Applying Security Risk Analysis to a Service-Based System

Howard Chivers¹ and Martyn Fletcher²
Department of Computer Science, University of York, Heslington, York, YO10 5DD, UK.

SUMMARY

Risk analysis is the only effective way of making value judgments about the need for security. Established analysis methods apply to whole operational systems, taking a necessarily holistic view of security, but this makes them difficult to integrate into the design process for service-based applications, where design and implementation is independent of operational deployment. However, the most costly mistakes occur early in the development lifecycle, and effective security can be difficult to retrofit, motivating the need for early security analysis. This paper describes SeDAn (Security Design Analysis), a security risk analysis framework that is adapted for use in the design phase of service-based systems, and its application to a significant grid-based application (DAME). The complete lifecycle of the risk analysis is described, and the effectiveness of the process in identifying design defects validates both the need for, and the effectiveness of, this type of analysis.

KEY WORDS: Security, Risk, Design, Distributed Systems, Service-Based, Grid

1. INTRODUCTION

Security Risk analysis is important because it provides a criterion for the value of security in the business and social context of a system. It is the only viable method of providing a cost benefit justification for security controls, and is therefore the usual basis for information security management standards and methods [1-3]. Security risk evaluation necessarily includes the whole business environment: business processes, people, and physical infrastructure, as well as the information system. However, it is difficult to relate this holistic approach to the lifecycle of systems engineered using service-based architectures, such as web-services or grid applications, because the deployment of these systems is deliberately separated from their design.

The high cost of cascading early design mistakes is well established [4, 5] and it may be prohibitively difficult to correct some security design flaws ([4, and [6] Chapter 2). It is therefore desirable to consider security early in the system lifecycle, but this requires the analyst to reason about security in an abstract design, with only partial knowledge of how and where it will be deployed.

To address these problems we have developed the SeDAn (Security Design Analysis) framework, which allows security risk analysis to be applied early in the system design process. At this stage in the system lifecycle it is not possible to account for implementation defects, which have to be evaluated later, but the framework does provide a systematic approach to analysing, defining and documenting the security controls in a high-level design and, importantly, allows a design to be analysed for deep structural security flaws.

The Distributed Aircraft Maintenance Environment (DAME [7]) is an e-Science pilot project to provide a diagnostic system for aero engines, implemented as a set of collaborating services using the Grid computing paradigm [8]. The input to DAME is monitoring and sensor data obtained during flight, and the system provides a collaborative environment where expert users in different

¹ Email: hchivers@iee.org
² Email: martyn.fletcher@cs.york.ac.uk
organisations work together to investigate sensor data features that may indicate particular engine conditions. The system includes a range of diagnostic functions, including tools to search previous patterns of behaviour and model expected engine performance. The collaborating users develop diagnoses and prognoses, and optimise the planning of remedial maintenance to minimise its operational impact.

The final DAME system will span several companies and support high-valued contractual relationships, so it is inevitable that the stakeholders of this system have critical security requirements. The most significant of these relates to the provenance of a diagnosis: how it was determined, what supporting evidence or data was used, and how it was communicated. The project has two industrial customers:

- Rolls-Royce plc, who will use DAME as a diagnostic tool during engine testing, and provide expertise to advise on operational diagnoses.
- Data Systems and Solutions Llc, who will use DAME within their aero-engine data management, maintenance planning and maintenance prediction systems, and provide diagnosis as a service to fleet operators.

Other industrial stakeholders include the aero-fleet operators and the owners of industrial properties used in the system. Four development teams each contribute unique expertise to the project:

- The University of York: pattern matching, grid services and security analysis.
- The University of Leeds: workflow, provenance and service level agreements.
- The University of Sheffield: engine simulation and case based reasoning.
- The University of Oxford: signal processing and data management.

This project is therefore a demanding test case for design-level security analysis: it has essential non-functional requirements and is a complex, distributed, service-based system. The aim of security analysis is to provide stakeholders with a review of the pilot system design, and since the deployment of the pilot is not representative of the operational system it is important to be able to separate security flaws in the design from details of its experimental deployment.

This paper describes our experience in applying the SeDAn framework to DAME [7]. Our initial report [9], presented at the Oxford Grid Security Workshop, is extended by describing the experience in detail and by including more comprehensive analysis results that were achieved using an automated risk-analysis tool, the Security Analyst Workbench (SAW). Because the framework is new, the process is described in enough detail to allow a reader to understand how the practical results were obtained; other aspects of the framework, including theory, metamodels and tool support, are not described in detail.

This study exposed design flaws in DAME that will need to be addressed in future work. This does not imply criticism of the project or any of its partners, all of whom welcomed and helped with this study; problems are simply to be expected in the design of large distributed systems, and exposing real design issues both helps the development of DAME, and validates the analysis process. This work also highlights some aspects of DAME that may be of wider relevance to grid engineering: part of the system is not suitable for grid deployment, and other parts have security profiles that are particularly suitable.

This rest of this paper is organized in five parts. Section 2 describes related work. Section 3 outlines conventional security risk analysis and then describes how this is adapted by SeDAn for use with service-based systems. The practical application of this framework to DAME is described and summarized in section 4. Section 5 provides general conclusions about the analysis process, and Section 6 concludes the paper.
2. RELATED WORK

There is no directly comparable work that supports risk analysis in the design process [10], partly because the separation of system design from physical deployment is itself a developing concept.

Risk management methods are the basis of most national standards for information security management [1-3, 11] and best practice [2, 12], and are proposed as a rational approach to broader security choices in society [13]. Differences between the various methods include how risks are quantified, the degree of skill required by the practitioner, and the degree to which the process is organization rather than system focused. The subject has been summarized from an academic perspective by Baskerville [14] and from a practice viewpoint by Straub [15]. Baskerville’s analysis is still particularly relevant. He describes the design of secure systems in three generations: the first is checklist based, the second engineering based and the third should use abstract reasoning.

Various combinations of checklist and engineering-based methods can be found in the established methods, and some include both (e.g. OCTAVE [1]). These methods generally support organizational audit, rather than system design, so the associated tools support risk tables, questionnaires, and standard report formats [16]. More integrated tools also exist, usually adding checklists of standard attacks and operational practice, and supporting specific risk management standards (e.g. [17]). Baskerville’s final generation foresees the need to separate design from its physical realization. This problem is still open, and is the subject of the work described here.

