Multilevel Security and Quality of Protection

Simon N. Foley[1], Stefano Bistarelli[3,4], Barry O'Sullivan[1,2], John Herbert[1], and Garret Swart[5]

[1] Department of Computer Science, University College, Cork, Ireland.
[2] Cork Constraint Computation Centre, University College Cork, Ireland
[3] Dipartimento di Scienze, Università "G. D'Annunzio" di Chieti-Pescara, Italy
[4] Istituto di Informatica e Telematica, CNR, Pisa, Italy
[5] IBM Almaden Research Center, San Jose, CA, USA

Abstract. Constraining how information may flow within a system is at the heart of many protection mechanisms and many security policies have direct interpretations in terms of information flow and multilevel security style controls. However, while conceptually simple, multilevel security controls have been difficult to achieve in practice. In this paper we explore how the traditional assurance measures that are used in the network multilevel security model can be re-interpreted and generalised to provide the basis of a framework for reasoning about the quality of protection provided by a secure system configuration.

1 Introduction

Multilevel security is concerned with controlling the flow of information in systems. The traditional view of multilevel security is one of ensuring that information at a high security classification cannot flow down to a lower security classification [2, 9, 32]. However, constraining how information may flow within a system is at the heart of many protection mechanisms and many security policies have direct interpretations in terms of multilevel security style controls. These include: Chinese Walls [13, 25]; separation of duties and well formed transactions [13, 14, 18]; Role Based Access Control [26] and a variety of policies where a degree of data separation is required, for instance, Digital Rights Management [24] and Multi-applicative Smart Cards [28].

Multilevel security, while conceptually simple, has been notoriously difficult to achieve in practice [27]. From the earliest models, there have been problems in reconciling multilevel security models with actual multilevel secure systems, leading to problems such as covert channels [22] and how to properly interpret the model [20]. This led to more abstract formal definitions such as [11, 16, 30] that effectively attempted to capture the meaning of information flow in some possibilistic information-theoretic sense. These properties of non-interference, information flow and a great many variations have been extensively studied. Designing and verifying security mechanisms that uphold these classes of property is accepted to be difficult [21, 29].

Using formal methods to analyse and verify information flow properties of secure systems requires considerable specification effort. The cost of such in-depth

specification and subsequent analysis may be justified for small critical security mechanisms such as authentication protocols and security kernels. However, such in-depth security analysis would not scale to the configuration of a large and/or complex application system.

We are interested in developing shallow and pragmatic security analysis methods for systems. This is achieved through the analysis of how a system is *configured*, rather than an analysis of its underlying mechanisms and protocols. Instead of concentrating on detailed semantics and complete formal verification of components, we are concerned more with the ability to trace, at a practical level of abstraction, how component security requirements relate to each other and any overall security requirements. We believe that a complete security verification of a system is not achievable in practice; we seek some degree of useful feedback from an analysis that a particular system configuration is reasonable.

We adopt this view when re-visiting the problem of multilevel security. Rather than seeking 'beyond A1' multilevel security [21, 27, 32], we seek to measure the degree and/or quality of the multilevel protection that is provided by a system configuration. Weaker, but reasoned, assurances of security are a pragmatic way of providing practical multilevel systems, such as [19], that can be built from Commercial Off-The-Shelf (COTS) components. Systems may be configured from components in which we have varying degrees of confidence in their security. In [15], confidence rated information flow policies are used to model interoperation between PDAs and Workstations: we have a higher degree of confidence in the flows that are constrained by the workstation security mechanism than we have in flows constrained by the PDA application. In [1] we consider how best to configure Storage Area Networks from components having varying security guarantees, while ensuring that mandatory security rules are enforced. These approaches do not consider covert-channels or in-depth formal analysis of protection mechanisms. Rather, they seek useful feedback that a particular configuration is reasonable.

In this paper we describe a general framework for measuring quality of protection for information flow and/or multilevel security. The model builds on earlier models of information flow security [12–15] by considering the relative risks of configuring various components into multilevel systems. Risk measurement is used to characterise the quality of protection that is provided by a multilevel system configuration. This gives rise to a novel approach to describing multilevel security policies that combine both risk and information flow. The model that is developed in this paper is a consistent interpretation of multilevel security, allowing us to draw on a wide range of existing results from the area.

The model that is proposed in this paper forms a part of our ongoing research in using constraint solving techniques as a practical approach for reasoning about security [1, 3–5, 31]. Building on the results in [4] we demonstrate in this paper that determining whether a particular system configuration meets a quality of protection measure can be described as a constraint satisfaction problem. Constraint solving is an emerging software technology for modelling and solving

large-scale optimisation problems [3, 34] and there are many results on solving this problem for large systems of constraints in a fully mechanised manner.

