



# A Survey of Techniques for Security Architecture Analysis

*Brendan Lawlor and Linh Vu*

**Information Networks Division**  
Information Sciences Laboratory

DSTO-TR-1438

## **ABSTRACT**

This technical report is a survey of existing techniques which could potentially be used in the analysis of security architectures. The report has been structured to section the analysis process over three phases: the capture of a specific architecture in a suitable representation, discovering attacks on the captured architecture, and then assessing and comparing different security architectures. Each technique presented in this report has been recognised as being potentially useful for one phase of the analysis.

By presenting a set of potentially useful techniques, it is hoped that designers and decision-makers involved in the development and maintenance of security architectures will be able to develop a more complete, justified and usable methodology other than those currently being used to perform analyses.

## **RELEASE LIMITATION**

*Approved for public release*

*Published by*

*DSTO Information Sciences Laboratory  
PO Box 1500  
Edinburgh South Australia 5111 Australia*

*Telephone: (08) 8259 5555  
Fax: (08) 8259 6567*

*© Commonwealth of Australia 2003  
AR- 012-778  
May 2003*

**APPROVED FOR PUBLIC RELEASE**

# A Survey of Techniques for Security Architecture Analysis

## Executive Summary

In today's increasingly hostile environment, security, and in particular information security, is a vital issue requiring significant attention. As Australia increases its involvement in overseas operations, protecting the Australian Defence Force's (ADF) classified information will be a foremost concern due to the increasing threat levels. Overseas adversaries are not the only problem; attacks from within our own country also need to be considered, along with the vulnerabilities in the resources employed by the ADF. Thus protecting the confidentiality, integrity and availability of Defence information is becoming increasingly difficult and a methodical approach to security is required.

Security architectures provide this disciplined approach to developing security solutions. A security architecture is a high level design identifying and describing all the components used to satisfy a system's security requirements. These security requirements are generally defined in an organisation's security policy, which is a top level specification describing the rules and procedures for using information. There are a number of documents describing the Defence Security Policy, and the Commonwealth policy in general, including the Protective Security Manual (PSM), Australian Communications-Electronic Security Instruction 33 (ACSI33) and the Security Manual (SECMAN) series. When designing systems, it is important that a methodical approach is taken to ensure the security policy is adhered to and that appropriate measures are put in place to counter threats. This is the role of security architectures.

Designing security architectures is a distinct process based on an assessment of the threats to the specific system being implemented. It is therefore difficult to provide support for designing security architectures. However, an area where support is required is security architecture analysis. Currently, there are no defined systematic methods or tools used in the Department of Defence for capturing and, perhaps more significantly, analysing and comparing security architectures.

This technical report is a survey of existing techniques which could potentially be used in the analysis of security architectures. The report has been structured to section the analysis process over three broad phases: the capture of a specific architecture in a suitable representation, discovering attacks on the captured architecture, and then assessing and comparing different security architectures. Each technique presented in this report has been recognised as being potentially useful for one phase of the analysis.

Whilst this report does not specifically outline the best approach to security architecture analysis, it does make a number of recommendations about the most useful techniques for capturing security architectures, discovering attacks on the captured architecture, and assessing and comparing different security architectures. It also proposes the way forward for future research.

# Authors

## **Brendan Lawlor**

### **Information Networks Division**

*Brendan Lawlor is a Professional Officer in the Trusted Computer Systems Group in Information Networks Division, and has been working at DSTO since the start of 2001. His research interests include information security, security architectures, Bayesian Networks and formal methods. Brendan has a Bachelor of Science from the University of Adelaide, with a major in Computer Science and Pure Mathematics. He is currently studying for a Masters in Computer Science at the University of South Australia.*

---

## **Linh Vu**

### **Information Networks Division**

*Linh Vu is a Professional Officer in the Trusted Computer Systems Group in Information Networks Division, working at DSTO since January 2002. She has been working on security architectures and information security devices. She has a Bachelor of Science (Mathematical and Computer Sciences) and a Bachelor of Engineering (Computer Systems) from the University of Adelaide.*

---

# Contents

<b>ACRONYMS</b>	
<b>1. INTRODUCTION.....</b>	<b>1</b>
<b>2. ARCHITECTURE CAPTURE .....</b>	<b>3</b>
<b>2.1 Domain Approach.....</b>	<b>3</b>
2.1.1 Infosec Business Model.....	3
2.1.2 Infosec Infrastructure Model.....	4
2.1.3 Infosec Architecture Model .....	5
2.1.4 Application to Architecture Capture .....	6
2.1.5 Discussion .....	6
<b>2.2 Defence Architecture Framework .....</b>	<b>7</b>
2.2.1 Developing Specific Architecture Descriptions.....	9
2.2.2 Application to Architecture Capture .....	10
2.2.3 Discussion .....	10
<b>2.3 Common Criteria Protection Profiles .....</b>	<b>11</b>
2.3.1 Examples .....	12
2.3.2 Application to Architecture Capture .....	12
2.3.3 Discussion .....	13
<b>3. DISCOVERING ATTACKS .....</b>	<b>14</b>
<b>3.1 Threat Database .....</b>	<b>14</b>
3.1.1 Application to Discovering Attacks .....	15
3.1.2 Discussion .....	15
<b>3.2 Attack Trees.....</b>	<b>16</b>
3.2.1 Application to Discovering Attacks .....	18
3.2.2 Discussion .....	18
<b>3.3 Security Protocol Analysis.....</b>	<b>18</b>
3.3.1 Application to Discovering Attacks .....	20
3.3.2 Discussion .....	20
<b>3.4 Security Failure Analysis.....</b>	<b>21</b>
3.4.1 Application to Discovering Attacks .....	24
3.4.2 Discussion .....	24
<b>4. COMPARISON AND ASSESSMENT .....</b>	<b>26</b>
<b>4.1 Bayesian Networks .....</b>	<b>26</b>
4.1.1 Application to Comparison and Assessment .....	28
4.1.2 Discussion .....	28
<b>4.2 Simulation .....</b>	<b>28</b>
4.2.1 Simulation Efforts in Information Security .....	30
4.2.2 Application to Comparison and Assessment .....	31
4.2.3 Discussion .....	31
<b>4.3 Risk Analysis .....</b>	<b>32</b>

4.3.1	Security Risk Analysis.....	33
4.3.2	Application to Comparison and Assessment .....	33
4.3.3	Discussion .....	34
<b>4.4</b>	<b>IATF Approaches .....</b>	<b>34</b>
4.4.1	Robustness Strategy.....	35
4.4.2	Application to Comparison and Assessment .....	37
4.4.3	Discussion .....	37
<b>4.5</b>	<b>Game Theory.....</b>	<b>38</b>
4.5.1	Game Model Representations .....	39
4.5.2	Advanced Game Models .....	40
4.5.3	Application to Comparison and Assessment .....	41
4.5.4	Discussion .....	41
<b>4.6</b>	<b>Survivability Analysis.....</b>	<b>43</b>
4.6.1	Survivable Systems Analysis Method.....	44
4.6.2	Formal Approach.....	46
4.6.3	Application to Comparison and Assessment .....	47
4.6.4	Discussion .....	47
<b>4.7</b>	<b>Economic Models .....</b>	<b>47</b>
4.7.1	Application to Comparison and Assessment .....	50
4.7.2	Discussion .....	51
<b>5.</b>	<b>CONCLUSION.....</b>	<b>52</b>
<b>6.</b>	<b>REFERENCES.....</b>	<b>54</b>

## Acronyms

ADF	Australian Defence Force
ALE	Annual Loss Expectancy
C4ISR	Command, Control, Communications, Intelligence, Surveillance and Reconnaissance
CC	Common Criteria
CCA	Cause-Consequence Analysis
CIO	Chief Information Officer
CPT	Conditional Probability Table
CV	Common View
DAF	Defence Architecture Framework
DIE	Defence Information Environment
DoD	Department of Defence
EAL	Evaluation Assurance Level
ETA	Event Tree Analysis
FMEA	Failure Modes and Effects Analysis
FPG	Failure Propagation Graph
FTA	Fault Tree Analysis
HAZOP	Hazard and Operability studies
IATF	Information Assurance Technical Framework
InfoSec	Information Security
IT	Information Technology
OV	Operational View
PP	Protection Profile
SML	Strength of Mechanism Level
SSA	Survivable Systems Analysis
SV	Systems View
TOE	Target of Evaluation
TV	Technical View



# 1. Introduction

In today's increasingly hostile environment, security, and in particular information security, is a vital issue requiring significant attention. As Australia increases its involvement in overseas operations, protecting the ADF's classified information will be a foremost concern due to the increasing threat levels. Overseas adversaries are not the only problem; attacks from within our own country also need to be considered, along with the vulnerabilities in the resources employed by the ADF. Thus protecting the confidentiality, integrity and availability of Defence information is becoming increasingly difficult and a methodical approach to security is required.

Security architectures provide this disciplined approach to developing security solutions. A security architecture is a high level design identifying and describing all the components used to satisfy a system's security requirements. These security requirements are generally defined in an organisation's security policy, which is a top level specification describing the rules and procedures for using information. There are a number of documents describing the Defence Security Policy, and the Commonwealth policy in general, including the Protective Security Manual (PSM), Australian Communications-Electronic Security Instruction 33 (ACSI33) and the Security Manual (SECMAN) series. When designing systems, it is important that a methodical approach is taken to ensure the security policy is adhered to and that appropriate measures are put in place to counter threats. This is the role of security architectures.

Designing security architectures is a distinct process based on an assessment of the threats to the specific system being implemented. It is therefore difficult to provide support for designing security architectures. However, an area where support is required is security architecture analysis. Currently, there are no defined systematic methods or tools used in the Department of Defence for capturing and, perhaps more significantly, analysing and comparing security architectures.

Security architectures research is currently being conducted at the Defence Science Technology Organisation (DSTO). The three major aims of the security architectures task are the ability to:

1. capture and describe security architectures;
2. generate attack and vulnerability scenarios from the architecture description; and
3. assess and compare security architectures.

A number of existing techniques that could potentially be applied to security architecture analysis were identified and investigated. This report is a summary of initial research conducted in these areas.

This report is structured as follows. Section 2 describes the prospective approaches to security architecture capture. Section 3 discusses potential techniques for discovering

attacks from a security architecture description, and Section 4 considers the possible techniques for comparing and assessing security architectures. Finally, there is a conclusion summarising the research and providing directions for future security architectures work.

## 2. Architecture Capture

A significant aspect of designing a security architecture is to capture the architecture in an appropriate way. The representation should be clear, concise and consistent to facilitate easy analysis and comparison of architectures. To ensure that these are features of all architecture descriptions, tools and methods need to be provided for capturing an architecture. This section discusses three potential techniques for capturing security architectures, namely the UK's Domain Approach, the Australian Defence Architecture Framework and the International Common Criteria's Protection Profiles.

### 2.1 Domain Approach

The Domain Approach [1] [2] [3] is a UK approach to security architecture capture. Its focus is the data exchanges between discrete, classified information systems.

The basic idea of the Domain Approach is that people who collaborate closely and share the same IT facilities on a daily basis work together in a **domain**. Domains represent logical places, within an information system, where people can perform their work by means of software acting on their behalf. People who work in the same domain are given the authority, and have the need, to freely exchange data. The only information security controls are who may work in the domain, the nature of the data handled in the domain, and the IT tools and applications that can be used in the domain.

On the other hand, the exchange of data between people who work in different domains is only allowed by means of clearly defined **connections**. Security constraints are placed on these connections, where they are seen as minimally interfering with the ability of authorised users (**members** of the domain) to conduct their daily business. The security constraints are also located where the protection is most required, as members of one domain do not have tight control over the users, data and applications present in other domains.

#### 2.1.1 Infosec Business Model

The **Infosec Business Model** of the Domain Approach describes **domains** and the **connections** (data exchanges) that are allowed between them. Connections may be two-way or one-way exchanges. Connections can be of different types, allowing people to interact with each other in different ways, for example, email connections or database connections.

The Infosec Business Model also describes **environments** and **portals**. Environments are the physical places where people work and where electronic equipment is located. Portals are the means by which people may interact with the domain, that is, the input and output devices. Within one environment there may be portals for several domains, and the people present in that environment may or may not be members of any of them.

An example Infosec Business Model is given in Figure 1.

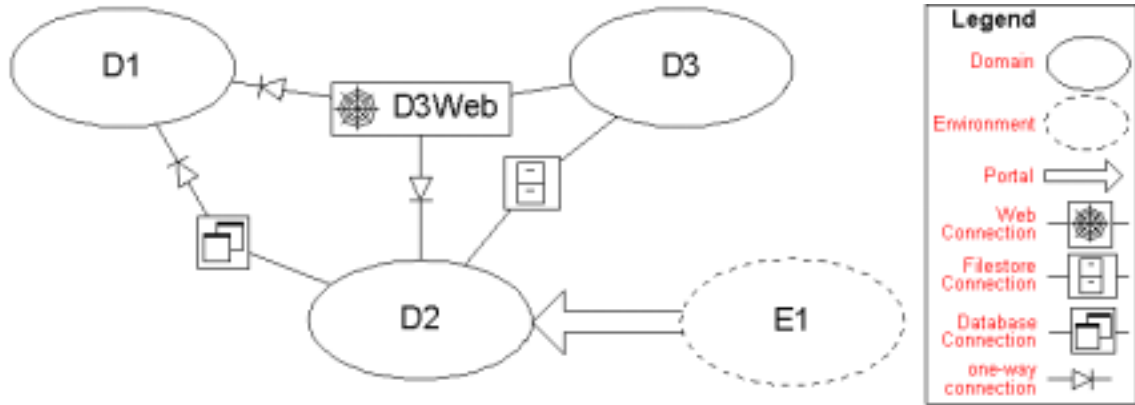


Figure 1: Example Infosec Business Model

### 2.1.2 Infosec Infrastructure Model

A single domain may be distributed over a number of computer networks, or multiple domains may share the same network infrastructure (which may include wireless and telephone networks). For this reason, the Domain Approach uses the concept of **islands** and **causeways** to describe the security properties of an infrastructure. The islands are isolated infrastructures that provide an “impenetrable” boundary, and the causeways provide the sole means of transferring data between the islands.

The **Infosec Infrastructure Model** is used to describe the **islands** of infrastructure and the **causeways** for transferring data between them. The causeways are the points where strong security controls should be provided, if they can be provided at all (depending on the available technology). The model is a high-level, simple view of the infrastructure implementation. It does not, for example, model the physical wiring, geographical positioning of devices or choice of cryptographic algorithms.

An example Infosec Architecture Model is given in Figure 2.

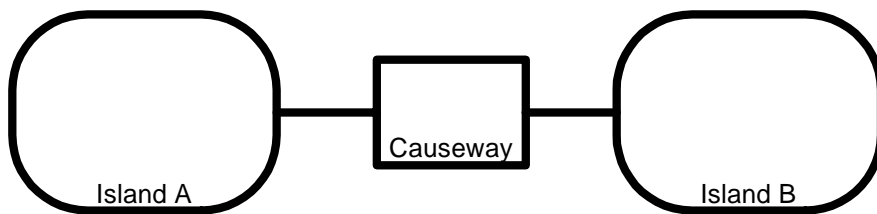


Figure 2: Example Infosec Infrastructure Model

### 2.1.3 Infosec Architecture Model

The **Infosec Architecture Model** describes how the business model is implemented by the infrastructure. It is generated by superimposing the Infosec Business Model and the Infosec Architecture Model, as seen in Figure 3. “Domains” in the business model are implemented by “islands” in the infrastructure model. However, there may be more than one domain on an island. “When domains are implemented by the same island, there is the risk that if members of one domain are able to ‘break out’ of their desktop applications and work at the level of the infrastructure, the data being handled by other domains becomes visible to them... unlike a causeway, [connections] are not required to be implemented within a small, well-defined component that provides the only link between the two domains” [3]. Implementations of a connection may be distributed throughout the infrastructure, and hence there is more potential for these measures to be bypassed or corrupted.

Domains distributed over several islands must be respecified as a number of component domains, one for each island, otherwise the causeways between these islands cannot be regarded as providing strong protection.

