

available at www.sciencedirect.comjournal homepage: www.elsevier.com/locate/cose

**Computers
&
Security**



Risk profiles and distributed risk assessment

Howard Chivers^{a,*}, John A. Clark^b, Pau-Chen Cheng^c

^aDepartment of Information Systems, Cranfield University, UK

^bDepartment of Computer Science, University of York, UK

^cIBM Watson Research Center, Hawthorne, NY, USA

ARTICLE INFO

Article history:

Received 12 August 2008

Received in revised form

26 April 2009

Accepted 27 April 2009

Keywords:

Risk management

Security

Distributed systems

Security model

Distributed algorithm

ABSTRACT

Risk assessment is concerned with discovering threat paths between potential attackers and critical assets, and is generally carried out during a system's design and then at fixed intervals during its operational life. However, the currency of such analysis is rapidly eroded by system changes; in dynamic systems these include the need to support ad-hoc collaboration, and dynamic connectivity between the system's components. This paper resolves these problems by showing how risks can be assessed incrementally as a system changes, using *risk profiles*, which characterize the risk to a system from subverted components. We formally define risk profiles, and show that their calculation can be fully distributed; each component is able to compute its own profile from neighbouring information. We further show that profiles converge to the same risks as systematic threat path enumeration, that changes in risk are efficiently propagated throughout a distributed system, and that the distributed computation provides a criterion for when the security consequences of a policy change are local to a component, or will propagate into the wider system. Risk profiles have the potential to supplement conventional risk assessments with useful new metrics, maintain accurate continuous assessment of risks in dynamic distributed systems, link a risk assessment to the wider environment of the system, and evaluate defence-in-depth strategies.

© 2009 Elsevier Ltd. All rights reserved.

1. Introduction

Security is fundamentally about reducing the risk of operating an organization, business or system to an acceptable level. Risk assessment or analysis methods (e.g. *Risk Management Guide for Information Technology Systems*, 2002) are therefore at the heart of security management systems (*Information Technology*, 2005), and generally identify threat paths between assets and potential attackers, in order to characterize residual risks. Such risks are quantified in terms of their likelihood, and their impact.

Modern systems are highly interconnected, and increasingly dynamic. Commercial systems change continuously

with joint ventures and mergers, as new business processes are crafted from existing services and as new products are added. Military systems are in transition from an era in which they could be isolated, to one where dynamic information security policies are needed to support ad-hoc collaborators, including civilian Governmental and Non-Governmental organizations.

In all these circumstances, risk assessments carried out during the design cycle, or even periodically, rapidly lose their currency. However, there is no established theory for continuously calculating risk in a dynamic distributed system. This paper provides such a theory; it introduces *risk profiles*, and demonstrates that they provide a risk calculation method which:

* Corresponding author.

E-mail addresses: hrchivers@iee.org (H. Chivers), jac@cs.york.ac.uk (J.A. Clark), pau@us.ibm.com (P.-C. Cheng).

0167-4048/\$ – see front matter © 2009 Elsevier Ltd. All rights reserved.

doi:10.1016/j.cose.2009.04.005

- can be fully distributed;
- is computationally efficient; and
- converges rapidly following a change to the system.

Assessing risk continuously in a distributed system supports the delegation of security decisions to sub-elements of the system, since it provides a principled means of determining if the risks resulting from a proposed policy change will propagate to the wider system, or remain local. Continuous distributed risk calculation is therefore a novel and potentially important security mechanism.

Risk profiles were originally developed as part of the SeDAN risk analysis framework (Chivers, 2006a) to supplement conventional risk assessment; a profile identifies the set of unwanted asset impacts (*concerns*) that a particular element of the system is able to attack. This calculation can be carried out for any part of the system – components, communications, or even its users – providing valuable additional risk-based information to stakeholders, including:

- Implementation guidance: the extent that the deployment or implementation of services needed to be constrained to prevent additional risks to the system, for example from physical access.
- The ability to compare users' actual authority with the privilege that they were expected to have.
- The ability to check if security controls had compromised the liveness of the system, by denying users' legitimate access to assets.
- The ability to balance ICT controls, such as communications security, against the need for physical protection of network components.

Risk profiles have been used to support risk assessments in practice; for example, in the design of a large distributed system to support the maintenance of aero-engines (Chivers and Fletcher, 2005). Profiles were computed for the services in the system, then the services grouped by profile, to provide a ready indication of the risk-sensitivity of different service groups. This provided the customers with the necessary assurance that the most computationally demanding services could be outsourced to a computing grid, with an acceptable level of risk.

One way of viewing risk profiles is that they localise what is otherwise a systemic problem. If the risk profile of a component is known, then adding a single communication to a new component results in the same risk profile at the new component, unless it is otherwise constrained by security controls. This calculation can be carried out locally, without the need to re-evaluate the threat paths in the entire system. This principle can be extended to give a risk calculation method that is fully distributed: components of a distributed system each calculate their own risk profile, given information about risks in neighbouring components.

In this paper we develop and formalise this concept, and then demonstrate that the distributed risk calculation converges to provide the same information as a standard threat path analysis. Furthermore, we show that the calculation method has the properties claimed above: distribution, efficiency, and incremental updating following a system change.

The contribution of this paper is to provide the theory of risk profiles, show how they are calculated, and to evaluate some of their important properties. Although examples in this paper are drawn from specific domains, such as system design or organizational modelling, the theory is generic, and avoids commitment to any particular domain.

This paper is organized as follows. A discussion of related work is given in Section 2. Section 3 provides the motivation for risk profiles, introduces risk analysis terms and standard threat path analysis, describes the problem of interfacing a systematic risk analysis to its wider environment, and informally defines risk profiles. Section 4 provides a formal model of risk profiles, including how they are calculated and how they are extended to attackers. Section 5 evaluates the theory by presenting informal proofs of important model properties, and how these are supported by constraints within the model. Section 6 briefly describes the formal model in algorithmic terms, and provides a worked example of a fully distributed calculation. Section 7 discusses potential limitations of the current formulation, and Section 8 describes possible practical applications and implementations of this work. The paper is concluded in Section 9.

2. Related work

Risk analysis approaches were reviewed by Baskerville (1993) over a decade ago, and his analysis is still relevant today. He identifies the need to use abstract models of systems, which allow a combination of security design and stakeholder analysis, independent of physical implementation. This characterization of the problem is one of the motivations for systematic calculation methods, since it identifies the scope for abstract modelling to make a fundamental contribution to risk analysis.

High level approaches to risk assessment consider a few key factors that contribute to risk, and focus on how they can be combined and communicated to stakeholders. Such approaches provide a decision rationale when it is necessary to estimate risk based on disparate factors, for example combining social, human, regulatory, environmental, quality and technical contributions. A valuable behavioural approach contrasts Consequence Models built by domain experts with Mental Models held by stakeholders (Morgan et al., 2002), to determine areas of weakness in either. Mathematically based approaches include the use of Bayesian networks to provide a decision basis for such models (Fenton and Neil, 2009). A review of methods to combine expert estimates of risk (Clemen and Winkler, 1999), highlights the value of systematic methods, mathematical or behavioural, as opposed to simply using intuition, but is unable to separate the performance of different methods.

Practical approaches to risk assessment are often focussed on identifying paths through a system (which may be technical, organizational, social, etc.) by which an attacker can realise an unwanted impact to an asset. All such methods are concerned with eliciting information, such as assets, system information and security goals, from domain stakeholders, and communicating risks and proposed controls to non-specialists. The CORAS research

project (Braber et al., 2007), exemplifies this approach, by providing a language for documenting risks, developed originally from process metamodels and threat stereotypes for UML. The CORAS language has a graphical representation (Hogganvik and Stølen, 2006) which is focussed on ensuring that stakeholders engaged in the elicitation of risks share a common framework of understanding with the security professionals who build risk models. Other established approaches include OCTAVE (Alberts and Dorofee, 2003), which is focussed on structured elicitation, supported by threat modelling.

There are similar risk analysis methods based on threat modelling (e.g. Swiderski and Snyder, 2004), which describe good practices and principles. Such approaches are often necessarily selective; for example formal models in which only selected threats are modelled for scalability purposes (Xu and Nygard, 2006), or threat trees built from known system vulnerabilities to conduct system penetration evaluations (Sheyner and Wing, 2003).

