and criteria. The TCSEC made clear that they were not well matched with the private sector, because of its lack of concern with the integrity of its information. Although issues in integrity were added in the Trusted Network Initiative, with the allusion to the Biba Model for Integrity, the fact remained the same: the criteria were not all that concerned with integrity. A system that qualifies for TCSEC scrutiny and has added functionalities for integrity checks would not receive any recognition for this because it is outside the scope of the TCSEC. Even worse, a system that is designed to support Integrity in other forms than the restrictive Biba Model, such as the Clark Wilson Model, may not even qualify for evaluation.

TOWARDS INTEGRITY: BIBA, C-W, AND CHINESE WALLS

Biba

The Biba Model is the Bell La Padula equivalent for integrity. Objects and subjects have hierarchical security classification related to their individual integrity, or trustworthiness. The integrity of each object and subject can be compared, so long as they follow two security properties:

1. If a subject can modify an object, then the integrity level of the subject must be higher than the integrity level of the object.
2. If a subject has read access to a particular object, then the subject can have write access to a second object only if the integrity level of the first object is greater than or equal to the integrity of second object.

The parallel is quite clear with the Bell La Padula model, particularly with its own two axioms. However, confidentiality and integrity seem to be the inverse of each other. In the Bell La Padula Model, the restriction was that a subject can not read information for which it is not cleared and a subject can not move information from an object with a higher security classification to an object with a lower classification. In comparison to the Biba Model it argues that if a subject can read information then the subject must have a higher security level than the object, and if the subject can move information from one object to another, then the latter object must have a lower security level than the first. Biba's model inverses the latter axiom, and demands that a high integrity file must not be corrupted with data from a low integrity file.

The parallels between the latter two axioms of the Bell La Padula and the Biba models are very much at the very heart of their systems. For Biba, this axiom is to prevent the flow of non-trusted information into a file with a high integrity classification, while for Bell La Padula, this second axiom tries to prevent the leaking of highly confidential information to a lower classified file. This property for Biba prevents a subject accessing a file from contaminating it with information of lower integrity than the file itself, thus preventing the corruption of a high integrity file with data created or derived from a less trustworthy one. The first axiom aims to prohibit the modification of a file with a high integrity classification, unless the subject has a higher integrity classification.

As is the case with the Bell La Padula model, the Biba model is difficult to implement. Its rigidity on the creation of information based on integrity classes, or levels of trust, although it seems novel and necessary, in practice this is too restrictive; thus very few systems have actually implemented the model. The model demands the classification of integrity sources and that strict rules apply to this - the integrity policy will have to be at the very heart of the organization; however integrity policies have not been studied as carefully as confidentiality policies, even though some sort of integrity policy governs the operation of every commercial data-processing system. To demand an integrity policy of this level within an organization is to demand that the organization has a clear vision on the trust mechanisms involved within its own organization - i.e. that the organization operates merely on the formal and technical level, without the ambiguity of the informal side of the organization.

THE CLARK-WILSON MODEL

The Clark-Wilson model is based on the assumption that bookkeeping in financial institutions is the most important integrity check. The model recognizes that the recording of data has an internal structure such that it accurately models the real world financial state of the organization. However, it is noted that the integrity of the integrity check is also a problem, since someone who is attempting a fraud could also create a false sense of financial integrity by altering the financial checks, by such methods as creation of false records of payments and receipts, for example. The solution would be to separate responsibilities as much as possible, disallowing the opportunity for a person to have as much authority over the integrity checks; if a person responsible for recording the receipts of goods is not authorized to make an entry regarding a payment, then the false entry on receipt of goods could not be balanced by the corresponding payment entry (Longley, 1991), thus leaving the books unbalanced, and the integrity check still valid. The Clark-Wilson Model attempts to implement this separation and integrity check into an information system, while drawing on the criteria provided by the US Department of Defense in their TCSEC.

There are two key concepts to the model: the Constrained Data Item (CDI) and the Transformation Procedure. The CDI is related to the balancing entries in account books, as in the above example. The TP is the set of legitimate processes that may be performed on the specified sets of CDIs, akin to the notion of double bookkeeping, or integrity checks.

To begin with, however, the model does impose a form of mandatory access control, but not as restrictive as the *no read up* and *no write down* criteria of the previously analyzed models: in non-military organizations, there is rarely a set of security classifications of users and data. In this case, the mandatory access control is concerned with the access of users to Transformation Procedures, and Transformation Procedures to Constrained Data Items. The CDIs may not be accessed arbitrarily for writing to other CDIs - this would result in a decreased integrity of the system. There are instead a set of requirements which the CDI can be processed in accordance to. As well, in order to enforce the sense of separation of duties, users may only invoke some Transformation Procedures, and a pre-specified set of data objects or CDIs, as their duties see fit.