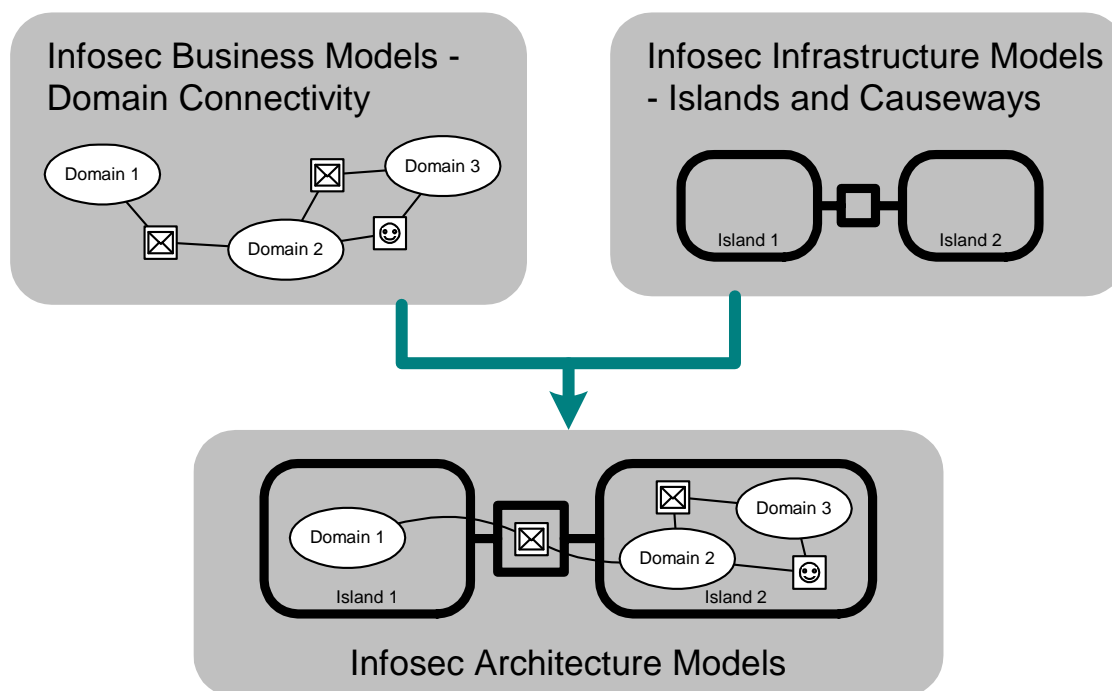


Figure 3: Example Infosec Architecture Model [2]

Infosec Architecture Models may be delivered with extra detail in the form of  $n \times n$  matrices ( $n$  being the number of elements in the model) and/or tables. The matrix/ table entries define the security properties and security functions of each element in the model.

### 2.1.4 Application to Architecture Capture

The Domain Approach is well established and widely used in the UK. It was specifically designed for the architecture capture of information systems handling classified information. Thus, it can be directly used for the development and maintenance of such systems.

### 2.1.5 Discussion

The models of the Domain Approach are easy to understand, as the starting point is the model of how business is done. Physical aspects can be incorporated (e.g. offices and buildings, and input/output devices). The models thus provide traceability between the infrastructure implemented and the model of how business is done.

Using the concept of a “domain”, the models of the Domain Approach highlight where the security risks are. The Infosec Business Model highlights security risks arising from how business is done (which can never be completely removed by technological advances) – there is a risk associated with each connection. The Infosec Infrastructure Model highlights where in the system it is possible to exchange business data, and where such exchange is demonstrably inhibited/enforced by the presence of causeways. If two domains are connected, either directly or through intermediate domains, the models allow an exhaustive search of all paths between the two domains to be made (by applying graph theory), and thus the minimum hurdles which any attack has to face.

Complex and highly detailed models can be built up in both a hierarchical and modular fashion, making the Domain Approach feasible for capturing complex architectures. Multiple diagrams can be used to depict different subparts and/or different levels of detail. Moreover, it is easy (by inspection) to verify that the diagrams depicting the subparts and/or the different levels of detail are consistent with each other. Each diagram can have a clear focus and be easy to understand in its own right.

An issue with the Domain Approach is that its models are purely descriptive (qualitative). Unless extensions are found or an underlying formal model applied, there are no metrics readily derivable from the models for use in analysis. It is not clear how different information security architectures can be assessed or compared. The clearest use for the Domain Approach is the capture and description of an architecture instance in an easy to understand way. The models need to be presented clearly for this purpose. Extensible drawing tools would be required to achieve user-friendliness.

The Domain Approach can really only be directly applied to the architecture capture of classified information networks. It may not be broad enough to handle the capture of architectures without networks unless appropriate extensions or mappings can be found.

## 2.2 Defence Architecture Framework

The Defence Architecture Framework (DAF) was developed to improve the design and coherency of the Defence Information Environment (DIE)<sup>1</sup> [4]. It draws on both the US DoD C4ISR Architecture Framework [5] and the META Group Enterprise Architecture Strategy [6]. A diagram and description of the components that constitute the DAF is shown in Figure 4 [4].

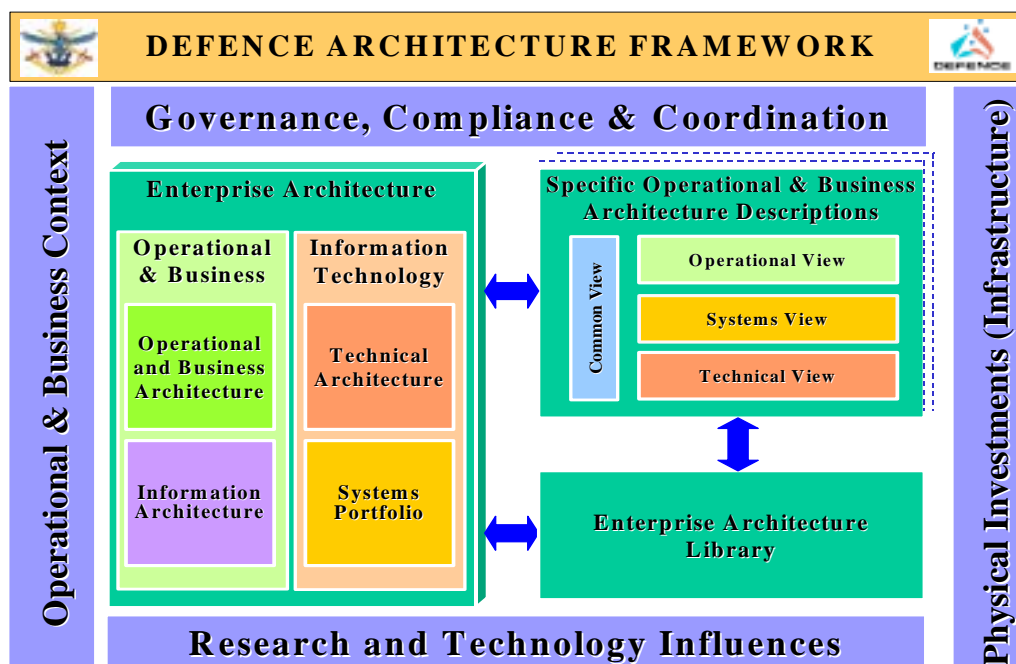


Figure 4: Defence Architecture Framework [4]

There are a number of **external factors** that influence architecture development:

- **Governance, Compliance & Coordination:** Governance ensures that the roles and responsibilities are in place for the compliance process, audit function and enterprise architecture development and management. Compliance is the process of ensuring that a change initiative is in accordance with relevant policies, standards and guidelines specified in the Architecture. Coordination is the evolution of the Architecture.
- **Operational & Business Context:** Describes the external setting which influences and shapes the way Defence operates (Includes Defence & Government strategies, international and national legislation and regulation and national alliance arrangements).

<sup>1</sup> The DIE is an inclusive view of the information environment within Defence, including intelligence and surveillance capabilities, communications, information warfare, command and headquarter systems, and management (logistic and business) applications.

- **Research and Technology Influences:** Describes the research activities and technological advances on the sustainability, improvement and evolution of the DIE.
- **Physical Investments (Infrastructure):** Describes the resources used to support the DIE (Including, for example, hardware, software, communication networks, applications and qualified staff).

The **Enterprise Architecture (EA)** provides the organisation with the methods, processes, discipline, and organisational structure to create, manage, organise, and use models for managing the impact of change. The EA is split into the **Operational and Business Architecture (OBA)**, which is technology independent, and the **Information Technology (IT)**, which supports the business. The components of the EA are:

- **Operational and Business Architecture (OBA):** The key component driving the Enterprise, it describes the business strategies, processes, structures, business roles, activities and organisation of Defence.
- **Information Architecture (IA):** Describes and models the enterprise's use of information. It includes the **Defence Language (DL)**, which publishes the meaning, source and associated business rules for all important terms used in Defence, and data models, which define relationships between Defence entities.
- **Technical Architecture (TA):** Describes the principles, technologies, products and standards which support the current DIE and from which new capabilities are built. Some of the domains which the TA is based on include operating systems, hardware, distributed computing and security.
- **Systems Portfolio (SP):** Describes the collection of information systems both in-service and under development used to support Defence business.

For each problem/operation/issue it is necessary to develop its **Specific Operational/Business Architecture Description**. The specific architecture enables the problem/operation/issue and suggested solutions to be explained unambiguously, essential assumptions and information to be made explicit and comparison of related problems and issues. The architecture is described through a standard set of **views**:

- **Operational View (OV):** Describes the tasks and activities, operational elements, and information flows associated with a Defence capability or information system.
- **Systems View (SV):** Describes (including graphics) systems and interconnections providing for, or supporting, warfighting and non-warfighting functions.
- **Technical View (TV):** A minimal set of rules governing the arrangement, interaction, and interdependence of system parts or elements, whose purpose is to ensure that a conformant system satisfies a specified set of requirements.
- **Common View (CV):** Describes information that is essential to the development and use of the Operational, Systems and Technical views. It also records the compliance information to assess the degree to which specific architecture descriptions comply with the Defence Enterprise Architecture.

The **Enterprise Architecture Library** holds both current and proposed architectural descriptions, tools and supporting reference material.

### 2.2.1 Developing Specific Architecture Descriptions

An architecture description consists of a set of **products**. Architecture products are those graphical, textual, and tabular items that describe characteristics of the architecture. For each of the four views (operational, systems, technical and common), there are a number of essential and supporting products that make up the architecture, as listed below:

#### *Operational View (OV) Products*

- OV-1: High-level Operational Concept Graphic (Essential).
- OV-2: Operational Node Connectivity Description (Essential).
- OV-3: Operational Information Exchange Matrix (Essential).
- OV-4: Command Relationship Chart (Supporting).
- OV-5: Activity Model (Essential).
- OV-6a: Operational Rules Model (Supporting).
- OV-6b: Operational State Transition Description (Supporting).
- OV-6c: Operational Event/Trace Description (Supporting).
- OV-7: Logical Data Model (Supporting).

#### *Systems View (SV) Products*

- SV-1: Systems Interface Description (Essential).
- SV-2: Systems Communication Description (Supporting).
- SV-3: Systems to Systems Matrix (Supporting).
- SV-4: Systems Functionality Description (Supporting).
- SV-5: Operational Activity to System Function Traceability Matrix (Supporting).
- SV-6: Systems Information Exchange Matrix (Supporting).
- SV-7: System Performance Parameters Matrix (Supporting).
- SV-8: System Evolution Description (Supporting).
- SV-9: System Technology Forecast (Supporting).
- SV-10a: System Rules Model (Supporting).
- SV-10b: Systems State Transition Description (Supporting).
- SV-10c: Systems Event/Trace Description (Supporting).
- SV-11: Physical Data Model (Supporting).

#### *Technical View (TV) Products*

- TV-1: Technical Architecture Profile (Essential).
- TV-2: Standards Technology Forecast (Supporting).

#### *Common View (CV) Products*

- CV-1: Overview and Summary Information (Essential).
- CV-2: Integrated Dictionary (Essential).
- CV-3: Capability Maturity Profile (Supporting).

- CV-4: Architecture Compliance Statement (Essential).

### 2.2.2 Application to Architecture Capture

The DAF would provide guidance in the development of a security architecture and its architecture products could be clearly used to capture aspects of a security architecture. Some products are specifically designed to describe security aspects, for example:

- SV-2 can be used to describe the measures in place to achieve secure communications and network isolation within a specific communications network.
- SV-6 can be used to describe security specific aspects of information exchanges between systems within a node and across nodes.

The US DoD C4ISR Architecture Framework (on which the DAF is based) provides methodology and examples to assist in developing architecture products. Also, within the ADO, the Chief Information Officer (CIO) and Knowledge Staff have developed a tutorial and tools, such as Microsoft Office templates, to assist with architecture development.

### 2.2.3 Discussion

The fact that there exist tools and methodology for the development of architecture products is a considerable advantage of using the DAF. Also, since it is based on the US DoD C4ISR Architecture Framework, anything developed in Australia will be understood, and perhaps supported, by the US.

The use of the DAF is being strongly encouraged throughout Defence by the CIO and Knowledge Staff. However, the DAF is still in a developmental stage and particular areas, such as security, have yet to be incorporated into the DAF.

Considering these factors, the DAF is considered an insufficient basis for security architecture capture. The main issue is the fact that the DAF does not directly deal with security issues. Developing a complete architecture description requires building all the essential products. However, many of these have no relevance to security and capturing aspects of security requires creating a number of the supporting products. Hence capturing a security architecture would be a time-consuming process, with much of this time devoted to developing products that would not be applicable to the security architecture and thus would be difficult to define. The resulting security architecture would not be clear and concise, which is a critical aim of architecture capture.

Whilst at this stage it seems the DAF will not be used, it is possible portions of the DAF may be used in the future to assist with architecture capture, for example, those products discussed in section 2.2.2. Also, the existence of general architecture products can provide valuable information to the security architecture definition process.

## 2.3 Common Criteria Protection Profiles

The Common Criteria (CC) is a set of internationally recognised criteria for the evaluation of IT security products and systems.

In the CC framework [7] [8], a Protection Profile (PP) is a formal document which states the high-level set of security requirements for a particular category of IT product or system. It states the security requirements to address an identified set of security objectives in an identified threat environment, without dictating how these requirements will be implemented. A PP is therefore intended to be reusable and to be met by more than one implementation.

The security requirements expressed in a PP can be taken either directly from the CC or stated explicitly if not found in the CC. The CC itself catalogues a set of standardised IT security requirements, expressed as modular components, which are generic and of known validity to information security based on past experience. Following the CC approach to security requirements, any explicitly stated requirements in a PP must be written in such a way that compliance or non-compliance of the requirement in an IT product or system can be demonstrated. (This allows a pass/fail verdict in an evaluation of the security properties of the IT product or system.)

CC security requirements are split into two distinct categories: functional requirements and assurance requirements. Functional requirements define what the IT product or system must do. Assurance requirements are the grounds for having confidence that the security functions in the IT product or system are effective and implemented correctly.

A PP is structured to incorporate several related kinds of security information [9]:

1. A statement of the **security problem** which an IT product or system, called the Target of Evaluation (TOE), is to address.
2. A description of the **security environment**: the threats to be countered, the assumptions, and the organisational security policies. (Thus, the security problem is refined to an intended environment of use.)
3. A statement of the **security objectives** for the TOE and its environment, giving information about how and to what extent the security problem will be addressed. (The scope of the TOE is determined at this stage.)
4. A statement of the **security functional requirements** and **security assurance requirements** identified for the TOE.
5. Optional **application notes**: additional supporting information that is considered relevant or useful for the construction, evaluation, or use of the TOE.

6. A **rationale**, which demonstrates that the security functional and assurance requirements are sufficient to meet the security problem in the intended environment. The rationale must show that the security objectives address the environment, and that the security requirements address the objectives.

A PP undergoes an evaluation to check that it is complete, internally consistent, and technically sound. The security objectives are not simply a negation of the threat, but should be realistic and achievable. The security requirements are appropriate for a specific threat environment.

If two or more PPs are needed to meet a security requirement, it will be necessary to demonstrate that the PPs are consistent and do not conflict.

### 2.3.1 Examples

The following is a list of examples of how PPs may be used [10].

- A PP may apply to a particular type of TOE. For example,
  - Operating system, database management system (DBMS), firewall, smart card.
  - Application software for electronic financial transactions.
- It could apply to a set of products grouped together into a composite TOE.
- The development of “PP families”, a set of closely related PPs which typically apply to the same product or system type. These include:
  - A series of PPs for the same type of product or system, but which provide different assurance levels. For example, operating system PPs for low-risk, medium-risk, and high-risk environments.
  - A set of PPs that apply to different components of an IT system. For example, a smart card family might, for example, include PPs for the integrated circuit card, operating system, application or smart card reader.
- A statement of user requirements. For example,
  - Efforts to express FIPS 140-1, the US security requirements for cryptographic modules (defined for four levels of rigour), as four PPs.
  - PPs for the “System High” and “Multi Level” mode of operation in Defence IT systems [11].
  - PPs for the finance/banking industry, and healthcare industry.

### 2.3.2 Application to Architecture Capture

PPs are formal, evaluated documents that capture IT security requirements as a complete, consistent and technically sound set. As they record the traceability from a security problem to its security requirements, with rationale, they can be used for security

architecture capture. The set of evaluated PPs forms a knowledge repository for well-known and understood domains.

### 2.3.3 Discussion

PPs are formal, evaluated documents that capture IT security requirements as a complete, consistent and technically sound set. This sounds nice in principle; however, there are a number of hurdles encountered with using PPs.

One problem is that PPs are the expression of IT security requirements of known validity. The CC has been developed on the basis that the security requirements catalogued represent a well-known and understood domain. PPs need to be updated to incorporate any newly discovered threats and attack methods. In addition, it is not certain how the CC will capture certain classes of security requirements, for example, cryptographic algorithms, virus scanners, and physical attacks.

Another issue is that the CC General Model does not clearly provide constructs for the definition of a “system”. Thus, the PP construct has no clear criteria for how to specify systems. It is quite likely that some PPs may be related, but there are no criteria to define the relationship. It is not certain how the resultant assurance level of a system is evaluated. (For clarification, note that the main difference between a product and a system is that, for a product, very broad assumptions are made on its actual working environment. A product can therefore be inserted into multiple systems.)

One major problem with PPs is that they take a substantial amount of effort to write. It takes technical expertise to derive specific security requirements from their generic expression in the CC. It takes a *lot* of technical expertise to evaluate a PP for completeness, consistency and soundness. Efforts have been made to develop a database of standardised threats and a database mapping threats to security objectives, as tools to help make PPs slightly easier to write [12].