Section 2 describes the underlying model of multilevel security. As with past security criteria, this model is extended in Section 3 to support assurance levels. However, our interpretation of assurance is more general: every system component has an assurance level that reflects the degree of confidence that it cannot be compromised. In Section 4 we illustrate how configurations within our model can exhibit cascade vulnerabilities [23, 32] and outline in Section 5 a soft constraint based framework [4] that can be used in their detection and elimination. The advantage of taking a soft-constraint approach is that assurance can be described in terms of a c-semiring [3] and Section 6 explores how aggregate risk measurements can be made across configurations. Section 7 provides further discussion on how this framework provides a basis for quality of protection.

2 Interpreting Multilevel Security

An information flow policy is defined in terms of a lattice ordering $(\_ \leq \_)$ over a set of security labels $\mathcal{L}$. Given $x, y : \mathcal{L}$ then $x \leq y$ means that information may flow from level x to level y. The simplest interpretation of an information flow policy is multilevel security [2] whereby the labels correspond to sensitivity levels, for example, $\mathsf{unclass} \leq \mathsf{secret} \leq \mathsf{topsecret}$. A more general interpretation [14] is that a label represents an abstract data type that is used to encode security relevant characteristics of entities that are subject to flow constraints. With this interpretation a wide variety of access control policies can be represented within the multilevel security model. Techniques for specifying more general (non-lattice) information flow constraints and translating them into lattice based policies are considered in [12–14].

Let the set of entities $\mathcal{E}$ represent the set of all components that can source and/or sink information. In addition to the conventional 'subject' and 'object' interpretation, entities are regarded as anything that can store, process and/or manage information [12, 15]. Examples include devices, workstations, controllers, sessions, datasets and applications (examples can be found in [1, 14, 15]). An entity is anything that can have an associated security state (and to which the flow constraints must apply).

Every entity, e, is bound to an interval of the policy lattice, where $int(e) = [x, y] \in \mathcal{L} \times \mathcal{L}$, and $x \leq y$, is interpreted to mean that entity e may sink information at class y or lower and may source information at class x or higher [12]. We also write $int(e) = [int_\perp(e), int_\top(e)]$. If entity e is a 'subject' then $int(e) = [x, y]$ corresponds to a partially trusted subject (in the sense of [8]) that may view/read information at class y and lower and may write/alter information at class x and higher; these are defined as $vmax$ and $amin$, respectively, in [8]. Conventional objects may be interpreted within this model as entities that are bound to a point interval $[x, x]$ with a single level. Intuitively, we interpret [12], $int(e) = [x, y]$ to mean that the entity can be trusted to properly manage multilevel information within the security interval $[x, y]$.

Let $A \rightsquigarrow B$ represent information flow in our system from entity A to entity B. We do not consider a semantics for $\rightsquigarrow$; it could be simply based on read-write access controls (effectively [2, 8]), based on a non-interference interpretation, or even based on some informal characterisation of what flows are considered to be possible [15] in a system. Under this interpretation, a system is secure if for all entities, A, B, such that $A \rightsquigarrow B$ then $int_\bot(A) \leq int_\top(B)$ holds [12]. In this paper we use a variant of this definition to reflect the *specific* information that can flow. Let $A_x \rightsquigarrow B_y$ represent a flow of x information in entity A to y information in entity B. A system is secure, if for all entities A and B then,

$$A_x \rightsquigarrow B_y \Rightarrow x \leq y \wedge int_\bot(A) \leq x \leq int_\top(A) \wedge int_\bot(B) \leq y \leq int_\top(B)$$

Example 1 A multilevel secure network is composed of systems A and B. System A is a multilevel secure and configured to manage $\mathsf{unclass}$ and secret information and is thus partially trusted with $int(A) = [\mathsf{u}, \mathsf{s}]$. Similarly, system B is trusted to manage secret and topsec information, and $int(B) = [\mathsf{s}, \mathsf{t}]$. The systems communicate/share secret information. The flows are defined as $A_\mathsf{s} \rightsquigarrow B_\mathsf{s}$ and $B_\mathsf{s} \rightsquigarrow A_\mathsf{s}$, and by definition, the configuration is secure. Note that we may use the initial character(s) of a security level to represent it, if no ambiguity can arise. In general, a flow between entities need not necessarily be sourced and sunk at the same level. For example, the flow $F_\mathsf{s} \rightsquigarrow P_\mathsf{t}$ might represent a secret file F that is read by a single level process P with $int(P) = [\mathsf{t}, \mathsf{t}]$. △

3 Interpreting Assurance

Define a lattice, $\mathcal{A}$, of assurance levels with ordering $\leq$. Given $x, y : \mathcal{A}$, then $x \leq y$ means that a system evaluated at y is no less secure than a system evaluated at x, or alternatively, that an attacker that can compromise a system evaluated at y can compromise a system evaluated at x. For example, the 'Orange' and 'Red' Book security criteria [32, 33] define assurance levels $\mathsf{A1} > \mathsf{B3} > \mathsf{B2} > \mathsf{B1} > \ldots$. This conventional notion of assurance can be generalised to assurance for entities [15] if we regard assurance as reflecting our degree of confidence that an entity can be relied upon to properly manage the information that is entrusted to it. For example, we might have high confidence in a firewall-based email proxy (entity) managing multilevel information, but have low confidence in a `sendmail` process (entity) managing the same information.