The four requirements of this particular model are as follows:

1. the system must separately identify and authenticate every user
2. the system must ensure that specified data items can be manipulated only by a restricted set of programs, and the data center controls must ensure that these programs meet the *well formed transaction rule*, which have already been identified as Transformation Procedures
3. the system must associate with each user a valid set of programs to be run, and the data center must ensure that these sets meet the separation of duty rule
4. the system must maintain an auditing log that records every program executed, and the name of the authorizing user.

The four requirements noticeably relate heavily to the principles of security. The first alludes to maintaining authorized access, which falls under confidentiality; the second ensures the integrity of the data; the third discusses the need to clearly set out responsibilities; while the fourth alludes to the accountability of users and programs. In order to maintain and enforce system security, the system, in addition to the above requirements, must contain mechanisms to ensure that the system enforces its requirements at all times, and the mechanisms must be protected against unauthorized change; both requirements relate to the reference monitor concept from the TCSEC.

The security of the system, however, hinges on the state of the CDI's. Integrity rules are applied to data items in the system, and the outcome are the CDIs; the CDIs must meet the Integrity Validation Procedures (IVPs), and upon doing so, the system will be deemed secure.

The system has a point of origin where it must be in a secure state. This initial state is ensured by the Integrity Validation Procedures, which will validate the CDIs. From here, all changes to CDI's must be restricted to these well formed transactions, or Transformation Procedures, which evaluate and preserve the integrity of the information. When data is entered into the system, it is either identified as valid data, and thus granted a state of being secure and is validated as a CDI, or if the data does not meet the requirements of being a CDI, it is rejected and labeled an Unconstrained Data Item (UDI). A Transformation Procedure is then called upon to check the data item and transform it into a valid CDI, or reject the input. From this initial secure state, this Transformation Procedure that accepts/rejects the data is part of a set of Transformation Procedures, that when invoked fully, they will transfer the system from one secure state to another.

Yet what is key to this model, as opposed to the previous models with their rigid controls is that its certification is application-specific. The process by which Integrity Validation Procedures and Transformation Procedures ensure the integrity of the entire system will actually be defined differently for each person, based on the role they play within the organization, which is based on their duties, which necessarily enforces the notion of the separation of duties within the computer system. This level of adaptability, user orientation, and application subjectivity are what set this model apart from the rest; along with its accent on data integrity, rather than solely on confidentiality.

This level of subjectivity has a side effect, however. The integrity requirements are specified in terms of the actions based on Transformation Procedures on CDI's, which are only used in particular circumstances based on the user and the application, which in turn depends on the organization's procedures in order to develop the integrity of the data. Because of the bottom up nature of this approach, in addition to its subjectivity, it is not possible to create a single statement of security policy in the Clark Wilson Model, which the Bell La Padula model was centered on. The consequence of this dependence on applications and users for the level of controls that are to be used, commensurated by the fact that there is no single statement of security policy, the Clark Wilson model can not actually be evaluated to guarantee a given level of security, as the various criteria schemes demand.

At this point we face a little stand-off between effectiveness and criteria: can objective criteria truly assess the security of system based on differing roles, duties, and applications, which are then based on the organization? Although all the criteria consider organizations, they only consider organizations as being the creators of the policy, and the users, while the models enforce the policy, and it is expected that the users are merely trusted. This is a lot to be desired for, but also raises the issue that perhaps objective criteria are ineffective in gauging to what extent the system guarantees security. The Clark Wilson model provides a strong example of this. Under TCSEC it would not be recognized beyond its confidentiality functionalities. Meanwhile its integrity can not be gauged due to the fact that there is no guaranteed level of security since different mechanisms are called upon for different tasks. Finally, security in general is not well matched under the criteria since this model can not even provide a single statement of security, as the criteria dictate as required.

The organization is more than what sets the security policy - it is the environment that should dictate the entire information system. In organizing security in all types of organizations, military and non-military alike, the function of the organization must first be assessed and understood, and then the information system should be drawn from this. This process of deductive work should work for security as well, and that is what has been argued continuously in an implicit manner throughout this paper. Restrictive models often dictate the structure of the organization, and in this we see failures; for this reason the Biba model, despite how efficiently it would maintain integrity, organizations can not adapt to it, and thus it is not used. It would be possible for organizations to change their structure to cater for the rigid security classifications of the access controls models mentioned under the military organization, but this is not at all recommended because it would result in a loss of functionality and flexibility, traded off for control and security. The loss of functionality and flexibility would prove to be devastating to non-military organizations, particularly commercial organizations.

An example of a model created for a particular organization is the Bell La Padula model, and that is why it works well for the military organization, because it was developed with that structure and culture in mind, with the trust, classifications, and responsibilities. Another example is the Brewer-Nash Chinese Wall Security Policy.

DISCUSSION AND CONCLUSIONS

We have been presented with a variety of models, placed within the context of evaluation criteria, principles, and policies. It would be tempting to argue that one model is stronger than another because it better deals with integrity, while another is more valid because it solidly confronts confidentiality. This, would be a flaw, and this paper would have been a failure if the reader is considering such a venture.