Other UML-based work includes UMLSec (Jürjens, 2001), which builds on the formal semantics of UML to allow the complete specification of security problems, which can then be subject to formal proof, or exported to a model-checker. This work is typical of a wide range of formal approaches to security; it is promising for small hard security problems, such as protocol analysis, but there is little evidence that this approach will scale to large-scale practical systems, or that it can accommodate risk as a criterion.

As noted above, risk profiles were originally conceived to supplement these types of threat modelling, specifically to allow forms of systematic analysis and communication that are generally absent, for example, checking that security controls do not prejudice functional liveness. This contribution would be a valuable adjunct to the work described above; however, this paper is also concerned with the continuous and distributed calculation of risks, and these issues are not addressed in the existing literature.

The structured enumeration of threat paths has some similarities with Fault Tree Analysis in safety engineering (Vesely et al., 1981); both establish causal trees to undesired events. The safety community has considered the possibility of compositional safety analysis, both to allow the reuse of component safety models and to facilitate design refinement. In one approach, as the design progresses, failure models for components are embedded in a higher level architecture which allows the synthesis of a fault tree (Papadopoulos et al., 2001). Elaborations of this concept include substituting the fault tree with an acyclic directed graph, in order to better represent common-mode failures (Kaiser et al., 2003), and providing a richer logic and the possibility of making choices between component variants (Giese and Tichy, 2006). Unfortunately, the compositional model for safety is not fully applicable to security. Informally, the safety model is a functional one – the propagation of faults through a component is effectively modelled as a function – and although this type of property is important for security it is unable to specify information-flow constraints, such as the controls described in our model. More formally, it is well established that preserving safety properties in refinement does not necessarily preserve security (Jacob, 1989).

Security risk management is essentially a form of requirements engineering, and the goal-refinement community are active in developing new requirements management models (Kalloniatis, 2004), some of which are tool supported, but this work has yet to accommodate risk metrics, or threat analysis. Other researchers in the requirements community focus on security requirements as constraints (Moffett et al., 2004), and provide an argumentation framework that links security goals to security requirements, including identifying the assumptions (trust) on which the arguments depend (Haley et al., 2005).

None of these focus on risk calculation, but some requirements-oriented approaches, including the analysis of attackers as actors (Liu et al., 2003), and abuse cases (McDermott and Fox, 1999) may identify attackers outside the normal system boundary. Related work includes interpreting sequence diagrams (e.g. use and abuse cases) as traces (formal event sequences), which are used to formalise a specification (Haugen et al., 2005). From the security perspective this corresponds to classical refinement, with the well-known problem for security noted above. There is also a long history of attempts to use properties over traces (rather than specific trace enumerations) for security, unfortunately none are completely satisfactory from the perspective of composition or refinement (for example, see McLean, 1994).

Finally, work on security management in dynamic systems, such as mobile ad-hoc networks (Eschenauer et al., 2002), often focuses on trust between nodes; dynamic management approaches include recommendation-based trust measures (Azzedin and Maheswaran, 2003). Approaches that explicitly consider risk in such systems are rare; those that introduce risk do so by using trust calculations as the basis of the likelihood of unwanted security events (Cahill et al., 2003), rather than by distributed systematic evaluation of threat paths.

In brief, the risk profiles described in this paper may be used to supplement existing approaches to risk assessment, by providing additional metrics as described in Section 1; the theory presented here also addresses an issue that is not found in the current literature: the distributed calculation of security risk.

3. Risks outside the scope of system models

Systematic risk analysis is inevitably carried out on a model of a system; different researchers propose and use different models, varying from domain models that include physical zones (Srivatanakul et al., 2004), requirements models that specify an abstract machine in its environment (Moffett et al., 2004), architectural models of information systems (Schneider, 1999), or design models that are relatively close to an implementation (Lodderstedt et al., 2002). There is general agreement that security is a ‘system’ problem; in other words, it is not sufficient to analyze an information system without taking account of its environment.

A feature of all models is that they are, to some extent, abstract. A business process model may not describe paper cheques, or electronic credit mechanisms, although it may identify the need for payments. An architectural model of

a computer system may not describe the physical environment in which software will be deployed, although it may identify abstract services. Implementation features may introduce new security risks because they enrich the domain of discourse (the objects in the system) as well as introducing new and unexpected behaviours. For example, the implementation of a communications path (e.g. sending cheques by post, or electronic transfers over a wireless network) may provide unexpected access mechanisms, or software implementation may introduce vulnerabilities such as buffer-overflows that allow unplanned remote access. Features of this sort may interact with parts of the real world that are outside the scope of the model which is subject to risk analysis. For example, a system model may include users, clients, organizations and administrators, but have no means of representing a social engineering attack against one of these roles, or the path from an external attacker via a physical attack on network infrastructure.

In brief, models used for risk analysis are both bounded and abstract; from a security perspective, this introduces the problem that analysis carried out in the model may not apply to its instantiation in the real world.

What is needed is a process for literally ‘thinking out of the box’ of the model, in such a way that risk analysis carried out in the model can be associated with implementation features that introduce new means of access to the system, perhaps from attackers that are outside the normal system environment.

This is the function of risk profiles; they characterize the risks to a system that would accrue from the subversion of one of its components. Risk profiles were originally developed to support security analysis in the system design process, for example the architectural design of service oriented applications and systems. The remainder of this paper will use examples from this domain, but the principles described here are applicable to the risk analysis of a broad range of different types of system.

The following sub-sections introduce risk profiles informally, beginning with their relationship to standard risk analysis. This is followed by a formal account in Section 4.

3.1. Risk analysis

The terms used in this paper follow the generally accepted definitions in RFC2828 (Shirey, 2000). Security is concerned with defending against attacks that are intentioned by an attacker or adversary, although they may sometimes be more general threat actions (e.g. fire, flood, and lightning). An *attack* may exploit a *vulnerability*, which is a security defect, or may misuse the normal functions of a system to obtain access to an *asset*, resulting in a negative outcome, or *impact*. A *threat path* is the complete path of attack from attacker to asset; a countermeasure, or *control*, blocks a particular threat path. A *risk* is the combination of the likelihood that a threat is realized, and its negative impact.

Concerns. There is no standard word for the undesired events against which assets are to be protected; the term *concern* is used here, to refer to an outcome that stakeholders wish to avoid. A concern includes the impact or cost of a potential security incident. However, it also defines the

incident in other ways, for example: “an integrity failure resulting in loss of provenance.” A concern is a form of security objective which may apply to one or more assets, or classes of asset.

The process of risk analysis discovers the risks in a system by mapping all possible threat paths from potential attackers to assets of concern. This process may be carried out in one of three ways:

- starting with a list of known vulnerabilities and determining if they can be exploited by attackers to cause specific harm to assets;
- starting with an asset of concern and searching for paths that may facilitate an attack; or
- starting with the attacker and searching for ways to reach assets of concern.

The process of starting from the attacker and determining if the tree of possible actions facilitates an attack is usually known as threat tree modelling (Swiderski and Snyder, 2004). The second and third approaches both result in a systematic search of the complete system and can produce equivalent results. Starting from vulnerabilities allows known vulnerabilities to be prioritised in terms of risk by their relation to attackers and assets, but does not necessarily result in a comprehensive risk assessment. The problem of identifying risks associated with unknown vulnerabilities, or equivalently, assessing how much a particular component matters in security terms, is an important contribution of this work.

Risk assessment or analysis is part of a larger process of security management; risks are identified and then considered as candidates for action. Possible management actions include accepting the risk, mitigating the risk with extra security controls, and transferring or mitigating the risk outside the system. This overall process is an important part of the security lifecycle, but is not the subject of this paper.

A typical risk is depicted in Fig. 1. A system contains some active components, such as services (S1–S4), assets of concern (A1, A2) and users (U1). The system model is analyzable: it has a navigable topology, which is constrained by security requirements such as access controls. The available connectivity is indicated by arrows, but operations and access

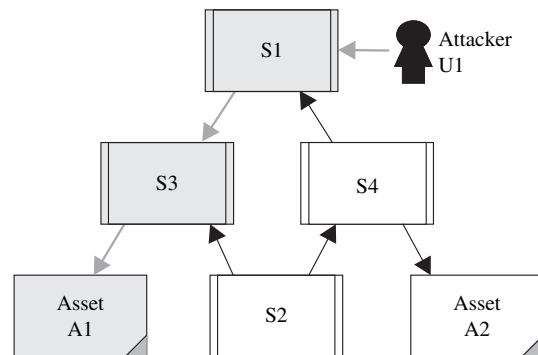


Fig. 1 – Threat paths in a system. The objective of risk analysis is to enumerate the risks in a system by searching for threat paths between attackers and assets.

controls are omitted. Attackers are characterized by the access they have to the system, for example users or system administrators.