We define $rating : \mathcal{E} \rightarrow \mathcal{A}$ where $rating(e)$ gives the assurance rating of entity e, and is also taken to represent the minimum effort that is required by an attacker to compromise entity e.

Security evaluation criteria [32] also define a minimum required assurance function $req : \mathcal{L} \times \mathcal{L} \rightarrow \mathcal{A}$, such that $req(l, l')$ defines the minimum required assurance for a system managing information at classes $l, l' : \mathcal{L}$. For example, $req(\mathsf{unclass}, \mathsf{topsec}) = \mathsf{B3}$ means that in order for an entity to manage information with labels between unclass and topsec, a $\mathsf{B3}$ assurance rating is needed. In general a system must meet the minimum required assurance.

$$\forall e \in \mathcal{E} : req(int_\bot(e), int_\top(e)) \leq rating(e)$$

This has a similar interpretation for the more general notion of an entity used in this paper. Entities represent anything that can source and/or sink information. For example, the rating of an entity may incorporate the methodology that was used to develop the entity, as in the conventional Orange/Red Book rating, the level of testing the entity has received, or the level of complexity of the function the entity is implementing. A general purpose workstation, W, may be just fine for managing single level of information, like [secret, secret] but be unacceptable for managing multilevel data such as [secret, topsec]. In the model this is represented by setting $rating(W)$ to FAIR, and setting $req(\mathsf{secret}, \mathsf{secret})$ to FAIR. But $req(\mathsf{secret}, \mathsf{topsec})$ must be set to a higher assurance rating, say GOOD, to ensure that W and other workstations like it are not used for such information. Another example, $rating(A)$ could represent how much we can rely on the user A (given their associated security interval); for example, one would presume that a CEO has a higher assurance rating than a clerk in the same organisation.

A further example, $rating(S)$ could represent the rating of application software S: a COTS product may have a low rating, while an in-house developed application may have a high rating, when handling multilevel information. While it may be acceptable to trust the high assurance email proxy process with multilevel information (for example, $int(\text{proxy}) = [\mathsf{u}, \mathsf{t}]$), it may only be acceptable to trust `sendmail` with single-level information (for example, $int(\text{sendmail}) = [\mathsf{s}, \mathsf{s}]$). This could be reflected by requirement $req(\mathsf{u}, \mathsf{t}) = hi$ and $req(\mathsf{s}, \mathsf{s}) = lo$, where $lo < li$, and so forth.

Example 2 In the Chinese Wall policy a stock market analyst may not advise an organisation if he has insider knowledge of another competing organisation. Encoding this policy in terms of a multilevel security policy has been demonstrated elsewhere [13, 14, 25]. In this example we describe a new multilevel encoding of the Chinese Wall policy in terms of an assurance requirement.

Let $\mathcal{L} = 2^{\{\mathsf{ibm},\mathsf{hp},\mathsf{sun},\mathsf{elf},\mathsf{shell},\ldots\}}$ be the powerset of organisations. Define the assurance lattice as: audit $<$ cons $<$ over, where audit represents the degree of trust in an auditor, cons represents the degree of trust in a consultant, and over represents the degree of trust in a stock exchange partner who is trusted to access everything for the purposes of oversight. Consultants are trusted to consult for multiple organisations so long as there is no conflict of interest. We define some minimum required assurance levels for intervals of trust as follows.

$$
\begin{array}{l|l|l}
req(\{\}, \{\mathsf{hp}\}) = \mathsf{aud} & req(\{\}, \{\mathsf{ibm}, \mathsf{elf}\}) = \mathsf{cons} & req(\{\}, \{\mathsf{ibm}, \mathsf{hp}\}) = \mathsf{over} \\
req(\{\}, \{\mathsf{elf}\}) = \mathsf{aud} & req(\{\}, \{\mathsf{hp}, \mathsf{shell}\}) = \mathsf{cons} & req(\{\}, \{\mathsf{ibm}, \mathsf{hp}, \mathsf{elf}\}) = \mathsf{over} \\
req(\{\}, \{\mathsf{ibm}\}) = \mathsf{aud} & req(\{\}, \{\mathsf{hp}, \mathsf{elf}\}) = \mathsf{cons} & req(\{\}, \{\mathsf{ibm}, \mathsf{hp}\}) = \mathsf{over}
\end{array}
$$