If anything, this paper has outlined the beauty of the model: it is an abstraction of an abstraction. It is the abstraction of security measures and a policy. However, the second abstraction is easily forgotten: the measures and policy are in themselves abstractions of the requirements and specifications of the organization. Thus, the context of the abstraction is key, and this context is the environment - the organization, its culture, and its operations.

So, the Trusted Computer System Evaluation Criteria are valid and complete. The Bell La Padula and Denning Models for confidentiality of access controls are valid and complete. Rushby's Separation Model showed that the completeness of the reference monitor could be maintained without the inconsistency of the trusted processes. The Biba Model for integrity is valid and complete. The reasons for their validity, however, are not only because they are complete within their inner workings, their unambiguous language and derivations through axioms. Their completeness and validity are due to the fact that the abstraction that they represent, the world that they are modeling and the organization for which they are ensuring the security policy, are all well defined: the military organization. This military organization comes with a culture of trust in its members, a system of clear roles and responsibilities, while the classification of the information security levels within the models are not constructs of the models, but instead reflect the very organization they are modeling. Meanwhile, integrity was never really much of a concern for the US Department of Defense.

This is where the line is drawn between the military and the non-military organization. In the non-military organization, integrity of the information is key to the well being of the organization. Particularly in the commercial world, what is key to the well being of the organization is key to its very survival, so integrity can not be taken lightly. However the TCSEC and the aforementioned models do not reflect this need within this new context: where trust should not be assumed, where information flows freely with a notion of needing-to-withhold rather than knowing, where roles and responsibilities are not static, and where information carries no classification without its meaning. The Clark-Wilson model reflected on how integrity was key to the non-military organization, and the consequence of this was that it showed how the TCSEC could not cater for its strengths. Subsequent criteria took on the role of developing non-military criteria, which was where the TCSEC stopped, after all the TCSEC was only ever a standard for developing systems for the US military complex. Yet even the Clark-Wilson model showed that to attempt to scrutinize systems objectively in general is a problematic task, particularly since the model did not even have a single security policy. This is attributed to the fact that it is heavily decentralized in nature, while criteria can not be expected to analyze this formally on a wide scale. After all, the model should reflect the organization, and the organization is not generic, while the model may be. This demands further analysis and models, such as the Brewer-Nash Chinese Wall Security Policy, which derives a model for consultancy based organizations. While this model is not as interesting in its inner workings, it is an step in the right direction, towards a model that is based on its organization, instead of requiring that the organization base itself on a model.

It seems we have come full circle, with a little bit of confusion occurring in the mid 1980s through to the 1990s. The TCSEC were released, the Bell La Padula and Denning type access controls made standard within these criteria, because the criteria and models were based on a specific type of organization. Yet somewhere upon this occurring, the message was lost. The field of computer security began believing that the TCSEC were the ingredients to a truly secure computer system for all organizations, and thus systems should be modeled on its criteria. Debates have gone on about the appropriateness of the TCSEC for the commercial organization, while this debate should never have happened, because the TCSEC were never meant for non-military organizations. So the ITSEC arrived, along with the CTCPEC and FC-ITS, and started considering more than what was essential for the military organization. Integrity became key, organizational policies gained further importance. The need for considering the organization had finally returned to the limelight. The organization should drive the model, which is enabled by the technology. This is the basic criteria for a security policy. The model should never drive the organization, because this is a failure of the abstraction.

As the non-military organizations learn this large yet simple lesson, much work is still required. Models are powerful and necessary, but a solid analysis of the environment is also necessary: the culture and the operations need to be understood before the policy is made, and the awareness needs to be promulgated, and hopefully the trust will arise out of the process. In the meantime, progress is required in research into the integrity of the information within the system, after all this is the lifeblood of the organization.

REFERENCES

Chalmers, L. S. (1986) An analysis of the differences between the computer security practices in the military and private sectors, *Proceedings of the Symposium on Security and Privacy.*

Dhillon, G. (ed.) (2001) *Information security management: global challenges in the new millennium*, Idea Group Publishing, Hershey.

Dhillon, G., and Backhouse, J. (2000) Information system security management in the new millennium, *Communications of the ACM,* 43, 7, 125-128.

Gasser, M. (1988) *Building a secure computer system*, Van Nostrand Reinhold.

Longley, D. (1991) Security of stored data and programs, in W. Caelli, *et al.* (eds.), *Information security handbook,* Macmillan, UK, Basingstoke, 545-648.

Wordsworth, J. B. (1999) Getting the best from formal methods, *Information and Software Technology,* 41, 1027-1032.

Analysis of Gait Flow Image and Gait Gaussian Image Using Extension Neural Network for Gait Recognition
Parul Arora, Smriti Srivastavaand Shivank Singhal (2016). *International Journal of Rough Sets and Data Analysis (pp. 45-64).*
www.irma-international.org/article/analysis-of-gait-flow-image-and-gait-gaussian-image-using-extension-neural-network-for-gait-recognition/150464