The traceability of security constraints to goals is a feature of goal-based requirements development [18, 19], which has also been tested in industrial settings, and justifies the use of high level goals to provide precise criteria for the completeness and pertinence of the resulting requirements. Goal attributes (rather than separate goals) have been suggested as a means to specify general non-functional requirements [20, 21] including security [22, 23], but a fundamental limitation of the goal based approach is that they do not naturally take account of the system design: the goal structure is a refinement structure independent of system topology.

Other approaches have sought to constrain a functional design by annotating it with security controls. A number of authors have suggested refining security and functionality goals separately, and merging the resulting requirements [24, 25]. Moffett [24] takes this further by suggesting that security controls can be defined as constraints on operationalized requirements, providing a conceptual basis for combining security and functionality requirements.

One approach to the integration of risk with the design process is to express all the associated models in UML; for example, the CORAS project [26, 27] did not introduce any new methods, but proposed process metamodels and threat stereotypes for UML.

The work described in this paper builds on important aspects of these approaches. It provides traceability to goals, but the security controls and system development is carried out in the context of a system design, rather than a goal-trace graph. Security controls are developed in parallel with the system design, and not merged after the fact, ensuring that security and design trade-offs can be made together. Finally, the process described here makes use of standard UML representations for both the system design and the threat environment, but these are views of a single coherent model, allowing automated analysis of the design.
3. BACKGROUND - SECURITY DESIGN ANALYSIS

Important terms used in risk analysis include:

**Asset**: A resource of value to an organization (e.g. hardware, software, data, people).

**Threat (or Concern)**: A potential harm that could occur to an asset. Threats are defined in terms of their **Impact** on an asset (e.g. loss of availability, cost) rather than their cause (e.g. earthquake).

**Vulnerability**: A weakness in a system that allows an attack to realize a threat.

The conventional risk analysis process (see figure 1) contrasts a model of a system’s assets, threats and associated impacts, with possible attackers and means of attack (vulnerabilities). Risk **management** then uses this risk assessment to determine if safeguards are required to reduce identified risks to an acceptable level.

![Figure 1. The Conventional Security Risk Analysis Process](image)

The first step in the process establishes the **system context**, which defines the scope of the analysis. This usually includes people (users, administrators) and physical locations, as well as computer equipment. This step also identifies **assets** and **actors** (people who interact with the system) and documents environmental assumptions.

The next step is asset based threat elicitation, which identifies threats of concern for each asset, and the impact if those threats were realized. Impacts are usually quantified using a coarse linguistic scale (low/medium/high) defined in terms of business consequences, and define one axis of the risk matrix.

The other axis of the risk matrix is the likelihood that a threat will be realized. This is the product of two factors, which are evaluated separately: the frequency of an attack and its chance of success. Identifying potential attackers provides the frequency with which attacks are expected. Vulnerability analysis then identifies paths that attackers can use to realize each threat, and the prospect of success for each path of attack.

This process is followed by risk management. The analysis delivers a risk matrix, where the likelihood and impact of each threat is quantified. Risks that are judged to be unacceptable are resolved by reducing either the likelihood of success (e.g. by adding security mechanisms) or the impact (e.g. by recovery mechanisms). This changes the system, which in turn requires a re-assessment of the risk profile.

The remainder of this section describes how risk analysis has been adapted for use with service-based designs, before their physical realization has been decided.
3.1 Risk Analysis without Deployment Knowledge

A major feature of grid or web-service life cycles is that their deployment is deferred, in some cases to run-time. Standard risk analysis methods include physical as well as logical assets, such as software or data, but the biggest problem in assessing risk prior to deployment is vulnerability analysis. This establishes the mechanisms or paths that can be exploited by an attacker, but the features analysed by this process are implementation specific (e.g. properties of networks, computer platforms, middleware), preventing the direct use of these techniques in the system design phase. It is therefore necessary to find an analogue of vulnerability analysis that can be carried out when services, data items, and their interactions have been defined, but before they are deployed to specific hardware.

3.1.1 Paths of Attack.

Vulnerability analysis regards the attacker as the originator of a tree of actions that may create a path to an asset of concern. These paths are generally blocked by security controls (or operationalized security requirements), which are either standard protection measures (e.g. operating system access controls) or constrained applications. When extra security controls are added in response to a risk, they block a path identified in the analysis.

It is possible to identify similar paths in abstract system designs. For example, if a stakeholder is concerned about the confidentiality of a data item, without any assumptions to the contrary, those concerns flow to any data derived from it, creating a path to anyone who has access to the derived data. Constraints that block such paths amount to security controls that must be implemented in the final system. The pattern of how concerns flow across the system topology depends upon their type (confidentiality, integrity…), but the essence is that it is possible to identify chains of dependency, rooted at assets of concern, and document controls that block these paths.

![Figure 2. Paths in the System Design](image)

Figure 2 illustrates this process with a fragment of a system in which a service consuming a data item produces two others, one of which is consumed by a second service. The services and data items are shown as simple stereotypes and the arrows show data being consumed or generated by each service. Service invocations that cause the movement of data items are omitted for clarity.
In the first diagram (figure 2a) a primary stakeholder concern of confidentiality flows to subsequent data items. In the second (figure 2b) an assumption is made that Service A can be constrained to filter the data to remove confidential content; this is documented (‘filter’) as a constraint and prevents the propagation of the concern to subsequent data.

This process provides a high-level version of the paths that are found in vulnerability analysis, but it is still necessary to make the connection to attackers in order to establish the likelihood of a risk. This requires two things: how attackers can be characterized, and how to link them to asset concerns.

3.1.2 Attackers and Likelihood

The standard risk process identifies classes of attacker in order to distinguish between different frequencies of attack, degree of access to the system, and level of resource or sophistication. The last of these is concerned with the type of vulnerability that an attacker can exploit; for example, bribing an employee is more expensive than external hacking. Since this is implementation dependent, it cannot be factored into a system-level analysis, but the other factors, frequency of attack and type of access, can be used directly.

The usual access distinction is between insiders and outsiders, but service-based systems are distributed across security domains whose organizations may have conflicting security requirements, so it is also necessary to represent different administrations. The primary categories of access are therefore: internal user, administrator, and external attacker. If the system spans several organizations with different security concerns it may be necessary to represent each separately, so a particular analysis may need to include several organizations’ and their internal users.

Deployment puts part of the system into the domain of a particular administrator, so to complete the chain of risk between asset and attacker it is necessary to be able to express constraints on the deployment process. The analysis method divides the system into Deployment Groups (minimal deployable elements), so deployment controls are simply represented as constraints that specify administrations where a service may not be deployed.