Assume that any entity that is controlled by a consultant will never have an assurance rating higher than cons. While a consultant may be trusted to simultaneously manage ibm and elf information (bound to interval $[\{\}, \{\mathsf{ibm}, \mathsf{elf}\}]$), the minimum assurance rule dictates that a consultant cannot be trusted to access conflicting ibm and hp data (bound to interval $[\{\}, \{\mathsf{ibm}, \mathsf{hp}\}]$). △

Note that we assume that the execution system will properly classify entities. For example, a session entity corresponding to a consultant executing low assurance software would have an assurance level equal to the greatest lower bound of the consultant assurance and the software assurance level. Similar calculations are necessary to determine the interval for the session (the greatest lower bound of the intervals of the entities involved). For reasons of space we do not consider the execution model in this paper, however models such as [14] are applicable in this case.

4 The Cascade Problem

The *cascade vulnerability problem* [23, 32] is concerned with secure interoperation, and considers the *assurance risk* of composing multilevel secure systems that are evaluated to different levels of assurance according to the criteria specified in [32]. The transitivity of the multilevel security policy upheld across all secure systems ensures that their multilevel composition is secure; however, interoperability and data sharing between systems may increase the risk of compromise beyond that accepted by the assurance level. For example, it may be an acceptable risk to store only secret and top-secret data on a medium assurance system, and only classified and secret data on another medium assurance system; classified and top-secret data may be stored simultaneously only on 'high' assurance systems. However, if these medium assurance systems interoperate at classification secret, then the acceptable risk of compromise is no longer adequate as there is an unacceptable cascading risk from top-secret across the network to classified.

Example 3 Continuing the Chinese Wall example, consider two consultant sessions (entities) A and B, that are trusted to the following extent.

$$\begin{array}{ll} rating(A) = \mathsf{cons} & int(A) = [\{\}, \{\mathsf{ibm}, \mathsf{elf}\}] \\ rating(B) = \mathsf{cons} & int(B) = [\{\}, \{\mathsf{hp}, \mathsf{elf}\}] \end{array}$$

Suppose that the system permits these sessions to share information classified at $\{\mathsf{elf}\}$, that is, we have $A_{\{\mathsf{elf}\}} \rightsquigarrow B_{\{\mathsf{elf}\}}$ and $B_{\{\mathsf{elf}\}} \rightsquigarrow A_{\{\mathsf{elf}\}}$. While the individual entities are secure based on the *req* assurance rule defined above, their interoperation is not. There is a cascading path from $\{\mathsf{ibm}\}$ on entity A to $\{\mathsf{hp}\}$ on entity B via shared channel $\{\mathsf{elf}\}$. The assurance rules require an assurance level of at least over in order to be able to simultaneously access both $\{\mathsf{hp}\}$ and $\{\mathsf{ibm}\}$ information. However, with a configuration that allows A and B share elf information, entities with an assurance rating of just cons can obtain this access.

This can be interpreted in two ways. The assurance level reflects how much we can rely on an entity to properly manage the different information. The configuration implies that we have cons level confidence that hp and ibm information is properly managed, which is contrary to the requirement. The second interpretation is when one regards assurance as representing the degree of confidence that one can have that an entity cannot be compromised. In this case the effort

required by an attacker corresponds to the effort to compromise cons rated systems to effectively copy hp into ibm data. However, the requirement is that it must require at least the effort to compromise a level over rated entity. △

The above example illustrates that avoiding conflict of interest when entities share information corresponds to detecting and eliminating the cascade vulnerability problem. Existing research has considered schemes for detecting these cascading security vulnerabilities and for eliminating them by reconfiguring system interoperation. While the detection of cascade vulnerabilities can be easily achieved [23, 32], their *optimal* elimination is NP-complete [17].

5 Soft Constraints and Semirings

In [4], a soft constraint-based framework is described for modelling, detecting and eliminating the cascade vulnerability problem. A soft constraint may be seen as a constraint where each instantiation of its variables has an associated value from a partially ordered set that can be interpreted as a set of preference values. Combining constraints will then have to take into account such additional values, and thus the formalism has also to provide suitable operations for combination ($\times$) and comparison ($+$) of tuples of values and constraints. This is why this formalisation is based on the concept of c-semiring, which is just a set plus two operations.

The framework described in [4] is directly applicable to the information flow model described in this paper. A network (a system of entities) is modelled in terms of constraints, reflecting all possible flows as a result of the network configuration (the $\rightsquigarrow$ relation). This constraint network also considers the effective assurance along all possible communication paths in the network. The network is cascade free if these constraints uphold the overall assurance criteria (the *req* relation).