Finally, as PPs are a record and statement of good security sets, they can in theory be used for the following purposes: as an expression of procurement specifications by consumers, as security requirements documents by developers, as a reference which captures industry standards, or as the basis for product or system evaluations. However, it is arguable whether a PP is the clearest or most efficient method for capturing information for all these different uses.

## 3. Discovering Attacks

Once the security architecture has been captured, vulnerabilities can be identified and attack scenarios generated. This is normally a manual and subjective process, specific to the particular architecture. However, with a consistent representation for all architecture descriptions, the process of discovering attacks may be simplified and become somewhat automated. A clear and accurate representation of attacks is also important to assist with analysis and comparison of security architectures. This section discusses potential techniques for discovering and representing attacks, including a threat database, attack trees, security protocol analysis and security failure analysis.

### 3.1 Threat Database

A threat database is a complete listing of threats against information security (confidentiality, integrity and availability). Before describing the components of a threat database, it is important to define a number of related terms:

- A **threat** is defined as any potential circumstance or event which could damage or misuse an asset. A threat is usually characterised in terms of:
  - A threat agent (optional).
  - A presumed attack method.
  - The assets that require protection.
  - The likelihood of the threat developing into an actual attack.
- An **asset** is information or resources that have value to stakeholders, either individuals or organisations. Examples include information, hardware, software and people.
- An **attack** is a threat that has become an event rather than a possibility.
- **Threat agents** are individuals or organisations with the capabilities and intentions to damage or misuse an asset. Example threat agents include hackers, nation states, terrorists and employees.
- An **attack method** is the means or manner of procedure for exploiting one or more vulnerabilities. Example attack methods include viruses, password cracking and denial of service.
- A **vulnerability** is a weakness in an information system's design, implementation, or operation and management that could be exploited. A very well known, general vulnerability is buffer overflows. There are thousands of software product specific vulnerabilities, many of which have been published in vulnerability databases that are widely available on the Internet.

- A **countermeasure** is a procedure, policy, device or other measure that is imposed to prevent, detect, or recover from attacks. Examples include passwords, virus scanners and firewalls.

A threat database therefore includes a complete description of threats, along with the vulnerabilities, attack methods that exploit vulnerabilities, countermeasures used to reduce vulnerabilities, and the threat agents who perform attacks. Perhaps the most useful components of a threat database would be descriptions of the likelihood and severity of threats, together with rationales for these assessments.

There currently exist very few threat databases. Perhaps one of the best examples is the All.Net Security Database [13] developed by Fred Cohen. It includes threat agents, attack methods, countermeasures, and cross-references that link these components together to describe the threats. Another example, which describes generic threats, is the CC Profiling Knowledge Base [14] developed by the National Information Assurance Partnership (NIAP).

### 3.1.1 Application to Discovering Attacks

In theory, a threat database would describe all the potential likely attacks on a security architecture. Therefore, the process of discovering attacks would simply involve considering whether each attack in the database could possibly be performed on the architecture being considered. This process could be made somewhat automatic. For example, the fact that a security architecture contains a particular countermeasure may prevent the possibility of a specific attack.

### 3.1.2 Discussion

It seems that a threat database would make the process of discovering attacks quite straightforward, requiring matching assets, threat agents, vulnerabilities and countermeasures in the architecture with those in the database. But the major problem is the lack of existing threat databases, and the ones that do exist are very high level and of little use for discovering specific attacks against a Defence security architecture. They also provide no threat assessment information, as this is obviously very specific to the systems and organisations being considered.

Therefore, a threat database specific to Australian Defence needs to be developed as part of the security architectures work. Some information for this database could certainly be taken from other sources, such as the existing threat and vulnerability databases. But even with using existing data, the process of creating a threat database would be difficult and time-consuming work requiring input from a wide variety of sources and experts. The work would never be finished as new threats constantly emerge and the threat database would need frequent updates to incorporate these changes. Also, due to the level of detail to be considered, the database is likely to be highly classified.

Besides its direct application to security architectures, a threat database would be an extremely valuable resource with a variety of other applications in Defence.

### 3.2 Attack Trees

Attack trees provide a methodical way of describing threats against, and countermeasures protecting, a system [15]. Attack tree is the term associated with using fault trees<sup>2</sup> for attack modelling [16], [17].

The **goal** of an attacker is identified as the root of the tree. The ways that an attacker can achieve this goal iteratively and incrementally are represented as lower level nodes of the tree. Each **path** through an attack tree then represents a unique attack on a system.

Each node of an attack tree is decomposed as either an:

- **AND-decomposition:** A set of attack subgoals, all of which must be achieved for the attack to succeed.
- **OR-decomposition:** A set of attack subgoals, any one of which must be achieved for the attack to succeed.

Values can be assigned to nodes and basic calculations performed to determine the likelihood of attacks. Possible values include boolean values, numerical values and even probabilistic values.

Attack trees can be represented both graphically and in written form. Figure 5 is a graphical example of an attack tree describing attacks against a safe [15]. Figure 6 is a written example of an attack tree describing the potential for a particular organisation's (ACME's) proprietary secrets to be disclosed [18].

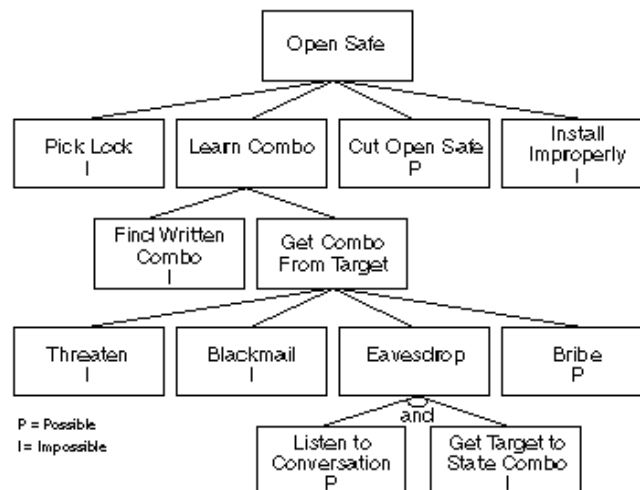


Figure 5: Graphical example of an attack tree [15]

<sup>2</sup> Fault trees are used to represent and analyse failure conditions in complex systems.

**Survivability Compromise: Disclosure of ACME proprietary secrets**

- OR** 1. Physically scavenge discarded items from ACME
- OR** 1. Inspect dumpster content on-site  
2. Inspect refuse after removal from site
2. Monitor emanations from ACME machines
- AND** 1. Survey physical perimeter to determine optimal monitoring position  
2. Acquire necessary monitoring equipment  
3. Setup monitoring site  
4. Monitor emanations from site
3. Recruit help of trusted ACME insider
- OR** 1. Plant spy as trusted insider  
2. Use existing trusted insider
4. Physically access ACME networks or machines
- OR** 1. Get physical, on-site access to Intranet  
2. Get physical access to external machines
5. Attack ACME intranet using its connections with Internet
- OR** 1. Monitor communications over Internet for leakage  
2. Get trusted process to send sensitive information to attacker over Internet  
3. Gain privileged access to Web server
6. Attack ACME intranet using its connections with public telephone network (PTN)
- OR** 1. Monitor communications over PTN for leakage of sensitive information  
2. Gain privileged access to machines on intranet connected via Internet

*Figure 6: Written example of an attack tree [18]*

**Attack patterns** [18] are a generic representation of attacks that support reuse of attack trees. An attack pattern contains:

- The overall goal of the attack specified by the pattern.
- A list of preconditions for its use.
- The steps for carrying out the attack.
- A list of postconditions that are true if the attack is successful.

Related attack patterns are organised into an **attack profile**. An attack profile contains:

- A common reference model.
- A set of variants.
- A set of attack patterns.
- A glossary of defined terms and phrases.

The reference model represents an architecture template with parameters that may include specific variants. The attack patterns are also defined in terms of the variants.

### 3.2.1 Application to Discovering Attacks

Traditional methods of discovering attacks are generally *ad hoc* but attack trees can provide a more methodical approach. This is achieved by constructing the tree from top to bottom, first identifying the goal of an attacker (e.g. to gain valuable/confidential data) and then identifying the attacks that could lead to the goal being achieved. This process repeats, by considering how each of these attacks could occur, eventually yielding a tree of attacks. In fact, it may be possible to automatically generate attack trees from a security architecture description, perhaps with assistance from a threat database.

Along with discovering and representing attacks, attack trees have other possible applications for security architectures. They could be used to assess the risk of each attack by assigning probabilities to nodes in the tree and using standard probabilistic calculations to determine the risk. (This is similar to the Bayesian Network approach discussed in section 4.1.) Also, attack patterns could be developed for common vulnerabilities and attack profiles generated for particular security architectures.

### 3.2.2 Discussion

There are a number of issues regarding the use of attack trees. Firstly, they may lack some detail and flexibility. For example, they do not specifically include vulnerabilities and countermeasures. This could invalidate the calculations used to determine the risk of attacks. Secondly, a graphical representation is the clearest depiction of attack trees but, as the number of attacks increase, the trees become too large and unusable. A textual representation can then be used but this potentially has little more structure than simply listing the attacks. Other issues include the usability and flexibility of existing attack tree tools and whether Bayesian Networks (discussed in section 4.1) can be used instead of attack trees.

A significant concern with attack trees, and all techniques used to discover attacks, is completeness. When is it apparent that all the possible attacks have been generated? The answer is never. It is always possible that one attack has been forgotten and it is therefore impossible to presume an attack tree is complete. To ensure attacks are not ignored, the process of generating attack trees should be performed interactively by a number of people with diverse backgrounds and relevant expertise and experience.

Whilst attack trees are a relatively new concept, they are based on the well understood technique of fault trees and fault tree analysis. Therefore, techniques from fault tree analysis can be easily applied to attack trees to provide a more systematic approach.

## 3.3 Security Protocol Analysis

**Security protocols** (also called cryptographic protocols) are communication protocols that use cryptography to achieve goals such as sender authentication and key distribution [19]. Well-known examples from the Internet domain are IKE, IPsec, SSL and S/MIME.

Security protocols tend to be relatively short and are carefully designed to begin with. However, they may still contain subtle “errors”, leading to security violations. Security protocol analysis, using formal methods, is used to search for errors in a protocol or prove that a protocol is “correct” with respect to an explicitly stated set of essential security properties.

Formal methods can potentially be used in the stages of specification, construction, and/or verification of security protocols. Most research in the area has been done for the formal verification stage [20]. The formal specification and verification of a security protocol can be performed using a number of approaches, each having its particular set of **languages** and **tools**. The approaches have been categorised below [20] [21].

A security protocol can be modelled or verified using **general-purpose** specification languages and tools. The first step is to specify the protocol and its correctness requirements using the chosen language. Next, investigation can proceed using the tools available in the language. Examples of general-purpose specification languages/ tools are:

- **LOTOS**: A modelling language based on processes; a tool is the CAESAR/ALDEBARAN toolkit [22];
- **Model checking**: Where a finite model of the system is built, and desired properties checked for by performing an *exhaustive* state-space search; a tool is the Failures-Divergence Refinement software package [23];
- **Isabelle**: A popular generic theorem prover; and
- **Petri nets**: A structure used to model systems with concurrently occurring events.

This approach contains a number of inherent weaknesses. Firstly, is the difficulty of generating subtle errors or attacks that are peculiar to the security domain (e.g. replay attacks, where replaying past messages leads to an old session key being used). Thus the approach can prove correctness but not necessarily “security”. Due to the full generality of the languages and tools, another weakness is the state explosion problem – as the number of interacting components increases, the size of the transition system increases exponentially.

Another approach is **special-purpose rule-based** tools, which allow various scenarios for security protocols to be generated and/or investigated. These systems begin with an undesirable state and attempt to discover if this state is reachable from an initial state. Example tools are **Interrogator** and the **NRL Protocol Analyzer**.

A weakness of the approach is that it is often inefficient because it performs an exhaustive search. It sometimes does not even halt, so human intervention is required. Although a protocol can be checked to see if it has a given flaw, previously unknown attacks are unlikely to be discovered. An advantage of the approach (similar to model checking) is that if it discovers a flaw, then the attack scenario that exploits the flaw is directly available.

Another approach is to model or verify a security protocol using **modal logics** that have been developed for the analysis of knowledge and belief [24]. The most well-known and influential logic of this type is **BAN logic**. The syntax provides constructs for expressing intuitive properties, for example, “A said X” or “K is a good key”, and deducing security properties, for example, “A and B believe K is a good key”. BAN logic is simple but needs many universal (often subtle) assumptions, and many issues need to be addressed in the informal mapping from protocol specification to BAN logic specification. Concerns have been raised about semantics and some limitations of the BAN logic, leading to extensions and variants. **GNY logic** is based on BAN logic and can cover more types of protocols at the expense of increased complexity. The **Automatic Authentication Protocol Analyzer** is a tool based on an extension of GNY logic.

Some other developments towards the modelling or verifying of security protocols are the **strand space** concept [25] and the **Common Authentication Protocol Specification Language**.

### 3.3.1 Application to Discovering Attacks

Security protocol analysis is applicable to a security architecture which has separated architecture components, and sequences of messages are required to be communicated between these architecture components. Furthermore, there has to exist the threat that attackers can intrude between the communicating components.

Some of the methods used for finding attacks on security protocols could be applied to find attacks on other security architecture components. An example of this is the methodology which investigates scenarios by beginning with an undesirable state and attempting to discover if this state is reachable from an initial state.

### 3.3.2 Discussion

Security protocol analysis is a very specialised field of research that is specific to the information security domain. Its specific nature allows it to find and analyse subtle attacks. Security protocol analyses are not undertakings which can be properly performed by non-experts. However, results generated by the experts can be made accessible and useful to a wider audience. Examples of some well-known attacks against protocols are “man-in-the-middle”, replay attacks, and “spoof-the-server”. Furthermore, the analysis of a security protocol, or even just a communication protocol (in a less hostile environment), could return specific attack sequences, if any exist.

Some inherent issues with security protocol analysis (and formal methods in general) exist. Firstly, security protocols proven “correct” using one formal technique have been shown to be insecure using another method. There are multiple ways to reason about the security of protocols and to prove that they are correct, so the definition of “security” appears not to be sufficiently formally defined. Another issue is that a formal specification

always includes assumptions made about the operational environment, although this is true of all techniques. A proof of correctness is only valid when the assumptions hold, so a system proved to be “correct” can be broken by an attacker violating the assumptions. Finally, security protocol *implementations* would need to be verified at some stage, but software and operating system verification is generally infeasible at present.

### 3.4 Security Failure Analysis

Security is a system property, reflecting the ability of the system to protect itself from accidental or deliberate attack. Failure analysis is a study performed to determine how a system can fail. Therefore, by “security failure analysis” is meant a study performed to determine how a system’s security measures can fail.

First, some terminology will be introduced [26]. **System failure** is an event that occurs when the system does not deliver a service as expected by its users. **System error** is behaviour of the system which does not conform to the system’s specification. **System fault** is an incorrect system state, that is, a state that is unexpected by the designers of the system. **Human error** is human user behaviour that results in the introduction of faults into a system.

Failures are usually a result of system errors brought about from faults in the system [26]. However, faults do not necessarily result in system errors, as the faulty system state may be transient and “corrected” before an error arises (“fault tolerance” during run-time). A failure can be prevented by error detection and recovery. Failures that have serious consequences are given greater weight by users than failures that are inconvenient. Complex systems should be designed to be resilient to a single point of failure.

Security failure is difficult in that it is more than just hardware component failures. Security failure is far less observable/detectable, as it can involve sophisticated intentional attacks against a system. Intrusion can change the executing system and/or its data, so that the executing system is no longer the same as the developed system [27] (e.g. buffer overflows, physical tampering), thus leading directly to system errors or failures. Successful attacks may not be detected. Different techniques for breaking into systems are constantly being developed. Of particular concern are software failures, which differ from hardware failures in that software does not wear out and continues to operate even after an incorrect or undesigned-for result has been produced.

Security failure analysis does not have an extensive research or publication presence. Thus the following discussion, which presents some types of failure analysis techniques, has been taken from the safety-critical and mission-critical domains. However, there is the hope that enough similarities exist with the security-critical domain to make the techniques at least partly applicable.

A **fault propagation graph** (FPG) is a directed graph that shows how component faults can affect other components in the system. The nodes in the graph are component faults,

discrepancies (anomalies) in the system behaviour, and sensors. The edges in the graph reflect the propagation of failures and capture the interactions between failures. An example graph is shown in Figure 7 [28]. The square nodes represent component failure modes, the circles represent discrepancies, and the ellipses denote sensors in the system. Discrepancies that contain a dot are monitored by sensors.