By the definition of the level of abstraction of the system description, services are atomic units of software deployment. A deployment group is therefore a service together with the data items it consumes or produces, as shown in figure 3. These groups are straightforward to determine;
services and data items must be chosen at the right level of abstraction, but no further design decisions are needed. It is not necessary to identify which data items are persistent, the fact that they are inputs or outputs to a service mean that they are deployed with that service.

Attackers are classified as belonging to one of three access categories, so there are three ways of constructing paths from assets of concern to attackers:

- **Internal Users.** These are part of the system design, so paths are sought between assets of concern and normal system users, usually grouped into roles.
- **Administrators or Organizations:** Paths flow from assets of concern to other services, and then to the organisations in which those services may be deployed.
- **External:** A path exists from any service or data item to every external attacker, unless constraints are specified for their protection.

Paths to external attackers are similar to those within the system: normal flows are propagated to and from external attackers, unless blocked by explicit constraints. For example, to protect the deployment groups in figure 3 against external attackers, the environment in which group A is situated must protect the confidentiality of its data and the integrity of the service, unlike the components in deployment group B which do not identify a need for protection.

Path analysis can be used in a number of different ways. Section 4 describes the elimination of paths in a design by specifying and testing protection strategies for each attack. This process exposes security design flaws, because such flaws tend to result in inconsistencies between security controls needed to eliminate paths of attack and specified dynamic behaviour, such as operations with no permitted access, or services with no valid deployment location.

Alternatively, this form of analysis can be used to produce a range of metrics for a baseline design, by evaluating the number, risk level, and length, of paths of attack.

### 3.2 The Analysis Model

This section does not describe the theory of the underlying model, but indicates how the theory impacts the practical process by clarifying the nature of paths of attack in information flow terms.

The analysis method maps the system to an information flow graph, which directly supports searching for paths between assets of concern and attackers, in the way outlined above. The vertexes of the graph are data items, and the edges model the behaviour of services and operations. Since the fundamental analysis process is to show the existence of a path, rather than discover all possible paths, the model checking process can be made efficient enough to analyse industrial scale designs.

There is a long history of abstract information-flow security models, but the approach taken in this analysis differs from these by linking information flow and related constraints to asset concerns.

For example, returning to Figure 2, Data Item 1 may be a user account (including name, address etc) and the purpose of Service A may be to provide the date a particular user last accessed the system. The extent that this amounts to an information flow path depends on the threat: this service protects the confidentiality of names and addresses, but not of system use. The analysis model therefore generates a separate graph for each asset concern, ensuring that flow constraints apply to only specific threats.

To follow this example further (assuming that address confidentiality is the concern), the risk analysis identifies a path through Service A, which the analyst blocks by specifying a security flow constraint. This control documents the assumption that the service will protect address information
and, although the design is unchanged, records a significant implementation assumption that may also inform future design iterations.

This illustrates two important features of this framework: it exposes and documents security assumptions that may otherwise have remained in the mind of the original system designer, and it maintains traceability between flow constraints and the threats that they address.

### 3.3 Tool Support

The initial experience report [9] identified the need for a tool to support the analysis; this resulted in the development of the Security Analyst Workbench (SAW), which was used in this study. Although DAME is not a large system the complete model (design, goals, and threats) has 157 objects (classes, objects, operations) with 247 associations. Automation was therefore essential to carry out the analysis rigorously, and to support the analyst in managing the model.

SAW uses a model that is created in a standard UML design tool. Preparation of the design model for analysis is therefore very straightforward – a few stereotypes are added to the existing design to identify security related features, and security goals, concerns, and the threat environment are modelled in UML. The tool interactively supports the method of working reported in section 4: the analyst proposes a protection strategy to deal with an attack, applies security controls to implement the strategy, and then tests it. Rigor is brought to the process by automated analysis, and a type-safe and context-sensitive environment for setting security controls.

Automated analysis also allows more elaborate evaluations to be carried out using the same underlying model; for example, the risk-based valuation of security controls: each control is removed from the system in turn, and the worst-risk path is then found. This provides the analyst with a metric for the value of each control, and identifies over-constrained systems by identifying controls that serve no purpose.

### 3.4 Summary

The SeDAn process is very similar to that shown in figure 1. The difference is that vulnerability analysis is replaced by path analysis, which has three main components:

- Directly evaluating paths to internal Users.
- Evaluating paths to administrators or organizations via deployment groups.
- Evaluating paths to external attackers.

There are two ways to resolve unacceptable risks, the first is to change the system design, and the second is to add policies that specify security controls. These may:

- **Constrain Deployment.** A deployment constraint states that a service must never be deployed to a given administration.
- **Constrain the system topology.** This can be achieved by specifying fine grain access controls.
- **Constrain the behaviour of a service.** This specifies a dataflow constraint, traceable to a specific threat, which blocks a path in the system.
- **Constrain external access to a Deployment Group.** This protects a service against some external behaviour, such as direct access to the system software.

The purpose of these constraints is to act as requirements for implementers, and to document the security intent in the system design; further discussion of implementation issues is beyond the scope of this paper.
4. RISK ANALYSIS IN PRACTICE

This section describes the security-risk analysis of the DAME system. The section is presented in three main parts, the first (4.1) describes the security requirements elicitation process that sets the system context, establishes goals, and documents the threat environment. The second part (4.2) describes the review of function completeness in the light of these requirements, and the final part (4.3) describes the security analysis, following the process outlined in the previous section.

The information in this section serves three purposes:

- To demonstrate the process, and explain the artefacts used and generated.
- To describe the analysis of DAME, and its results.
- To draw general conclusions about the effectiveness of risk analysis for this type of system.

In order to fully demonstrate the process, extracts from the actual analysis artefacts will be presented, within the restrictions of available space. The complete process will be described, as will the more significant results. Each section in turn summarizes the implications for DAME; more general conclusions for SeDAn are summarized in section 5.

4.1 Requirements Elicitation

Requirements Elicitation is the first part of a standard risk analysis (see the introduction to part 3); there are three deliverables:

The System Context, which defines the scope of the system under consideration, and identifies organizations, roles, actors, system assets and boundary dependencies.

Non-functional Goals and Concerns (also called ‘asset analysis’), which identifies the main security goals and the threats or concerns that relate to each asset.

The Threat Environment (also called ‘attacker analysis’), which identifies attacks, their frequency, and associated attackers.

A precursor to establishing the system context is identifying relevant stakeholders who may include customers, system developers, and outsiders such as regulatory authorities.