In Fig. 1 a particular user (usually a user role rather than an individual) is a potential attacker, and there is a threat path (S1, S3) between this user and an asset of concern (A1). The elements of the system to which the attacker is able to gain access are shown shaded.

This form of analysis is effective at enumerating risks within the modelled system; however, it is also necessary to evaluate risks relating to attackers that do not have any form of legitimate access (e.g. competitors, remote hackers). Attacks from these sources, by definition, must use access mechanisms that are not described in the system model.

3.2. Attacks from outside the system boundary

Consider Fig. 1, again. Although it may not be possible to represent an unknown vulnerability within this model, it is possible to use normal threat path analysis to determine the consequences of such a vulnerability. Suppose that an external attacker is able to access the system via a vulnerability in service S2, the resulting attack is illustrated in Fig. 2.

In Fig. 2 the access mechanism may be unknown, but it is possible to use standard threat path analysis to determine how such an attack will propagate through the system. In this case the attacker is able to gain access to both A1 and A2. In practical terms, this models how an attacker uses trust and privilege relationships within a system to pursue an attack, having gained access via a vulnerability.

This concept extends the usual model of risk analysis to include attackers that are outside the normal system context, but who may be identified by abuse cases (McDermott and Fox, 1999) or other forms of attacker elucidation. However, since the method of access is not part of the system specification, it is also not possible to manage the access point with a security control. Two options are available for risk management:

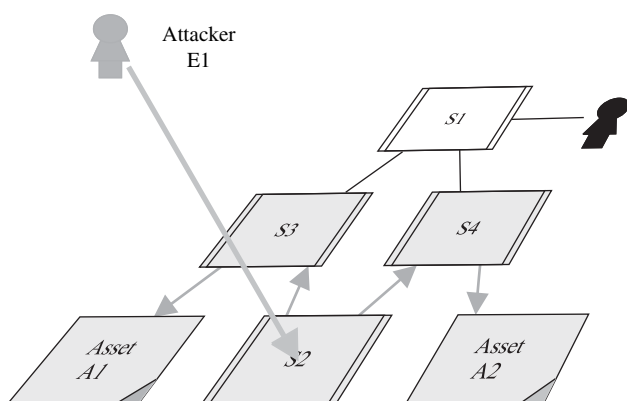


Fig. 2 – Attacks from outside the normal system model. An attacker without normal access to the system gains access to an element of the system via an unspecified vulnerability.

- to change the design of the system, or its security controls, to reduce the propagation of such an attack after it has entered the system; or
- to record the risks that would accrue from a vulnerability that allows a particular attacker to access S2, and use this information to guide the implementation.

These are both valuable additions to a standard risk assessment. The first allows security architectures to be evaluated for the effects of potential component defects, and the second provides implementers with a metric for the quality of assurance or protection required in the implementation.

3.3. Risk profiles

The concepts outlined in the previous section allow the evaluation of threat paths that lie partly outside the normal system model. Such paths are different in nature from the normal paths evaluated during risk analysis, because the point of entry of the attacker is not known. Normal users, for example, access systems at specific known interfaces; however, an attacker from outside the system is likely to use any convenient weak, accessible point, and the extent that a system component offers a weak or accessible access may depend on the implementation. For example, an attacker may physically install a wireless access point on an unprotected wired local area network, use social engineering to obtain user or administrator privileges, or mount a remote attack via a software vulnerability.

In the absence of better qualified information, therefore, such an attacker may attack any part of the system. This could be modelled directly as many different connections, all of which can be evaluated separately; however, the resulting risks can be factored into two parts:

- the external attackers, and their likelihood of attack; and
- the risks that would arise if each system component were subverted.

This concept is formalised in Section 4; informally, a risk is a combination of the likelihood and impact of a particular combination of attacker, threat path and asset:

Risk = impact to asset * likelihood of successful attack

Risk can be factored into two parts: the likelihood that a particular component is successfully attacked and the possibility that subversion of that component exposes a security critical asset:

The likelihood that a component is subverted * (risk to an asset | subverted component)

Both these parts define sets, whose cross product specifies the set of risks from possible attacks against that component to all possible outcomes that are possible via that component. A complete risk assessment would consider all possible components.

The second element, the set of risks given the subversion of a component, is the *risk profile* for that component. A *component* is any part of a system model; so, for example, a risk profile may be calculated for services, communications or even users. Attacks from outside a system are often a significant risk, so risk profiles can be used to indicate the extent to which a component must be protected in its implementation context. For example:

- servers that host services may need to be protected against both physical theft, and implementation defects. A risk profile for a software service indicates the degree to which the server must be protected from its environment, and the required quality of software implementation;
- associations or navigable links in a system model may be implemented as communications. A risk profile may indicate the need for communications security; and
- users may be subverted by social engineering. A risk profile indicates the actual authority exercised by users; this may motivate user education, or the need to separate duties within a business process, for example, by introducing workflow approval tasks. Risk profiles can also be used to check users' actual capability against their planned authority.

Risk profiles essentially compile the systemic problem of evaluating threat paths into local properties for individual system components. This locality is what makes risk profiles valuable for implementers, who are usually concerned with parts of the system. The next section provides a more concrete account of risk profiles and how they are calculated, after which it will be evident that the calculation of risk profiles can also be localised in such a way that risk assessment can be distributed across a system, rather than carried out over a single overall system model.

4. A formal model

The previous section introduced risk profiles informally; before their full value can be explored it is necessary to provide a concrete description of what risk profiles are, and how they can be calculated. This provides a sound basis for justifying the properties described in the next section.

The formal model presented in this section provides a generic model of risk analysis using risk profiles, using a subset of the Z language (Spivey, 1989). The theory presented here has been type-checked using the Formaliser tool (Flynn et al., 1990).

This model is generic because it makes no commitment to a particular level of modelling (e.g. system design or social) or to particular types of functional behaviour or security control. The reader should not therefore interpret the formal names used below as implying any specific type of system.

4.1. Definitions

Four base types are needed:

[Asset, Concern, Comp, Info]

They are disjoint sets, and can be interpreted as follows:

- *Asset* is part of a system which may need to be protected from unwanted security outcomes.
- *Concern* identifies a security objective, or a specific unwanted outcome.
- *Comp* is a component in the system, perhaps a user, administrator, service, business function or even a business department; and,
- *Info* is information; it characterizes messages passed between components.

Concern is used, rather than outcome or impact, since the security objective for an asset has several attributes (see Section 3.1). Those of interest in this model include the asset or assets to which the concern relates, and the impact:

$$\begin{array}{l} \text{ConcernA} : \text{Concern} \leftrightarrow \text{Asset} \\ \text{Impact} : \text{Concern} \rightarrow \mathbb{N} \end{array}$$

ConcernA relates a particular security concern to a number of possible assets, and *Impact* is used as a measure of the cost of the unwanted security event. Impact is modelled as a number, without commitment to any particular scheme of quantification. In most applications *Concern* would also have other attributes, including the direction of the threat (away from or toward the asset).

Note that this model does not specify a direction of information flow in a threat path; messages are defined between two components (left, right) and the flow of risks is from left to right. However, there is nothing in the formalism that requires a component to send information left to right – the actual data could be flowing in the opposite direction to the risks. In the case of a confidentiality concern, the risk will flow out from the asset together with the information; in the case of an integrity concern then the risk will flow from any component able to send information to the asset.

Because this model does not require that risks propagate in the same direction as information, it applies equally to confidentiality and integrity. However, a direction of away from, or toward, an asset should not be taken to imply that this treatment is limited to confidentiality or integrity. In the aero-engine maintenance system mentioned briefly in the introduction (Chivers and Fletcher, 2005), the security goals analyzed included availability and provenance.