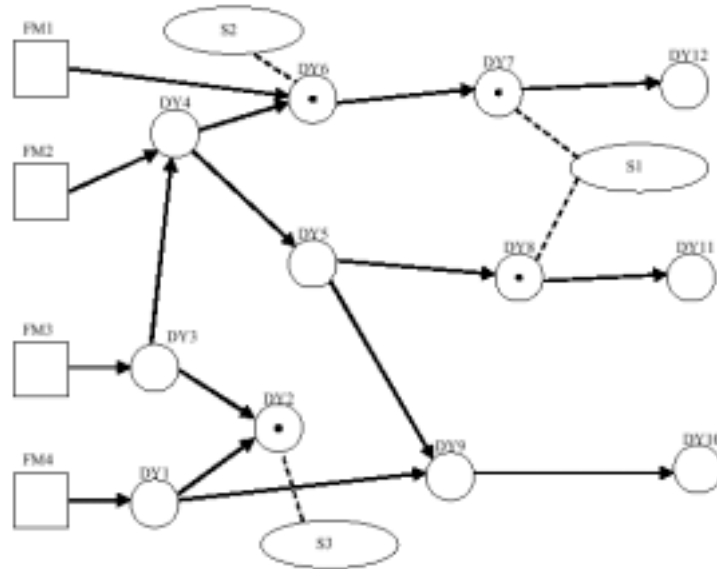


Figure 7: Example Failure Propagation Graph [28]

**Failure Modes and Effects Analysis (FMEA)** is “a procedure where every possible failure mode of the system or its components is analysed for their system-level effects and classified according to severity” [28]. The information to be analysed can be captured by FPG graphs. The analysis (whether hardware or software) identifies hazards, their causes, methods of control, and corrective actions for each component or function. Corrective actions include fail-safe mechanisms, redundant controls, error-handling routines, fault-tolerance, alarms, testing activities, and user warnings. These measures should be traceable through the FMEA. The resulting documents have the look of a checklist. Disadvantages of this approach include time required to perform the analysis and lack of scalability and data reuse. FMEA can be extended by a criticality analysis to reveal areas of the system that are vital to failure prevention. The technique has adapted itself to other forms such as Misuse Mode and Effects Analysis. The use of a knowledge base system for the automation of FMEA has been proposed. FMEA can be used to analyse product designs or production processes. Due to product/process complexity, they tend to be inductive, that is, starting with the lowest level failures and ending at the system level.

**Fault Tree Analysis (FTA)** is a widely used safety analysis method. In the security domain an “attack tree” could be used (See section 3.2 on Attack Trees).

FTA is described as being a “deductive” (top-down) procedure, identifying the high-level consequence first and seeking causes. On the other hand, **Event Tree Analysis (ETA)** is “inductive” (bottom-up), identifying an initial event as the root of the tree and seeking possible consequences [29]. Here is an example event tree scenario. Given an attack, the firewall system protects against an intrusion with probability  $p$  and does not protect against it with probability  $q = (1 - p)$ . By following the path from attack to consequences, when the probabilities of the mitigating events are known, we can determine the likelihood of each path. ETA can be combined with FTA. For example, the firewall system fails with probability  $q$ . If the failure of the firewall is exhaustively caused by independent events A1, A2 and A3 – identified in a FTA – then  $q$  is the product of the probability of A1, the probability of A2, and the probability of A3.

**Cause-Consequence Analysis (CCA)** is a mixture of FTA and ETA [29]. Hence it is both a deductive and inductive analysis. The purpose of CCA is to identify chains of events that can result in undesirable consequences. An example CCA diagram is shown in Figure 8.

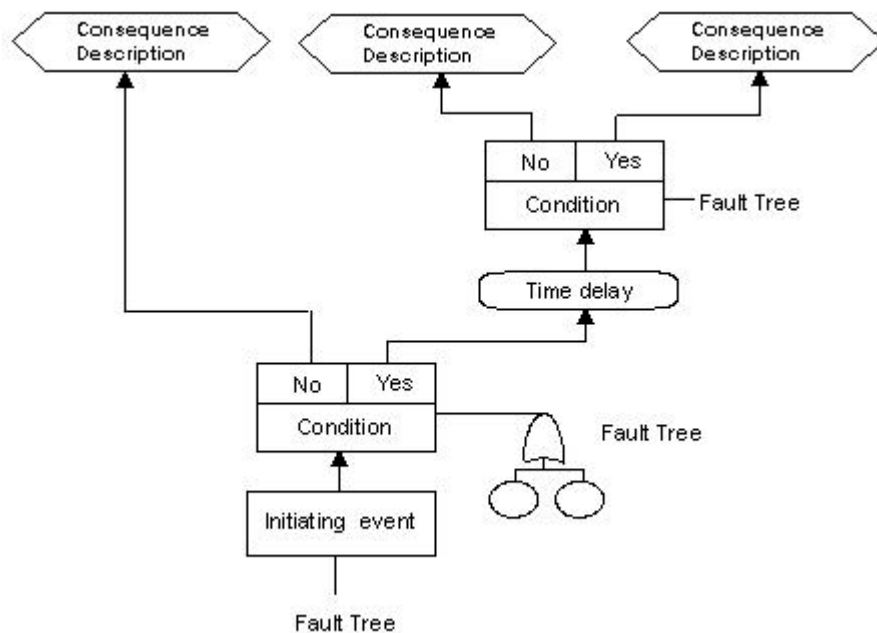


Figure 8: An example Cause-Consequence Analysis diagram [29]

Combinatorial models such as fault trees cannot accurately model dynamic behaviours [27]. Examples of such behaviours include repairs, common-cause and dependent failures, standby components, configuration changes, and complex error handling and recovery mechanisms. **Markov analysis** using Markov state transition diagrams could be used to model some of these cases. Another example is dynamic fault trees, an extension to fault trees, which changes to reflect state changes within a system.

**HAZOP** (Hazard and Operability studies) is a technique originating from Imperial Chemical Industries Ltd. for identifying hazards in the operation of a chemical process plant [29]. It is performed by looking at each “flow” in the system, and then considering deviations from the normal flow. To help discover hazards, a checklist of **guide words** is used. Typical guide words used are No, More of, Less of, As well as, Part Of, Reverse, More than, Other than. Both the causes and effects of deviations are considered, as well as any deviation “parameters”. For example, for a chemical process, some of the deviation parameters might be mass, pressure, temperature, density, and pH. So “more” may mean more mass, more pressure, more temperature, and so on. An adaptation of HAZOP to computer systems is CHAZOP (Computer HAZOP), where the guidewords are extended with Early, Late, Before, After.

**Behavioural modelling** describes systems by what they “do”, rather than what they are composed of. This could be potentially useful for analysing security failures, since security deals with undesired system behaviour, possibly behaviour that was not designed into the system, as a result of an attack. With behavioural modelling, the system can be thought of as an input/output device that is impacted by faults. The fault inputs are treated as a special class of inputs. Along with the behaviour model, there may be an associated observation model which is used to model how sensors react to inputs of the system.

### 3.4.1 Application to Discovering Attacks

Failure analysis is used to discover what components could fail in a system, the effects of those failures at the system-level, and the likelihood of those failures occurring. It should be possible to account for any fault-tolerance or error-handling measures put in place in the system by referring to the failure analysis.

Security failure analysis does not have to focus itself on finding every single possible failure. It could be applied to finding just single points of failure in a security architecture, for example. It could also take into account the likelihood of failures, for example, by contrasting architecture components evaluated to CC EAL2 versus EAL6. Furthermore, FPGs are just directed graphs, so automated graph analysis and search methods could be employed.

### 3.4.2 Discussion

For complex systems security failure analysis is bound to be a time consuming task. Finding the appropriate tools to draw and verify any graphs would be a big help. It is probably the case that anticipating all problem combinations in software-controlled systems, in particular, is infeasible. Analysis techniques that cannot be performed in a hierarchical and/or modular manner will not produce reusable results.

In the security domain, it also does not help that IT systems have a history of being susceptible to network attacks, including: design flaws (e.g. TCP/IP); software bugs; misconfiguration; proprietary technology; lack of information about potential attackers

and their aims, abilities, strategies and resources; and the difficulty of incident analysis after a successful attack.

## 4. Comparison and Assessment

After discovering the attacks, the attack scenarios can be analysed to determine their likelihood. A formal, systematic approach for this process is required, preferably mathematical, to ensure that an accurate likelihood is determined for each attack scenario. This information can then be used to assess and compare security architectures. This section discusses a number of possible techniques for comparison and assessment, including Bayesian networks, simulation, risk analysis, approaches from the Information Assurance Technical Framework (IATF), game theory, survivability analysis and economic models.

### 4.1 Bayesian Networks

A Bayesian network is a graphical representation of a probabilistic dependency model [30] [31] [32]. It is a **directed acyclic graph** (DAG) consisting of **nodes** and **edges**. Each node represents a random variable, and edges represent the causal dependencies between the variables. Each node has a number of **states** that correspond to the states of the random variable it represents. The belief, or certainty, of these states is determined by the belief in the states of the nodes directly connected to the node. This is achieved by associating a **conditional probability table** (CPT) with each node, which contains the probability of a node being in a given state given the states of its parents.

The mathematical basis for Bayesian networks is the Bayesian calculus, which is classical probability calculus. Nodes in a Bayesian network satisfy the basic axioms of probability. That is, for events A and B:

- (i)  $P(A)$  is a number in the interval  $[0,1]$ .
- (ii)  $P(A) = 1$  if and only if A is certain.
- (iii) If A & B are mutually exclusive, then
 
$$P(A \text{ or } B) = P(A) + P(B)$$

The basic concept in Bayesian networks is conditional probability.  $P(A | B)$  denotes the probability of the event A, given that B has occurred.

The fundamental rule for probability calculus is

$$P(A | B)P(B) = P(A \text{ and } B)$$

Bayes' Theorem follows from this and states that

$$P(A | B)P(B) = P(B | A)P(A)$$

Bayes' Theorem facilitates updating the belief in each state of a node as new information is made available. This process is called **inference**.

Figure 9 is a very small example of a Bayesian network [33]. The diagram shows the causal dependencies and CPTs for each node. The purpose of the Bayesian network is to determine why an apple tree is losing its leaves. There are two possible reasons: it is dry (perhaps due to drought) or the tree is sick.

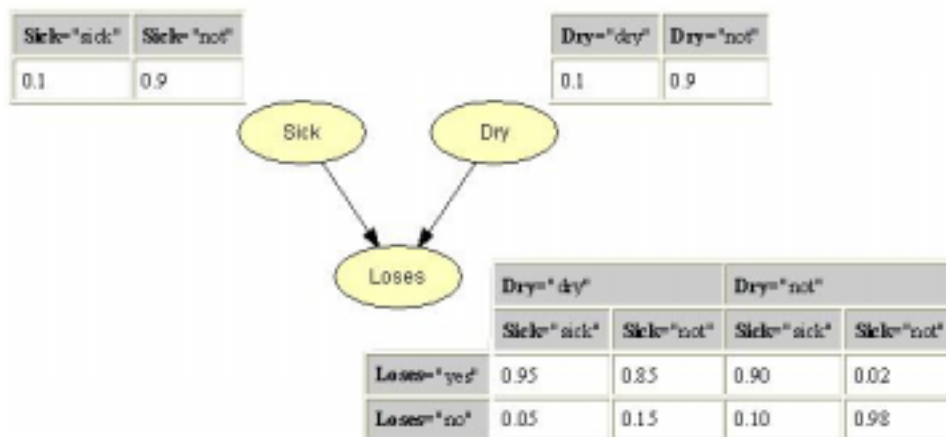


Figure 9: Apple tree Bayesian network example [33]

If, for example in Figure 10, we know the tree is losing leaves and it is dry, then we infer (using Bayes' Theorem) that there is only an 11% chance of the tree being sick.

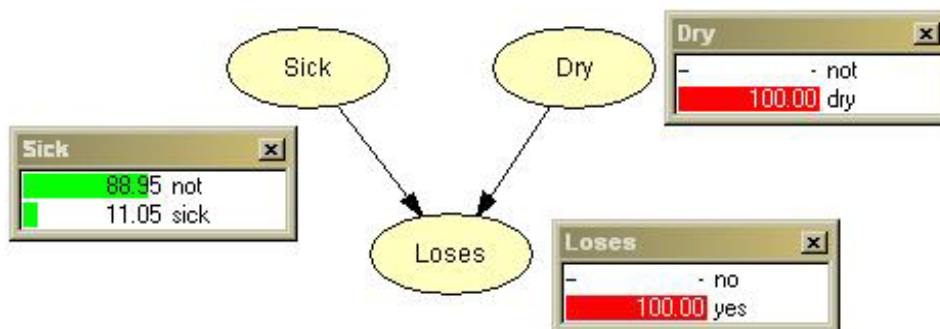


Figure 10: Using inference on the apple tree Bayesian network

**Influence diagrams** are Bayesian networks extended with decision and utility nodes. Decision nodes represent decisions that can be taken to influence the state of the world and utility nodes represent the utilities of decisions.

Tools are available for the construction of Bayesian networks. **Hugin** [33] is one such tool that enables the construction of Bayesian networks and Influence Diagrams. (Figure 10 is an example of Hugin output.) One particular feature of the latest version of Hugin (version 6.0) is the ability to create Object Oriented Bayesian networks. These can assist in reducing complexity and increasing the accuracy of Bayesian networks.

### 4.1.1 Application to Comparison and Assessment

Bayesian networks can provide a simple graphical representation of security threats, vulnerabilities and countermeasures, similar to an attack tree. (Attack trees are described in section 3.2.) More importantly, behind this graphical representation is a formal mathematical methodology that can be used, for example, to assess the effect of countermeasures on particular vulnerabilities and thus on the likelihood of threats. Hence Bayesian networks can be used to assess and compare security architectures. Also, they provide a mechanism for conducting “what-if” games on potential security architectures. For example, questions such as “what would be the effect if a firewall was placed here?” could be answered. Influence diagrams could also provide a tool for supporting security decision-making.

### 4.1.2 Discussion

Perhaps the most significant problem when using Bayesian networks is determining CPT values. For each node, if there are  $n$  edges coming into that node, then there are  $2^n$  values that need to be determined. This can be a difficult process requiring input from a variety of sources to ensure the values entered are accurate, otherwise the results of the entire network may be compromised. The CAST (CAusal STrength) logic [34] is one technique that may reduce the complexity of entering CPT values.

There are a number of other issues that need to be considered. Bayesian networks are frequently large and difficult to modify and use, although Object Orientation may help in this area. Also, Bayesian networks and attack trees are very similar techniques, thus it is important to determine which is the most appropriate and useful technique for security architectures (if either is used).

## 4.2 Simulation

**Modelling** is the process of capturing some specific issues of a real world system into a symbolic representation. Most models, by their nature, are not dynamic. **Simulation** is the repeated exercising of a model to add the dynamics and predict how the real world system will change over time, time being the main independent variable [35].

Simulation is useful because it may provide some benefits that real world experiences and physical experiments cannot:

- Computer execution of a simulation can be replayed or “fast forwarded” – long term activity compressed into short time periods.
- Simulations may provide scenarios that would be much too risky or impossible to duplicate in the real world. For example, component failure in an airplane control system.
- Through simulations, answers to “what if” questions can be discovered or suggested; it would be too late to discover the answer after a real crisis had

occurred or after the system had failed to perform. Here simulation would be used to test a single proposed system, or to compare different proposed systems (design trade-off decisions) before one was chosen for actual deployment. For example, testing a digital integrated circuit before manufacture.

- Simulation may cost less than, for example, utilising real computers, networks, software, protocols or personnel. There would be greater ease of scalability and reconfiguration. For example, training pilots or military personnel.
- Simulations allow the representation of multiple levels of abstraction. For example, simulate a data packet's traversal over the network (each intermediate hop), or simulate the TCP/IP processes at each endpoint computer.

A system being simulated is allowed to have random elements. As an example, consider a queue where customers arrive and depart. This involves randomness as nobody can guess the exact time the next customer will arrive at the queue. The behaviour of this system may be described by graphing the number of people in the queue against time, where an arrival to the queue increases the count by one while a departure decreases it by one. This graph could be obtained from observation of a real queue, but it can also be artificially generated (i.e. simulated).

A **random variable** is a quantity that is uncertain, for example, the inter-arrival time between two people to the queue. A statistical function (or **probability distribution**) is used to artificially generate a random variable, by mimicking the probabilistic features of the random variable. Data collection in the real system is required to estimate or validate the statistical function.

There are three major ways to approach discrete simulation [36] [37], depending on the programming language used to implement the simulation.

- **Event Scheduling.** An **event** is anything that changes the system state (other than the passage of time). The idea of event scheduling is to move along simulated time until an event occurs and then modify the system state depending on the event, possibly scheduling new events in the process. The events are stored in an **event queue**, which lists all events in the order they are to be processed. As an example, the arrival of a customer to a queue is an event. This event will generate a departure event for that customer (at a randomly-chosen later time), and a new arrival event for the next customer (also at a randomly-chosen later time).
- **Activity Scanning.** An **activity** is a period of time between two events in a simulation. For example, the two events may be the arrival of a customer and the departure of a customer. The activity is then "serving the customer". This activity is not triggered by any other activity. Instead, it occurs when the following two **conditions** are satisfied: there are customers in the queue, and the server is idle. During simulation, this activity is waiting to be executed, and the system must be continually scanned for the fulfilment of the conditions. The main advantage of this approach is that it is simpler to think about and formalise (as it focuses on

activities that occur over time rather than instantaneous events), however it is more inefficient.

- **Process Interaction.** In this approach, only the building blocks of a system need to be defined, not the details of simulation execution. For example, a queue system is defined by: a “source” component to generate the arriving customers, a “queue” component to hold the waiting customers, and a “server” component to serve the customers. The source sends customers to the server (if the queue is empty) or it places them in the queue (otherwise). When a customer arrives at the server, a service time must be generated. When the server becomes idle, the queue must be checked for more customers. Events still need to be scheduled in this simulation, but that fact can be hidden from the modeller. Simulation software provides the tools to define the system.