The system context is a high-level system design, and should be obtained directly from design documentation. However, it must also be at an appropriate level of detail. Too much design information will prevent customers from discussing the system in business terms, so the correct level of detail is one that exposes the internal workings of the system as a business process, ensuring that it is meaningful to all system stakeholders.

An Asset may be any component of a system; in a risk analysis aimed at a business or operational system, assets include a wide range resource types, but since this is concerned with a system design they are principally the services and data types of the system. Asset analysis is a structured elicitation of asset concerns, carried out by identifying and documenting concerns for each asset individually. The completeness and pertinence of these concerns is ensured by tracing concerns to non-functional system goals, which are also elicited as part of the process.

The threat environment identifies the attackers in the system, their type or degree of access and the frequency of attack. The process of establishing the threat environment is a straightforward elicitation, using the non-functional goals as the focus of discussion.

The following sections describe production of each of these deliverables in practice.
4.1.1 The System Context

The DAME system context includes the system use cases, the top-level interaction diagrams and a UML system design. The system design is too large to present here, but the final context used for analysis has 63 UML classes, split into 18 services, 4 clients (user interfaces) and 41 Data types.

The services are divided between three main sub-systems: portal and workflow, distributed search and associated tools, engine simulation and event database. As well as providing complete UML models, the context includes sub-models that correspond to deployment groups (see figure 3): services together with associated data types. These sub-models are useful but not essential for security purposes; they were necessary as a focus for agreement with individual developers, whose perspective tended to be services or sub-systems, rather than the whole system.

Identifying stakeholders raised some unexpected issues. Because the system was in its pilot phase there were a number of stakeholders for the final system that had yet to be identified with individuals or companies. These included aero-fleet operators (customers for aero-engines) and grid-service providers who might host the system. In the absence of the actual stakeholders it was necessary to find proxies to present their viewpoint. It was natural for Roll Royce representatives to represent end-customer views, which were distinguished from the views of Rolls Royce as an engine manufacturer.

Constructing an agreed context should have been the quickest part of this process, but was actually the most time-consuming. There were two sources of difficulty; the first is that the actual design had progressed beyond the top-level business processes and services, meaning that some parts of the top-level view had to be reconstructed from lower-level detail. The second was that the DAME team was concerned that this study should, as far as possible, be valid for the final pilot deliverable, rather than being based on an intermediate snapshot of the system. However, the final deliverable was subject to ongoing design iteration and debate amongst the development teams. The top-level use-cases and business level interaction diagrams remained stable, but the details of services and operations changed significantly during the course of the study.

The stakeholder issues are likely to arise in any design where there is not yet a commitment to a particular deployment. In most systems design stability should be less of a problem, because security analysis would more typically be carried out on a snapshot of an existing design to inform its next iteration.

4.1.2 Non-functional Goals and Concerns

In order to structure the elicitation of asset concerns it was necessary to define the impact of a threat in business terms. A four-point scale was used, although only the top three were documented as concerns:

- **Zero**: may be a nuisance or carry a small cost, but is not significant enough to warrant analysis.
- **Low**: may carry a significant cost, but a number of incidents can be absorbed by the organisation in any year.
- **Medium**: will have a perceptible result on the bottom line of the organisation, but no serious long-term effect in subsequent years.
- **High**: will prejudice part of the business, sufficient to impact the results of the business over a long period.

Keywords used to prompt the elicitation were compiled from an initial brainstorming with customers and extra words were added if they were suggested as a concern against any particular asset. The list included: confidentiality, integrity, availability, reliability, privacy, completeness,
provenance, and non-repudiation. These were later rationalized into a small number that applied to all assets of a given type (e.g. availability of services) and those that were particular to services. In later reviews some of these categories were dropped, but the primary elicitation simply recorded stakeholders’ views, rather than trying to rationalize them.

Table 1. Typical Asset Concerns

<table>
<thead>
<tr>
<th>Data Assets</th>
<th>Confidentiality</th>
<th>Integrity</th>
<th>Provenance</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2 Engine Data Record Performance. (These concerns may change if the data are deployed outside DAME)</td>
<td>RR / DS&amp;S Could divulge proprietary engine information to 3rd party</td>
<td>RR / DS&amp;S. Need to protect accuracy of reference data</td>
<td>RR / DS&amp;S. Protect the reference source of decision data</td>
<td>Notes</td>
</tr>
<tr>
<td>I Engine Performance</td>
<td>III Reliability (L)</td>
<td>IV Record decision basis</td>
<td>Goal</td>
<td></td>
</tr>
<tr>
<td>C.A Unauthorised Access</td>
<td>LA Loss or Corruption</td>
<td>P.B Source of Reference Data</td>
<td>Concern</td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
<td>Impact</td>
<td></td>
</tr>
</tbody>
</table>

Table 1 shows a typical entry in the asset register. In this case only the keywords confidentiality, integrity and provenance apply. Each has a different concern, and their interpretation is documented in a separate table, which is referenced by the Concern entry. The Notes entries identify the stakeholder and motivate the concern, and each is traceable to a business goal for the system.

Six goals were identified for DAME, table 2 provides an abbreviated list of goals, and table 3 shows how these are documented.

Table 2. DAME goals

<table>
<thead>
<tr>
<th>Number</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>To maintain the Confidentiality of Detailed Engine Design and Performance Data</td>
</tr>
<tr>
<td>II</td>
<td>To maintain the Confidentiality of Operational Data</td>
</tr>
<tr>
<td>III</td>
<td>To ensure that any Diagnostic advice provided by the system is Reliable</td>
</tr>
<tr>
<td>IV</td>
<td>To record the provenance of diagnostic decisions and identify individuals’ actions in the diagnostic process</td>
</tr>
<tr>
<td>V</td>
<td>To provide predictable availability.</td>
</tr>
<tr>
<td>VI</td>
<td>To protect the confidentiality of technical industrial property used in the system’s implementation</td>
</tr>
</tbody>
</table>

Table 3. Typical DAME goal

<table>
<thead>
<tr>
<th>Number</th>
<th>IV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>To record the provenance of diagnostic decisions and identify individuals’ actions in the diagnostic process</td>
</tr>
<tr>
<td>Owner</td>
<td>Rolls-Royce and Data Systems &amp; Solutions</td>
</tr>
<tr>
<td>Impact</td>
<td>Medium</td>
</tr>
<tr>
<td>Description</td>
<td>The process by which diagnostic decisions are made must be recorded with sufficient quality to allow the investigation of problems or marginal decisions after the fact. Individuals that contribute to the diagnostic workflow must be accountable for their contribution to the process.</td>
</tr>
</tbody>
</table>

The DAME goals are well suited to their function because they define threats, objects of protection and motivation, but as a consequence they are difficult to elicit in abstract.