Assets are identified with particular components. In general, risk analysis is concerned with classes of assets (e.g. confidential documents, personal data), so a particular class of assets may be 'managed' by more than one component:

$$| \text{Manages} : \text{Comp} \leftrightarrow \text{Asset}$$

There are two dimensions to risk: impact and likelihood. In this model, likelihood is represented as a frequency (*Freq*), and similar to impact, a frequency is a number, with no commitment to any particular scheme of quantification:

$$| \text{Freq} : \mathbb{P} \mathbb{N}$$

It is, however, necessary to represent certainty (100% likelihood); *Freq* therefore has a *Top* value, which is regarded as certainty:

```

Top : N
-----
Top ∈ Freq ∧ (∀ f : Freq • Top ≥ f)
    
```

Risk is usually considered a two-dimensional quantity, in which each risk is characterized by impact and likelihood. It is also necessary to identify the asset and other features of the unwanted security outcome, so we model risk as a combination of likelihood (*Freq*) and *Concern*, which includes impact and also identifies the asset under threat:

```

Risk : Freq ↔ Concern
    
```

For presentational purposes it is useful to be able to extract the individual components of risk:

```

RL : Risk → Freq
RC : Risk → Concern
-----
∀ l : Freq; a : Concern •
  RL (l, a) = l ∧ RC (l, a) = a
    
```

These are the primary definitions used in the model, it is now necessary to model a component, its security properties and the information flows that allow an attacker to access to an asset.

4.2. Introduction to modelling components

Fig. 3 illustrates the main security features of a component that are to be modelled.

The arrowed lines in Fig. 3 indicate information flow; for the purpose of description only the direction away from the asset will be described (confidentiality), but as noted above, the formal model is not limited to a particular direction and can be interpreted equally in both directions.

Information flows that may be part of threat paths originate at assets which are directly associated (*Managed*) with a component, or reach components via messages. The behaviour of a component may support an information flow, and hence a threat path, to other components in the messages that it sends. A component may have security features, or controls, which limit the extent that information about assets and incoming messages are revealed to other components. For example, access controls may prohibit certain messages, or the functionality of a component may be constrained to

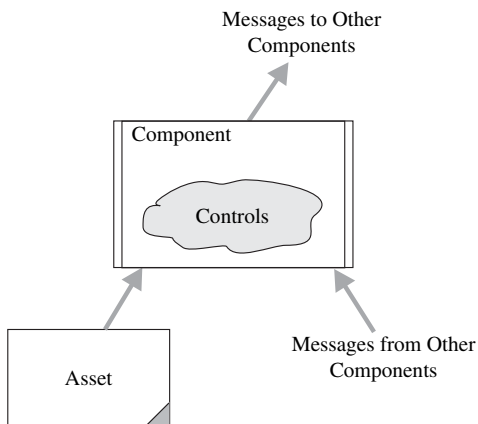


Fig. 3 – Generic component features.

filter information. In this abstract model there is no need to model state explicitly, since the model has, in effect, access to the complete set of possible messages, which could include the complete message history.

In the previous section a risk profile was described in terms of its use: to act as a bridge between the system model and attackers and implementation features that are outside that model. From this viewpoint a risk profile is an intermediate element in the risk calculation; any system component can similarly be viewed as partitioning the system level risk calculation, as shown in Fig. 4.

In Fig. 4 the risk profile for Component A is the set of attacks that are available to that component. The threat paths available to another component via Component A are those available to Component A, moderated by any security controls enforced by that component. This is the motivation behind the formal model that follows.

4.3. The component model

Threat paths terminate at Assets, so a component is directly able to attack the assets within its own context; we identify a function, *Local*, which is able to identify such assets:

```

Local : Comp → (Asset ↔ Concern)
-----
∀ c : Comp; a : Asset; x : Concern •
  (a, x) ∈ Local c
  ⇔ (c, a) ∈ Manages ∧ (x, a) ∈ ConcernA
    
```

At this stage there is no need to identify a likelihood (it is *Top*), so the *Local* function identifies the set of assets with security concerns, which are managed by a specific component.

Threat paths may be facilitated by messages exchanged with other components. Before risks arising from these paths can be formalised it is necessary to model both messages and security controls. A Message is an element of information that is sent between two components:

```

Msg : P(Comp × Info × Comp)
    
```

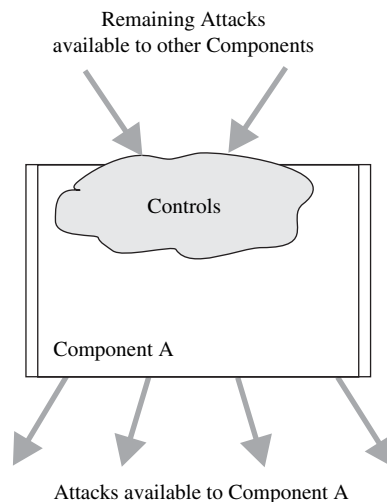


Fig. 4 – The use of risk profiles as intermediate elements in calculating threat paths.

Similar to Risk, it is useful to define functions that extract the elements of this relation:

$$\begin{array}{l} \text{ML} : \text{Msg} \rightarrow \text{Comp} \\ \text{MR} : \text{Msg} \rightarrow \text{Comp} \end{array}$$

$$\forall r, l : \text{Comp}; i : \text{Info} \bullet \\ \text{ML}(r, i, l) = l \wedge \text{MR}(r, i, l) = r$$

The two components are labelled L(ef) and R(ight), if the reader prefers to think in terms of confidentiality (i.e. flow away from the asset), rather than bidirectionally, then the information flow is Left to Right.

The effect of a security control is to reduce the likelihood that a path can be used to attack an asset. It may reduce the likelihood to zero; on the other hand security controls may themselves introduce vulnerabilities or have imperfect strength, so the general case is a function that may reduce the likelihood of attack:

$$\text{Control} : \mathbb{N} \rightarrow \mathbb{N}$$

$$\forall x, y : \mathbb{N} \bullet (x, y) \in \text{Control} \Rightarrow x \geq y$$

Control models the effect of a control, but not its position in the system. We give a generic model for the security behaviour of a component, which is to apply a control, between either assets and messages (XA), or messages and messages (XM). The relations are also characterized by Concern, since in practice different controls are effective in defending against different types of outcome:

$$\begin{array}{l} \text{XA} : \mathbb{P}(\text{Asset} \times \text{Concern} \times \text{Control} \times \text{Msg}) \\ \text{XM} : \mathbb{P}(\text{Msg} \times \text{Concern} \times \text{Control} \times \text{Msg}) \end{array}$$

$$\forall m_l, m_r : \text{Msg}; c : \text{Concern}; x_l, x_r : \mathbb{N} \bullet \\ (m_l, c, (x_l, x_r), m_r) \in \text{XM} \Rightarrow \\ \neg (\exists z_l, z_r : \mathbb{N} \bullet \\ (m_l, c, (z_l, z_r), m_r) \in \text{XM} \\ \wedge x_l > z_l \wedge x_r < z_r)$$

The predicate in this declaration relates to XM only; it ensures that for any particular combination of input message, concern, and output message, the control is consistent, in the sense that it does not invert any pair of risks taken input to output. (Note that two different input risks may result in the same exported risk, and this is common in coarse quantified or qualitative risk calculations.) An example of an inconsistent control, which is prohibited, would be a medium input risk resulting in a medium output risk while a high input risk is reduced to a low output risk (see Theorem 2 in Section 5).

XA does not require a similar constraint; the prior likelihood is always Top for a local asset, so there is no opportunity for this type of inconsistency.