The simulation process involves designing the simulation experiments, running the simulations, and then analysing the output to draw conclusions. Animation, which is a visualisation of the simulation using sequences of images, can also be used to increase the observer’s understanding of the model or the real world system.

#### 4.2.1 Simulation Efforts in Information Security

Modelling and simulation can theoretically be utilised in the following areas of information security [35]:

- Research and development of new security countermeasures
- Testing of both attacks and defences
- Analysis of intrusions and attacks
- Education and training of personnel

As an example, existing network simulation tools can be used by system administrators who need a detailed understanding of, for example, packet flows, buffer overflow or operating system compromise. Sniffer data, together with “network modelling and simulation” software packages, can provide visualisation of split-second detailed traces to month-long statistical data. Among other things, these tools can provide insight on “unusual” network traffic, or be used to model server availability. A difficulty with these tools is the volumes of data that have to be gathered.

Fred Cohen [38] looked at models which were essentially schemes for classifying threats, attack mechanisms, protective mechanisms, and consequences. He concluded that classification schemes tended to miss out on some of the important aspects required for simulation: association between actors and actions, cause and effect relationships (e.g. if there is a case of physical destruction, a highly likely cause is operators), and notions of time or effort required to carry out an attack. The final model he used was based on a database interlinking threats, attacks, and defences to generate cause-consequence sequences of events. A set of statistical functions was used to model the effectiveness of each defence against each attack, which depended on attacker and defender skill levels.

This was laid over a concrete network model, consisting of a set of active nodes and links to describe the network under attack.

An effort worth mentioning is the **Easel** survivability simulation language and tool, developed at Carnegie Mellon University. The system is “intended for simulating, depicting, and gathering information about networks, software agents, and other active entities of the physical, electronic and software worlds, about their interactions, and about their collective global effects” [39]. The language allows simulation of a large number of cooperating, semi-autonomous entities that have neither global visibility nor central control, for example, the simulation of “unbounded” networks and highly distributed infrastructures.

BBN Technologies, under a DARPA contract, developed the **Cyber Command System (CCS)** [40]. The CCS provides a central management station with operator display and controls, allowing command and control of deployed network security components. In the event of an attack, CCS provides decision support to the human operator by recommending courses of action calculated to mitigate risk and damage.

#### 4.2.2 Application to Comparison and Assessment

One of the reasons for performing a simulation is to test a single proposed system, or to compare between different proposed systems (design trade-off decisions) before one is chosen for actual deployment. Simulation of proposed security architectures offers the possibility of obtaining an overview of their security properties and performance – before putting in place any detailed infrastructure.

#### 4.2.3 Discussion

The value of simulation relies on the accuracy of the models and the accuracy of the data upon which the simulation is based. However, for the information security domain, there is difficulty in obtaining reasonable models and precise metrics. Any model would attempt to capture human behaviour mixed in with the interactions of interdependent hardware/software systems, and with time intervals ranging from nanoseconds to years [38]. There is the question of how to represent social engineering or the level of training of personnel alongside automated attack scripts, for example. Many seemingly dissimilar elements will need to be combined into one model. There is no base set of metrics from which to base measurements for use in simulation. Reliable data collection needs to be performed.

The value of simulation also relies on the ability to explore the simulation space by running multiple simulations. Runs through the simulation space are sequences of events, brought about by the modelled actors and modelled automated systems. In computer security, the precise order of events is an important factor in determining the final outcome (whether an attack is successful or not, and its level of impact). Thus, missing a run through the space may miss a major result.

Alternatively, automatic or scripted sequences of events could be used as input to the simulation process. A threat database would be useful here. While this would not provide much in the way of new discoveries or insights, it can be used to compare and assess security architectures against known attack methods. An example of a comparison and assessment question (from Cohen's paper [37]) is as follows: Can a few single-level defences be better than defence-in-depth (the full set of defences)? Possibly yes – say, if the reaction times for the single-level defences are considerably shorter than for the defence-in-depth architecture.

### 4.3 Risk Analysis

**Risk** is defined as exposure to the chance of injury or loss [42]. Risk analysis consists of the following three components:

- **Risk Assessment:** Estimates the risk by relating actions to their probable consequences. The goal of risk assessment is to produce information to assist decision-makers.
- **Risk Communication:** Characterises and presents information about risks and uncertainties to decision-makers and stakeholders.
- **Risk Management:** Applies principles for choosing among different risk management strategies.

There are three types of risk analysis methods: qualitative, semi-quantitative and quantitative. **Qualitative analysis** uses words or descriptive scales (e.g. low, medium and high) to describe the magnitude of potential consequences and the likelihood that those consequences will occur. Generally, qualitative techniques are only used as an initial starting point for the risk analysis process, or when reliable data required for quantitative approaches is unavailable. Examples of qualitative techniques include **Failure Mode and Effect Analysis** (FMEA) and **Hazard and Operability Studies** (HAZOP), both discussed in section 3.4.

In **semi-quantitative analysis**, qualitative scales are given values that can be combined using defined formulae. Semi-quantitative techniques are a more ad-hoc approach to risk analysis and there are no defined methods.

**Quantitative analysis** is the most common approach and involves the calculation of probability, and sometimes consequences, using numerical data. Examples of quantitative techniques include **Fault Tree Analysis** (FTA) and **Event Tree Analysis** (ETA), discussed in section 3.4, and **Bayesian Networks**, discussed in section 4.1.

Risk Analysis techniques are widely used to assess health, safety and environmental issues. Examples of direct applications include using risk analysis to determine the risks from smoking, and the probability of an explosion in a nuclear reactor. A number of software packages have been developed in support of risk analysis in these industries.

### 4.3.1 Security Risk Analysis

Security is another area where risk analysis techniques are being used. Security risk analysis is an examination of the interrelationships between assets, threats, vulnerabilities and countermeasures to determine the level of risk [43]. Security risk analysis then consists of:

- **Risk Assessment:** Estimates the risk by analysing the threats to, and vulnerabilities of, an information system to determine the potential impact.
- **Risk Communication:** Characterises and presents information about risks and uncertainties to decision-makers and stakeholders.
- **Risk Management:** The process of identifying and applying countermeasures to secure the assets requiring protection as identified by the risk assessment.

It is possible to use qualitative or quantitative methods to perform a security risk analysis, either manually or using an automated software product. There are a number of security risk analysis tools currently being used:

- **InfoSec Assessment Methodology** [44]: Developed by NSA but not publicly available (in fact, only available to US citizens).
- **COBRA** [45]: A questionnaire based system using expert system principles (qualitative) to determine and assess the risks of an IT system. Compliant with ISO 17799<sup>3</sup>.
- **Buddy System** [43]: Offers both qualitative and quantitative risk analysis of information or physical security. Used by the US DoD.
- **CRAMM** [47]: A risk analysis tool compliant with ISO 17799.

The most important component of any risk analysis is risk assessment. Organisations conducting a security risk assessment are often interested in the financial impact. Thus quantitative methods are used to determine the **Annual Loss Expectancy** (ALE) [48]. This is calculated by taking all the threats, estimating the expected loss per incident and multiplying this by the expected number of incidents (probability of a threat). So if, for example, the ALE is \$10 million per year, then buying, installing, and maintaining a firewall to mitigate the risk for \$25,000 a year is a bargain. But spending more than \$10 million per year to reduce the threat would be a waste of money.

### 4.3.2 Application to Comparison and Assessment

Risk analysis techniques are already being widely used in the InfoSec field so using them to assess and compare security architectures should be a relatively straightforward process. A quantitative risk assessment method or tool could be used to evaluate all the

---

<sup>3</sup> ISO 17799 is "a comprehensive set of controls comprising best practices in information security" [46]. It is essentially an internationally recognized generic information security standard.

potential attacks on a security architecture and provide an estimate of the risk associated with each attack. This would provide valuable information when assessing the effectiveness of a security architecture.

Calculating the ALE could provide some cost justification for security spending and assist decision-makers. (This idea is discussed further in section 4.7 on Economic Models.)

#### 4.3.3 Discussion

A major issue with risk analysis techniques is unreliable and inaccurate data. It is often difficult to obtain the required data to perform an effective risk analysis. In particular, quantitative approaches, which generally use probabilities, suffer from inaccuracy as probabilities can rarely be precise and can, in some cases, promote complacency. Whilst qualitative approaches avoid this problem, the information they provide can often be of little use in conducting an accurate assessment.

There are many potential risk analysis techniques, some of which have been discussed in previous sections of this report. However, a more in depth analysis of such techniques may be required if risk analysis is to be used to assess and compare security architectures. And whilst there are a range of commercially available risk analysis products, an evaluation of these products would also be required to determine which of them, if any, could be used to assess security architectures.

### 4.4 IATF Approaches

The Information Assurance Technical Framework Forum (IATFF) is a US organisation, sponsored by the National Security Agency (NSA), created to develop solutions for information assurance problems. The Forum's main purpose is to support the **Information Assurance Technical Framework (IATF)** [49], a document that provides technical guidance for protecting information and information infrastructures. An **information infrastructure** processes, stores and transmits information critical to the mission/business operations of an organisation. Protecting this information is achieved through **Information Assurance (IA)**.

A major focus of the IATF is the Defence-in-Depth strategy, which describes how to protect an information structure using layers of security. Four major technology areas are identified, being the Network and Infrastructure, Enclave Boundary, Computing Environment and Support Infrastructures. Each of these areas is discussed in detail, with a description of the available technologies and the gaps between the current solutions and the desired capabilities.

Of particular interest is Chapter 4 of the document, which describes the principles for determining appropriate security countermeasures. Included in this chapter is a description of the robustness strategy.

#### 4.4.1 Robustness Strategy

Robustness is defined as the recommended level of security mechanism strength and assurance for an InfoSec solution. The robustness strategy therefore provides guidance on assessing the degree of robustness.

The aim of the robustness strategy is to help developers assess what **strength of mechanisms** and what **levels of assurance** (in development methodology, evaluation and testing) are required for a particular product, based on the value of the information to be protected and the associated (static) threat environment. The IATF does not define the exact security mechanisms to be used, rather it provides guidance on selecting appropriate mechanisms based on the determined SML (Strength of Mechanism Level).

The process of determining the degree of robustness should be applied to all components of a solution, both products and systems. The process first involves considering the value of the information to be protected. The IATF defines five levels of **information value**:

- **V1:** Violation of the information protection policy would have negligible adverse effects or consequences.
- **V2:** Violation of the information protection policy would adversely affect and/or cause minimal damage to the security, safety, financial posture, or infrastructure of the organization.
- **V3:** Violation of the information protection policy would cause some damage to the security, safety, financial posture, or infrastructure of the organization.
- **V4:** Violation of the information protection policy would cause serious damage to the security, safety, financial posture, or infrastructure of the organization.
- **V5:** Violation of the information protection policy would cause exceptionally grave damage to the security, safety, financial posture, or infrastructure of the organization.

The next step is to consider the perceived threat environment. Factors to consider when determining the threat to a particular solution include level of access, risk tolerance, expertise, and resources available to the adversary. The IATF defines seven **threat levels**:

- **T1:** Inadvertent or accidental events (e.g. tripping over a power cord).
- **T2:** Passive, casual adversary with minimal resources who is willing to take little risk (e.g. listening).
- **T3:** Adversary with minimal resources who is willing to take significant risk (e.g. unsophisticated hackers).
- **T4:** Sophisticated adversary with moderate resources who is willing to take little risk (e.g. organized crime, sophisticated hackers, international corporations).
- **T5:** Sophisticated adversary with moderate resources who is willing to take significant risk (e.g. international terrorists).

- **T6:** Extremely sophisticated adversary with abundant resources who is willing to take little risk (e.g. well-funded national laboratory, nation-state, international corporation).
- **T7:** Extremely sophisticated adversary with abundant resources who is willing to take extreme risk (e.g. nation-states in time of crisis).

After determining the value of the information and the threat environment, Table 1 below can be used to determine the appropriate SML and EAL (Evaluation Assurance Level).

*Table 1: Degree of Robustness*

Information Value	Threat Levels						
	T1	T2	T3	T4	T5	T6	T7
V1	SML1 EAL1	SML1 EAL1	SML1 EAL1	SML1 EAL2	SML1 EAL2	SML1 EAL2	SML1 EAL2
V2	SML1 EAL1	SML1 EAL1	SML1 EAL1	SML2 EAL2	SML2 EAL2	SML2 EAL3	SML2 EAL3
V3	SML1 EAL1	SML1 EAL2	SML1 EAL2	SML2 EAL3	SML2 EAL3	SML2 EAL4	SML2 EAL4
V4	SML2 EAL1	SML2 EAL2	SML2 EAL3	SML3 EAL4	SML3 EAL5	SML3 EAL5	SML3 EAL6
V5	SML2 EAL2	SML2 EAL3	SML3 EAL4	SML3 EAL5	SML3 EAL6	SML3 EAL6	SML3 EAL7

As shown in Table 1, there are three SMLs and seven EALs defined. The EALs are those defined in the CC. The SMLs are as follows:

- **SML1:** Basic strength or good commercial practice; Used to protect low value data; Resistant to unsophisticated threats (T1 to T3), such as inadvertent errors.
- **SML2:** Medium strength; Used to protect medium-value data; Resistant to sophisticated threats (T4 and T5), such as those from an organised group of hackers.
- **SML3:** High strength or high grade. Used to protect high-value data; Resistant to threats from national laboratories and nation-states (T6 to T7).

For each SML, the IATF provides recommendations for specific security services, including security management, confidentiality, integrity, availability, identification and authentication, access control, accountability and non-repudiation. As an example, Table 2 lists the IATF recommendations for supporting identification and authentication.

Table 2: Identification and Authentication Mechanisms

	Identification		Human-to-Machine Authentication		Peer-to-Peer Authentication			
	System IDs	Bio-metrics	Passwords PINs Challenge/ Response	Tokens	Certifi- cates	Cryptographic Algorithm		Personnel Security
						Effective Key Length	Key Management	
SML1	Uniqueness	Not applicable	Use of any of these methods.	Badge/ key static	Bind with SML1 cryptographic algorithm	40+ bits symmetric key length, 80+ exponent 512+ modulus public key length	SMI Cat. X, 80+ exponent 512+ modulus public key length, 80+ hash key length	Commercial hiring practices
SML2	Uniqueness and minimum character length	Use of any biometric method	Minimum effective length – TBD	Memory device, updated periodically	Bind with SML2 cryptographic algorithm	80+ bits symmetric key length, 160+ exponent 1024+ modulus public key length	SMI Cat Y, 160+ exponent 1024+ modulus public key length, 160+ hash key length	Equivalent of secret clearance
SML3	Uniqueness and minimum number of characters, minimum distance (e.g., Hamming)	Use of any biometric mechanism with a liveness test	Minimum effective length – TBD	CIK, updated every time	Bind with SML3 cryptographic algorithm	Because of the complicated nature of this level, consult a qualified systems security engineer 6	SMI Cat Z, also consult with a qualified systems security engineer <sup>1</sup>	Equivalent of top secret clearance

#### 4.4.2 Application to Comparison and Assessment

Annotating security architecture diagrams (such as the Domain Approach models described in section 2.1) with SMLs and EALs would provide a mechanism for assessing and comparing security architectures. For example, by examining the SMLs a developer could observe the threat levels the architecture is intending to protect against.

The robustness strategy also has an application to architecture capture as it provides a simple mechanism for describing the strength of particular components and the required level of assurance for those components. This would assist development of Protection Profiles (discussed in section 2.3) as the EALs for the robustness strategy are those defined in the CC.

#### 4.4.3 Discussion

One potential problem is the incompleteness of the robustness strategy in the current version of the IATF document (version 3.1). The IATF does not consider the strength of mechanism tables completed or adequately refined. Also, given that it took two years for an upgrade from version 3.0 to version 3.1, it may be a long time before the robustness strategy is completely defined by the IATF. Another issue is that the IATF Forum is based only in the US and thus the requirements will all be US specific.

There are plans to incorporate more CC language into the robustness strategy, rather than only in the description of the EALs. This could be particularly useful if Protection Profiles were being used as a tool for security architecture capture.

## 4.5 Game Theory

Game theory [50] [51] [52] is a mathematical theory of decision-making by participants in a competitive environment. The theory is concerned with the problem of finding an optimal course of action which takes into account the possible actions of all participants and any chance elements. No single participant or chance alone can determine the outcome completely.

The basic element is a **game**, which is described by its set of **rules**. These rules specify what each participant – a **player** – is allowed or required to do under all possible circumstances. The rules define the amount of information, if any, each player receives. They also define the interpretation of any chance events, how the game ends, the amount each player pays or receives (defined by a player's **payoff** function), and the objective of each player.

To simplify the mathematical description of a game, the concept of a “strategy” is introduced. In the actual play of a game, instead of making a decision at each move, the players use a **strategy** – a complete plan for playing the game from beginning to end, which covers all possibilities that may arise in the play, for example, the plan would incorporate any information which might become available to the player in accordance with the rules of the game. Thus, when each player makes the single decision of which strategy to use, a play of the game is determined. The payoff to each player as a result of the play can then be determined. Game theory suggests optimal player strategies (“**solutions**”) which give a stable outcome; game theory does not specify the actions that players do take.