The problem with developing goals in abstract is that it is difficult to establish an appropriate level of detail; on the other hand it is relatively easy to elicit requirements in terms of specific business assets, because customers are comfortable thinking in terms of assets and concerns. The strategy used in DAME was to establish initial asset concerns, and then cluster these concrete examples into putative goals, which were then revised by stakeholders. The asset concerns were
then checked for completeness in the light of the goals, and pertinence was established for each concern by documenting its traceability to a goal.

Context elicitation and asset analysis naturally overlapped, and iteration between the two was necessary: additional assets discovered during asset analysis required a revision of the system context. Goal discovery was also an iterative process and proved a useful check of the correctness and consistency of asset concerns, as well as providing customers with confidence that they were related to business goals.

Asset concerns were finally rationalized by collecting together common requirements and removing keywords that were no longer needed. The keywords were good prompts for discussion, but not specific enough for requirements purposes; that job was fulfilled by the specific concerns attached to each asset. One effect of eliciting the keywords, rather than working from a standard list, was that non-functional requirements emerged with a wider scope than security: the goals included both reliability and availability.

During the course of this structured elicitation, it became evident that some of the non-functional goals were not consistent with existing functional requirements. The most significant incompatibility was the definition of provenance, which stakeholders extended to persistent reference data, but behavioural requirements had previously only applied to user interactions.

Threat elicitation was straightforward, and was helped by customers who were able to express their concerns in business, rather than implementation, terms. One technical issue that did arise was that real business concerns are not necessarily static; the impact to a customer of some threats depended on the business cycle – for example the progress of a new contract. These issues were simply recorded, but they may complicate the later lifecycle of the system.

### 4.1.3 The Threat Environment

The threat environment characterizes each threat by access type, attacker, target goal and frequency. Attackers are grouped by access type into five main categories:

- **Administrator**: organizations with administrative rights over parts of the system.
- **Legitimate Users**: users with legitimate DAME roles.
- **Other Users**: employees of companies that use DAME, but who do not have DAME roles.
- **External with Significant Resources**: for example, journalists or competitors.
- **External with limited resources**: for example, hackers.

Attack frequency is quantified on a four-point scale, as follows:

- **High**: Frequent attacks: many per day, to at least one per month.
- **Medium**: An attack is likely, but infrequent, 0.1 to 10 per year.
- **Low**: An attack is unlikely (1 per decade).
- **Unlikely**: Not zero probability, but very unlikely.

A scale of four was used because it relates well to the business context of the system. High and Medium attacks are respectively equivalent to many, or at least one, attack in any financial year. Low and Unlikely levels relate to the lifetime of the system, they indicate that an attack is likely, or not, during the system’s operational life.

<table>
<thead>
<tr>
<th>Access Type</th>
<th>Attacker</th>
<th>Goal</th>
<th>Frequency</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legitimate Users</td>
<td>Domain Expert</td>
<td>IV</td>
<td>Low</td>
<td>Users may seek to change or remove records of inappropriate decisions or actions</td>
</tr>
<tr>
<td></td>
<td>Maintenance Analyst</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Maintenance Engineer</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A typical threat entry is shown in table 4. This record is a potential attack on the goal shown at table 3. It relates to several possible attackers, showing how the documentation clusters attackers and goals where necessary to produce a compact representation.

The threat environment was initially generated from asset concerns, and from the roles and organizations identified by the system context. Elicitation with stakeholders used this base material as a focus. In the event this process did not result in iteration of the goals or concerns, although it did review that content from a different perspective. An important contribution of stakeholders was to assess attack frequency from their own experience; for example, the companies concerned had strong views about the possibilities of their mounting an attack on their competitors (it would not be considered an option), and thus regarded an attack from their competitors as equally unlikely.

4.1.4 Summary

This requirements elicitation followed a relatively standard pattern for risk analysis, but because the study was concerned with a system design, the assets identified were limited to design components, such as data and services.

The primary difficulty was in obtaining a suitable design, at a level of abstraction at which its components could be understood as business assets. This highlights the need to more closely integrate the security analysis and system design process: this elicitation was carried out after the fact, it would have been much more straightforward to carry it out as part of the design process.

The process of eliciting goals, concerns and threats proved straightforward, and was well assisted by customers who were able to express their concerns in business, rather than implementation, terms. This process also highlighted gaps in the existing behavioural requirements of the system, and generated other non-functional requirements such as reliability and availability. Although this elicitation was motivated by security, it produced useful perspectives on other functional and non-functional requirements; it is therefore important that these other issues are allowed to emerge, and are not stifled by the early application of pre-defined security checklists.

4.2 Review of Non-functional goals for Additional System Behaviour

Primitive security constraints are concerned with the loss of confidentiality, or integrity; what these constitute depends on the user requirement, rather than any abstract notion of information flow (see 3.2). The security analysis deals directly with these issues; however, there is a gap between stakeholders’ security requirements and security expressed as a series of controls (constraints on the system). For example, privacy concerns must be interpreted in the light of data protection legislation which requires the confidentiality of personal data, but also adds behavioural requirements to the system: subjects have the right to check and correct their data.

Each security goal may therefore give rise to further functional requirements, which may include new assets and services, and these in turn may have confidentiality or integrity concerns. Before security analysis begins, it is therefore necessary to ensure that all the behaviour implied by the stakeholders’ security goals is present in the system.

4.2.1 Goal review

Each security goal was reviewed in turn, and two were found to have functional implications. One of these (Goal V: Availability) usually needs a range of functions, including intrusion detection, auditing, backup and system recovery. None of these are in the present design, but it was
recommended that they should be recorded as requirements, even if they do not necessarily need to be shown in a business-focused top-level design model.

The second problem (Goal IV: Provenance) is more fundamental. The goal review exposed coverage problems with the existing provenance architecture, in addition to the issues noted above (see 4.1.2). The system is designed to record the process that gives rise to a diagnosis by recording the actions of the workflow system. However, the assets subject to provenance concerns include the results generated by analysis tools, and these can also be accessed directly by users without using the workflow system: there are three user interfaces that do not route their interactions via the workflow, and between them they are able to access much of the system functionality.

This problem has probably developed as a result of well-meaning design iterations in different parts of the project, but constitutes a major architectural flaw; as it stands the design is unable to meet this goal.