4.4. Exported risk and profiles

The model is now sufficiently well defined to describe the risk calculation. Consider confidentiality. A component may have many different input messages, each originating indirectly from the same asset. Each of these inputs may be subject to different security behaviour within the component, depending on their origin. In order to calculate the extent that a component provides a threat path via a message to a second

component, it is necessary to enumerate the relationship between all the input messages and the output message. There are therefore two steps in the calculation of threat paths that are exported from a component: Enumeration of all possible threat paths to a given output message, then selection of the Exported threat path from the enumerated set:

$$\begin{array}{l} \text{Exported} : \text{Msg} \leftrightarrow \text{Risk} \\ \text{Enumerated} : \text{Msg} \leftrightarrow \text{Risk} \end{array}$$

$$\begin{array}{l} \text{Enumerated} \\ = \{ m : \text{Msg}; r : \text{Risk} \mid \\ (\exists a : \text{Asset} \bullet \\ (a, \text{RC } r) \in \text{Local}(\text{ML } m) \\ \wedge (a, \text{RC } r, (\text{Top}, \text{RL } r), m) \in \text{XA}) \\ \vee (\exists im : \text{Msg}; ir : \text{Risk} \bullet \\ (im, ir) \in \text{Exported} \\ \wedge \text{RC } r = \text{RC } ir \\ \wedge \text{MR } im = \text{ML } m \\ \wedge (im, \text{RC } ir, (\text{RL } ir, \text{RL } r), m) \\ \in \text{XM}) \} \end{array}$$

$$\begin{array}{l} \text{Exported} \\ = \{ m : \text{Msg}; r : \text{Risk} \mid \\ (m, r) \in \text{Enumerated} \\ \wedge \neg (\exists x : \text{Risk} \bullet \\ (m, x) \in \text{Enumerated} \\ \wedge \text{RL } x > \text{RL } r \\ \wedge \text{RC } x = \text{RC } r) \} \end{array}$$

The Enumerated relation is the set of all possible risks for each message, based on both a component's local assets, and also threat paths carried by other messages.

Local assets are identified as those that are managed by the component originating the exporting message (ML m); the component security specification XA can be regarded as establishing the likelihood of the associated risk (RL r) given an asset, concern, message and Top as the prior likelihood.

Risks arising from messages are similarly enumerated. There is, of course, an element of recursion in this definition, since the input risk associated with a message (ir) must have been previously exported (im) by another component. The component specification (XM) maps incoming risk likelihood (RL ir) to outgoing risk likelihood (RL r).

The Exported function comprises the Enumerated risk with the highest likelihood for each concern and message. This selection function is comparable with current qualitative risk assessment, which selects the most likely threat path for each possible concern/attacker pair.

Risk profiles can now be determined for each component:

$$\text{Profile} : \text{Comp} \rightarrow \mathbb{P} \text{Risk}$$

$$\forall c : \text{Comp} \bullet \\ \text{Profile } c \\ = \{ r : \text{Risk} \mid \\ (\text{RL } r = \text{Top} \\ \wedge (\exists a : \text{Asset} \bullet \\ (a, \text{RC } r) \in \text{Local}(c) \\ \vee (\exists m : \text{Msg} \bullet \text{MR } m = c \\ \wedge (m, r) \in \text{Exported})) \\ \wedge \neg (\exists x : \text{Risk} \bullet x \in \text{Profile } c \\ \wedge \text{RC } x = \text{RC } r \wedge \text{RL } r < \text{RL } x) \} \}$$

Profile specifies the risks that would accrue from the subversion of a given component; these are the risks associated with local assets, or risks that have been Exported from other components that communicate with the given component. If

there are several threat paths for the same *Concern*, only the highest likelihood is reported.

Note that the component *Profile* is not the same as the *Exported* risks by that component, because a component may include security controls. Furthermore, since the behaviour of the component includes input to output relations, the *Profile* itself is not sufficient to calculate exported risks: it is necessary to use the input risks to enumerate the possible outputs. However, although *Profile* is defined for only components, note that for any given message path, *Exported* defines a set of risks, which can be informally regarded as a message profile (i.e. the set of asset concerns that could be attacked via that message path).

This completes the formal definition of a Risk profile and its calculation. Section 5 discusses some important properties of this definition, following Section 4.5, which shows how attackers are incorporated in this model.

4.5. Attackers

This section briefly describes how attackers are linked to risk profiles to build a complete risk assessment. The essence is a simple cross product of risks, but the formalisation will be given here for completeness.

A new base type of *Attacker* is needed, and a relation between *Attacker*, *Component* and *Risk* that specifies an attack with a given likelihood of success:

```
[Attacker]
|Attack: P(Attacker × Comp × Risk)
```

We interpret *Risk* at this stage as indicating the attacker's goal (the *Concern*) and the likelihood that the attack will gain access to the given component, rather than the likelihood of the complete threat path.

As previously, it is useful to provide functions that extract the elements of the *Attack* relation:

```
AA : Attack → Attacker
AC : Attack → Comp
AR : Attack → Risk

∀ a : Attacker; c : Comp; r : Risk •
  AC (a, c, r) = c ∧ AR (a, c, r) = r
  ∧ AA (a, c, r) = a
```

The overall system risk combines the likelihood of an attacker gaining access to a component, with the *Profile* that specifies the risks that accrue if that component were subverted:

```
SystemRisk : Attacker ↔ Risk
Product : N × N → N

SystemRisk
= { z : Attacker; r : Risk |
  ∃ a : Attacker; p : Profile;
    pr : Risk •
    pr ∈ Profile (AC a) ∧ z = AA a
    ∧ RC r = RC pr
    ∧ RL r
    = Product (RL pr, RL (AR a)) }
```

This is straightforward matching of *Concerns* at each component between the *Attacker* and *Profile*, with a suitable

product function (e.g. normalised multiplication) of the resulting likelihoods.

5. Properties of risk calculation via profiles

For a risk assessment method to be useful it must converge to estimate the maximum likelihood of any particular risk. Additionally, to be effective for distributed operation, it must be possible to fully distribute the risk calculation between the system components, and correctly propagate the risk consequences of incremental changes to the system without the need to restart the whole calculation. The model described in the previous section has all these properties; this section gives definitions for these properties and justifies why they hold.

The proofs in this section are presented informally, since a formal treatment would require a significant expansion of the formal model presented above, in particular the introduction of temporal aspects that are stated below as assumptions.

Theorem 1. *Component Profiles converge simultaneously at every component, in a time that is no worse than O (number of components).*

Proof. First consider a single asset concern. Divide time into steps; in each step the calculation of *Enumerated* values, *Exported* Values and *Profile* is carried out once at every component. In a system with a single asset concern c , n components, and in which the connectivity is a directed graph with no cycles, after n time steps all reachable nodes will have been given a value for the likelihood of achieving an attack against c . Weakening these assumptions: (1) if the directed graph has cycles: because the control function results in attack likelihoods that are monotonically decreasing, and the *Exported* set contains only the maximum likelihood risk, then cycles do not change existing values; so the values set after n cycles are stable. (2) If there are several asset concerns: they are treated simultaneously in superposition.

Theorem 2. *Profiles specify the maximum likelihood with which the component is able to build a threat path to an asset concern.*

Proof. Consider a threat path comprising a series of *Exported* risks from components c_0, c_1, \dots culminating in a risk profile in a component c_n . If there is a threat path with higher likelihood, then by definition it must result in an imported risk to some component c_x , where x is in the existing path $0, \dots, n$. If $x = n$, then the higher likelihood is included in the risk profile for component n . Otherwise, because the *Exported* relation includes the maximum likelihood risk, it will result in a higher value exported from c_x . Because the control function is consistent (see Section 4.3) a higher value input into the remainder of the threat path will result in either the same or a higher value imported into c_n . If the value imported into c_n is the same, there is no overall higher likelihood path. Therefore, after the system has converged (see [Theorem 1](#)) there can be no higher likelihood path than those given in the risk profiles,

because any such path would have resulted in a higher likelihood in the profile.

Theorem 3. *The calculation of risk profiles can be fully distributed.*

Proof. The elements of *SystemRisk* required for risk calculation are classified by component; messages specify the components to which they relate, and the *Enumerated*, *Exported* and *Profile* relations require only *Enumerated* risks from components with which messages are exchanged, and knowledge of local assets. Similarly, *Attacker* information is classified by component, and does not require information from other components. In other words, the calculation requires only local information, together with the exchange of risk information between components that are directly exchanging messages.

Theorem 4. *Any change to the system can be incrementally propagated.*

Proof. Consider the cyclic calculations described in [Theorem 1](#); if the recalculation of a component's *Exported* risks results in a new value, then this change will be exported to all reachable elements of the system in exactly the same way. The reachable elements of the system from a given exported risk are the same as those from the original exported risk, so the system will be fully updated. This has been described in terms of continuous recalculation; however, repeated enumeration will produce the same result unless either an input changes or the security model for the component changes. It is therefore necessary only to calculate and propagate changes.