The security label ordering $(\mathcal{L}, \leq)$ is modelled as a lattice and the assurance ordering $(\mathcal{A}, \leq)$ in [4] is modelled as a more general c-semiring structure [3, 6]. While [4] only considered the cascade problem for conventional lattice based assurance ordering, the framework is applicable for any c-semiring. A semiring is a tuple $\langle \mathcal{S}, +, \times, \mathbf{0}, \mathbf{1} \rangle$ such that: $\mathcal{S}$ is a set and $\mathbf{0}, \mathbf{1} \in \mathcal{S}$; $+$ is commutative, associative and $\mathbf{0}$ is its unit element; $\times$ is associative, distributes over $+$, $\mathbf{1}$ is its unit element and $\mathbf{0}$ is its absorbing element. A c-semiring is a semiring $\langle \mathcal{S}, +, \times, \mathbf{0}, \mathbf{1} \rangle$ such that: $+$ is idempotent, $\mathbf{1}$ is its absorbing element and $\times$ is commutative.

Let us consider the relation $\leq_S$ over $\mathcal{S}$ such that $a \leq_S b$ iff $a + b = b$. Then it is possible to prove that (see [6]): $\leq_S$ is a partial order; $+$ and $\times$ are monotone on $\leq_S$; $\mathbf{0}$ is its minimum and $\mathbf{1}$ its maximum; Informally, the relation $\leq_S$ gives us a way to compare semiring values and constraints. In fact, when we have $a \leq_S b$, we will say that b *is better than* a. In the following, when the semiring will be clear from the context, $a \leq_S b$ will be often indicated by $a \leq b$.

The classical Constraint Satisfaction Problem (CSP) is a Soft CSP (SCSP) where the chosen c-semiring is: $S_{CSP} = \langle \{false, true\}, \vee, \wedge, false, true \rangle$. Fuzzy

CSPs (FCSP) can instead be modelled in the SCSP framework by choosing the c-semiring $S_{FCSP} = \langle [0,1], max, min, 0, 1\rangle$. Many other soft CSPs (probabilistic, weighted, ...) can be modelled by using a suitable semiring structure ($S_{prob} = \langle [0,1], max, \times, 0, 1\rangle$, $S_{weight} = \langle \mathcal{R}, min, +, +\infty, 0\rangle$, ...). Therefore, a wide range of 'soft' ways to consider degree of assurance can be considered and can be effectively reasoned about within our model.

6 Interpreting Risk

While conventional assurance ratings are defined in terms of a lattice, the model proposed in this paper can use any *measure* that can be defined as a c-semiring. The assurance rating, $rating(A)$ of an entity A provides a measure of how much the entity can be relied upon not to be compromised. Whether we use numbers, enumerations (A1, B3, ...), and so forth, in the c-semiring is not important; rather it is the ability to compare different rating which give a measure of how much we can rely on an entity.

We can interpret an assurance rating as providing an indication of the minimum amount of effort that is required by an attacker to compromise an entity. By 'compromise' we mean that the attacker can force the entity to violate its interval of trust, that is, to violate the information flow ordering. For example, compromising the system B in Example 1 corresponds to the attacker breaking the protection mechanism, and outputting topsec labelled as secret (copying topsec information to secret). This corresponds to the usual threat model used in [32]. In this paper we generalise this to any entity. An attacker could compromise another user by tricking them into revealing incorrectly labelled information; an attacker could compromise an application by a stack smashing attack, causing it to copy information from one file to another, violating the flow policy.

The c-semiring provides a convenient way to measure aggregate threats across the collections of entities that make up a system. The minimum effort required to break a series of entities along a path is given by the combination (under the c-semiring) of the ratings of the individual broken entities. In this case, the weighted c-semiring $\mathcal{S}_{weight} = \langle \mathcal{R}, min, +, +\infty, 0\rangle$ provides the appropriate measure. A path that can cause a flow of level x information to level y can start at any system that is trusted to manage x information and can end at any system trusted to manager y information. Given a series of possible compromising paths that facilitate a flow from level x to level y, where $x \not\leq y$, then the least effort required to create a compromise from x to y is the shortest path (using $\mathcal{S}_{weight}$) from x to y. There is a cascade vulnerability if the value calculated for this shortest path is more than $req(x, y)$.

This is effectively a characterisation of a cascading path from [32], but defined in terms of a c-semiring using the model [4]. It is the definition in terms of a c-semiring that allows the determination of effort along a cascading path as the combination of the efforts required to break individual systems along the path. Practical techniques for calculating shortest paths across weighted constraint networks are considered in [3].