4.2.2 Summary

User specified security goals frequently require behaviour as well as security controls. Before security analysis can begin it is important to ensure that the functional behaviour of the system is complete. A review of DAME in this respect exposed two issues. The first (availability) requires a range of supporting functions that must be present in an implementation, but may not be necessary in a high level design. The second exposed a coverage failure of one of the system’s most important non-functional goals.

This has probably occurred because of inappropriate design iteration, and demonstrates the need to continuously review security in an iterative design process.

4.3 Security Analysis

Security risk is the combination of the likelihood of an attack and its impact. Since this analysis is concerned with an abstract design it is not able to assess the exploitability of a given path, just whether a path is present or not. The risk metric used in this analysis is therefore, strictly, potential risk: the risk that would result from an easily exploitable path.

The combination of frequency (see 4.1.3) and impact (see 4.1.2) are mapped to high, medium or low potential risks, as shown in table 5.

<table>
<thead>
<tr>
<th>Impact</th>
<th>Frequency</th>
<th>Unlikely</th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>High</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This mapping is best understood in business terms: High risks indicate that deploying the system without security will result in long-term financial impact, Medium risks indicate that deploying the system will result in a bottom-line financial impact in one of the years that it is operational, and Low risks indicate that an attack is either unlikely, or can be absorbed financially.

DAME has 13 possible attacks and 32 asset concerns; however, there are only 26 different Potential Risks, being distinct valid combinations of asset concerns and attacks. Of these 5 are high risk, 6 medium and 15 low. The attacks from legitimate users have the potential for high impact, so
even relatively infrequent attacks are dangerous in business terms. External attacks are the next biggest threat, because of their likelihood rather than their impact. There are no high risk attacks from administrators in this system; despite their potential impact they are less likely than an attack from an internal user.

In an operational system it would be possible to mitigate risks outside the system, for example by procedure or insurance; however, the purpose of this study is to investigate an abstract design, so the object is to discover if it is possible, or not, to counter a risk with security controls. If it is not possible, then this may indicate an inherent design defect.

There are two dimensions along which the analysis can proceed: risk level or type of access. Paths to users tend to have most impact on the system design, and there is more scope to tailor the system to mitigate attacks by administrators than external attackers, because the system can constrain deployment to particular administrations. The analysis strategy was therefore to consider all user and administrator attacks at each risk level, and then consider external attacks separately.

The following sections follow this order by presenting the results of analysing high, medium and low potential risks from both users and administrators, and then external attacks.

4.3.1 High Risk

This section will describe the artefacts used in each step of the analysis to provide an overview of the process in practice, subsequent sections will simply describe the most significant conclusions.

Each analysis is structured in the same way. Combinations of concerns and attacks (potential risks) that give rise to paths of attack are identified, and a protection strategy is proposed, which includes the policy required to mitigate the risk. This policy is then checked using the Security Analyst Workbench, to demonstrate that it is effective and not over-specified. In normal circumstances the policy and controls remain as part of the system documentation for subsequent design cycles, and for the guidance of implementers; because this is a design review, we are more concerned with the cases where this process fails in some way.

Potential risks can either be determined by the Security Analyst Workbench, or by inspection from the UML form of the goal and threat environment. In practice the analysis tool is used to list all potential risks, and the UML visualization helps understand their significance.

There are no potential risks from administrators at this level, but there are two from users, given in tables 6 and 7.

<table>
<thead>
<tr>
<th>Table 6. Potential Risk 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goal of attack</strong></td>
</tr>
<tr>
<td><strong>Risk/Impact</strong></td>
</tr>
<tr>
<td><strong>Assets</strong></td>
</tr>
<tr>
<td><strong>Notes</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 7. Potential Risk 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goal of attack</strong></td>
</tr>
<tr>
<td><strong>Risk/Impact</strong></td>
</tr>
<tr>
<td><strong>Assets</strong></td>
</tr>
<tr>
<td><strong>Notes</strong></td>
</tr>
</tbody>
</table>

Since the same concern may apply to a range of assets, it is often more practical to group the potential risks, rather than deal with each individually. This is the case in table 6, where several assets share the same concern.
The protection strategies for each of these risks are quite straightforward, the first requires that normal users of the system are not able to modify the assets that are the basis of the authentication and authorization system, and that the integrity of these assets is protected inside the system. The specific policy is:

- Normal users of the system must not be allowed to invoke administrative operations.
- Other Services in the system must not be allowed to invoke administrative operations.
- The Portal and MyProxy services must protect the integrity of the User, Role and accessPermittedPage data items, to ensure that only legitimate security administrators modify them, and that services are provided with authentic data.

The protection strategy for the second potential risk is even simpler: non-DAME users must not be allowed access to the system. This would seem quite obvious, but one of the benefits of a structured approach to analysis is that it records such information as part of the system documentation.

These policies were formulated using the analysis tool, which was then used to check their effectiveness. The correspondence between the English policies and the predicates used in the tool is quite close, tables 8 and 9 show the tool versions of the complete set of policies at this risk level.

<table>
<thead>
<tr>
<th>Service or Client</th>
<th>Policy</th>
<th>From</th>
<th>To Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portal</td>
<td>No Access</td>
<td>Analyst</td>
<td>administerUser</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Engineer</td>
<td>administerUser</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Expert</td>
<td>administratorUser</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NonDameUser</td>
<td>8 operations, omitted for space</td>
</tr>
<tr>
<td>EngineGUI</td>
<td>No Access</td>
<td>NonDameUser</td>
<td>2 operations, omitted for space</td>
</tr>
<tr>
<td>EnginePerformanceVisualiser</td>
<td>No Access</td>
<td>NonDameUser</td>
<td>selectDataToView</td>
</tr>
<tr>
<td>SignalDataExplorer</td>
<td>No Access</td>
<td>NonDameUser</td>
<td>3 operations, omitted for space</td>
</tr>
</tbody>
</table>

Access control policies block access from a user or another service to an operation within a service. The tool assumes that any association between services allows access to all the operations of that service, unless specified otherwise. The underlying predicates are in the form of access permissions rather than denials, as they would be in a system; however, this presentation of policies as exceptions to the design model is more compact and informative.