Note that only changes to a component's security model are discussed here; the definitional style used to specify this model is such that it is possible to consider the model as already encompassing all possible objects (components, messages, etc.), but they are simply not incorporated in the risk analysis until a component security model specifies that this is the case.

An important consequence of this argument is that it provides a criterion for deciding if a security policy change is local, or if it may have wider consequences. If a change to a security policy does not result in a change to the exported risks, then its effect is local. This has important applications in dynamic systems, where it may be necessary to judge if a local policy change has wider consequences.

These proofs rely on important properties of the *Control*, *XM* and *Exported* relations, namely:

- that the application of a sequence of controls results in a monotonically decreasing sequence of risk likelihood;
- that controls are consistent, in the sense defined in [Section 4.3](#); and
- that the *Exported* function takes the maximum likelihood value from candidate enumerated risks.

Other risk calculation functions are possible and may be desirable; however, the design of alternative functions that support the properties described in this section is a topic for future work.

6. Algorithmic description

Following from [Theorem 3](#), the model given in [Section 4](#) provides a forward-propagation algorithm for calculating the risk profiles in a distributed system; risks are simultaneously propagated out from assets. For each component, the algorithm is:

- (1) Carry out the following steps if:
 - the component is new; or
 - the security model of the component is changed (e.g. new access policy); or
 - the risks exported to this component by another component change.
- (2) Compute the *Risk Profile* for the component, essentially by summing the input risks.
- (3) Enumerate the output risks by filtering the input risks and assets through the component's security model.
- (4) Select the maximum likelihood for each risk and notify the resulting set of *Exported* risks to neighbouring components.

We remark that although this process can be fully distributed, it does not have to be. It is unlikely that every node in a practical system will calculate its own risk profile, both for practical and security reasons. For example, system users may be regarded as components, allowing the calculation of standard threat paths between users and assets, but this calculation is likely to be carried out in an authorization or risk-based information management server, rather than by the users themselves.

6.1. Worked example

This section provides a simple worked example to demonstrate distributed risk calculation. The system for this example is given in [Fig. 5](#).

[Fig. 5](#) presents a simple office system: users have access to a *Front* service which allows them to place and edit orders; a *Back* office service retrieves orders and associated personal information, and maintains user accounts; finally there is a management facility that allows an administrator to view and modify orders and accounts, via the other services.

An important security objective of this configuration is to ensure that users do not have access to back office data, such as user account or credit card information, even in the event of a unidentified security vulnerability in the *Front* server. Systematic analysis of this objective requires a way of quantifying defence-in-depth (a chain of security controls), in addition to conventional threat path analysis.

[Fig. 5](#) also depicts the primary message flow; this example will be restricted to just these messages, without loss of generality; a practical system would have many more messages to accommodate, and readers are referred to [Chivers \(2006a\)](#) for a detailed account of how service oriented systems may be mapped to messages, and how practical security requirements can be modelled in such a system.

In this example:

- The components are three services (*Front*, *Back*, and *Admin*) and two users (*User*, *Manager*).

- The only messages allowed between components are those indicated in Fig. 5, together with those that flow between the *User* or *Manager* and their respective service. In the tables below, messages will be abbreviated to indicate position, flow and type. For example *A-F-Write* is the *Write_Order* message from the *Admin* to the *Front* service.
- There are two Concerns, A at Accounts and O at Orders, they are both for confidentiality, so risk propagates away from assets in the same direction as the message flow.
- The scale of likelihood is 0, ..., 8.

The security model for each service (XA and XM in the formal model) specifies the control, or reduction in risk, offered by the component. In this example the *Concern* element is omitted (i.e. the control in each component is simple and does not distinguish between different types of message).

The security models in this example are relatively simple; they represent the change in risk through a component as one of two possible factors: 1 or 0.5. The first represents no control over the message flow; the second represents a security control which significantly reduces the risk (e.g. an access control). Such a control could be modelled as reducing the risk to zero; however, there is a benefit in representing such a control as a fractional product, since this allows judgements to be made about defence-in-depth, as shown below.

The security model for the *Admin* service is shown in Table 1.

In essence, this policy allows the *Manager* to access either the *Front* or the *Back* service, but maintains separation between the two services by preventing information read from the *Back* service being sent to the *Front* service, and vice-versa. The result is that risks imported from the *Front* service will be correspondingly reduced before they are exported to the *Back* service.

The equivalent security model for the *Front* service is given in Table 2.

This model specifies that the component reduces the risk of users seeing orders; it does not specify the actual nature of the security control. For example, users may be allowed to see their own orders, but not those of other users.

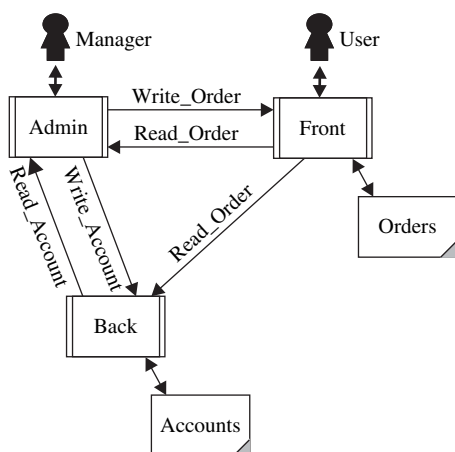


Fig. 5 – System for worked example: front/back office.

The distinction between a user’s own data and that of others could be modelled using message *Info*, and brought within the model; alternatively, the risk reduction required could be documented and remain as an implementation objective. Both approaches are supported by this model.

The equivalent security model for the *Back* server has no controls that reduce risk, in this system the risk is managed by placing a boundary around the server rather than by local controls. Of course, the implementation of the messages depicted in this example would require the *Back* server to authenticate requests for information (e.g. is it from the genuine *Admin* service), but that is beyond the scope of this example.

Table 3 shows how the risk calculation progresses from an initial null start. Each pair of rows in the table is a new time-slot; the first row in each pair shows the risk profile, and the second shows the risks that are exported along message paths to continue the calculation. Risks are a combination of the concern (A or O) and likelihood (0, ..., 8).

Risks are calculated following the above algorithm. For example, in timeslot 3: enumerating the risks exported to *A-F-Write* results in a profile of A4, O4, as a result of applying the security model to the profile imported via *B-A-Read* (A8, O8) and a profile of O8, as a result of applying the security model to the profile imported via *F-A-Read* (O8); the result is the combined profile of A4, O8. Although the risks imported to the enumeration have changed in this timestep, the *Exported* risks are the same; the process therefore terminates since further iterations do not change the risk profiles.

Theorem 1 states that component profiles converge simultaneously at every component in a time that is no worse than O (number of components). This example has converged in three steps in a system with five components, so the example is consistent with the theorem.

Theorem 3 states that the calculation of risk profiles can be fully distributed. This example demonstrates that result, since only risks exported along message paths are used to calculate risk profiles for components. The risk calculation for each component is therefore local to each component; specifically, there is no requirement for global knowledge or calculation. In a naive implementation each component could calculate its own profile, with the same result.

Although this is a very simple model, the distributed calculation has provided some important information: it confirms that the *Front* server does not have a direct path of attack to the *Accounts* asset, since the risk profile for the *Front* server is (A4, O8); furthermore, in the worst case there are two security barriers between this asset and normal system users. This confirms the design of the system: that the *Front* server is

Table 1 – Admin service security model.

		Output		
		A-B-Write	A-F-Write	Manager
Input	B-A-Read	1	0.5	1
	F-A-Read	0.5	1	1
	Manager	1	1	-

Table 2 – Front service security model.

		Output			
		F-A-Read	F-B-Read	User	Order
Input	A-F-Write	1	1	0.5	1
	User	1	1	-	1
	Order	1	1	0.5	-

the primary barrier, but if this is subverted, there is a further security control between the attacker and back office data.

Theorem 2 states that profiles specify the maximum likelihood with which the component is able to build a threat path to an asset concern. Because this is a small system, this property can be confirmed by inspection: there is no higher likelihood path between the *Front* server and *Accounts* data.

Theorem 4 states that the calculation process is incremental, in other words that the risk calculation is guaranteed to correctly converge following a system change, and in consequence it is possible to determine locally if a change in security policy has wider consequences. Consider a change to the security model of the *Admin* server: a local administrator decides that there is no harm in allowing information to flow from *Orders* to *Accounts*, and implements new software that bypasses the previously strict separation.