Definition 1 Quality of Protection. Let constraint specification $\mathcal{CONFIG}$ represent the flows (and cascades) that are a consequence of entity interoperation constructed using the model [4]. The assurance requirements function $req(x, y)$ provides an acceptable lower bound on the quality of protection for this system configuration. Let constraint specification $\mathcal{QOP}$ represent these ratings for all permitted flows. A configuration $\mathcal{CONFIG}$ meets the quality of protection requirement $\mathcal{QOP}$ if no path in $\mathcal{CONFIG}$ violates $\mathcal{QOP}$. △

The assurance requirements function $req(x, y)$ provides an acceptable lower bound on the quality of protection for an overall system configuration. The soft-constraint model described in [4] can be used to encode the quality of protection problem as a constraint satisfaction problem. For reasons of space we do not provide the details of the constraint model.

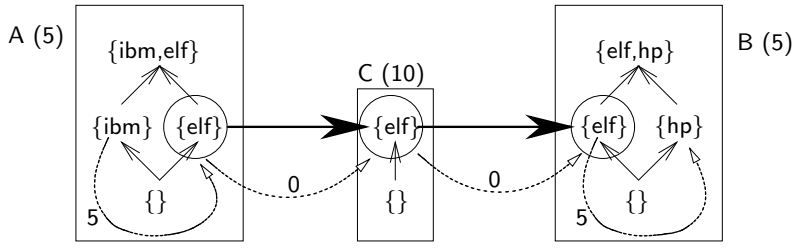

Fig. 1. Entity Information Flows with Weighted Cascading Path

Example 4 Figure 1 depicts a configuration of the consultant sessions A and B from Example 3. We introduce a third session entity C, that connects entity A and entity B, permitting controlled sharing of $\{\mathsf{elf}\}$ information. This is represented by the flows $A_{\{\mathsf{elf}\}} \rightsquigarrow C_{\{\mathsf{elf}\}}$ and $C_{\{\mathsf{elf}\}} \rightsquigarrow B_{\{\mathsf{elf}\}}$. Using the weighted c-semiring to define assurance ratings, the sessions are trusted to the following extent.

$$\begin{array}{ll} rating(A) = 5 & int(A) = [\{\}, \{\mathsf{ibm}, \mathsf{elf}\}] \\ rating(B) = 5 & int(B) = [\{\}, \{\mathsf{hp}, \mathsf{elf}\}] \\ rating(C) = 10 & int(C) = [\{\}, \{\mathsf{elf}\}] \end{array}$$

and some defined minimum required ratings are as follows.

$$\begin{array}{lll|lll|lll} req(\{\}, \{\mathsf{hp}\}) & = 3 & & req(\{\}, \{\mathsf{ibm}, \mathsf{elf}\}) & = 5 & & req(\{\}, \{\mathsf{ibm}, \mathsf{hp}\}) & = 15 \\ req(\{\}, \{\mathsf{elf}\}) & = 3 & & req(\{\}, \{\mathsf{hp}, \mathsf{shell}\}) & = 5 & & req(\{\}, \{\mathsf{ibm}, \mathsf{hp}, \mathsf{elf}\}) & = 18 \\ req(\{\}, \{\mathsf{ibm}\}) & = 3 & & req(\{\}, \{\mathsf{hp}, \mathsf{elf}\}) & = 5 & & req(\{\}, \{\mathsf{ibm}, \mathsf{hp}\}) & = 15 \end{array}$$

This configuration has a cascading path vulnerability. The effort required to break entity A and copy $\{\mathsf{ibm}\}$ information to $\{\mathsf{elf}\}$, copy this to entity C, and copy it again to a broken entity B which allows it to be copied to $\{\mathsf{hp}\}$ is 5+5=10, which is less than the minimum effort required, that is, $req(\{\}, \{\mathsf{ibm}, \mathsf{hp}\}) = 15$.

Note that it is not necessary to break entity C as the attacker inputs and outputs $\{\mathsf{elf}\}$ information, and thus, the effort to carry out this copy is 0. This is dealt within the cascade framework [4] by defining permitted flows as having minimum rating, the lowest level in the lattice. In the case of $\mathcal{S}_{weight}$ this is the value 0, that is, $req(\{\mathsf{elf}, \mathsf{elf}\}) = 0$. The dashed arcs in Figure 1 represents the weighted cascading path calculated from $\{\mathsf{ibm}\}$ to $\{\mathsf{hp}\}$. △

In suggesting the use of the weighted c-semiring $\mathcal{S}_{weight} = \langle \mathcal{R}, min, +, +\infty, 0 \rangle$ as one example of a risk measure, we are assuming that the effort required by an attacker to compromise one entity is independent of the effort to compromise any other entity. This means that having expended 'effort' $rating(A)$ to compromise system A, an attacker must, in addition, expend 'effort' $rating(B)$ to subsequently compromise system B, regardless of whether lessons learnt in compromising A can be subsequently used to attack B. This is quite a restrictive assumption; however, there are examples where this kind of measure is useful. For example, in practice, the more firewalls/subnets that have to be traversed to directly access a system, then the more 'secure' the system is considered to be. The notion of *security distance* is defined in [31] as the minumum number of servers/firewalls that an attacker on the Internet must compromise to obtain direct access to some protected service. We conjecture that security distance in this case is equivalent to using a weighted c-semiring with each system having an equal rating of '1'.