<table>
<thead>
<tr>
<th>Service or Client</th>
<th>Policy</th>
<th>Concern</th>
<th>Data Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portal</td>
<td>TypeRestrictedFlow</td>
<td>IV_High_Integrity</td>
<td>User</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IV_High_Integrity</td>
<td>AccessPermittedPage</td>
</tr>
<tr>
<td>MyProxy</td>
<td>TypeRestrictedFlow</td>
<td>IV_High_Integrity</td>
<td>User</td>
</tr>
</tbody>
</table>

The Security Analyst Workbench (SAW) supports several types of data flow constraint; restrictions that constrain a single primitive behaviour of a service are available, but many primitive restrictions are often necessary to implement a single informal policy requirement. To better match the analyst’s policies, SAW provides a number of flow-restriction patterns, and Type Restriction is one of these. An Integrity Type Restriction specifies that only data items of the same type can change the value of the protected objects. This is quite a common control requirement in simple file systems, where a store operation updates persistent data items, and query operations return the same types unchanged. The concerns in table 9 are references to the goal, impact and other related information about the threat to which this constraint applies.

These policies demonstrate that DAME is relatively easy to protect against high-level risks, since the required policies are mostly simple access controls at the system boundary.
The Portal sub-system specifies authorization and authentication functions and associated data types, which have to be protected. However, this design detail is inconsistent across the system, since these data are not propagated to the other user interfaces (EngineGUI etc). The analysis shows that to protect against risks at this level, all interfaces must prevent access from non-DAME users. This highlights the need for consistent treatment of security infrastructure in the system under analysis; in this case the inconsistency was highlighted, and the need for all user interfaces to be equally protected was stressed.

4.3.2 Medium Risk

Two of the three potential risks at this risk level can be straightforwardly mitigated with technical controls. The first, another attack on diagnostic provenance, places requirements on the integrity of source sensor data and on its subsequent storage, and also places similar requirements on persistent workflow records.

The second potential risk, on the Confidentiality of Engine Simulation Algorithms, is more interesting from the grid perspective because the potential attacker is one of the fleet operators. The most straightforward protection strategy is to restrict the deployment of the engine simulation sub-system to either Rolls Royce or Data Systems and Solutions. This does not imply that other parts of DAME cannot be deployed to a grid, but it does illustrate that grid based systems may have components that need to be more restricted.

The final potential risk at this level, the possibility that organizations may attempt to modify the provenance record by manipulating user authentication or authorization data (i.e. by faking user roles), is much harder to counter. There are two possible solutions, the first is to constrain the deployment of the portal and associated user authentication to a trusted third party, the other is to distribute authentication and authorization functions between the principal organizations in such a way that the identity of the organization is bound to the role of its users. The trusted third party approach may be valid in circumstances where a ‘virtual organization’ is tangible and is capable of administering the system, but this will not apply to DAME: the system is likely to be either fully distributed or administered by one of the main customers. Centralized authentication is therefore inappropriate; a more effective design would make individual organisations responsible for the authentication of their own users, in such a way that a user claiming a role would also identify the authenticating organization.

This issue is significant for grid engineering. Usually the need to identify organizations as well as roles is assessed on the basis of access need: does the organization, or other attribute, feature in the access requirement as well as the current role. This highlights another dimension to this problem: organisations as potential attackers and their ability to manipulate authentication systems deployed within their domain.

4.3.3 Low Risk

Unlike the high or medium risk levels, it is much more likely that an organisation will accept a system with unmitigated low risks, taking the view that the operational value of the system outweighs the possible damage. There are five potential risks at this level, three do not raise major issues, but analysis of the other two raises significant design questions.

The first design issue is exposed by another attack on diagnostic provenance, in this instance from one of the principal customer organisations seeking to misrepresent its own provenance records. Since the attack is from one of the principal organisations, and it targets their own data,
then the only options are the deployment of the workflow system to an independent trusted third party, or an alternative design approach.

The workflow system records an exchange that may become the subject of contractual dispute or investigation. Normal business practice for such documentation is well established – both parties need separate, equally valid, verifiable evidence of the nature of a contract and its execution. Unless a trusted third party is employed, business workflows between collaborating companies with single point records will always be vulnerable to an attack by the organisation that holds the record, and this is the situation here.

Although this is a low risk item, the vulnerability introduced by this workflow design is sufficient to prevent the evolution of the business role taken by this system, and that is an important reason to reconsider the advisability of a centralized design at this stage in its lifecycle.

The second design issue also relates to an attack from an organisation, this time directed at discovering confidential engine design information for commercial advantage. The asset of concern is Engine Metadata, and the attacker is a fleet operator. It is possible to constrain the deployment of services to operators, but much of the system uses Engine Metadata, so this strategy would effectively prevent the deployment of the system to a grid: in DAME the grid enables the processing of high volumes of data where it is collected, which is at locations managed by fleet operators.

The scope of this problem can be reduced; rationalization of the use of engine and case references, metadata, and metadata encapsulated in other data records, would probably result in a reduction in the number of assets of concern. On balance, however, this risk can be reduced but not eliminated, and stakeholders will have to consider accepting a low risk of attack in return for the value of distributed processing.

It is notable that both these issues arise from attacks from participating organisations, or their system administrators, rather than from users. Designers are probably accustomed to thinking in terms of user access control, but the problem of system deployment to one or more administrations has received little practical exposure.

4.3.4 External Risk

External attacks are more frequent than attacks from legitimate users or from organisations that host systems, so if there is a path of attack to an asset, the risk level is often higher. This is offset by the fact that most systems can police their boundaries fairly effectively, so the important question is the degree of boundary protection required. This analysis considered external attacks at only the high and medium levels, because of the outstanding system design issues at the low-risk level.

There are three high or medium-level risk external attacks: the confidentiality of engine design (Goal I) data from a competitor, the confidentiality of operational performance (Goal II) from investigative journalists, and reliability and availability (Goals III/V) against hackers.

Security controls change the way that data flows in the system, and therefore the degree to which elements of the system are vulnerable; it is therefore possible to reduce the overall risk profile by design, before countering residual risks with boundary protection.

The flow of engine design information is constrained by the protection strategies already proposed, and there is little that can be done to reduce the system profile to hackers (their target is so broad). There are four assets to which the attack on Goal II applies, and it is straightforward to limit their distribution with a protection strategy that records existing assumptions about how they are used. The profile of external attacks is given in table 10, after applying this strategy.
The first group of services is associated with data searching and reporting, the second with workflow, and the third is concerned with engine simulation and analysis. Restricting the exposure of the first group to external attacks is a positive result for grid computing, since these services are most likely those that will be deployed to a grid. The second group is where many of the design issues are centred, and the third group will, in any case, have restricted deployment (see 4.3.2).