A **pure strategy** is a strategy that is always played, regardless of other players' choice of strategy. However, the solution of a game may involve a **mixed strategy**, which is a probability distribution function over all the strategies of a given player. For example, a mixed strategy might be to play Strategy A 0% of the time, Strategy B 18% of the time, and Strategy C 82% of the time. Mixed strategies involve a player who wishes to keep their choice of strategy a secret from other players; the choice of strategy is assuredly kept secret by using a chance device to select the strategy.

Some classifications of games exist. A game is **finite** if each player has a finite number of moves and a finite number of available actions at each move. A game can be classified based on the number of players, for example 2-person and 3-person. Games can also be partitioned between those whose payoff functions are **zero-sum**, and those whose are not. In zero-sum games the players make payments only to each other (so the sum of the winnings is equal to the sum of the losses). In nonzero-sum games, win-win and lose-lose outcomes are possible. There is also a class of games referred to as games with perfect

information. **Perfect information** means that, at each move of the game, all players are completely informed about the previous moves of other players or of chance.

There are many well-known game situations that have been studied. For example,

- Hide-and-seek (e.g. a submarine trying to avoid the detection of a sensor).
- Duels (e.g. when to open fire, with an increasing probability of hitting an opponent with time).
- Blotto games (named after the legendary Colonel Blotto, who has to divide his attacking forces among several forts without knowing how the defenders are distributed among those forts).
- Barrier operations (where a searcher has the choice of going slow, to hear better, or fast, to cover as much ground as possible, and a penetrator has the choice of going slow, to be quiet, or fast, to minimise the length of an encounter).
- Pursuit and evasion (e.g. air combat, where a slow but more maneuverable airplane is being pursued by a faster but less maneuverable craft).

#### 4.5.1 Game Model Representations

Game models can be represented in either normal form (also called strategic form) or extensive form. The same game can be represented in *both* normal form and extensive form, but one of the representations may be more intuitive or less complex.

In a **normal form** game each player chooses a strategy once and for all, and these choices are made “simultaneously”. That is, players are not informed of other players’ actions before making their own decision. Normal form games having few strategies can be described conveniently with a matrix. The matrix for an example 2-person game is shown in Table 3. Player 1’s actions are identified with the rows of the matrix, and Player 2’s actions with the columns. Each entry shows the row player’s payoff followed by the column player’s payoff.

*Table 3: Example 2-person game matrix*

		Player 2	
		Fight	Flee
Player 1	Fight	5,5	0,10
	Flee	10,0	1,1

All finite normal form games have a “solution concept” called the **Nash equilibrium**. The Nash equilibrium is a set of strategies with the property that no player can benefit by changing their strategy while the other players keep their strategies unchanged. (The Nash equilibrium may include a mixed strategy.)

In the example matrix above, the Nash Equilibrium occurs when both players “flee”. From the point of view of Player 1, if he “flees” and Player 2 “fights”, he gets 10 units of payoff instead of 5. If, instead, Player 2 “flees”, he gets 1 unit of payoff instead of 0. However, Player 2 will be thinking the same thing, and so they both get 1 unit of payoff. Note that this is less than if they were to both **cooperate** by “fighting”, and so get 5 units of payoff each.

In an **extensive form** game the order of the moves and paths of play are captured. The order of the moves may directly affect the options available to the players (e.g. tic-tac-toe or chess). These games can be conveniently described with a game tree. The nodes represent a game’s **position** (or “state”) and the branches represent possible actions which lead onto successor positions. Terminal nodes show the final position and final player payoffs. Branches may have a probability measure, which is the probability of the associated action being taken (e.g. an action dependent on a dice throw). An example game tree is shown in Figure 11. The dashed oval indicates that Player 1, whose turn it is to move, does not know what strategy has been chosen by Player 2 in the previous move.

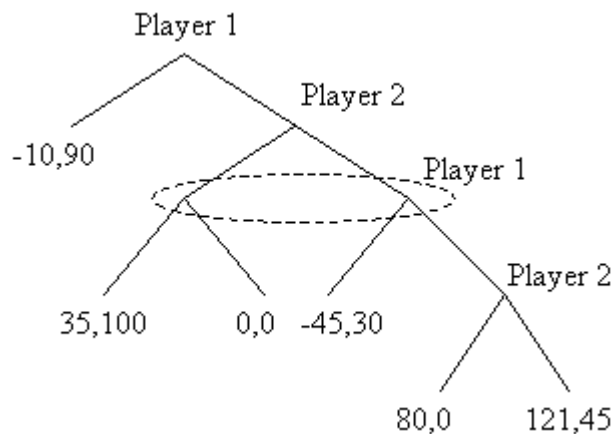


Figure 11: Example game tree

A technique called backwards induction can be used to “solve” extensive form games with perfect information (e.g. tic-tac-toe or chess). A player’s payoffs depend on what their opponents will do later in the game, so players can look ahead to future positions and then reason backwards to calculate the best current action. This is equivalent to traversing the game tree “backwards”. It is a search process, and can be infeasible if the number of possible positions is too large.

#### 4.5.2 Advanced Game Models

The following is a list of short descriptions of some advanced game models that have appeared in the literature.

- In a **repeated game** the *same* game is played multiple times, without past play affecting the payoff functions or set of available actions. Players can then take into

account the past play of their opponents, and the effect of their current play on the future behaviour of their opponents. Players seek to maximise their average payoff over all iterations played. In a repeated game players may be more likely to cooperate. The set of equilibrium strategies here is huge, so the concept of equilibrium lacks predictive power. Learning is typically modelled by assuming that some players find out about each other's strategies over repetitions of the game. Evolutionary models assume that strategies that do better on the average are played more often in the population over repetitions of the game.

- In a **multi-stage game** players move simultaneously at each discrete stage, and can observe others' actions taken in previous stages. Players may face a different set of available actions at each stage of the game, which is dependent on the past play.
- In a **cooperative game**, the strategies of the players are coordinated so as to attain the best result for a group of players – a “coalition” (which may be comprised of only one player). Players may change coalitions, for example, if they can do better for themselves by deserting their current coalition for another one. In cooperative games, pre-game contracts, bribes, threats and bargaining can exist.

Newer game models need to be the subject of empirical study, as any mathematical result must be confirmed by intuition or observation in real life.

#### 4.5.3 Application to Comparison and Assessment

*Theoretically* speaking, game theory is applicable to the information security problem. Game theory could be applied to model the interactions of people tasked with setting up a security architecture to defend an information system, and those many who will try to attack this security architecture. The interaction between defenders and attackers of an information system can be seen as an ongoing competitive game.

Game theory could then be used to predict or suggest the best strategies to use in setting up a security architecture, and to discover the strategies which attackers may use in trying to attack that security architecture. This analysis can be used to compare or assess security architectures.

Game models could theoretically be used as an input into simulations of attack and defence sequences of moves.

Game trees of common, experience-based moves and countermoves in information security could be useful knowledge (e.g. for use in simulations).

#### 4.5.4 Discussion

Current literature on the application of game theory to information security is sparse and immature. The game theory approach to information security is currently not yet well

developed, nor does it look to be well developed in the near future unless revolutionary discoveries are made.

However, it has been observed that attackers and defenders of information systems may employ “strategies”. Human attackers may make strategic decisions when selecting their attack methods. For example, some may take what they perceive to be the path of least resistance, some may take the path of least detection, and others may select the fastest attacks.

Cohen’s work [53] on attack and defence simulations lead to some example strategies:

primarily the notions of stealth and speed for the attacker, and speed and skill for the defender. The stealth strategy is one where the attacker tries to use methods that are unlikely to be detected, while the speed strategy exploits high speed attacks in the hopes that the likelihood of success before detection and reaction is higher. The defensive strategy of speed for the defender is [to detect and react to attacks before they cause damage, while skill is based on what proportion of the defender’s actions are performed correctly in the attempt to thwart attacks].

...Based on these results, a strong attack strategy would seem to be to attack as quickly as possible for a period of time less than the response time of the defenders, while doing so in a way that is hard to trace after the fact. ...There is also a clear advantage to knowing more about the defender’s defences because the more the attacker knows, the more likely it will be to find a workable stealth strategy. For this reason, it might be prudent for defenders to not demonstrate their full reaction capability on every attack. Thus the deception strategy wherein attacks are rerouted to a *honey pot* may be more effective than simply defeating an attacker by forceful termination of sessions. Needless to say, this discussion could go on almost without end.

In information security there exist strategies which are not purely technical in nature. Social engineering is a prime example. Another example (from Cohen) is getting a job within the organisation in order to break into a site – something a spy is likely to do while a hacker probably would not. If a purely technical attack is going to work, it will usually work quickly – in the order of seconds to hours. If a series of technical attacks fail and the attacker decides to use human effort, there is a relatively large time gap – in the order of weeks to months, even years.

In the information security domain, players have different goals, priorities, abilities and circumstances [54]. For example:

- An attacker's incomplete information of the network configuration may make them uncertain of a strategy's success and of how close they are to reaching their goals.
- However, the attacker has information regarding which attacks are successful (e.g. a captured password) which the defender may never know the outcome of.
- The defender may be able to observe the attacker's actions (e.g. using system logs/monitoring and audit trails).
- An attacker may be a risk-taker, much more likely to play strategies that have high payoff but low probability of success.

There are many other aspects to consider, observed in real life, when trying to model information security using a game theoretic approach [54]. For example, players can make multiple, simultaneous moves. It can take a variable amount of time to carry out a move, and moves may have effects that were not intended. Also, the set of known available moves may change at any time during the game as unknown moves (new techniques) are frequently introduced. Input used to improve the situational awareness is taken from a variety of sources: human, network data, previous events, and system scans.

A general criticism of game theory is that there are many ways in which rational behaviour and strategic reasoning can be interpreted. Any assumptions made about "rational" entities are under perpetual attack by experimental psychologists. However, while the game theory assumption of rationality may not be appropriate in modelling the behaviour of a standalone amateur attacker, it may be appropriate in the case of an organised hacker group or intelligence organisation with many resources to pool from.

## 4.6 Survivability Analysis

**Survivability** is the capability of a system to fulfil its mission, in a timely manner, in the presence of attacks, failures, or accidents [55]. The term **system** is used in its broadest sense to include networks and large-scale, highly distributed systems such as the telecommunication system, the electricity system and transportation systems. In particular, the focus of survivability is on **unbounded networked systems**, such as the Internet, where traditional security precautions are inadequate. It is important to recognize that it is **mission fulfilment**, which often includes an aspect of timeliness, that must survive, not any particular subsystem or system component.

A key characteristic of survivable systems is their capability to deliver essential services. **Essential services** are defined as the functions of the system that must be maintained when the environment is hostile, or when failures or accidents occur that threaten the system. Central to the delivery of essential services is the capability of a system to maintain **essential properties** such as specified levels of integrity, confidentiality and performance. **Essential components** are then those components of the architecture that must be available to deliver essential services and maintain essential properties.

To maintain their capabilities to deliver essential services, survivable systems must exhibit the four key properties of resistance, recognition, recovery (“the three R’s”) and adaptation:

- **Resistance** is the existence of strategies for repelling attacks. Potential approaches to resistance include common security countermeasures such as user authentication, access control, encryption and firewalls.
- **Recognition** is the existence of strategies for detecting attacks and evaluating the extent of damage. Examples of recognition strategies include intrusion detection systems, auditing, system monitoring and integrity checking.
- **Recovery** is the existence of strategies for restoring compromised information or functionality, limiting the extent of damage, maintaining or restoring essential services within the time constraints of the mission and restoring full services. Examples of techniques that assist recovery include backups, redundant modules, data replication and restoration procedures.
- **Adaptation** and evolution is the existence of strategies for improving system survivability based on knowledge gained from intrusions. This generally involves improving resistance, recognition and recovery mechanisms based on information obtained from intrusions.

**Survivability analysis** is the process of understanding the survivability risks to a system and then identifying the changes that can improve survivability. There are currently two known techniques for conducting survivability analysis.

#### 4.6.1 Survivable Systems Analysis Method

The CERT Coordination Center at Carnegie Mellon University has developed the **Survivable Systems Analysis (SSA) method** [56] [57] (formerly known as the Survivable Networks Analysis (SNA) method) to evaluate the survivability of systems. SSA is conducted by looking at the whole system architecture, or selected components, and considering the possible attack scenarios. A diagram of the steps involved in the analysis is shown in Figure 12.

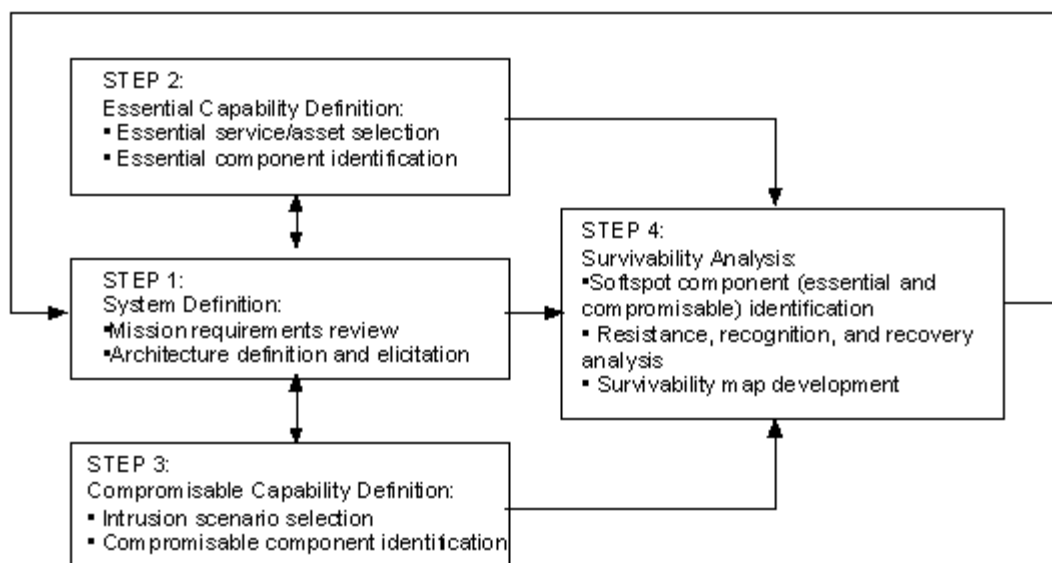


Figure 12: The four step SSA method

Step 1, **System Definition**, involves reviewing the mission requirements of the current or candidate system and determining the structure and properties of the architecture in terms of hardware components and connections, software configurations and information residency.

In step 2, **Essential Capability Definition**, essential services and **essential assets** (assets whose integrity, confidentiality, availability, and other properties must be maintained during attack) are identified, based on mission objectives and consequences of failure. These services and asset uses are characterized by **usage scenarios**, which are the steps required for users to invoke the services and access the assets. The usage scenarios are then mapped onto the architecture through **execution traces** to identify the corresponding essential components.

Step 3, **Compromisable Capability Definition**, involves selecting a set of representative **intrusion scenarios** based on the system environment and assessment of risks and intruder capabilities. These scenarios are likewise mapped onto the architecture to identify corresponding **compromisable components** (components that could be penetrated and damaged by intrusion).

Finally, in step 4, **Survivability Analysis**, **softspot components** are identified as those components that are both essential and compromisable, based on the results of steps 2 and 3. The softspot components and the supporting architecture are then analysed for the key survivability properties of resistance, recognition, and recovery. The analysis of “the three R’s” is summarized in a **Survivability Map**, an example of which is shown in Table 4 [58]. The map enumerates, for every intrusion scenario and corresponding softspot effects, the current and recommended architecture strategies for resistance, recognition, and recovery.

The survivability map provides feedback to developers and may result in an iterative process of survivability evaluation and improvement.

Table 4: Example Survivability Map examining a health care system [58]

Intrusion Scenario	Resistance Strategy	Recognition Strategy	Recovery Strategy
An unauthorised user corrupts the database (DB) leading to loss of trust in all validated treatment plans by all providers.	<b>Current:</b> Security model in DB protects TPs against corruption	<b>Current:</b> None, except when a provider happens to recognise a corrupted TP	<b>Current:</b> Locate an uncorrupted backup or reconstruct TPs from scratch.
<b>Softspot:</b> Treatment Plans (TPs)	<b>Recommended:</b> Implement live replicated DBs that cross check for validity (supported by many DBs)	<b>Recommended:</b> Add and check crypto-checksums on TPs in the DB.	<b>Recommended:</b> Reduce the backup cycle to quickly rebuild once a corrupted DB is detected

#### 4.6.2 Formal Approach

An alternative to the SSA method is a formal approach that uses automatic verification techniques such as model checking [59]. **Model checking** (also discussed in section 3.3) is a technique for proving properties about a system, where the properties are expressed in a temporal logic.

The first step in the formal analysis is to model the network as a set of concurrently executing **finite state machines**. Each node has a set of input channels and a set of output channels, along with associated finite queues. When an input arrives at a channel, it is appended to the associated queue. Similarly, when a system processes an output, it appends it to the relevant queue. A node can be in one of a finite set of states. In any given state, a node receives inputs from queues associated with a set of input channels, transitions to a state depending on the data it receives, and then outputs data on queues associated with a set of output channels. A network is a set of nodes and interconnections, where an interconnection is simply a pairing of an input channel to an output channel.