An alternative measure to using the weighted c-semiring is to interpret the probabilistic c-semiring $\mathcal{S}_{prob} = \langle \{x | \in [0, 1]\}, max, \times, 0, 1 \rangle$ [3] in terms of aggregation of risk along a path, which is calculated as combination (multiplication) of probabilities. As an attacker compromises systems along a cascading path, then overall, less and less 'effort' is required to attack subsequent systems.

These measures are unlike the lattice based assurance measure used by the Orange/Red Book. This reflects an assumption that once one system rated at degree x (for example, B2) is compromised then all systems rated at this degree or lower (for example, B2, B1,...) are considered compromised. In practice, we believe that a practical risk measure will use a variety of such measures; exploring suitable c-semirings is a topic for future research.

7 Discussion and Conclusion

In this paper we describe how the network multilevel security model can be generalised to provide an approach to measuring the degree of confidence that one can have in the security of a system configuration. A system configuration is modelled as a collection of entities. These entities can represent system components, users, COTS components, and so forth, whose potential accesses and interoperation are articulated abstractly in terms of information flows. It is not necessary for these components to have an *explicit* access control mechanism; the flow relations represents the access limitations that we believe the entities effectively uphold. Thus, in the sense of [7], every entity in the system can be re-

garded as contributing to the overall Trusted Computing Base. In our framework we can distinguish the merit of each entity's contribution.

While the results in this paper are presented in terms of a multilevel security model, we argue that they have wider application. Constraining how information may flow within a system is at the heart of many protection mechanisms. Many security policies have direct interpretations in terms of multilevel security style controls. Furthermore, modelling a configuration in terms of information flows provides a form of traceability on interoperation that can provide useful feedback on the quality of protection achieved.

A multilevel security model for Storage Area Networks (SANs) is proposed in [1]. This SAN model also takes a measurement approach to achieving security. Hard (crisp) constraints are used to measure the risk associated with SAN configurations. However, the SAN model uses an ad-hoc adaptation of multilevel security, and does not have the same strict interpretation within the network security model as does the model proposed in this paper. As a consequence, the SAN model does not address the cascading channel problem. We are currently investigating how the risk framework in [1] can be re-coded in terms of the c-semiring based framework proposed in this paper. The advantage of this is a simplification of the SAN model that solves the channel cascade problem, and provides access to a greater range of measures (c-semirings) for risk. A soft-constraint encoding of the revised SAN model will also provide access to techniques for exploring and manipulating SAN configurations. This will be especially useful when making tradeoffs of quality of protection against other attributes such as cost and performance [1, 31]. Exploring how our soft-constraint framework can facilitate making such tradeoffs is a topic for future research.

Determining whether a system configuration provides quality of protection as required by $req(x, y)$ is easily achieved as it corresponds to the channel cascade *detection* problem. Any solution to the constraint model represents a cascading path, which provides significantly more information regarding the vulnerabilities in the network than existing approaches for detecting cascading paths [17, 23]. The set of solutions to the constraint model provides a basis for removing the cascade vulnerability problem.

Reconfiguring such a system by attempting to eliminate an optimal minimum number of links (flows) between entities is NP-complete as it corresponds to the cascade *elimination* problem [17]. Previous approaches [10, 17, 23] detect a single cascading path in polynomial time, but eliminating the cascade in an optimal way is NP-complete. Detecting all paths in the constraint model is NP-hard, however elimination of a minimal number of links is polynomial. While constraint solving is NP-complete in general, this has not detracted from its uptake as a practical approach to solving many real-world problems [34]. Using a constraint model, we can rely on a significant body of successful techniques for finding the set of cascading paths, which once found, can be eliminated in polynomial time.

Acknowledgements

The authors would like to thank the anonymous referees for their useful comments and feedback on the paper. This work has received partial support from from Enterprise Ireland under their Basic Research Grant Scheme (SC/02/289 and SC/2003/007) and their International Collaboration Programme (IC/2003/88) and from the Italian MIUR project "Constraint-Based Verification of Reactive Systems" (COVER).