This modifies the security model in Table 1: the risk propagation value from *F-A-Read* to *A-B-Write* is now '1', to indicate there is no longer any security control.

Because the security model for the component has changed, the algorithm requires re-evaluation of the profiles and exported risks in the *Admin* component. Since the component security policy does not influence its profile directly, the component profile is unchanged. In this case, *A-B-Write* is the only output that might change, and its profile is already equal to *F-A-Read*, so the exported risks do not change. This confirms that the change is genuinely local, and is not propagated to the rest of the system.

This demonstrates the convergence property in Theorem 4 – there was no need to restart the risk evaluation from scratch – and also the localisation that is achieved by this approach. It is possible for a component to determine which changes in policy must be propagated, and which remain local.

This example has demonstrated a simple implementation of the model; practical risk assessments require more complex types of security model. However, even in this

example it has been possible to demonstrate features that are not found in conventional risk path analysis:

- efficient distributed calculation;
- enumeration of defence-in-depth; and
- a criterion for when a policy change in a distributed system is localised in its consequences.

7. Potential limitations

Potential limitations of the model presented here arise in its application, and in the constraints needed to ensure the properties described in Section 5.

The SeDAn framework (Chivers, 2006a) supports both the calculation of risk profiles, and also conventional threat path analysis, and gives a detailed taxonomy of controls and their associated models. The current tooling maps a system architecture or design in UML to an information model (Chivers, 2006b) which is a specialization of the model presented here. One reason for this specialization is to allow a range of practical security requirements to be expressed (Chivers and Jacob, 2005), as a refinement of the generic model of behaviour described in Section 3.3. The specialization includes binding of user roles to message information, a type system that facilitates the specification of communications security controls, and further attributes to Concerns, in order to specify threat path exploitability and attack direction. This demonstrates the successful application of profile calculations; however, it also highlights the need to be wary of the interpretation of 'connectivity' in the context of end-to-end security services.

End-to-end communications security services allow security properties, such as message confidentiality, to be maintained between two endpoints, regardless of the intermediate components. As a result, message flows occur between immediate communication neighbours, as would be expected, and also between endpoints that are 'virtually' connected via an end-to-end security service. This problem does not limit the use of the model presented in this paper; however, it does highlight the need for a careful mapping between this model and concrete system architectures.

Section 5 describes how constraints on security controls (monotonically decrease risk, consistent) and aggregation

Table 3 – Incremental calculation of the risk profiles in Fig. 5.

Time	Services			Users		Assets		Message paths				
	Admin	Back	Front	User	Manager	Order	Account	B-A-Read	F-A-Read	A-B-Write	F-B-Read	A-F-Write
0						O8	A8	-	-	-	-	-
1	-	A8	O8	-	-	O4	A8	A8	O8	-	O8	
2	A8, O8	A8, O8	O8	O4	A8, O8	O8	A8, O8	A8, O8	O8	A8, O4	O8	A4, O8
3	A8, O8	A8, O8	A4, O8	A2, O4	A8, O8	A4, O8	A8, O8	A8, O8	A4, O8	A8, O4	A4, O8	A4, O8

(maximum risk) ensure that the risk calculation converges to stable maximum likelihood risks. It is possible to imagine systems in which a different risk aggregation function is required; for example, a weighted combination of risks that reflects the intuitive view that many threat paths to a given asset are a higher risk than a single path, even if the maximum risk likelihood is the same. We conjecture that other functions will also allow convergence, but the general issue of how these constraints can be weakened to allow different risk aggregation strategies is an open question.

Constraints on the component security model are already minimal, in the sense that there is no scope to weaken the consistency constraint and maintain the properties discussed in Section 5. However, the specification of component security models is flexible enough to encode a range of approaches to the assessment of defence-in-depth. Section 6.1 provides one possible approach, but other possibilities include controls that are not regarded as reducing risk exponentially, but saturate after a small number of controls. For example, some accreditation schemes specify that a second control in a given threat path is regarded as reducing risk, but subsequent controls have no further effect, because of potential common-mode vulnerabilities. These practical models could be accommodated within the constraints required by our formulation.

In summary, the theory presented here is believed to be widely applicable; other risk propagation models are a topic for future research, as are the deployment of distributed calculations within a dynamic distributed system. The mapping of messages in this model to actual system messages needs particular care.

8. Applications

As noted in Section 1, risk profiles were originally developed as part of the SeDAn risk analysis framework (Chivers, 2006a) to supplement conventional risk assessment. They are integrated within the analysis tooling to provide profiles for different parts of the system (e.g. services, communications, users) with the result that novel and useful metrics can be derived. For example, it is possible to enumerate a users' actual authority, to determine the exposure of the system if a service were outsourced or deployed, and to infer the degree of physical protection required of a communications link.

This is the most direct application of risk profiles, to supplement conventional risk assessment, and is currently under further development to address the problem of risk comprehension in large industrial and government assessments. A classical risk assessment of a real system may result in a very large risk register, which is difficult to communicate directly to system stakeholders, or represent graphically. Risk profiles may be used to identify 'hot components', which are not vulnerabilities, but bottlenecks through which attacks tend to flow; these bottlenecks often provide good candidate positions for security controls. We also exploit the relationship between risk profiles and conventional risk analysis by annotating the risks in the risk register with references to hot components, providing stakeholders with evidence that

focussing on these components does provide coverage of the critical risks.

There are two potential applications of risk profiles in distributed systems: determining if policy changes are local, and distributed calculation. Both are concerned with managing security when systems change.

Large systems are subject to constant change, for example by the introduction of new applications, and so the currency of a risk assessment is always at risk. The critical question from a security perspective is if the consequences of the change are local, or if re-assessment of the whole system is required. The theory described here provides a principled way of making that decision.

Finally, because the calculation of risk profiles can be fully distributed, it provides a means to continuously assess risks within dynamic networks. In such networks, components may join and leave during normal operation, and the reliability of components and users may change with time. Such networks include ambient or pervasive computing systems, and ad-hoc military networks. The theory presented here facilitates real-time risk calculation in such systems.

This is a brief review of some possible applications of the theory of risk profiles presented in this paper; the first two are already finding application within industrial practice, the application to distributed systems is still a research topic.

9. Conclusions

This paper has described *risk profiles*, which provide a mechanism for distributed and incremental risk assessment by characterizing the risks to which a system would be exposed if particular components are subverted.

Risk profiles have important applications in standard risk analysis, since they allow connections to be made between the systematic analysis of threat paths in a system model, attackers who may be outside the domain of the model, and security defects in its implementation. For example, they allow the review of the actual degree of privilege exercised by system users, and this can be related to the opportunity for insider attacks, collusion, or external social engineering.

We show that the calculation of risk profiles can be fully distributed in such a way that the risk estimates converge to the same values that would be achieved by threat path analysis. The distributed calculation also has some important properties that are not usually found in conventional threat analysis, including the ability to enumerate the value of defence-in-depth using a user-defined policy.

The distributed algorithm also provides a locally enumerable criterion to determine if a policy change in a distributed system is localised. This has particular significance for policy management in dynamic distributed systems, since it allows the security consequences of system changes to be evaluated dynamically within a distributed system.

This paper introduces a new concept in risk calculation, and inevitably leaves scope for future work. The practical application of the theory presented here is an important topic for future study, and there is also scope to enhance the formal model sufficiently to allow fully formal proofs of the

properties we describe. The convergence of risk aggregation functions, other than the maximum-likelihood function used here, is also an open question.

Acknowledgement

This research is continuing through participation in the International Technology Alliance sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence. We are also grateful for the constructive comments of anonymous referees.