The next step is to inject faults and intrusions. For each node, a special variable called “fault” is introduced. It indicates whether a node is in the normal mode of operation, faulty, or compromised by an intruder, that is,  $\text{fault} = \{\text{normal}, \text{failed}, \text{intruded}\}$ . The user specifies the actual behaviour of the node in each mode of operation. Transitions between the modes can be specified by the user or can be non-deterministic.

Next, **survivability properties** are expressed using a temporal logic, such as Computation Tree Logic. There are two classes of properties. **Fault detection** properties express whether

the network can enter a faulty state. **Transactional properties** are related to the specific system services.

Model checking is then performed to verify properties about the network. If a certain property turns out to be false, then a scenario graph is generated. A **scenario graph** is a compact representation of all the traces that are counterexamples of a given property.

Further analysis can then be conducted from the scenario graph. **Symbolic analysis** is where the designer assigns symbolic probabilities, such as high and low, to events of interest (faults and intrusions). A formalism based on Bayesian Networks is used to specify the probabilities of the events which are then combined with the scenario graph to answer queries. **Reliability analysis** is where the designer provides numeric probabilities instead of symbolic ones.

#### 4.6.3 Application to Comparison and Assessment

Survivability obviously includes security so any survivability analysis technique would provide important information in regard to security. Both the SSA method and formal approach discussed in this section could be particularly useful for analysing security architectures containing networks. In the case of the SSA method, the resistance and recognition strategies identified in the Survivability Map are particularly relevant for security analysis.

#### 4.6.4 Discussion

The survivability analysis techniques discussed in this section focus on networks which therefore restricts the types of security architectures that can be analysed using these techniques.

The SSA method requires a set of representative intrusion scenarios (or attack scenarios – techniques for determining these are discussed in section 3). Without a complete set, the results of the analysis could be incomplete. The SSA method is certainly thorough and not only analyses the current system but provides recommendations that improve survivability, and thus security. Contrasting the recommendations for different architectures could therefore be used as a comparison technique.

The formal approach to survivability analysis is very mathematical and requires a knowledge of finite state machines, model checking and temporal logics, as well as tools to assist with the use of these techniques.

### 4.7 Economic Models

The global popularity of, for example, the computer industry, software product monopolies, the Internet and open-source community has required a revised economic

perspective to account for the behaviours of the consumers, users, and communities in this arena.

Economic models of information security are the result of researchers like Ross Anderson, who has observed that many information security problems can be *explained* more clearly using the language of microeconomics rather than technical measures [60]. He sees the discipline of microeconomics<sup>4</sup> as a source of alternative models which may be usefully imported into the information security domain. In particular, the question is whether information security is a problem having only partial technical solutions. For example, environmental problems have partial technical solutions – pollution abatement technology such as catalytic converters – however, the source of the problem lies in the behaviour of individuals, so practical and feasible solutions must take this into account.

The traditional view is that information security comes down to purely technical measures. It has been observed that designers of cryptographic systems concentrate on what can possibly happen in theory, rather than on what is likely to happen in practice, and assume that criminals have the expertise – and use the techniques – of a government signals agency. Designers have concentrated on building reliable *components* rather than systems, therefore tending to ignore the system aspects and human factors. Security problems that continue to be observed regularly are caused by economic and social factors [61], for example, three-letter passwords.

The argument by Gordon and Loeb [62] is that, economically, organisations have a finite amount of resources, which should be allocated accordingly based on a comparison of costs and benefits. Businesses that over-protect will have spent too much on information security, and those that under-protect will suffer greater losses as a result of the ensuing security breaches. It is regarded as neither feasible nor usable to protect all information absolutely (the traditional view of information security from Defence origins) as the costs would be too prohibitive. For example, information can be more than adequately protected if companies isolate all their computer networks from the Internet, however, they are not likely to do this because of the costs and restrictions of such a security policy.

In the economic-based **optimisation model**, the goal would be to implement security procedures up to the point where the difference between the total benefits – directly related to losses associated with security breaches – and the total costs associated with information security activities was a maximum. In Figure 13,  $S^*$  is the point where the difference between the benefits and costs is greatest.

---

<sup>4</sup> Microeconomics is the bottom-up view of the economy, focusing on individual households, firms and industries. This is in contrast to macroeconomics: the top-down view of the economy, focusing on aggregate characteristics like unemployment rate and inflation rate.

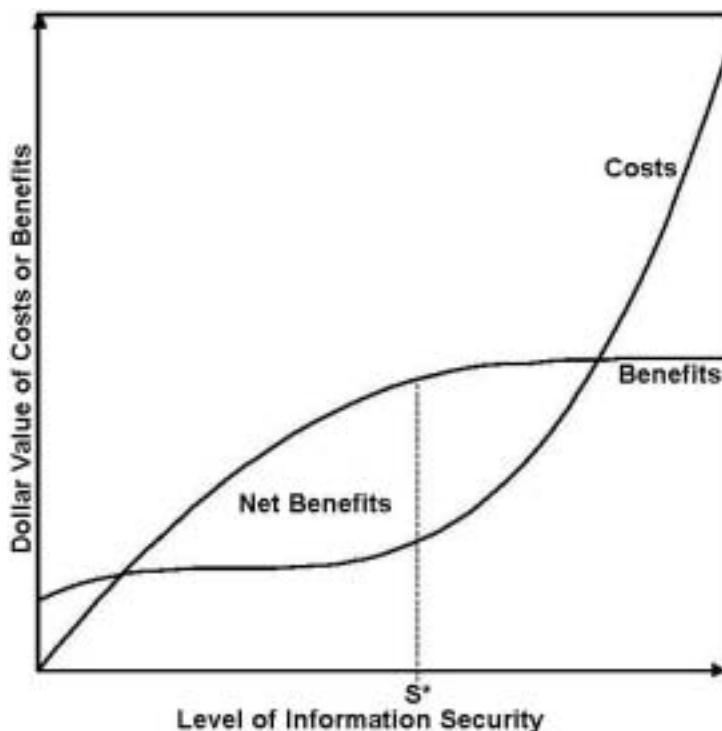


Figure 13: Benefits and Costs of Information Security

The major difficulty in trying to implement the cost-benefit approach is accurately estimating the potential benefits from information security. A security breach may have unrecoverable spillover effects like loss of customer confidence and loss of intellectual property, and may draw in third-parties. In practice, it is assumed that there is some initial level of resources being used, and then a **net present value (NPV) model** for analysing incremental expenditures is applied. The risk (or uncertainty) associated with benefits and costs is explicitly considered as part of the decision rules for accepting or rejecting incremental expenditures.

For individual users, on the other hand, network insecurity is acceptable because it is usually viewed as an annoyance rather than as a substantial cost. Many individuals (both at home or at work) perceive a low level of risk from network attacks. The impact of a break-in is often invisible and the remaining risk is addressed by the ability to backup and restore the system.

A strong economic issue is the demand from users for features and user-friendliness over security [61]. As an economic model, the industry is at an *equilibrium* that has consumers tolerating a certain amount of fault in software in return for functionality. It is widely accepted that software vendors could provide more reliable software if the demand for it from consumers existed, as evidenced by game consoles. When there is a demand, vendors can make corrections in a future software release, or provide patches for vulnerabilities that need to be corrected immediately.

A software phenomenon is the “user innovation network”, examples of such networks being “free” and “open source” software projects. These networks have innovation development, production, distribution and consumption all being performed by users/self-manufacturers. “User innovation networks can function entirely independently of manufacturers because (1) at least some users have sufficient incentive to innovate, (2) at least some users have an incentive to voluntarily reveal their innovations, and (3) diffusion of innovations by users is low cost and can compete with commercial production and distribution” [63]. As a result of this, software supply is provided through the open-source software development process *as well as* via market/price mechanisms.

The same concept can be applied to the more general “user content networks” that offer content which users either post as of interest to others and/or questions that users post to the network for a possible answer. Examples are hacking exploits and tools, much web site material, and newsgroups.

**Metcalfe’s Law** is a phenomenon where the more people use a typical network, the more useful it becomes. For example, the more people use the phone system, Internet or credit cards, the more people there are to interact with and the more useful it is to each user. So, while these networks may grow very slowly at first, “once positive feedback gets established, they can grow very rapidly” [60]. The same principles apply to virtual networks, e.g. “the community of users of a mass-market software architecture” – as seen by the volumes of software available for the PC rather than the Mac.

A related model is **tipping models**. This situation arises when there is one or a small group of organisations occupying such as strategic position that if it does things a certain way, all others will find it in their interests to do the same. This is relevant for policy-making, as it means there are some key players whom it is particularly important to persuade, as a substitute for working with the population as a whole.

#### 4.7.1 Application to Comparison and Assessment

Economic models can be used for decision-making, to reason about which approaches are practical and feasible. They take a systems and human factors view of a security architecture, as opposed to a technical view.

The argument presented by Gordon and Loeb [62] can be used in the comparison and assessment of different security architectures. It recognises that total security is usually not an option, and getting close to the maximum possible benefits from information security for a particular organisation incurs too much of a cost. These models would need to be altered to be situation and industry specific, for example, the model would be different for a government intelligence agency, a public library, a high-technology company, and a manufacturing company. Economic models take into account the *business* of the organisation, not just information security as an isolated goal. Should an organisation roll out a centrally controlled, overarching public-key infrastructure? Or should different

departments be left to manage their own information security infrastructures? Which is better for the organisation, from an economic, systems, human factors, and technical point of view?

Economic models could be another source of input into simulations.

#### 4.7.2 Discussion

It is fitting that this report should end with this topic, as a look at a 'paradigm shift' away from the traditional focus of information security.

An issue with the economic argument is how to measure the value of loss of data confidentiality and data integrity, in particular. The leakage of sensitive data or the damage resulting from contaminated data may have unrecoverable effects. Economic models are reasonable with regard to businesses and individuals but may arguably be too risky for governments and military organisations, who stake national security and reputations on sensitive data being kept confidential and precise. When information networks become interconnected this kind of information may become, to a significant degree, dependent on public technologies like the Internet and COTS hardware and software.

"Defence economic models" would have to be specifically looked into. For example, the role of Defence and government agencies is sometimes that of providing the "best shot" solution. In the provision of public goods, Hirshleifer [64] describes the best-shot case as arising in situations where there is a single prize of overwhelming importance to the community, with any individual's effort having a chance of securing the prize. Examples are proving a longstanding mathematical conjecture, or creating the first usable EAL7 operating system. If there is a greater probability that someone else will solve the problem first, there is a lesser incentive to try and solve it oneself, hence only participants with the highest benefit-cost ratio will contribute. Only the most capable government research laboratories are likely to attempt the building of formally-modelled operating systems.

## 5. Conclusion

The aim of this technical report was to present an overview of a very broad range of techniques which could potentially be used in the analysis of security architectures. Whilst all of the techniques are relevant to security architecture analysis, some are considered more useful than others.

For the architecture capture phase, three techniques were presented: the Domain Approach, the Defence Architecture Framework and the Common Criteria Protection Profile. The Domain Approach is seen as a promising approach. It is easy to understand and would allow a concise representation of an organisation's discrete information sets along with any appropriate physical elements such as buildings, server rooms, and printers. As the UK is actively promoting the Domain Approach, there is enough motivation to make drawing and analysis tools available, and thus make it fully useable. The Defence Architecture Framework does not deal specifically with information security, and is likely to be *too* broad to be ideally suited to architecture capture. Protection Profiles are formal documents that could certainly capture a security architecture, but perhaps at an unnecessary level of detail.

For the discovering attacks phase, four techniques were presented: a threat database, attack trees, security protocol analysis and security failure analysis. The threat database is seen as a very useful tool in general, but particularly for discovering potential attacks against a security architecture. The main problem is that it would need to be specifically generated for the organisation, requiring a significant amount of effort and input from a wide variety of sources and experts. Attack trees can provide a systematic approach to discovering attacks and certainly supply a tool for representing attacks. However, they may be lacking in detail and flexibility. Security protocol analysis is an extremely formal, specialised area of work that provides only minimal assistance in discovering attacks and thus is considered unusable. Security failure analysis includes a number of potentially useful techniques. The analysis is suitable for trying to completely identify and prioritise points of failure in a system, and justifying the existence of any countermeasures. To be useable for complex architectures, security failure analysis would need to have tools and/or the ability to be at least partly automated.

For the comparison and assessment phase, seven techniques were presented: Bayesian networks, simulation, risk analysis, Information Assurance Technical Framework (IATF) approaches, game theory, survivability analysis and economic models of information security. Bayesian networks could provide a useful tool for security architecture assessment by considering the effect of countermeasures on potential attacks. However, justifying the data used in Bayesian networks is a serious issue that needs to be considered. Simulation would be a very useful tool due to its dynamic nature, giving decision-makers a "feel" for the architecture. However, it relies on the existence of an accurate model, which is hard to obtain in the information security domain. Risk analysis techniques are certainly useful for assessing and comparing security architectures but, as

with Bayesian Networks, unavailable or inaccurate data can reduce their effectiveness. The IATF robustness strategy provides minimum requirements on architectures, but the incompleteness of the strategy and its US specific requirements are issues to be considered. Game theory could theoretically provide optimal designs for security architectures. Unfortunately, it is not well developed enough for the information security domain to be relied upon in the near future. Survivability analysis techniques are useful for architecture assessment, but are restricted to architectures containing networks. Economic models have practical, non-technical uses, incorporating a human factors and system view into the security architecture analysis. However, they do not provide the most important answers for government and Defence information systems.

Having researched a very broad range of existing techniques, future work would be to consider which ones to actually use in a complete methodology for security architecture analysis. If the final methodology partitions the process into three distinct phases, then the techniques used for each phase need to be interoperable. Simple and “standard” case studies will have to be considered to test a methodology’s usability and completeness, followed by more “real life” and complex architectures. In the end, it is hoped that this work will provide useful tools and methods for security architecture analysis and greatly assist the decision making process.

## 6. References

### Domain Approach:

- [1] Defence Evaluation and Research Agency (DERA) 2000a, *Annex A to DIAN/08: Domain Approach Techniques: Overview of Domain Approach*, Defence Evaluation and Research Agency (DERA), United Kingdom.
- [2] Defence Evaluation and Research Agency (DERA) 2000b, *Annex B to DIAN/08: Domain Approach Techniques: Introduction to Infosec Architecture Models*, Defence Evaluation and Research Agency (DERA), United Kingdom.
- [3] Defence Evaluation and Research Agency (DERA) 2000c, *Annex C to DIAN/08: Domain Approach Techniques: Infosec Architecture Models – Developer’s Guide*, Defence Evaluation and Research Agency (DERA), United Kingdom.

### Defence Architecture Framework:

- [4] Chief Information Officer 2002, Department of Defence, Australia, viewed 19 September 2002, <[http://intranet.defence.gov.au/cio/24/3315\\_1.html](http://intranet.defence.gov.au/cio/24/3315_1.html)>.
- [5] C4ISR Architectures Working Group 1997, *C4ISR Architecture Framework*, Version 2.0, Department of Defence, United States.
- [6] META Group 2002, META Group, Connecticut, United States, viewed 24 September 2002, <[http://www.metagroup.com/products/insights/eas\\_1\\_sco.htm](http://www.metagroup.com/products/insights/eas_1_sco.htm)>.

### Common Criteria Protection Profiles:

- [7] Common Criteria Implementation Board (CCIB) 1999, *Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and General Model*, ver. 2.1, CCIB-99-01, August 1999.
- [8] Syntegra Inc. n.d., *Common Criteria 2 – An Introduction*, a brochure produced on behalf of the Common Criteria Project Sponsoring Organisations, United Kingdom.
- [9] ISO/IEC 2000, *Guide for the Production of Protection Profiles and Security Targets*, ISO/IEC JTC 1/SC 27 N 2449, April 2000.
- [10] Ross, R 2002, *Defining IT Security Requirements for Federal Systems and Networks: Employing Common Criteria Protection Profiles in Key Technology Areas*, presented at the Workshop on Distributed Objects and Components Security 2002, National Institute of Standards and Technology, United States, viewed 23 September 2002, <<http://niap.nist.gov/niap/events/pp-development-projectv2.pdf>>.
- [11] Fahs, R & Wiseman, SR 1999, *Re-Floating the Titanic: Multi Level Security in contemporary environments*, European Institute for Computer Anti-Virus Research (EICAR), viewed 20 September 2002, <<http://www.eicar.org/download/titanic.doc>>.
- [12] National Information Assurance Partnership (NIAP) 2001, National Institute of Standards and Technology (NIST), National Security Agency (NSA),

Department of Defence, United States, viewed 20 September 2002,  
<<http://niap.nist.gov/tools/cctool.html>>.

#### Threat Database:

- [13] Fred Cohen & Associates 1999, United States, viewed 28 October 2002,  
<<http://www.all.net/CID/Threat/Threat.html>>.
- [14] National Information Assurance Partnership (NIAP) 2001, National Institute of Standards and Technology (NIST), National Security Agency (NSA), Department of Defence, United States, viewed 20 September 2002,  
<<http://niap.nist.gov/tools/cctool.html>>.