References

1. B. Aziz, S.N. Foley, J. Herbert, and G. Swart. Configuring storage area networks for mandatory security. In *Proceedings of the* 18^{th} *IFIP Annual Conference on Data and Applications Security*. Kluwer, 2004.
2. D. E. Bell and L. J. La Padula. Secure computer system: unified exposition and MULTICS interpretation. Report ESD-TR-75-306, The MITRE Corporation, March 1976.
3. S. Bistarelli. *Semirings for Soft Constraint Solving and Programming*, volume LNCS 2962. Springer, 2004.
4. S. Bistarelli, S.N. Foley, and B. O'Sullivan. Detecting and eliminating the cascade vulnerability problem from multi-level security networks using soft constraints. In *Proceedings of AAAI/IAAI-2004 (16th Innovative Applications of AI Conference)*, pages 808–813. AAAI Press San Jose, July 2004.
5. S. Bistarelli, S.N. Foley, and B. O'Sullivan. Reasoning about secure interoperation using soft constraints. In *Proceedings of FAST-2004 Workshop on Formal Aspects of Security and Trust*, 2004.
6. S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Solving and Optimization. *JACM*, 44(2):201–236, 1997.
7. G.R. Blakley and D.M. Kienzle. Some weaknesses of the TCB model. In *IEEE Symposium on Security and Privacy*. IEEE CS Press, May 1997.
8. M. Branstad et al. Trusted Mach design issues. In *Proceedings Third Aerospace Computer Security Conference*, 1987.
9. D.E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, 1976.
10. J.A. Fitch and L.J. Hoffman. A shortest path network security model. *Computers and Security*, 12:169–189, 1993.
11. S.N. Foley. A universal theory of information flow. In *Proceedings 1987 IEEE Symposium on Security and Privacy*, pages 116–121, 1987.
12. S.N. Foley. A model for secure information flow. In *Proceedings of the Symposium on Security and Privacy*, Oakland, CA, May 1989. IEEE Computer Society Press.
13. S.N. Foley. Aggregation and separation as noninterference properties. *Journal of Computer Security*, 1(2):159–188, 1992.
14. S.N. Foley. The specification and implementation of commercial security requirements including dynamic segregation of duties. In *ACM Conference on Computer and Communications Security*, pages 125–134, 1997.
15. S.N. Foley. Conduit cascades and secure synchronization. In *ACM New Security Paradigms Workshop*, 2000.
16. J. A. Goguen and J. Meseguer. Unwinding and inference control. In *Proceedings 1984 IEEE Symposium on Security and Privacy*, pages 75–86, 1984.

17. R.J. Horton et al. The cascade vulnerability problem. *Journal of Computer Security*, 2(4):279–290, 1993.
18. T.M.P. Lee. Using mandatory integrity to enforce 'commerical' security. In *Proceedings of the Symposium on Security and Privacy*, pages 140–146, 1988.
19. S. Lewis and S.R. Wiseman. Securing an object relational database. In *ACSAC*, pages 59–68. IEEE Computer Society, 1997.
20. J. McLean. Reasoning about security models. In *Proceedings 1987 IEEE Symposium on Security and Privacy*, pages 123–131, 1987.
21. J.D. McLean. 20 years of formal methods. In *IEEE Symposium on Security and Privacy*, pages 113–114, 1999.
22. J.K. Millen. 20 years of covert channel modeling and analysis. In *IEEE Symposium on Security and Privacy*, pages 113–114, 1999.
23. J.K. Millen and M.W. Schwartz. The cascading problem for interconnected networks. In *4th Aerospace Computer Security Applications Conference*, pages 269–273. IEEE CS Press, December 1988.
24. B.C. Popescu, B Crispo, and A.S. Tanenbaum. Support for multi-level security policies in drm architectures. In *13th New Security Paradigms Workshop*, 2004.
25. R.S. Sandhu. Lattice based access control models. *IEEE Computer*, 26(11):9–19, November 1993.
26. R.S Sandhu. Role hierarchies and constraints for lattice-based access controls. In *ESORICS*, 1996.
27. M. Schaefer. If A1 is the answer, what was the question? an edgy naif's retrospective on promulgating the trusted computer systems evaluation criteria. In *Annual Computer Security Applications Conference*, pages 204–228. IEEE Press, 2004.
28. G. Schellhorn, W. Reif, A. Schairer, P.A. Karger, V. Austel, and D. Toll. Verification of a formal security model for multiapplicative smart cards. In *ESORICS*, pages 17–36, 2000.
29. F.B. Schneider. Enforcable security policies. *ACM Transactions on Information and Systems Security*, 3(1):30–50, February 2000.
30. D. Sutherland. A model of information. In *Proceedings 9th National Computer Security Conference*, 1986.
31. G. Swart, B. Aziz, S.N. Foley, and J. Herbert. Trading off security in a service oriented architecture. In *19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, 2005.
32. TNI. Trusted computer system evaluation criteria: trusted network interpretation. Technical report, National Computer Security Center, 1987. Red Book.
33. U. S. Department of Defense. Trusted computer system criteria. Technical Report CSC-STD-001-83, U. S. National Computer Security Center, August 1983.
34. M. Wallace. Practical applications of constraint programming. *Constraints*, 1(1–2):139–168, 1996.