Table 10. External Attack profile: medium and high potential risks

<table>
<thead>
<tr>
<th>Service</th>
<th>Attacker Type</th>
<th>Goal</th>
<th>Potential Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hacker</td>
<td>III/V</td>
<td>High</td>
</tr>
<tr>
<td>CBRAnalyser-G, CBRWorkflowAdvisor-G, DatabaseMiner-G, Portal, SDM-G, WorkflowManager</td>
<td>Competitor</td>
<td>II</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>Hacker</td>
<td>III/V</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>Journalist</td>
<td>II</td>
<td>High</td>
</tr>
<tr>
<td>EngineGUI, EnginePerformanceDatabase-G, EnginePerformanceVisualiser, EngineSimulation-G</td>
<td>Competitor</td>
<td>II</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>Competitor</td>
<td>I</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>Hacker</td>
<td>III/V</td>
<td>High</td>
</tr>
</tbody>
</table>

4.3.5 Summary

From the perspective of DAME, this analysis has found some important issues that need to be investigated and resolved. However, viewed as an overall system, rather than a specific design, the analysis also provides good news: the most significant high-risk attacks can be dealt with by straightforward protection strategies at the system boundary, and the flow of data within the system can be constrained in such a way that the most sensitive information does not need to be present in the part of the system best suited to grid deployment. On the other hand, there are parts of the system that are ill-suited to the grid because they need to be protected against inappropriate administrative access, and the stakeholders will probably have to accept residual low-level risks as the price for distributed processing.

The prevalence of administrative attacks among the risks that expose design flaws is significant. The centralized design of the authentication and provenance sub-systems allows a host administration to attack them; however, in both cases distributed solutions are possible and desirable. It seems likely that designers are accustomed to considering access management from system users, but not the accesses that might result from different deployment strategies.

Finally, the problem that arose in the previous section (4.2), of security inconsistencies between the various user interfaces, was also highlighted in the analysis. In this case we noted an inconsistency rather than a design flaw, but its presence here underlines the danger of iterating a design without an associated security analysis.

5. PROCESS EXPERIENCE

Specific results from the analysis of DAME are recorded above (see summaries in Section 4); this section presents more general conclusions about the analysis process.

The SeDAn framework is now in its second generation of practice, and includes tool support; in consequence it was straightforward to apply. The DAME study also demonstrated the value of the framework, by demonstrating that it is effective in finding design flaws in real systems.
The part of the process that took an unexpectedly long elapsed time was the development of the system context, but this was related to project circumstances. In general, the whole process is relatively lightweight, taking a few man-months. The elicitation elements needed several cycles of stakeholder discussion and review, which increased the elapsed time. However, if the process was integrated in a design iteration cycle, then much of the elicitation would be a once-off cost.

The level of detail or abstraction of the various artefacts must be carefully controlled if the process is to be effective. The system context needed to be relevant to business stakeholders to allow asset-based elicitation of threats - the system model had to be understood as a functional business process. On the other hand, the goals are more detailed than are often reported; we conclude that words such as ‘secure’ or ‘confidential’ may be useful keywords for prompting elicitation, but are not sufficiently specific to be useful goals.

The goal and threat elicitations also identified gaps in the existing behavioural specification; DAME is probably typical in using use-case oriented functional elicitation, so asset-driven elicitation motivated by security provides a different perspective on the system, and contributes to functional as well as non-functional requirements. Non-functional requirements other than security were also generated, justifying an open elicitation style, rather than one that is conditioned by pre-determined security checklists.

The review of security goals, to ensure that the system is functionally complete before security analysis, is a step that is rarely made explicit by system analysts. Security goals give rise to asset concerns, but may also require additional system behaviour. This experience demonstrates the scope for design flaws in which functional requirements derived from security goals are inconsistent with previously established behavioural specifications.

The approach to security analysis can vary from attempting to specify and test the whole system protection policy at once, to setting and testing individual constraints. The middle ground presented here has proved to be flexible and modular: protection strategies and informal policies are specified for groups of attacks, working from the highest risks down. Tool support introduced an important element of rigor into the process: some of the issues reported here were not found by informal analysis, and the level of risk associated with each design defect can now be quantified.

From the project perspective, the most important lesson is that security analysis must be integrated in a timely way with the design cycle. The asset-based requirements elicitation impacts functional as well as non-functional requirements, and some of the security defects reported have probably developed because design iterations moved the system away from its original security architecture.

6. CONCLUSIONS

This paper describes the application of the SeDAn (Security Design Analysis) framework to a significant grid-based application (DAME). The risk-analysis process was successful in identifying design flaws, and expressing their potential impact in terms of business risk. Specific DAME conclusions are outlined in the sectional summaries of part 4.

As noted in the introduction, these flaws do not imply criticism of the project or any of its partners, all of whom welcomed and helped with this study; problems are simply to be expected in the design of large distributed systems, and exposing design defects both helps the development of DAME, and validates the effectiveness of SeDAn. However, the number of issues discovered, and the fact that some were introduced by design iteration, underlines the need for security risk-analysis to be an integral part of the requirements capture and design iteration of large distributed systems.
The process experience is summarized in part 5; the asset-based elicitation exposed missing functional requirements, and each stage in the analysis highlighted different design issues. The study was greatly helped by business-focused customers, and an open approach to eliciting their requirements.

Several of the design flaws expose the system to attack from organisations that host system services. In some cases it is possible to partition the system and restrict deployment of critical elements, but in others the design inappropriately centralizes data (authentication, workflow provenance records) allowing whichever administration hosts that component to attack them. Distributed solutions to these problems are both possible and desirable; it seems probable that the higher prevalence of issues related to administrative attacks, compared to attacks from users, may indicate that that designers are accustomed to considering access management from system users, but not the accesses that result from different deployment strategies.

From the grid engineering perspective, it is possible to constrain the flow of data in DAME so that the part of the system best suited to grid deployment does not need the most sensitive information. On the other hand there are parts of this system that are ill-suited to the grid because their deployment must be restricted, and the analysis shows that stakeholders will have accept residual low-level risks as the price for distributed processing.

This is the second-generation application of this framework, which now includes the interactive Security Analyst Workbench. Applying the lessons learned in previous work has made the analysis smoother and more rigorous. There is scope to look further at the question of deployment, and how developers manage the accesses that result, and to test the framework on other types of system.

ACKNOWLEDGEMENTS

The work reported in this paper was supported the Royal Academy of Engineering and by the DAME project under UK Engineering and Physical Sciences Research Council Grant GR/R67668/01. We are also grateful for the support of the DAME partners, including the help of staff at Rolls Royce, Data Systems & Solutions, Cybula, and the Universities of York, Leeds, Sheffield and Oxford.

REFERENCES