#### Attack Trees:

- [15] Schneier, B 2000, *Secret & Lies: Digital Security in a Networked World*, John Wiley & Sons, New York.
- [16] Steffan, J & Schumacher, M 2000, *Collaborative Attack Modelling*. Department of Computer Science, Darmstadt University of Technology, Germany, viewed 19 September 2002, <<http://www.ito.tu-darmstadt.de/publs/papers/sac2002.pdf>>.
- [17] Johnston, W, Kearney, P & Yesberg, J 2000, *Structured Threat Identification Techniques: An Initial Case Study*, Software Verification Research Centre, University of Queensland, viewed 11 November 2002,  
<<http://svrc.it.uq.edu.au/Bibliography/svrc-tr.html?00-35>>.
- [18] Moore, A, Ellison, R & Linger, R 2001, *Attack Modelling for Information Security and Survivability*, Software Engineering Institute, Carnegie Mellon University, Pennsylvania United States, viewed 19 September 2002,  
<<http://www.cert.org/archive/pdf/01tn001.pdf>>.

#### Security Protocol Analysis:

- [19] Song, D 2000, *Athena: A New Approach to Efficient Automatic Security Protocol Analysis* (Abstract), Berkeley University of California, United States, viewed 8 October 2002,  
<<http://www.eecs.berkeley.edu/IPRO/Summary/00abstracts/dawnsong.1.html>>.
- [20] Pandya, A 2002, *Formal Specification and Verification of Security Protocols*, Indian Institute of Technology Bombay, India, viewed 8 October 2002,  
<<http://www.it.iitb.ac.in/~abhinay/seminar/whole.html>>.
- [21] Gritzalis, S, Spinellis, D & Georgiadis, P 1999, 'Security Protocols Over Open Networks and Distributed Systems: Formal Methods for their Analysis, Design, and Verification', *Computer Communications*, vol. 22, no. 8, May 1999, pp. 695-707.
- [22] Drayton, L, Chetwynd, A & Blair, G 1992, 'An Introduction to LOTOS Through a Worked Example', *Computer Communications*, vol. 15, no. 2, Butterworth-Heinemann, March 1992, pp. 70-85.
- [23] Clarke, EM & Wing, JM 1996, *Formal Methods: State of the Art and Future Directions*, Carnegie Mellon University, Pennsylvania United States, viewed 8

October 2002, <<http://www-2.cs.cmu.edu/afs/cs/usr/wing/www/mit/paper/paper.html>>.

- [24] Hutchinson, A 1997, *Logic Analysis of Security Protocols*, University of Cape Town, Cape Town, viewed 8 October 2002, <<http://www.cs.uct.ac.za/courses/CS400W/NIS/papers97/pdegoede/report.html>>.
- [25] Fábrega, F, Herzog, JC & Guttman, JD 1998, 'Strand Spaces: Why is a Security Protocol Correct?', *Proceedings of the 1998 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, California United States, May 1998, pp. 160-171.

#### Security Failure Analysis

- [26] Heinrichs, GW 2001, *UML Design Techniques, Safety Systems*, Lecture 9 of Comp 474 Software Engineering, Loyola University, Chicago United States, viewed 21 October 2002, <<http://www.math.luc.edu/faculty/gheinric/comp474.htm>>.
- [27] Sommerville, I 2000, *Dependability*, lecture notes for CS 415-002 Software Design and Development, University of Alabama, United States, viewed 21 October 2002, <<http://cs.ua.edu/415S/ch16.ppt>>.
- [28] Davis, JR 2000, *Integrated Safety, Reliability, and Diagnostics of High Assurance, High Consequence Systems*, Ph.D. Dissertation, Vanderbilt University, Tennessee United States, May 2000.
- [29] Keong TH, *Risk Analysis Methodologies*, viewed 21 October 2002, <<http://home1.pacific.net.sg/~thk/risk.html>>.

#### Bayesian Networks:

- [30] Krieg, M 2001, *A Tutorial on Bayesian Belief Networks*, Surveillance Systems Division, DSTO, viewed 23 September 2002, <<http://www.dsto.defence.gov.au/corporate/reports/DSTO-TN-0403.pdf>>.
- [31] Balaram, D 2000, *Representing Uncertainties Using Bayesian Networks*, Information Technology Division, DSTO, viewed 24 September 2002, <<http://www.dsto.defence.gov.au/corporate/reports/DSTO-TR-0918.pdf>>.
- [32] Jensen, F 1996, *An Introduction to Bayesian Networks*, UCL Press, London.
- [33] Hugin Expert A/S 2001, Hugin Expert A/S, Denmark, viewed 25 September 2002, <<http://www.hugin.com>>.
- [34] McCarthy, J 2001, *A Forgiving View of the CAST Logic*, Information Technology Division, DSTO.

#### Simulation:

- [35] Saunders, JH 2001, *The Case for Modeling and Simulation of Information Security*, SANS Institute, viewed 29 November 2002, <<http://rr.sans.org/aware/curtain.php>>.
- [36] Trick, MA 1995, *An Introduction to Simulation*, Carnegie Mellon University, Pennsylvania United States, viewed 29 November 2002, <<http://mat.gsia.cmu.edu/simul/simul.html>>.

- [37] Sanden, BI 2000, *Simulation Program Principles: Chapter 3*, lecture notes for CS 750 Simulation and Teaching, Colorado Technical University, Colorado United States, viewed 29 November 2002, <<http://iis-web.coloradotech.edu/bsanden/CS750/Ch3.ppt>>.
- [38] Cohen, F 1999, *Simulating Cyber Attacks, Defenses, and Consequences*, Fred Cohen & Associates, United States, viewed 30 November 2002, <<http://www.all.net/journal/ntb/simulate/simulate.html>>.
- [39] Fisher, DA n.d., *Design and Implementation of EASEL: A Language for Simulating Highly Distributed Systems*, CERT – Carnegie Mellon University, Pennsylvania United States, viewed 29 November 2002, <[http://www.cert.org/easel/easel\\_foundations.html](http://www.cert.org/easel/easel_foundations.html)>.
- [40] BBN Technologies, *Cyber Command System*, BBN Technologies, viewed 29 November 2002, <<http://www.bbn.com/infosec/ccs.html>>.
- [41] Arsham, H 2002, *Systems Simulation: The Shortest Route to Applications*, University of Baltimore, Maryland United States, viewed 29 November 2002, <<http://www.ubmail.ubalt.edu/~harsham/simulation/sim.htm>>.

#### Risk Analysis:

- [42] Cox, L Jr 2002, *Risk Analysis: Foundations, Models and Methods*, International Series in Operations Research and Management Science, Kluwer Academic Publishers, Boston.
- [43] Countermeasures Corporation 2002, Countermeasures Corporations, Hollywood, United States, viewed 12 December 2002, <<http://www.buddysystem.net>>.
- [44] Information Assurance Directorate 2002, National Security Agency (NSA), United States, viewed 11 December 2002, <<http://www.nsa.gov/isso/iam/iam.htm>>.
- [45] C & A Security Risk Analysis Group 2002, C & A Security Risk Analysis Group, Cheshire, United Kingdom, viewed 10 December 2002, <<http://www.security-risk-analysis.com>>.
- [46] ISO 17799 Information Security Group 2001, ISO 17799 Information Security Group, Bucks, United Kingdom, viewed 10 December 2002, <<http://www.iso-17799.com>>.
- [47] Insight Consulting 2002, Insight Consulting, Surrey, United Kingdom, viewed 12 December 2002, <<http://www.insight.co.uk/cramm>>.
- [48] Schneier, B 2000, *Secret & Lies: Digital Security in a Networked World*, John Wiley & Sons, New York.

#### IATF Approaches:

- [49] Information Assurance Technical Framework Forum 2002, *Information Assurance Technical Framework*, Release 3.1, National Security Agency, United States, viewed 29 November 2002, <[http://www.iatf.net/framework\\_docs/version-3\\_1/index.cfm](http://www.iatf.net/framework_docs/version-3_1/index.cfm)>.

## Game Theory:

- [50] Dresher, M 1961, *Games of Strategy: Theory and Applications*, Prentice-Hall Inc., New Jersey United States.
- [51] Osborne, MJ & Rubinstein 1999, A, *A Course in Game Theory*, 6th edn, Massachusetts Institute of Technology (MIT), United States.
- [52] Fudenberg, D & Tirole, J 2000, *Game Theory*, 7th edn, Massachusetts Institute of Technology (MIT), United States.
- [53] Cohen, F 1999, *Simulating Cyber Attacks, Defenses, and Consequences*, Fred Cohen & Associates, United States, viewed 30 November 2002, <<http://www.all.net/journal/ntb/simulate/simulate.html>>.
- [54] Hamilton, SN, Miller, WL, Ott, A & Saydjari, OS 2001, *Challenges in Applying Game Theory to the Domain of Information Warfare*, CERT – Carnegie Mellon University, Pennsylvania United States, viewed 1 October 2002, <<http://www.cert.org/research/isw/isw2001/papers/Hamilton-31-08-a.pdf>>.

## Survivability Analysis:

- [55] Ellison, R, Fisher, D, Linger, R, Lipson, H, Longstaff, T & Mead, N 1999, *Survivability: Protecting Your Critical Systems*, CERT – Carnegie Mellon University, Pennsylvania United States, viewed 26 November 2002, <<http://www.cert.org/archive/html/protect-critical-systems.html>>
- [56] CERT Coordination Centre 2002, Carnegie Mellon University, Pennsylvania United States, viewed 26 November 2002, <<http://www.cert.org/archive/html/analysis-method.html>>.
- [57] Ellison, R, Linger, R, Longstaff, T, Mead, N & McHugh, J 2000, *Survivable Network Analysis Method*, CERT – Carnegie Mellon University, Pennsylvania United States, viewed 27 November 2002, <<http://www.cert.org/archive/pdf/00tr013.pdf>>.
- [58] Lipson, H 2000, *Survivability – A New Security Paradigm for Protecting Highly Distributed Mission-Critical Systems*, CERT – Carnegie Mellon University, Pennsylvania United States, viewed 11 December 2002, <<http://www.cert.org/archive/pdf/surviv-paradigm.pdf>>.
- [59] Jha, S, Wing, J, Linger, R & Longstaff, T 2000, *Survivability Analysis of Network Specifications*, Carnegie Mellon University, Pennsylvania United States, viewed 10 December 2002, <<http://www-2.cs.cmu.edu/afs/cs/project/calder/papers/dsn00/paper.ps>>.

## Economic Models:

- [60] Anderson, R 2001, *Why Information Security is Hard – An Economic Perspective*, University of Cambridge, United Kingdom, viewed 9 December 2002, <<http://www.cl.cam.ac.uk/ftp/users/rja14/econ.pdf>>.
- [61] Odlyzko, A 2000, *Cryptographic Abundance and Pervasive Computing*, University of Minnesota, United States, viewed 9 December 2002, <<http://www.dtc.umn.edu/~odlyzko/doc/crypto.abundance.txt>>.

- [62] Gordon LA & Loeb, MP 2001, *Economic Aspects of Information Security*, Rainbow Technologies, viewed 9 December 2002, <<http://www.rainbow.com/library/8/EconomicsAspectsOfInformationSecurity.pdf>>.
- [63] Von Hippel, E 2002, *Open Source Software Projects as User Innovation Networks*, Conference on Open Source Software 2002, Institut d'Economie Industrielle, France, viewed 9 December 2002, <<http://www.idei.asso.fr/Commun/Conferences/Internet/OSS2002/Papiers/VonHippel.pdf>>.
- [64] Hirshleifer, J 1983, 'From Weakest-Link to Best-Shot: The Voluntary Provision of Public Goods', *Public Choice*, vol. 41, pp. 371-86.

## DISTRIBUTION LIST

### A Survey of Techniques for Security Architecture Analysis

Brendan Lawlor and Linh Vu

#### AUSTRALIA

##### DEFENCE ORGANISATION

###### Task Sponsor (DSD)

Assistant Secretary Information Security  
Assistant Secretary Capability Assurance  
Assistant Secretary Capability Provision

###### S&T Program

Chief Defence Scientist	} shared copy	
FAS Science Policy		
AS Science Corporate Management		
Director General Science Policy Development		
Counsellor Defence Science, London		Doc Data Sheet
Counsellor Defence Science, Washington		Doc Data Sheet
Scientific Adviser to MRDC Thailand		Doc Data Sheet
Scientific Adviser Joint		
Navy Scientific Adviser		Doc Data Sheet & dist list
Scientific Adviser - Army		Doc Data Sheet & dist list
Air Force Scientific Adviser		Doc Data Sheet & dist list
Director Trials		

###### Information Sciences Laboratory

Chief of Information Networks Division (CIND)  
Research Leader Information Assurance Branch (RLIA)  
Head Trusted Computer Systems Group (HTCS)  
Task Manager, Dr. Maris Ozols  
Dr. John Yesberg  
Peter Duffett  
Brendan Lawlor  
Linh Vu

###### DSTO Library and Archives

Library Edinburgh	Doc Data Sheet
Australian Archives	

###### Capability Systems Staff

Director General Maritime Development	Doc Data Sheet
Director General Aerospace Development	Doc Data Sheet
Director General Information Capability Development	

###### Office of the Chief Information Officer

Director General Information Policy and Plans	
Chief Information Officer	Doc Data Sheet

Deputy CIO	Doc Data Sheet
AS Information Structures and Futures	Doc Data Sheet
AS Information Architecture and Management	Doc Data Sheet
Director General Australian Defence Information Office	Doc Data Sheet
Director General Australian Defence Simulation Office	Doc Data Sheet

### **Army**

ABCA National Standardisation Officer, Land Warfare Development Sector,  
Puckapunyal e-mailed Doc Data Sheet

SO (Science), Deployable Joint Force Headquarters (DJFHQ) (L), Enoggera QLD  
Doc Data Sheet

SO (Science) - Land Headquarters (LHQ), Victoria Barracks NSW  
Doc Data Sheet & Executive Summary

### **Air Force**

Staff Officer (Science), RAAF Amberley	Doc Data Sheet
Staff Officer (Science), RAAF Williamtown	Doc Data Sheet

### **Intelligence and Security Program**

DGSTA Defence Intelligence Organisation  
Manager, Information Centre, Defence Intelligence Organisation  
Head, Defence Security Authority

### **Defence Libraries**

Library Manager, DLS-Canberra	Doc Data Sheet
Library Manager, DLS - Sydney West	Doc Data Sheet

### **UNIVERSITIES AND COLLEGES**

Australian Defence Force Academy  
Library  
Head of Aerospace and Mechanical Engineering  
Serials Section (M list), Deakin University Library, Geelong, VIC  
Hargrave Library, Monash University Doc Data Sheet  
Librarian, Flinders University

### **OTHER ORGANISATIONS**

National Library of Australia  
NASA (Canberra)  
State Library of South Australia

### **OUTSIDE AUSTRALIA**

#### **INTERNATIONAL DEFENCE INFORMATION CENTRES**

TTCP C3I TP-11, 4 copies  
US Defense Technical Information Center, 2 copies  
UK Defence Research Information Centre, 2 copies  
Canada Defence Scientific Information Service, 1 copy  
NZ Defence Information Centre, 1 copy

#### **ABSTRACTING AND INFORMATION ORGANISATIONS**

Library, Chemical Abstracts Reference Service

Engineering Societies Library, US  
Materials Information, Cambridge Scientific Abstracts, US  
Documents Librarian, The Center for Research Libraries, US

**INFORMATION EXCHANGE AGREEMENT PARTNERS**

Acquisitions Unit, Science Reference and Information Service, UK

SPARES (5 copies)

**Total number of copies: 47 full copies, 22 Doc Data Sheets, 3 Distribution Lists, 1 Executive Summary**

<b>DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION DOCUMENT CONTROL DATA</b>				1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)	
2. TITLE A Survey of Techniques for Security Architecture Analysis			3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)  Document (U) Title (U) Abstract (U)		
4. AUTHOR(S) Brendan Lawlor and Linh Vu			5. CORPORATE AUTHOR Information Sciences Laboratory PO Box 1500 Edinburgh South Australia 5111 Australia		
6a. DSTO NUMBER DSTO-TR-1438		6b. AR NUMBER AR-012-778	6c. TYPE OF REPORT Technical Report		7. DOCUMENT DATE May 2003
8. FILE NUMBER 9505-25-40	9. TASK NUMBER INT02/251	10. TASK SPONSOR ASINFOSEC	11. NO. OF PAGES 59		12. NO. OF REFERENCES 64
13. URL on the World Wide <a href="http://www.dsto.defence.gov.au/corporate/reports/DSTO-TR-1438.pdf">http://www.dsto.defence.gov.au/corporate/reports/DSTO-TR-1438.pdf</a>			14. RELEASE AUTHORITY Chief, Information Networks Division		
15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT  <i>Approved for public release</i>					
OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE, PO BOX 1500, EDINBURGH, SA 5111					
16. DELIBERATE ANNOUNCEMENT  No Limitations					
17. CITATION IN OTHER DOCUMENTS			Yes		
18. DEFTTEST DESCRIPTORS  Computer network architecture; Network security; Computer security; Vulnerability analysis.					
19. ABSTRACT <b>This technical report is a survey of existing techniques which could potentially be used in the analysis of security architectures. The report has been structured to section the analysis process over three phases: the capture of a specific architecture in a suitable representation, discovering attacks on the captured architecture, and then assessing and comparing different security architectures. Each technique presented in this report has been recognised as being potentially useful for one phase of the analysis.</b>  <b>By presenting a set of potentially useful techniques, it is hoped that designers and decision-makers involved in the development and maintenance of security architectures will be able to develop a more complete, justified and usable methodology other than those currently being used to perform analyses.</b>					