REFERENCES

- Alberts C, Dorofee A. Managing information security risks – the octave approach, SEI series in software engineering. Addison-Wesley; 2003.
- Azzedin F, Maheswaran M. Trust modeling for peer-to-peer based computing systems. In: Proceedings of the international parallel and distributed processing symposium (IPDPS'03). IEEE Computer Society; 2003.
- Baskerville R. Information systems security design methods: implications for information systems development. *ACM Computing Surveys* 1993;25(4):375–414.
- Braber F, Hogganvik I, Lund MS, Stølen K, Vraalsen F. Model-based security analysis in seven steps – a guided tour to the CORAS method. *BT Technology Journal* 2007;25(1):101–17.
- Cahill V, et al. Using trust for secure collaboration in uncertain environments. *IEEE Pervasive Computing* 2003;2(3):52–61.
- Chivers H, Fletcher M. Applying security design analysis to a service based system. *Software Practice and Experience: Special Issue on Grid Security* 2005;35(9):873–97.
- Chivers H, Jacob J. Specifying information-flow controls. In: Proceedings of the 25th IEEE international conference on distributed computing systems workshops (ICDCSW'05) second international workshop on security in distributed computing systems (SDCS). IEEE Computer Society; 2005. p. 114–20.
- Chivers H. Security design analysis. York, UK: Department of Computer Science, The University of York; 2006a. p. 484.
- Chivers H. Information modeling for automated risk analysis. In: Proceedings of the 10th IFIP open conference on communications and multimedia security (CMS 2006); 2006b.
- Clemen RT, Winkler RL. Combining probability distributions from experts in risk analysis. *Risk Analysis* 1999;19(2):187–203.
- Eschenauer L, Gligor VD, Baras J. On trust establishment in mobile ad-hoc networks. In: Proceedings of the security protocols workshop. Lecture notes in computer science (LNCS), vol. 2845. Springer-Verlag; 2002. p. 47–66.
- Fenton N, Neil M. Combining evidence in risk analysis using Bayesian networks, Agenda white paper, W0704/01; 2004. Available from: <http://www.agenarisk.com/resources/technology_articles/Combining_Evidence.pdf> [accessed January 2009].
- Flynn M, Hoverd T, Brazier D. Formaliser – an interactive support tool for Z. In: Proceedings of the fourth annual Z user meeting. British Computer Society; 1990. p. 128–41.
- Giese H, Tichy M. Component-based hazard analysis: optimal designs, product lines, and online-reconfiguration. In: Proceedings of the SAFECOMP '04. Lecture notes in computer science, vol. 4166. Berlin/Heidelberg: Springer; 2006. p. 156–69.
- Haley CB, Laney R, Moffett JD, Nuseibeh B. Arguing satisfaction of security requirements. In: Mouratidis H, Giorgini P, editors. Integrating security and software engineering: advances and future vision. Hershey, PA and London: Idea Group Publishing; 2005. p. 15–42.
- Haugen Ø, Husa KE, Runde RK, Stølen K. STAIRS towards formal design with sequence diagrams. *Software and Systems Modeling* 2005;4(4):355–7.
- Hogganvik I, Stølen K. A graphical approach to risk identification, motivated by empirical investigations. In: Proceedings of the ninth international conference on model driven engineering and systems (MoDELS '06). Lecture notes in computer science (LNCS), vol. 4199. Berlin: Springer; 2006. p. 574–88.
- Information technology – security techniques – information security management systems – requirements. International Organisation for Standards (ISO); 2005. ISO/IEC 27001:2005.
- Jürjens J. Towards development of secure systems using UMLsec. In: Proceedings of the fundamental approaches to software engineering: fourth international conference, FASE 2001: held as part of the joint European conferences on theory and practice of software, ETAPS 2001. Lecture notes in computer science, vol. 2029. Springer-Verlag; 2001.
- Jacob JL. On the derivation of secure components. In: Proceedings of the 1989 IEEE symposium on security and privacy. IEEE Computer Society; 1989. p. 242–7.
- Kaiser B, Liggesmeyer P, Mäckel O. A new component concept for fault trees. In: Proceedings of the eighth Australian workshop on safety critical systems and software; 2003. p. 37–46.
- Kalloniatis C. Security requirements engineering for e-government applications: analysis of current frameworks. In: Proceedings of the electronic government: third international conference, EGOV 2004. Lecture notes in computer science, vol. 3183/2004. Springer-Verlag; 2004. p. 66–71.
- Liu L, Yu E, Mylopoulos J. Security and privacy requirements analysis within a social setting. In: Proceedings of the 11th IEEE international requirements engineering conference (RE'03). IEEE Computer Society; 2003.
- Lodderstedt T, Basin D, Doser J. SecureUML: a UML-based modeling language for model-driven security. In: Proceedings of the fifth international conference on the unified modelling language. Lecture notes in computer science (LNCS), vol. 2460. Springer-Verlag; 2002. p. 426–41.
- McDermott J, Fox C. Using abuse case models for security requirements analysis. In: Proceedings of the 15th annual computer security applications conference; 1999.
- McLean J. A general theory of composition for trace sets closed under selective interleaving functions. In: Proceedings of the IEEE symposium on research in security and privacy. IEEE Computer Society; 1994. p. 79–93.
- Moffett JD, Haley CB, Nuseibeh BA. Core security requirements artifacts. Open University, Department of Computing. Available from: http://computing-reports.open.ac.uk/index.php/content/download/166/999/file/2004_23.pdf; 2004. Technical Report no. 2004/23 [accessed January 2006].
- Morgan MG, Fischhoff B, Bostrom A, Atman CJ. Risk communication: a mental models approach. Cambridge, UK: Cambridge University Press; 2002.
- Papadopoulos Y, McDermid J, Sasse R, Heiner G. Analysis and synthesis of the behaviour of complex programmable electronic systems in conditions of failure. *Reliability Engineering and System Safety* 2001;71(3):229–47.
- Risk management guide for information technology systems. SP 800-30. National Institute of Standards and Technology (NIST). Available from: <http://csrc.nist.gov/publications/nistpubs/800-30/sp800-30.pdf>; 2002 [accessed January 2006].
- Schneider EA. Security architecture-based system design. In: Proceedings of the 1999 workshop on new security paradigms. ACM Press; 1999. p. 25–31.

- Sheyner O, Wing J. Tools for generating and analyzing attack graphs. In: Proceedings of the formal methods for components and objects symposium, vol. 2004/3188. Berlin, Heidelberg: Springer-Verlag; 2003. p. 344-71.
- Shirey R. Internet security glossary. Reston, USA: The Internet Society. Available from: <http://ietfreport.isoc.org/rfc/rfc2828.txt>; 2000. RFC 2828[accessed January 2006].
- Spivey JM. In: Hoare CAR, editor. The Z notation: a reference manual. Prentice hall international series in computer science. Prentice Hall; 1989.
- Srivatanakul T, Clark J, Polack F. Security zonal analysis. York, UK: University of York, Department of Computer Science. Available from: <http://www.cs.york.ac.uk/ftpd/ftpdir/reports/YCS-2004-374.pdf>; 2004. Technical Report YCS-2004-374.
- Swiderski F, Snyder W. Threat modelling, Microsoft professional. Microsoft Press; 2004.
- Vesely WE, Goldberg FF, Roberts NH, Haasl DF. Fault tree handbook. U.S. Nuclear Regulatory Commission; 1981. Report NUREG-0492209.
- Xu D, Nygard KE. Threat-driven modeling and verification of secure software using aspect-oriented petri nets. IEEE Transactions on Software Engineering 2006;32(4): 265-78.

Howard Chivers is Director of the Centre for Forensic Computing at Cranfield University. His research interests are in system security and computer forensics, and current security projects include risk management in dynamic collaborative networks, the identification of subtle intrusions within computer networks, and the security of industrial GRID applications. He is also a security practitioner, providing security advice and methodology for air traffic management

within the EEC. His previous career includes time in both Industry, developing cryptographic products, and Government, managing the computer security research program for the UK National Authority for Information Security.

John A. Clark is Professor of Critical Systems at the University of York. His work is focussed on software engineering (particularly testing) and secure systems engineering. He has adopted techniques inspired by natural systems to address problems including automated testing of implementations against formal specifications, automated secure protocol synthesis, the design of cryptographic components, cryptanalysis, and most recently the use of genetic programming to evolve quantum circuitry. Before joining York in 1992 he worked on UK Government-funded evaluation and R&D security projects, and he has provided consultancy to industry on various aspects of dependability modelling.

Pau-Chen Cheng is a researcher at the Watson Research Center. His areas of interest are in the system aspects of computer and network security, including the design and implementation of solutions for data encryption and authentication, key management, user authentication, and Internet security. Pau-Chen joined the Computing System Department of the Research Center in 1990 to work on the development of security functions of AIX. In 1994 he joined the Network Security Group in Research to work on IP Security technology. He is the principal developer of the IPSec and VPN technology on the AIX operating system.